aws

Guida allo sviluppo di Amazon EMR su EKS

Amazon EMR

Amazon EMR: Guida allo sviluppo di Amazon EMR su EKS

Table of Contents

Che cos'è Amazon EMR su EKS?

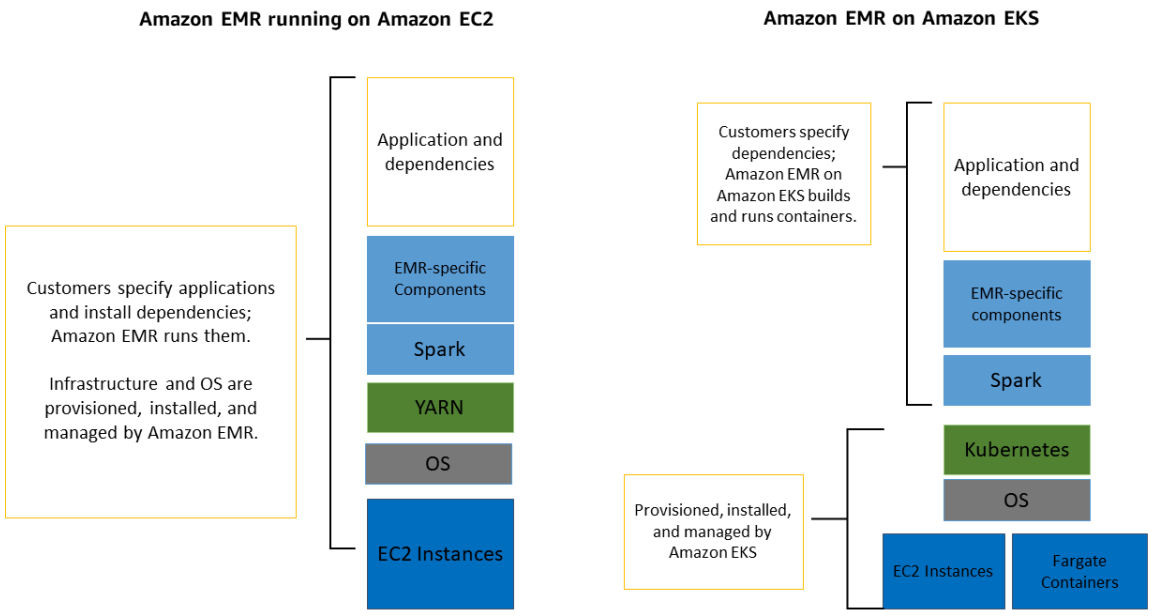
Amazon EMR su EKS offre un'opzione di implementazione per Amazon EMR che consente di eseguire framework per Big Data open source su Amazon Elastic Kubernetes Service (Amazon EKS). Questa opzione di implementazione consente all'utente di concentrarsi sull'esecuzione di carichi di lavoro di analisi dei dati mentre Amazon EMR su EKS crea, configura e gestisce container per applicazioni open source.

Se già si utilizza Amazon EMR, adesso è possibile eseguire applicazioni basate su Amazon EMR con altri tipi di applicazioni nello stesso cluster Amazon EKS. Questa opzione di implementazione migliora inoltre l'utilizzo delle risorse e semplifica la gestione dell'infrastruttura in più zone di disponibilità. Se già si eseguono framework di Big Data su Amazon EKS, adesso è possibile utilizzare Amazon EMR per automatizzare il provisioning e la gestione ed eseguire Apache Spark più rapidamente.

Amazon EMR su EKS consente al proprio team di collaborare in modo più efficiente ed elaborare un'enorme quantità di dati in modo più semplice ed economico:

- È possibile eseguire applicazioni su un pool comune di risorse senza dover eseguire il provisioning dell'infrastruttura. Puoi utilizzare Amazon EMR Studio e l' AWS SDK o AWS CLI per sviluppare, inviare e diagnosticare applicazioni di analisi in esecuzione su cluster EKS. È possibile eseguire processi pianificati su Amazon EMR su EKS utilizzando Apache Airflow o Amazon Managed Workflows for Apache Airflow (MWAA) autogestiti.
- I team dell'infrastruttura possono gestire centralmente una piattaforma di calcolo comune per consolidare i carichi di lavoro Amazon EMR con altre applicazioni basate su container. È possibile semplificare la gestione dell'infrastruttura con i comuni strumenti Amazon EKS e sfruttare un cluster condiviso per carichi di lavoro che richiedono versioni diverse di framework open source. È inoltre possibile ridurre il sovraccarico operativo con la gestione automatizzata dei cluster Kubernetes e l'applicazione di patch del sistema operativo. Con Amazon EC2 and AWS Fargate, puoi abilitare più risorse di elaborazione per soddisfare requisiti prestazionali, operativi o finanziari.

Nel diagramma seguente sono mostrati i due diversi modelli di implementazione per Amazon EMR.



Argomenti

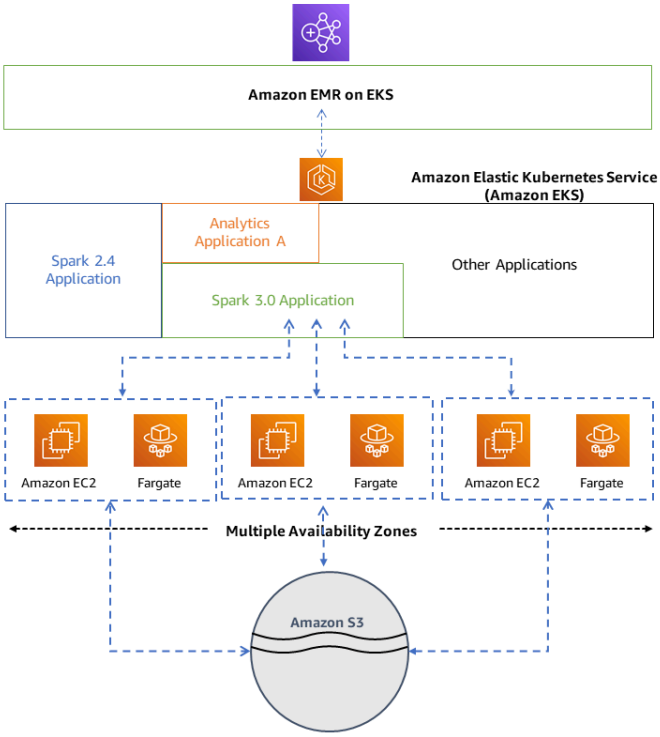
- Architettura per Amazon EMR su EKS
- Comprendere i concetti e la terminologia di Amazon EMR su EKS
- Cosa succede quando invii un lavoro a un cluster virtuale Amazon EMR su EKS?

Architettura per Amazon EMR su EKS

Amazon EMR su EKS associa liberamente le applicazioni all'infrastruttura su cui vengono eseguite. Ogni livello di infrastruttura fornisce l'orchestrazione per il livello successivo. Quando si invia un processo ad Amazon EMR, la definizione di processo contiene tutti i parametri specifici dell'applicazione. Amazon EMR utilizza questi parametri per indicare ad Amazon EKS quali pod e container implementare. Amazon EKS mette quindi online le risorse di elaborazione di Amazon EC2 AWS Fargate necessarie per eseguire il lavoro.

Con questa associazione libera di servizi, è possibile eseguire più processi isolati contemporaneamente in modo sicuro. Inoltre, è possibile eseguire il valore di riferimento dello stesso processo con diversi back-end di calcolo o distribuire il lavoro in più zone di disponibilità per incrementare la disponibilità.

Il diagramma seguente illustra come Amazon EMR su EKS funziona con altri servizi. AWS



Comprendere i concetti e la terminologia di Amazon EMR su EKS

Amazon EMR su EKS offre un'opzione di implementazione per Amazon EMR che consente di eseguire framework per Big Data open source su Amazon Elastic Kubernetes Service (Amazon EKS). Questo argomento fornisce un contesto su alcuni dei termini più diffusi, tra cui namespace, cluster virtuali e job run, che sono unità di lavoro che invii per l'elaborazione.

Spazio dei nomi Kubernetes

Amazon EKS utilizza gli spazi dei nomi Kubernetes per suddividere le risorse del cluster tra più utenti e applicazioni. Questi spazi dei nomi costituiscono la base degli ambienti multi-tenant. Uno spazio dei nomi Kubernetes può avere Amazon EC2 o AWS Fargate come provider di elaborazione. Questa flessibilità offre diverse opzioni di prestazioni e costi per l'esecuzione dei processi.

Cluster virtuale

Un cluster virtuale è uno spazio dei nomi Kubernetes con cui è registrato Amazon EMR. Amazon EMR utilizza cluster virtuali per eseguire processi e ospitare endpoint. Più cluster virtuali possono

essere supportati dallo stesso cluster fisico. Tuttavia, ogni cluster virtuale esegue la mappatura a uno spazio dei nomi in un cluster EKS. I cluster virtuali non creano risorse attive che incrementano i costi in fattura o che richiedono la gestione del ciclo di vita all'esterno del servizio.

Esecuzione del processo

L'esecuzione di un job è un'unità di lavoro, ad esempio un jar Spark, PySpark uno script o una query SparkSQL, che invii ad Amazon EMR su EKS. Un processo può avere più esecuzioni. Quando si invia l'esecuzione di un processo, occorre includere le seguenti informazioni:

- Un cluster virtuale in cui deve essere eseguito il processo.
- Un nome per identificare il processo.
- Ruolo di esecuzione: un ruolo IAM definito che esegue il processo e consente di specificare le risorse a cui è possibile accedere tramite il processo.
- L'etichetta di rilascio di Amazon EMR che specifica la versione delle applicazioni open source da utilizzare.
- Gli artefatti da utilizzare durante l'invio del processo, ad esempio i parametri spark-submit.

Per impostazione predefinita, i log vengono caricati su Spark History Server e sono accessibili da Console di gestione AWS. Puoi anche inviare log di eventi, log di esecuzione e metriche ad Amazon S3 e Amazon. CloudWatch

Container di Amazon EMR

Un container di Amazon EMR è il nome dell'API per Amazon EMR su EKS. Il prefisso `emr-containers` viene utilizzato nei seguenti scenari:

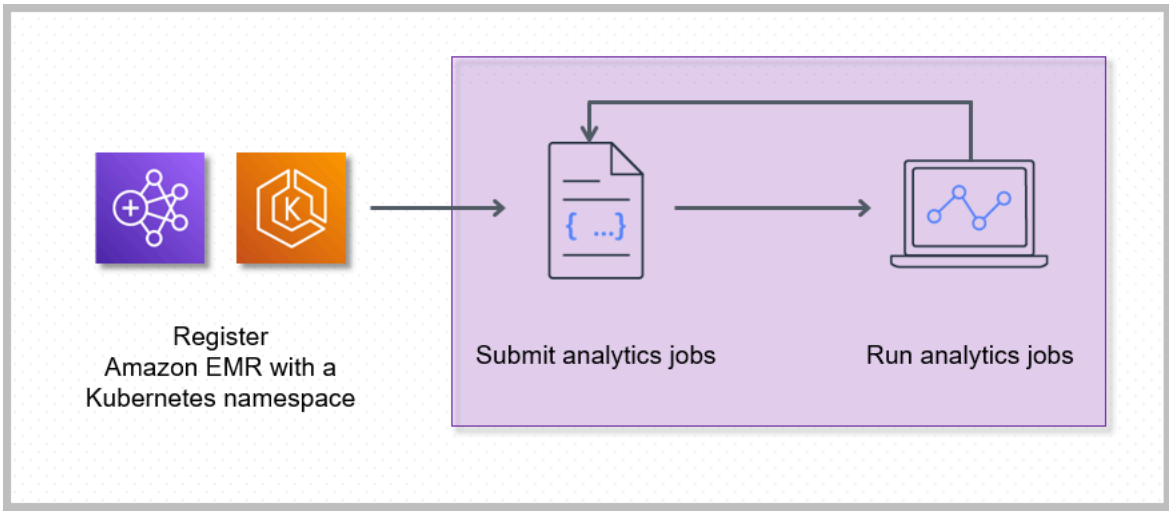
- È il prefisso nei comandi CLI per Amazon EMR su EKS. Ad esempio, `aws emr-containers start-job-run`.
- È il prefisso prima delle operazioni di policy IAM per Amazon EMR su EKS. Ad esempio, `"Action": [ "emr-containers:StartJobRun"]`. Per ulteriori informazioni, consulta Operazioni di policy per Amazon EMR su EKS.
- È il prefisso utilizzato negli endpoint del servizio di Amazon EMR su EKS. Ad esempio, `emr-containers.us-east-1.amazonaws.com`. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Cosa succede quando invii un lavoro a un cluster virtuale Amazon EMR su EKS?

La registrazione di Amazon EMR con uno spazio dei nomi Kubernetes su Amazon EKS crea un cluster virtuale. Amazon EMR può quindi eseguire carichi di lavoro di analisi dei dati in tale spazio dei nomi. Quando si utilizza Amazon EMR su EKS per inviare i processi Spark al cluster virtuale, Amazon EMR su EKS richiede al pianificatore Kubernetes su Amazon EKS di pianificare i pod.

I seguenti passaggi e il seguente diagramma illustrano il flusso di lavoro di Amazon EMR su EKS:

- Utilizza un cluster Amazon EKS esistente o creane uno utilizzando l'utility a riga di comando eksctl o la console di Amazon EKS.
- Crea un cluster virtuale registrando Amazon EMR con uno spazio dei nomi in un cluster EKS.
- Invia il lavoro al cluster virtuale utilizzando l'SDK AWS CLI o.



Per ogni processo eseguito, Amazon EMR su EKS crea un container con un'immagine di base Amazon Linux 2, Apache Spark e le dipendenze associate. Ogni processo viene eseguito in un pod che effettua il download del container e inizia a eseguirlo. Il pod termina dopo la fine del processo. Se l'immagine del container è stata già precedentemente impiegata nel nodo, viene utilizzata un'immagine memorizzata nella cache e il download viene ignorato. I container Sidecar, come ad esempio i server d'inoltro di log e parametri, possono essere implementati sul pod. Una volta terminato il processo, è comunque possibile eseguire il debug utilizzando l'interfaccia utente dell'applicazione Spark nella console di Amazon EMR.

Guida introduttiva ad Amazon EMR su EKS

Questo argomento consente di iniziare a utilizzare Amazon EMR su EKS implementando un'applicazione Spark Python in un cluster virtuale. Include i passaggi per configurare le autorizzazioni corrette e avviare un lavoro. Prima di iniziare, verifica di aver completato le fasi in Configurazione di Amazon EMR su EKS. Questo ti aiuta a ottenere strumenti come la AWS CLI configurazione prima di creare il tuo cluster virtuale. Per altri modelli che possono aiutarti a iniziare, consulta la nostra guida alle best practice per i contenitori EMR su. GitHub

Nella procedura di configurazione sono necessarie le seguenti informazioni:

- ID del cluster virtuale per il cluster Amazon EKS e lo spazio dei nomi Kubernetes registrato con Amazon EMR

  > **Important**
  >
  > Quando crei un cluster EKS, assicurarsi di utilizzare m5.xlarge come tipo di istanza o qualsiasi altro tipo di istanza con CPU e memoria superiori. L'utilizzo di un tipo di istanza con CPU o memoria inferiore a m5.xlarge può causare errori di processo a causa di risorse insufficienti disponibili nel cluster.

- Nome del ruolo IAM utilizzato per l'esecuzione del processo
- Etichetta di rilascio per la versione di Amazon EMR (ad esempio, `emr-6.4.0-latest`)
- I target di destinazione per la registrazione e il monitoraggio:
  - Nome del gruppo di CloudWatch log Amazon e prefisso del flusso di log
  - Ubicazione Amazon S3 per memorizzare i log degli eventi e del container

> **Important**
>
> I job Amazon EMR su EKS utilizzano Amazon CloudWatch e Amazon S3 come destinazioni di destinazione per il monitoraggio e la registrazione. È possibile monitorare l'avanzamento dei processi e risolvere i fallimenti visualizzando i registri dei processi inviati a queste destinazioni. Per abilitare la registrazione, la policy IAM associata al ruolo IAM per l'esecuzione del processo deve disporre delle autorizzazioni necessarie per accedere alle risorse di destinazione. Se la policy IAM non dispone delle autorizzazioni richieste, devi seguire i passaggi descritti in Configurare un job run Aggiornamento della policy di

affidabilità del ruolo di esecuzione di processo for use Amazon S3 logs e Configure a job run to CloudWatch use Logs prima di eseguire questo job di esempio.

Eseguire un'applicazione Spark

Seguire la procedura seguente per eseguire una applicazione semplice Applicazione Spark su Amazon EMR su EKS. Il file `entryPoint` di applicazione per le applicazioni Spark Python si trova in `s3://`*`REGION`*`.elasticmapreduce/emr-containers/samples/wordcount/scripts/wordcount.py`. *`REGION`*È la regione in cui risiede il tuo cluster virtuale Amazon EMR su EKS, ad esempio. *`us-east-1`*

1. Aggiornare la policy IAM per il ruolo di esecuzione del processo con le autorizzazioni richieste, come dimostrano le istruzioni riportate di seguito.

   JSON

```
{
  "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:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket-b",
        "arn:aws:s3:::amzn-s3-demo-bucket-b/*"
      ]
    },
    {
      "Sid": "WriteToLoggingAndOutputDataBuckets",
      "Effect": "Allow",
      "Action": [
```

```
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket-b/*"
      ]
    },
    {
      "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/*"
      ]
    }
  ]
}
```

- La prima istruzione `ReadFromLoggingAndInputScriptBuckets` in questa policy concede `ListBucket` e `GetObjects` l'accesso ai seguenti bucket Amazon S3:
  - *REGION*`.elasticmapreduce`: il bucket in cui il file dell'applicazione `entryPoint` si trova.
  - *amzn-s3-demo-destination-bucket*- un bucket definito dall'utente per i dati di output.
  - *amzn-s3-demo-logging-bucket*- un bucket definito dall'utente per i dati di registrazione.

- La seconda istruzione `WriteToLoggingAndOutputDataBuckets` in questa policy concede al processo le autorizzazioni per scrivere i dati nei bucket di output e per registrarli rispettivamente.
- La terza istruzione `DescribeAndCreateCloudwatchLogStream` concede al job le autorizzazioni per descrivere e creare Amazon CloudWatch Logs.
- La quarta istruzione `WriteToCloudwatchLogs` concede le autorizzazioni per scrivere log su un gruppo di CloudWatch log di Amazon denominato *`my_log_group_name`* sotto un flusso di log denominato. *`my_log_stream_prefix`*

2. Per eseguire un'applicazione Spark Python, utilizzare il comando seguente. Sostituisci tutti i valori sostituibili con *`red italicized`* i valori appropriati. *`REGION`*È la regione in cui risiede il tuo cluster virtuale Amazon EMR su EKS, ad esempio. *`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://amzn-s3-demo-destination-bucket/
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://amzn-s3-demo-logging-bucket"
    }
  }
}'
```

I dati di output di questo processo saranno disponibili a `s3://amzn-s3-demo-destination-bucket/wordcount_output`.

Puoi anche creare un file JSON con i parametri specificati per l'esecuzione del processo. Poi, esegui il comando `start-job-run` con un percorso del file JSON. Per ulteriori informazioni, consulta Invio di un'esecuzione di processo con `StartJobRun`. Per ulteriori dettagli sulla configurazione dei parametri di esecuzione del processo, consulta Opzioni per la configurazione di un'esecuzione di processo.

3. Per eseguire un'applicazione Spark SQL, utilizzare il comando seguente. Sostituisci tutti i *red italicized* valori con i valori appropriati. *REGION*È la regione in cui risiede il tuo cluster virtuale Amazon EMR su EKS, ad esempio. *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://amzn-s3-demo-logging-bucket"
    }
  }
}'
```

Di seguito un esempio di file di query. È necessario disporre di un archivio file esterno, ad esempio S3, in cui sono archiviati i dati per le tabelle.

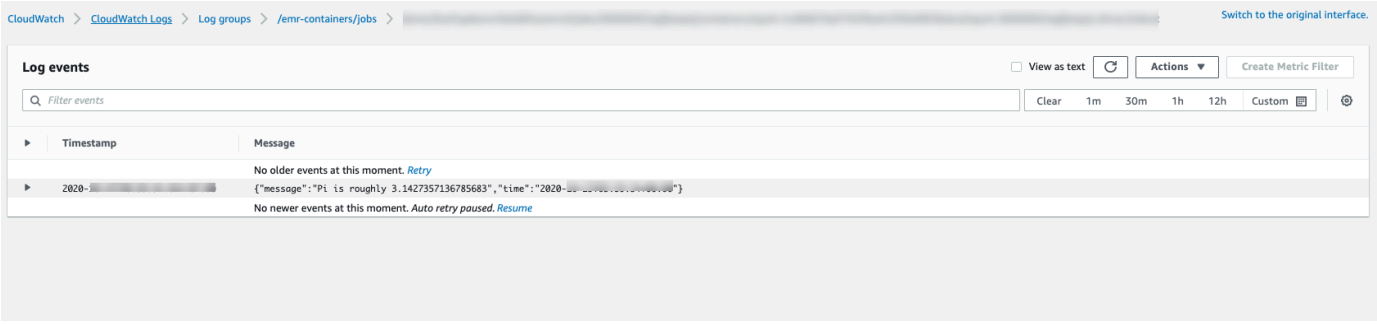
```
CREATE DATABASE demo;
CREATE EXTERNAL TABLE IF NOT EXISTS demo.amazonreview( marketplace string,
 customer_id string, review_id  string, product_id  string, product_parent  string,
 product_title  string, star_rating  integer, helpful_votes  integer, total_votes
  integer, vine  string, verified_purchase  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;
```

L'output di questo processo sarà disponibile nei log stdout del driver in S3 o CloudWatch, a seconda della configurazione. `monitoringConfiguration`

4. Puoi anche creare un file JSON con i parametri specificati per l'esecuzione del processo. Poi, esegui il comando start-job-run con un percorso del file JSON. Per ulteriori informazioni, vedere Invio di un'esecuzione di processo. Per ulteriori informazioni sulla configurazione dei parametri di esecuzione del processo, vedere Opzioni per la configurazione di un'esecuzione di processo.

   Per monitorare lo stato di avanzamento del processo o per eseguire il debug degli errori, puoi controllare i log caricati su Amazon S3, Logs o entrambi. CloudWatch Fai riferimento al percorso di log in Amazon S3 in Configure a job run per usare i log S3 e per i log di Cloudwatch in Configure a job run to use Logs. CloudWatch Per visualizzare i log in Logs, segui le istruzioni riportate di seguito. CloudWatch

   - Apri la CloudWatch console all'indirizzo. https://console.aws.amazon.com/cloudwatch/
   - Nel pannello Navigation (Navigazione), scegli Logs (Log). Quindi, scegli Log groups (Gruppi di log).
   - Scegli il gruppo di log per Amazon EMR su EKS e visualizza gli eventi di log caricati.



Important

I processi hanno una policy di ripetizione configurata predefinita. Per informazioni sulla modifica o la disabilitazione della configurazione, fai riferimento a Utilizzo delle policy di ripetizione dei processi.

Collegamenti ad Amazon EMR su EKS, guide sulle best practice su GitHub

Abbiamo creato la Amazon EMR on EKS Best Practices Guide utilizzando la collaborazione della community open source in modo da poter iterare rapidamente e fornire consigli sugli aspetti della creazione e della gestione di un cluster virtuale. È preferibile utilizzare la Guida sulle procedure consigliate per Amazon EMR su EKS per le sezioni. Scegli i link in ogni sezione per accedere al GitHub sito.

Sicurezza

> **Note**
>
> Per ulteriori informazioni sulla sicurezza con Amazon EMR su EKS, consulta Best practice di sicurezza per Amazon EMR su EKS.

Best practice per la crittografia: come utilizzare la crittografia per dati a riposo e in transito.

Gestione della sicurezza della rete descrive come configurare gruppi di sicurezza per i pod per Amazon EMR su EKS mentre ti connetti a origini dati ospitate in Servizi AWS , ad esempio Amazon RDS e Amazon Redshift.

Utilizzo del gestore AWS dei segreti per archiviare i segreti.

Invio di processi Pyspark

Invio di processi Pyspark: specifica diverse modalità di creazione pacchetti per applicazioni PySpark con formati di pacchetti come zip, egg, wheel e pex.

Storage

Utilizzo di volumi EBS: come utilizzare il provisioning statico e dinamico per i processi che richiedono volumi EBS.

Utilizzo dei volumi Amazon FSx for Lustre: come utilizzare il provisioning statico e dinamico per lavori che richiedono volumi Amazon FSx for Lustre.

Utilizzo di volumi di archivio dell'istanza: come utilizzare i volumi di archivio dell'istanza per l'elaborazione di processi.

Integrazione metastore

Utilizzo del metastore Hive: offre diversi metodi per utilizzare il metastore Hive.

Utilizzo di AWS Glue: offre diversi modi per configurare il catalogo AWS Glue.

Debug

Utilizzo del debug Spark: come modificare il livello di log.

Connessione all'interfaccia utente Spark sul pod driver.

Come utilizzare il server di cronologia Spark in hosting autonomo con Amazon EMR su EKS.

Soluzione dei problemi di Amazon EMR su EKS

Soluzione dei problemi.

Posizionamento dei nodi

Utilizzo dei selettori dei nodi Kubernetes per `single-az` e altri casi d'uso.

Utilizzo del posizionamento dei nodi Fargate.

Prestazioni

Utilizzo dell'allocazione dinamica delle risorse (DRA).

Procedure consigliate per EKS per il plug-in Amazon VPC Container Network Interface (CNI), Cluster Autoscaler e Core DNS.

Ottimizzazione dei costi

Utilizzo delle istanze spot: best practice per le istanze EC2 Spot di Amazon e come utilizzare la funzionalità di decommissionamento dei nodi Spark.

Usando AWS Outposts

Personalizzazione delle immagini Docker per Amazon EMR su EKS

È possibile utilizzare immagini Docker personalizzate con Amazon EMR su EKS. La personalizzazione dell'immagine di runtime di Amazon EMR su EKS fornisce i seguenti vantaggi:

- Confezionare le dipendenze delle applicazioni e l'ambiente di runtime in un unico container immutabile che promuove la portabilità e semplifica la gestione delle dipendenze per ogni carico di lavoro.
- Installare e configurare pacchetti ottimizzati per i carichi di lavoro. Questi pacchetti potrebbero non essere ampiamente disponibili nella distribuzione pubblica dei runtime Amazon EMR.
- Integra Amazon EMR su EKS con i processi correnti di costruzione, test e implementazione all'interno della tua organizzazione, inclusi sviluppo e test locali.
- Applica processi di sicurezza consolidati, quali la scansione delle immagini, che soddisfano i requisiti di conformità e governance all'interno dell'organizzazione.

Argomenti

Come personalizzare le immagini Docker

Segui questi passaggi per personalizzare le immagini Docker per Amazon EMR su EKS. I passaggi mostrano come ottenere un'immagine di base, personalizzarla e pubblicarla e inviare un carico di lavoro utilizzando l'immagine.

> **Note**
>
> Altre opzioni da prendere in considerazione quando si personalizzano le immagini Docker sono la personalizzazione per gli endpoint interattivi, cosa che si fa per assicurarsi di avere le dipendenze richieste, o l'utilizzo di immagini di contenitori multiarchitetturali:
>
> - Personalizzazione delle immagini Docker per endpoint interattivi
> - Lavorare con immagini multi-architettura

Prerequisiti

- Completa le fasi Configurazione di Amazon EMR su EKS per Amazon EMR su EKS.
- Installa Docker nel tuo ambiente. Per ulteriori informazioni, consulta Ottieni Docker.

Fase 1: recupero di un'immagine di base da Amazon Elastic Container Registry (Amazon ECR)

L'immagine di base contiene il runtime Amazon EMR e i connettori utilizzati per accedere ad altri servizi AWS . Per Amazon EMR 6.9.0 e versioni successive, è possibile ottenere le immagini di base da Amazon ECR Public Gallery. Sfoglia la galleria per trovare il collegamento all'immagine e trasferiscila nel tuo Workspace locale. Ad esempio, per la versione Amazon EMR 7.12.0, il `docker pull` comando seguente consente di ottenere l'immagine di base standard più recente. Puoi sostituire `emr-7.12.0:latest` con `emr-7.12.0-spark-rapids:latest` per recuperare l'immagine che dispone dell'acceleratore RAPIDS Nvidia. Puoi anche sostituire `emr-7.12.0:latest` con `emr-7.12.0-java11:latest` per recuperare l'immagine con runtime Java 11.

```
docker pull public.ecr.aws/emr-on-eks/spark/emr-7.12.0:latest
```

Se desideri recuperare l'immagine di base per il rilascio 6.9.0 o precedenti di Amazon EMR o se preferisci recuperarla dagli account di registro Amazon ECR in ogni Regione, completa la procedura seguente:

1. Scegli un URI dell'immagine di base. L'URI dell'immagine presenta questo formato, *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/spark/`*container-image-tag*, come illustrato nell'esempio seguente.

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

Per scegliere un'immagine di base nella tua Regione, consulta Dettagli per la selezione dell'URI di un'immagine di base.

2. Accedi al repository Amazon ECR in cui è memorizzata l'immagine di base. Sostituisci *895885662937* e *us-west-2* con l'account del registro Amazon ECR e la AWS regione che hai selezionato.

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

3. Estrai l'immagine di base nel Workspace locale. *emr-6.6.0:latest*Sostituiscilo con il tag dell'immagine del contenitore che hai selezionato.

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

Fase 2: personalizzazione di un'immagine di base

Segui questi passaggi per personalizzare l'immagine di base che hai estratto da Amazon ECR.

1. Crea un nuovo `Dockerfile` nel Workspace locale.
2. Modifica il `Dockerfile` appena creato e aggiungi i seguenti contenuti. Questo `Dockerfile` usa l'immagine di container estratto da `895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest`.

```
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. Aggiungi comandi in `Dockerfile` per personalizzare l'immagine di base. Ad esempio, aggiungi un comando per installare le librerie Python, come mostrato qui di seguito in `Dockerfile`.

```
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. Dalla stessa directory in cui viene creato `Dockerfile`, esegui il comando seguente per generare l'immagine Docker. Fornisci un nome per l'immagine Docker, ad esempio. *emr6.6_custom*

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

Fase 3: (facoltativo ma consigliato) convalida di un'immagine personalizzata

Si consiglia di testare la compatibilità dell'immagine personalizzata prima di pubblicarla. Puoi utilizzare la CLI di immagine personalizzata di Amazon EMR su EKS per verificare se l'immagine ha le strutture di file richieste e le configurazioni corrette per l'esecuzione su Amazon EMR su EKS.

> **Note**
>
> La CLI di immagine personalizzata di Amazon EMR su EKS non può confermare che l'immagine sia priva di errori. Prestare attenzione quando si rimuovono le dipendenze dalle immagini di base.

Seguire i seguenti fasi per convalidare l'immagine personalizzata.

1. Download e installa la CLI di immagine personalizzata di Amazon EMR su EKS. Per ulteriori informazioni, consulta Guida all'installazione della CLI di immagine personalizzata di Amazon EMR su EKS.
2. Eseguire il comando seguente per testare l'installazione.

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

   Di seguito è illustrato un esempio di output.

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

3. Eseguire il comando seguente per convalidare l'immagine personalizzata.

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

- `-i` specifica l'URI dell'immagine locale che deve essere convalidato. Questo può essere l'URI dell'immagine, qualsiasi nome o tag definito per l'immagine.
- `-r` specifica la versione di rilascio esatta per l'immagine di base, ad esempio `emr-6.6.0-latest`.
- `-t` specifica il tipo di immagine. Se si tratta di un'immagine Spark, inserisci `spark`. Il valore predefinito è `spark`. L'attuale versione CLI per immagine personalizzata di Amazon EMR su EKS supporta solo le immagini runtime Spark.

Se si esegue correttamente il comando e l'immagine personalizzata soddisfa tutte le configurazioni e le strutture di file richieste, l'output restituito visualizza i risultati di tutti i test, come dimostra l'esempio seguente.

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

Se l'immagine personalizzata non soddisfa le configurazioni o le strutture di file richieste, si verificano messaggi di errore. L'output restituito fornisce informazioni sulle configurazioni o sulle strutture di file errate.

Fase 4: Pubblicazione di un'immagine personalizzata

Pubblica la nuova immagine Docker nel registro Amazon ECR.

1. Esegui il comando seguente per creare un repository Amazon ECR per archiviare l'immagine Docker. Fornisci un nome per il tuo repository, ad esempio,. *emr6.6_custom_repo* Sostituisci *us-west-2* con la tua regione.

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

   Per ulteriori informazioni, consulta Creazione di un repository nella Guida per l'utente di Amazon ECR.

2. Esegui il comando seguente per autenticarti nel registro di default.

   ```
   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
   ```

   Per ulteriori informazioni, consulta Autenticazione nel registro di default nella Guida per l'utente di Amazon ECR.

3. Tagga e pubblica un'immagine nel repository Amazon ECR creato.

   Tagga l'immagine.

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

   Invia l'immagine.

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

   Per ulteriori informazioni, consulta Invio di un'immagine ad Amazon ECR nella Guida per l'utente di Amazon ECR.

Fase 5: Invio di un carico di lavoro Spark in Amazon EMR utilizzando un'immagine personalizzata

Dopo aver creato e pubblicato un'immagine personalizzata, puoi inviare un processo Amazon EMR su EKS utilizzando un'immagine personalizzata.

Innanzitutto, crea un start-job-run-request file.json e specifica il `spark.kubernetes.container.image` parametro per fare riferimento all'immagine personalizzata, come dimostra il seguente file JSON di esempio.

> **Note**
>
> Puoi utilizzare lo schema `local://` per fare riferimento ai file disponibili nell'immagine personalizzata come mostrato con l'argomento `entryPoint` nel frammento JSON riportato di seguito. È possibile utilizzare anche la schema `local://` per fare riferimento alle dipendenze dell'applicazione. Tutti i file e le dipendenze a cui si fa riferimento utilizzando lo schema `local://` devono essere già presenti nel percorso specificato nell'immagine personalizzata.

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

Puoi fare riferimento all'immagine personalizzata con le proprietà `applicationConfiguration` come illustrato nell'esempio seguente.

```
{
    "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"
                }
            }
        ]
    }
}
```

Successivamente, esegui il comando `start-job-run` per inviare il processo.

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

Negli esempi JSON precedenti, sostituiscilo *`emr-6.6.0-latest`* con la tua versione di rilascio di Amazon EMR. Consigliamo vivamente di utilizzare la versione `-latest` per garantire che la versione selezionata contenga gli aggiornamenti di sicurezza più recenti. Per ulteriori informazioni sulle versioni di Amazon EMR e sui relativi tag di immagine, consulta Dettagli per la selezione dell'URI di un'immagine di base.

> **Note**
>
> È possibile utilizzare `spark.kubernetes.driver.container.image` e `spark.kubernetes.executor.container.image` per specificare un'immagine diversa per i pod di driver ed executor.

Personalizzazione delle immagini Docker per endpoint interattivi

È anche possibile personalizzare le immagini Docker per gli endpoint interattivi in modo da poter eseguire immagini kernel di base personalizzate. In questo modo puoi garantire di disporre delle dipendenze necessarie quando esegui carichi di lavoro interattivi da EMR Studio.

1. Segui le fasi da 1 a 4 descritte in precedenza per personalizzare un'immagine Docker. Per i rilasci 6.9.0 e successivi di Amazon EMR, puoi ottenere le immagini di base da Amazon ECR Public Gallery. Per le versioni precedenti ad Amazon EMR 6.9.0, puoi ottenere l'immagine negli account di Amazon ECR Registry in ciascuna Regione AWS e l'unica differenza è l'URI dell'immagine di base nel tuo Dockerfile. L'URI dell'immagine di base segue il formato:

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

   Nell'URI dell'immagine di base è necessario utilizzare `notebook-spark`, anziché `spark`. L'immagine di base contiene il runtime Spark e i kernel del notebook che vengono eseguiti con esso. Per ulteriori informazioni sulla selezione delle Regioni e dei tag di immagine di container, consulta Dettagli per la selezione dell'URI di un'immagine di base.

   > **Note**
   >
   > Attualmente sono supportate solo le sostituzioni delle immagini di base e non è supportata l'introduzione di kernel completamente nuovi di tipo diverso da quello fornito dalle immagini AWS di base.

2. Crea un endpoint interattivo che si possa utilizzare con l'immagine personalizzata.

   Innanzitutto, crea un file JSON denominato `custom-image-managed-endpoint.json` con i seguenti contenuti.

```
{
    "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"
                        }
                    }
                ]
            }
        ]
    }
}
```

Successivamente, crea un endpoint interattivo utilizzando le configurazioni specificate nel file JSON, come dimostrato nell'esempio seguente.

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

Per ulteriori informazioni, consulta Creazione di un endpoint interattivo per il cluster virtuale.

3. Connettiti all'endpoint interattivo attraverso EMR Studio. Per ulteriori informazioni, consulta Connessione da Studio.

Lavorare con immagini multi-architettura

Amazon EMR su EKS supporta le immagini di container multi-architettura per Amazon Elastic Container Registry (Amazon ECR). Per ulteriori informazioni, consulta Presentazione delle immagini di container multi-architettura per Amazon ECR.

Le immagini personalizzate di Amazon EMR su EKS supportano sia istanze basate su AWS Graviton EC2 che istanze. non-Graviton-based EC2 Le immagini basate su Graviton sono archiviate negli stessi archivi di immagini in Amazon ECR delle immagini. non-Graviton-based

Ad esempio, per ispezionare l'elenco manifest Docker per le immagini 6.6.0, esegui il comando seguente.

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

Ecco l'output. L'architettura `arm64` è per un'istanza Graviton. Quella `amd64` è per un'istanza non Graviton.

```
{
   "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"
         }
      }
   ]
}
```

Completa la procedura seguente per creare immagini multi-architettura:

1. Crea un `Dockerfile` con i seguenti contenuti in modo da poter estrarre l'immagine `arm64`.

   ```
   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. Segui le istruzioni riportate in Presentazione delle immagini di container multi-architettura per Amazon ECR per creare un'immagine multi-architettura.

   > **Note**
   >
   > Devi creare immagini `arm64` su istanze `arm64`. Analogamente, devi creare immagini `amd64` su istanze `amd64`.

   Puoi anche creare immagini multi-architettura senza basarti su ogni tipo di istanza specifico utilizzando il comando Docker `buildx`. Per ulteriori informazioni, consulta Uso del supporto per l'architettura multi-CPU.

3. Dopo aver creato l'immagine multi-architettura, puoi inviare un processo con lo stesso parametro `spark.kubernetes.container.image` e indirizzarlo verso l'immagine. In un cluster eterogeneo composto sia da istanze basate su AWS Graviton che da non-Graviton-based EC2 istanze, l'istanza determina l'immagine dell'architettura corretta in base all'architettura dell'istanza che estrae l'immagine.

Dettagli per la selezione dell'URI di un'immagine di base

> **Note**
>
> Per i rilasci 6.9.0 e successivi di Amazon EMR, puoi recuperare l'immagine di base da Amazon ECR Public Gallery, quindi non è necessario creare l'URI dell'immagine di base come indicato nelle istruzioni in questa pagina. Per trovare il tag dell'immagine di container per la tua immagine di base, consulta la pagina delle note di rilascio della versione corrispondente di Amazon EMR su EKS.

Le immagini Docker di base che puoi selezionare sono archiviate in Amazon Elastic Container Registry (Amazon ECR). L'URI dell'immagine presenta il formato *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/spark/`*container-image-tag*, come illustrato nell'esempio seguente.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.12.0:latest
```

L'URI dell'immagine per gli endpoint interattivi presenta il formato *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/notebook-spark/`*container-image-tag*, come illustrato nell'esempio seguente. Nell'URI dell'immagine di base è necessario utilizzare `notebook-spark`, anziché `spark`.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-spark/emr-7.12.0:latest
```

Analogamente, per immagini `python3` non Spark per gli endpoint interattivi, l'URI dell'immagine è *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/notebook-python/`*container-image-tag*. Il seguente URI di esempio presenta la formattazione corretta:

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-python/emr-7.12.0:latest
```

Per trovare il tag dell'immagine di container per la tua immagine di base, consulta la pagina delle note di rilascio della versione corrispondente di Amazon EMR su EKS.

Account di registro Amazon ECR per Regione

Per evitare un'elevata latenza di rete, estraete un'immagine di base dalla vostra immagine più vicina Regione AWS. Seleziona l'account di registro Amazon ECR che corrisponde alla Regione da cui stai estraendo l'immagine in base alla tabella seguente.

Regioni	Account del registro Amazon ECR	
ap-east-1	736135916053	
ap-northeast-1	059004520145	
ap-northeast-2	996579266876	
ap-northeast-3	705689932349	
ap-southeast-3	946962994502	
ap-south-1	235914868574	
ap-south-2	691480105545	
ap-southeast-1	671219180197	
ap-southeast-2	038297999601	
ca-central-1	351826393999	
eu-central-1	107292555468	
eu-central-2	314408114945	
eu-north-1	830386416364	
eu-west-1	483788554619	
eu-west-2	118780647275	
eu-west-3	307523725174	

Regioni	Account del registro Amazon ECR	
eu-south-1	238014973495	
eu-south-2	350796622945	
il-central-1	395734710648	
me-south-1	008085056818	
me-central-1	818935616732	
sa-east-1	052806832358	
us-gov-west-1	299385240661	
us-gov-east-1	299393998622	
us-east-1	755674844232	
us-east-2	711395599931	
us-west-1	608033475327	
us-west-2	895885662937	
af-south-1	358491847878	
cn-north-1	068337069695	
cn-northwest-1	068420816659	

Considerazioni sulla personalizzazione delle immagini

Quando personalizzi le immagini Docker, puoi scegliere il runtime esatto per il tuo processo a livello granulare. Prendi in considerazione queste best practice quando utilizzi questa funzionalità. Queste includono considerazioni sulla sicurezza, sulla configurazione e sul montaggio di un'immagine:

- La sicurezza è una responsabilità condivisa tra te AWS e te. Sei responsabile dell'applicazione di patch di sicurezza dei dati binari che aggiungi all'immagine. Attieniti a Best practice di sicurezza per Amazon EMR su EKS, in particolare Ricevi gli ultimi aggiornamenti di sicurezza per le immagini personalizzate e Applicazione del principio del privilegio minimo.
- Quando personalizzi un'immagine di base, devi modificare l'utente Docker in `hadoop:hadoop` per garantire che i processi non vengano eseguiti con l'utente root.
- Amazon EMR su EKS monta i file sulle configurazioni dell'immagine, come ad esempio `spark-defaults.conf`, durante il runtime. Per ignorare questi file di configurazione, consigliamo di utilizzare il parametro `applicationOverrides` durante l'invio del processo e di non modificare i file nell'immagine personalizzata.
- Amazon EMR su EKS monta determinate cartelle durante il runtime. Le eventuali modifiche che apporti a tali cartelle non sono disponibili nel container. Se desideri aggiungere un'applicazione o le relative dipendenze per le immagini personalizzate, consigliamo di scegliere una directory che non faccia parte dei percorsi predefiniti seguenti:
  - `/var/log/fluentd`
  - `/var/log/spark/user`
  - `/var/log/spark/apps`
  - `/mnt`
  - `/tmp`
  - `/home/hadoop`
- Puoi caricare l'immagine personalizzata su qualsiasi repository compatibile con Docker, ad esempio Amazon ECR, Docker Hub o un repository aziendale privato. Per ulteriori informazioni su come configurare l'autenticazione del cluster Amazon EKS con il repository Docker selezionato, consulta Estrazione di un'immagine da un registro privato.

Esecuzione di processi Flink con Amazon EMR su EKS

I rilasci 6.13.0 e successivi di Amazon EMR supportano Amazon EMR su EKS con Apache Flink, ovvero l'operatore Flink Kubernetes, come modello di invio dei processi per Amazon EMR su EKS. Con Amazon EMR su EKS con Apache Flink, puoi implementare e gestire applicazioni Flink con il runtime di rilascio Amazon EMR sui tuoi cluster Amazon EKS personali. Dopo aver implementato l'operatore Flink Kubernetes nel cluster Amazon EKS, puoi inviare direttamente le applicazioni Flink con l'operatore. L'operatore gestisce il ciclo di vita delle applicazioni Flink.

Argomenti

Configurazione e utilizzo dell'operatore Flink Kubernetes

Le pagine seguenti descrivono come configurare e utilizzare l'operatore Flink Kubernetes per l'esecuzione di processi Flink con Amazon EMR su EKS. Gli argomenti disponibili includono i prerequisiti richiesti, come configurare l'ambiente e l'esecuzione di un'applicazione Flink su Amazon EMR su EKS.

Argomenti

Configurazione dell'operatore Flink Kubernetes per Amazon EMR su EKS

Completa le seguenti attività per eseguire la configurazione prima di installare l'operatore Flink Kubernetes su Amazon EKS. Se hai già effettuato la registrazione ad Amazon Web Services (AWS) e hai utilizzato Amazon EKS, ti manca poco per cominciare a utilizzare Amazon EMR su EKS. Completa le seguenti attività per eseguire la configurazione per l'operatore Flink su Amazon EKS. Se hai già completato uno dei prerequisiti, puoi saltarli e passare a quello successivo.

- Installa o aggiorna alla versione più recente di AWS CLI: se hai già installato il AWS CLI, conferma di disporre della versione più recente.
- Configura kubectl ed eksctl: eksctl è uno strumento da riga di comando che usi per comunicare con Amazon EKS.
- Installa Helm: il programma di gestione del pacchetto Helm per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes.
- Inizia a usare Amazon EKS — eksctl — Segui i passaggi per creare un nuovo cluster Kubernetes con nodi in Amazon EKS.
- Scegli un'etichetta di release di Amazon EMR (versione 6.13.0 o successiva): l'operatore Flink Kubernetes è supportato con le versioni 6.13.0 e successive di Amazon EMR.
- Abilita i ruoli IAM per gli account di servizio (IRSA) sul cluster Amazon EKS.
- Crea un ruolo di esecuzione di processo.
- Aggiorna la policy di affidabilità del ruolo di esecuzione di processo .
- Crea un ruolo di esecuzione dell'operatore. Questa fase è facoltativa. Puoi utilizzare lo stesso ruolo per i processi e l'operatore Flink. Se desideri utilizzare un ruolo IAM diverso per l'operatore, puoi creare un ruolo separato.
- Aggiorna la policy di affidabilità del ruolo di esecuzione dell'operatore. Devi aggiungere esplicitamente una voce di policy di affidabilità per i ruoli che desideri utilizzare per l'account di servizio dell'operatore Flink Kubernetes di Amazon EMR. Puoi seguire questo formato di esempio:

  JSON

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

```
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "aws:userid": "system:serviceaccount:emr:emr-containers-sa-flink-
operator"
        }
      },
      "Sid": "AllowSTSAssumerolewithwebidentity"
    }
  ]
}
```

Installazione dell'operatore Flink Kubernetes per Amazon EMR su EKS

Questo argomento ti aiuta a iniziare a utilizzare l'operatore Flink Kubernetes su Amazon EKS preparando una distribuzione Flink.

Installa l'operatore Kubernetes

Utilizza la procedura seguente per installare l'operatore Kubernetes per Apache Flink.

1. Se non lo hai già fatto, completa le fasi in the section called “Configurazione”.
2. Installa *`cert-manager`* (una volta per cluster Amazon EKS) per abilitare l'aggiunta del componente webhook.

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

3. Installa il grafico Helm.

```
export VERSION=7.12.0 # The Amazon EMR release version
export NAMESPACE=The Kubernetes namespace to deploy the operator

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

Output di esempio:

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

4. Attendi il completamento dell'implementazione e verifica l'installazione del grafico.

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

5. Una volta completata l'implementazione, dovrebbe apparire il messaggio seguente.

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

6. Utilizza il comando seguente per visualizzare l'operatore implementato.

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

Di seguito viene mostrato un esempio di output, in cui la versione dell'app `x.y.z-amzn-n` corrisponde alla versione dell'operatore Flink per la versione di Amazon EMR su EKS. Per ulteriori informazioni, consulta Rilasci supportati per Amazon EMR su EKS con Apache Flink.

```
NAME                            NAMESPACE    REVISION    UPDATED
               STATUS      CHART                                    APP VERSION

flink-kubernetes-operator    $NAMESPACE   1           2023-02-22 16:43:45.24148
 -0500 EST    deployed    flink-kubernetes-operator-emr-7.12.0    x.y.z-amzn-n
```

Aggiorna l'operatore Kubernetes

Per aggiornare la versione dell'operatore Flink, procedi nel seguente modo:

1. Disinstalla il vecchio`flink-kubernetes-operator`:`helm uninstall flink-kubernetes-operator -n <NAMESPACE>`.

2. Elimina CRD (poiché helm non eliminerà automaticamente il vecchio CRD):. `kubectl delete crd flinkdeployments.flink.apache.org flinksessionjobs.flink.apache.org`
3. Reinstalla `flink-kubernetes-operator` con la versione più recente.

Esecuzione di un'applicazione Flink

Con Amazon EMR 6.13.0 e rilasci successivi, puoi eseguire un'applicazione Flink con l'operatore Flink Kubernetes in modalità Applicazione su Amazon EMR su EKS. Con Amazon EMR 6.15.0 e rilasci successivi, puoi anche eseguire un'applicazione Flink in modalità Sessione. Questa pagina descrive entrambi i metodi che puoi utilizzare per eseguire un'applicazione Flink con Amazon EMR su EKS.

> **Note**
>
> Devi disporre di un bucket Amazon S3 creato per archiviare i metadati ad alta disponibilità del processo quando invii il processo Flink. Se non desideri utilizzare questa funzionalità, puoi disattivarla. È abilitata per impostazione predefinita.

Prerequisito: per poter eseguire un'applicazione Flink con l'operatore Flink Kubernetes, completa le fasi indicate in the section called “Configurazione” e the section called “Installa l'operatore Kubernetes”.

Application mode

Con Amazon EMR 6.13.0 e rilasci successivi, puoi eseguire un'applicazione Flink con l'operatore Flink Kubernetes in modalità Applicazione su Amazon EMR su EKS.

1. Crea un file di `FlinkDeployment` definizione `basic-example-app-cluster.yaml` come nell'esempio seguente. Se hai attivato e utilizzi uno degli opt-in Regioni AWS, assicurati di decommentare e configurare la configurazione. `fs.s3a.endpoint.region`

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example-app-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
```

```
    #fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
    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. Invia l'implementazione Flink con il comando seguente. L'operazione creerà anche un oggetto `FlinkDeployment` denominato `basic-example-app-cluster`.

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

3. Accedi all'interfaccia utente Flink.

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

4. Apri `localhost:8081` per visualizzare localmente i tuoi processi Flink.
5. Ripulisci il processo. Ricordati di ripulire gli artefatti di S3 che sono stati creati per questo lavoro, come i metadati di checkpoint, l'alta disponibilità, i savepointing e i log. CloudWatch

Per ulteriori informazioni sull'invio di applicazioni a Flink tramite l'operatore Flink Kubernetes, consulta Esempi di operatori Flink Kubernetes nella cartella on. `apache/flink-kubernetes-operator` GitHub

Session mode

Con Amazon EMR 6.15.0 e rilasci successivi, puoi eseguire un'applicazione Flink con l'operatore Flink Kubernetes in modalità Sessione su Amazon EMR su EKS.

1. Create un file di definizione denominato come nell'esempio seguente. `FlinkDeployment` `basic-example-app-cluster.yaml` Se hai attivato e utilizzi uno degli opt-in Regioni AWS, assicurati di decommentare e configurare la configurazione. `fs.s3a.endpoint.region`

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example-session-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    #fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
    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. Invia l'implementazione Flink con il comando seguente. L'operazione creerà anche un oggetto `FlinkDeployment` denominato `basic-example-session-cluster`.

```
kubectl create -f basic-example-app-cluster.yaml -n NAMESPACE
```

3. Usa il seguente comando per confermare che il cluster di sessione `LIFECYCLE` è `STABLE`:

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

L'output visualizzato dovrebbe essere come il seguente esempio:

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

4. Crea un file di risorse di definizione personalizzato `FlinkSessionJob` `basic-session-job.yaml` con il seguente contenuto di esempio:

```
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. Invia il processo della sessione Flink con il comando seguente. L'operazione creerà un oggetto `FlinkSessionJob` denominato `basic-session-job`.

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

6. Utilizza il comando seguente per confermare che il cluster di sessione `LIFECYCLE` è `STABLE` e `JOB STATUS` è `RUNNING`:

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

   L'output visualizzato dovrebbe essere come il seguente esempio:

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

7. Accedi all'interfaccia utente Flink.

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

8. Apri `localhost:8081` per visualizzare localmente i tuoi processi Flink.
9. Ripulisci il processo. Ricordati di ripulire gli artefatti di S3 che sono stati creati per questo lavoro, come i metadati di checkpoint, l'alta disponibilità, i savepointing e i log. CloudWatch

Autorizzazioni dei ruoli di sicurezza per l'esecuzione di un'applicazione Flink

Questo argomento descrive i ruoli di sicurezza per la distribuzione e l'esecuzione di un'applicazione Flink. Sono necessari due ruoli per gestire una distribuzione e creare e gestire i lavori, il ruolo di operatore e il ruolo professionale. Questo argomento li presenta e ne elenca le autorizzazioni.

Controllo degli accessi basato sui ruoli

Per implementare l'operatore ed eseguire i processi Flink, dobbiamo creare due ruoli Kubernetes: un ruolo di operatore e un ruolo di processo. Amazon EMR crea i due ruoli per impostazione predefinita durante l'installazione dell'operatore.

Ruolo di operatore

Utilizziamo il ruolo di operatore `flinkdeployments` per gestire e creare JobManager per ogni lavoro Flink e altre risorse, come i servizi.

Il nome predefinito del ruolo di operatore è `emr-containers-sa-flink-operator` e richiede le seguenti autorizzazioni.

```
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:
  - '*'
```

Ruolo di processo

JobManager Utilizza il ruolo professionale per creare e gestire TaskManagers e ConfigMaps per ogni lavoro.

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

Disinstallazione dell'operatore Flink Kubernetes per Amazon EMR su EKS

Segui questa procedura per disinstallare l'operatore Flink Kubernetes.

1. Elimina l'operatore.

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

2. Elimina le risorse Kubernetes che Helm non disinstalla.

```
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. (Facoltativo) Elimina il cert-manager.

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

Utilizzo di Flink Native Kubernetes

I rilasci 6.13.0 e successivi di Amazon EMR supportano Flink Native Kubernetes come strumento di riga di comando che puoi utilizzare per inviare ed eseguire applicazioni Flink in un cluster Amazon EMR su EKS.

Argomenti

- Configurazione di Flink Native Kubernetes per Amazon EMR su EKS
- Nozioni di base su Flink Native Kubernetes per Amazon EMR su EKS
- Requisiti di sicurezza degli account JobManager di servizio Flink per Native Kubernetes

Configurazione di Flink Native Kubernetes per Amazon EMR su EKS

Completa le seguenti attività per definire la configurazione prima di poter eseguire un'applicazione con la CLI di Flink in Amazon EMR su EKS. Se hai già effettuato la registrazione ad Amazon Web Services (AWS) e hai utilizzato Amazon EKS, ti manca poco per cominciare a utilizzare Amazon EMR su EKS. Se hai già completato uno dei prerequisiti, puoi saltarli e passare a quello successivo.

- Installa o aggiorna alla versione più recente di AWS CLI: se hai già installato il AWS CLI, conferma di disporre della versione più recente.
- Inizia a usare Amazon EKS — eksctl — Segui i passaggi per creare un nuovo cluster Kubernetes con nodi in Amazon EKS.
- Seleziona un URI dell'immagine di base Amazon EMR (rilascio 6.13.0 o successivo): il comando Flink Kubernetes è supportato con i rilasci 6.13.0 e successivi di Amazon EMR.

- Verifica che l'account del JobManager servizio disponga delle autorizzazioni appropriate per creare e guardare i pod. TaskManager Per ulteriori informazioni, consulta Requisiti di sicurezza degli account di JobManager servizio Flink per Native Kubernetes.
- Configura il tuo profilo di credenziali AWS locale.
- Crea o aggiorna un file kubeconfig per un cluster Amazon EKS su cui desideri eseguire le applicazioni Flink.

Nozioni di base su Flink Native Kubernetes per Amazon EMR su EKS

Questi passaggi mostrano come configurare, configurare un account di servizio ed eseguire un'applicazione Flink. Flink Native Kubernetes viene utilizzato per distribuire Flink su un cluster Kubernetes in esecuzione.

Configura ed esegui un'applicazione Flink

Amazon EMR 6.13.0 e versioni successive supportano Flink Native Kubernetes per l'esecuzione di applicazioni Flink su un cluster Amazon EKS. Per eseguire un'applicazione Flink, completa questa procedura:

1. Prima di poter eseguire un'applicazione Flink con il comando Flink Native Kubernetes, completa le fasi indicate in the section called “Configurazione”.
2. Scarica e installa Flink.
3. Imposta i valori delle seguenti variabili di ambiente.

```
#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-Regione AWS-.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. Crea un account di servizio per gestire le risorse Kubernetes.

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

5. Esegui il comando della CLI `run-application`.

```
$FLINK_HOME/bin/flink run-application \
    --target kubernetes-application \
    -Dkubernetes.namespace=$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
```

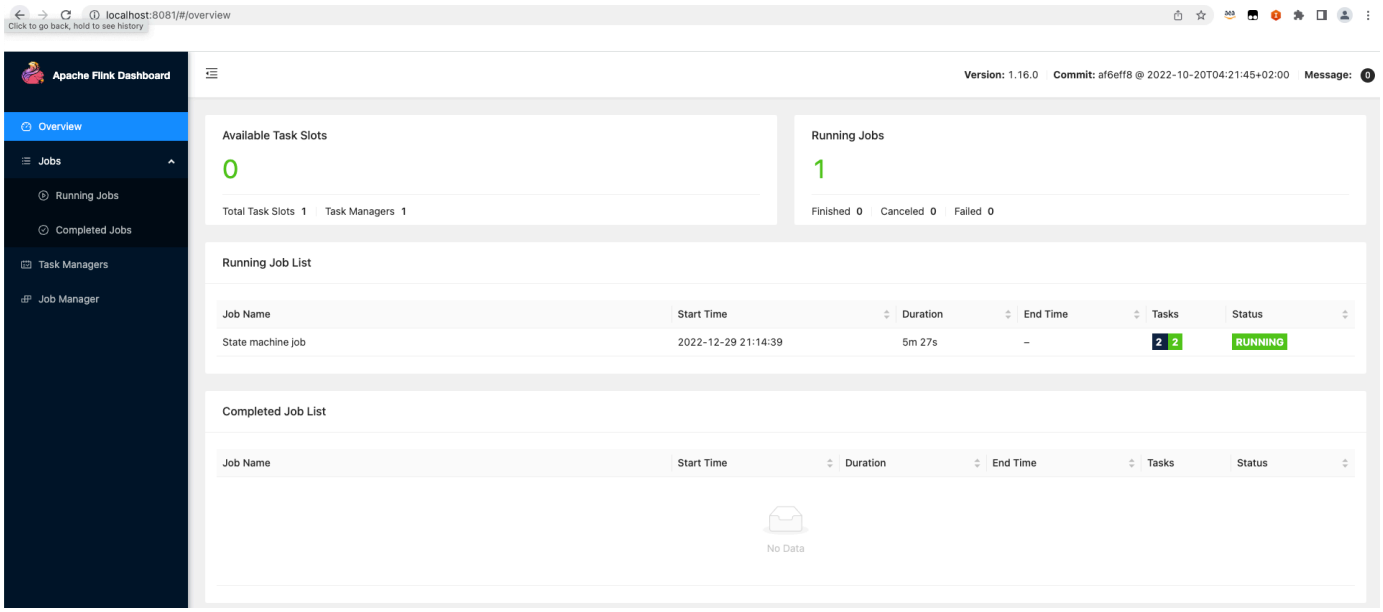
6. Esamina le risorse Kubernetes create.

```
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. Esegui l'inoltro della porta a 8081.

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

8. Accedi localmente all'interfaccia utente Flink.



9. Elimina l'applicazione Flink.

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

Per ulteriori informazioni sull'invio di applicazioni a Flink, consulta Native Kubernetes nella documentazione di Apache Flink.

Requisiti di sicurezza degli account JobManager di servizio Flink per Native Kubernetes

Il JobManager pod Flink utilizza un account di servizio Kubernetes per accedere al server API Kubernetes per creare e guardare i pod. TaskManager L'account del JobManager servizio deve disporre delle autorizzazioni appropriate per i create/delete TaskManager pod e consentire al responsabile ConfigMaps di to watch di TaskManager recuperare l'indirizzo del e nel cluster. JobManager ResourceManager

Per questo account di servizio vigono le regole elencate di seguito.

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

Personalizzazione delle immagini Docker per Flink e FluentD

Segui i passaggi seguenti per personalizzare le immagini Docker per Amazon EMR su EKS con immagini Apache Flink o FluentD. Questi includono indicazioni tecniche per ottenere un'immagine di base, personalizzarla, pubblicarla e inviare un carico di lavoro.

Argomenti

Prerequisiti

Prima di personalizzare l'immagine Docker, assicurati di aver soddisfatto i seguenti prerequisiti:

- Completata la procedura di configurazione dell'operatore Flink Kubernetes per Amazon EMR su EKS.
- Hai installato Docker nel tuo ambiente. Per ulteriori informazioni, consulta Ottieni Docker.

Fase 1: recuperare un'immagine di base da Amazon Elastic Container Registry

L'immagine di base contiene il runtime di Amazon EMR e i connettori necessari per accedere ad altri. Servizi AWS Se utilizzi Amazon EMR su EKS con Flink versione 6.14.0 o successiva, puoi ottenere le immagini di base dalla Galleria pubblica di Amazon ECR. Sfoglia la galleria per trovare il collegamento all'immagine e trasferiscila nel tuo Workspace locale. Ad esempio, per la versione Amazon EMR 6.14.0, il `docker pull` comando seguente restituisce l'immagine di base standard più recente. `emr-6.14.0:latest`Sostituiscila con la versione di rilascio desiderata.

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

Di seguito sono riportati i collegamenti all'immagine della galleria Flink e all'immagine della galleria Fluentd:

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

Fase 2: personalizzazione di un'immagine di base

I passaggi seguenti descrivono come personalizzare l'immagine di base estratta da Amazon ECR.

1. Crea un nuovo `Dockerfile` nel Workspace locale.
2. Modifica `Dockerfile` e aggiungi il seguente contenuto. Questo `Dockerfile` utilizza l'immagine del contenitore da `public.ecr.aws/emr-on-eks/flink/emr-7.12.0-flink:latest` cui hai estratto.

   ```
   FROM public.ecr.aws/emr-on-eks/flink/emr-7.12.0-flink:latest
   ```

```
USER root
### Add customization commands here ####
USER hadoop:hadoop
```

Usa la seguente configurazione se stai usando`Fluentd`.

```
FROM public.ecr.aws/emr-on-eks/fluentd/emr-7.12.0:latest
USER root
### Add customization commands here ####
USER hadoop:hadoop
```

3. Aggiungi comandi in `Dockerfile` per personalizzare l'immagine di base. Il comando seguente mostra come installare le librerie Python.

```
FROM public.ecr.aws/emr-on-eks/flink/emr-7.12.0-flink:latest
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. Nella stessa directory in cui hai creato`DockerFile`, esegui il comando seguente per creare l'immagine Docker. Il campo che fornisci dopo la `-t` bandiera è il tuo nome personalizzato per l'immagine.

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

Passaggio 3: Pubblica l'immagine personalizzata

Ora puoi pubblicare la nuova immagine Docker nel tuo registro Amazon ECR.

1. Esegui il seguente comando per creare un repository Amazon ECR per archiviare la tua immagine Docker. Fornisci un nome per il tuo repository, ad esempio `emr_custom_repo.` Per ulteriori informazioni, consulta Create a repository nella Amazon Elastic Container Registry User Guide.

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

2. Esegui il comando seguente per autenticarti nel registro di default. Per ulteriori informazioni, consulta Authenticate to your default registry nella 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. Invia l'immagine. Per ulteriori informazioni, consulta Push an image to Amazon ECR nella Amazon Elastic Container Registry User Guide.

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

Fase 4: invia un carico di lavoro Flink in Amazon EMR utilizzando un'immagine personalizzata

Apporta le seguenti modifiche alle tue `FlinkDeployment` specifiche per utilizzare un'immagine personalizzata. A tale scopo, inserisci la tua immagine nella `spec.image` riga delle specifiche di distribuzione.

```
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"
```

Per utilizzare un'immagine personalizzata per il tuo job Fluentd, inserisci la tua immagine nella `monitoringConfiguration.image` riga delle specifiche di distribuzione.

```
  monitoringConfiguration:
       image: <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
       cloudWatchMonitoringConfiguration:
```

```
        logGroupName: flink-log-group
        logStreamNamePrefix: custom-fluentd
```

Monitoraggio dell'operatore Flink Kubernetes e dei processi Flink

Questa sezione descrive diversi modi per monitorare i processi Flink con Amazon EMR su EKS. Queste includono l'integrazione di Flink con Amazon Managed Service for Prometheus, l'utilizzo di Flink Web Dashboard, che fornisce lo stato del lavoro e le metriche, o l'utilizzo di una configurazione di monitoraggio per inviare dati di log ad Amazon S3 e. Amazon CloudWatch

Argomenti

- Usa Amazon Managed Service for Prometheus per monitorare i lavori di Flink
- Usa l'interfaccia utente Flink per monitorare i lavori Flink
- Utilizza la configurazione di monitoraggio per monitorare l'operatore Flink Kubernetes e i lavori Flink

Usa Amazon Managed Service for Prometheus per monitorare i lavori di Flink

Puoi integrare il servizio Apache Flink con Amazon Managed Service per Prometheus (portale di gestione). Amazon Managed Service per Prometheus supporta l'acquisizione di parametri dai server di Amazon Managed Service per Prometheus in cluster in esecuzione su Amazon EKS. Amazon Managed Service per Prometheus funziona insieme a un server Prometheus già in esecuzione sul tuo cluster Amazon EKS. L'esecuzione dell'integrazione di Amazon Managed Service per Prometheus con l'operatore Amazon EMR Flink implementerà e configurerà in automatico un server Prometheus per l'integrazione con Amazon Managed Service per Prometheus.

1. Crea un Workspace Amazon Managed Service per Prometheus. Questo Workspace funge da endpoint di acquisizione. L'URL di scrittura remoto sarà necessario in un secondo momento.
2. Imposta ruoli IAM per gli account di servizio.

   Per questo metodo di onboarding, utilizza i ruoli IAM per gli account di servizio nel cluster Amazon EKS in cui il server Prometheus è in esecuzione. Questi ruoli sono denominati ruoli di servizio.

   Se non disponi già dei ruoli, configura i ruoli di servizio per l'acquisizione di parametri dai cluster Amazon EKS.

Prima di continuare, crea un ruolo IAM denominato `amp-iamproxy-ingest-role`.

3. Installa l'operatore Amazon EMR Flink per Amazon Managed Service per Prometheus.

Ora che disponi di un Workspace Amazon Managed Service per Prometheus, un ruolo IAM dedicato ad Amazon Managed Service per Prometheus e disponi delle autorizzazioni necessarie, puoi installare l'operatore Flink di Amazon EMR.

Creare un file `enable-amp.yaml` Questo file consente di utilizzare una configurazione personalizzata per sovrascrivere le impostazioni di Amazon Managed Service for Prometheus. Assicurati di utilizzare i tuoi ruoli.

```
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
```

Utilizza il comando Helm Install --set per trasmettere gli override al grafico `flink-kubernetes-operator`.

```
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
```

Questo comando installa automaticamente un reporter Prometheus nell'operatore sulla porta 9999. Ogni `FlinkDeployment` futura espone anche una porta `metrics` su 9249.

- I parametri dell'operatore Flink vengono visualizzati in Prometheus sotto l'etichetta `flink_k8soperator_`.
- I parametri di Flink Task Manager vengono visualizzati in Prometheus sotto l'etichetta `flink_taskmanager_`.
- I parametri di Flink Job Manager vengono visualizzate in Prometheus sotto l'etichetta `flink_jobmanager_`.

Usa l'interfaccia utente Flink per monitorare i lavori Flink

Per monitorare lo stato e le prestazioni di un'applicazione Flink in esecuzione, utilizza Flink Web Dashboard. Questa dashboard fornisce informazioni sullo stato del lavoro, sul numero di TaskManagers, sulle metriche e sui registri del lavoro. Consente inoltre di visualizzare e modificare la configurazione del processo Flink e di interagire con il cluster Flink per inviare o annullare i processi.

Procedura per accedere a Flink Web Dashboard per un'applicazione Flink in esecuzione su Kubernetes:

1. Utilizzate il `kubectl port-forward` comando per inoltrare una porta locale alla porta su cui è in esecuzione Flink Web Dashboard nei pod dell'applicazione Flink. TaskManager Per impostazione predefinita, la porta è 8081. Sostituisci *`deployment-name`* con il nome della distribuzione dell'applicazione Flink riportato sopra.

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

   Output di esempio:

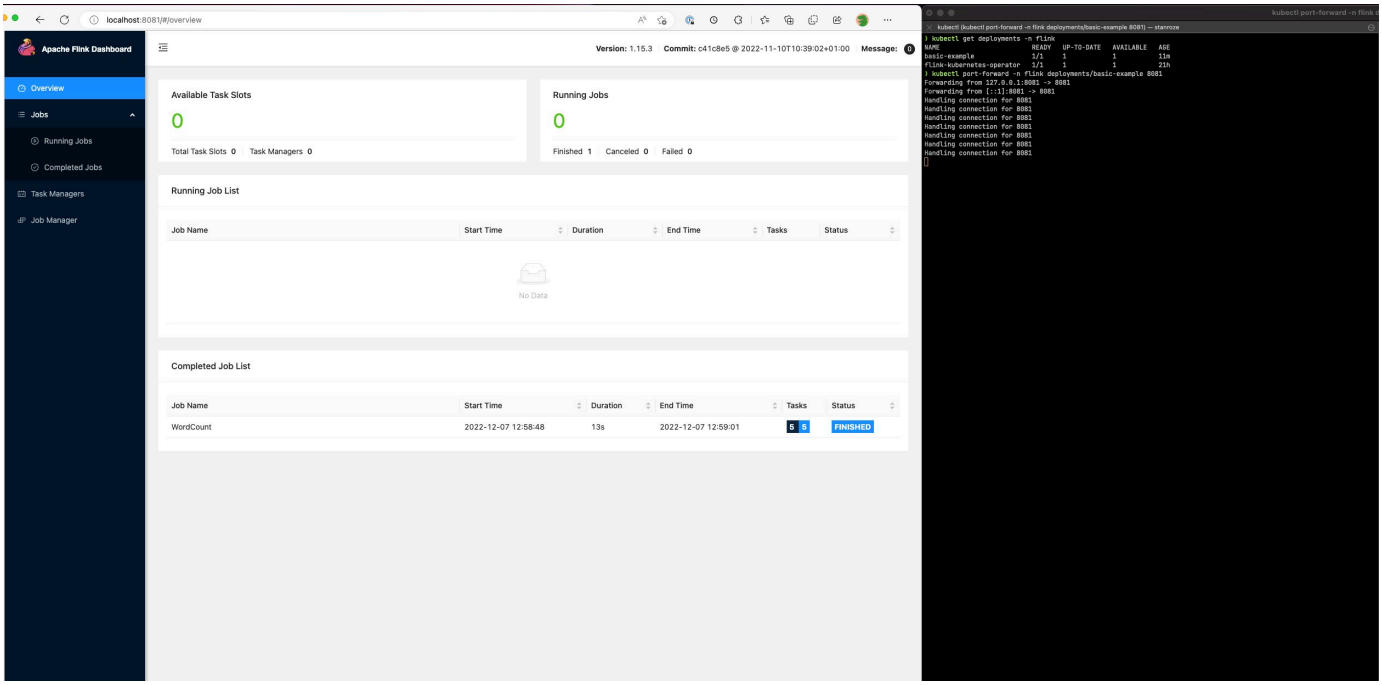
   ```
   kubectl get deployments -n flink-namespace
   NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
   basic-example               1/1       1            1           11m
   flink-kubernetes-operator   1/1       1            1           21h
   ```

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

2. Se desideri utilizzare una porta diversa a livello locale, usa il parametro:8081*`local-port`*.

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

3. In un browser web, apri `http://localhost:8081` (o `http://localhost:`*`local-port`*, se hai utilizzato una porta locale personalizzata) per accedere a Flink Web Dashboard. Questa dashboard mostra informazioni sull'applicazione Flink in esecuzione, come lo stato del lavoro, il numero di TaskManagers, le metriche e i registri del lavoro.



Utilizza la configurazione di monitoraggio per monitorare l'operatore Flink Kubernetes e i lavori Flink

La configurazione di monitoraggio consente di configurare facilmente l'archiviazione dei log dell'applicazione Flink e dei registri degli operatori su S3 and/or CloudWatch (è possibile sceglierne uno o entrambi). In questo modo aggiungi un sidecar FluentD ai JobManager tuoi pod TaskManager e successivamente inoltra i registri di questi componenti ai lavandini configurati.

> **Note**
>
> Devi configurare i ruoli IAM per l'account di servizio per l'operatore Flink e il processo Flink (Account di servizio) per poter utilizzare questa funzionalità, perché richiede l'interazione con altri Servizi AWS. Per la configurazione è necessario utilizzare IRSA in Configurazione dell'operatore Flink Kubernetes per Amazon EMR su EKS.

Log dell'applicazione Flink

Puoi definire questa configurazione nel modo seguente.

```
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
```

Di seguito sono elencate le opzioni di configurazione.

- `s3MonitoringConfiguration`: chiave di configurazione per impostare l'inoltro a S3
  - `logUri` (obbligatorio): il percorso del bucket S3 in cui desideri archiviare i log.

- Il percorso su S3 una volta caricati i log sarà simile al seguente.
  - Rotazione dei log non abilitata:

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

  - La rotazione dei log è abilitata. Puoi utilizzare sia un file ruotato sia un file corrente (uno senza marcatura temporale).

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

    Il formato seguente è un numero crescente.

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

- Per utilizzare questo strumento di inoltro, sono necessarie le seguenti autorizzazioni IAM.

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

- `cloudWatchMonitoringConfiguration`— chiave di configurazione verso cui impostare l'inoltro. CloudWatch
  - `logGroupName`(obbligatorio) — nome del gruppo di CloudWatch log a cui si desidera inviare i log (crea automaticamente il gruppo se non esiste).
  - `logStreamNamePrefix` (facoltativo): nome del flusso di log a cui si desidera inviare i log. Il valore predefinito è una stringa vuota. Il formato è il seguente:

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

  - Per utilizzare questo strumento di inoltro, sono necessarie le seguenti autorizzazioni IAM.

    ```
    {
    ```

```
    "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` (facoltativo): la chiave di configurazione per impostare i limiti delle risorse sul container sidecar Fluentbit avviato.
  - `memoryLimit` (facoltativo): il valore predefinito è 512Mi. Regola secondo necessità.
  - `cpuLimit` (facoltativo): questa opzione non ha un valore predefinito. Regola secondo necessità.
- `containerLogRotationConfiguration` (facoltativo): controlla il comportamento di rotazione dei log di container. È abilitato per impostazione predefinita.
  - `rotationSize` (obbligatorio): specifica la dimensione del file per la rotazione dei log. L'intervallo di valori possibili è compreso tra 2 KB e 2 GB. La parte relativa all'unità numerica del parametro rotationSize viene trasmessa come numero intero. Poiché i valori decimali non sono supportati, puoi specificare una dimensione di rotazione di 1,5 GB, ad esempio, con il valore 1.500 MB. Il valore predefinito è 2 GB.
  - `maxFilesToKeep` (obbligatorio): specifica il numero massimo di file da mantenere nel container in seguito alla rotazione. Il valore minimo è 1, quello massimo è 50. Il valore predefinito è 10.

Log dell'operatore Flink

È anche possibile abilitare l'archiviazione dei log dell'operatore utilizzando le opzioni seguenti nel file `values.yaml` nell'installazione del grafico Helm. Puoi abilitare S3 o entrambi. CloudWatch

```
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
```

Di seguito sono elencate le opzioni di configurazione disponibili in `monitoringConfiguration`.

- `s3MonitoringConfiguration`: imposta questa opzione per l'archiviazione su S3.
- `logUri` (obbligatorio): il percorso del bucket S3 in cui desideri archiviare i log.
- Di seguito sono riportati i formati che mostrano come potrebbero presentarsi i percorsi del bucket S3 una volta caricati i log.
  - Rotazione dei log non abilitata.

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

  - La rotazione dei log è abilitata. Puoi utilizzare sia un file ruotato sia un file corrente (uno senza marcatura temporale).

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

    L'indice di formato seguente è un numero crescente.

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

- `cloudWatchMonitoringConfiguration`— la chiave di configurazione verso cui configurare l'inoltro. CloudWatch
  - `logGroupName`(obbligatorio) — nome del gruppo di CloudWatch log a cui si desidera inviare i log. Se non esiste, il gruppo viene creato in automatico.
  - `logStreamNamePrefix` (facoltativo): nome del flusso di log a cui si desidera inviare i log. Il valore predefinito è una stringa vuota. Il formato in CloudWatch è il seguente:

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

- `sideCarResources` (facoltativo): la chiave di configurazione per impostare i limiti delle risorse sul container sidecar Fluentbit avviato.
  - `memoryLimit` (facoltativo): il limite di memoria. Regola secondo necessità. Il valore predefinito è 512Mi.

  - `cpuLimit`: il limite della CPU. Regola secondo necessità. Nessun valore predefinito.
- `containerLogRotationConfiguration` (facoltativo): controlla il comportamento di rotazione dei log di container. È abilitato per impostazione predefinita.
  - `rotationSize` (obbligatorio): specifica la dimensione del file per la rotazione dei log. L'intervallo di valori possibili è compreso tra 2 KB e 2 GB. La parte relativa all'unità numerica del parametro rotationSize viene trasmessa come numero intero. Poiché i valori decimali non sono supportati, puoi specificare una dimensione di rotazione di 1,5 GB, ad esempio, con il valore 1.500 MB. Il valore predefinito è 2 GB.
  - `maxFilesToKeep` (obbligatorio): specifica il numero massimo di file da mantenere nel container in seguito alla rotazione. Il valore minimo è 1, quello massimo è 50. Il valore predefinito è 10.

In che modo Flink supporta l'elevata disponibilità e la resilienza del lavoro

Le sezioni seguenti descrivono come Flink rende i lavori più affidabili e altamente disponibili. Lo fa attraverso funzionalità integrate come l'elevata disponibilità di Flink e varie funzionalità di ripristino in caso di guasti.

Argomenti

Uso dell'alta disponibilità (high availability, HA) per operatori Flink e applicazioni Flink

Questo argomento mostra come configurare l'alta disponibilità e descrive come funziona per alcuni casi d'uso diversi. Questi includono quando utilizzi il Job manager e quando utilizzi i kubernetes nativi di Flink.

Alta disponibilità dell'operatore Flink

Abilitiamo l'alta disponibilità per l'operatore Flink in modo da poter effettuare il failover su un operatore Flink in standby per ridurre al minimo i tempi di inattività nel circuito di controllo dell'operatore in caso

di errori. L'alta disponibilità è abilitata per impostazione predefinita e il numero predefinito di repliche dell'operatore di avvio è 2. È possibile configurare il campo delle repliche nel file `values.yaml` per il grafico Helm.

I seguenti campi sono personalizzabili:

- `replicas` (facoltativo, il valore predefinito è 2): l'impostazione di questo numero su un valore maggiore di 1 crea altri operatori in standby e consente un ripristino più rapido del processo.
- `highAvailabilityEnabled` (facoltativo, l'impostazione predefinita è true): controlla se desideri abilitare l'HA. La specificazione di questo parametro come true abilita il supporto per l'implementazione multi-AZ e imposta i parametri`flink-conf.yaml` corretti.

Puoi disabilitare l'HA per il tuo operatore impostando la seguente configurazione nel file `values.yaml`.

```
...
imagePullSecrets: []

replicas: 1

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

Implementazione multi-AZ

Creiamo i pod dell'operatore in più zone di disponibilità. Si tratta di un vincolo leggero e i pod degli operatori verranno programmati nella stessa AZ se non disponi di risorse sufficienti in un'altra AZ.

Determinazione della replica leader

Se HA è abilitato, le repliche utilizzano un lease per determinare quale dei due JMs è il leader e utilizzano un lease K8s per l'elezione del leader. È possibile descrivere il lease e consultare il campo .Spec.Holder Identity per determinare il leader attuale

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

Interazione Flink-S3

Configurazione delle credenziali di accesso

Assicurati di aver configurato IRSA con le autorizzazioni IAM appropriate per accedere al bucket S3.

Recupero dei jar dei processi dalla modalità di applicazione S3

L'operatore Flink supporta anche il recupero dei jar delle applicazioni da S3. È sufficiente fornire la posizione S3 per il JARuri nelle specifiche. FlinkDeployment

Puoi anche usare questa funzione per scaricare altri artefatti come gli script. PyFlink Lo script Python risultante viene inserito nel percorso `/opt/flink/usrlib/`.

L'esempio seguente mostra come utilizzare questa funzionalità per un lavoro. PyFlink Osserva i campi jarURI e 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
```

Connettori S3 di Flink

Flink viene fornito con due connettori S3 (elencati di seguito). Nelle sezioni seguenti viene spiegato quando utilizzare un determinato connettore.

Creazione di checkpoint: connettore Presto S3

- Imposta lo schema S3 su s3p://
- Il connettore consigliato da utilizzare per la creazione di checkpoint su s3. Per ulteriori informazioni, consulta S3-specific nella documentazione di Apache Flink.

Specificazione di esempio: FlinkDeployment

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

Lettura e scrittura su S3: connettore Hadoop S3

- Imposta lo schema S3 su `s3://` o (`s3a://`)
- Il connettore consigliato per la lettura e la scrittura di file da S3 (unico connettore S3 a implementare l'interfaccia FileSystem di Flink).
- Per impostazione predefinita, abbiamo impostato `fs.s3a.aws.credentials.provider` il `flink-conf.yaml` file, che è. `com.amazonaws.auth.WebIdentityTokenCredentialsProvider` Se ignori completamente il valore `flink-conf` predefinito e stai interagendo con S3, assicurati di utilizzare questo provider.

FlinkDeployment Specifiche di esempio

```
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

High Availability (HA) for Flink Deployments consente ai lavori di continuare a progredire anche se si verifica un errore temporaneo e si verificano arresti anomali. JobManager I processi verranno riavviati, ma dall'ultimo checkpoint riuscito, se l'HA è abilitata. Se l'HA non è abilitato, Kubernetes riavvierà il tuo lavoro JobManager, ma il tuo lavoro inizierà come un nuovo lavoro e perderà i progressi. Dopo aver configurato HA, possiamo dire a Kubernetes di archiviare i metadati HA in uno storage persistente a cui fare riferimento in caso di errore temporaneo JobManager e quindi riprendere i lavori dall'ultimo checkpoint riuscito.

L'HA è abilitata per impostazione predefinita per i processi Flink (il numero di repliche è impostato su 2, il che richiederà di fornire una posizione di archiviazione S3 per la persistenza dei metadati HA).

Configurazioni HA

```
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
```

Di seguito sono riportate le descrizioni delle configurazioni HA di cui sopra in Job Manager (definite in .spec.jobManager):

- `highAvailabilityEnabled` (facoltativa, l'impostazione predefinita è true): impostala su `false` se non desideri abilitare l'HA e non desideri utilizzare le configurazioni HA fornite. Puoi comunque manipolare il campo "repliche" per configurare manualmente l'HA.
- `replicas`(facoltativo, l'impostazione predefinita è 2): l'impostazione di questo numero su un valore maggiore di 1 crea altri standby JobManagers e consente un ripristino più rapido del lavoro. Se disabiliti l'HA, devi impostare il numero di repliche su 1, altrimenti continuerai a ricevere errori di convalida (è supportata solo 1 replica se l'HA non è abilitata).
- `storageDir` (obbligatoria): poiché per impostazione predefinita si utilizza 2 come numero di repliche, è necessario fornire una storageDir persistente. Attualmente questo campo accetta solo percorsi S3 come posizione di archiviazione.

Località dei pod

Se abiliti HA, cerchiamo anche di collocare i pod nella stessa AZ, il che comporta un miglioramento delle prestazioni (latenza di rete ridotta grazie alla presenza di pod nella stessa). AZs Si tratta di una procedura basata sul migliore tentativo, il che significa che se non disponi di risorse sufficienti nella zona in cui è programmata la maggior parte dei tuoi pod, i pod rimanenti verranno comunque programmati, ma potrebbero finire su un nodo esterno a tale AZ.

Determinazione della replica leader

Se HA è abilitato, le repliche utilizzano un lease per determinare quale delle due JMs è la leader e utilizzano una K8s Configmap come archivio dati per archiviare questi metadati. Se vuoi determinare il leader, puoi verificare il contenuto della Configmap e la chiave `org.apache.flink.k8s.leader.restserver` sotto i dati per trovare il pod K8s con l'indirizzo IP. Inoltre, puoi utilizzare i seguenti comandi bash.

```
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"
```

Processo Flink: Kubernetes nativo

Amazon EMR 6.13.0 e versioni successive supportano Kubernetes nativo di Flink per l'esecuzione di applicazioni Flink in modalità ad alta disponibilità su un cluster Amazon EKS.

> **Note**
>
> Devi disporre di un bucket Amazon S3 creato per archiviare i metadati ad alta disponibilità del processo quando invii il processo Flink. Se non desideri utilizzare questa funzionalità, puoi disattivarla. È abilitata per impostazione predefinita.

Per attivare la funzionalità di alta disponibilità di Flink, fornisci i seguenti parametri Flink quando esegui il comando CLI `run-application`. I parametri sono definiti sotto l'esempio.

```
-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`**: il bucket Amazon S3 in cui desideri archiviare i metadati ad alta disponibilità per il tuo processo.

  `Dkubernetes.jobmanager.replicas`: Il numero di pod Job Manager da creare come numero intero maggiore di 1.

  `Dkubernetes.cluster-id`: un ID univoco che identifica il cluster Flink.

Ottimizzazione dei tempi di riavvio dei processi Flink per le operazioni di ripristino delle attività e dimensionamento con Amazon EMR su EKS

Quando un'attività non riesce o quando si verifica un'operazione di dimensionamento, Flink tenta di rieseguire l'attività dall'ultimo checkpoint completato. L'esecuzione del processo di riavvio potrebbe richiedere un minuto o più, a seconda delle dimensioni dello stato del checkpoint e del numero di attività parallele. Durante il periodo di riavvio, le attività di backlog relative al processo possono

accumularsi. Esistono tuttavia alcuni modi in cui Flink ottimizza la velocità di ripristino e riavvio dei grafici di esecuzione per migliorare la stabilità del processo.

Questa pagina descrive alcuni dei modi in cui Amazon EMR Flink può migliorare il tempo di riavvio del lavoro durante il ripristino delle attività o la scalabilità delle operazioni su istanze locali. Le istanze Spot sono capacità di elaborazione inutilizzata disponibile a un prezzo scontato. Ha comportamenti unici, tra cui interruzioni occasionali, quindi è importante capire come Amazon EMR su EKS li gestisce, incluso il modo in cui Amazon EMR on EKS esegue la disattivazione e il riavvio dei lavori.

Argomenti

Ripristino locale delle attività

> **Note**
>
> Il ripristino locale delle attività è supportato con Flink su Amazon EMR su EKS 6.14.0 e versioni successive.

Con i checkpoint Flink, ogni attività produce un'istantanea del suo stato che Flink scrive in uno storage distribuito come Amazon S3. In caso di ripristino, le attività recuperano il loro stato dall'archiviazione distribuita. L'archiviazione distribuita offre tolleranza ai guasti e può ridistribuire lo stato durante il dimensionamento perché è accessibile a tutti i nodi.

Tuttavia, un archivio distribuito remoto presenta anche uno svantaggio: tutte le attività devono leggere il proprio stato da una posizione remota della rete. Ciò può comportare lunghi tempi di ripristino per stati di grandi dimensioni durante le operazioni di ripristino delle attività o di dimensionamento.

Il problema dei lunghi tempi di ripristino viene risolto mediante il ripristino locale delle attività. Le attività scrivono il loro stato su checkpoint in una memoria secondaria locale all'attività, ad esempio su un disco locale. Inoltre, memorizzano il loro stato nell'archiviazione principale, o su Amazon

S3, come nel nostro caso. Durante il ripristino, lo scheduler pianifica le attività sullo stesso Task Manager in cui le attività erano state eseguite in precedenza, in modo che possano essere ripristinate dall'archivio di stato locale anziché essere lette dall'archivio di stato remoto. Per ulteriori informazioni, consulta l'argomento relativo al ripristino locale delle attività nella documentazione di Apache Flink.

I nostri test di benchmark con processi di esempio hanno dimostrato che il tempo di ripristino è stato ridotto da pochi minuti a pochi secondi con il ripristino locale delle attività abilitato.

Per abilitare il ripristino locale delle attività, imposta le seguenti configurazioni nel file `flink-conf.yaml`. Specifica il valore dell'intervallo di checkpoint in millisecondi.

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

Ripristino locale delle attività tramite montaggio su volumi Amazon EBS

> **Note**
>
> Il ripristino locale delle attività di Amazon EBS è supportato con Flink su Amazon EMR su EKS 6.15.0 e versioni successive.

Con Flink su Amazon EMR su EKS, puoi fornire automaticamente i volumi Amazon EBS ai pod TaskManager per il ripristino locale delle attività. Il montaggio di un overlay predefinito include un volume di 10 GB, sufficiente per processi con uno stato inferiore. I lavori con stati di grandi dimensioni possono abilitare l'opzione montaggio automatico del volume EBS. I pod TaskManager vengono creati e montati automaticamente durante la creazione dei pod e rimossi durante l'eliminazione dei pod.

Utilizza i seguenti passaggi per abilitare il montaggio automatico dei volumi EBS per Flink in Amazon EMR su EKS:

1. Esporta i valori per le seguenti variabili che utilizzerai nei passaggi successivi.

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

2. Crea o aggiorna un file YAML `kubeconfig` per il cluster.

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

3. Crea un account di servizio IAM per il driver CSI (Container Storage Interface) di Amazon EBS sul cluster Amazon EKS.

```
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. Crea il driver CSI di Amazon EBS con il seguente comando:

```
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. Crea la classe di storage Amazon EBS con il seguente comando:

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

   Quindi applica la classe:

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

6. Helm installa l'operatore Kubernetes di Amazon EMR Flink con opzioni per creare un account di servizio. Questa operazione crea il `emr-containers-sa-flink` da utilizzare nell'implementazione Flink.

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

7. Per inviare il processo Flink e abilitare la fornitura automatica di volumi EBS per il ripristino locale delle attività, imposta le seguenti configurazioni nel file `flink-conf.yaml`. Regola il limite di dimensione in base alla dimensione dello stato del processo. Imposta `serviceAccount` su `emr-containers-sa-flink`. Specifica il valore dell'intervallo di checkpoint in millisecondi. Quindi, ometti `executionRoleArn`.

```
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
```

Quando sei pronto per eliminare il plug-in del driver CSI di Amazon EBS, usa i seguenti comandi:

```
  # 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
```

Checkpoint incrementale generico basato su log

> **Note**
>
> Il checkpoint incrementale generico basato su log è supportato con Flink su Amazon EMR su EKS 6.14.0 e versioni successive.

Il checkpoint incrementale generico basato su log è stato aggiunto in Flink 1.16 per migliorare la velocità dei checkpoint. Un intervallo di checkpoint più rapido spesso comporta una riduzione del lavoro di ripristino perché è necessario rielaborare un minor numero di eventi dopo il ripristino. Per ulteriori informazioni, consulta Improving speed and stability of checkpointing with generic log-based incremental checkpoints sul blog di Apache Flink.

Con processi di esempio, i nostri test di benchmark hanno dimostrato che il tempo di checkpoint si è ridotto da pochi minuti a pochi secondi con il checkpoint incrementale generico basato su log.

Per abilitare i checkpoint incrementali generici basati su log, imposta le seguenti configurazioni nel tuo file `flink-conf.yaml`. Specifica il valore dell'intervallo di checkpoint in millisecondi.

```
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
```

Ripristino granulare

> **Note**
>
> Il supporto per il ripristino granulare nel pianificatore predefinito è supportato con Flink su Amazon EMR su EKS 6.14.0 e versioni successive. Il supporto per il ripristino granulare nel pianificatore adattivo è disponibile con Flink su Amazon EMR su EKS 6.15.0 e versioni successive.

Quando un'attività riporta un errore durante l'esecuzione, Flink reimposta l'intero grafico di esecuzione e attiva una riesecuzione completa dall'ultimo checkpoint completato. Questa procedura

è più costosa della semplice riesecuzione delle attività non riuscite. Il ripristino granulare riavvia solo il componente connesso alla pipeline dell'attività non riuscita. Nell'esempio seguente, il grafico del processo ha 5 vertici (da `A` a `E`). Tutte le connessioni tra i vertici avvengono tramite pipeline con distribuzione uniforme e il comando `parallelism.default` per il processo è impostato su `2`.

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

Per questo esempio, le attività totali in esecuzione sono 10. La prima pipeline (da `a1` a `e1`) viene eseguita su un TaskManager (`TM1`) e la seconda pipeline (da `a2` a `e2`) viene eseguita su un altro TaskManager (`TM2`).

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

Esistono due componenti collegati tramite pipeline: `a1 # e1` e `a2 # e2`. Se `TM1` o `TM2` restituiscono un errore, l'errore influisce solo sulle 5 attività della pipeline in cui TaskManager era in esecuzione. La strategia di riavvio avvia solo il componente della pipeline interessato.

Il ripristino granulare funziona solo con processi Flink perfettamente paralleli. Non è supportato con le operazioni `keyBy()` o `redistribute()`. Per ulteriori informazioni, consulta FLIP-1: Fine Grained Recovery from Task Failures nel progetto Jira Flink Improvement Proposal.

Per abilitare il ripristino granulare, imposta le seguenti configurazioni nel file `flink-conf.yaml`.

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

Meccanismo di riavvio combinato nel pianificatore adattivo

> **Note**
>
> Il meccanismo di riavvio combinato nel pianificatore adattivo è supportato con Flink su Amazon EMR su EKS 6.15.0 e versioni successive.

Il pianificatore adattivo può regolare il parallelismo del processo in base agli slot disponibili. Riduce automaticamente il parallelismo se non sono disponibili abbastanza slot per soddisfare il parallelismo configurato del processo. Se diventano disponibili nuovi slot, il processo viene nuovamente

dimensionato in base al parallelismo configurato del processo. Un pianificatore adattivo evita i tempi di inattività del processo quando le risorse disponibili non sono sufficienti. Questo è il pianificatore supportato per Autoscaler di Flink. Per questi motivi, con Flink di Amazon EMR consigliamo il pianificatore adattivo. Tuttavia, i pianificatori adattivi potrebbero eseguire più riavvii in un breve periodo di tempo, un riavvio per ogni nuova risorsa aggiunta. Questo potrebbe comportare un calo delle prestazioni nel processo.

Con Amazon EMR 6.15.0 e versioni successive, Flink dispone di un meccanismo di riavvio combinato nel pianificatore adattivo che apre una finestra di riavvio quando viene aggiunta la prima risorsa e quindi attende fino all'intervallo di finestra configurato di 1 minuto predefinito. Esegue un singolo riavvio quando sono disponibili risorse sufficienti per eseguire il processo con il parallelismo configurato o quando scade l'intervallo.

Con processi di esempio, i nostri test di benchmark hanno dimostrato che questa funzionalità elabora il 10% dei record in più rispetto al comportamento predefinito quando si utilizzano pianificatori adattivi e autoscaler di Flink.

Per abilitare il meccanismo di riavvio combinato, imposta le seguenti configurazioni nel file `flink-conf.yaml`.

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

Disattivazione graduale delle istanze spot con Flink su Amazon EMR su EKS

Flink con Amazon EMR su EKS può migliorare il tempo di riavvio del processo durante le operazioni di ripristino delle attività o dimensionamento.

Panoramica

I rilasci 6.15.0 e successivi di Amazon EMR su EKS supportano la disattivazione graduale dei Task Manager sulle istanze spot in Amazon EMR su EKS con Apache Flink. Come parte di questa funzionalità, Amazon EMR su EKS con Flink offre le seguenti funzionalità:

- Just-in-time checkpoint: i job di streaming Flink possono rispondere all'interruzione dell'istanza Spot, eseguire il checkpoint just-in-time (JIT) dei processi in esecuzione e impedire la pianificazione di attività aggiuntive su queste istanze Spot. Il checkpoint JIT è supportato con un pianificatore predefinito e adattivo.

- Meccanismo di riavvio combinato: un meccanismo di riavvio combinato tenta di riavviare il processo dopo aver raggiunto il parallelismo delle risorse previsto o la fine della finestra configurata corrente. Ciò impedisce anche il riavvio consecutivo del processo che potrebbe essere causato da più terminazioni di istanze spot. Il meccanismo di riavvio combinato è disponibile solo con il pianificatore adattivo.

Queste funzionalità offrono i seguenti vantaggi:

- Puoi sfruttare le istanze spot per eseguire i Task Manager e ridurre le spese del cluster.
- Il miglioramento del riconoscimento di Task Manager per le istanze spot si traduce in una maggiore resilienza e in una pianificazione più efficiente dei processi.
- I tuoi lavori Flink avranno più tempo di attività perché ci saranno meno riavvii dopo la terminazione dell'istanza spot.

Come funziona lo smantellamento elegante

Considera il seguente esempio: esegui il provisioning di un cluster Amazon EMR su EKS che esegue Apache Flink e specifichi nodi On-Demand per Job Manager e nodi di istanza spot per Task Manager. Due minuti prima della terminazione, Task Manager riceve un avviso di interruzione.

In questo scenario, il Job Manager gestisce il segnale di interruzione dell'istanza spot, blocca la programmazione di attività aggiuntive sull'istanza spot e avvia il checkpoint JIT per il processo in streaming.

Quindi, il Job Manager riavvia il grafico dei processi solo dopo che la disponibilità di nuove risorse è sufficiente a soddisfare il parallelismo del processo corrente nella finestra dell'intervallo di riavvio. L'intervallo della finestra di riavvio viene deciso in base alla durata della sostituzione dell'istanza spot, alla creazione di nuovi pod di Task Manager e alla registrazione con Job Manager.

Prerequisiti

Per utilizzare il decommissioning grazioso, crea ed esegui un processo di streaming su un cluster Amazon EMR su EKS che esegue Apache Flink. Abilita il pianificatore adattivo e Task Manager pianificati su almeno un'istanza spot, come mostrato nell'esempio seguente. Devi utilizzare i nodi On-Demand per Job Manager e puoi utilizzare i nodi On-Demand per i Task Manager purché sia presente almeno un'istanza spot.

```
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
```

Configurazione

Questa sezione copre la maggior parte delle configurazioni che puoi specificare per le tue esigenze di disattivazione.

Chiave	Descrizione	Valore predefinito	Valori accettabili
`cluster.taskmanager.graceful-decommission.enabled`	Abilita la disattivazione graduale di Task Manager.	`true`	`true`, `false`

Chiave	Descrizione	Valore predefinito	Valori accettabili
`jobmanager.adaptive-scheduler.combined-restart.enabled`	Abilita il meccanismo di riavvio combinato nel pianificatore adattivo.	`false`	`true`, `false`
`jobmanager.adaptive-scheduler.combined-restart.window-interval`	L'intervallo combinato della finestra di riavvio per eseguire riavvii combinati per il processo. Un numero intero senza unità viene interpretato come millisecondi.	`1m`	Esempi: `30`, `60s`, `3m`, `1h`

Uso di Autoscaler per le applicazioni Flink

L'autoscaler dell'operatore può contribuire ad alleviare la congestione raccogliendo i parametri dai processi Flink e regolando in automatico il parallelismo a livello di vertice di processo. Di seguito è riportato un esempio di come potrebbe presentarsi la tua configurazione:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  ...
spec:
  ...
  flinkVersion: v1_18
  flinkConfiguration:
    job.autoscaler.enabled: "true"
    job.autoscaler.stabilization.interval: 1m
    job.autoscaler.metrics.window: 5m
    job.autoscaler.target.utilization: "0.6"
    job.autoscaler.target.utilization.boundary: "0.2"
    job.autoscaler.restart.time: 2m
    job.autoscaler.catch-up.duration: 5m
```

```
    pipeline.max-parallelism: "720"
  ...
```

Questa configurazione utilizza i valori predefiniti per l'ultima versione di Amazon EMR. Se utilizzi altre versioni, potresti avere valori diversi.

> **Note**
>
> A partire da Amazon EMR 7.2.0, non è necessario includere il prefisso nella configurazione`kubernetes.operator`. Se utilizzi la versione 7.1.0 o una versione precedente, devi utilizzare il prefisso prima di ogni configurazione. Ad esempio, è necessario specificare. `kubernetes.operator.job.autoscaler.scaling.enabled`

Di seguito sono elencate le opzioni di configurazione dell'autoscaler.

- `job.autoscaler.scaling.enabled`— specifica se abilitare l'esecuzione della scala dei vertici da parte dell'autoscaler. Il valore predefinito è `true`. Se disabilitate questa configurazione, l'autoscaler raccoglie solo le metriche e valuta il parallelismo suggerito per ogni vertice, ma non aggiorna i lavori.
- `job.autoscaler.stabilization.interval`: il periodo di stabilizzazione in cui non verrà eseguita alcun nuovo dimensionamento. L'impostazione predefinita è 5 minuti.
- `job.autoscaler.metrics.window`: la dimensione della finestra di aggregazione dei parametri di dimensionamento. Più grande è la finestra, più è fluida e stabile, ma l'autoscaler potrebbe essere più lento a reagire a variazioni improvvise del carico. L'impostazione predefinita è 15 minuti. Consigliamo di sperimentare utilizzando un valore compreso tra 3 e 60 minuti.
- `job.autoscaler.target.utilization`: l'utilizzo del vertice obiettivo per fornire prestazioni lavorative stabili e un certo buffer per le fluttuazioni del carico. L'impostazione predefinita è `0.7` il 70% utilization/load per i vertici del lavoro.
- `job.autoscaler.target.utilization.boundary`: il limite di utilizzo del vertice obiettivo che funge da buffer aggiuntivo per evitare il dimensionamento immediato in caso di fluttuazioni del carico. L'impostazione predefinita è`0.3`, il che significa che è consentita una deviazione del 30% dall'utilizzo target prima di attivare un'azione di ridimensionamento.
- `ob.autoscaler.restart.time`: il tempo previsto per il riavvio dell'applicazione. L'impostazione predefinita è 5 minuti.

- `job.autoscaler.catch-up.duration`: il tempo previsto per il recupero, ovvero l'elaborazione completa di qualsiasi backlog dopo il completamento di un'operazione di dimensionamento. L'impostazione predefinita è 5 minuti. Riducendo la durata del tempo di recupero, l'autoscaler deve riservare maggiore capacità aggiuntiva per le azioni di dimensionamento.
- `pipeline.max-parallelism`: il parallelismo massimo che l'autoscaler può utilizzare. L'autoscaler ignora questo limite se è superiore al parallelismo massimo configurato nella configurazione di Flink o direttamente su ciascun operatore. L'impostazione predefinita è -1. Tieni presente che l'autoscaler calcola il parallelismo come divisore del numero di parallelismo massimo, pertanto si consiglia di scegliere impostazioni di parallelismo massimo che prevedano molti divisori, invece di fare affidamento sui valori predefiniti forniti da Flink. Consigliamo di utilizzare multipli di 60 per questa configurazione, ad esempio 120, 180, 240, 360, 720, eccetera.

Per una pagina di riferimento più dettagliata sulla configurazione, consulta Configurazione di autoscaler.

Autotuning dei parametri Autoscaler

Questa sezione descrive il comportamento di ottimizzazione automatica per varie versioni di Amazon EMR. Inoltre, approfondisce le diverse configurazioni di auto-scaling.

> **Note**
>
> Amazon EMR 7.2.0 e versioni successive utilizzano la configurazione open source `job.autoscaler.restart.time-tracking.enabled` per consentire la stima del tempo di ridimensionamento. La stima del tempo di riscala ha le stesse funzionalità dell'autotuning di Amazon EMR, quindi non è necessario assegnare manualmente valori empirici al tempo di riavvio.
> Puoi comunque utilizzare l'autotuning di Amazon EMR se utilizzi Amazon EMR 7.1.0 o versioni precedenti.

7.2.0 and higher

Amazon EMR 7.2.0 e versioni successive misurano il tempo di riavvio effettivo richiesto per applicare le decisioni di scalabilità automatica. Nelle versioni 7.1.0 e precedenti, era necessario utilizzare la configurazione per configurare manualmente il tempo massimo di `job.autoscaler.restart.time` riavvio stimato. Utilizzando la

configurazione`job.autoscaler.restart.time-tracking.enabled`, è sufficiente inserire un orario di riavvio per il primo ridimensionamento. Successivamente, l'operatore registra l'orario di riavvio effettivo e lo utilizzerà per i ridimensionamenti successivi.

Per abilitare questo tracciamento, utilizzate il seguente comando:

```
job.autoscaler.restart.time-tracking.enabled: true
```

Di seguito sono riportate le configurazioni correlate per la stima del tempo di ridimensionamento.

Configurazione	Obbligatorio	Predefinita	Description
job.autoscaler.restart.time-tracking.enabled	No	False	Indica se Flink Autoscaler deve ottimizzare automatic amente le configurazioni nel tempo per ottimizza re le decisioni di ridimensi onamento. Nota che Autoscaler può solo regolare automaticamente il parametro Autoscaler. `restart.time`
job.autoscaler.restart.time	No	5 min	Il tempo di riavvio previsto utilizzato da Amazon EMR su EKS fino a quando l'operato re non sarà in grado di determinare il tempo di riavvio effettivo in base alle scalature precedenti.
job.autoscaler.restart.time-tracking.limit	No	15 min	Il tempo di riavvio massimo osservato quando è impostato su. `job.autos caler.restart.time-tracking.enabled true`

Di seguito è riportato un esempio di specifica di distribuzione che è possibile utilizzare per provare la stima del tempo di ridimensionamento:

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

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

    job.autoscaler.restart.time-tracking.enabled: "true"
    job.autoscaler.restart.time: "2m"
    job.autoscaler.restart.time-tracking.limit: "10m"

    jobmanager.scheduler: adaptive
    taskmanager.numberOfTaskSlots: "1"
    pipeline.max-parallelism: "12"

  executionRoleArn: <JOB ARN>
  emrReleaseLabel: emr-7.12.0-flink-latest
  jobManager:
    highAvailabilityEnabled: false
    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: stateless
```

Per simulare la contropressione, utilizza le seguenti specifiche di distribuzione.

```
job:
    jarURI: s3://<s3_bucket>/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

Carica il seguente script Python nel tuo bucket S3.

```
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()
```

Per verificare che la stima del tempo di ridimensionamento funzioni, assicurati che la registrazione dei DEBUG livelli dell'operatore Flink sia abilitata. L'esempio seguente mostra come aggiornare il file del grafico Helm. `values.yaml` Quindi reinstalla la tabella di comando aggiornata ed esegui nuovamente il job Flink.

```
log4j-operator.properties: |+
  # Flink Operator Logging Overrides
  rootLogger.level = DEBUG
```

Ottieni il nome del tuo leader pod.

```
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"
```

Esegui il comando seguente per ottenere il tempo di riavvio effettivo utilizzato nelle valutazioni delle metriche.

```
kubectl logs <FLINK-OPERATOR-POD-NAME> -c flink-kubernetes-operator -n <OPERATOR-
NAMESPACE> -f | grep "Restart time used in scaling summary computation"
```

Dovrebbero essere visualizzati log simili ai seguenti. Nota che viene utilizzato solo il primo ridimensionamento. `job.autoscaler.restart.time` I ridimensionamenti successivi utilizzano il tempo di riavvio osservato.

```
2024-05-16 17:17:32,590 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT2M
```

```
2024-05-16 17:19:03,787 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:19:18,976 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:20:50,283 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:22:21,691 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
```

7.0.0 and 7.1.0

Il Flink Autoscaler open source integrato utilizza numerose metriche per prendere le migliori decisioni di scalabilità. Tuttavia, i valori predefiniti che utilizza per i suoi calcoli sono pensati per essere applicabili alla maggior parte dei carichi di lavoro e potrebbero non essere ottimali per un determinato lavoro. La funzionalità di autotuning aggiunta alla versione Amazon EMR on EKS di Flink Operator esamina le tendenze storiche osservate su specifiche metriche acquisite e quindi cerca di calcolare il valore più ottimale su misura per il determinato lavoro.

Configurazione	Obbligatorio	Predefinita	Description
kubernetes.operator.job.autoscaler.autotune.enable	No	False	Indica se Flink Autoscaler deve ottimizzare automaticamente le configurazioni nel tempo per ottimizzare le decisioni di ridimensionamento degli autoscaler. Attualmente, Autoscaler può solo regolare automaticamente il parametro Autoscaler. `restart.time`
kubernetes.operator.job.autoscaler.autotune.metrics.history.max.count	No	3	Indica il numero di parametri storici di Amazon EMR su EKS che Autoscaler conserva nella mappa di configurazione dei parametri di Amazon EMR on EKS.

Configurazione	Obbligatorio	Predefinita	Description
kubernetes.operator.job.autoscaler.autotune.metrics.restart.count	No	3	Indica il numero di riavvii che Autoscaler esegue prima di iniziare a calcolare il tempo di riavvio medio per un determinato lavoro.

Per abilitare l'autotuning, è necessario aver completato quanto segue:

- Imposta `kubernetes.operator.job.autoscaler.autotune.enable:` su `true`
- Imposta `metrics.job.status.enable:` su `TOTAL_TIME`
- È seguita la configurazione di Using Autoscaler for Flink applications per abilitare la scalabilità automatica.

Di seguito è riportato un esempio di specifica di distribuzione che è possibile utilizzare per provare l'autotuning.

```
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
```

Per simulare la contropressione, utilizzate le seguenti specifiche di distribuzione.

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

Carica il seguente script Python nel tuo bucket S3.

```
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()
```

Per verificare che l'autotuner funzioni, usa i seguenti comandi. Nota che devi usare le informazioni del tuo leader pod per l'operatore Flink.

Per prima cosa procuratevi il nome del vostro leader pod.

```
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"
```

Una volta che hai il nome del tuo leader pod, puoi eseguire il seguente comando.

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

Dovreste vedere dei log simili ai seguenti.

```
[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
```

Manutenzione e risoluzione dei problemi per i job Flink su Amazon EMR su EKS

Le seguenti sezioni descrivono come mantenere i job Flink a esecuzione prolungata e forniscono indicazioni su come risolvere alcuni problemi comuni relativi ai job Flink.

Manutenzione delle applicazioni Flink

Argomenti

- Modalità di aggiornamento

Le applicazioni Flink sono in genere progettate per funzionare per lunghi periodi di tempo come settimane, mesi o persino anni. Come per tutti i servizi a lunga durata, le applicazioni di streaming Flink devono essere mantenute. La manutenzione include correzioni di bug, miglioramenti e migrazione a un cluster Flink di una versione successiva.

Quando le specifiche cambiano per le risorse `FlinkDeployment` e `FlinkSessionJob`, è necessario aggiornare l'applicazione in esecuzione. A tale scopo, l'operatore interrompe il processo in

esecuzione (a meno che non sia già sospeso) e lo ridistribuisce con le specifiche più recenti e, per le applicazioni stateful, con lo stato dell'esecuzione precedente.

Gli utenti controllano come gestire lo stato quando le applicazioni stateful si interrompono e vengono ripristinate con l'impostazione `upgradeMode` di `JobSpec`.

Modalità di aggiornamento

Introduzione opzionale

Stateless

Aggiornamenti di applicazioni stateless da stato vuoto.

Ultimo stato

Gli aggiornamenti rapidi in qualsiasi stato dell'applicazione (anche in caso di processi non riusciti) non richiedono un processo integro in quanto utilizzano sempre l'ultimo checkpoint corretto. Il ripristino manuale può essere necessario in caso di perdita dei metadati HA. Per limitare il periodo di tempo in cui il processo può tornare indietro quando sceglie l'ultimo checkpoint, è possibile configurare `kubernetes.operator.job.upgrade.last-state.max.allowed.checkpoint.age`. Se il checkpoint è più vecchio del valore configurato, al posto di un processo integro verrà utilizzato un savepoint. Questa funzionalità non è supportata in modalità Sessione.

Savepoint

Usa savepoint per l'aggiornamento, offrendo la massima sicurezza e la possibilità di fungere da punto. backup/fork Il savepoint verrà creato durante il processo di aggiornamento. Nota che il processo Flink deve essere in esecuzione per consentire la creazione del savepoint. Se il lavoro non è integro, verrà utilizzato l'ultimo checkpoint (a meno che kubernetes.operator.job.upgrade). last-state-fallback.enabled è impostato su false). Se l'ultimo checkpoint non è disponibile, l'aggiornamento del processo avrà esito negativo.

risoluzione dei problemi

La presente sezione descrive come risolvere i problemi con Amazon EMR su EKS. Per informazioni sulla risoluzione di problemi generali con Amazon EMR, consulta Risoluzione dei problemi di un cluster nella Guida alla gestione di Amazon EMR.

- Risoluzione dei problemi relativi ai lavori che utilizzano PersistentVolumeClaims (PVC)

- Risoluzione dei problemi relativi al dimensionamento automatico verticale di Amazon EMR su EKS
- Risoluzione dei problemi relativi all'operatore Amazon EMR su EKS

Risoluzione dei problemi di Apache Flink su Amazon EMR su EKS

Mappatura delle risorse non trovata durante l'installazione del grafico Helm

Durante l'installazione del grafico Helm, potresti visualizzare il seguente messaggio di errore.

```
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].
```

Per risolverlo, installa cert-manager per abilitare l'aggiunta del componente webhook. È necessario installare cert-manager in ogni cluster Amazon EKS che si utilizza.

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

Servizio AWS errore di accesso negato

Se visualizzi un errore access denied, conferma che il ruolo IAM per `operatorExecutionRoleArn` nel file grafico Helm `values.yaml` disponga delle autorizzazioni corrette. Verifica inoltre che il ruolo IAM in `executionRoleArn` nelle tue specifiche `FlinkDeployment` disponga delle autorizzazioni corrette.

`FlinkDeployment` è bloccato

Se il `FlinkDeployment` si blocca in uno stato di arresto, usa i seguenti passaggi per forzare l'eliminazione dell'implementazione:

1. Modifica l'esecuzione dell'implementazione.

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

2. Rimuovi questo finalizer.

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

3. Elimina l'implementazione.

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

AWSBadRequestException problema s3a quando si esegue un'applicazione Flink in un opt-in Regione AWS

Se esegui un'applicazione Flink in un opt-in Regione AWS, potresti visualizzare i seguenti errori:

```
Caused by: org.apache.hadoop.fs.s3a.AWSBadRequestException: getFileStatus on
s3://flink.txt: com.amazonaws.services.s3.model.AmazonS3Exception: Bad Request
(Service: Amazon S3; Status Code: 400; Error Code: 400 Bad Request; Request ID:
 ABCDEFGHIJKL; S3 Extended Request ID:
ABCDEFGHIJKLMNOP=; Proxy: null), S3 Extended Request ID: ABCDEFGHIJKLMNOP=:400 Bad
 Request: Bad Request
(Service: Amazon S3; Status Code: 400; Error Code: 400 Bad Request; Request ID:
 ABCDEFGHIJKL; S3 Extended Request ID: ABCDEFGHIJKLMNOP=; Proxy: null)
```

```
Caused by: org.apache.hadoop.fs.s3a.AWSBadRequestException: getS3Region on flink-
application: software.amazon.awssdk.services.s3.model.S3Exception: null
(Service: S3, Status Code: 400, Request ID: ABCDEFGHIJKLMNOP, Extended Request ID:
 ABCDEFGHIJKLMNOPQRST==):null: null
(Service: S3, Status Code: 400, Request ID: ABCDEFGHIJKLMNOP, Extended Request ID:
 AHl42uDNaTUFOus/5IIVNvSakBcMjMCH7dd37ky0vE6jhABCDEFGHIJKLMNOPQRST==)
```

Per correggere questi errori, utilizzate la seguente configurazione nel file di `FlinkDeployment` definizione.

```
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
```

Ti consigliamo inoltre di utilizzare il provider di SDKv2 credenziali:

```
fs.s3a.aws.credentials.provider:
 software.amazon.awssdk.auth.credentials.WebIdentityTokenFileCredentialsProvider
```

Se desideri utilizzare il provider di SDKv1 credenziali, assicurati che l'SDK supporti la tua regione di attivazione. Per ulteriori informazioni, consulta il repository. aws-sdk-java GitHub

Se ottieni `S3 AWSBadRequestException` quando esegui le istruzioni SQL Flink in una regione con attivazione, assicurati di impostare la configurazione `fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME` nelle specifiche di configurazione di flink.

S3A AWSBad RequestException quando si esegue un job di sessione Flink nelle regioni CN

Per le versioni 6.15.0 - 7.2.0 di Amazon EMR, potresti riscontrare i seguenti messaggi di errore quando esegui un processo di sessione Flink nelle regioni CN. Questi includono Cina (Pechino) e Cina (Ningxia):

```
Error:
 {"type":"org.apache.flink.kubernetes.operator.exception.ReconciliationException","message":"or
                   getFileStatus on s3://ABCDPath:
 software.amazon.awssdk.services.s3.model.S3Exception: null (Service: S3, Status Code:
 400, Request ID: ABCDEFGH, Extended Request ID:
                   ABCDEFGH:null: null (Service: S3, Status Code: 400, Request ID:
 ABCDEFGH, Extended Request ID: ABCDEFGH","additionalMetadata":{},"throwableList":

 [{"type":"org.apache.hadoop.fs.s3a.AWSBadRequestException","message":"getFileStatus on
 s3://ABCDPath: software.amazon.awssdk.services.s3.model.S3Exception:
                   null (Service: S3, Status Code: 400, Request ID: ABCDEFGH, Extended
 Request ID: ABCDEFGH:null: null (Service: S3, Status Code: 400, Request ID: ABCDEFGH,
                   Extended Request ID: ABCDEFGH","additionalMetadata":{}},
{"type":"software.amazon.awssdk.services.s3.model.S3Exception","message":"null
 (Service: S3, Status Code: 400,
                   Request ID: ABCDEFGH, Extended Request ID:
 ABCDEFGH","additionalMetadata":{}}]}
```

C'è una consapevolezza di questo problema. Il team sta lavorando per applicare le patch agli operatori flink per tutte queste versioni di rilascio. Tuttavia, prima di completare la patch, per correggere questo errore, è necessario scaricare la tabella di comando degli operatori flink, decomprimerla (estrarre il file compresso) e apportare modifiche alla configurazione nella tabella di comando.

I passaggi specifici sono i seguenti:

1. Passa alla cartella locale del Helm Chart, in particolare cambia le directory, ed esegui la seguente riga di comando per estrarre il diagramma di comando e decomprimerlo (estrarlo).

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

```
tar -zxvf flink-kubernetes-operator-$VERSION.tgz
```

2. Vai nella cartella Helm Chart e trova il file. `templates/flink-operator.yaml`
3. Trova `flink-operator-config` ConfigMap e aggiungi la seguente `fs.s3a.endpoint.region` configurazione in. `flink-conf.yaml` Esempio:

```
{{- if .Values.defaultConfiguration.create }}
apiVersion: v1
kind: ConfigMap
metadata:
  name: flink-operator-config
  namespace: {{ .Release.Namespace }}
  labels:
    {{- include "flink-operator.labels" . | nindent 4 }}
data:
  flink-conf.yaml: |+
fs.s3a.endpoint.region: {{ .Values.emrContainers.awsRegion }}
```

4. Installa la tabella di comando locale ed esegui il tuo lavoro.

Rilasci supportati per Amazon EMR su EKS con Apache Flink

Apache Flink è disponibile con le seguenti rilasci di Amazon EMR su EKS. Per informazioni su tutti i rilasci disponibili, consulta Rilasci di Amazon EMR su EKS.

Etichetta di rilascio	Java	Flink	Operatore Flink
emr-7.2.0-flink-latest	17	1,18,1	-
emr-7.2.0-flink-k8s-operator-latest	11	-	1.8.0

Etichetta di rilascio	Java	Flink	Operatore Flink
emr-7.1.0-flink-latest	17	1.18.1	-
emr-7.1.0-flink-k8s-operator-latest	11	-	1.6.1
emr-7.0.0-flink-latest	11	1.18.0	-
emr-7.0.0-flink-k8s-operator-latest	11	-	1.6.1
emr-6.15.0-flink-latest	11	1.17.1	-
emr-6.15.0-flink-k8s-operator-latest	11	-	1.6.0
emr-6.14.0-flink-latest	11	1.17.1	-
emr-6.14.0-flink-k8s-operator-latest	11	-	1.6.0
emr-6.13.0-flink-latest	11	1.17.0	-
emr-6.13.0-flink-k8s-operator-latest	11	-	1.5.0

Esecuzione di job Spark con Amazon EMR su EKS

L'esecuzione di un job è un'unità di lavoro, ad esempio un jar Spark, PySpark uno script o una query SparkSQL, che invii ad Amazon EMR su EKS. Questo argomento fornisce una panoramica sulla gestione delle esecuzioni di job utilizzando AWS CLI, sulla visualizzazione delle esecuzioni di job utilizzando la console Amazon EMR e sulla risoluzione dei comuni errori di esecuzione dei job.

Tieni presente che non puoi eseguire i job IPv6 Spark su Amazon EMR su EKS.

> **Note**
>
> Prima di inviare un'esecuzione di processo con Amazon EMR su EKS, devi completare le fasi indicate in Configurazione di Amazon EMR su EKS.

Argomenti

Esecuzione di processi Spark con `StartJobRun`

Questa sezione include passaggi di configurazione dettagliati per preparare l'ambiente all'esecuzione dei job Spark e quindi fornisce step-by-step istruzioni per inviare un job eseguito con parametri specificati.

Argomenti

Configurazione di Amazon EMR su EKS

Completa le attività seguenti per effettuare la configurazione di Amazon EMR su EKS. Se hai già effettuato la registrazione ad Amazon Web Services (AWS) e stai utilizzando Amazon EKS, ti manca poco per cominciare a utilizzare Amazon EMR su EKS. Salta le eventuali attività che hai già completato.

> **Note**
>
> È inoltre possibile seguire il Workshop su Amazon EMR su EKS per configurare tutte le risorse necessarie per eseguire i processi Spark in Amazon EMR su EKS. Il workshop fornisce anche l'automazione utilizzando CloudFormation modelli per creare le risorse necessarie per iniziare. Per altri modelli e best practice, consulta la nostra Guida alle best practice per i contenitori EMR su. GitHub

1. Installa o aggiorna alla versione più recente di AWS CLI
2. Configura kubectl ed eksctl.
3. Inizia a usare Amazon EKS — eksctl
4. Abilita l'accesso al cluster per Amazon EMR su EKS
5. Abilita IAM Roles per il cluster EKS
6. Concessione dell'accesso ad Amazon EMR su EKS agli utenti
7. Registrazione del cluster Amazon EKS con Amazon EMR

Abilitare l'accesso ai cluster per Amazon EMR su EKS

Le sezioni seguenti mostrano un paio di modi per abilitare l'accesso al cluster. La prima consiste nell'utilizzare la gestione degli accessi al cluster Amazon EKS (CAM) e la seconda mostra come eseguire passaggi manuali per abilitare l'accesso al cluster.

Abilita l'accesso al cluster utilizzando EKS Access Entry (consigliato)

> **Note**
>
> La funzionalità `aws-auth` ConfigMap è obsoleta. Il metodo consigliato per gestire l'accesso a Kubernetes APIs è Access Entries.

Amazon EMR è integrato con Amazon EKS Cluster Access Management (CAM), quindi puoi automatizzare la configurazione delle policy AuthN e AuthZ necessarie per eseguire i job Amazon EMR Spark nei namespace dei cluster Amazon EKS. Quando crei un cluster virtuale da uno spazio dei nomi del cluster Amazon EKS, Amazon EMR configura automaticamente tutte le autorizzazioni necessarie, quindi non è necessario aggiungere passaggi aggiuntivi ai flussi di lavoro correnti.

> **Note**
>
> L'integrazione di Amazon EMR con Amazon EKS CAM è supportata solo per i nuovi Amazon EMR su cluster virtuali EKS. Non puoi migrare i cluster virtuali esistenti per utilizzare questa integrazione.

Prerequisiti

- Assicurati di utilizzare la versione 2.15.3 o successiva di AWS CLI
- Il tuo cluster Amazon EKS deve avere la versione 1.23 o successiva.

Configurazione

Per configurare l'integrazione tra Amazon EMR e le operazioni AccessEntry API di Amazon EKS, assicurati di aver completato i seguenti elementi:

- Assicurati che quello `authenticationMode` del tuo cluster Amazon EKS sia impostato su`API_AND_CONFIG_MAP`.

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

  Se non lo è già, imposta `authenticationMode` su`API_AND_CONFIG_MAP`.

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

Per ulteriori informazioni sulle modalità di autenticazione, vedere Modalità di autenticazione del cluster.

- Assicurati che il ruolo IAM che stai utilizzando per eseguire le operazioni `CreateVirtualCluster` e `DeleteVirtualCluster` API disponga anche delle seguenti autorizzazioni:

```
{
  "Effect": "Allow",
  "Action": [
    "eks:CreateAccessEntry"
  ],
  "Resource":
 "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:cluster/<EKS_CLUSTER_NAME>"
},
{
  "Effect": "Allow",
  "Action": [
    "eks:DescribeAccessEntry",
    "eks:DeleteAccessEntry",
    "eks:ListAssociatedAccessPolicies",
    "eks:AssociateAccessPolicy",
    "eks:DisassociateAccessPolicy"
  ],
  "Resource": "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:access-entry/
<EKS_CLUSTER_NAME>/role/<AWS_ACCOUNT_ID>/AWSServiceRoleForAmazonEMRContainers/*"
}
```

Concetti e terminologia

Di seguito è riportato un elenco di terminologie e concetti relativi ad Amazon EKS CAM.

- Cluster virtuale (VC): rappresentazione logica dello spazio dei nomi creato in Amazon EKS. È un collegamento 1:1 a uno spazio dei nomi del cluster Amazon EKS. Puoi usarlo per eseguire carichi di lavoro Amazon EMR su un cluster Amazon EKS all'interno dello spazio dei nomi specificato.
- Namespace: meccanismo per isolare gruppi di risorse all'interno di un singolo cluster EKS.

- Politica di accesso: autorizzazioni che garantiscono l'accesso e le azioni a un ruolo IAM all'interno di un cluster EKS.
- Accesso: una voce creata con un ruolo arn. Puoi collegare la voce di accesso a una policy di accesso per assegnare autorizzazioni specifiche nel cluster Amazon EKS.
- Cluster virtuale integrato con accesso EKS: il cluster virtuale creato utilizzando le operazioni API di accesso di Amazon EKS.

Abilita l'accesso al cluster utilizzando **`aws-auth`**

È necessario consentire ad Amazon EMR su EKS l'accesso a uno spazio dei nomi specifico nel cluster mediante le seguenti operazioni: creazione di un ruolo Kubernetes, associazione del ruolo a un utente Kubernetes e mappatura dell'utente Kubernetes con il ruolo collegato al servizio `AWSServiceRoleForAmazonEMRContainers`. Queste operazioni sono automatizzate in `eksctl` quando il comando di mappatura di identità IAM viene utilizzato con `emr-containers` come nome di servizio. È possibile eseguire facilmente queste operazioni utilizzando il comando seguente.

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

Sostituiscilo *`my_eks_cluster`* con il nome del tuo cluster Amazon EKS e sostituiscilo *`kubernetes_namespace`* con lo spazio dei nomi Kubernetes creato per eseguire carichi di lavoro Amazon EMR.

> **Important**
>
> È necessario scaricare la versione più recente di eksctl utilizzando il passaggio precedente. Configurare kubectl ed eksctl per utilizzare questa funzionalità.

Procedure manuali per abilitare l'accesso al cluster per Amazon EMR su EKS

È anche possibile utilizzare la seguente procedura manuale per abilitare l'accesso cluster per Amazon EMR su EKS.

1. Creazione di un ruolo Kubernetes in uno spazio dei nomi specifico

Amazon EKS 1.22 - 1.29

Con Amazon EKS 1.22 - 1.29, esegui il comando seguente per creare un ruolo Kubernetes in uno spazio dei nomi specifico. Questo ruolo concede le necessarie autorizzazioni RBAC ad Amazon EMR su EKS.

```
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

Con Amazon EKS 1.21 e versioni precedenti, esegui il comando seguente per creare un ruolo Kubernetes in uno spazio dei nomi specifico. Questo ruolo concede le necessarie autorizzazioni RBAC ad Amazon EMR su EKS.

```
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. Creazione dell'associazione di un ruolo Kubernetes allo spazio dei nomi

   Esegui questo comando per creare l'associazione di un ruolo Kubernetes a uno spazio dei nomi specifico. Questa associazione di ruolo concede le autorizzazioni definite nel ruolo creato nel passaggio precedente a un utente denominato `emr-containers`. Questo utente identifica i ruoli collegati ai servizi di Amazon EMR su EKS e, di conseguenza, consente ad Amazon EMR su EKS di eseguire le operazioni definite dal ruolo che hai creato.

   ```
   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. Aggiornamento della mappa di configurazione **`aws-auth`** di Kubernetes

   Puoi utilizzare una delle seguenti opzioni per mappare il ruolo collegato ai servizi di Amazon EMR su EKS con l'utente `emr-containers` associato al ruolo Kubernetes nella fase precedente.

Opzione 1: usando **`eksctl`**

Esegui il comando `eksctl` seguente per mappare il ruolo collegato ai servizi di Amazon EMR su EKS con l'utente `emr-containers`.

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

Opzione 2: Senza utilizzare eksctl

1. Esegui il seguente comando per aprire la mappa di configurazione `aws-auth` nell'editor di testo.

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

   > **Note**
   >
   > Se ricevi un messaggio di erroreError `from server (NotFound): configmaps "aws-auth" not found`, consulta la procedura in Aggiungere ruoli utente nella Guida per l'utente di Amazon EKS per applicare lo stock ConfigMap.

2. Aggiungi i dettagli del ruolo collegato ai servizi di Amazon EMR su EKS alla sezione `mapRoles` di `ConfigMap`, in `data`. Aggiungi questa sezione se non esiste già nel file. La sezione `mapRoles` aggiornata in dati ha l'aspetto seguente.

   ```
   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. Salva il file ed esci dall'editor di testo.

Abilita IAM Roles per il cluster EKS

I seguenti argomenti descrivono in dettaglio le opzioni per abilitare i ruoli IAM.

Argomenti

Opzione 1: abilitare EKS Pod Identity sul cluster EKS

Le associazioni Amazon EKS Pod Identity offrono la possibilità di gestire le credenziali per le tue applicazioni, in modo simile al modo in cui i profili di EC2 istanza Amazon forniscono le credenziali alle istanze Amazon EC2 . Amazon EKS Pod Identity fornisce le credenziali per i tuoi carichi di lavoro con un'API EKS Auth aggiuntiva e un pod di agenti che viene eseguito su ogni nodo.

Amazon EMR su EKS inizia a supportare l'identità dei pod EKS dalla versione emr-7.3.0 per il modello di invio. StartJobRun

Per ulteriori informazioni sulle identità dei pod EKS, consulta Comprendere come funziona EKS Pod Identity.

Perché EKS Pod Identities?

Come parte della configurazione EMR, il Job Execution Role deve stabilire limiti di fiducia tra un ruolo IAM e gli account di servizio in uno spazio dei nomi specifico (di cluster virtuali EMR). Con IRSA, ciò è stato ottenuto aggiornando la politica di fiducia dell'EMR Job Execution Role. Tuttavia, a causa del limite massimo di 4096 caratteri alla lunghezza della policy di fiducia IAM, esisteva un vincolo alla condivisione di un singolo ruolo IAM Job Execution su un massimo di dodici (12) cluster EKS.

Con il supporto di EMR per Pod Identities, il confine di fiducia tra i ruoli IAM e gli account di servizio viene ora gestito dal team EKS tramite l'associazione di EKS pod identity. APIs

> **Note**
>
> Il limite di sicurezza per l'identità del pod EKS è ancora a livello di account di servizio, non a livello di pod.

Considerazioni sull'identità dei pod

Per informazioni sulle limitazioni relative all'identità dei Pod, consulta le considerazioni relative all'identità di EKS Pod.

Prepara EKS Pod Identity in EKS Cluster

Verifica se l'autorizzazione richiesta esiste in NodeInstanceRole

Il ruolo del nodo `NodeInstanceRole` richiede l'autorizzazione dell'agente per eseguire l'`AssumeRoleForPodIdentity`azione nell'API di autenticazione EKS. Puoi aggiungere quanto segue ad Amazon EKSWorker NodePolicy, definito nella Guida per l'utente di Amazon EKS, o utilizzare una politica personalizzata.

Se il tuo cluster EKS è stato creato con una versione eksctl successiva alla 0.181.0, Amazon EKSWorkerNodePolicy, inclusa l'`AssumeRoleForPodIdentity`autorizzazione richiesta, verrà assegnato automaticamente al ruolo del nodo. Se l'autorizzazione non è presente, aggiungi manualmente la seguente autorizzazione ad Amazon EKSWorker NodePolicy che consente di assumere un ruolo per l'identità del pod. Questa autorizzazione è necessaria all'agente di identità dei pod EKS per recuperare le credenziali dei pod.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks-auth:AssumeRoleForPodIdentity"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEKSAUTHAssumeroleforpodidentity"
    }
  ]
}
```

Crea il componente aggiuntivo EKS pod identity agent

Usa il seguente comando per creare il componente aggiuntivo EKS Pod Identity Agent con la versione più recente:

```
aws eks create-addon --cluster-name cluster-name --addon-name eks-pod-identity-agent

kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Utilizza i seguenti passaggi per creare il componente aggiuntivo EKS Pod Identity Agent dalla console Amazon EKS:

1. Apri la console Amazon EKS: console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi scegli il nome del cluster per cui configurare il componente aggiuntivo di EKS Pod Identity Agent.
3. Selezionare la scheda Componenti aggiuntivi.
4. Scegli Ottieni altri componenti aggiuntivi.
5. Seleziona la casella nella parte superiore destra di quella del componente aggiuntivo relativo a EKS Pod Identity Agent e scegli Avanti.
6. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona una versione qualsiasi nell'elenco a discesa Versione.
7. (Facoltativo) Espandi Impostazioni di configurazione facoltative per inserire una configurazione aggiuntiva. Ad esempio, puoi fornire una posizione alternativa per l'immagine del container e `ImagePullSecrets`. Lo Schema JSON con le chiavi accettate sono mostrate in Schema di configurazione del componente aggiuntivo.

   Inserisci le chiavi e i valori di configurazione in Valori di configurazione.
8. Scegli Next (Successivo).
9. Verifica che i pod dell'agente siano in esecuzione sul tuo cluster tramite la CLI.

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

Un esempio di output è il seguente:

```
NAME                              READY   STATUS    RESTARTS      AGE
eks-pod-identity-agent-gmqp7      1/1     Running   1 (24h ago)   24h
eks-pod-identity-agent-prnsh      1/1     Running   1 (24h ago)   24h
```

Questo ne imposta uno nuovo DaemonSet nel `kube-system` namespace. Amazon EKS Pod Identity Agent, in esecuzione su ogni nodo EKS, utilizza l'AssumeRoleForPodIdentityazione per recuperare credenziali temporanee dall'API EKS Auth. Queste credenziali vengono quindi rese disponibili per l' AWS SDKs esecuzione all'interno dei contenitori.

Per ulteriori informazioni, consulta il prerequisito nel documento pubblico: Configurazione di Amazon EKS Pod Identity Agent.

Creare un ruolo Job Execution

Crea o aggiorna un ruolo di esecuzione del lavoro che consenta EKS Pod Identity

Per eseguire carichi di lavoro con Amazon EMR su EKS, devi creare un ruolo IAM. Nella presente documentazione, tale ruolo viene definito ruolo di esecuzione di processo. Per ulteriori informazioni su come creare il ruolo IAM, consulta Creazione di ruoli IAM nella Guida per l'utente.

Inoltre, è necessario creare una policy IAM che specifichi le autorizzazioni necessarie per il ruolo di esecuzione del lavoro e quindi allegare questa policy al ruolo per abilitare EKS Pod Identity.

Ad esempio, avete il seguente ruolo di esecuzione del lavoro. Per ulteriori informazioni, vedere Creare un ruolo di esecuzione del lavoro.

```
arn:aws:iam::111122223333:role/PodIdentityJobExecutionRole
```

> **Important**
>
> Amazon EMR su EKS crea automaticamente account di servizio Kubernetes, in base al nome del ruolo di esecuzione del lavoro. Assicurati che il nome del ruolo non sia troppo lungo, poiché il lavoro potrebbe fallire se la combinazione di e `service_account_name` supera `cluster_name` il `pod_name` limite di lunghezza.

Configurazione del ruolo di esecuzione del lavoro: assicurati che il ruolo di esecuzione del lavoro sia creato con l'autorizzazione di fiducia seguente per EKS Pod Identity. Per aggiornare un ruolo di esecuzione del lavoro esistente, configuralo in modo che consideri attendibile il seguente principale del servizio EKS come autorizzazione aggiuntiva nella politica di fiducia. Questa autorizzazione di fiducia può coesistere con le politiche di trust IRSA esistenti.

```
cat >trust-relationship.json <<EOF
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
EOF
```

Autorizzazione utente: gli utenti richiedono l'`iam:PassRole`autorizzazione per eseguire chiamate `StartJobRun` API o inviare lavori. Questa autorizzazione consente agli utenti di passare il ruolo di esecuzione del lavoro a EMR su EKS. Gli amministratori di Job devono disporre dell'autorizzazione per impostazione predefinita.

Di seguito è riportata l'autorizzazione necessaria per un utente:

```
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::111122223333:role/PodIdentityJobExecutionRole",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "pods.eks.amazonaws.com"
        }
    }
}
```

Per limitare ulteriormente l'accesso degli utenti a cluster EKS specifici, aggiungi il filtro AssociatedResourceArn degli attributi alla policy IAM. Limita l'assunzione di ruoli ai cluster EKS autorizzati, rafforzando i controlli di sicurezza a livello di risorsa.

```
"Condition": {
        "ArnLike": {
```

```
            "iam:AssociatedResourceARN": [
                "arn:aws:eks:us-west-2:111122223333:cluster/*"
            ]
        }
```

Configura le associazioni di identità dei pod EKS

Prerequisito

Assicurati che l'identità IAM che crea l'associazione di identità del pod, ad esempio un utente amministratore EKS, disponga dell'autorizzazione `eks:CreatePodIdentityAssociation` e`iam:PassRole`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks:CreatePodIdentityAssociation"
      ],
      "Resource": [
        "arn:aws:eks:*:*:cluster/*"
      ],
      "Sid": "AllowEKSCreatepodidentityassociation"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "pods.eks.amazonaws.com"
        }
      },
      "Sid": "AllowIAMPassrole"
```

```
    }
  ]
}
```

Creare associazioni per il ruolo e l'account del servizio EMR

Create EMR role associations through the AWS CLI

Quando invii un lavoro a un namespace Kubernetes, un amministratore deve creare associazioni tra il ruolo di esecuzione del lavoro e l'identità dell'account del servizio gestito EMR. Ricorda che l'account di servizio gestito da EMR viene creato in automatico all'invio del processo, nell'ambito dello spazio dei nomi in cui viene inviato il processo.

Con la AWS CLI (precedente versione 2.24.0), esegui il comando seguente per creare associazioni di ruoli con l'identità del pod.

Esegui il comando seguente per creare associazioni di ruoli con l'identità del pod:

```
aws emr-containers create-role-associations \
        --cluster-name mycluster \
        --namespace mynamespace \
        --role-name JobExecutionRoleIRSAv2
```

Nota:

- Ogni cluster può avere un limite di 1.000 associazioni. La mappatura di ogni ruolo di esecuzione del lavoro (namespace) richiederà 3 associazioni per i pod Job Submitter, Driver ed Executor.
- È possibile associare solo ruoli che si trovano nello stesso account del cluster. AWS È possibile delegare l'accesso da un altro account al ruolo di questo account configurato per l'utilizzo delle associazioni EKS Pod Identity. Per un tutorial sulla delega dell'accesso e`AssumeRole`, consulta il tutorial IAM: delega l'accesso tra AWS account utilizzando i ruoli IAM.

Create EMR role associations through Amazon EKS

EMR crea un account di servizio con un determinato schema di denominazione quando viene inviato un lavoro. Per creare associazioni manuali o integrare questo flusso di lavoro con l' AWS SDK, procedi nel seguente modo:

Costruisci il nome dell'account del servizio:

```
emr-containers-sa-spark-%(SPARK_ROLE)s-%(AWS_ACCOUNT_ID)s-
%(BASE36_ENCODED_ROLE_NAME)s
```

Gli esempi seguenti creano un'associazione di ruoli per un ruolo di Job execution di esempio JobExecutionRoleIRSAv2.

Esempi di associazioni di ruoli:

```
RoleName: JobExecutionRoleIRSAv2
Base36EncodingOfRoleName: 2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
```

Esempio di comando CLI:

```
# setup for the client service account (used by job runner pod)
# emr-containers-sa-spark-client-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
client-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

# driver service account
# emr-containers-sa-spark-driver-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
driver-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

# executor service account
# emr-containers-sa-spark-executor-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
executor-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
```

Dopo aver completato tutti i passaggi richiesti per l'identità del pod EKS, puoi saltare i seguenti passaggi per la configurazione IRSA:

- Abilita IAM Roles for Service Accounts (IRSA) sul cluster EKS
- Crea un ruolo di esecuzione del lavoro

- Aggiorna la politica di fiducia del ruolo di esecuzione del lavoro

Puoi passare direttamente al passaggio seguente: Concedi agli utenti l'accesso ad Amazon EMR su EKS

Elimina le associazioni di ruoli

Ogni volta che si elimina un cluster virtuale o un ruolo di esecuzione del lavoro e non si desidera più concedere l'accesso a EMR ai relativi account di servizio, è necessario eliminare le associazioni per il ruolo. Questo perché EKS consente associazioni con risorse inesistenti (namespace e account di servizio). Amazon EMR su EKS consiglia di eliminare le associazioni se lo spazio dei nomi viene eliminato o il ruolo non è più in uso, per liberare spazio per altre associazioni.

> **Note**
>
> Le associazioni persistenti potrebbero potenzialmente influire sulla tua capacità di scalabilità se non le elimini, poiché EKS ha dei limiti sul numero di associazioni che puoi creare (limite minimo: 1000 associazioni per cluster). Puoi elencare le associazioni di identità dei pod in un determinato namespace per verificare se ci sono associazioni persistenti che devono essere ripulite:

```
aws eks list-pod-identity-associations --cluster-name mycluster --namespace mynamespace
```

Con AWS CLI (versione 2.24.0 o successiva), esegui il seguente comando emr-containers per eliminare le associazioni di ruolo di EMR:

```
aws emr-containers delete-role-associations \
        --cluster-name mycluster \
        --namespace mynamespace \
        --role-name JobExecutionRoleIRSAv2
```

Migra automaticamente l'IRSA esistente su Pod Identity

Puoi utilizzare lo strumento eksctl per migrare gli IAM Roles for Service Accounts (IRSA) esistenti alle associazioni di identità pod:

```
eksctl utils migrate-to-pod-identity \
```

```
    --cluster mycluster \
    --remove-oidc-provider-trust-relationship \
    --approve
```

L'esecuzione del comando senza il `--approve` flag produrrà solo un piano che riflette le fasi della migrazione e non si verificherà alcuna migrazione effettiva.

risoluzione dei problemi

Il mio lavoro non è riuscito con NoClassDefinitionFound ClassNotFound Exception for Credentials Provider o non sono riuscito a ottenere il provider di credenziali.

EKS Pod Identity utilizza il Container Credentials Provider per recuperare le credenziali necessarie. Se hai specificato un provider di credenziali personalizzato, assicurati che funzioni correttamente. In alternativa, assicurati di utilizzare una versione AWS SDK corretta che supporti EKS Pod Identity. Per ulteriori informazioni, consulta la sezione Introduzione ad Amazon EKS.

Job non riuscito con l'errore «Impossibile recuperare le credenziali a causa di [x] Size Limit» visualizzato nel eks-pod-identity-agent registro.

EMR su EKS crea account di servizio Kubernetes in base al nome del ruolo di esecuzione del lavoro. Se il nome del ruolo è troppo lungo, EKS Auth non riuscirà a recuperare le credenziali perché la combinazione di e supera il limite di `cluster_name` lunghezza. `pod_name` `service_account_name` Identifica quale componente occupa più spazio e regola le dimensioni di conseguenza.

Job non riuscito con l'errore «Failed to Retrieve Credentials xxx» visualizzato nel eks-pod-identity registro.

Una possibile causa di questo problema potrebbe essere che il cluster EKS è configurato in sottoreti private senza una configurazione PrivateLink corretta per il cluster. Verifica se il cluster si trova in una rete privata e configuralo AWS PrivateLink per risolvere il problema. Per istruzioni dettagliate, consulta la sezione Guida introduttiva ad Amazon EKS.

Opzione 2: abilitare IAM Roles for Service Accounts (IRSA) sul cluster EKS

La funzionalità dei ruoli IAM per gli account di servizio è disponibile per i nuovi cluster Amazon EKS con Kubernetes versione 1.14 e successiva e per i cluster EKS aggiornati alle versioni 1.13 o successiva dopo il 3 settembre 2019 incluso. Per poter utilizzare questa funzionalità, puoi

aggiornare i cluster EKS esistenti alla versione 1.14 o successiva. Per ulteriori informazioni, consulta Aggiornamento di una versione Kubernetes del cluster Amazon EKS.

Se supporta i ruoli IAM per gli account di servizio, il cluster è associato a un URL dell'emittente OpenID Connect. Puoi visualizzare questo URL nella console Amazon EKS oppure puoi utilizzare il seguente AWS CLI comando per recuperarlo.

> **Important**
>
> È necessario utilizzare la versione più recente di AWS CLI per ricevere l'output corretto da questo comando.

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

L'output previsto è il seguente:

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

Per utilizzare i ruoli IAM per gli account di servizio nel cluster è necessario creare un provider di identità OIDC utilizzando eksctl o la Console di gestione AWS.

Creazione di un provider di identità IAM OIDC per il cluster con **`eksctl`**

Controlla la versione della `eksctl` con il comando seguente. Questa procedura presuppone che `eksctl` sia installato e che la versione di `eksctl` sia 0.32.0 o successiva.

```
eksctl version
```

Per ulteriori informazioni sull'installazione o sull'aggiornamento di eksctl, consulta Installazione o aggiornamento di eksctl.

Crea il provider di identità OIDC per il cluster con il comando seguente. Sostituire *`cluster_name`* con il proprio valore.

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

Per creare un provider di identità IAM OIDC per il cluster con Console di gestione AWS

Recupera l'URL dell'emittente OIDC dalla descrizione della console Amazon EKS del cluster o usa il seguente comando. AWS CLI

Utilizza il seguente comando per recuperare l'URL dell'emittente OIDC dalla AWS CLI.

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

Utilizza la seguente procedura per recuperare l'URL dell'emittente OIDC dalla console Amazon EKS.

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, scegli Identity Providers (Provider di identità), quindi seleziona Create Provider (Crea provider).
    1. Per Tipo di provider, seleziona Scegli un tipo di provider e quindi OpenID Connect.
    2. Per URL provider, incolla l'URL dell'emittente OIDC del cluster.
    3. Per Audience (Pubblico), digita sts.amazonaws.com e scegli Next Step (Fase successiva).
3. Verifica che le informazioni del provider siano corrette, quindi seleziona Crea per creare il provider di identità.

Creazione di un ruolo di esecuzione di processo

Per eseguire carichi di lavoro su Amazon EMR su EKS, occorre creare un ruolo IAM. Nella presente documentazione, tale ruolo viene definito ruolo di esecuzione di processo. Per ulteriori informazioni su come creare ruoli IAM, consulta Creazione di ruoli IAM nella Guida per l'utente IAM.

È necessario anche creare una policy IAM che specifichi le autorizzazioni per il ruolo di esecuzione di processo e, successivamente, collegare la policy IAM al ruolo di esecuzione di processo.

La seguente politica per il ruolo di esecuzione del lavoro consente l'accesso a destinazioni di risorse, Amazon S3 e. CloudWatch Queste autorizzazioni sono necessarie per monitorare i processi e i log di accesso. Per seguire la stessa procedura, utilizza: AWS CLI

Creazione del ruolo IAM per l'esecuzione del lavoro: creiamo il ruolo che EMR utilizzerà per l'esecuzione del lavoro. Questo è il ruolo che i lavori EMR assumeranno quando verranno eseguiti su EKS.

```
cat <<EoF > ~/environment/emr-trust-policy.json
 {
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Principal": {
         "Service": "elasticmapreduce.amazonaws.com"
       },
       "Action": "sts:AssumeRole"
     }
   ]
 }
 EoF

 aws iam create-role --role-name EMRContainers-JobExecutionRole --assume-role-policy-
document file://~/environment/emr-trust-policy.json
```

Successivamente, dobbiamo collegare le politiche IAM richieste al ruolo in modo che possa scrivere i log su s3 e cloudwatch.

```
cat <<EoF > ~/environment/EMRContainers-JobExecutionRole.json
 {
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": [
                 "s3:PutObject",
                 "s3:GetObject",
                 "s3:ListBucket"
             ],
             "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
         },
         {
             "Effect": "Allow",
             "Action": [
                 "logs:PutLogEvents",
                 "logs:CreateLogStream",
               "logs:DescribeLogGroups",
                 "logs:DescribeLogStreams"
             ],
```

```
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
EoF
aws iam put-role-policy --role-name EMRContainers-JobExecutionRole --policy-name
EMR-Containers-Job-Execution --policy-document file://~/environment/EMRContainers-
JobExecutionRole.json
```

> **Note**
>
> L'accesso deve essere compreso in modo appropriato, non concesso a tutti gli oggetti S3 nel ruolo di esecuzione di processo.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ],
      "Sid": "AllowS3Putobject"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogStream",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
```

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

Per ulteriori informazioni, consulta Usare i ruoli di esecuzione dei job, Configurare un job run per utilizzare i log di S3 e Configurare un job run per usare Logs. CloudWatch

Aggiornamento della policy di affidabilità del ruolo di esecuzione di processo

Quando si utilizzano ruoli IAM per gli account di servizio (IRSA) per eseguire processi in uno spazio dei nomi Kubernetes, un amministratore deve creare una relazione di fiducia tra il ruolo di esecuzione di processo e l'identità dell'account del servizio gestito da EMR. La relazione di fiducia può essere creata aggiornando la policy di affidabilità del ruolo di esecuzione di processo. Ricorda che l'account di servizio gestito da EMR viene creato in automatico all'invio del processo, nell'ambito dello spazio dei nomi in cui viene inviato il processo.

Per aggiornare la policy di affidabilità, esegui il comando seguente:

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

Per ulteriori informazioni, consulta Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS.

> **Important**
>
> L'operatore che esegue il comando precedente deve disporre delle seguenti autorizzazioni: `eks:DescribeCluster`, `iam:GetRole` e `iam:UpdateAssumeRolePolicy`.

Concessione dell'accesso ad Amazon EMR su EKS agli utenti

Per qualsiasi azione eseguita su Amazon EMR su EKS, occorre disporre di un'autorizzazione IAM corrispondente. Innanzitutto, dovrai creare una policy IAM che ti permetta di eseguire le azioni di Amazon EMR su EKS e di allegare la policy all'utente o al ruolo IAM che utilizzi.

In questo argomento vengono illustrate le fasi per la creazione di una nuova policy e il relativo collegamento a un utente. Inoltre, vengono elencate le autorizzazioni di base necessarie per configurare l'ambiente Amazon EMR su EKS. Consigliamo di perfezionare le autorizzazioni a risorse specifiche ove possibile in base alle esigenze aziendali.

Creazione di una nuova policy IAM e collegamento della stessa a un utente nella console IAM

Creazione di una nuova policy IAM

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione della console IAM seleziona Policy.
3. Nella pagina Policy, scegli Crea policy.
4. Nella finestra Crea policy, vai alla scheda Modifica JSON. Crea un documento di policy con una o più istruzioni JSON, come illustrato negli esempi che seguono questa procedura. Poi, scegli Verifica policy.
5. Nella schermata Verifica policy, inserisci il Nome policy, ad esempio `AmazonEMROnEKSPolicy`. Immetti una descrizione facoltativa e quindi seleziona Crea policy.

Collegamento della policy a un utente o ruolo

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione, seleziona Policy.
3. Nell'elenco di policy, seleziona la casella di spunta accanto alla policy creata nella sezione precedente. Puoi utilizzare il menu Filtro e la casella di ricerca per filtrare l'elenco di policy.
4. Scegli Operazioni di policy, quindi Collega.
5. Seleziona l'utente o il ruolo a cui desideri collegare la policy. Puoi usare il menu Filtro e la casella di ricerca per filtrare l'elenco delle entità principali. Dopo aver scelto l'utente o il ruolo a cui collegare la policy, scegli Collega policy.

Autorizzazioni per la gestione dei cluster virtuali

Per gestire i cluster virtuali nel tuo AWS account, crea una policy IAM con le seguenti autorizzazioni. Queste autorizzazioni ti consentono di creare, elencare, descrivere ed eliminare i cluster virtuali nel tuo account. AWS

JSON

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

Amazon EMR è integrato con Amazon EKS Cluster Access Management (CAM), quindi puoi automatizzare la configurazione delle policy AuthN e AuthZ necessarie per eseguire i job Amazon EMR Spark nei namespace dei cluster Amazon EKS. A tale scopo, devi disporre delle seguenti autorizzazioni:

```
{
  "Effect": "Allow",
  "Action": [
    "eks:CreateAccessEntry"
  ],
  "Resource":
 "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:cluster/<EKS_CLUSTER_NAME>"
},
{
  "Effect": "Allow",
  "Action": [
    "eks:DescribeAccessEntry",
    "eks:DeleteAccessEntry",
    "eks:ListAssociatedAccessPolicies",
    "eks:AssociateAccessPolicy",
    "eks:DisassociateAccessPolicy"
  ],
  "Resource": "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:access-
entry/<EKS_CLUSTER_NAME>/role/<AWS_ACCOUNT_ID>/AWSServiceRoleForAmazonEMRContainers/*"
}
```

Per ulteriori informazioni, consulta Automatizzare l'abilitazione dell'accesso ai cluster per Amazon EMR su EKS.

Quando l'`CreateVirtualCluster`operazione viene richiamata per la prima volta da un AWS account, sono necessarie anche le `CreateServiceLinkedRole` autorizzazioni per creare il ruolo collegato al servizio per Amazon EMR su EKS. Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Amazon EMR su EKS.

Autorizzazioni per inviare i processi

Per inviare lavori sui cluster virtuali del tuo AWS account, crea una policy IAM con le seguenti autorizzazioni. Queste autorizzazioni ti consentono di avviare, elencare, descrivere ed annullare esecuzioni di processo per tutti i cluster virtuali nel tuo account. È consigliabile aggiungere autorizzazioni per elencare o descrivere cluster virtuali, il che ti consente di controllare lo stato del cluster virtuale prima di inviare i processi.

JSON

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

Autorizzazioni per il debug e il monitoraggio

Per accedere ai log inviati ad Amazon S3 CloudWatch o per visualizzare i registri degli eventi delle applicazioni nella console Amazon EMR, crea una policy IAM con le seguenti autorizzazioni. Consigliamo di perfezionare le autorizzazioni a risorse specifiche quando possibile in base alle esigenze aziendali.

> **Important**
>
> Se non hai creato un bucket Amazon S3, dovrai aggiungere l'autorizzazione `s3:CreateBucket` per la dichiarazione di policy. Se non hai creato un gruppo di log, dovrai aggiungere `logs:CreateLogGroup` alla dichiarazione di policy.

JSON

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeJobRun",
        "elasticmapreduce:CreatePersistentAppUI",
        "elasticmapreduce:DescribePersistentAppUI",
        "elasticmapreduce:GetPersistentAppUIPresignedURL"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEMRCONTAINERSDescribejobrun"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowS3Getobject"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:Get*",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowLOGSGet"
    }
  ]
}
```

Per ulteriori informazioni su come configurare l'esecuzione di un job per inviare i log ad Amazon S3 CloudWatch e, consulta Configurare un job run per utilizzare i log S3 e Configurare un job run to use Logs. CloudWatch

Registrazione del cluster Amazon EKS con Amazon EMR

La registrazione del cluster è l'ultimo passaggio necessario per configurare Amazon EMR su EKS per l'esecuzione dei carichi di lavoro.

Utilizza il comando seguente per creare un cluster virtuale con un nome a scelta per lo spazio dei nomi e il cluster Amazon EKS impostati nelle fasi precedenti.

> **Note**
>
> Ogni cluster virtuale deve avere un nome univoco in tutti i cluster EKS. Se due cluster virtuali hanno lo stesso nome, il processo di implementazione non riuscirà anche se i due cluster virtuali appartengono a cluster EKS diversi.

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

In alternativa, puoi creare un file JSON che includa i parametri richiesti per il cluster virtuale e, successivamente, eseguire il comando `create-virtual-cluster` con il percorso del file JSON. Per ulteriori informazioni, consulta Gestione dei cluster virtuali.

> **Note**
>
> Per convalidare la corretta creazione di un cluster virtuale, visualizza lo stato dei cluster virtuali utilizzando l'operazione `list-virtual-clusters` o andando alla pagina Virtual Clusters (Cluster virtuali) nella console di Amazon EMR.

Invio di un'esecuzione di processo con `StartJobRun`

Invio di un'esecuzione di processo mediante un file JSON con parametri specificati

1. Crea un file `start-job-run-request.json` e specifica i parametri richiesti per l'esecuzione di processo, come illustrato nel file JSON di esempio seguente. Per ulteriori informazioni sui parametri, consulta Opzioni per la configurazione di un'esecuzione di processo.

```
{
  "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. Utilizza il comando `start-job-run` con un percorso per il file `start-job-run-request.json` archiviato localmente.

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

Avvio di un'esecuzione di processo con il comando **`start-job-run`**

1. Fornisci tutti i parametri specificati nel comando `StartJobRun`, come illustrato nell'esempio seguente.

```
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. Per Spark SQL, fornisci tutti i parametri specificati nel comando `StartJobRun`, come illustrato nell'esempio seguente.

```
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" }}}'
```

Uso della classificazione del mittente di processi

Panoramica

La richiesta `StartJobRun` di Amazon EMR su EKS crea un pod del mittente di processi (noto anche come pod job-runner) per generare il driver Spark. È possibile utilizzare la `emr-job-submitter` classificazione per configurare i selettori di nodi per il pod Job Submitter, nonché impostare l'immagine, la CPU e la memoria per il contenitore di registrazione del pod Job Submitter.

Le seguenti impostazioni sono disponibili nella classificazione: `emr-job-submitter`

`jobsubmitter.node.selector.[labelKey]`

Si aggiunge al selettore di nodi del pod del mittente di processi, con chiave *`labelKey`* e il valore come valore di configurazione per la configurazione. Ad esempio, è possibile impostare `jobsubmitter.node.selector.identifier` su `myIdentifier` e il pod del mittente di processi avrà un selettore di nodi con un valore identificativo chiave di `myIdentifier`. Questo può essere usato per specificare su quali nodi può essere posizionato il pod Job Submitter. Per aggiungere più chiavi di selettore di nodi, imposta più configurazioni con questo prefisso.

`jobsubmitter.logging.image`

Imposta un'immagine personalizzata da utilizzare per il contenitore di registrazione sul pod job submitter.

`jobsubmitter.logging.request.cores`

Imposta un valore personalizzato per il numero di CPUs, in unità CPU, per il contenitore di registrazione sul pod job submitter. Per impostazione predefinita, questo valore è impostato su 100 m.

`jobsubmitter.logging.request.memory`

Imposta un valore personalizzato per la quantità di memoria, in byte, per il contenitore di registrazione sul pod job submitter. Per impostazione predefinita, è impostato su 200 Mi. Un mebibyte è un'unità di misura simile a un megabyte.

Consigliamo di posizionare i job submitter pod sulle istanze On-Demand. L'inserimento dei pod Job Submitter sulle istanze Spot potrebbe causare un errore del lavoro se l'istanza su cui viene eseguito il pod Job Submitter è soggetta a un'interruzione dell'istanza Spot. Puoi anche posizionare il pod del mittente di processi in un'unica zona di disponibilità o utilizzare qualsiasi etichetta Kubernetes applicata ai nodi.

Esempi di classificazione del mittente di processi

In questa sezione

- Richiesta StartJobRun con posizionamento dei nodi on demand per il pod del mittente di processi
- Richiesta StartJobRun con posizionamento dei nodi su AZ singola per il pod del mittente di processi
- StartJobRunrichiesta con posizionamento del tipo di EC2 istanza Single-AZ e Amazon per il pod Job Submitter
- StartJobRunrichiesta con immagine, CPU e memoria del contenitore di registrazione personalizzate
- StartJobRunrichiesta con immagine del contenitore personalizzata

Richiesta **`StartJobRun`** con posizionamento dei nodi on demand per il pod del mittente di processi

```
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
```

Richiesta **`StartJobRun`** con posizionamento dei nodi su AZ singola per il pod del mittente di processi

```
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`richiesta con posizionamento del tipo di EC2 istanza Single-AZ e Amazon per il pod Job Submitter

```
{
  "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"
      }
    }
```

```
  }
}
```

`StartJobRun`richiesta con immagine, CPU e memoria del contenitore di registrazione personalizzate

```
{
  "name": "spark-python",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.logging.image": "YOUR_ECR_IMAGE_URL",
            "jobsubmitter.logging.request.memory": "200Mi",
            "jobsubmitter.logging.request.cores": "0.5"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

`StartJobRun`richiesta con immagine del contenitore personalizzata

```
cat >spark-python-in-s3-nodeselector-custom-container-image-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.container.image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/emr6.11_custom_repo",
            "jobsubmitter.container.image.pullPolicy": "kubernetes pull policy"
        }
      }
    ],
    "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-custom-container-image-job-submitter.json
```

Utilizzo della classificazione delle impostazioni predefinite dei container Amazon EMR

Panoramica

Le seguenti impostazioni sono disponibili nella classificazione: `emr-containers-defaults`

`job-start-timeout`

Per impostazione predefinita, un processo scade se non può essere avviato e rimane nello `SUBMITTED` stato per 15 minuti. Questa configurazione modifica il numero di secondi di attesa prima del timeout del processo.

`logging.image`

Imposta un'immagine personalizzata da utilizzare per il contenitore di registrazione sui pod driver ed executor.

`logging.request.cores`

Imposta un valore personalizzato per il numero di CPUs, in unità CPU, per il contenitore di registrazione sui driver e sui pod dell'executor. Per impostazione predefinita, questo valore non è impostato.

`logging.request.memory`

Imposta un valore personalizzato per la quantità di memoria, in byte, per il contenitore di registrazione sui pod driver e executor. Per impostazione predefinita, questo valore è impostato su 512 Mi. Un mebibyte è un'unità di misura simile a un megabyte.

Esempi di classificazione del mittente di processi

In questa sezione

`StartJobRun`richiesta con timeout di lavoro personalizzato

```
{
```

```
  "name": "spark-python",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emr-containers-defaults",
        "properties": {
            "job-start-timeout": "1800"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

`StartJobRun`richiesta con immagine, CPU e memoria del contenitore di registrazione personalizzate

```
{
  "name": "spark-python",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
```

```
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emr-containers-defaults",
        "properties": {
            "logging.image": "YOUR_ECR_IMAGE_URL",
            "logging.request.memory": "200Mi",
            "logging.request.cores": "0.5"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

Esecuzione dei processi Spark con l'operatore Spark

I rilasci 6.10.0 o successivi di Amazon EMR supportano l'operatore Kubernetes per Apache Spark, ovvero l'operatore Spark, come modello di invio dei processi per Amazon EMR su EKS. Con l'operatore Spark, puoi implementare e gestire applicazioni Spark con il runtime di rilascio Amazon EMR sui tuoi cluster Amazon EKS personali. Una volta implementato l'operatore Spark nel cluster Amazon EKS, puoi inviare direttamente le applicazioni Spark con l'operatore. L'operatore gestisce il ciclo di vita delle applicazioni Spark.

> **Note**
>
> Amazon EMR calcola i prezzi di Amazon EKS in base al consumo di vCPU e memoria. Questo calcolo si applica ai driver e ai pod executor. Questo calcolo inizia dal momento in cui scarichi l'immagine dell'applicazione Amazon EMR fino alla chiusura del pod Amazon EKS e viene arrotondato al secondo più vicino.

Argomenti

Configurazione dell'operatore Spark per Amazon EMR su EKS

Completa le seguenti attività per definire la configurazione prima di installare l'operatore Spark su Amazon EKS. Se hai già effettuato la registrazione ad Amazon Web Services (AWS) e hai utilizzato Amazon EKS, ti manca poco per cominciare a utilizzare Amazon EMR su EKS. Completa le seguenti attività per definire la configurazione per l'operatore Spark su Amazon EKS. Se hai già completato uno dei prerequisiti, puoi saltarli e passare a quello successivo.

- Installa o aggiorna alla versione più recente di AWS CLI: se hai già installato il AWS CLI, conferma di disporre della versione più recente.
- Configura kubectl ed eksctl: eksctl è uno strumento da riga di comando che usi per comunicare con Amazon EKS.
- Installa Helm: il programma di gestione del pacchetto Helm per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes.
- Inizia a usare Amazon EKS — eksctl — Segui i passaggi per creare un nuovo cluster Kubernetes con nodi in Amazon EKS.
- Seleziona un URI dell'immagine di base Amazon EMR (rilascio 6.10.0 o successivo): l'operatore Spark è supportato con i rilasci 6.10.0 e successivi di Amazon EMR.

Nozioni di base sull'operatore Spark per Amazon EMR su EKS

Questo argomento offre assistenza per cominciare a utilizzare l'operatore Spark su Amazon EKS implementando un'applicazione Spark e un'applicazione Schedule Spark.

Installazione dell'operatore Spark

Utilizza la procedura seguente per installare l'operatore Kubernetes per Apache Spark.

1. Se non lo hai già fatto, completa le fasi in Configurazione dell'operatore Spark per Amazon EMR su EKS.
2. Autentica il client Helm nel registro Amazon ECR. Nel comando seguente, sostituisci *region-id* i valori con i tuoi preferiti Regione AWS e il *ECR-registry-account* valore corrispondente per la regione dalla pagina. Account di registro Amazon ECR per Regione

```
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. Installa l'operatore Spark con il comando seguente.

   Per il parametro `--version` del grafico Helm, utilizza l'etichetta di rilascio di Amazon EMR rimuovendo il prefisso `emr-` e il suffisso della data. Ad esempio, con il rilascio `emr-6.12.0-java17-latest`, specifica `6.12.0-java17`. L'esempio nel comando seguente utilizza il rilascio `emr-7.12.0-latest`, quindi specifica `7.12.0` per il grafico Helm `--version`.

```
helm install spark-operator-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/spark-operator \
  --set emrContainers.awsRegion=region-id \
  --version 7.12.0 \
  --namespace spark-operator \
  --create-namespace
```

   Per impostazione predefinita, il comando crea un account di servizio `emr-containers-sa-spark-operator` per l'operatore Spark. Per utilizzare un account di servizio diverso, fornisci l'argomento `serviceAccounts.sparkoperator.name`. Esempio:

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

   Se desideri utilizzare il dimensionamento automatico verticale con l'operatore Spark aggiungi la seguente riga al comando di installazione per consentire i webhook per l'operatore:

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

4. Verifica di aver installato il grafico Helm con il comando `helm list`:

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

Il comando `helm list` dovrebbe restituire le informazioni sul rilascio del grafico Helm appena implementato:

```
app_version: v1beta2-1.3.8-3.1.1
chart: spark-operator-7.12.0
name: spark-operator-demo
namespace: spark-operator
revision: "1"
status: deployed
updated: 2023-03-14 18:20:02.721638196 +0000 UTC
```

5. Completa l'installazione con tutte le opzioni aggiuntive necessarie. Per ulteriori informazioni, consulta la `spark-on-k8s-operator`documentazione su. GitHub

Esecuzione di un'applicazione Spark

L'operatore Spark è supportato con Amazon EMR 6.10.0 o versioni successive. Quando installi l'operatore Spark, viene creato per impostazione predefinita l'account di servizio `emr-containers-sa-spark` per eseguire le applicazioni Spark. Completa le fasi seguenti per eseguire un'applicazione Spark con l'operatore Spark in Amazon EMR su EKS 6.10.0 o versioni successive.

1. Prima di poter eseguire un'applicazione Spark con l'operatore Spark, completa le fasi indicate in Configurazione dell'operatore Spark per Amazon EMR su EKS e Installazione dell'operatore Spark.
2. Crea un file `spark-pi.yaml` di definizione `SparkApplication` con il seguente contenuto di esempio:

```
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. A questo punto, invia l'applicazione Spark con il comando seguente. L'operazione creerà anche un oggetto `SparkApplication` denominato `spark-pi`:

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

4. Controlla gli eventi dell'oggetto `SparkApplication` con il comando seguente:

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

Per maggiori informazioni sull'invio di applicazioni a Spark tramite l'operatore Spark, consulta [Using a nella documentazione su `SparkApplication`](). `spark-on-k8s-operator` GitHub

Usa Amazon S3 per lo storage

Per utilizzare Amazon S3 come opzione di archiviazione dei file, aggiungi le seguenti configurazioni al tuo file YAML.

```
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
```

Se utilizzi le versioni 7.2.0 e successive di Amazon EMR, le configurazioni sono incluse per impostazione predefinita. In tal caso, puoi impostare il percorso del file `s3://<bucket_name>/<file_path>` anziché `local://<file_path>` nel file YAML dell'applicazione Spark.

Quindi invia l'applicazione Spark come di consueto.

Usa la scalabilità automatica verticale con l'operatore Spark per Amazon EMR su EKS

A partire da Amazon EMR 7.0, puoi utilizzare Amazon EMR su scalabilità automatica verticale EKS per semplificare la gestione delle risorse. Ottimizza in automatico le risorse di memoria e CPU per adattarle alle esigenze del carico di lavoro fornito per le applicazioni Spark di Amazon EMR. Per ulteriori informazioni, consulta Uso del dimensionamento automatico verticale con i processi Spark di Amazon EMR.

Questa sezione descrive come configurare l'operatore Spark per utilizzare il dimensionamento automatico verticale.

Prerequisiti

Prima di configurare il monitoraggio, assicurati di completare le seguenti attività di configurazione:

- Completa le fasi descritte in Configurazione dell'operatore Spark per Amazon EMR su EKS.
- (opzionale) Se in precedenza hai installato una versione precedente dell'operatore Spark, elimina il SparkApplication/ScheduledSparkApplication CRD.

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

- Completa le fasi descritte in Installazione dell'operatore Spark. Nel passaggio 3, aggiungi la seguente riga al comando di installazione per consentire l'uso dei webhook all'operatore:

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

- Completa le fasi descritte in Configurazione del dimensionamento automatico verticale per Amazon EMR su EKS.
- Consenti l'accesso ai file nella tua posizione Amazon S3:

  1. Annota il tuo account di servizio per conducente e operatore con un account `JobExecutionRole` che dispone delle autorizzazioni S3.

```
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. Aggiorna la politica di fiducia del tuo ruolo di esecuzione del lavoro in quel namespace.

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

3. Modifica la policy di fiducia dei ruoli IAM del tuo ruolo di esecuzione del lavoro e aggiorna il form `serviceaccount` to`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. Se utilizzi Amazon S3 come archivio di file, aggiungi le seguenti impostazioni predefinite al tuo file yaml.

```
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
```

Esegui un processo con dimensionamento automatico verticale sull'operatore Spark

Prima di poter eseguire un'applicazione Spark con l'operatore Spark, devi completare i passaggi indicati in Prerequisiti.

Per utilizzare la scalabilità automatica verticale con l'operatore Spark, aggiungi la seguente configurazione al driver per le specifiche dell'applicazione Spark per attivare la scalabilità automatica verticale:

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

Questa configurazione consente la scalabilità automatica verticale ed è una configurazione di firma obbligatoria che ti consente di scegliere una firma per il tuo lavoro.

Per ulteriori informazioni sulle configurazioni e sui valori dei parametri, consulta Configurazione della scalabilità automatica verticale per Amazon EMR su EKS. Per impostazione predefinita, il processo viene inviato nella modalità Disattivato di solo monitoraggio del dimensionamento automatico verticale. Questo stato di monitoraggio consente di calcolare e visualizzare consigli sulle risorse

senza eseguire il dimensionamento automatico. Per ulteriori informazioni, consulta Modalità di scalabilità automatica verticale.

Di seguito è riportato un esempio di file di `SparkApplication` definizione denominato `spark-pi.yaml` con le configurazioni richieste per utilizzare la scalabilità automatica verticale.

```
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-7.12.0: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"
```

A questo punto, invia l'applicazione Spark con il comando seguente. L'operazione creerà anche un oggetto `SparkApplication` denominato `spark-pi`:

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

Per maggiori informazioni sull'invio di applicazioni a Spark tramite l'operatore Spark, consulta Using a `SparkApplication` nella documentazione su. `spark-on-k8s-operator` GitHub

Verifica della funzionalità di dimensionamento automatico verticale

Per verificare che il dimensionamento automatico verticale funzioni correttamente per il processo inviato, utilizza kubectl per ottenere la risorsa personalizzata `verticalpodautoscaler` e visualizzare i consigli di dimensionamento.

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

L'output della query deve assomigliare al seguente:

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

Se l'output non è simile o contiene un codice di errore, consulta la procedura indicata in Risoluzione dei problemi relativi al dimensionamento automatico verticale di Amazon EMR su EKS per risolvere il problema.

Per rimuovere i pod e le applicazioni, esegui il seguente comando:

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

Disinstallazione dell'operatore Spark per Amazon EMR su EKS

Utilizza le fasi seguenti per disinstallare l'operatore Spark.

1. Elimina l'operatore Spark utilizzando lo spazio dei nomi corretto. In questo esempio lo spazio dei nomi è `spark-operator-demo`.

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

2. Elimina l'account di servizio dell'operatore Spark:

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

3. Eliminate l'operatore Spark `CustomResourceDefinitions` ()CRDs:

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

Utilizzo della configurazione di monitoraggio per monitorare l'operatore Spark Kubernetes e i job Spark

La configurazione di monitoraggio consente di configurare facilmente l'archiviazione dei log dell'applicazione Spark e dei log degli operatori su Amazon S3 o su. Amazon CloudWatch Puoi sceglierne uno o entrambi. In questo modo viene aggiunto un sidecar log Agent ai pod Spark Operator, Driver ed Executor e successivamente inoltra i log di questi componenti ai sink configurati.

Prerequisiti

Prima di configurare il monitoraggio, assicurati di completare le seguenti attività di configurazione:

1. (Facoltativo) Se in precedenza hai installato una versione precedente dell'operatore Spark, elimina il SparkApplicationScheduledSparkApplication/CRD.

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

2. Crea un ruolo di operator/job esecuzione in IAM se non ne hai già uno.
3. Esegui il comando seguente per aggiornare la politica di fiducia del ruolo di operator/job esecuzione che hai appena creato:

```
aws emr-containers update-role-trust-policy \
--cluster-name cluster \
--namespace namespace \
--role-name iam_role_name_for_operator/job_execution_role
```

4. Modifica la policy di fiducia dei ruoli IAM del tuo ruolo di operator/job esecuzione nel modo seguente:

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

5. Crea una policy di MonitoringConfiguration in IAM con le seguenti autorizzazioni:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams",
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:log_group_name",
        "arn:aws:logs:*:*:log-group:log_group_name:*"
      ],
```

```
      "Sid": "AllowLOGSDescribelogstreams"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::bucket_name",
        "arn:aws:s3:::bucket_name/*"
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}
```

6. Allega la politica di cui sopra al tuo ruolo di esecuzione operator/job .

Registri degli operatori Spark

Puoi definire la configurazione del monitoraggio nel modo seguente quando esegui: `helm install`

```
helm install spark-operator spark-operator \
--namespace namespace \
--set emrContainers.awsRegion=aws_region \
--set emrContainers.monitoringConfiguration.image=log_agent_image_url \
--set
 emrContainers.monitoringConfiguration.s3MonitoringConfiguration.logUri=S3_bucket_uri \
--set
 emrContainers.monitoringConfiguration.cloudWatchMonitoringConfiguration.logGroupName=log_group
 \
```

```
--set
 emrContainers.monitoringConfiguration.cloudWatchMonitoringConfiguration.logStreamNamePrefix=lo
 \
--set emrContainers.monitoringConfiguration.sideCarResources.limits.cpuLimit=500m \
--set emrContainers.monitoringConfiguration.sideCarResources.limits.memoryLimit=512Mi \
--set
 emrContainers.monitoringConfiguration.containerLogRotationConfiguration.rotationSize=2GB
 \
--set
 emrContainers.monitoringConfiguration.containerLogRotationConfiguration.maxFilesToKeep=10
 \
--set webhook.enable=true \
--set emrContainers.operatorExecutionRoleArn=operator_execution_role_arn
```

Configurazione del monitoraggio

Di seguito sono riportate le opzioni di configurazione disponibili in MonitoringConfiguration.

- Immagine (opzionale): URL dell'immagine dell'agente di registro. Verrà recuperato entro emrReleaseLabel se non fornito.
- s3 MonitoringConfiguration — Imposta questa opzione per l'archiviazione su Amazon S3.
  - logURI — (obbligatorio) — Il percorso del bucket Amazon S3 in cui desideri archiviare i log.
  - Di seguito sono riportati alcuni esempi di formati per i percorsi dei bucket di Amazon S3, dopo il caricamento dei log. Il primo esempio mostra che la rotazione dei log non è abilitata.

    ```
    s3://${logUri}/${POD NAME}/operator/stdout.gz
    s3://${logUri}/${POD NAME}/operator/stderr.gz
    ```

    La rotazione del registro è abilitata per impostazione predefinita. È possibile visualizzare sia un file ruotato, con un indice incrementale, sia un file corrente, che è lo stesso dell'esempio precedente.

    ```
    s3://${logUri}/${POD NAME}/operator/stdout_YYYYMMDD_index.gz
    s3://${logUri}/${POD NAME}/operator/stderr_YYYYMMDD_index.gz
    ```
- cloudWatchMonitoringConfigurazione: la chiave di configurazione a cui impostare l'inoltro. Amazon CloudWatch
  - logGroupName(obbligatorio): nome del gruppo di Amazon CloudWatch log a cui si desidera inviare i log. Se non esiste, il gruppo viene creato in automatico.

  - logStreamNamePrefisso (opzionale): nome del flusso di log a cui si desidera inviare i log. Il valore predefinito è una stringa vuota. Il formato in Amazon CloudWatch è il seguente:

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

- sideCarResources(opzionale) — La chiave di configurazione per impostare i limiti delle risorse sul contenitore sidecar Fluentd lanciato.
  - memoryLimit (opzionale): il limite di memoria. Regola secondo necessità. Il valore predefinito è 512Mi.
  - CPULimit (opzionale) — Il limite della CPU. Regola secondo necessità. L'impostazione predefinita è 500 m.
- containerLogRotationConfigurazione (opzionale): controlla il comportamento di rotazione del registro del contenitore. È abilitato per impostazione predefinita.
  - rotationSize (obbligatorio): specifica la dimensione del file per la rotazione del registro. L'intervallo di valori possibili è compreso tra 2 KB e 2 GB. La parte relativa all'unità numerica del parametro rotationSize viene trasmessa come numero intero. Poiché i valori decimali non sono supportati, puoi specificare una dimensione di rotazione di 1,5 GB, ad esempio, con il valore 1.500 MB. Il valore predefinito è 2 GB.
  - maxFilesToKeep (obbligatorio): specifica il numero massimo di file da conservare nel contenitore dopo la rotazione. Il valore minimo è 1, quello massimo è 50. Il valore predefinito è 10.

Dopo aver configurato MonitoringConfiguration, dovresti essere in grado di controllare i log dei pod dell'operatore Spark su un bucket Amazon S3 o su entrambi. Amazon CloudWatch Per un bucket Amazon S3, devi attendere 2 minuti prima che il primo file di log venga svuotato.

Per trovare i log in Amazon CloudWatch, puoi accedere a quanto segue: CloudWatch> Gruppi di log > > /operator/stderr ***Log group namePod name***

Oppure puoi accedere a: > Gruppi di log > > /operator/stdout CloudWatch***Log group namePod name***

Registri delle applicazioni Spark

Puoi definire questa configurazione nel modo seguente.

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
```

```
  name: spark-pi
  namespace: namespace
spec:
  type: Scala
  mode: cluster
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  emrReleaseLabel: emr_release_label
  executionRoleArn: job_execution_role_arn
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  monitoringConfiguration:
    image: "log_agent_image"
    s3MonitoringConfiguration:
      logUri: "S3_bucket_uri"
    cloudWatchMonitoringConfiguration:
      logGroupName: "log_group_name"
      logStreamNamePrefix: "log_stream_prefix"
    sideCarResources:
```

```
      limits:
        cpuLimit: "500m"
        memoryLimit: "250Mi"
    containerLogRotationConfiguration:
      rotationSize: "2GB"
      maxFilesToKeep: "10"
```

Di seguito sono riportate le opzioni di configurazione disponibili in MonitoringConfiguration.

- Immagine (opzionale): URL dell'immagine dell'agente di registro. Verrà recuperato entro emrReleaseLabel se non fornito.
- s3 MonitoringConfiguration — Imposta questa opzione per l'archiviazione su Amazon S3.
  - logURI (obbligatorio): il percorso del bucket Amazon S3 in cui desideri archiviare i log. Il primo esempio mostra che la rotazione dei log non è abilitata:

    ```
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stdout.gz
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stderr.gz
    ```

    La rotazione dei log è abilitata per impostazione predefinita. È possibile utilizzare sia un file ruotato (con indice incrementale) sia un file corrente (uno senza il timbro della data).

    ```
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/
    stdout_YYYYMMDD_index.gz
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/
    stderr_YYYYMMDD_index.gz
    ```

- cloudWatchMonitoringConfigurazione: la chiave di configurazione a cui impostare l'inoltro. Amazon CloudWatch
  - logGroupName(obbligatorio) — Il nome del gruppo di log di Cloudwatch a cui desideri inviare i log. Il gruppo viene creato automaticamente se non esiste.
  - logStreamNamePrefisso (opzionale): il nome del flusso di log a cui si desidera inviare i log. Il valore predefinito è una stringa vuota. Il formato in CloudWatch è il seguente:

    ```
    ${logStreamNamePrefix}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stdout
    ${logStreamNamePrefix}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stderr
    ```

- sideCarResources(opzionale) — La chiave di configurazione per impostare i limiti delle risorse sul contenitore sidecar Fluentd lanciato.

  - memoryLimit (opzionale): il limite di memoria. Regola secondo necessità. L'impostazione predefinita è 250 Mi.
  - CPULimit: il limite della CPU. Regola secondo necessità. L'impostazione predefinita è 500 m.
- containerLogRotationConfigurazione (opzionale): controlla il comportamento di rotazione del registro del contenitore. È abilitato per impostazione predefinita.
  - rotationSize (obbligatorio): specifica la dimensione del file per la rotazione del registro. L'intervallo di valori possibili è compreso tra 2 KB e 2 GB. La parte relativa all'unità numerica del parametro rotationSize viene trasmessa come numero intero. Poiché i valori decimali non sono supportati, puoi specificare una dimensione di rotazione di 1,5 GB, ad esempio, con il valore 1.500 MB. Il valore predefinito è 2 GB.
  - maxFilesToKeep (obbligatorio): specifica il numero massimo di file da conservare nel contenitore dopo la rotazione. Il valore minimo è 1. Il valore massimo è 50. Il valore predefinito è 10.

Dopo aver configurato MonitoringConfiguration, dovresti essere in grado di controllare i log del driver dell'applicazione Spark e dell'executor su un bucket Amazon S3 o su entrambi. CloudWatch Per un bucket Amazon S3, devi attendere 2 minuti prima che il primo file di log venga svuotato. Ad esempio, in Amazon S3, il percorso del bucket appare come segue:

Amazon S3 > Benne > > ***Bucket name***> stderr.gz ***Spark application name - UUID Pod Name***

O:

Amazon S3 > Benne > > ***Bucket name***> stdout.gz ***Spark application name - UUID Pod Name***

In CloudWatch, il percorso appare come segue:

CloudWatch> Gruppi di log > ***Log group name***>*Spark application name - UUID*/*Pod name*/stderr

O:

CloudWatch> Gruppi di log > >/***Log group name***/stdout ***Spark application name - UUID Pod name***

Sicurezza e operatore Spark con Amazon EMR su EKS

Esistono due modi per configurare le autorizzazioni di accesso al cluster quando si utilizza l'operatore Spark. Il primo consiste nell'utilizzare il controllo degli accessi basato sui ruoli, il controllo degli accessi basato sui ruoli (RBAC) limita l'accesso in base al ruolo di una persona all'interno di un'organizzazione. È diventato il modo principale per gestire l'accesso. Il secondo metodo di accesso consiste nell'assumere un AWS Identity and Access Management ruolo, che fornisce l'accesso alle risorse mediante autorizzazioni specifiche assegnate.

Argomenti

Configurazione delle autorizzazioni di accesso al cluster con controllo degli accessi basato sui ruoli (role-based access control, RBAC)

Per implementare l'operatore Spark, Amazon EMR su EKS crea due ruoli e account di servizio per l'operatore Spark e le app Spark.

Argomenti

Account e ruolo di servizio dell'operatore

Amazon EMR su EKS crea l'account e il ruolo di servizio dell'operatore per gestire `SparkApplications` per i lavori Spark e per altre risorse come i servizi.

Il nome predefinito per questo account di servizio è `emr-containers-sa-spark-operator`.

Per questo ruolo di servizio vigono le regole elencate di seguito:

```
 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 }}
```

Account e ruolo di servizio Spark

Un pod del driver Spark necessita di un account di servizio Kubernetes nello stesso spazio dei nomi del pod. Questo account di servizio necessita delle autorizzazioni per creare, ottenere, elencare, applicare patch ai ed eliminare i pod dell'executor e per creare un servizio headless Kubernetes per il driver. Il driver si interrompe e si chiude senza l'account di servizio a meno che l'account di servizio predefinito nello spazio dei nomi del pod disponga delle autorizzazioni richieste.

Il nome predefinito per questo account di servizio è `emr-containers-sa-spark`.

Per questo ruolo di servizio vigono le regole elencate di seguito:

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

Configurazione delle autorizzazioni di accesso al cluster con ruoli IAM per gli account di servizio (IAM roles for service account, IRSA)

Questa sezione utilizza un esempio per dimostrare come configurare un account di servizio Kubernetes per assumere un ruolo. AWS Identity and Access Management I pod che utilizzano l'account di servizio possono quindi accedere a qualsiasi AWS servizio a cui il ruolo dispone delle autorizzazioni per accedere.

L'esempio seguente esegue un'applicazione Spark per contare le parole da un file in Amazon S3. A tale scopo, puoi configurare i ruoli IAM per gli account di servizio (IRSA) in modo da autenticare e autorizzare gli account di servizio Kubernetes.

> **Note**
>
> Questo esempio utilizza lo spazio dei nomi "spark-operator" per l'operatore Spark e per lo spazio dei nomi in cui invii l'applicazione Spark.

Prerequisiti

Prima di provare l'esempio in questa pagina, completa i prerequisiti descritti di seguito:

- Esegui la configurazione per l'operatore Spark.
- Installazione dell'operatore Spark.
- Crea un bucket Amazon S3.
- Salva la tua poesia preferita in un file di testo denominato `poem.txt` e carica il file nel bucket S3. L'applicazione Spark che crei in questa pagina leggerà il contenuto del file di testo. Per ulteriori informazioni, consulta Carica un oggetto nel tuo bucket S3 nella Guida per l'utente di Amazon Simple Storage Service.

Configurazione di un account di servizio Kubernetes per l'assunzione di un ruolo IAM

Utilizza i seguenti passaggi per configurare un account di servizio Kubernetes in modo che assuma un ruolo IAM che i pod possono utilizzare per accedere ai AWS servizi a cui il ruolo dispone delle autorizzazioni di accesso.

1. Dopo aver completato ilPrerequisiti, usa AWS Command Line Interface per creare un `example-policy.json` file che consenta l'accesso in sola lettura al file che hai caricato su Amazon S3:

```
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. Successivamente, crea una policy IAM `example-policy`:

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

3. Quindi, crea un ruolo IAM `example-role` e associalo a un account di servizio Kubernetes per il driver Spark:

```
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. Crea un file yaml con le associazioni di ruoli del cluster necessarie per l'account di servizio del driver Spark:

```
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. Applica le configurazioni delle associazioni di ruoli del cluster:

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

Il comando kubectl dovrebbe confermare la corretta creazione dell'account:

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

Esecuzione di un'applicazione dall'operatore Spark

Dopo aver configurato l'account di servizio Kubernetes, puoi eseguire un'applicazione Spark che conti il numero di parole nel file di testo che hai caricato come parte di Prerequisiti.

1. Crea un nuovo file`word-count.yaml`, con una `SparkApplication` definizione per la tua applicazione di conteggio parole, basato sulla versione 6 di Amazon EMR.

```
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
```

Se utilizzi l'operatore spark con una versione 7, modifichi alcuni valori di configurazione:

```
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-7.7.0: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/aws-java-sdk-v2/*:/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/aws-java-sdk-v2/*:/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. Invia l'applicazione Spark.

```
kubectl apply -f word-count.yaml
```

Il comando kubectl dovrebbe restituire la conferma che hai creato correttamente un oggetto `SparkApplication` denominato `word-count`.

```
sparkapplication.sparkoperator.k8s.io/word-count configured
```

3. Per controllare gli eventi dell'oggetto `SparkApplication`, esegui il comando seguente:

```
kubectl describe sparkapplication word-count -n spark-operator
```

Il comando kubectl dovrebbe restituire la descrizione di `SparkApplication` con gli eventi:

```
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
```

Ora l'applicazione sta contando le parole nel tuo file S3. Per trovare il numero di parole, fai riferimento ai file di log del driver:

```
kubectl logs pod/word-count-driver -n spark-operator
```

Il comando kubectl dovrebbe restituire il contenuto del file di log con i risultati dell'applicazione di conteggio parole.

```
INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:53, took 5.146519 s
                Software: 1
```

Per maggiori informazioni su come inviare le applicazioni a Spark tramite l'operatore Spark, consulta Using a SparkApplication nella documentazione di Kubernetes Operator for Apache Spark (8s-operator) su. spark-on-k GitHub

Esecuzione di processi Spark con spark-submit

I rilasci 6.10.0 o successivi di Amazon EMR supportano `spark-submit` come strumento di riga di comando che puoi utilizzare per inviare ed eseguire applicazioni Spark in un cluster Amazon EMR su EKS.

> **Note**
>
> Amazon EMR calcola i prezzi di Amazon EKS in base al consumo di vCPU e memoria. Questo calcolo si applica ai driver e ai pod executor. Questo calcolo inizia dal momento in cui scarichi l'immagine dell'applicazione Amazon EMR fino alla chiusura del pod Amazon EKS e viene arrotondato al secondo più vicino.

Argomenti

- Configurazione di spark-submit per Amazon EMR su EKS
- Nozioni di base su spark-submit per Amazon EMR su EKS
- Verifica i requisiti di sicurezza dell'account del servizio driver Spark per spark-submit

Configurazione di spark-submit per Amazon EMR su EKS

Completa le seguenti attività per definire la configurazione prima di poter eseguire un'applicazione con spark-submit in Amazon EMR su EKS. Se hai già effettuato la registrazione ad Amazon Web Services (AWS) e hai utilizzato Amazon EKS, ti manca poco per cominciare a utilizzare Amazon EMR su EKS. Se hai già completato uno dei prerequisiti, puoi saltarli e passare a quello successivo.

- Installa o aggiorna alla versione più recente di AWS CLI: se hai già installato il AWS CLI, conferma di disporre della versione più recente.
- Configura kubectl ed eksctl: eksctl è uno strumento da riga di comando che usi per comunicare con Amazon EKS.
- Inizia a usare Amazon EKS — eksctl — Segui i passaggi per creare un nuovo cluster Kubernetes con nodi in Amazon EKS.
- Seleziona un URI dell'immagine di base Amazon EMR (rilascio 6.10.0 o successivo): il comando `spark-submit` è supportato con i rilasci 6.10.0 e successivi di Amazon EMR.
- Verifica che l'account di servizio del driver disponga delle autorizzazioni adeguate per creare e osservare i pod dell'executor. Per ulteriori informazioni, consulta Verifica i requisiti di sicurezza dell'account del servizio driver Spark per spark-submit.
- Configura il tuo profilo di credenziali AWS locale.
- Dalla console Amazon EKS, scegli il tuo cluster EKS, quindi trova l'endpoint del cluster EKS, che si trova in Panoramica, Dettagli, quindi Endpoint del server API.

Nozioni di base su spark-submit per Amazon EMR su EKS

Amazon EMR 6.10.0 e versioni successive supportano spark-submit per l'esecuzione di applicazioni Spark su un cluster Amazon EKS. La sezione che segue mostra come inviare un comando per un'applicazione Spark.

Esecuzione di un'applicazione Spark

Per eseguire l'applicazione Spark, completa questa procedura:

1. Prima di poter eseguire un'applicazione Spark con il comando `spark-submit`, completa le fasi indicate in Configurazione di spark-submit per Amazon EMR su EKS.
2. Esegui un contenitore con un'immagine di base Amazon EMR su EKS. Per ulteriori informazioni, consulta Come selezionare l'URI di un'immagine di base.

   ```
   kubectl run -it containerName --image=EMRonEKSImage --command -n namespace /bin/
   bash
   ```

3. Imposta i valori delle seguenti variabili di ambiente:

   ```
   export SPARK_HOME=spark-home
   ```

```
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

4. A questo punto, invia l'applicazione Spark con il comando seguente:

```
$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
```

Per ulteriori informazioni sull'invio di applicazioni a Spark, consulta Invio di applicazioni nella documentazione di Apache Spark.

> **Important**
>
> `spark-submit` supporta solo la modalità cluster come meccanismo di invio.

Verifica i requisiti di sicurezza dell'account del servizio driver Spark per spark-submit

Il pod del driver Spark utilizza un account di servizio Kubernetes per accedere al server dell'API di Kubernetes per creare e osservare i pod dell'executor. L'account di servizio del driver deve disporre delle autorizzazioni appropriate per elencare, creare, modificare, applicare patch ai ed eliminare i pod nel cluster. Puoi verificare di poter elencare queste risorse eseguendo il comando seguente:

```
kubectl auth can-i list|create|edit|delete|patch pods
```

Verifica di disporre delle autorizzazioni necessarie eseguendo ogni comando.

```
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
```

Per questo ruolo di servizio vigono le regole elencate di seguito:

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

Configurazione dei ruoli IAM per gli account di servizio (IRSA) per spark-submit

Le seguenti sezioni spiegano come configurare i ruoli IAM per gli account di servizio (IRSA) per autenticare e autorizzare gli account di servizio Kubernetes in modo da poter eseguire le applicazioni Spark archiviate in Amazon S3.

Prerequisiti

Prima di provare uno qualsiasi degli esempi di questa documentazione, assicurati di aver soddisfatto i seguenti prerequisiti:

- Hai completato la configurazione di spark-submit

- Hai creato un bucket S3 e caricato il jar dell'applicazione Spark

Configurazione di un account di servizio Kubernetes per assumere un ruolo IAM

I passaggi seguenti spiegano come configurare un account di servizio Kubernetes per assumere un ruolo (IAM). AWS Identity and Access Management Dopo aver configurato i pod per utilizzare l'account di servizio, possono accedere a qualsiasi account a Servizio AWS cui il ruolo dispone delle autorizzazioni di accesso.

1. Crea un file di policy per consentire l'accesso in sola lettura all'oggetto Amazon S3 che hai caricato:

```
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. Creare la policy IAM.

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

3. Crea un ruolo IAM e associalo a un account di servizio Kubernetes per il driver Spark

```
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. Crea un file YAML con le autorizzazioni richieste per l'account del servizio driver Spark:

```
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. Applica le configurazioni di associazione dei ruoli del cluster.

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

6. Il `kubectl` comando dovrebbe restituire la conferma dell'account creato.

```
serviceaccount/emr-containers-sa-spark created
clusterrolebinding.rbac.authorization.k8s.io/emr-containers-role-spark configured
```

Esecuzione dell'applicazione Spark

Amazon EMR 6.10.0 e versioni successive supportano spark-submit per l'esecuzione di applicazioni Spark su un cluster Amazon EKS. Per eseguire l'applicazione Spark, completa questa procedura:

1. Assicurati di aver completato i passaggi descritti in Configurazione di spark-submit per Amazon EMR su EKS.
2. Imposta i valori delle seguenti variabili di ambiente:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

3. A questo punto, invia l'applicazione Spark con il comando seguente:

```
$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. Dopo che lo spark driver ha terminato il job Spark, dovresti vedere una riga di registro alla fine dell'invio che indica che il job Spark è terminato.

```
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
```

Rimozione

Quando hai finito di eseguire le applicazioni, puoi eseguire la pulizia con il seguente comando.

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

Utilizzo di Apache Livy con Amazon EMR su EKS

Con le versioni 7.1.0 e successive di Amazon EMR, puoi utilizzare Apache Livy per inviare lavori su Amazon EMR su EKS. Utilizzando Apache Livy, puoi configurare il tuo endpoint REST Apache Livy e utilizzarlo per distribuire e gestire le applicazioni Spark sui tuoi cluster Amazon EKS. Dopo aver installato Livy nel tuo cluster Amazon EKS, puoi utilizzare l'endpoint Livy per inviare applicazioni Spark al tuo server Livy. Il server gestisce il ciclo di vita delle applicazioni Spark.

> **Note**
>
> Amazon EMR calcola i prezzi di Amazon EKS in base al consumo di vCPU e memoria. Questo calcolo si applica ai driver e ai pod executor. Questo calcolo inizia dal momento in cui scarichi l'immagine dell'applicazione Amazon EMR fino alla chiusura del pod Amazon EKS e viene arrotondato al secondo più vicino.

Argomenti

Configurazione di Apache Livy per Amazon EMR su EKS

Prima di poter installare Apache Livy sul tuo cluster Amazon EKS, devi installare e configurare un set di strumenti prerequisiti. Questi includono AWS CLI, uno strumento da riga di comando fondamentale per lavorare con AWS le risorse, strumenti da riga di comando per lavorare con Amazon EKS e un

controller utilizzato in questo caso d'uso per rendere l'applicazione cluster disponibile su Internet e per instradare il traffico di rete.

- Installa o aggiorna alla versione più recente di AWS CLI: se hai già installato il AWS CLI, conferma di disporre della versione più recente.
- Configura kubectl ed eksctl: eksctl è uno strumento da riga di comando che usi per comunicare con Amazon EKS.
- Installa Helm: il programma di gestione del pacchetto Helm per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes.
- Inizia a usare Amazon EKS — eksctl — Segui i passaggi per creare un nuovo cluster Kubernetes con nodi in Amazon EKS.
- Seleziona un'etichetta di release di Amazon EMR: Apache Livy è supportato dalle versioni 7.1.0 e successive di Amazon EMR.
- Installa il controller ALB: il controller ALB gestisce i cluster ELB for Kubernetes. AWS Crea un AWS Network Load Balancer (NLB) quando crei un Kubernetes Ingress durante la configurazione di Apache Livy.

Guida introduttiva ad Apache Livy su Amazon EMR su EKS

Completa i seguenti passaggi per installare Apache Livy. Includono la configurazione del gestore di pacchetti, la creazione di uno spazio dei nomi per l'esecuzione dei carichi di lavoro Spark, l'installazione di Livy, l'impostazione del bilanciamento del carico e i passaggi di verifica. Devi completare questi passaggi per eseguire un processo in batch con Spark.

1. Se non l'hai già fatto, configura Apache Livy per Amazon EMR su EKS.
2. Autentica il client Helm nel registro Amazon ECR. Puoi trovare il `ECR-registry-account` valore corrispondente per i tuoi account del Regione AWS registro Amazon ECR per regione.

   ```
   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. La configurazione di Livy crea un account di servizio per il server Livy e un altro account per l'applicazione Spark. Per configurare IRSA per gli account di servizio, consulta Configurazione delle autorizzazioni di accesso con i ruoli IAM per gli account di servizio (IRSA).
4. Crea uno spazio dei nomi per eseguire i carichi di lavoro Spark.

```
kubectl create ns <spark-ns>
```

5. Usa il seguente comando per installare Livy.

Questo endpoint Livy è disponibile solo internamente per il VPC nel cluster EKS. Per abilitare l'accesso oltre il VPC, imposta il comando `–-set loadbalancer.internal=false` di installazione Helm.

> **Note**
>
> Per impostazione predefinita, SSL non è abilitato all'interno di questo endpoint Livy e l'endpoint è visibile solo all'interno del VPC del cluster EKS. Se imposti `loadbalancer.internal=false` and`ssl.enabled=false`, esponi un endpoint non sicuro all'esterno del tuo VPC. Per configurare un endpoint Livy sicuro, consulta Configurazione di un endpoint Apache Livy sicuro con TLS/SSL.

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.12.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.12.0:latest \
  --set sparkNamespace=<spark-ns> \
  --create-namespace
```

Vedrai il seguente output.

```
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
```

I nomi degli account di servizio predefiniti per il server Livy e la sessione Spark sono e. `emr-containers-sa-livy` `emr-containers-sa-spark-livy` Per utilizzare nomi personalizzati, usa i parametri `serviceAccounts.name` and`sparkServiceAccount.name`.

```
--set serviceAccounts.name=my-service-account-for-livy
--set sparkServiceAccount.name=my-service-account-for-spark
```

6. Verifica di aver installato il grafico Helm.

```
helm list -n livy-ns -o yaml
```

Il `helm list` comando dovrebbe restituire informazioni sul nuovo grafico Helm.

```
app_version: 0.7.1-incubating
chart: livy-emr-7.12.0
name: livy-demo
namespace: livy-ns
revision: "1"
status: deployed
updated: 2024-02-08 22:39:53.539243 -0800 PST
```

7. Verificare che il Network Load Balancer sia attivo.

```
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. Ora verifica che il gruppo target nel Network Load Balancer sia sano.

```
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
```

Di seguito è riportato un esempio di output che mostra lo stato del gruppo target:

```
{
```

```
    "TargetHealthDescriptions": [
        {
            "Target": {
                "Id": "<target IP>",
                "Port": 8998,
                "AvailabilityZone": "us-west-2d"
            },
            "HealthCheckPort": "8998",
            "TargetHealth": {
                "State": "healthy"
            }
        }
    ]
}
```

Una volta raggiunto lo status del vostro NLB `active` e del vostro gruppo target`healthy`, potete continuare. Questo processo può richiedere alcuni minuti.

9. Recupera l'endpoint Livy dall'installazione di Helm. Il fatto che il tuo endpoint Livy sia sicuro o meno dipende dal fatto che tu abbia abilitato SSL.

```
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. Recupera l'account del servizio Spark dall'installazione di Helm

```
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"
```

L'output visualizzato è simile al seguente:

```
emr-containers-sa-spark-livy
```

11. Se imposti di `internalALB=true` abilitare l'accesso dall'esterno del tuo VPC, crea un' EC2 istanza Amazon e assicurati che Network Load Balancer consenta il traffico di rete proveniente dall'istanza. EC2 Devi farlo affinché l'istanza abbia accesso al tuo endpoint Livy. Per ulteriori informazioni sull'esposizione sicura dell'endpoint all'esterno del VPC, consulta Configurazione con un endpoint Apache Livy sicuro con TLS/SSL.
12. L'installazione di Livy crea l'account di servizio per eseguire le applicazioni Spark. `emr-containers-sa-spark` Se la tua applicazione Spark utilizza AWS risorse come S3 o richiama operazioni AWS API o CLI, devi collegare un ruolo IAM con le autorizzazioni necessarie al tuo account del servizio spark. Per ulteriori informazioni, consulta Configurazione delle autorizzazioni di accesso con i ruoli IAM per gli account di servizio (IRSA).

Apache Livy supporta configurazioni aggiuntive che è possibile utilizzare durante l'installazione di Livy. Per ulteriori informazioni, consulta Proprietà di installazione per Apache Livy su Amazon EMR sulle versioni EKS.

Esecuzione di un'applicazione Spark con Apache Livy per Amazon EMR su EKS

Prima di poter eseguire un'applicazione Spark con Apache Livy, assicurati di aver completato i passaggi in Configurazione di Apache Livy per Amazon EMR su EKS e Guida introduttiva ad Apache Livy per Amazon EMR su EKS.

Puoi usare Apache Livy per eseguire due tipi di applicazioni:

- Sessioni batch: un tipo di carico di lavoro Livy per inviare lavori batch Spark.
- Sessioni interattive: un tipo di carico di lavoro Livy che fornisce un'interfaccia programmatica e visiva per eseguire le query Spark.

> **Note**
>
> I driver pod ed executor di sessioni diverse possono comunicare tra loro. I namespace non garantiscono alcuna sicurezza tra i pod. Kubernetes non consente autorizzazioni selettive su un sottoinsieme di pod all'interno di un determinato spazio dei nomi.

Esecuzione di sessioni in batch

Per inviare un processo batch, utilizzare il comando seguente.

```
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-7.12.0:latest",
                "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-
service-account>"
            }
          }' <livy-endpoint>/batches
```

Per monitorare il processo batch, utilizzare il comando seguente.

```
curl -s -k -H 'Content-Type: application/json' -X GET <livy-endpoint>/batches/my-
session
```

Esecuzione di sessioni interattive

Per eseguire sessioni interattive con Apache Livy, consulta i passaggi seguenti.

1. Assicurati di avere accesso a un notebook Jupyter ospitato autonomamente o gestito, come un notebook AI Jupyter. SageMaker Sul tuo notebook jupyter deve essere installato sparkmagic.
2. Crea un bucket per la configurazione di Spark. `spark.kubernetes.file.upload.path` Assicurati che l'account del servizio Spark abbia accesso in lettura e scrittura al bucket. Per maggiori dettagli su come configurare l'account del servizio Spark, consulta Configurazione delle autorizzazioni di accesso con i ruoli IAM per gli account di servizio (IRSA)
3. Carica sparkmagic nel notebook Jupyter con il comando. `%load_ext sparkmagic.magics`
4. Esegui il comando `%manage_spark` per configurare il tuo endpoint Livy con il notebook Jupyter. Scegli la scheda Aggiungi endpoint, scegli il tipo di autenticazione configurato, aggiungi l'endpoint Livy al notebook, quindi scegli Aggiungi endpoint.

5. Esegui **`%manage_spark`** di nuovo per creare il contesto Spark e poi vai alla sessione Crea. Scegli l'endpoint Livy, specifica un nome di sessione univoco, scegli una lingua e aggiungi le seguenti proprietà.

```
{
  "conf": {
    "spark.kubernetes.namespace": "livy-namespace",
    "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
emr-7.12.0:latest",
    "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-service-
account>",
    "spark.kubernetes.file.upload.path": "<URI_TO_S3_LOCATION_>"
  }
}
```

6. Invia l'applicazione e attendi che crei il contesto Spark.
7. Per monitorare lo stato della sessione interattiva, esegui il comando seguente.

```
curl -s -k -H 'Content-Type: application/json' -X GET livy-endpoint/sessions/my-
interactive-session
```

Monitoraggio delle applicazioni Spark

Per monitorare lo stato di avanzamento delle tue applicazioni Spark con l'interfaccia utente Livy, usa il link. `http://<livy-endpoint>/ui`

Disinstallazione di Apache Livy con Amazon EMR su EKS

Segui questi passaggi per disinstallare Apache Livy.

1. Elimina la configurazione di Livy utilizzando i nomi del tuo namespace e del nome dell'applicazione. In questo esempio, il nome dell'applicazione è `livy-demo` e lo spazio dei nomi è. `livy-ns`

```
helm uninstall livy-demo -n livy-ns
```

2. Durante la disinstallazione, Amazon EMR su EKS elimina il servizio Kubernetes in Livy, i sistemi di bilanciamento AWS del carico e i gruppi target creati durante l'installazione. L'eliminazione

delle risorse può richiedere alcuni minuti. Assicurati che le risorse vengano eliminate prima di installare nuovamente Livy nello spazio dei nomi.

3. Elimina lo spazio dei nomi Spark.

```
kubectl delete namespace spark-ns
```

Sicurezza per Apache Livy con Amazon EMR su EKS

Consulta i seguenti argomenti per saperne di più sulla configurazione della sicurezza per Apache Livy con Amazon EMR su EKS. Queste opzioni includono l'utilizzo della sicurezza a livello di trasporto, il controllo degli accessi basato sui ruoli, ovvero l'accesso basato sul ruolo di una persona all'interno di un'organizzazione, e l'utilizzo dei ruoli IAM, che forniscono l'accesso alle risorse, in base alle autorizzazioni concesse.

Argomenti

- Configurazione di un endpoint Apache Livy sicuro con TLS/SSL
- Configurazione delle autorizzazioni delle applicazioni Apache Livy e Spark con il controllo degli accessi basato sui ruoli (RBAC)
- Configurazione delle autorizzazioni di accesso con i ruoli IAM per gli account di servizio (IRSA)

Configurazione di un endpoint Apache Livy sicuro con TLS/SSL

Consulta le seguenti sezioni per saperne di più sulla configurazione di Apache Livy per Amazon EMR su EKS con crittografia end-to-end TLS e SSL.

Configurazione della crittografia TLS e SSL

Per configurare la crittografia SSL sul tuo endpoint Apache Livy, segui questi passaggi.

- Installa il driver CSI Secrets Store e AWS Secrets and Configuration Provider (ASCP): il driver CSI Secrets Store e ASCP archiviano in modo sicuro i certificati JKS e le password di Livy necessari al pod del server Livy per abilitare SSL. Puoi anche installare solo il driver CSI Secrets Store e utilizzare qualsiasi altro provider di segreti supportato.
- Crea un certificato ACM: questo certificato è necessario per proteggere la connessione tra il client e l'endpoint ALB.

- Imposta un certificato JKS, una password di chiave e una password di archivio chiavi per, necessari per Gestione dei segreti AWS proteggere la connessione tra l'endpoint ALB e il server Livy.
- Aggiungi le autorizzazioni all'account del servizio Livy da cui recuperare i segreti Gestione dei segreti AWS : il server Livy necessita di queste autorizzazioni per recuperare i segreti da ASCP e aggiungere le configurazioni Livy per proteggere il server Livy. Per aggiungere le autorizzazioni IAM a un account di servizio, consulta Configurazione delle autorizzazioni di accesso con i ruoli IAM per gli account di servizio (IRSA).

Configurazione di un certificato JKS con una chiave e una password di archivio chiavi per Gestione dei segreti AWS

Segui questi passaggi per configurare un certificato JKS con una chiave e una password del keystore.

1. Genera un file keystore per il server Livy.

```
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. Crea un certificato.

```
keytool -export -alias <host> -keystore mykeystore.jks -rfc -
file mycertificate.cert -storepass <storePassword>
```

3. Crea un file truststore.

```
keytool -import -noprompt -alias <host>-file <cert_file> -
keystore <truststore_file> -storepass <truststorePassword>
```

4. Salva il certificato JKS in. Gestione dei segreti AWS Sostituiscilo `livy-jks-secret` con il tuo segreto e `fileb://mykeystore.jks` con il percorso del certificato JKS del keystore.

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy keystore JKS secret" \
--secret-binary fileb://mykeystore.jks
```

5. Salva il keystore e la password della chiave in Secrets Manager. Assicurati di utilizzare i tuoi parametri.

```
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. Crea uno spazio dei nomi del server Livy con il seguente comando.

```
kubectl create ns <livy-ns>
```

7. Crea l'`ServiceProviderClass`oggetto per il server Livy che ha il certificato JKS e le password.

```
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>
```

Guida introduttiva a Apache Livy abilitato per SSL

Dopo aver abilitato SSL sul tuo server Livy, devi `serviceAccount` configurarlo per avere accesso a and secrets su. `keyStore` `keyPasswords` Gestione dei segreti AWS

1. Crea lo spazio dei nomi del server Livy.

```
kubectl create namespace <livy-ns>
```

2. Configura l'account del servizio Livy per avere accesso ai segreti in Secrets Manager. Per ulteriori informazioni sulla configurazione di IRSA, vedere Configurazione di IRSA durante l'installazione di Apache Livy.

```
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. Installa Livy. Per il parametro Helm chart --version, usa l'etichetta di rilascio di Amazon EMR, ad esempio. `7.1.0` È inoltre necessario sostituire l'ID dell'account del registro Amazon ECR e l'ID regionale con i propri. IDs Puoi trovare il `ECR-registry-account` valore corrispondente per i tuoi account del Regione AWS registro Amazon ECR per regione.

```
helm install <livy-app-name> \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.12.0 \
  --namespace livy-namespace-name \
  --set image=<ECR-registry-account.dkr.ecr>.<region>.amazonaws.com/livy/
emr-7.12.0: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. Continua dal passaggio 5 dell'Installazione di Apache Livy su Amazon EMR su EKS.

Configurazione delle autorizzazioni delle applicazioni Apache Livy e Spark con il controllo degli accessi basato sui ruoli (RBAC)

Per distribuire Livy, Amazon EMR su EKS crea un account e un ruolo del servizio server e un account e un ruolo di servizio Spark. Questi ruoli devono disporre delle autorizzazioni RBAC necessarie per completare la configurazione ed eseguire le applicazioni Spark.

Autorizzazioni RBAC per l'account e il ruolo del servizio server

Amazon EMR su EKS crea l'account e il ruolo del servizio server Livy per gestire le sessioni Livy per i job Spark e il routing del traffico da e verso l'ingresso e altre risorse.

Il nome predefinito per questo account di servizio è `emr-containers-sa-livy`. Deve disporre delle seguenti autorizzazioni.

```
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"
```

Autorizzazioni RBAC per l'account e il ruolo del servizio Spark

Un pod del driver Spark necessita di un account di servizio Kubernetes nello stesso spazio dei nomi del pod. Questo account di servizio necessita delle autorizzazioni per gestire gli executor pod e tutte le risorse richieste dal driver pod. A meno che l'account di servizio predefinito nel namespace non disponga delle autorizzazioni richieste, il driver si guasta e si chiude. Sono richieste le seguenti autorizzazioni RBAC.

```
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"
```

Configurazione delle autorizzazioni di accesso con i ruoli IAM per gli account di servizio (IRSA)

Per impostazione predefinita, il server Livy e i driver e gli esecutori dell'applicazione Spark non hanno accesso alle risorse. AWS L'account del servizio server e l'account del servizio spark controllano l'accesso alle AWS risorse per il server Livy e i pod dell'applicazione Spark. Per concedere l'accesso, devi mappare gli account di servizio con un ruolo IAM dotato delle autorizzazioni necessarie. AWS

È possibile configurare la mappatura IRSA prima di installare Apache Livy, durante l'installazione o dopo aver completato l'installazione.

Configurazione di IRSA durante l'installazione di Apache Livy (per l'account del servizio server)

> **Note**
>
> Questa mappatura è supportata solo per l'account del servizio server.

1. Assicurati di aver completato la configurazione di Apache Livy per Amazon EMR su EKS e che sia in corso l'installazione di Apache Livy con Amazon EMR su EKS.
2. Crea uno spazio dei nomi Kubernetes per il server Livy. In questo esempio, il nome dello spazio dei nomi è. `livy-ns`
3. Crea una policy IAM che includa le autorizzazioni Servizi AWS per le quali desideri che i tuoi pod accedano. L'esempio seguente crea una policy IAM per ottenere risorse Amazon S3 per il punto di ingresso Spark.

   ```
   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. Usa il comando seguente per impostare il tuo Account AWS ID su una variabile.

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

5. Imposta il provider di identità OpenID Connect (OIDC) del cluster su una variabile di ambiente.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
```

6. Impostare le variabili per il namespace e il nome dell'account del servizio. Assicurati di utilizzare i tuoi valori.

```
export namespace=default
export service_account=my-service-account
```

7. Create un file di criteri di fiducia con il seguente comando. Se desideri concedere l'accesso al ruolo a tutti gli account di servizio all'interno di un namespace, copia il comando seguente e sostituiscilo con e `StringEquals` sostituiscilo con`StringLike`. `$service_account *`

```
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. Crea il ruolo.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

9. Utilizzate il seguente comando Helm install per impostare l'`serviceAccount.executionRoleArn`IRSA per mappare. Di seguito è riportato un esempio del comando Helm install. Puoi trovare il `ECR-registry-account` valore corrispondente per i tuoi account del Regione AWS registro Amazon ECR per regione.

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.us-west-2.amazonaws.com/livy \
  --version 7.12.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.12.0:latest \
  --set sparkNamespace=spark-ns \
  --set serviceAccount.executionRoleArn=arn:aws:iam::123456789012:role/my-role
```

Mappatura di IRSA su un account di servizio Spark

Prima di mappare IRSA a un account di servizio Spark, assicurati di aver completato i seguenti elementi:

- Assicurati di aver completato la configurazione di Apache Livy per Amazon EMR su EKS e che sia in corso l'installazione di Apache Livy con Amazon EMR su EKS.

- È necessario disporre di un provider IAM OpenID Connect (OIDC) esistente per il cluster. Per vedere se ne hai già uno o come crearne uno, consulta Creare un provider IAM OIDC per il tuo cluster.
- Assicurati di aver installato la versione 0.171.0 o successiva della `eksctl` CLI installata o. AWS CloudShell Per installare o aggiornare`eksctl`, consulta Installazione della documentazione. `eksctl`

Segui questi passaggi per mappare IRSA al tuo account di servizio Spark:

1. Usa il seguente comando per ottenere l'account del servizio Spark.

```
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. Imposta le variabili per il namespace e il nome dell'account del servizio.

```
export namespace=default
export service_account=my-service-account
```

3. Utilizza il comando seguente per creare un file di policy di fiducia per il ruolo IAM. L'esempio seguente autorizza tutti gli account di servizio all'interno del namespace a utilizzare il ruolo. A tale scopo, sostituisci `StringEquals` con `StringLike` e sostituisci `$service_account` con *.

```
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. Crea il ruolo.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-relationship.json --description "my-role-description"
```

5. Mappa l'account del server o del servizio Spark con il seguente `eksctl` comando. Assicurati di usare i tuoi valori.

```
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
```

Proprietà di installazione per Apache Livy su Amazon EMR su versioni EKS

L'installazione di Apache Livy consente di selezionare una versione del grafico Livy Helm. Il diagramma Helm offre una varietà di proprietà per personalizzare l'esperienza di installazione e configurazione. Queste proprietà sono supportate per Amazon EMR nelle versioni 7.1.0 e successive di EKS.

Argomenti

- Proprietà di installazione di Amazon EMR 7.1.0

Proprietà di installazione di Amazon EMR 7.1.0

La tabella seguente descrive tutte le proprietà Livy supportate. Quando si installa Apache Livy, è possibile scegliere la versione cartografica Livy Helm. Per impostare una proprietà durante l'installazione, usa il comando. `--set <property>=<value>`

Proprietà	Description	Predefinita
image	L'URI di rilascio di Amazon EMR del server Livy. Questa è una configurazione richiesta.	""
SparkNamespace	Namespace per eseguire le sessioni di Livy Spark. Ad esempio, specifica «livy». Questa è una configurazione richiesta.	""
NameOverride	Fornisci un nome invece di. `livy` Il nome è impostato come etichetta per tutte le risorse Livy	«Livy»
Nome completo Override	Fornisci un nome da utilizzar e al posto dei nomi completi delle risorse.	""
ssl. abilitato	Abilita end-to-end SSL dall'endpoint Livy al server Livy.	FALSE
Certificato SSL. ARN	Se SSL è abilitato, questo è l'ARN del certificato ACM per l'NLB creato dal servizio.	""
ssl. secretProviderClassNome	Se SSL è abilitato, questo è il nome segreto della classe del provider per proteggere l'NLB per la connessione al server Livy con SSL.	""
ssl. keyStoreObjectNome	Se SSL è abilitato, il nome dell'oggetto per il certifica	""

Proprietà	Description	Predefinita
	to keystore nella classe del provider segreto.	
ssl. keyPasswordsObject Nome	Se SSL è abilitato, il nome dell'oggetto segreto contenent e il keystore e la password della chiave.	""
rbac.create	Se vero, crea risorse RBAC.	FALSE
Account di servizio. Crea	Se vero, crea un account di servizio Livy.	TRUE
Account di servizio. Nome	Il nome dell'account di servizio da utilizzare per Livy. Se non imposti questa proprietà e non crei un account di servizio, Amazon EMR su EKS genera automaticamente un nome utilizzando la proprietà `fullname` override.	"emr-containers-sa-livy"
Account di servizio. execution RoleArn	L'ARN del ruolo di esecuzione dell'account del servizio Livy.	""
sparkServiceAccount.creare	Se vero, crea l'account del servizio Spark in `.Release. Namespace`	TRUE

Proprietà	Description	Predefinita
sparkServiceAccount.name	Il nome dell'account di servizio da usare per Spark. Se non imposti questa proprietà e non crei un account di servizio Spark, Amazon EMR su EKS genera automaticamente un nome con `fullname0 verride` la proprietà - `spark-livy` con suffisso.	«-livido» emr-containers-sa-spark
nome.servizio	Nome del servizio Livy	`"emr-containers-li vy"`
servizio.annotations	annotazioni del servizio Livy	`{}`
loadbalancer.abilitato	Se creare un load balancer per il servizio Livy utilizzato per esporre l'endpoint Livy all'esterno del cluster Amazon EKS.	`FALSE`
loadbalancer.internal	Se configurare l'endpoint Livy come interno al VPC o esterno. L'impostazione di questa proprietà per `FALSE` esporre l'endpoint a sorgenti esterne al VPC. Ti consigliamo di proteggere l'endpoint con TLS/SSL. Per ulteriori informazioni, consulta Configurazione della crittografia TLS e SSL.	`FALSE`

Proprietà	Description	Predefinita
imagePullSecrets	L'elenco dei `imagePullSecret` nomi da usare per estrarre l'immagine di Livy dagli archivi privati.	[]
risorse	Le richieste e i limiti di risorse per i contenitori Livy.	{}
nodeSelector	I nodi per i quali pianificare i pod Livy.	{}
tolleranze	Un elenco contenente le tolleranze dei pod Livy da definire.	[]
affinità	Le regole di affinità dei Livy pod.	{}
persistenza. abilitata	Se true, abilita la persistenza per le directory delle sessioni.	FALSE
Persistence.subpath	Il sottopercorso PVC da montare nelle directory delle sessioni.	""
Persistence.ExistingClaim	Il PVC da usare invece di crearne uno nuovo.	{}

Proprietà	Description	Predefinita
Persistence.StorageClass	La classe di archiviazione da usare. Per definire questo parametro, utilizzat e il formato`storageCl assName: <storageC lass>` . L'impostazione di questo parametro su "-" disattiva il provision ing dinamico. Se imposti questo parametro su null o non specifichi nulla, Amazon EMR su EKS non imposta storageClassName a e utilizza il provisioner predefinito.	""
Persistence.accessMode	La modalità di accesso in PVC.	`ReadWriteOnce`
persistenza. Dimensione	La dimensione del PVC.	20 Gi
persistenza. annotazioni	Annotazioni aggiuntive per il PVC.	{}
invidia. *	Env aggiuntivi da impostare sul contenitore Livy. Per ulteriori informazioni, consulta Inserimento delle proprie configurazioni Livy e Spark durante l'installazione di Livy.	{}
ENV da. *	Env aggiuntivi da impostare su Livy da una mappa di configurazione o segreta di Kubernetes.	[]

Proprietà	Description	Predefinita
LivyConf. *	Voci livy.conf aggiuntive da impostare da una mappa di configurazione o segreta di Kubernetes montata.	{}
sparkDefaultsConf.*	`spark-defaults.conf` Voci aggiuntive da impostare da una mappa di configurazione o segreta di Kubernetes montata.	{}

Risolvi gli errori più comuni relativi al formato delle variabili di ambiente

Quando inserisci le configurazioni Livy e Spark, ci sono formati di variabili d'ambiente che non sono supportati e possono causare errori. La procedura illustra una serie di passaggi per garantire l'utilizzo dei formati corretti.

Inserimento delle proprie configurazioni Livy e Spark durante l'installazione di Livy

È possibile configurare qualsiasi variabile di ambiente Apache Livy o Apache Spark con la proprietà Helm. `env.*` Segui i passaggi seguenti per convertire la configurazione di esempio in un formato di variabile `example.config.with-dash.withUppercase` di ambiente supportato.

1. Sostituisci le lettere maiuscole con 1 e una minuscola della lettera. Ad esempio `example.config.with-dash.withUppercase` diventa `example.config.with-dash.with1uppercase`.
2. Sostituisci i trattini (-) con 0. Ad esempio, diventa `example.config.with-dash.with1uppercase` `example.config.with0dash.with1uppercase`
3. Sostituisci i punti (.) con i caratteri di sottolineatura (_). Ad esempio `example.config.with0dash.with1uppercase` diventa `example_config_with0dash_with1uppercase`.
4. Sostituisci tutte le lettere minuscole con lettere maiuscole.
5. Aggiungi il prefisso al nome `LIVY_` della variabile.

6. Usa la variabile durante l'installazione di Livy tramite il helm chart usando il formato --set env. *YOUR_VARIABLE_NAME*.valore= *yourvalue*

Ad esempio, per impostare le configurazioni `livy.server.recovery.state-store = filesystem` Livy e Spark e utilizzare queste proprietà Helm: `spark.kubernetes.executor.podNamePrefix = my-prefix`

```
–set env.LIVY_LIVY_SERVER_RECOVERY_STATE0STORE.value=filesystem
–set env.LIVY_SPARK_KUBERNETES_EXECUTOR_POD0NAME0PREFIX.value=myprefix
```

Gestione delle esecuzioni di processi Amazon EMR su EKS

Le sezioni seguenti trattano argomenti che semplificano la gestione delle esecuzioni di processi Amazon EMR su EKS. Queste includono la configurazione dei parametri di esecuzione dei lavori quando si utilizza il AWS CLI, la configurazione della modalità di archiviazione dei dati di registro, l'esecuzione degli script SQL di Spark per eseguire le query, la comprensione degli stati di esecuzione dei processi e la conoscenza di come monitorare i lavori. Puoi esaminare questi argomenti, generalmente con ordine, se desideri configurare e completare un'esecuzione di job per elaborare i dati.

Argomenti

- La gestione dei job viene eseguita con AWS CLI
- Esecuzione degli script SQL StartJobRun di Spark tramite l'API
- Stati delle esecuzioni di processi
- Visualizzazione dei processi nella console Amazon EMR
- Errori comuni durante l'esecuzione di processi

La gestione dei job viene eseguita con AWS CLI

In questo argomento viene illustrato come gestire le esecuzioni dei job con AWS Command Line Interface (AWS CLI). Approfondisce le proprietà, come i parametri di sicurezza, il driver e varie impostazioni di override. Include anche argomenti secondari che coprono vari modi per configurare la registrazione.

Argomenti

- Opzioni per la configurazione di un'esecuzione di processo

- Configurazione di un'esecuzione di processo per utilizzare i log Amazon S3
- Configurare un job run per usare Amazon CloudWatch Logs
- Elenco delle esecuzioni di processi
- Descrizione di un'esecuzione di processo
- Annullamento di un'esecuzione di processo

Opzioni per la configurazione di un'esecuzione di processo

Utilizza le seguenti opzioni per configurare i parametri dell'esecuzione di processo:

- `--execution-role-arn`: è necessario fornire un ruolo IAM per eseguire i processi. Per ulteriori informazioni, consulta Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS.
- `--release-label`: è possibile implementare Amazon EMR su EKS con le versioni Amazon EMR 5.32.0, 6.2.0 e versioni successive. Amazon EMR su EKS non è supportato nelle versioni precedenti di Amazon EMR. Per ulteriori informazioni, consulta Rilasci di Amazon EMR su EKS.
- `--job-driver`: il driver di processo viene utilizzato per fornire input sul processo principale. Si tratta di un campo di tipo unione in cui è possibile far passare solo uno dei valori per il tipo di processo che si desidera eseguire. I tipi di processo supportati includono:
  - Processi Spark Submit: utilizzati per eseguire un comando tramite Spark Submit. Puoi usare questo tipo di lavoro per eseguire Scala, SparkR PySpark, SparkSQL e qualsiasi altro lavoro supportato tramite Spark Submit. Questo tipo di processo include i seguenti parametri:
    - Entrypoint - Questo è il riferimento HCFS (file system compatibile con Hadoop) al file principale che desideri eseguire. jar/py
    - EntryPointArguments - Questa è una serie di argomenti da passare al file principale. jar/py Dovrai gestire la lettura di questi parametri con il tuo codice di entrypoint. Ogni argomento nell'array deve essere separato da una virgola. EntryPointArguments non può contenere parentesi o parentesi, come (), {} o [].
    - SparkSubmitParameters - Questi sono i parametri spark aggiuntivi che desideri inviare al job. Utilizza questo parametro per sovrascrivere le proprietà Spark di default, come la memoria del driver o il numero di executor, tra cui —conf o —class. Per ulteriori informazioni, consulta Avvio delle applicazioni con spark-submit.
  - Processi Spark SQL: utilizzati per eseguire un file di query SQL tramite Spark SQL. Questo tipo di processo può essere utilizzato per eseguire i processi SparkSQL. Questo tipo di processo include i seguenti parametri:

- Punto d'ingresso: riferimento HCFS (file system compatibile con Hadoop) al file di query SQL da eseguire.

  Per l'elenco dei parametri Spark aggiuntivi da utilizzare nei processi Spark SQL, vedere Esecuzione degli script SQL StartJobRun di Spark tramite l'API.

- `--configuration-overrides`: puoi sovrascrivere le configurazioni predefinite per le applicazioni fornendo un oggetto di configurazione. Puoi utilizzare una sintassi abbreviata per fornire la configurazione oppure fare riferimento all'oggetto di configurazione in un file JSON. Gli oggetti di configurazione sono composti da una classificazione, proprietà e configurazioni nidificate opzionali. Le proprietà sono costituite dalle impostazioni da ignorare in un dato file. Puoi specificare diverse classificazioni per più applicazioni in un singolo oggetto JSON. Le classificazioni di configurazione disponibili variano a seconda della versione di rilascio di Amazon EMR. Per un elenco delle classificazioni di configurazione disponibili per ciascuna versione di rilascio di Amazon EMR, consulta Rilasci di Amazon EMR su EKS.

  Se si passa la stessa configurazione nella sostituzione di un'applicazione e nei parametri Spark Submit, i parametri Spark Submit hanno la precedenza. Segue l'elenco completo delle priorità di configurazione, ordinate dalla più alta alla più bassa.

  - Configurazione fornita durante la creazione di `SparkSession`.
  - Configurazione fornita come parte di `sparkSubmitParameters` tramite `–conf`.
  - Configurazione fornita come parte delle sostituzioni dell'applicazione.
  - Configurazioni ottimizzate scelte da Amazon EMR per il rilascio.
  - Configurazioni open source predefinite per l'applicazione.

  Per monitorare le esecuzioni dei job utilizzando Amazon CloudWatch o Amazon S3, devi fornire i dettagli di configurazione per. CloudWatch Per ulteriori informazioni, consultare Configurazione di un'esecuzione di processo per utilizzare i log Amazon S3 e Configurare un job run per usare Amazon CloudWatch Logs. Se il bucket o il gruppo di CloudWatch log S3 non esiste, Amazon EMR lo crea prima di caricare i log nel bucket.

- Per un elenco aggiuntivo delle opzioni di configurazione di Kubernetes, consulta Proprietà Spark su Kubernetes.

  Le seguenti configurazioni Spark non sono supportate:

  - `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**
>
> È possibile utilizzare `spark.kubernetes.container.image` per immagini Docker personalizzate. Per ulteriori informazioni, consulta Personalizzazione delle immagini Docker per Amazon EMR su EKS.

Configurazione di un'esecuzione di processo per utilizzare i log Amazon S3

Per monitorare l'avanzamento del lavoro e risolvere gli errori, devi configurare i lavori in modo da inviare informazioni di log ad Amazon S3, Amazon CloudWatch Logs o entrambi. Questo argomento fornisce le nozioni di base per inviare i log dell'applicazione ad Amazon S3 sui processi avviati con Amazon EMR su EKS.

Policy IAM dei registri S3

Prima che i processi possano inviare i dati dei log ad Amazon S3, nella policy delle autorizzazioni per il ruolo di esecuzione di processo devono essere incluse le seguenti autorizzazioni. Sostituisci *amzn-s3-demo-logging-bucket* con il nome del bucket di accesso.

JSON

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

```
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}
```

> **Note**
>
> Amazon EMR su EKS può anche creare un bucket Amazon S3. Se un bucket Amazon S3 non è disponibile, includi l'autorizzazione "`s3:CreateBucket`" nella policy IAM.

Dopo aver assegnato al ruolo di esecuzione le autorizzazioni appropriate per l'invio dei log ad Amazon S3, i dati dei log vengono inviati alle seguenti posizioni Amazon S3 quando `s3MonitoringConfiguration` viene trasmesso nella sezione `monitoringConfiguration` di una richiesta `start-job-run`, come mostrato in La gestione dei job viene eseguita con AWS CLI.

- Registri degli utenti -//jobs/ /containers//(stderr.gz/stdout.gz) *logUri virtual-cluster-id job-id pod-name*
- Registri dei driver -/*logUrivirtual-cluster-id*/jobs/ *job-id* /containers/ /spark- *spark-application-id* -driver/ *job-id* (stderr.gz/stdout.gz)
- Registri degli esecutori -*logUri*/*virtual-cluster-id*/jobs/ *job-id* /containers/*executor-pod-name*/(*spark-application-id*stderr.gz/stdout.gz)

Configurare un job run per usare Amazon CloudWatch Logs

Per monitorare l'avanzamento dei lavori e risolvere i problemi, devi configurare i lavori per inviare informazioni di log ad Amazon S3, Amazon CloudWatch Logs o entrambi. Questo argomento ti aiuta a iniziare a utilizzare CloudWatch i log sui tuoi lavori lanciati con Amazon EMR su EKS. Per ulteriori informazioni sui CloudWatch log, consulta Monitoring Log Files nella Amazon CloudWatch User Guide.

CloudWatch Registra la politica IAM

Affinché i lavori inviino i dati di registro a CloudWatch Logs, è necessario includere le seguenti autorizzazioni nella politica delle autorizzazioni per il ruolo di esecuzione del lavoro. Sostituisci *my_log_group_name* e *my_log_stream_prefix* con i nomi rispettivamente del gruppo di

CloudWatch log e dei nomi dei flussi di log. Amazon EMR su EKS crea il gruppo di log e il flusso di log se non esistono ancora, purché l'ARN del ruolo di esecuzione disponga delle autorizzazioni appropriate.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ],
      "Sid": "AllowLOGSCreatelogstream"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
      ],
      "Sid": "AllowLOGSPutlogevents"
    }
  ]
}
```

> **Note**
>
> Amazon EMR su EKS può anche creare un flusso di log. Se un flusso di log non esiste, la policy IAM deve includere l'autorizzazione `"logs:CreateLogGroup"`.

Dopo aver assegnato al ruolo di esecuzione le autorizzazioni appropriate, l'applicazione invia i dati di registro a CloudWatch Logs quando `cloudWatchMonitoringConfiguration` vengono passati nella `monitoringConfiguration` sezione di una `start-job-run` richiesta, come mostrato in. La gestione dei job viene eseguita con AWS CLI

Nell'`StartJobRun`API, *`log_group_name`* è il nome del gruppo di log e *`log_stream_prefix`* il prefisso del nome del flusso di log per CloudWatch. CloudWatch Puoi visualizzare e ricercare tali log in Console di gestione AWS.

- Registri dell'utente -*`logGroup`*/*`virtual-cluster-id`*/jobs/ /containers/*`logStreamPrefix`*/ (stderr/stdout*`job-id`*) *`pod-name`*
- Registri dei driver -*`logGroup`*/*`logStreamPrefixvirtual-cluster-id`*/jobs/ *`job-id`* / containers/ /spark- *`spark-application-id`* -driver/ *`job-id`* (stderrstdout)
- Registri degli esecutori - *`logGrouplogStreamPrefix`*/*`virtual-cluster-id`*/jobs/ *`job-id`* / containers/*`executor-pod-name`*/(*`spark-application-id`*stderr/stdout)

Elenco delle esecuzioni di processi

Puoi eseguire `list-job-run` per visualizzare gli stati delle esecuzioni di processi, come illustrato nell'esempio seguente.

```
aws emr-containers list-job-runs --virtual-cluster-id <cluster-id>
```

Descrizione di un'esecuzione di processo

Puoi eseguire `describe-job-run` per ottenere ulteriori dettagli sul processo, ad esempio lo stato del processo, i dettagli dello stato e il nome del processo, come illustrato nell'esempio seguente.

```
aws emr-containers describe-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Annullamento di un'esecuzione di processo

Puoi eseguire `cancel-job-run` per annullare i processi in esecuzione, come illustrato nell'esempio seguente.

```
aws emr-containers cancel-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Esecuzione degli script SQL StartJobRun di Spark tramite l'API

Amazon EMR su EKS versione 6.7.0 e successive include un nuovo driver di processi Spark SQL che permette di eseguire gli script Spark SQL attraverso l'API `StartJobRun`. PUoi fornire i file entry-point SQL per eseguire le query Spark SQL in Amazon EMR su EKS con l'API `StartJobRun`, senza modifiche agli script Spark SQL esistenti. La tabella seguente elenca i parametri Spark supportati per i job SQL di Spark tramite l'API. StartJobRun

Puoi scegliere tra i seguenti parametri Spark da inviare a un processo Spark SQL. Utilizza questi parametri per sovrascrivere le proprietà Spark predefinite.

Opzione	Description
--name NAME	Nome applicazione
--jars JARS	Elenco separato da virgole dei jar da includere nel classpath di driver ed esecuzione.
--packages	Elenco separato da virgole delle coordinate maven dei jar, da includere nei classpath di driver ed executor.
--exclude-packages	Elenco separato da virgole di groupId:artifactId , da escludere durante la risoluzione delle dipendenze fornite in –packages per evitare conflitti di dipendenze.
--repositories	Elenco separato da virgole di repository remoti aggiuntivi per la ricerca delle coordinate maven fornite con –packages.
--files FILES	Elenco separato da virgole di file da inserire nella directory di lavoro di ogni executor.
--conf PROP=VALUE	Proprietà di configurazione Spark.
--properties-file FILE	Percorso verso un file da cui caricare proprietà aggiuntive.

Opzione	Description
--driver-memory MEM	Memoria per il driver. Valore predefinito: 1.024 MB.
--driver-java-options	Opzioni Java extra da passare al driver.
--driver-library-path	Voci aggiuntive percorso libreria da passare al driver.
--driver-class-path	Voci aggiuntive classpath da passare al driver.
--executor-memory MEM	Memoria per ogni executor. Valore predefinito 1 GB.
--driver-cores NUM	Numero di core utilizzati dal driver.
-- NUM total-executor-cores	Numero totale di core per tutti gli executor.
--executor-cores NUM	Numero di core utilizzati da ogni executor.
--num-executors NUM	Numero di executor da avviare.
-hivevar <key=value>	Sostituzione di variabile da applicare ai comandi Hive, ad esempio `-hivevar A=B`
-hiveconf <property=value>	Valore da usare per la proprietà data.

Per un job SQL Spark, crea un start-job-run-request file.json e specifica i parametri richiesti per l'esecuzione del job, come nell'esempio seguente:

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

Stati delle esecuzioni di processi

Quando invii un'esecuzione di processo nella coda di processi di Amazon EMR su EKS, il suo stato diventa `PENDING`. Dopodiché passa per i seguenti stati fino a quando non ha esito positivo (codice di uscita `0`) o negativo (codice di uscita diverso da zero).

Le esecuzioni di processi possono avere i seguenti stati:

- `PENDING`: lo stato iniziale del processo quando l'esecuzione di processo viene inviata ad Amazon EMR su EKS. Il processo è in attesa di essere inviato al cluster virtuale e Amazon EMR su EKS sta lavorando per inviare questo processo.
- `SUBMITTED`: un'esecuzione di processo che è stata inviata correttamente al cluster virtuale. Il pianificatore di cluster tenta quindi di eseguire questo processo nel cluster.
- `RUNNING`: un processo in esecuzione nel cluster virtuale. Nelle applicazioni Spark, ciò significa che il processo del driver Spark si trova nello stato `running`.

- `FAILED`: un'esecuzione di processo che non è stato possibile inviare al cluster virtuale o il cui completamento ha avuto esito negativo. Guarda StateDetails e FailureReason per trovare ulteriori informazioni su questo errore di lavoro.
- `COMPLETED`: un'esecuzione di processo completata con successo.
- `CANCEL_PENDING`: un'esecuzione di processo per la quale è stato richiesto l'annullamento. Amazon EMR su EKS sta tentando di annullare il processo nel cluster virtuale.
- `CANCELLED`: un'esecuzione di processo annullata correttamente.

Visualizzazione dei processi nella console Amazon EMR

È possibile visualizzare i dati relativi all'esecuzione del job, in modo da poter monitorare ogni processo mentre attraversa gli stati. Per visualizzare i processi nella console Amazon EMR, completa la procedura seguente.

1. Nel menu a destra della console Amazon EMR, in Amazon EMR su EKS, seleziona Cloud virtuali.
2. Dall'elenco dei cluster virtuali seleziona il cluster virtuale per il quale desideri visualizzare i processi.
3. Nella tabella Esecuzioni dei processi, seleziona Visualizza log per visualizzare i dettagli di un'esecuzione di processo.

> **Note**
>
> Il supporto per l'esperienza con un clic è abilitato per impostazione predefinita. Può essere disattivato impostando `persistentAppUI` su `DISABLED` in `monitoringConfiguration` durante l'invio del processo. Per ulteriori informazioni, consulta Visualizzazione delle interfacce utente dell'applicazione persistente.

Errori comuni durante l'esecuzione di processi

Quando si esegue l'API `StartJobRun`, possono verificarsi i seguenti errori. La tabella elenca ogni errore e fornisce le misure di mitigazione in modo da poter risolvere rapidamente i problemi.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
errore: argomento -- *argument* è obbligatorio	Parametri obbligatori mancanti.	Aggiungere gli argomenti mancanti alla richiesta API.
Si è verificato un errore (AccessDeniedException) durante la chiamata dell' StartJobRunoperazione: User: *ARN* is not authorized to perform: emr-containers: StartJobRun	Ruolo di esecuzione mancante.	Consulta Utilizzo di Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS.
Si è verificato un errore (AccessDeniedException) durante la chiamata dell' StartJobRunoperazione: User: *ARN* is not authorized to perform: emr-containers: StartJobRun	Il chiamante non dispone dell'autorizzazione per il ruolo di esecuzione [formato valido/ non valido] tramite le chiavi di condizione.	Per informazioni, consulta Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS.
Si è verificato un errore (AccessDeniedException) durante la chiamata dell' StartJobRunoperazione: User: *ARN* is not authorized to perform: emr-containers: StartJobRun	Il mittente di processi e l'ARN del ruolo di esecuzion e provengono da account diversi.	Assicurati che il mittente di processi e l'ARN del ruolo di esecuzione provengano dallo stesso account AWS .
È stato rilevato 1 errore di convalida: il valore *Role* in 'executionRoleArn' non è riuscito a soddisfare il modello di espressione regolare ARN: ^arn :(aws [a-zA-Z0-9-] *) :iam: :(\ d {12})? : (ruolo ((\ u002F) | (\ u002F [\ u0021-\	Il chiamante dispone delle autorizzazioni per il ruolo di esecuzione tramite chiavi di condizione, ma il ruolo non soddisfa i vincoli del formato ARN.	Fornisci il ruolo di esecuzione seguendo il formato ARN. Per informazioni, consulta Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
u007F] +\ u002F)) [\ w+=, .@-] +)		
Si è verificato un StartJobRun errore () durante la chiamata dell'operazione: Il cluster virtuale non esiste. ResourceNotFoundException *`Virtual Cluster ID`*	L'ID del cluster virtuale non è stato trovato.	Fornisci un ID del cluster virtuale registrato con Amazon EMR su EKS.
Si è verificato un errore (ValidationException) durante la chiamata dell' StartJobRunoperazione: lo stato del cluster virtuale non *`state`* è valido per creare la risorsa JobRun.	Il cluster virtuale non è pronto per eseguire il processo.	Per informazioni, consulta Stati dei cluster virtuali.
Si è verificato un errore (ResourceNotFoundException) durante la chiamata dell' StartJobRunoperazione: Release *`RELEASE`* doesn't exist.	Il rilascio specificato nell'invio del processo non è corretto.	Per informazioni, consulta Rilasci di Amazon EMR su EKS.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
Si è verificato un errore (AccessDeniedException) durante la chiamata all' StartJobRunoperazione: User: *ARN* is not authorized to perform: emr-containers: StartJobRun on resource: *ARN* con una negazione esplicita. Si è verificato un errore (AccessDeniedException) durante la chiamata all' StartJobRunoperazione: User: *ARN* is not authorized to perform: emr-containers: on resource: StartJobRun *ARN*	L'utente non è autorizzato a chiamare. StartJobRun	Per informazioni, consulta Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS.
Si è verificato un errore (ValidationException) durante la chiamata dell' StartJobR unoperazione: configura tionOverrides.MonitoringCon figuration.s3 Monitorin gConfiguration .LogURI non è riuscito a soddisfare il vincolo: %s	Sintassi URI del percorso S3 non valida.	logUri dovrebbe essere nel formato di s3://...

Quando si esegue l'API di `DescribeJobRun` prima di eseguire un processo, possono verificarsi i seguenti errori.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
JobRun StateDetails: invio non riuscito.	I parametri inseriti non StartJobRun sono validi.	Per informazioni, consulta Rilasci di Amazon EMR su EKS.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
Classificazione *classification* non supportata. failureReason: VALIDATION_ERROR (ERRORE DI CONVALIDA) state: FAILED (NON RIUSCITO)		
StateDetails: il cluster *EKS Cluster ID* non esiste. failureReason: CLUSTER_UNAVAILABLE (CLUSTER NON DISPONIBILE) state: FAILED (NON RIUSCITO)	Il cluster EKS non è disponibile.	Verifica se il cluster EKS esiste e dispone delle autorizzazioni corrette. Per ulteriori informazioni, consulta Configurazione di Amazon EMR su EKS.
StateDetails: il cluster *EKS Cluster ID* non dispone di autorizzazioni sufficienti. failureReason: CLUSTER_UNAVAILABLE (CLUSTER NON DISPONIBILE) state: FAILED (NON RIUSCITO)	Amazon EMR non dispone delle autorizzazioni per accedere al cluster EKS.	Verifica che le autorizzazioni siano impostate per Amazon EMR nello spazio dei nomi registrato. Per ulteriori informazioni, consulta Configurazione di Amazon EMR su EKS.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
StateDetails: Il cluster *EKS Cluster ID* non è attualmente raggiungibile. failureReason: CLUSTER_UNAVAILABLE (CLUSTER NON DISPONIBILE) state: FAILED (NON RIUSCITO)	Il cluster EKS non è raggiungibile.	Verifica se il cluster EKS esiste e dispone delle autorizzazioni corrette. Per ulteriori informazioni, consulta Configurazione di Amazon EMR su EKS.
StateDetails: JobRun invio non riuscito a causa di un errore interno. failureReason: INTERNAL_ERROR (ERRORE INTERNO) state: FAILED (NON RIUSCITO)	Si è verificato un errore interno nel cluster EKS.	N/D
StateDetails: il cluster *EKS Cluster ID* non dispone di risorse sufficienti. failureReason: USER_ERROR (ERRORE UTENTE) state: FAILED (NON RIUSCITO)	Le risorse presenti nel cluster EKS sono insufficienti per eseguire il processo.	Aggiungi più capacità al gruppo di nodi EKS o imposta EKS Autoscaler. Per ulteriori informazioni, consulta Cluster Autoscaler.

Quando si esegue l'API di `DescribeJobRun` dopo aver eseguito un processo, possono verificarsi i seguenti errori.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
StateDetails: Problemi nel monitoraggio del tuo. JobRun *EKS Cluster ID*Il cluster non esiste. failureReason: CLUSTER_UNAVAILABLE (CLUSTER NON DISPONIBILE) state: FAILED (NON RIUSCITO)	Il cluster EKS non esiste.	Verifica se il cluster EKS esiste e dispone delle autorizzazioni corrette. Per ulteriori informazioni, consulta Configurazione di Amazon EMR su EKS.
StateDetails: Problemi nel monitoraggio del tuo. JobRun *EKS Cluster ID*Il cluster non dispone di autorizzazioni sufficienti. failureReason: CLUSTER_UNAVAILABLE (CLUSTER NON DISPONIBILE) state: FAILED (NON RIUSCITO)	Amazon EMR non dispone delle autorizzazioni per accedere al cluster EKS.	Verifica che le autorizzazioni siano impostate per Amazon EMR nello spazio dei nomi registrato. Per ulteriori informazioni, consulta Configurazione di Amazon EMR su EKS.
StateDetails: Problemi nel monitoraggio del tuo. JobRun *EKS Cluster ID*Il cluster non è attualmente raggiungibile. failureReason: CLUSTER_UNAVAILABLE (CLUSTER NON DISPONIBILE)	Il cluster EKS non è raggiungibile.	Verifica se il cluster EKS esiste e dispone delle autorizzazioni corrette. Per ulteriori informazioni, consulta Configurazione di Amazon EMR su EKS.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
state: FAILED (NON RIUSCITO)		
StateDetails: problemi di monitoraggio JobRun dovuti a un errore interno failureReason: INTERNAL_ERROR (ERRORE INTERNO) state: FAILED (NON RIUSCITO)	Si è verificato un errore interno che impedisce JobRun il monitoraggio.	N/D

Il seguente errore può verificarsi quando un processo non può essere avviato e il processo rimane in attesa nello stato INVIATO per 15 minuti. Ciò può essere causato dalla mancanza di risorse del cluster.

Messaggio di errore	Condizione di errore	Fase successiva consigliata
timeout del cluster	Il lavoro è rimasto nello stato INVIATO per 15 minuti o più.	È possibile sovrascrivere l'impostazione predefinita di 15 minuti per questo parametro con l'override della configura zione mostrato di seguito.

Utilizza la seguente configurazione per modificare l'impostazione predefinita di 30 minuti per il timeout del cluster. Tieni presente il fatto che il nuovo valore `job-start-timeout` è fornito in secondi:

```
{
"configurationOverrides": {
  "applicationConfiguration": [{
      "classification": "emr-containers-defaults",
      "properties": {
          "job-start-timeout":"1800"
      }
  }]
```

```
}
```

Utilizzo dei modelli di processo

Un modello di processo memorizza valori che possono essere condivisi tra le chiamate API `StartJobRun` quando si avvia l'esecuzione di un processo. Supporta due casi d'uso:

- Per evitare che si ripetano valori di richiesta API `StartJobRun` ricorrenti.
- Per far rispettare una regola secondo cui determinati valori devono essere forniti tramite richieste API `StartJobRun`.

I modelli di processo consentono di definire un modello riutilizzabile per le esecuzioni dei processi per applicare ulteriori personalizzazioni, ad esempio:

- Configurazione della capacità di elaborazione dell'esecutore e del driver
- Impostazione di proprietà di sicurezza e governance come i ruoli IAM
- Personalizzazione di un'immagine Docker da utilizzare in più applicazioni e pipeline di dati

I seguenti argomenti forniscono informazioni dettagliate sull'utilizzo dei modelli, incluso come utilizzarli per avviare un processo e come modificare i parametri dei modelli.

Argomenti

Creare e utilizzare un modello di processo per avviare l'esecuzione di un processo

Questa sezione descrive la creazione di un modello di lavoro e l'utilizzo del modello per avviare un processo eseguito con AWS Command Line Interface (AWS CLI).

Per creare un modello di processo

1. Crea un file `create-job-template-request.json` e specifica i parametri richiesti per l'esecuzione di modello, come illustrato nel file JSON di esempio seguente. Per informazioni su tutti i parametri disponibili, consulta l'CreateJobTemplateAPI.

   La maggior parte dei valori richiesti per l'API `StartJobRun` sono necessari anche per `jobTemplateData`. Se desideri utilizzare i segnaposto per qualsiasi parametro e fornire valori quando richiami StartJobRun utilizzando un modello di lavoro, consulta la sezione successiva sui parametri del modello di lavoro.

```
{
   "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. Utilizza il comando `create-job-template` con un percorso per il file `create-job-template-request.json` archiviato localmente.

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

Per avviare un'esecuzione di processo utilizzando un modello di processo

Fornisci l'ID di cluster virtuale, l'ID modello di processo e il nome di processo nel comando `StartJobRun`, come mostrato nell'esempio seguente.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--job-template-id 1234abcd
```

Definizione di parametri di processo

I parametri del modello di processo consentono di specificare variabili nel modello di processo. I valori per queste variabili di parametro dovranno essere specificati quando si avvia un'esecuzione di processo utilizzando quel modello di processo. I parametri del modello di processo sono specificati nel formato `${parameterName}`. Puoi scegliere di specificare qualsiasi valore in un campo `jobTemplateData` come parametro del modello di processo. Per ciascuna delle variabili dei parametri del modello di processo, specifica il tipo di dati (`STRING` o `NUMBER`) e, facoltativamente, un valore predefinito. L'esempio seguente mostra come specificare i parametri del modello di processo per la posizione del punto di ingresso, la classe principale e i valori della posizione del log S3.

Per specificare la posizione del punto di ingresso, la classe principale e la posizione del log di Amazon S3 come parametri del modello di processo

1. Crea un file `create-job-template-request.json` e specifica i parametri richiesti per l'esecuzione di modello, come illustrato nel file JSON di esempio seguente. Per ulteriori informazioni sui parametri, consulta l'API. CreateJobTemplate

```
{
```

```
  "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. Utilizza il comando `create-job-template` con un percorso per il file `create-job-template-request.json` archiviato localmente o in Amazon S3.

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

Per avviare un'esecuzione di processo utilizzando un modello di processo con parametri di processo

Per avviare un processo eseguito con un modello di processo contenente i parametri del modello di processo, specifica l'id del modello di processo e i valori per i parametri del modello di processo nella richiesta API `StartJobRun`, come mostrato di seguito.

```
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/"}'
```

Controllo dell'accesso ai modelli di processo

La policy `StartJobRun` consente di imporre che un utente o un ruolo possa eseguire processi solo utilizzando modelli di processo specificati dall'utente e non eseguire operazioni `StartJobRun` senza utilizzare i modelli di processo specificati. A tal fine, assicurati innanzitutto di concedere all'utente o al ruolo un'autorizzazione di lettura per i modelli di processo specificati, come mostrato di seguito.

JSON

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

```
        "emr-containers:DescribeJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:*:jobtemplate/job_template_1_id",
        "arn:aws:emr-containers:*:*:jobtemplate/job_template_2_id"
      ],
      "Sid": "AllowEMRCONTAINERSDescribejobtemplate"
    }
  ]
}
```

Per far sì che un utente o un ruolo sia in grado di invocare un'operazione `StartJobRun` solo quando si utilizzano modelli di processo specificati, è possibile assegnare la seguente autorizzazione della policy `StartJobRun` a un determinato utente o ruolo.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:StartJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:*:/virtualclusters/virtual_cluster_id"
      ],
      "Condition": {
        "ArnLike": {
          "emr-containers:JobTemplateArn": [
            "arn:aws:emr-containers:*:*:jobtemplate/job_template_1_id",
            "arn:aws:emr-containers:*:*:jobtemplate/job_template_2_id"
          ]
        }
      },
      "Sid": "AllowEMRCONTAINERSStartjobrun"
    }
  ]
}
```

Se il modello di processo specifica un parametro del modello di processo all'interno del campo ARN del ruolo di esecuzione, l'utente sarà in grado di fornire un valore per questo parametro e quindi potrà richiamare `StartJobRun` utilizzando un ruolo di esecuzione arbitrario. Per limitare i ruoli di esecuzione che l'utente può fornire, consulta Controllo dell'accesso al ruolo di esecuzione in Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS.

Se non viene specificata alcuna condizione nella policy di operazione `StartJobRun` di cui sopra per un determinato utente o ruolo, l'utente o il ruolo potranno invocare un'operazione `StartJobRun` sul cluster virtuale specificato utilizzando un modello di processo arbitrario a cui hanno accesso in lettura o utilizzando un ruolo di esecuzione arbitrario.

Uso dei modelli di pod

A partire da Amazon EMR versioni 5.33.0 o 6.3.0, Amazon EMR su EKS supporta la funzione dei modelli di pod di Spark. Un pod è un gruppo di uno o più container, con risorse di rete e archiviazione condivise e una specifica per l'esecuzione dei container. I modelli di pod sono specifiche che determinano la modalità di esecuzione di ciascun pod. È possibile utilizzare i file dei modelli di pod per definire le configurazioni dei pod driver o executor non supportate dalle configurazioni Spark. Per ulteriori informazioni sulla funzione dei modelli di pod di Spark, consulta Modelli di pod.

> **Note**
>
> La funzione dei modelli di pod funziona solo con i pod driver ed executor. Non è possibile configurare i pod Job Submitter utilizzando il modello di pod.

Scenari comuni

È possibile definire come eseguire i processi Spark in cluster EKS condivisi utilizzando i modelli di pod con Amazon EMR su EKS e, al contempo, risparmiare sui costi e migliorare l'utilizzo e le prestazioni delle risorse.

- Per ridurre i costi, puoi pianificare le attività dei driver Spark da eseguire su istanze Amazon EC2 On-Demand mentre pianificare le attività di esecuzione di Spark da eseguire su istanze Amazon Spot. EC2
- Per incrementare l'utilizzo delle risorse, è possibile supportare più team che eseguono i propri carichi di lavoro nello stesso cluster EKS. Ogni team riceverà un gruppo di EC2 nodi Amazon

designato su cui eseguire i propri carichi di lavoro. È possibile utilizzare i modelli di pod per applicare una tolleranza corrispondente al carico di lavoro.

- Per migliorare il monitoraggio, è possibile eseguire un container di log separato per inoltrare i log all'applicazione di monitoraggio esistente.

Ad esempio, il file del modello pod riportato di seguito illustra uno scenario di utilizzo comune.

```
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>
```

Il modello di pod completa le attività seguenti:

- Aggiungi un nuovo container init da eseguire prima dell'avvio del container principale Spark. Il contenitore init condivide il EmptyDirvolume chiamato `source-data-volume` con il contenitore principale di Spark. È possibile fare in modo che il container init esegua i passaggi di inizializzazione, come il download di dipendenze o la generazione di dati di input. In seguito, il container principale Spark consuma i dati.
- Aggiungi un altro container sidecar da eseguire insieme al container principale Spark. I due container stanno condividendo un altro volume `EmptyDir` chiamato`metrics-files-volume`. Il processo Spark può generare parametri quali i parametri Prometheus. In seguito il processo Spark può inserire i parametri in un file e fare in modo che il container sidecar carichi i file nel proprio sistema BI per analisi future.
- Aggiungi una nuova variabile ambiente al container principale Spark. Puoi fare in modo che il processo utilizzi la variabile ambiente.
- Definisci un selettore di nodi, in modo che il pod sia pianificato solo sul gruppo di nodi `emr-containers-nodegroup`. Ciò consente di isolare le risorse di calcolo tra i processi e i team.

Attivazione di modelli di pod con Amazon EMR su EKS

Per abilitare la funzione dei modelli di pod con Amazon EMR su EKS, configura le proprietà Spark `spark.kubernetes.driver.podTemplateFile` e `spark.kubernetes.executor.podTemplateFile` per puntare ai file dei modelli di pod in Amazon S3. Spark scarica quindi il file del modello di pod e lo utilizza per costruire pod driver ed executor.

> **Note**
>
> Spark utilizza il ruolo di esecuzione del processo per caricare il modello di pod, pertanto il ruolo di esecuzione del processo deve disporre delle autorizzazioni per accedere ad Amazon S3 per caricare i modelli di pod. Per ulteriori informazioni, consulta Creazione di un ruolo di esecuzione di processo.

È possibile utilizzare `SparkSubmitParameters` per specificare il percorso Amazon S3 al modello di pod, come dimostra il seguente file JSON dell'esecuzione di processo.

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

In alternativa, è possibile utilizzare `configuration0verrides` per specificare il percorso Amazon S3 al modello di pod, come dimostra il seguente file JSON dell'esecuzione di processo.

```
{
  "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. Quando si utilizza la funzione dei modelli di pod con Amazon EMR su EKS, è necessario seguire le linee guida di sicurezza, ad esempio isolando un eventuale codice dell'applicazione non attendibile. Per ulteriori informazioni, consulta Best practice di sicurezza per Amazon EMR su EKS.
> 2. Non è possibile modificare i nomi dei container principali Spark utilizzando `spark.kubernetes.driver.podTemplateContainerName` e `spark.kubernetes.executor.podTemplateContainerName`, perché questi nomi sono codificati come `spark-kubernetes-driver` e `spark-kubernetes-executors`. Se si desidera personalizzare il container principale Spark, è necessario specificarlo in un modello di pod con questi nomi codificati.

Campi del modello di pod

Quando configuri un modello di pod con Amazon EMR su EKS, prendi in considerazione le seguenti restrizioni sul campo.

- Amazon EMR su EKS accetta solo i seguenti campi in un modello di pod per consentire una corretta pianificazione dei processi.

  Seguono i campi a livello di pod consentiti:

  - `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`

Seguono i campi a livello di container principale Spark consentiti:

- `env`
- `envFrom`
- `name`
- `lifecycle`
- `livenessProbe`
- `readinessProbe`
- `resources`
- `startupProbe`
- `stdin`
- `stdinOnce`
- `terminationMessagePath`
- `terminationMessagePolicy`
- `tty`
- `volumeDevices`
- `volumeMounts`
- `workingDir`

Quando si utilizzano campi non consentiti nel modello di pod, Spark genera un'eccezione e il processo ha esito negativo. L'esempio seguente mostra un messaggio di errore nel log del controller Spark a causa di campi non consentiti.

```
Executor pod template validation failed.
Field container.command in Spark main container not allowed but specified.
```

- Amazon EMR su EKS predefinisce i seguenti parametri in un modello di pod. I campi specificati in un modello di pod non devono sovrapporsi a questi campi.

  Seguono i nomi predefiniti dei volumi:

  - `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`

Seguono i montaggi di volume predefiniti che si applicano solo al container principale Spark:

- Nome:;: `emr-container-communicate` MountPath `/var/log/fluentd`
- Nome:`emr-container-application-log-dir`; MountPath: `/var/log/spark/user`
- Nome:`emr-container-event-log-dir`; MountPath: `/var/log/spark/apps`
- Nome:`mnt-dir`; MountPath: `/mnt`
- Nome:`temp-data-dir`; MountPath: `/tmp`
- Nome:`home-dir`; MountPath: `/home/hadoop`

Seguono le variabili ambiente predefinite che si applicano solo al container principale Spark:

- `SPARK_CONTAINER_ID`
- `K8S_SPARK_LOG_URL_STDERR`
- `K8S_SPARK_LOG_URL_STDOUT`
- `SIDECAR_SIGNAL_FILE`

> **Note**
>
> È comunque possibile utilizzare questi volumi predefiniti e montarli nei container sidecar aggiuntivi. Ad esempio, è possibile utilizzare `emr-container-application-log-dir` e montarlo sul proprio container sidecar definito nel modello di pod.

Se i campi specificati sono in conflitto con uno qualsiasi dei campi predefiniti nel modello di pod, Spark genera un'eccezione e il processo ha esito negativo. L'esempio seguente mostra un messaggio di errore nel log dell'applicazione Spark a causa di conflitti con i campi predefiniti.

```
Defined volume mount path on main container must not overlap with reserved mount
 paths: [<reserved-paths>]
```

Considerazioni sui container sidecar

Amazon EMR controlla il ciclo di vita dei pod forniti da Amazon EMR su EKS. I container sidecar devono seguire lo stesso ciclo di vita del container principale Spark. Se inserisci ulteriori container sidecar nei tuoi pod, ti consigliamo di integrare la gestione del ciclo di vita dei pod definita da Amazon EMR in modo che il container sidecar possa bloccarsi quando il container principale Spark esce.

Per ridurre i costi, è consigliabile implementare un processo che impedisce ai pod driver con container sidecar di continuare a funzionare al termine del processo. Il driver Spark elimina i pod executor quando l'executor viene terminato. Tuttavia, al termine di un programma driver, i container sidecar aggiuntivi continuano a funzionare. Il pod viene fatturato fino a quando Amazon EMR su EKS non elimina il pod driver, generalmente meno di un minuto dopo il completamento del container principale del driver Spark. Per ridurre i costi, è possibile integrare i container sidecar aggiuntivi con il meccanismo di gestione del ciclo di vita definito da Amazon EMR su EKS sia per i pod driver che per i pod executor, come descritto nella sezione seguente.

Il container principale Spark nei pod driver ed executor invia `heartbeat` a un file `/var/log/fluentd/main-container-terminated` ogni due secondi. Aggiungendo il montaggio del volume `emr-container-communicate` predefinito di Amazon EMR sul container sidecar, è possibile definire un sottoprocesso del container sidecar per tenere traccia periodicamente dell'ora dell'ultima modifica di questo file. Il sottoprocesso si arresta se scopre che il container principale Spark arresta il `heartbeat` per una durata di tempo maggiore.

Nell'esempio seguente viene illustrato un sottoprocesso che tiene traccia del file heartbeat e si arresta. Sostituisci *`your_volume_mount`* con il percorso in cui monti il volume predefinito. Lo script è raggruppato all'interno dell'immagine utilizzata dal container sidecar. In un file del modello di pod, è possibile specificare un container sidecar con i seguenti comandi `sub_process_script.sh` e `main_command`.

```
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
```

Uso delle policy di ripetizione dei processi

Nella versione 6.9.0 e successive di Amazon EMR su EKS, puoi impostare una policy di ripetizione per le esecuzioni dei processi. Con le policy di ripetizione, il pod driver di un processo viene riavviato

in automatico in caso di errori o se viene eliminato. Ciò incrementa la resilienza agli errori dei processi di streaming Spark di lunga durata.

Impostazione della policy di ripetizione per un processo

Per configurare una politica di riprova, fornisci un `RetryPolicyConfiguration` campo utilizzando l'StartJobRunAPI. Di seguito è mostrato un esempio di `retryPolicyConfiguration`:

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

> **Note**
>
> `retryPolicyConfiguration`è disponibile solo dalla versione AWS CLI 1.27.68 in poi. Per aggiornare il file AWS CLI alla versione più recente, vedere Installazione o aggiornamento della versione più recente di AWS CLI

Configura il campo `maxAttempts` con il numero massimo di volte per cui desideri che il pod driver del processo venga riavviato in caso di errore o di eliminazione. L'intervallo di esecuzione tra due tentativi di ripetizione del driver del processo è un intervallo di ripetizione esponenziale di (10 secondi, 20 secondi, 40 secondi...) limitato a 6 minuti, come descritto nella Documentazione di Kubernetes.

> **Note**
>
> Ogni esecuzione aggiuntiva del driver del processo verrà fatturata come un'ulteriore esecuzione del processo e sarà soggetta ai Prezzi di Amazon EMR su EKS.

Valori di configurazione della policy di ripetizione

- Policy di ripetizione predefinita per un processo: `StartJobRun` include una policy di ripetizione impostata su 1 tentativo massimo per impostazione predefinita. Puoi configurare la policy di ripetizione in base alle tue necessità.

  > **Note**
  >
  > Se `maxAttempts` di `retryPolicyConfiguration` è impostato su 1, in caso di errori non verrà effettuato alcun nuovo tentativo di apertura del pod driver.

- Disabilitazione della politica di nuovi tentativi per un processo: per disabilitare una politica di nuovi tentativi, imposta il valore massimo di tentativi su 1. retryPolicyConfiguration

```
"retryPolicyConfiguration": {
    "maxAttempts": 1
}
```

- Imposta maxAttempts per un processo nell'intervallo valido: la chiamata di `StartJobRun` avrà esito negativo se il valore `maxAttempts` non rientra nell'intervallo valido. L'intervallo `maxAttempts` valido è compreso tra 1 e 2.147.483.647 (numero intero a 32 bit), l'intervallo supportato per l'impostazione della configurazione `backOffLimit` di Kubernetes. Per ulteriori informazioni, consulta Policy degli errori backoff dei pod nella Documentazione di Kubernetes. Se il valore `maxAttempts` non è valido, viene restituito il seguente messaggio di errore:

```
{
 "message": "Retry policy configuration's parameter value of maxAttempts is invalid"
```

```
}
```

Recupero di uno stato della policy di ripetizione per un processo

È possibile visualizzare lo stato dei nuovi tentativi di esecuzione di un lavoro con and. [ListJobRunsDescribeJobRun](#) APIs Non appena richiedi un processo con una configurazione di policy di ripetizione abilitata, le risposte di `ListJobRun` e `DescribeJobRun` conterranno lo stato della policy di ripetizione nel campo `RetryPolicyExecution`. Inoltre, la risposta di `DescribeJobRun` conterrà il valore `RetryPolicyConfiguration` inserito nella richiesta `StartJobRun` del processo.

Esempi di risposte

ListJobRuns response

```
{
  "jobRuns": [
    ...
    ...
    "retryPolicyExecution" : {
      "currentAttemptCount": 2
    }
    ...
    ...
  ]
}
```

DescribeJobRun response

```
{
  ...
  ...
  "retryPolicyConfiguration": {
    "maxAttempts": 5
   },
   "retryPolicyExecution" : {
    "currentAttemptCount": 2
  },
  ...
  ...
```

```
}
```

Questi campi non saranno visibili quando la policy di ripetizione è disabilitata nel processo, come descritto di seguito in Valori di configurazione della policy di ripetizione.

Monitoraggio di un processo con una policy di ripetizione

Quando si abilita una politica di nuovi tentativi, viene generato un CloudWatch evento per ogni job driver creato. Per sottoscrivere questi eventi, impostate una regola di CloudWatch evento utilizzando il seguente comando:

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run New Driver Attempt"]}'
```

L'evento restituirà informazioni su `newDriverPodName`, timestamp `newDriverCreatedAt`, `previousDriverFailureMessage` e `currentAttemptCount` sui driver dei processi. Questi eventi non verranno creati se la policy di ripetizione è disabilitata.

Per ulteriori informazioni su come monitorare il lavoro con CloudWatch gli eventi, consultaMonitora i lavori con Amazon CloudWatch Events.

Ricerca di log per driver ed executor

I nomi dei pod driver seguono il formato `spark-<job id>-driver-<random-suffix>`. Lo stesso valore `random-suffix` viene aggiunto ai nomi dei pod dell'executor generati dal driver. Utilizzando questo valore `random-suffix`, puoi trovare i log di un driver e degli executor associati. `random-suffix` è presente solo se la policy di ripetizione è abilitata per il processo; in caso contrario, `random-suffix` è assente.

Per ulteriori informazioni sulla modalità di configurazione di processi con la configurazione di monitoraggio per la creazione di log, consulta Eseguire un'applicazione Spark.

Uso della rotazione dei log di eventi Spark

Con Amazon EMR 6.3.0 e versioni successive, è possibile attivare la funzionalità di rotazione dei log di eventi Spark per Amazon EMR su EKS. Anziché generare un singolo file di log di eventi, questa

funzionalità ruota il file in base all'intervallo di tempo configurato e rimuove i file di log di eventi meno recenti.

La rotazione dei log di eventi Spark consente di evitare potenziali problemi con un file di log eventi Spark di grandi dimensioni generato per lunghi processi di streaming o di esecuzione. Supponiamo, ad esempio, di avviare un processo Spark di lunga durata con un log eventi abilitato con il parametro `persistentAppUI`. Il driver Spark genera un file di log eventi. Se il processo viene eseguito per ore o giorni e lo spazio su disco è limitato nel nodo Kubernetes, il file di log eventi può consumare tutto lo spazio disponibile su disco. Attivando la funzionalità di rotazione dei log di eventi Spark, il problema viene risolto dividendo il file di log in più file e rimuovendo i file meno recenti.

> **Note**
>
> Questa funzionalità è disponibile solo con Amazon EMR su EKS. Amazon EMR in esecuzione su Amazon EC2 non supporta la rotazione dei log degli eventi Spark.

Per attivare la funzionalità di rotazione dei log di eventi Spark, configura i seguenti parametri Spark:

- `spark.eventLog.rotation.enabled`: attiva la rotazione dei log. Questa opzione è disabilitata di default nel file di configurazione di Spark. Impostalo su true (vero) per attivare questa funzionalità.
- `spark.eventLog.rotation.interval`: specifica l'intervallo di tempo per la rotazione dei log. Il valore minimo è 60 secondi. Il valore predefinito è 300 secondi.
- `spark.eventLog.rotation.minFileSize`: specifica una dimensione minima del file per ruotare il file di log. Il valore minimo e predefinito è 1 MB.
- `spark.eventLog.rotation.maxFilesToRetain`: specifica il numero di file di log ruotati da mantenere durante la pulizia. L'intervallo consentito è da 1 a 10. Il valore predefinito è 2.

Puoi specificare questi parametri nella sezione `sparkSubmitParameters` dell'API di `StartJobRun`, come illustrato nell'esempio seguente.

```
"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"
```

Uso della rotazione dei log di container Spark

Con Amazon EMR 6.11.0 e versioni successive, è possibile attivare la funzionalità di rotazione dei log di container Spark per Amazon EMR su EKS. Anziché generare un singolo file di log `stdout` o `stderr`, questa funzionalità ruota il file in base alle dimensioni di rotazione configurate e rimuove i file di log meno recenti dal container.

La rotazione dei log di container Spark consente di evitare potenziali problemi dovuti a file di log Spark di grandi dimensioni generati per processi di streaming o di lunga durata. Ad esempio, potresti avviare un processo Spark di lunga durata e in tal caso il driver Spark creerebbe un file di log di container. Se il processo viene eseguito per ore o giorni e lo spazio su disco nel nodo Kubernetes è limitato, il file di log di container può consumare tutto lo spazio disponibile su disco. Attivando la rotazione dei log di container Spark, dividi il file di log in più file e rimuovi quelli meno recenti.

Per attivare la funzionalità di rotazione dei log di container Spark, configura i seguenti parametri Spark:

`containerLogRotationConfiguration`

Includi questo parametro in `monitoringConfiguration` per attivare la rotazione dei log. L'opzione è disabilitata per impostazione predefinita. È necessario utilizzare `containerLogRotationConfiguration` in aggiunta a `s3MonitoringConfiguration`.

`rotationSize`

Il parametro `rotationSize` specifica la dimensione del file per la rotazione dei log. L'intervallo di valori possibili è compreso tra `2KB` e `2GB`. La parte relativa all'unità numerica del parametro `rotationSize` viene trasmessa come numero intero. Poiché i valori decimali non sono supportati, puoi specificare una dimensione di rotazione di 1,5 GB, ad esempio, con il valore `1500MB`.

`maxFilesToKeep`

Il parametro `maxFilesToKeep` specifica il numero massimo di file da mantenere nel container in seguito alla rotazione. Il valore minimo è 1, quello massimo è 50.

Puoi specificare questi parametri nella sezione `monitoringConfiguration` dell'API di `StartJobRun`, come illustrato nell'esempio seguente. In questo esempio, con `rotationSize = "10 MB"` e `maxFilesToKeep = 3`, Amazon EMR su EKS ruota i log a 10 MB, genera un nuovo file di log e quindi elimina il file di log meno recente una volta che il numero di file di log arriva a 3.

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

Per avviare un processo con la rotazione dei log di container Spark, includi un percorso del file json che hai configurato con questi parametri nel comando `StartJobRun`.

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

Uso del dimensionamento automatico verticale con i processi Spark di Amazon EMR

Il dimensionamento automatico verticale di Amazon EMR su EKS ottimizza in automatico le risorse di memoria e CPU per adattarle alle esigenze del carico di lavoro fornito per le applicazioni Spark di Amazon EMR. Ciò semplifica la gestione delle risorse.

Per tracciare l'utilizzo cronologico e in tempo reale delle risorse delle tue applicazioni Spark di Amazon EMR, il dimensionamento automatico verticale sfrutta il Vertical Pod Autoscaler (VPA) di Kubernetes. La funzionalità di dimensionamento automatico verticale utilizza i dati raccolti da VPA per ottimizzare in automatico le risorse di memoria e CPU assegnate alle applicazioni Spark. Questo processo semplificato migliora l'affidabilità e ottimizza i costi.

Argomenti

Configurazione del dimensionamento automatico verticale per Amazon EMR su EKS

Questo argomento ti aiuta a preparare il cluster Amazon EKS per inviare i processi Spark di Amazon EMR con dimensionamento automatico verticale. Il processo di configurazione richiede la conferma o il completamento delle attività nelle seguenti sezioni:

Argomenti

Prerequisiti

Completa le seguenti attività prima di installare l'operatore Kubernetes con scalabilità automatica verticale sul cluster. Se hai già completato uno dei prerequisiti, puoi saltarli e passare a quello successivo.

- Installa o aggiorna alla versione più recente di AWS CLI: se hai già installato il AWS CLI, conferma di disporre della versione più recente.
- Installa kubectl: kubectl è uno strumento a riga di comando utilizzato per comunicare con il server di API Kubernetes. kubectl è necessario per installare e monitorare gli artefatti relativi al dimensionamento automatico verticale nel cluster Amazon EKS.
- Installa Operator SDK: Amazon EMR su EKS utilizza Operator SDK come gestore di pacchetti per l'intera durata dell'operatore di dimensionamento automatico verticale che installi sul cluster.
- Installa Docker: devi accedere alla CLI Docker per autenticare e recuperare le immagini Docker verticali relative al dimensionamento automatico da installare sul cluster Amazon EKS.
- Installa il server Kubernetes Metrics: devi prima installare il server Metrics in modo che il pod autoscaler verticale possa recuperare le metriche dal server dell'API Kubernetes.
- Inizia a usare Amazon EKS — eksctl (versione 1.24 o successiva) — La scalabilità automatica verticale è supportata dalle versioni 1.24 e successive di Amazon EKS. Una volta creato il cluster, registralo per utilizzarlo con Amazon EMR.
- Seleziona un URI dell'immagine di base Amazon EMR (rilascio 6.10.0 o successivo): il dimensionamento automatico verticale è supportato con i rilasci 6.10.0 e successivi di Amazon EMR.

Installazione di Operator Lifecycle Manager (OLM) sul cluster Amazon EKS

Utilizza la CLI di Operator SDK per installare Operator Lifecycle Manager (OLM) nel cluster Amazon EMR su EKS in cui desideri configurare il dimensionamento automatico verticale, come mostrato nell'esempio seguente. Una volta configurato, puoi utilizzare OLM per installare e gestire il ciclo di vita dell'operatore di dimensionamento automatico verticale di Amazon EMR.

```
operator-sdk olm install
```

Per confermare l'installazione, esegui il comando `olm status`:

```
operator-sdk olm status
```

Verifica che il comando restituisca un risultato corretto, simile al seguente output di esempio:

```
INFO[0007] Successfully got OLM status for version X.XX
```

Se l'installazione ha esito negativo, consulta Risoluzione dei problemi relativi al dimensionamento automatico verticale di Amazon EMR su EKS.

Installazione dell'operatore di dimensionamento automatico verticale di Amazon EMR su EKS

Utilizza le fasi seguenti per installare l'operatore di dimensionamento automatico verticale sul cluster Amazon EKS:

1. Configura le seguenti variabili di ambiente che utilizzerai per completare l'installazione:
   - **$REGION** indica la Regione AWS del tuo cluster. Ad esempio, `us-west-2`.
   - **$ACCOUNT_ID** punta indica l'ID dell'account Amazon ECR della tua Regione. Per ulteriori informazioni, consulta Account di registro Amazon ECR per Regione.
   - **$RELEASE** punta indica la versione di Amazon EMR che desideri utilizzare per il cluster. Con il dimensionamento automatico verticale, è necessario utilizzare la versione 6.10.0 o successiva di Amazon EMR.
2. Successivamente, ottieni i token di autenticazione al registro Amazon ECR per l'operatore.

   ```
   aws ecr get-login-password \
    --region region-id | docker login \
    --username AWS \
    --password-stdin $ACCOUNT_ID.dkr.ecr.region-id.amazonaws.com
   ```
3. Installa l'operatore di dimensionamento automatico verticale di Amazon EMR su EKS con il comando seguente:

   ```
   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
   ```

Questo creerà una versione dell'operatore di dimensionamento automatico verticale nello spazio dei nomi predefinito del tuo cluster Amazon EKS. Utilizza questo comando per l'installazione in uno spazio dei nomi diverso:

```
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**
>
> Se lo spazio dei nomi specificato non esiste, OLM non installerà l'operatore. Per ulteriori informazioni, consulta Impossibile trovare lo spazio dei nomi Kubernetes.

4. Verifica di aver installato correttamente l'operatore con lo strumento a riga di comando Kubernetes kubectl.

```
kubectl get csv -n operator-namespace
```

Il comando `kubectl` dovrebbe restituire all'operatore di dimensionamento automatico verticale appena implementato uno stato Phase (Fase) impostato su Succeeded (Riuscito). Se hai problemi con l'installazione o la configurazione, consulta Risoluzione dei problemi relativi al dimensionamento automatico verticale di Amazon EMR su EKS.

Nozioni di base sul dimensionamento automatico verticale per Amazon EMR su EKS

Usa la scalabilità automatica verticale per Amazon EMR su EKS quando desideri ottimizzare automaticamente le risorse di memoria e CPU per adattarle al carico di lavoro dell'applicazione Amazon EMR Spark. Per ulteriori informazioni, consulta Usare la scalabilità automatica verticale con i job Amazon EMR Spark.

Invio di un processo Spark con dimensionamento automatico verticale

Quando invii un lavoro tramite l'StartJobRunAPI, aggiungi le seguenti due configurazioni al driver per il tuo job Spark per attivare la scalabilità automatica verticale:

```
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing":"true",
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature":"YOUR_JOB_SIGNATURE"
```

Nel codice precedente, la prima riga abilita la funzionalità di dimensionamento automatico verticale. La riga successiva è una configurazione di firma obbligatoria che consente di scegliere una firma per il processo.

Per ulteriori informazioni su queste configurazioni e sui valori dei parametri accettabili, consulta Configurazione del dimensionamento automatico verticale per Amazon EMR su EKS. Per impostazione predefinita, il processo viene inviato nella modalità Disattivato di solo monitoraggio del dimensionamento automatico verticale. Questo stato di monitoraggio consente di calcolare e visualizzare consigli sulle risorse senza eseguire il dimensionamento automatico. Per ulteriori informazioni, consulta Modalità di dimensionamento automatico verticale.

L'esempio seguente mostra come completare un comando `start-job-run` di esempio con dimensionamento automatico verticale:

```
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.annotation.emr-containers.amazonaws.com/
dynamic.sizing": "true",
          "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature": "test-signature"
        }
    }]
  }'
```

Verifica della funzionalità di dimensionamento automatico verticale

Per verificare che il dimensionamento automatico verticale funzioni correttamente per il processo inviato, utilizza kubectl per ottenere la risorsa personalizzata `verticalpodautoscaler` e visualizzare i consigli di dimensionamento. Ad esempio, il seguente comando invia una query per richiedere consigli sul processo di esempio dalla sezione Invio di un processo Spark con dimensionamento automatico verticale:

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

L'output della query deve assomigliare al seguente:

```
NAME                                                              MODE   CPU         MEM
 PROVIDED   AGE
ds-jceyefkxnhrvdzw6djum3naf2abm6o63a6dvjkkedqtkhlrf25eq-vpa   Off    3304504865  True
         87m
```

Se l'output non è simile o contiene un codice di errore, consulta la procedura indicata in Risoluzione dei problemi relativi al dimensionamento automatico verticale di Amazon EMR su EKS per risolvere il problema.

Configurazione del dimensionamento automatico verticale per Amazon EMR su EKS

Puoi configurare la scalabilità automatica verticale quando invii lavori Amazon EMR Spark tramite l'API. StartJobRun Imposta i parametri di configurazione relativi al dimensionamento automatico nel pod del driver Spark, come mostrato nell'esempio in Invio di un processo Spark con dimensionamento automatico verticale.

L'operatore di dimensionamento automatico verticale di Amazon EMR su EKS ascolta i pod del driver con dimensionamento automatico, quindi configura l'integrazione con il Vertical Pod Autoscaler (VPA) di Kubernetes con le impostazioni sul pod del driver. Ciò facilita il tracciamento delle risorse e il dimensionamento automatico dei pod dell'executor Spark.

Le seguenti sezioni descrivono i parametri che puoi utilizzare quando configuri il dimensionamento automatico verticale per il cluster Amazon EKS.

> **Note**
>
> Configura il parametro di attivazione/disattivazione della funzionalità come etichetta e configura i parametri rimanenti come annotazioni nel pod del driver Spark.
> I parametri del dimensionamento automatico appartengono al dominio `emr-containers.amazonaws.com/` e hanno il prefisso `dynamic.sizing`.

Parametri obbligatori

Devi includere i due parametri seguenti nel driver di processi Spark quando invii il processo:

Chiave	Descrizione	Valori accettati	Valore predefinito	Tipo	Parametro Spark[1]
`dynamic.sizing`	Attivazione/disattivazione della funzionalità	`true`, `false`	non impostato	etichetta	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing`
`dynamic.sizing.signature`	Firma del processo	stringa	non impostato	annotazione	`spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.signature`

[1] Utilizza questo parametro come `SparkSubmitParameter` o `ConfigurationOverride` nell'API `StartJobRun`.

- **`dynamic.sizing`**: puoi attivare e disattivare il dimensionamento automatico verticale con l'etichetta `dynamic.sizing`. Per attivare il dimensionamento automatico verticale, imposta `dynamic.sizing` su `true` nel pod del driver Spark. Se ometti questa etichetta o la imposti su un valore diverso da `true`, il dimensionamento automatico verticale è disattivato.
- **`dynamic.sizing.signature`**: imposta la firma del processo con l'annotazione `dynamic.sizing.signature` nel pod del driver. Il dimensionamento automatico verticale aggrega i dati sull'utilizzo delle risorse in diverse esecuzioni di processi Spark di Amazon EMR per ricavare consigli sulle risorse. Sei tu a fornire l'identificatore univoco per associare i processi.

> **Note**
>
> Se il processo si ripete a intervalli fissi, ad esempio giornalieri o settimanali, la firma deve rimanere la stessa per ogni nuova istanza del processo. Ciò garantisce che il dimensionamento automatico verticale possa calcolare e aggregare i consigli su diverse esecuzioni del processo.

[1] Utilizza questo parametro come `SparkSubmitParameter` o `ConfigurationOverride` nell'API `StartJobRun`.

Parametri facoltativi

Il dimensionamento automatico verticale supporta anche i seguenti parametri opzionali. Impostali come annotazioni nel pod del driver.

Chiave	Descrizione	Valori accettati	Valore predefinito	Tipo	Parametro Spark[1]
dynamic.sizing.mode	Modalità di dimensionamento automatico verticale	`Off`, `Initial`, `Auto`	`Off`	annotazione	`spark.kubernetes.driver.annotation.emr-containers.amaz`

Chiave	Descrizione	Valori accettati	Valore predefinito	Tipo	Parametro Spark[1]
					onaws.com/dynamic.sizing.mode
dynamic.sizing.scale.memory	Abilita il dimensionamento della memoria	*true*, *false*	true	annotazione	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory
dynamic.sizing.scale.cpu	Attiva o disattiva il dimensionamento della CPU	*true*, *false*	false	annotazione	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu

Chiave	Descrizione	Valori accettati	Valore predefinito	Tipo	Parametro Spark[1]
dynamic.sizing.scale.memory.min	Limite minimo per il dimensionamento della memoria	stringa, quantità di risorse K8s es.: 1G	non impostato	annotazione	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.min
dynamic.sizing.scale.memory.max	Limite massimo per il dimensionamento della memoria	stringa, quantità di risorse K8s es.: 4G	non impostato	annotazione	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.max

Chiave	Descrizione	Valori accettati	Valore predefinito	Tipo	Parametro Spark[1]
dynamic.sizing.scale.cpu.min	Limite minimo per il dimensionamento della CPU	stringa, quantità di risorse K8s es.: 1	non impostato	annotazione	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.min
dynamic.sizing.scale.cpu.max	Limite massimo per il dimensionamento della CPU	stringa, quantità di risorse K8s es.: 2	non impostato	annotazione	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.max

Modalità di dimensionamento automatico verticale

Il parametro `mode` è mappato alle diverse modalità di dimensionamento automatico supportate dal VPA. Utilizza l'annotazione `dynamic.sizing.mode` nel pod del driver per impostare la modalità. Per questo parametro sono supportati i valori seguenti:

- Disattivato: una modalità di esecuzione di prova in cui è possibile monitorare i consigli, ma il dimensionamento automatico non viene eseguito. È la modalità predefinita per il dimensionamento automatico verticale. In questa modalità, la risorsa dell'autoscaler di pod verticale associata calcola i consigli e puoi monitorarli attraverso strumenti come kubectl, Prometheus e Grafana.

- Iniziale: in questa modalità, VPA dimensiona in automatico le risorse all'avvio del processo se sono disponibili consigli basati sulle esecuzioni storiche del processo, ad esempio nel caso di un processo ricorrente.
- Automatico: in questa modalità, VPA espelle i pod dell'executor Spark e li dimensiona in automatico con le impostazioni delle risorse consigliate quando il pod del driver Spark li riavvia. A volte, il VPA espelle i pod dell'executor Spark in esecuzione, quindi potrebbe causare una latenza aggiuntiva quando ritenta l'executor interrotto.

Dimensionamento delle risorse

Quando imposti il dimensionamento automatico verticale, puoi scegliere se dimensionare le risorse di CPU e memoria. Imposta le annotazioni `dynamic.sizing.scale.cpu` e `dynamic.sizing.scale.memory` su `true` o `false`. Per impostazione predefinita, il dimensionamento della CPU è impostato su `false` e il dimensionamento della memoria è impostato su `true`.

Valori minimi e massimi delle risorse (limiti)

In alternativa, puoi anche impostare limiti per le risorse della CPU e della memoria. Scegli un valore minimo e massimo per queste risorse con le annotazioni `dynamic.sizing.[memory/cpu].[min/max]` quando abiliti il dimensionamento automatico. Per impostazione predefinita, le risorse non hanno limitazioni. Imposta le annotazioni come valori di stringa che rappresentano una quantità di risorse Kubernetes. Ad esempio, imposta `dynamic.sizing.memory.max` su `4G` per rappresentare 4 GB.

Monitoraggio del dimensionamento automatico verticale per Amazon EMR su EKS

Puoi utilizzare lo strumento da riga di comando kubectl Kubernetes per elencare i consigli attivi e verticali relativi alla scalabilità automatica sul tuo cluster. Puoi anche visualizzare le firme dei processi tracciate ed eliminare tutte le risorse non necessarie associate alle firme.

Elencazione dei consigli di dimensionamento automatico verticale per il cluster

Utilizza kubectl per ottenere la risorsa `verticalpodautoscaler` e visualizzare lo stato attuale e i consigli. La seguente query di esempio restituisce tutte le risorse attive sul cluster Amazon EKS.

```
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
```

L'output della query assomiglia al seguente:

```
NAME                  SIGNATURE              MODE      MEM
ds-example-id-1-vpa   job-signature-1        Off       none
ds-example-id-2-vpa   job-signature-2        Initial   12936384283
```

Invio di query ed eliminazione dei consigli di dimensionamento automatico verticale per il cluster

Quando elimini una risorsa job-run con dimensionamento automatico verticale Amazon EMR, viene eliminato in automatico l'oggetto VPA associato che monitora e archivia i consigli.

L'esempio seguente utilizza kubectl per eliminare i consigli per un processo identificato da una firma:

```
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
```

Se non conosci la firma specifica del processo o desideri eliminare tutte le risorse nel cluster, puoi utilizzare `--all` o `--all-namespaces` nel comando anziché l'ID univoco del processo, come mostrato nell'esempio seguente:

```
kubectl delete jobruns --all --all-namespaces
jobrun.dynamicsizing.emr.services.k8s.aws "ds-example-id" deleted
```

Disinstallazione dell'operatore di dimensionamento automatico verticale di Amazon EMR su EKS

Se desideri rimuovere l'operatore di dimensionamento automatico verticale dal cluster Amazon EKS, utilizza il comando `cleanup` con la CLI di Operator SDK come mostrato nell'esempio seguente. Ciò elimina anche le dipendenze a monte installate con l'operatore, come Vertical Pod Autoscaler.

```
operator-sdk cleanup emr-dynamic-sizing
```

Se nel cluster sono presenti processi in esecuzione quando elimini l'operatore, tali processi continuano a essere eseguiti senza il dimensionamento automatico verticale. Se invii processi sul cluster dopo aver eliminato l'operatore, Amazon EMR su EKS ignorerà tutti i parametri relativi al dimensionamento automatico verticale che potresti aver definito durante la configurazione.

Esecuzione di carichi di lavoro interattivi in Amazon EMR su EKS

Un endpoint interattivo è un gateway che connette Amazon EMR Studio ad Amazon EMR su EKS per permetterti di eseguire carichi di lavoro interattivi. Puoi utilizzare gli endpoint interattivi con EMR Studio per eseguire analisi interattive con set di dati in datastore come Amazon S3 e Amazon DynamoDB.

Casi d'uso

- Crea uno script ETL con l'esperienza IDE di EMR Studio. L'IDE importa i dati on-premise e li archivia in Amazon S3 dopo le trasformazioni per la successiva analisi.
- Utilizza i notebook per esplorare i set di dati e addestra un modello di machine learning per rilevare anomalie nei set di dati.
- Crea script che generano report giornalieri per applicazioni di analisi come i pannelli di controllo aziendali.

Argomenti

Panoramica degli endpoint interattivi

Un endpoint interattivo offre a client interattivi come Amazon EMR Studio la possibilità di connettersi ad Amazon EMR su cluster EKS per eseguire carichi di lavoro interattivi. L'endpoint interattivo è supportato da un gateway Jupyter Enterprise che fornisce la capacità di gestione remota del ciclo di vita del kernel di cui i client interattivi hanno bisogno. I kernel sono processi specifici per linguaggio

che interagiscono con il client Amazon EMR Studio basato su Jupyter per eseguire carichi di lavoro interattivi.

Gli endpoint gestiti supportano i seguenti kernel:

- Python 3
- PySpark su Kubernetes
- Apache Spark con Scala

> **Note**
>
> I prezzi di Amazon EMR su EKS si applicano agli endpoint e ai kernel interattivi. Per ulteriori informazioni, consulta la pagina dei prezzi di Amazon EMR su EKS.

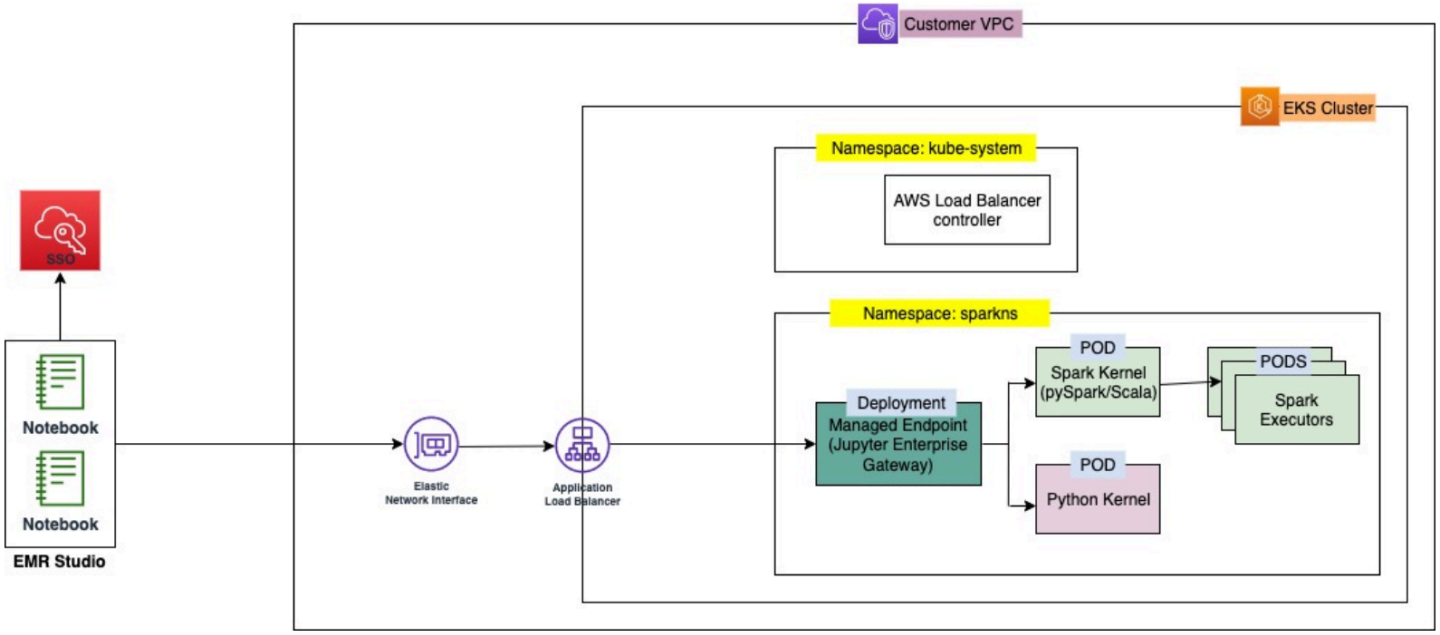
Le seguenti entità sono necessarie affinché EMR Studio si connetta con Amazon EMR su EKS.

- Cluster virtuale Amazon EMR su EKS: un cluster virtuale è uno spazio dei nomi Kubernetes con cui registri Amazon EMR. Amazon EMR utilizza cluster virtuali per l'esecuzione di processi e l'hosting di endpoint. Puoi supportare più cluster virtuali con lo stesso cluster fisico. Tuttavia, ogni cluster virtuale esegue la mappatura a uno spazio dei nomi in un cluster Amazon EKS. I cluster virtuali non creano risorse attive che incrementano i costi in fattura o che richiedono la gestione del ciclo di vita all'esterno del servizio.
- Endpoint interattivo Amazon EMR su EKS: un endpoint interattivo è un endpoint HTTPS a cui gli utenti di EMR Studio possono connettere un Workspace. Gli endpoint HTTPS sono accessibili solo dal tuo EMR Studio e vengono creati in una sottorete privata dell'Amazon Virtual Private Cloud (Amazon VPC) del tuo cluster Amazon EKS.

  I kernel Python e Spark Scala utilizzano le autorizzazioni definite nel tuo ruolo di esecuzione del lavoro Amazon EMR on EKS per richiamarne altri. PySpark Servizi AWS Tutti i kernel e gli utenti che si connettono all'endpoint interattivo utilizzano il ruolo che hai specificato durante la creazione dell'endpoint. Ti consigliamo di creare endpoint separati per utenti diversi e che gli utenti abbiano ruoli (IAM) diversi. AWS Identity and Access Management
- AWS Controller Application Load Balancer: il controller AWS Application Load Balancer gestisce Elastic Load Balancing per un cluster Amazon EKS Kubernetes. Il controller esegue il provisioning di un Application Load Balancer (ALB) quando crei una risorsa di ingresso Kubernetes. Un ALB espone un servizio Kubernetes, ad esempio un endpoint interattivo, all'esterno del cluster

Amazon EKS ma all'interno dello stesso Amazon VPC. Quando crei un endpoint interattivo, viene implementata anche una risorsa di ingresso che espone l'endpoint interattivo mediante l'ALB affinché i client interattivi vi si possano connettere. È sufficiente installare un controller AWS Application Load Balancer per ogni cluster Amazon EKS.

Il diagramma seguente illustra l'architettura degli endpoint interattivi in Amazon EMR su EKS. Un cluster Amazon EKS comprende il calcolo per eseguire i carichi di lavoro analitici e l'endpoint interattivo. Il controller di Application Load Balancer viene eseguito nello spazio dei nomi `kube-system`; i carichi di lavoro e gli endpoint interattivi vengono eseguiti nello spazio dei nomi specificato al momento della creazione del cluster virtuale. Quando crei un endpoint interattivo, il piano di controllo (control-plane) di Amazon EMR su EKS crea l'implementazione dell'endpoint interattivo nel cluster Amazon EKS. Inoltre, un'istanza dell'ingresso del load balancer dell'applicazione viene creata dal controller di AWS bilanciamento del carico. L'Application Load Balancer fornisce l'interfaccia esterna per consentire a client come EMR Studio di connettersi al cluster Amazon EMR per eseguire carichi di lavoro interattivi.



Prerequisiti per la creazione di un endpoint interattivo in Amazon EMR su EKS

Questa sezione descrive i prerequisiti per la configurazione di un endpoint interattivo che EMR Studio potrà utilizzare per connettersi a un cluster Amazon EMR su EKS ed eseguire carichi di lavoro interattivi.

AWS CLI

Segui la procedura descritta in Installare o aggiornare alla versione più recente di () AWS CLI per installare la versione più recente di (). AWS Command Line Interface AWS CLI

Installazione di eksctl

Segui i passaggi in Installa kubectl per installare l'ultima versione di eksctl. Se stai utilizzando la versione 1.22 o successiva di Kubernetes per il cluster Amazon EKS, utilizza una versione di eksctl successiva alla 0.117.0.

Cluster Amazon EKS

Crea un cluster Amazon EKS. Registra il cluster come cluster virtuale con Amazon EMR su EKS. Di seguito sono riportati i requisiti e le considerazioni per tale cluster:

- Il cluster deve trovarsi nello stesso Amazon Virtual Private Cloud (VPC) di EMR Studio.
- Il cluster deve disporre di almeno una sottorete privata per attivare gli endpoint interattivi, collegare repository basati su Git e avviare l'Application Load Balancer in modalità privata.
- Deve esserci almeno una sottorete privata in comune tra EMR Studio e il cluster Amazon EKS che utilizzi per registrare il cluster virtuale. Ciò garantisce che l'endpoint interattivo appaia come opzione nei Workspace Studio e attiva la connettività da Studio all'Application Load Balancer.

  Esistono due metodi tra cui scegliere per connettere Studio e il cluster Amazon EKS:
  - Crea un cluster Amazon EKS e associalo alle sottoreti che appartengono al tuo EMR Studio.
  - In alternativa, crea un EMR Studio e specifica le sottoreti private per il cluster Amazon EKS.
- ARM ottimizzato per Amazon EKS Amazon Linux AMIs non è supportato per Amazon EMR sugli endpoint interattivi EKS.
- Gli endpoint interattivi funzionano con i cluster Amazon EKS che utilizzano versioni di Kubernetes fino alla 1.30.
- Sono supportati solo i gruppi di nodi gestiti di Amazon EKS e i nodi con provisioning Karpenter.

Concessione dell'accesso ai cluster per Amazon EMR su EKS

Utilizza le fasi descritte in Concessione dell'accesso ai cluster per Amazon EMR su EKS per concedere ad Amazon EMR su EKS l'acceso a uno spazio dei nomi specifico nel cluster.

Attivazione di IRSA sul cluster Amazon EKS

Per attivare i ruoli IAM per gli account di servizio (IRSA) sul cluster Amazon EKS, segui la procedura descritta in Abilitazione dei ruoli IAM per gli account di servizio (IRSA).

Creazione di un ruolo di esecuzione di processo IAM

Per eseguire carichi di lavoro sugli endpoint interattivi di Amazon EMR su EKS, è necessario creare un ruolo IAM. Nella presente documentazione, tale ruolo IAM viene definito ruolo di esecuzione di processo. Questo ruolo IAM viene assegnato sia al container dell'endpoint interattivo sia ai container di esecuzione effettivi creati quando invii processi con EMR Studio. Sarà necessario il nome della risorsa Amazon (ARN) del ruolo di esecuzione di processo per Amazon EMR su EKS. A tal fine sono necessarie due fasi:

- Crea un ruolo IAM per l'esecuzione di processo.
- Aggiorna la policy di affidabilità del ruolo di esecuzione di processo.

Concessione dell'accesso ad Amazon EMR su EKS agli utenti

L'entità IAM (utente o ruolo) che effettua la richiesta di creazione di un endpoint interattivo deve inoltre disporre dei seguenti Amazon EC2 e delle seguenti `emr-containers` autorizzazioni. Segui la procedura descritta in Concessione dell'accesso ad Amazon EMR su EKS agli utenti per concedere queste autorizzazioni che consentono ad Amazon EMR su EKS di creare, gestire ed eliminare i gruppi di sicurezza che limitano il traffico in entrata al sistema di bilanciamento del carico dell'endpoint interattivo.

Le seguenti autorizzazioni `emr-containers` consentono all'utente di eseguire operazioni di endpoint interattivo di base:

```
"ec2:CreateSecurityGroup",
"ec2:DeleteSecurityGroup",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:RevokeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress"

"emr-containers:CreateManagedEndpoint",
"emr-containers:ListManagedEndpoints",
```

```
"emr-containers:DescribeManagedEndpoint",
"emr-containers:DeleteManagedEndpoint"
```

Registrazione del cluster Amazon EKS con Amazon EMR

Configura un cluster virtuale e mappalo allo spazio dei nomi nel cluster Amazon EKS in cui desideri eseguire i processi. Per i cluster AWS Fargate-only, usa lo stesso namespace sia per il cluster virtuale Amazon EMR su EKS che per il profilo Fargate.

Per informazioni sulla configurazione di un cluster virtuale Amazon EMR su EKS, consulta la sezione Registrazione del cluster Amazon EKS con Amazon EMR.

Implementa AWS Load Balancer Controller nel cluster Amazon EKS

È necessario un AWS Application Load Balancer per il cluster Amazon EKS. È sufficiente impostare un controller di Application Load Balancer per ogni cluster Amazon EKS. Per informazioni sulla configurazione del controller AWS Application Load Balancer, consulta Installazione del componente aggiuntivo Load AWS Balancer Controller nella Guida per l'utente di Amazon EKS.

Creazione di un endpoint interattivo per il cluster virtuale

Questo argomento descrive un paio di modi per creare un endpoint interattivo utilizzando l'interfaccia a riga di AWS comando (AWS CLI) e include dettagli sui parametri di configurazione disponibili.

Creazione di un endpoint interattivo con il comando **`create-managed-endpoint`**

Specifica i parametri nel comando `create-managed-endpoint` nel modo seguente. Amazon EMR su EKS supporta la creazione di endpoint interattivi con i rilasci 6.7.0 e successivi di Amazon EMR.

```
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"
        }
    }
}'
```

Per ulteriori informazioni, consulta Parametri per la creazione di un endpoint interattivo.

Creazione di un endpoint interattivo con parametri specificati in un file JSON

1. Crea un file `create-managed-endpoint-request.json` e specifica i parametri richiesti per l'endpoint, come illustrato nel file JSON seguente:

```
{
    "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. Utilizza il comando `create-managed-endpoint` con un percorso verso il file `create-managed-endpoint-request.json` archiviato localmente o in Amazon S3.

```
aws emr-containers create-managed-endpoint \
--cli-input-json  file://./create-managed-endpoint-request.json --region AWS-Region
```

Output della creazione di un endpoint interattivo

Dovresti visualizzare il seguente output nel terminale. L'output include il nome e l'identificatore del nuovo endpoint interattivo:

```
{
    "id": "1234567890abcdef0",
    "name": "example-endpoint-name",
    "arn": "arn:aws:emr-containers:us-west-2:111122223333:/
virtualclusters/444455556666/endpoints/444455556666",
    "virtualClusterId": "111122223333xxxxxxxx"
}
```

L'esecuzione di `aws emr-containers create-managed-endpoint` crea un certificato autofirmato che consente la comunicazione HTTPS tra EMR Studio e il server dell'endpoint interattivo.

Se esegui `create-managed-endpoint` e non hai completato i prerequisiti, Amazon EMR restituisce un messaggio di errore con le azioni da intraprendere per continuare.

Parametri per la creazione di un endpoint interattivo

Argomenti

Parametri obbligatori per gli endpoint interattivi

Quando crei un endpoint interattivo, devi specificare i parametri seguenti:

`--type`

Utilizza `JUPYTER_ENTERPRISE_GATEWAY`. Questo è l'unico tipo supportato.

`--virtual-cluster-id`

L'identificatore del cluster virtuale registrato con Amazon EMR su EKS.

`--name`

Un nome descrittivo per l'endpoint interattivo che aiuta gli utenti di EMR Studio a selezionarlo da un elenco a discesa.

`--execution-role-arn`

Il nome della risorsa Amazon (ARN) del ruolo IAM di esecuzione di processo IAM per Amazon EMR su EKS che è stato creato nell'ambito dei prerequisiti.

`--release-label`

L'etichetta di rilascio del rilascio di Amazon EMR da utilizzare per l'endpoint. Ad esempio, `emr-6.9.0-latest`. Amazon EMR su EKS supporta gli endpoint interattivi con i rilasci 6.7.0 e successivi di Amazon EMR.

Parametri opzionali per gli endpoint interattivi

Facoltativamente, quando crei un endpoint interattivo puoi anche specificare i parametri seguenti:

`--configuration-overrides`

Per ignorare le configurazioni predefinite per le applicazioni, fornisci un oggetto di configurazione. Puoi utilizzare una sintassi abbreviata per fornire la configurazione oppure fare riferimento all'oggetto di configurazione in un file JSON.

Gli oggetti di configurazione sono composti da una classificazione, proprietà e configurazioni nidificate opzionali. Le proprietà sono costituite dalle impostazioni che desideri ignorare in un dato file. Puoi specificare diverse classificazioni per più applicazioni in un singolo oggetto JSON. Le classificazioni di configurazione disponibili variano a seconda della versione di Amazon EMR su EKS. Per un elenco delle classificazioni di configurazione disponibili per ciascuna versione di Amazon EMR, consulta Rilasci di Amazon EMR su EKS. Oltre alle classificazioni di configurazione elencate per ciascuna versione, gli endpoint interattivi includono la classificazione aggiuntiva `jeg-config`. Per ulteriori informazioni, consulta Opzioni di configurazione di Jupyter Enterprise Gateway (JEG).

Configurazione delle impostazioni per gli endpoint interattivi

Questa sezione contiene una serie di argomenti che trattano varie configurazioni per gli endpoint interattivi e le impostazioni dei pod. Questi ti danno la possibilità di monitorare e risolvere i guasti, inviare informazioni di log ad Amazon S3 o a Amazon CloudWatch Logs o creare endpoint interattivi in cui specificare modelli di pod personalizzati.

Argomenti

- Monitoraggio dei processi Spark
- Specificazione di modelli di pod personalizzati con gli endpoint interattivi
- Implementazione di un pod JEG in un gruppo di nodi
- Opzioni di configurazione di Jupyter Enterprise Gateway (JEG)
- Modifica dei parametri della sessione PySpark
- Immagine di kernel personalizzata con endpoint interattivo

Monitoraggio dei processi Spark

Per monitorare e risolvere i problemi, configura gli endpoint interattivi in modo che i lavori avviati con l'endpoint possano inviare informazioni di log ad Amazon S3, Amazon Logs o entrambi. CloudWatch Le sezioni seguenti descrivono come inviare i log delle applicazioni Spark ad Amazon S3 per i processi Spark che avvii con gli endpoint interattivi di Amazon EMR su EKS.

Configurazione della policy IAM per i log di Amazon S3

Prima che i kernel possano inviare i dati dei log ad Amazon S3, nella policy delle autorizzazioni per il ruolo di esecuzione del processo devono essere incluse le seguenti autorizzazioni. Sostituisci *`amzn-s3-demo-destination-bucket`* con il nome del bucket di accesso.

JSON

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

> **Note**
>
> Amazon EMR su EKS può anche creare un bucket S3. Se un bucket Amazon S3 non è disponibile, includi l'autorizzazione `s3:CreateBucket` nella policy IAM.

Dopo avere assegnato al ruolo di esecuzione le autorizzazioni appropriate per l'invio dei log al bucket S3, i dati dei log vengono inviati ai seguenti percorsi di Amazon S3. Ciò accade quando `s3MonitoringConfiguration` viene passato nella sezione `monitoringConfiguration` di una richiesta `create-managed-endpoint`.

- Log dei driver: `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/spark-application-id-driver/(stderr.gz/stdout.gz)`
- Log degli executor: `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/executor-pod-name-exec-<Number>/(stderr.gz/stdout.gz)`

> **Note**
>
> Amazon EMR su EKS non carica i log degli endpoint sul bucket S3.

Specificazione di modelli di pod personalizzati con gli endpoint interattivi

Puoi creare endpoint interattivi in cui è possibile specificare modelli di pod personalizzati per driver ed executor. I modelli di pod sono specifiche che determinano la modalità di esecuzione di ciascun pod. È possibile utilizzare i file dei modelli di pod per definire le configurazioni dei pod driver o executor non supportate dalle configurazioni Spark. I modelli di pod sono attualmente supportati nei rilasci di Amazon EMR 6.3.0 e successivi.

Per ulteriori informazioni sui modelli di pod, consulta la sezione Uso dei modelli di pod nella Guida per gli sviluppatori di Amazon EMR su EKS.

L'esempio seguente mostra come creare un endpoint interattivo con i modelli di pod:

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

Implementazione di un pod JEG in un gruppo di nodi

Il posizionamento dei pod JEG (Jupyter Enterprise Gateway) è una funzionalità che consente di implementare un endpoint interattivo su un gruppo di nodi specifico. Con questa funzionalità, puoi configurare impostazioni come `instance type` per l'endpoint interattivo.

Associazione di un pod JEG a un gruppo di nodi gestito

La seguente proprietà di configurazione consente di specificare il nome di un gruppo di nodi gestito sul cluster Amazon EKS in cui il pod JEG verrà implementato.

```
//payload
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName
                }
            }
        ]
    }'
```

Un gruppo di nodi deve presentare l'etichetta Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* collegata a tutti i nodi che fanno parte del gruppo. Per elencare tutti i nodi di un gruppo di nodi che dispongono di tale tag, utilizza il comando seguente:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Se l'output del comando precedente non restituisce nodi che fanno parte del gruppo di nodi gestito, non ci sono nodi nel gruppo di nodi a cui è collegata l'etichetta Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`*. In questo caso, segui le fasi seguenti per collegare l'etichetta ai nodi del tuo gruppo di nodi.

1. Utilizza il comando seguente per aggiungere l'etichetta Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* a tutti i nodi in un gruppo di nodi gestito *`NodeGroupName`*:

```
kubectl label nodes --selector eks:nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

2. Verifica che i nodi siano stati etichettati correttamente utilizzando il comando seguente:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Un gruppo di nodi gestito deve essere associato al gruppo di sicurezza di un cluster Amazon EKS, come di solito accade se hai creato il cluster e il gruppo di nodi gestito utilizzando `eksctl`. Puoi verificarlo nella console utilizzando i seguenti passaggi. AWS

1. Accedi al tuo cluster nella console Amazon EKS.
2. Apri la scheda di rete del cluster e annota il gruppo di sicurezza del cluster.
3. Apri la scheda di calcolo del cluster e fai clic sul nome del gruppo di nodi gestito.
4. Nella scheda Dettagli del gruppo di nodi gestito, verifica che il gruppo di sicurezza del cluster che hai annotato in precedenza sia elencato in Gruppi di sicurezza.

Se il gruppo di nodi gestito non è collegato al gruppo di sicurezza del cluster Amazon EKS, devi allegare il tag `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* al gruppo di sicurezza del gruppo di nodi. Utilizza le fasi seguenti per allegare il suddetto tag.

1. Vai alla EC2 console Amazon e fai clic sui gruppi di sicurezza nel riquadro di navigazione a sinistra.
2. Seleziona il gruppo di sicurezza del gruppo di nodi gestito facendo clic sulla casella di spunta.
3. Nella scheda Tag, aggiungi il tag `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* utilizzando il pulsante Gestisci tag.

Associazione di un pod JEG a un gruppo di nodi autogestito

La seguente proprietà di configurazione consente di specificare il nome di un gruppo di nodi autogestito o non gestito sul cluster Amazon EKS in cui il pod JEG verrà implementato.

```
//payload
```

```
--configuration-overrides '{
      "applicationConfiguration": [
           {
               "classification": "endpoint-configuration",
               "properties": {
                   "self-managed-nodegroup-name": NodeGroupName
               }
           }
        ]
    }'
```

Il gruppo di nodi deve presentare l'etichetta Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* collegata a tutti i nodi che fanno parte del gruppo. Per elencare tutti i nodi di un gruppo di nodi che dispongono di tale tag, utilizza il comando seguente:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Se l'output del comando precedente non restituisce nodi che fanno parte del gruppo di nodi autogestito, non ci sono nodi nel gruppo di nodi a cui è collegata l'etichetta Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`*. In questo caso, segui le fasi seguenti per collegare l'etichetta ai nodi del tuo gruppo di nodi.

1. Se hai creato il gruppo di nodi autogestito con `eksctl`, utilizza il comando seguente per aggiungere l'etichetta Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* a tutti i nodi del gruppo di nodi autogestito *`NodeGroupName`* in un solo passaggio.

   ```
   kubectl label nodes --selector alpha.eksctl.io/nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

   Se non hai utilizzato `eksctl` per creare il gruppo di nodi autogestito, dovrai sostituire il selettore nel comando precedente con un'etichetta Kubernetes diversa collegata a tutti i nodi del gruppo di nodi.

2. Utilizza il comando seguente per verificare che i nodi siano stati etichettati correttamente:

   ```
   kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

Al gruppo di sicurezza del gruppo di nodi autogestito deve essere allegato il tag `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`*. Utilizza le fasi seguenti per allegare il tag al gruppo di sicurezza dalla Console di gestione AWS.

1. Accedi alla EC2 console Amazon. Seleziona Gruppi di sicurezza nel riquadro di navigazione a sinistra.
2. Seleziona la casella di spunta accanto al gruppo di sicurezza per il tuo gruppo di nodi autogestito.
3. Nella scheda Tag, utilizza il pulsante Gestisci tag per aggiungere il tag `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`*. Sostituisci *`ClusterName`* e *`NodeGroupName`* con i valori appropriati.

Associazione di un pod JEG a un gruppo di nodi gestito con istanze on demand

Puoi anche definire etichette aggiuntive, note come selettori di etichette Kubernetes, per specificare ulteriori vincoli o restrizioni all'esecuzione di un endpoint interattivo su un determinato nodo o gruppo di nodi. L'esempio seguente mostra come utilizzare le EC2 istanze Amazon On-Demand per un pod JEG.

```
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName,
                    "node-labels": "eks.amazonaws.com/capacityType:ON_DEMAND"
                }
            }
        ]
    }'
```

> **Note**
>
> Puoi utilizzare la proprietà `node-labels` solo con una proprietà `managed-nodegroup-name` o `self-managed-nodegroup-name`.

Opzioni di configurazione di Jupyter Enterprise Gateway (JEG)

Amazon EMR su EKS utilizza Jupyter Enterprise Gateway (JEG) per attivare gli endpoint interattivi. Puoi impostare i valori seguenti per le configurazioni di JEG consentite al momento della creazione dell'endpoint.

- **`RemoteMappingKernelManager.cull_idle_timeout`**: il timeout in secondi (numero intero) dopo il quale un kernel è considerato inattivo e pronto l'eliminazione. I valori pari o inferiori a `0` disattivano l'eliminazione. I timeout brevi potrebbero comportare l'eliminazione dei kernel per gli utenti con connessioni di rete scadenti.
- **`RemoteMappingKernelManager.cull_interval`**: l'intervallo in secondi (numero intero) a quale verificare la presenza di kernel inattivi che superano il valore di timeout per l'eliminazione.

Modifica dei parametri della sessione PySpark

A partire da Amazon EMR sulla release 6.9.0 di EKS, in Amazon EMR Studio puoi regolare la configurazione Spark associata a una PySpark sessione eseguendo il `%%configure` comando magico nella cella del notebook EMR.

L'esempio seguente mostra un payload di esempio che è possibile utilizzare per modificare la memoria, i core e altre proprietà per il driver e l'executor Spark. Utilizzando le impostazioni `conf`, è possibile configurare qualsiasi configurazione Spark menzionata nella documentazione di configurazione di Apache Spark.

```
%%configure -f
{
  "driverMemory": "16G",
  "driverCores": 4,
  "executorMemory" : "32G",
  "executorCores": 2,
  "conf": {
     "spark.dynamicAllocation.maxExecutors" : 10,
     "spark.dynamicAllocation.minExecutors": 1
  }
}
```

L'esempio seguente mostra un payload di esempio che puoi utilizzare per aggiungere file, PyFile e dipendenze jar a un runtime Spark.

```
%%configure -f
{
  "files": "s3://amzn-s3-demo-bucket-emr-eks/sample_file.txt",
  "pyFiles": : "path-to-python-files",
  "jars" : "path-to-jars
}
```

Immagine di kernel personalizzata con endpoint interattivo

Per assicurarti di disporre delle dipendenze corrette per l'applicazione quando esegui carichi di lavoro interattivi da Amazon EMR Studio, puoi personalizzare le immagini Docker per gli endpoint interattivi ed eseguire immagini di kernel personalizzate. Per creare un endpoint interattivo e collegarlo a un'immagine Docker personalizzata, completa la procedura seguente.

> **Note**
>
> È possibile sovrascrivere solo le immagini di base. Non è possibile aggiungere nuovi tipi di immagini del kernel.

1. Crea e pubblica un'immagine Docker personalizzata. L'immagine di base contiene il runtime Spark e i kernel del notebook che vengono eseguiti con esso. Per creare l'immagine, puoi seguire i passaggi da 1 a 4 riportati in Come personalizzare le immagini Docker. Nel passaggio 1, l'URI dell'immagine di base nel file Docker deve utilizzare `notebook-spark` al posto di `spark`.

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

   Per ulteriori informazioni su come selezionare Regioni AWS e contenere i tag delle immagini, consulta. Dettagli per la selezione dell'URI di un'immagine di base
2. Crea un endpoint interattivo che si possa utilizzare con l'immagine personalizzata.

   a. Crea un file JSON `custom-image-managed-endpoint.json` con i seguenti contenuti. Questo esempio utilizza la versione 6.9.0 di Amazon EMR.

      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. Crea un endpoint interattivo utilizzando le configurazioni specificate nel file JSON come illustrato nell'esempio seguente. Per ulteriori informazioni, consulta Creazione di un endpoint interattivo con il comando `create-managed-endpoint`.

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

3. Connettiti all'endpoint interattivo attraverso EMR Studio. Per ulteriori informazioni e passaggi da completare, consulta Connecting from Studio nella sezione Amazon EMR on EKS dei documenti di AWS Workshop Studio.

Monitoraggio degli endpoint interattivi

Con Amazon EMR su EKS versione 6.10 e successive, gli endpoint interattivi emettono parametri CloudWatch Amazon per il monitoraggio e la risoluzione dei problemi delle operazioni del ciclo di vita del kernel. I parametri vengono attivati da client interattivi, come EMR Studio o notebook Jupyter in hosting autonomo. A ciascuna delle operazioni supportate dagli endpoint interattivi sono associati parametri. Le operazioni sono modellate come dimensioni per ogni parametro, come mostrato nella tabella seguente. Le metriche emesse dagli endpoint interattivi sono visibili in uno spazio dei nomi personalizzato, nel tuo account. EMRContainers

Metrica	Description	Unità
RequestCount	Numero cumulativo di richieste di un'operazione elaborate dall'endpoint interattivo.	Conteggio
RequestLatency	Il tempo trascorso tra l'arrivo di una richiesta all'endpo int interattivo e l'invio di una risposta da parte dell'endpoint interattivo.	Millisecondi
4 XXError	Emesso quando una richiesta di operazione genera un errore 4xx durante l'elabora zione.	Conteggio
5XXError	Emesso quando una richiesta di operazione genera un errore 5xx lato server.	Conteggio
KernelLaunchSuccess	Applicabile solo per l' CreateKernel operazione. Indica il numero cumulativo di avvii del kernel riusciti fino alla presente richiesta (inclusa).	Conteggio

Metrica	Description	Unità
KernelLaunchFailure	Applicabile solo per l' CreateKernel operazione. Indica il numero cumulativo di errori di avvio del kernel fino alla presente richiesta (inclusa).	Conteggio

A ogni parametro di endpoint interattivo sono allegate le seguenti dimensioni:

- **`ManagedEndpointId`**: identificatore dell'endpoint interattivo
- **`OperationName`**: l'operazione attivata dal client interattivo

I valori possibili della dimensione **`OperationName`** sono riportati nella tabella seguente:

`operationName`	Descrizione dell'operazione
`CreateKernel`	Per richiedere che l'endpoint interattivo avvii un kernel.
`ListKernels`	Per richiedere che l'endpoint interattivo elenchi i kernel che sono stati avviati in precedenza utilizzando lo stesso token di sessione.
`GetKernel`	Per richiedere che l'endpoint interattivo ottenga i dettagli su un kernel specifico che è stato avviato in precedenza.
`ConnectKernel`	Per richiedere che l'endpoint interattivo stabilisca la connettività tra il client del notebook e il kernel.
`ConfigureKernel`	Per pubblicare `%%configure magic request` su un kernel pyspark.

operationName	Descrizione dell'operazione
`ListKernelSpecs`	Per richiedere che l'endpoint elenchi le specifiche di kerner disponibili.
`GetKernelSpec`	Per richiedere che l'endpoint interattivo ottenga le specifiche di kernel di un kernel specifico che è stato avviato in precedenza.
`GetKernelSpecResource`	Per richiedere che l'endpoint interattivo ottenga risorse specifiche associate alle specifiche di kernel avviate in precedenza.

Esempi

Per accedere al numero totale di kernel avviati per un endpoint interattivo in un determinato giorno:

1. Seleziona lo spazio dei nomi personalizzato: `EMRContainers`
2. Seleziona il tuo `ManagedEndpointId`, `OperationName – CreateKernel`
3. Il parametro `RequestCount` con la statistica `SUM` e il periodo `1 day` fornirà tutte le richieste di avvio di kernel effettuate nelle ultime 24 ore.
4. KernelLaunchSuccess metric with statistic `SUM` and period `1 day` fornirà tutte le richieste di avvio del kernel effettuate con successo nelle ultime 24 ore.

Per accedere al numero totale di errori relativi a kernel per un endpoint interattivo in un determinato giorno:

1. Seleziona lo spazio dei nomi personalizzato: EMRContainers
2. Seleziona il tuo `ManagedEndpointId`, `OperationName – CreateKernel`
3. Il parametro `KernelLaunchFailure` con la statistica `SUM` e il periodo `1 day` fornirà tutte le richieste di avvio di kernel non riuscite effettuate nelle ultime 24 ore. Puoi anche selezionare il parametro `4XXError` e `5XXError` per sapere che tipo di errore di avvio di kernel si è verificato.

Uso di notebook Jupyter in hosting autonomo

Puoi ospitare e gestire Jupyter o JupyterLab notebook su un'istanza Amazon o sul tuo cluster EC2 Amazon EKS come notebook Jupyter con hosting autonomo. Puoi quindi eseguire carichi di lavoro interattivi con i notebook Jupyter in hosting autonomo. Le sezioni seguenti illustrano il processo di configurazione e implementazione di un notebook Jupyter in hosting autonomo su un cluster Amazon EKS.

Creazione di un notebook Jupyter in hosting autonomo su un cluster EKS

- Creare un gruppo di sicurezza
- Creazione di un endpoint interattivo Amazon EMR su EKS
- Recupero dell'URL del server gateway dell'endpoint interattivo
- Recupero di un token di autenticazione per la connessione all'endpoint interattivo
- Esempio: implementa un notebook JupyterLab
- Eliminazione di un notebook Jupyter in hosting autonomo

Creare un gruppo di sicurezza

Prima di poter creare un endpoint interattivo ed eseguire un Jupyter o un notebook con hosting autonomo, è necessario creare un gruppo di sicurezza per controllare il traffico tra il JupyterLab notebook e l'endpoint interattivo. Per utilizzare la EC2 console Amazon o Amazon EC2 SDK per creare il gruppo di sicurezza, consulta i passaggi in Creare un gruppo di sicurezza nella Amazon EC2 User Guide. Dovresti creare il gruppo di sicurezza nel VPC in cui desideri installare il server notebook.

Per seguire l'esempio di questa guida, utilizza lo stesso VPC del cluster Amazon EKS. Se desideri ospitare il tuo notebook in un VPC diverso dal VPC del tuo cluster Amazon EKS, potresti dover creare una connessione peering tra i due. VPCs Per i passaggi per creare una connessione peering tra due VPCs, consulta Creare una connessione peering VPC nella Amazon VPC Getting Started Guide.

È necessario l'ID del gruppo di sicurezza per creare un endpoint interattivo Amazon EMR su EKS nella fase successiva.

Creazione di un endpoint interattivo Amazon EMR su EKS

Dopo aver creato un gruppo di sicurezza per il notebook, utilizza le fasi indicate in Creazione di un endpoint interattivo per il cluster virtuale per creare un endpoint interattivo. È necessario fornire l'ID del gruppo di sicurezza che hai creato per il notebook in Creare un gruppo di sicurezza.

Inserisci l'ID di sicurezza al posto di quello nelle seguenti impostazioni di *your-notebook-security-group-id* configurazione sostitutiva:

```
--configuration-overrides '{
    "applicationConfiguration": [
        {
            "classification": "endpoint-configuration",
            "properties": {
                "notebook-security-group-id": "your-notebook-security-group-id"
            }
        }
    ],
    "monitoringConfiguration": {
    ...'
```

Recupero dell'URL del server gateway dell'endpoint interattivo

Dopo aver creato un endpoint interattivo, recupera l'URL del server gateway con il comando `describe-managed-endpoint` nella AWS CLI. Questo URL è indispensabile per connettere il notebook all'endpoint. L'URL del server gateway è un endpoint privato.

```
aws emr-containers describe-managed-endpoint \
--region region \
--virtual-cluster-id virtualClusterId \
--id endpointId
```

Inizialmente, l'endpoint si trova nello stato CREATING. Dopo alcuni minuti, passa allo stato ACTIVE. Quando è nello stato ACTIVE, l'endpoint è pronto per l'uso.

Prendi nota dell'attributo `serverUrl` che il comando `aws emr-containers describe-managed-endpoint` restituisce dall'endpoint attivo. È necessario questo URL per connettere il notebook all'endpoint quando si distribuisce un Jupyter o un notebook ospitato autonomamente. JupyterLab

Recupero di un token di autenticazione per la connessione all'endpoint interattivo

Per connetterti a un endpoint interattivo da un Jupyter o da un JupyterLab notebook, devi generare un token di sessione con l'API. `GetManagedEndpointSessionCredentials` Il un token funge da prova di autenticazione per connetterti al server dell'endpoint interattivo.

Il comando seguente viene spiegato più nel dettaglio di seguito con un esempio di output.

```
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

L'ARN del tuo endpoint. Puoi trovare l'ARN nel risultato di una chiamata `describe-managed-endpoint`.

virtualClusterArn

L'ARN del cluster virtuale.

executionRoleArn

L'ARN del ruolo di esecuzione.

durationInSeconds

La durata in secondi per la quale il token è valido. La durata predefinita è 15 minuti (`900`) e il massimo è 12 ore (`43200`).

region

La stessa regione dell'endpoint.

L'output dovrebbe essere simile all'esempio seguente. Prendi nota del *`session-token`* valore che utilizzerai quando installerai un Jupyter o un notebook ospitato autonomamente. JupyterLab

```
{
    "id": "credentialsId",
    "credentials": {
        "token": "session-token"
    },
    "expiresAt": "2022-07-05T17:49:38Z"
}
```

Esempio: implementa un notebook JupyterLab

Una volta completati i passaggi precedenti, puoi provare questa procedura di esempio per distribuire un JupyterLab notebook nel cluster Amazon EKS con il tuo endpoint interattivo.

1. Crea uno spazio dei nomi per eseguire il server notebook.
2. Crea un file localmente, `notebook.yaml`, con i contenuti seguenti. I contenuti del file sono descritti di seguito.

```
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"
```

Se stai implementando un notebook Jupyter su un cluster solo per Fargate, etichetta il pod di Jupyter con un'etichetta `role` come mostrato nell'esempio seguente:

```
...
metadata:
  name: jupyter-notebook
  namespace: default
  labels:
    role: example-role-name-label
spec:
            ...
```

namespace

Lo spazio dei nomi Kubernetes in cui viene implementato il notebook.

serverUrl

L'attributo `serverUrl` che il comando `describe-managed-endpoint` ha restituito in Recupero dell'URL del server gateway dell'endpoint interattivo .

session-token

L'attributo `session-token` che il comando `get-managed-endpoint-session-credentials` ha restituito in Recupero di un token di autenticazione per la connessione all'endpoint interattivo.

KERNEL_LAUNCH_TIMEOUT

La quantità di tempo espressa in secondi per cui l'endpoint interattivo attende che il kernel raggiunga lo stato RUNNING. Garantisci un tempo sufficiente per il completamento dell'avvio del kernel impostando il timeout di avvio del kernel su un valore appropriato (massimo 400 secondi).

KERNEL_EXTRA_SPARK_OPTS

Facoltativamente, puoi trasmettere configurazioni Spark aggiuntive per i kernel Spark. Imposta questa variabile di ambiente con i valori come proprietà di configurazione Spark, come mostrato nell'esempio seguente:

```
- 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. Implementa una specifica di pod nel cluster Amazon EKS:

```
kubectl apply -f notebook.yaml -n namespace
```

Questo avvierà un JupyterLab notebook minimale collegato al tuo endpoint interattivo Amazon EMR su EKS. Attendi che il pod raggiunga lo stato RUNNING. Puoi verificarne lo stato con il comando seguente:

```
kubectl get pod jupyter-notebook -n namespace
```

Quando il pod è pronto, il `get pod` comando restituisce un output simile a questo:

```
NAME              READY  STATUS   RESTARTS  AGE
jupyter-notebook  1/1    Running  0         46s
```

4. Collega il gruppo di sicurezza del notebook al nodo in cui il notebook è pianificato.

   a. Per prima cosa, identifica il nodo in cui il pod `jupyter-notebook` è pianificato con il comando `describe pod`.

   ```
   kubectl describe pod jupyter-notebook -n namespace
   ```

   b. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/home#/clusters.

   c. Passa alla scheda Calcolo del cluster Amazon EKS e seleziona il nodo identificato dal comando `describe pod`. Seleziona l'ID dell'istanza per il nodo.

   d. Nel menu Azioni, seleziona Sicurezza > Modifica gruppi di sicurezza per allegare il gruppo di sicurezza creato in Creare un gruppo di sicurezza.

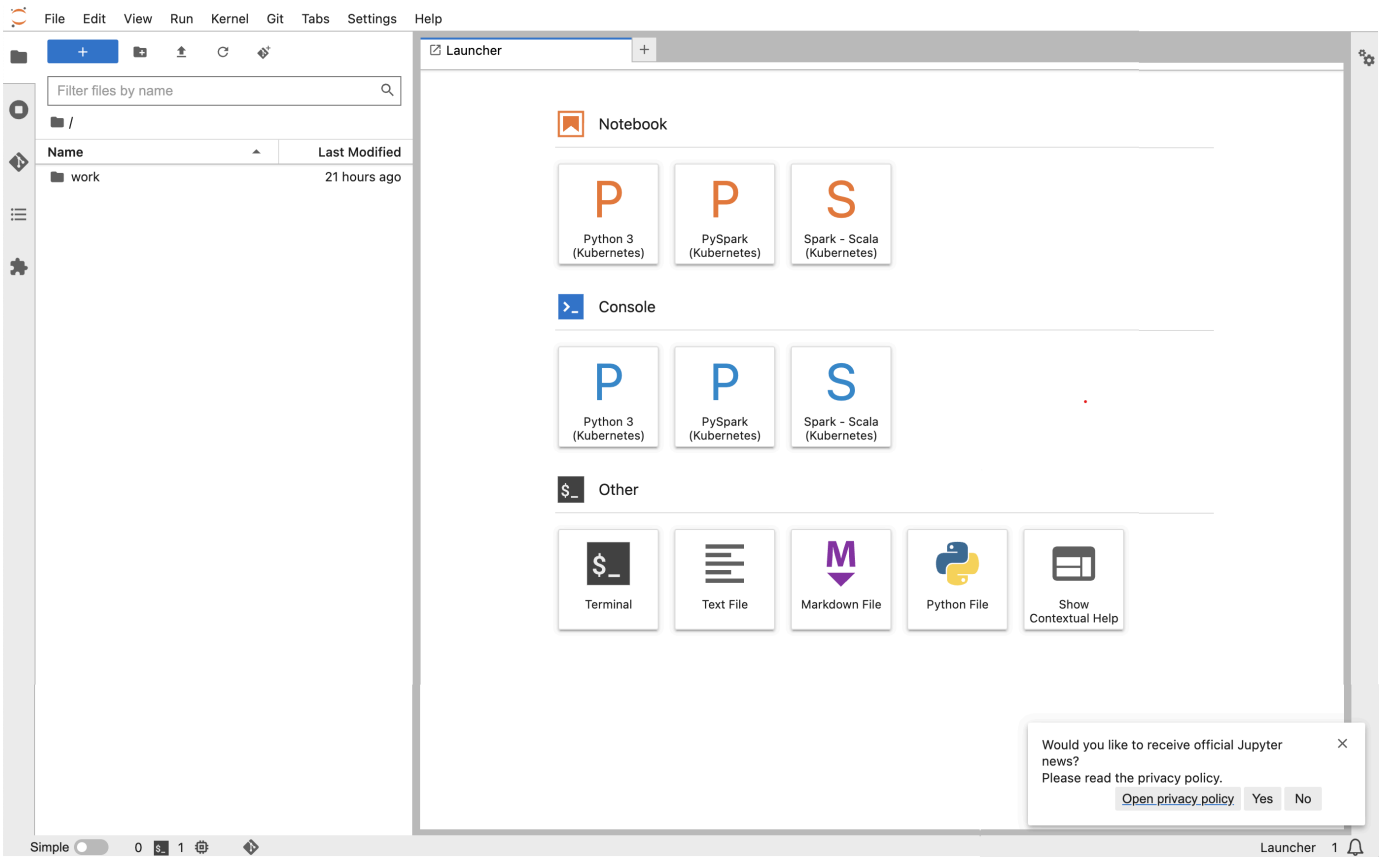
e. Se stai implementando Jupyter Notebook Pod su AWS Fargate, crea un pod da applicare SecurityGroupPolicyal notebook Jupyter con l'etichetta del ruolo:

```
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. Ora esegui il port-forward in modo da poter accedere localmente all'interfaccia: JupyterLab

```
kubectl port-forward jupyter-notebook 8888:8888 -n namespace
```

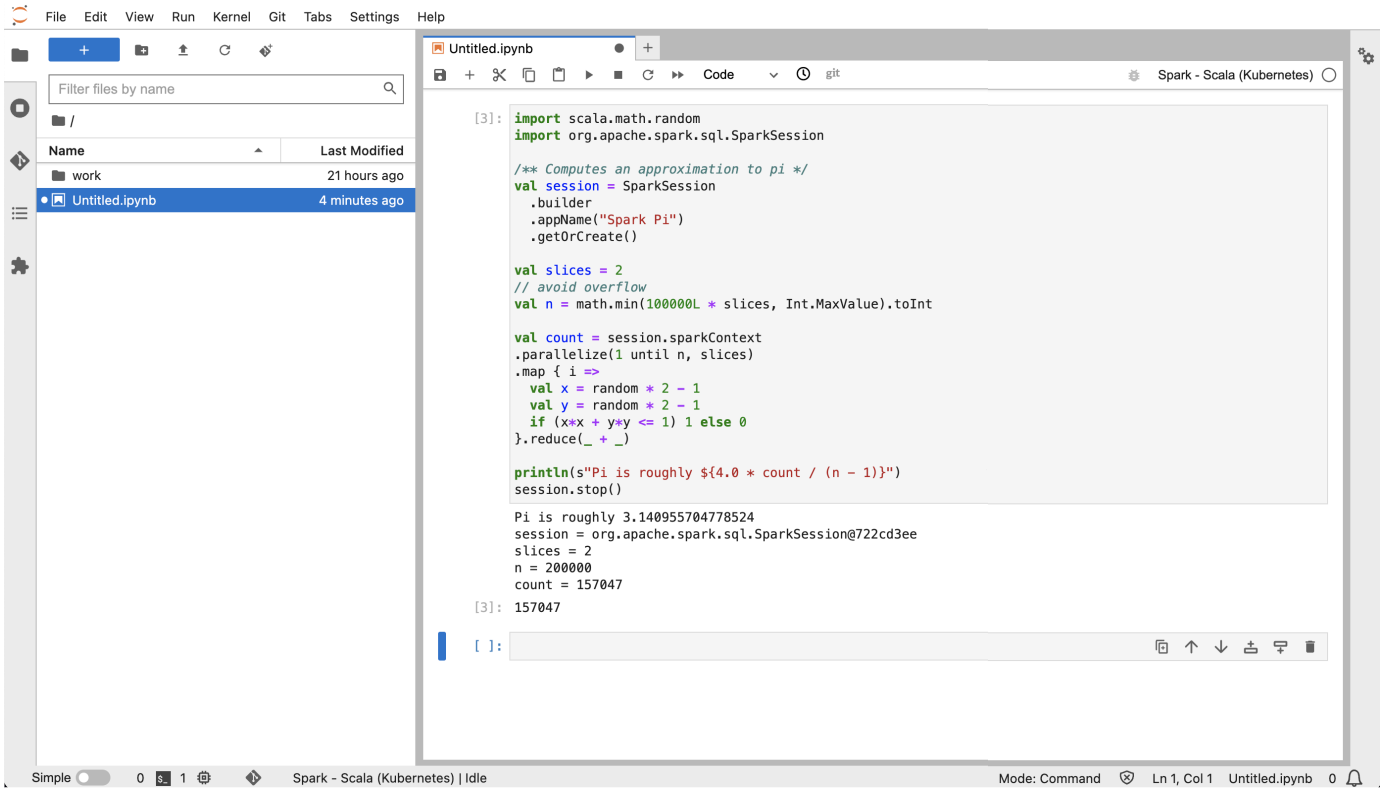
Una volta che è in esecuzione, vai al tuo browser locale e visita `localhost:8888` per vedere l' JupyterLab interfaccia:



6. Da JupyterLab, crea un nuovo taccuino Scala. Ecco un frammento di codice di esempio che puoi eseguire per approssimare il valore di Pi:

```
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()
```



Eliminazione di un notebook Jupyter in hosting autonomo

Quando è il momento di eliminare il notebook in hosting autonomo, puoi eliminare anche l'endpoint interattivo e il gruppo di sicurezza. Esegui le azioni nell'ordine seguente:

1. Utilizza il comando seguente per eliminare il pod `jupyter-notebook`:

   ```
   kubectl delete pod jupyter-notebook -n namespace
   ```

2. Quindi, elimina l'endpoint interattivo con il comando `delete-managed-endpoint`. La procedura per l'eliminazione di un endpoint interattivo è riportata in Eliminazione di un endpoint interattivo. Inizialmente, l'endpoint si troverà nello stato TERMINATING. Una volta ripulite tutte le risorse, passa allo stato TERMINATED.

3. Se non intendi utilizzare il gruppo di sicurezza dei notebook che hai creato in Creare un gruppo di sicurezza per altre implementazioni di notebook Jupyter, puoi eliminarlo. Per ulteriori informazioni, consulta Eliminare un gruppo di sicurezza nella Amazon EC2 User Guide.

Ottenere informazioni sugli endpoint interattivi con i comandi CLI

Questo argomento tratta le operazioni supportate su un endpoint interattivo diverse da `create-managed-endpoint`.

Recupero dei dettagli dell'endpoint interattivo

Dopo aver creato un endpoint interattivo, è possibile recuperarne i dettagli utilizzando il comando. `describe-managed-endpoint` AWS CLI Inserisci i tuoi valori per *`managed-endpoint-idvirtual-cluster-id`*, e: *`region`*

```
aws emr-containers describe-managed-endpoint --id managed-endpoint-id \
 --virtual-cluster-id virtual-cluster-id --region region
```

L'output è simile al seguente, con l'endpoint specificato, ad esempio ARN, ID e nome.

```
{
   "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-aaaaaaaaaaaaaa",
    "subnetIds": [
        "subnet-11111111111",
        "subnet-22222222222",
        "subnet-33333333333"
    ],
    "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
    "tags": {}
 }
```

Elencazione degli endpoint interattivi associati a un cluster virtuale

Usa il `list-managed-endpoints` AWS CLI comando per recuperare un elenco di tutti gli endpoint interattivi associati a un cluster virtuale specificato. Sostituisci `virtual-cluster-id` con l'ID del cluster virtuale.

```
aws emr-containers list-managed-endpoints --virtual-cluster-id virtual-cluster-id
```

L'output del comando `list-managed-endpoint` è mostrato di seguito:

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

```
}
```

Eliminazione di un endpoint interattivo

Per eliminare un endpoint interattivo associato a un Amazon EMR su un cluster virtuale EKS, usa `delete-managed-endpoint` AWS CLI il comando. Quando elimini un endpoint interattivo, Amazon EMR su EKS rimuove i gruppi di sicurezza predefiniti creati per tale endpoint.

Specifica i valori per i seguenti parametri nel comando:

- --id:: l'identificatore dell'endpoint interattivo che desideri eliminare.
- -- virtual-cluster-id — L'identificatore del cluster virtuale associato all'endpoint interattivo che desideri eliminare. Si tratta dello stesso ID di cluster virtuale specificato al momento della creazione dell'endpoint interattivo.

```
aws emr-containers delete-managed-endpoint --id managed-endpoint-id --virtual-cluster-
id virtual-cluster-id
```

Il comando restituisce un output simile all'esempio seguente per confermare che l'endpoint interattivo è stato eliminato:

```
{
    "id":"8gai4l4exxxxx",
    "virtualClusterId":"0b0qvauoy3ch1nqodxxxxxxxx"
}
```

Leggi, scrivi e carica dati in Amazon S3 Express One Zone con Amazon EMR su EKS

Con le versioni 7.2.0 e successive di Amazon EMR, puoi utilizzare Amazon EMR su EKS con la classe di storage Amazon S3 Express One Zone per migliorare le prestazioni durante l'esecuzione di job e carichi di lavoro. S3 Express One Zone è una classe di storage Amazon S3 a zona singola ad alte prestazioni che offre un accesso ai dati coerente a una cifra in millisecondi per la maggior parte delle applicazioni sensibili alla latenza. Al momento del suo rilascio, S3 Express One Zone offre lo storage di oggetti cloud con la latenza più bassa e le prestazioni più elevate in Amazon S3.

Prerequisiti

Prima di poter utilizzare S3 Express One Zone con Amazon EMR su EKS, è necessario disporre dei seguenti prerequisiti:

- Configurazione di Amazon EMR su EKS completata.
- Dopo aver configurato Amazon EMR su EKS, crea un cluster virtuale.

Nozioni di base su S3 Express One Zone

Segui questi passaggi per iniziare a usare S3 Express One Zone

1. Aggiungi l'`CreateSession`autorizzazione al tuo ruolo di esecuzione del lavoro. Quando S3 Express One Zone esegue inizialmente un'azione simile `GET` o `PUT` su un oggetto S3, la classe di storage chiama per tuo `CreateSession` conto. `LIST` Di seguito è riportato un esempio di come concedere l'`CreateSession`autorizzazione.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3express:*:*:bucket/DOC-EXAMPLE-BUCKET"
```

```
      ],
      "Action": [
        "s3express:CreateSession"
      ],
      "Sid": "AllowS3EXPRESSCreatesession"
    }
  ]
}
```

2. È necessario utilizzare il connettore Apache Hadoop S3A per accedere ai bucket S3 Express, quindi cambia Amazon S3 per utilizzare lo schema di utilizzo del connettore. URIs `s3a` Se non utilizzano lo schema, puoi modificare l'implementazione e gli schemi del file system utilizzati. `s3` `s3n`

   Per modificare lo schema `s3`, specifica le seguenti configurazioni del cluster:

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

   Per modificare lo schema s3n, specifica le seguenti configurazioni del cluster:

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A",
      "fs.s3a.endpoint.region": "us-west-2",
      "fs.s3a.change.detection.mode": "none",
      "fs.s3a.select.enabled": "false"
    }
  },
   {
    "Classification": "spark-defaults",
    "Properties": {
```

```
      "spark.hadoop.fs.s3a.aws.credentials.provider":
 "software.amazon.awssdk.auth.credentials.WebIdentityTokenFileCredentialsProvider",
      "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
    }
  }
]
```

3. Nella tua configurazione spark-submit, usa il provider di credenziali di identità web.

```
"spark.hadoop.fs.s3a.aws.credentials.provider=com.amazonaws.auth.WebIdentityTokenCredential
```

Monitoraggio dei processi

Puoi utilizzare Amazon CloudWatch Events per tenere traccia dei lavori eseguiti su un cluster virtuale Amazon EMR su EKS. È possibile utilizzare gli eventi per tenere traccia dell'attività e dell'integrità di un processo eseguito in un cluster virtuale. Gli argomenti che seguono mostrano come configurare il monitoraggio in modo efficace per mantenere l'integrità delle risorse.

Argomenti

Monitora i lavori con Amazon CloudWatch Events

Amazon EMR su EKS emette eventi quando cambia lo stato dell'esecuzione di un processo. Per ogni evento sono disponibili informazioni, ad esempio la data e l'ora in cui si è verificato, nonché ulteriori dettagli, ad esempio l'ID del cluster virtuale e l'ID dell'esecuzione di processo che ha interessato.

È possibile utilizzare gli eventi per tenere traccia dell'attività e dell'integrità di un processo eseguito in un cluster virtuale. Puoi anche utilizzare Amazon CloudWatch Events per definire un'azione da intraprendere quando l'esecuzione di un job genera un evento che corrisponde a uno schema da te specificato. Gli eventi sono utili per monitorare un'occorrenza specifica durante il ciclo di vita dell'esecuzione di un processo. Ad esempio, è possibile monitorare quando l'esecuzione di un processo cambia stato da `submitted` a `running`. Per ulteriori informazioni sugli CloudWatch eventi, consulta la Amazon EventBridge User Guide.

La tabella seguente elenca gli eventi di Amazon EMR su EKS, insieme allo stato o alla modifica dello stato che l'evento indica, la gravità dell'evento e messaggi relativi agli eventi. Ogni evento è rappresentato come un oggetto JSON inviato automaticamente a un flusso di eventi. L'oggetto JSON include ulteriori dettagli sull'evento. L'oggetto JSON è particolarmente importante quando si impostano le regole per l'elaborazione CloudWatch degli eventi utilizzando Events, perché le regole cercano di corrispondere ai modelli dell'oggetto JSON. Per ulteriori informazioni, consulta Amazon EventBridge Event Patterns e Amazon EMR on EKS Events nella Amazon EventBridge User Guide.

Eventi di modifica dello stato dell'esecuzione di processo

Stato	Gravità	Messaggio
SUBMITTED (INVIATO)	INFO	Job Run *JobRunId* (*JobRunName*) è stato inviato correttamente al cluster virtuale *VirtualClusterId* all'*Time*UTC.
RUNNING (ESECUZIONE IN CORSO)	INFO	Job Run *JobRunId* (*JobRunName*) nel cluster virtuale *VirtualClusterId* è iniziato a*Time*.
COMPLETED	INFO	Job Run *jobRunId* (*JobRunName*) nel cluster virtuale *VirtualClusterId* completato in*Time*. Il Job Run è iniziato a funzionare *Time* e il completamento ha richiesto alcuni *Num* minuti.
CANCELLED	WARN	La richiesta di annullamento è riuscita per Job Run *JobRunId* (*JobRunName*) nel cluster virtuale *VirtualClusterId* all'indirizzo *Time* e Job Run è ora annullato.
Non riuscito	ERRORE	Job Run *JobRunId* (*JobRunName*) nel cluster virtuale *VirtualClusterId* non è riuscito a*Time*.

Automatizza Amazon EMR su EKS con Events CloudWatch

Puoi utilizzare Amazon CloudWatch Events per automatizzare AWS i tuoi servizi e rispondere a eventi di sistema come problemi di disponibilità delle applicazioni o modifiche delle risorse. Gli eventi derivanti dai AWS servizi vengono trasmessi a CloudWatch Events quasi in tempo reale. Puoi compilare regole semplici che indichino quali eventi sono considerati di interesse per te e quali operazioni automatizzate intraprendere quando un evento corrisponde a una regola. Le azioni che possono essere attivate automaticamente includono le seguenti:

- Invocare una funzione AWS Lambda
- Richiamo del comando Amazon EC2 Run

- Inoltro dell'evento a Amazon Kinesis Data Streams
- Attivazione di una macchina a stati AWS Step Functions
- Notifica di un argomento Amazon Simple Notification Service (SNS) o di una coda Amazon Simple Queue Service (SQS)

Alcuni esempi di utilizzo di CloudWatch Events with Amazon EMR su EKS includono:

- Attivazione di una funzione Lambda quando un'esecuzione di processo ha esito positivo
- Notifica di un argomento Amazon SNS quando un'esecuzione di processo ha esito negativo

CloudWatch Gli eventi per "`detail-type:`" "`EMR Job Run State Change`" vengono generati da Amazon EMR su EKS per `SUBMITTED` `RUNNING` `CANCELLED`, `FAILED` e modifiche di `COMPLETED` stato.

Esempio: impostare una regola che richiami Lambda

Utilizza i seguenti passaggi per impostare una regola CloudWatch Events che richiami Lambda quando è presente un evento «EMR Job Run State Change».

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run State Change"]}'
```

Aggiungi la funzione Lambda di tua proprietà come nuova destinazione e concedi a CloudWatch Events il permesso di richiamare la funzione Lambda come segue. Sostituisci *123456789012* con l'ID del tuo account.

```
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**
>
> Non è possibile scrivere un programma che dipenda dall'ordine o dall'esistenza di eventi di notifica, poiché questi ultimi potrebbero essere fuori sequenza o mancanti. Gli eventi vengono emessi secondo il principio del massimo sforzo.

Monitora il driver pod del processo con una politica di riprova utilizzando Amazon Events CloudWatch

Utilizzando CloudWatch gli eventi, puoi monitorare i driver pod che sono stati creati in lavori che prevedono politiche di riprova. Per ulteriori informazioni sul tagging, consulta Monitoraggio di un processo con una policy di ripetizionein questa guida.

Gestione dei cluster virtuali

Un cluster virtuale è uno spazio dei nomi Kubernetes con cui è registrato Amazon EMR. Consente di creare, descrivere, elencare ed eliminare cluster virtuali. Non consumano risorse aggiuntive nel sistema. Un singolo cluster virtuale esegue la mappatura a un singolo spazio dei nomi Kubernetes. Data questa relazione, è possibile modellare cluster virtuali nello stesso modo in cui si modellano gli spazi dei nomi Kubernetes per soddisfare le proprie esigenze. Fai riferimento ai possibili casi d'uso nella documentazione della Panoramica sui concetti di Kubernetes.

Per registrare Amazon EMR con uno spazio dei nomi Kubernetes in un cluster Amazon EKS, è necessario il nome del cluster EKS e lo spazio dei nomi impostato per l'esecuzione del carico di lavoro. I cluster registrati in Amazon EMR sono chiamati cluster virtuali perché non gestiscono l'elaborazione fisica o l'archiviazione, ma puntano a uno spazio dei nomi Kubernetes in cui è pianificato il carico di lavoro.

> **Note**
>
> Prima di creare un cluster virtuale, è necessario innanzitutto completare i passaggi 1-8 indicati in Configurazione di Amazon EMR su EKS.

Argomenti

- Creazione di un cluster virtuale
- Elenco dei cluster virtuali
- Descrizione di un cluster virtuale
- Eliminazione di un cluster virtuale
- Stati dei cluster virtuali

Creazione di un cluster virtuale

Esegui il comando seguente per creare un cluster virtuale registrando Amazon EMR con uno spazio dei nomi in un cluster EKS. Sostituisci *virtual_cluster_name* con un nome che fornisci per il tuo cluster virtuale. Sostituisci *eks_cluster_name* con il nome del cluster EKS. Sostituisci *namespace_name* con lo spazio dei nomi con cui desideri registrare Amazon EMR.

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

In alternativa, è possibile creare un file JSON che includa i parametri richiesti per il cluster virtuale, come illustrato nell'esempio seguente.

```
{
    "name": "virtual_cluster_name",
    "containerProvider": {
        "type": "EKS",
        "id": "eks_cluster_name",
        "info": {
            "eksInfo": {
                "namespace": "namespace_name"
            }
        }
    }
}
```

Successivamente, esegui il seguente comando `create-virtual-cluster` con il percorso del file JSON.

```
aws emr-containers create-virtual-cluster \
--cli-input-json file://./create-virtual-cluster-request.json
```

> **Note**
>
> Per convalidare la corretta creazione di un cluster virtuale, visualizza lo stato dei cluster virtuali utilizzando l'operazione `list-virtual-clusters` o andando alla pagina Virtual Clusters (Cluster virtuali) nella console di Amazon EMR.

Elenco dei cluster virtuali

Per visualizzare lo stato dei cluster virtuali, esegui il seguente comando.

```
aws emr-containers list-virtual-clusters
```

Descrizione di un cluster virtuale

Esegui il comando seguente per ottenere maggiori dettagli su un cluster virtuale, come ad esempio lo spazio dei nomi, lo stato e la data di registrazione. Sostituiscilo *123456* con il tuo ID del cluster virtuale.

```
aws emr-containers describe-virtual-cluster --id 123456
```

Eliminazione di un cluster virtuale

Esegui questo comando per eliminare un cluster virtuale. *123456*Sostituiscilo con il tuo ID del cluster virtuale.

```
aws emr-containers delete-virtual-cluster --id 123456
```

Stati dei cluster virtuali

Nella tabella seguente vengono descritti i quattro stati possibili di un cluster virtuale.

State	Descrizione
RUNNING	Il cluster virtuale è nello stato RUNNING.
TERMINATING	È in corso la terminazione richiesta del cluster virtuale.
TERMINATED	La terminazione richiesta è stata completata.
ARRESTED	Terminazione richiesta non riuscita a causa di autorizzazioni insufficienti.

Tutorial per Amazon EMR su EKS

La presente sezione descrive casi d'uso comuni per l'uso di applicazioni Amazon EMR su EKS. Ogni applicazione è specializzata e può eseguire passaggi unici per la configurazione. Questi argomenti forniscono istruzioni per l'utilizzo di ciascuna applicazione.

Argomenti

Utilizzo di Delta Lake con Amazon EMR su EKS

Delta Lake è un framework di storage open source per la creazione di un'architettura Lakehouse. Di seguito viene illustrato come configurarlo per l'uso.

Utilizzo di Delta Lake con Amazon EMR su applicazioni EKS

1. Quando avvii un'esecuzione di processo per inviare un processo Spark nella configurazione dell'applicazione, includi i file JAR di Delta Lake:

```
--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"}}'
```

> **Note**
>
> Le versioni 7.0.0 e successive di Amazon EMR utilizzano Delta Lake 3.0, che viene rinominato in. `delta-core.jar` `delta-spark.jar` Se utilizzi Amazon EMR versione 7.0.0 o successive, assicurati di utilizzare il nome file corretto, come nell'esempio seguente:
>
> ```
> --jars local:///usr/share/aws/delta/lib/delta-spark.jar
> ```

2. Includi la configurazione aggiuntiva di Delta Lake e usa AWS Glue Data Catalog come 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
           }
        }]}'
```

Utilizzo di Apache Iceberg con Amazon EMR su EKS

Il runtime JAR per Iceberg contiene le classi Iceberg necessarie per il supporto del runtime Spark. La procedura seguente mostra come avviare l'esecuzione di un job utilizzando il runtime Iceberg spark.

Utilizzo di Apache Iceberg con Amazon EMR su applicazioni EKS

1. Quando avvii un'esecuzione di processo per inviare un processo Spark nella configurazione dell'applicazione, includi il file JAR del runtime di Iceberg Spark:

```
--job-driver '{"sparkSubmitJobDriver" : {"sparkSubmitParameters" : "--jars
 local:///usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar"}}'
```

2. Includi la configurazione aggiuntiva di Iceberg:

```
--configuration-overrides '{
    "applicationConfiguration": [
    "classification" : "spark-defaults",
    "properties" : {
        "spark.sql.catalog.dev.warehouse" : "s3://amzn-s3-demo-bucket/EXAMPLE-
PREFIX/ ",
        "spark.sql.extensions ":"
 org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions ",
        "spark.sql.catalog.dev" : "org.apache.iceberg.spark.SparkCatalog",
        "spark.sql.catalog.dev.catalog-impl" :
 "org.apache.iceberg.aws.glue.GlueCatalog",
        "spark.sql.catalog.dev.io-impl": "org.apache.iceberg.aws.s3.S3FileIO"
        }
    ]
}'
```

Per saperne di più sulle versioni di rilascio di Apache Iceberg per EMR, consulta la sezione Iceberg release history (Cronologia delle versioni di Iceberg).

Configurazioni delle sessioni Spark per l'integrazione del catalogo

Configurazioni delle sessioni Spark per l'integrazione del catalogo Iceberg AWS Glue

Questo esempio mostra come integrare Iceberg con: Crawler di AWS Glue

```
spark-sql \
  --conf spark.sql.catalog.rms = org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.rms.type = glue \
  --conf spark.sql.catalog.rms.glue.id = glue RMS catalog ID \
  --conf spark.sql.catalog.rms.glue.account-id = AWS account ID \

  --conf spark.sql.extensions=
    org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

Di seguito è riportata una query di esempio:

```
SELECT * FROM rms.rmsdb.table1
```

Configurazioni delle sessioni Spark per l'integrazione del catalogo Iceberg REST AWS Glue

Questo esempio mostra come integrare Iceberg REST con: Crawler di AWS Glue

```
spark-sql \
  --conf spark.sql.catalog.rms = org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.rms.type = rest \
  --conf spark.sql.catalog.rms.warehouse = glue RMS catalog ID \
  --conf spark.sql.catalog.rms.uri = glue endpoint URI/iceberg \
  --conf spark.sql.catalog.rms.rest.sigv4-enabled = true \
  --conf spark.sql.catalog.rms.rest.signing-name = glue \

  --conf spark.sql.extensions=
    org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

Di seguito è riportata una query di esempio:

```
SELECT * FROM rms.rmsdb.table1
```

Questa configurazione funziona solo per Redshift Managed Storage. FGAC per Amazon S3 non è supportato.

Usando PyFlink

Amazon EMR su EKS supporta le versioni 6.15.0 e successive. PyFlink Se disponi già di uno PyFlink script, puoi eseguire una delle seguenti operazioni:

- Crea un'immagine personalizzata con lo PyFlink script incluso.
- Carica lo script in una posizione Amazon S3

Se non disponi già di uno script, puoi utilizzare il seguente esempio per avviare un PyFlink lavoro. Questo esempio recupera lo script da S3. Se utilizzi un'immagine personalizzata con lo script già incluso nell'immagine, devi aggiornare il percorso dello script nella posizione in cui lo hai archiviato. Se lo script si trova in una posizione S3, Amazon EMR su EKS recupererà lo script e lo posizionerà nella directory `/opt/flink/usrlib/` del contenitore Flink.

```
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
```

AWS Usare Glue con Flink

Amazon EMR su EKS con Apache Flink nelle versioni 6.15.0 e successive supporta l'utilizzo di AWS Glue Data Catalog come archivio di metadati per flussi di lavoro SQL in streaming e batch.

È innanzitutto necessario creare un database AWS Glue denominato `default` che funga da Flink SQL Catalog. Questo catalogo Flink memorizza metadati come database, tabelle, partizioni, viste, funzioni e altre informazioni necessarie per accedere ai dati in altri sistemi esterni.

```
aws glue create-database \
    --database-input "{\"Name\":\"default\"}"
```

Per abilitare il supporto di AWS Glue, usa una `FlinkDeployment` specifica. Questa specifica di esempio utilizza uno script Python per emettere rapidamente alcune istruzioni SQL Flink per interagire con il AWS catalogo Glue.

```
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
```

Di seguito è riportato un esempio di come potrebbe apparire PyFlink lo script.

```
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()
```

Utilizzo di Apache Hudi con Apache Flink

Apache Hudi è un framework di gestione dei dati open source con operazioni a livello di record come inserimento, aggiornamento, annullamento ed eliminazione che puoi utilizzare per semplificare la gestione dei dati e lo sviluppo di pipeline di dati. In combinazione con una gestione efficiente dei dati in Amazon S3, Hudi consente di importare e aggiornare i dati in tempo reale. Hudi conserva i metadati di tutte le operazioni eseguite sul set di dati, in modo che tutte le azioni rimangano atomiche e coerenti.

Apache Hudi è disponibile su Amazon EMR su EKS con Apache Flink con le versioni 7.2.0 e successive di Amazon EMR. Consulta i passaggi seguenti per scoprire come iniziare e inviare lavori in Apache Hudi.

Inviare un lavoro in Apache Hudi

Segui i passaggi seguenti per imparare a inviare un lavoro in Apache Hudi.

1. Crea un database AWS Glue denominato`default`.

   ```
   aws glue create-database --database-input "{\"Name\":\"default\"}"
   ```

2. Segui l'esempio SQL dell'operatore Flink Kubernetes per creare il file. `flink-sql-runner.jar`

3. Crea uno script SQL Hudi come il seguente.

```
CREATE CATALOG hudi_glue_catalog WITH (
'type' = 'hudi',
'mode' = 'hms',
'table.external' = 'true',
'default-database' = 'default',
'hive.conf.dir' = '/glue/confs/hive/conf/',
'catalog.path' = 's3://<hudi-example-bucket>/FLINK_HUDI/warehouse/'
);

USE CATALOG hudi_glue_catalog;
CREATE DATABASE IF NOT EXISTS hudi_db;
use hudi_db;

CREATE TABLE IF NOT EXISTS hudi-flink-example-table(
    uuid VARCHAR(20),
    name VARCHAR(10),
    age INT,
    ts TIMESTAMP(3),
    `partition` VARCHAR(20)
)
PARTITIONED BY (`partition`)
WITH (
  'connector' = 'hudi',
  'path' = 's3://<hudi-example-bucket>/hudi-flink-example-table',
  'hive_sync.enable' = 'true',
  'hive_sync.mode' = 'glue',
  'hive_sync.table' = 'hudi-flink-example-table',
  'hive_sync.db' = 'hudi_db',
  'compaction.delta_commits' = '1',
  'hive_sync.partition_fields' = 'partition',
  'hive_sync.partition_extractor_class' =
 'org.apache.hudi.hive.MultiPartKeysValueExtractor',
  'table.type' = 'COPY_ON_WRITE'
);

EXECUTE STATEMENT SET
BEGIN

INSERT INTO hudi-flink-example-table VALUES
    ('id1','Alex',23,TIMESTAMP '1970-01-01 00:00:01','par1'),
    ('id2','Stephen',33,TIMESTAMP '1970-01-01 00:00:02','par1'),
```

```
    ('id3','Julian',53,TIMESTAMP '1970-01-01 00:00:03','par2'),
    ('id4','Fabian',31,TIMESTAMP '1970-01-01 00:00:04','par2'),
    ('id5','Sophia',18,TIMESTAMP '1970-01-01 00:00:05','par3'),
    ('id6','Emma',20,TIMESTAMP '1970-01-01 00:00:06','par3'),
    ('id7','Bob',44,TIMESTAMP '1970-01-01 00:00:07','par4'),
    ('id8','Han',56,TIMESTAMP '1970-01-01 00:00:08','par4');

END;
```

4. Carica lo script SQL Hudi e il `flink-sql-runner.jar` file in una posizione S3.
5. Nel tuo file `FlinkDeployments` YAML, imposta su. `hudi.enabled` `true`

```
spec:
  flinkConfiguration:
    hudi.enabled: "true"
```

6. Crea un file YAML per eseguire la configurazione. Questo file di esempio è denominato. `hudi-write.yaml`

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: hudi-write-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    hudi.enabled: "true"
  executionRoleArn: "<JobExecutionRole>"
  emrReleaseLabel: "emr-7.12.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: local:///opt/flink/usrlib/flink-sql-runner.jar
    args: ["/opt/flink/scripts/hudi-write.sql"]
```

```
    parallelism: 1
    upgradeMode: stateless
  podTemplate:
    spec:
      initContainers:
        - name: flink-sql-script-download
          args:
            - s3
            - cp
            - s3://<s3_location>/hudi-write.sql
            - /flink-scripts
          image: amazon/aws-cli:latest
          imagePullPolicy: Always
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          volumeMounts:
            - mountPath: /flink-scripts
              name: flink-scripts
        - name: flink-sql-runner-download
          args:
            - s3
            - cp
            - s3://<s3_location>/flink-sql-runner.jar
            - /flink-artifacts
          image: amazon/aws-cli:latest
          imagePullPolicy: Always
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          volumeMounts:
            - mountPath: /flink-artifacts
              name: flink-artifact
      containers:
        - name: flink-main-container
          volumeMounts:
            - mountPath: /opt/flink/scripts
              name: flink-scripts
            - mountPath: /opt/flink/usrlib
              name: flink-artifact
      volumes:
        - emptyDir: {}
          name: flink-scripts
        - emptyDir: {}
```

```
            name: flink-artifact
```

7. Invia un job Flink Hudi all'operatore Flink Kubernetes.

```
kubectl apply -f hudi-write.yaml
```

Uso dell'acceleratore RAPIDS per Apache Spark con Amazon EMR su EKS

Con Amazon EMR su EKS, puoi eseguire processi per l'acceleratore RAPIDS di Nvidia per Apache Spark. Questo tutorial spiega come eseguire i job Spark utilizzando RAPIDS su tipi di istanze di unità di elaborazione EC2 grafica (GPU). Il tutorial utilizza le seguenti versioni:

- Amazon EMR su EKS versione di rilascio 6.9.0 e successive
- Apache Spark 3.x

Puoi accelerare Spark con i tipi di istanze Amazon EC2 GPU utilizzando il plug-in Nvidia RAPIDS Accelerator for Apache Spark. Quando si utilizzano queste tecnologie insieme, si accelerano le pipeline di data science senza dover apportare modifiche al codice. Ciò riduce il tempo di esecuzione necessario per l'elaborazione dei dati e l'addestramento dei modelli. Facendo di più in meno tempo, si riducono i costi di infrastruttura.

Prima di iniziare, assicurati di disporre delle seguenti risorse.

- Cluster virtuale Amazon EMR su EKS
- Cluster Amazon EKS con un gruppo di nodi abilitato alla GPU

Un cluster virtuale Amazon EKS è un handle registrato per lo spazio dei nomi Kubernetes su un cluster Amazon EKS ed è gestito da Amazon EMR su EKS. L'handle consente ad Amazon EMR di utilizzare lo spazio dei nomi Kubernetes come destinazione per l'esecuzione di processi. Per ulteriori informazioni su come configurare un cluster virtuale, consulta la sezione Configurazione di Amazon EMR su EKS in questa guida.

È necessario configurare il cluster virtuale Amazon EKS con un gruppo di nodi con istanze GPU. È necessario configurare i nodi con un plug-in per dispositivi Nvidia. Per ulteriori informazioni, consulta la sezione Managed node groups (Gruppi di nodi gestiti).

Per configurare il cluster Amazon EKS per aggiungere gruppi di nodi compatibili con GPU, completa la seguente procedura:

Aggiunta di gruppi di nodi abilitati per GPU

1. Crea un gruppo di nodi abilitati per GPU con il comando create-nodegroup. Assicurati di sostituire i parametri corretti per il tuo cluster Amazon EKS. Utilizza un tipo di istanza che supporti Spark RAPIDS, ad esempio P4, P3, G5 o 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. Installa il plug-in per dispositivi Nvidia nel tuo cluster per emettere il numero di GPUs su ogni nodo del cluster e per eseguire contenitori abilitati alla GPU nel cluster. Esegui il comando seguente per installare il plug-in:

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/
nvidia-device-plugin.yml
```

3. Per verificare quanti ne GPUs sono disponibili su ciascun nodo del cluster, esegui il seguente comando:

```
kubectl get nodes  "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

Esecuzione di un processo Spark RAPIDS

1. Invia un processo Spark RAPIDS al cluster Amazon EMR su EKS. Il codice seguente mostra un esempio di comando per avviare il processo. La prima volta che esegui il processo, potrebbero essere necessari alcuni minuti per scaricare l'immagine e memorizzarla nella cache sul nodo.

```
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. Per verificare che l'acceleratore RAPIDS per Spark sia abilitato, controlla i log dei driver Spark. Questi log vengono archiviati in CloudWatch o nella posizione S3 specificata quando esegui il comando. `start-job-run` L'esempio seguente mostra l'aspetto generale delle righe del log:

```
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. Per visualizzare le operazioni che verranno eseguite su una GPU, completa la seguente procedura per abilitare la registrazione aggiuntiva. Prendi nota della configurazione "`spark.rapids.sql.explain : ALL`".

```
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"}}}'
```

Il comando precedente è un esempio di un processo che utilizza la GPU. L'output dovrebbe assomigliare all'esempio seguente. Per comprendere l'output, fai riferimento a questa legenda:

- *: contrassegna un'operazione che può essere eseguita su una GPU
- !: contrassegna un'operazione che non può essere eseguita su una GPU
- @: contrassegna un'operazione che può essere eseguita su una GPU ma che non verrà eseguita perché rientra in un piano che non può essere eseguito su una GPU

```
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
```

Uso dell'integrazione di Amazon Redshift per Apache Spark in Amazon EMR su EKS

Con la versione 6.9.0 e successive di Amazon EMR, ogni immagine del rilascio include un connettore tra Apache Spark e Amazon Redshift. In questo modo, puoi utilizzare Spark su Amazon EMR per elaborare i dati archiviati in Amazon Redshift. L'integrazione si basa sul connettore `spark-redshift` open source. Per Amazon EMR su EKS, l'integrazione di Amazon Redshift per Apache Spark è inclusa come integrazione nativa.

Argomenti

- Avvio di un'applicazione Spark utilizzando l'integrazione di Amazon Redshift per Apache Spark
- Autenticazione con l'integrazione di Amazon Redshift per Apache Spark
- Lettura e scrittura da e su Amazon Redshift
- Considerazioni e limitazioni relative all'utilizzo del connettore Spark

Avvio di un'applicazione Spark utilizzando l'integrazione di Amazon Redshift per Apache Spark

Per utilizzare l'integrazione, devi passare le dipendenze Spark Redshift richieste con il processo Spark. È necessario utilizzare `--jars` per includere le librerie relative al connettore Redshift. Per vedere le altre posizioni dei file supportate dall'opzione `--jars`, consulta la sezione Advanced Dependency Management (Gestione avanzata delle dipendenze) nella documentazione di Apache Spark.

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

Per avviare un'applicazione Spark con l'integrazione di Amazon Redshift per Apache Spark su Amazon EMR su EKS rilascio 6.9.0 o successivo, utilizza il seguente comando di esempio. Come vedrai, i percorsi elencati con l'opzione `--conf spark.jars` sono i percorsi predefiniti per i file JAR.

```
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"
                            }
            }'
```

Autenticazione con l'integrazione di Amazon Redshift per Apache Spark

Le seguenti sezioni mostrano le opzioni di autenticazione con Amazon Redshift durante l'integrazione con Apache Spark. Le sezioni mostrano come recuperare le credenziali di accesso e anche dettagli sull'utilizzo del driver JDBC con l'autenticazione IAM.

Utilizzalo Gestione dei segreti AWS per recuperare le credenziali e connetterti ad Amazon Redshift

Puoi archiviare le credenziali in Secrets Manager per autenticarti in modo sicuro su Amazon Redshift. Puoi fare in modo che il processo Spark chiami l'API `GetSecretValue` per recuperare le credenziali:

```
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
```

Utilizzo dell'autenticazione basata su IAM con il ruolo di esecuzione di processo Amazon EMR su EKS

A partire da Amazon EMR su EKS rilascio 6.9.0, il driver Amazon Redshift JDBC versione 2.1 o superiore è integrato nell'ambiente. Con il driver JDBC 2.1 e versioni successive, è possibile specificare l'URL di JDBC evitando di includere il nome utente e la password non elaborati. È

invece possibile specificare uno schema `jdbc:redshift:iam://`. Ciò istruisce il driver JDBC di utilizzare il tuo ruolo di esecuzione di processo Amazon EMR su EKS per il recupero automatico delle credenziali.

Per ulteriori informazioni, consulta la sezione Configure a JDBC or ODBC connection to use IAM credentials (Configurazione di una connessione JDBC oppure ODBC per l'utilizzo delle credenziali IAM) nella Guida alla gestione di Amazon Redshift.

L'URL di esempio seguente utilizza uno schema `jdbc:redshift:iam://`.

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Le autorizzazioni seguenti sono necessarie per il ruolo di esecuzione di processo quando soddisfa le condizioni richieste.

Autorizzazione	Condizioni richieste per il ruolo di esecuzione di processo
`redshift:GetClusterCredentials`	Obbligatoria affinché il driver JDBC recuperi le credenziali da Amazon Redshift
`redshift:DescribeCluster`	Obbligatoria se specifichi il cluster Amazon Redshift e Regione AWS nell'URL di JDBC anziché nell'endpoint
`redshift-serverless:GetCredentials`	Obbligatoria affinché il driver JDBC recuperi le credenziali da Amazon Redshift Serverless
`redshift-serverless:GetWorkgroup`	Obbligatoria se utilizzi Amazon Redshift Serverless e specifichi l'URL in termini di nome del gruppo di lavoro e Regione

La policy relativa al ruolo di esecuzione di processo dovrebbe disporre delle seguenti autorizzazioni.

```
{
            "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"
            ]
        }
```

Autenticazione su Amazon Redshift con un driver JDBC

Impostazione di nome utente e password all'interno dell'URL di JDBC

Per autenticare un processo Spark su un cluster Amazon Redshift, è possibile specificare il nome e la password del database Amazon Redshift nell'URL di JDBC.

> **Note**
>
> Se passi le credenziali del database nell'URL, anche chiunque abbia accesso all'URL può accedere alle credenziali. Questo metodo non è generalmente consigliato perché non è un'opzione sicura.

Se la sicurezza non è un problema per la tua applicazione, puoi utilizzare il formato seguente per impostare il nome utente e la password nell'URL di JDBC:

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

Lettura e scrittura da e su Amazon Redshift

I seguenti esempi di codice vengono utilizzati PySpark per leggere e scrivere dati di esempio da e verso un database Amazon Redshift con un'API di origine dati e con SparkSQL.

Data source API

PySpark Utilizzalo per leggere e scrivere dati di esempio da e verso un database Amazon Redshift con un'API di origine dati.

```
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

PySpark Utilizzalo per leggere e scrivere dati di esempio da e verso un database Amazon Redshift utilizzando SparkSQL.

```
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()
```

Considerazioni e limitazioni relative all'utilizzo del connettore Spark

Il connettore Spark supporta diversi modi per gestire le credenziali, configurare la sicurezza e connettersi con altri servizi. AWS Acquisisci familiarità con i consigli contenuti in questo elenco per configurare una connessione funzionale e resiliente.

- Si consiglia di attivare SSL per la connessione JDBC da Spark su Amazon EMR ad Amazon Redshift.
- Come best practice, è consigliabile gestire le credenziali per il cluster Amazon Redshift in Gestione dei segreti AWS . Gestione dei segreti AWS Per un esempio, consulta Utilizzo per recuperare le credenziali per la connessione ad Amazon Redshift.
- Si consiglia di passare un ruolo IAM con il parametro `aws_iam_role` per il parametro di autenticazione di Amazon Redshift.
- Il parametro `tempformat` attualmente non supporta il formato Parquet.
- L'URI `tempdir` indica una posizione Amazon S3. Questa directory temporanea non viene pulita in automatico e quindi potrebbe generare costi aggiuntivi.
- Prendi in considerazione i seguenti consigli per Amazon Redshift:
  - Si consiglia di bloccare l'accesso pubblico al cluster Amazon Redshift.
  - Si consiglia di attivare la registrazione di log di verifica di Amazon Redshift.

  - Si consiglia di attivare la crittografia dei dati inattivi di Amazon Redshift.
- Prendi in considerazione i seguenti consigli per Amazon S3:
  - Si consiglia di bloccare l'accesso pubblico ai bucket Amazon S3.
  - Si consiglia di utilizzare la crittografia lato server di Amazon S3 per crittografare i bucket S3 utilizzati.
  - Si consiglia di utilizzare le policy del ciclo di vita di Amazon S3 per definire le regole di conservazione del bucket S3.
  - Amazon EMR verifica sempre il codice importato dall'open source nell'immagine. Per motivi di sicurezza, non supportiamo la codifica delle chiavi di AWS accesso nell'`tempdir`URI come metodo di autenticazione da Spark ad Amazon S3.

Per ulteriori informazioni sull'utilizzo del connettore e dei parametri supportati, consulta le seguenti risorse:

- Amazon Redshift integration for Apache Spark (Integrazione di Amazon Redshift per Apache Spark) nella Guida alla gestione di Amazon Redshift
- Il repository della community `spark-redshift` su Github

Uso di Volcano come pianificatore personalizzato per Apache Spark in Amazon EMR su EKS

Con Amazon EMR su EKS, puoi utilizzare l'operatore Spark o spark-submit per eseguire processi Spark con pianificatori personalizzati di Kubernetes. Questo tutorial spiega come eseguire processi Spark con un pianificatore Volcano su una coda personalizzata.

Panoramica

Volcano può aiutarti a gestire la pianificazione di Spark con funzioni avanzate come la pianificazione delle code, la pianificazione della quota equa e la prenotazione delle risorse. Per ulteriori informazioni sui vantaggi di Volcano, consulta Perché Spark sceglie Volcano come pianificatore in batch integrato su Kubernetes nel blog CNCF di Linux Foundation.

Installazione e configurazione di Volcano

1. Scegli uno dei comandi kubectl seguenti per installare Volcano, a seconda delle tue esigenze architettoniche:

```
# 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. Prepara una coda Volcano campione. Una coda è una raccolta di. PodGroups La coda adotta FIFO ed è la base per la divisione delle risorse.

```
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. Carica un PodGroup manifesto di esempio su Amazon S3. PodGroup è un gruppo di pod con forti associazioni. In genere si utilizza un PodGroup per la pianificazione in batch. Inviate il seguente esempio PodGroup alla coda definita nel passaggio precedente.

```
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
```

Esecuzione di un'applicazione Spark con pianificatore Volcano con l'operatore Spark

1. Se non l'hai già fatto, completa la procedura nelle seguenti sezioni per effettuare la configurazione:

   a. Installazione e configurazione di Volcano

   b. Configurazione dell'operatore Spark per Amazon EMR su EKS

   c. Installazione dell'operatore Spark

   Quando esegui il comando `helm install spark-operator-demo`, includi gli argomenti seguenti:

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

2. Crea un file di definizione `SparkApplication` `spark-pi.yaml` con la configurazione di `batchScheduler`.

   ```
   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. Invia l'applicazione Spark con il comando seguente. Questa operazione crea anche un oggetto `SparkApplication` denominato `spark-pi`:

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

4. Controlla gli eventi dell'oggetto `SparkApplication` con il comando seguente:

```
kubectl describe pods spark-pi-driver --namespace spark-operator
```

Il primo evento di pod mostrerà che Volcano ha programmato i pod:

```
Type    Reason     Age   From                Message
----    ------     ----  ----                -------
Normal  Scheduled  23s   volcano             Successfully assigned default/spark-
pi-driver to integration-worker2
```

Esecuzione di un'applicazione Spark con pianificatore Volcano con **`spark-submit`**

1. Per prima cosa, completa le fasi indicate nella sezione Configurazione di spark-submit per Amazon EMR su EKS. Devi creare la tua distribuzione `spark-submit` con il supporto di Volcano. Per ulteriori informazioni, consulta la sezione Creazione di Uso di Volcano come pianificatore personalizzato per Spark su Kubernetes nella documentazione di Apache Spark.
2. Imposta i valori delle seguenti variabili di ambiente:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Invia l'applicazione Spark con il comando seguente:

```
$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. Controlla gli eventi dell'oggetto `SparkApplication` con il comando seguente:

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

Il primo evento di pod mostrerà che Volcano ha programmato i pod:

```
Type    Reason     Age   From               Message
----    ------     ----  ----               -------
Normal  Scheduled  23s   volcano            Successfully assigned default/spark-
pi-driver to integration-worker2
```

Utilizzo YuniKorn come scheduler personalizzato per Apache Spark su Amazon EMR su EKS

Con Amazon EMR su EKS, puoi utilizzare l'operatore Spark o spark-submit per eseguire processi Spark con pianificatori personalizzati di Kubernetes. Questo tutorial spiega come eseguire i job Spark con uno YuniKorn scheduler su una coda personalizzata e la pianificazione di gruppo.

Panoramica

Apache YuniKorn può aiutarti a gestire la pianificazione di Spark con una pianificazione basata sulle app, in modo da avere un controllo preciso sulle quote e sulle priorità delle risorse. Con la pianificazione in gruppo, YuniKorn pianifica un'app solo quando è possibile soddisfare la richiesta minima di risorse per l'app. Per ulteriori informazioni, consulta What is gang scheduling nel sito di documentazione di Apache YuniKorn .

Crea il tuo cluster e preparati per YuniKorn

Utilizza la procedura seguente per implementare un cluster Amazon EKS. Puoi modificare la Regione AWS (`region`) e le zone di disponibilità (`availabilityZones`).

1. Definisci il cluster Amazon EKS:

```
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. Crea il cluster:

```
eksctl create cluster -f eks-cluster.yaml
```

3. Crea lo spazio dei nomi `spark-job` in cui eseguirai il processo Spark:

```
kubectl create namespace spark-job
```

4. Quindi, crea un ruolo e un'associazione di ruoli Kubernetes. Ciò è necessario per l'account di servizio utilizzato dall'esecuzione di processo Spark.

   a. Definisci l'account di servizio, il ruolo e l'associazione di ruoli per i processi Spark.

```
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. Applica la definizione del ruolo e dell'associazione di ruoli Kubernetes con il comando seguente:

```
kubectl apply -f emr-job-execution-rbac.yaml
```

Installazione e configurazione YuniKorn

1. Utilizza il seguente comando kubectl per creare uno spazio dei nomi `yunikorn` per l'implementazione del pianificatore Yunikorn:

```
kubectl create namespace yunikorn
```

2. Per installare il pianificatore, esegui i seguenti comandi Helm:

```
helm repo add yunikorn https://apache.github.io/yunikorn-release
```

```
helm repo update
```

```
helm install yunikorn yunikorn/yunikorn --namespace yunikorn
```

Esegui un'applicazione Spark con YuniKorn scheduler con l'operatore Spark

1. Se non l'hai già fatto, completa la procedura nelle seguenti sezioni per effettuare la configurazione:

   a. Crea il tuo cluster e preparati per YuniKorn

   b. Installazione e configurazione YuniKorn

   c. Configurazione dell'operatore Spark per Amazon EMR su EKS

   d. Installazione dell'operatore Spark

   Quando esegui il comando `helm install spark-operator-demo`, includi gli argomenti seguenti:

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

2. Crea un file `spark-pi.yaml` di definizione `SparkApplication`.

   Per utilizzarlo YuniKorn come strumento di pianificazione per i tuoi lavori, devi aggiungere determinate annotazioni ed etichette alla definizione dell'applicazione. Le annotazioni e le etichette specificano la coda per il processo e la strategia di pianificazione che desideri utilizzare.

   Nell'esempio seguente, l'annotazione `schedulingPolicyParameters` imposta la pianificazione di gruppo per l'applicazione. Quindi, l'esempio crea gruppi di attività per specificare la capacità minima che deve essere disponibile prima di pianificare i pod per l'avvio dell'esecuzione di processo. Infine, specifica nella definizione del gruppo di attività di utilizzare i gruppi di nodi con l'etichetta `"app": "spark"`, come definito nella sezione Crea il tuo cluster e preparati per YuniKorn.

```
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. Invia l'applicazione Spark con il comando seguente. Questa operazione crea anche un oggetto `SparkApplication` denominato `spark-pi`:

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

4. Controlla gli eventi dell'oggetto `SparkApplication` con il comando seguente:

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

Il primo evento pod mostrerà che YuniKorn ha pianificato i pod:

```
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
```

Esegui un'applicazione Spark con scheduler con YuniKorn `spark-submit`

1. Per prima cosa, completa le fasi indicate nella sezione Configurazione di spark-submit per Amazon EMR su EKS.
2. Imposta i valori delle seguenti variabili di ambiente:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Invia l'applicazione Spark con il comando seguente:

   Nell'esempio seguente, l'annotazione `schedulingPolicyParameters` imposta la pianificazione di gruppo per l'applicazione. Quindi, l'esempio crea gruppi di attività per specificare la capacità minima che deve essere disponibile prima di pianificare i pod per l'avvio dell'esecuzione di processo. Infine, specifica nella definizione del gruppo di attività di utilizzare i gruppi di nodi con l'etichetta `"app": "spark"`, come definito nella sezione Crea il tuo cluster e preparati per YuniKorn.

```
$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. Controlla gli eventi dell'oggetto `SparkApplication` con il comando seguente:

```
kubectl describe pod spark-driver-pod --namespace spark-job
```

Il primo evento pod mostrerà che YuniKorn ha pianificato i pod:

```
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
```

Sicurezza in Amazon EMR su EKS

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di data center e architetture di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra te e te. AWS Il modello di responsabilità condivisa descrive questo aspetto come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per ulteriori informazioni sui programmi di conformità applicabili ad Amazon EMR, consulta AWS Services in Scope by Compliance Program Program.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. Sei anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della tua azienda e le leggi e normative vigenti.

Questa documentazione aiuta a comprendere come applicare il modello di responsabilità condivisa quando si utilizza Amazon EMR su EKS. Gli argomenti seguenti descrivono come configurare Amazon EMR su EKS per soddisfare gli obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse Amazon EMR su EKS.

Argomenti

- Sicurezza dell'infrastruttura in Amazon EMR su EKS
- Analisi della configurazione e delle vulnerabilità
- Connessione ad Amazon EMR su EKS con un endpoint VPC di interfaccia
- Configurazione dell'accesso multi-account per Amazon EMR su EKS

Best practice di sicurezza per Amazon EMR su EKS

Amazon EMR su EKS fornisce una serie di funzionalità di sicurezza da prendere in considerazione durante lo sviluppo e l'implementazione delle policy di sicurezza. Le seguenti best practice sono linee guida generali e non rappresentano una soluzione di sicurezza completa. Poiché potrebbero non essere appropriate o sufficienti per il tuo ambiente, prendile come considerazioni utili più che istruzioni.

> **Note**
>
> Per ulteriori best practice in materia di sicurezza, consulta Best practice di sicurezza per Amazon EMR su EKS.

Applicazione del principio del privilegio minimo

Amazon EMR su EKS fornisce una policy di accesso granulare per le applicazioni che utilizzano ruoli IAM, ad esempio i ruoli di esecuzione. Questi ruoli di esecuzione vengono mappati agli account del servizio Kubernetes tramite la policy di affidabilità del ruolo IAM. Amazon EMR su EKS crea pod all'interno di uno spazio dei nomi Amazon EKS registrato che eseguono il codice dell'applicazione fornito dall'utente. I job pod che eseguono il codice dell'applicazione assumono il ruolo di esecuzione quando si connettono ad altri AWS servizi. Si consiglia di concedere ai ruoli di esecuzione solo il set minimo di privilegi richiesti dal processo, ad esempio la copertura dell'applicazione e l'accesso alla destinazione del log. Si consiglia inoltre di verificare i processi per le autorizzazioni su base regolare e a qualsiasi modifica del codice dell'applicazione.

Lista di controllo accessi per gli endpoint

Gli endpoint gestiti possono essere creati solo per i cluster EKS configurati per l'utilizzo di almeno una sottorete privata nel VPC. Questa configurazione limita l'accesso ai bilanciatori del carico creati dagli endpoint gestiti in modo che sia possibile accedervi solo dal VPC. Per una maggiore protezione,

si consiglia di configurare i gruppi di sicurezza con questi bilanciatori del carico in modo che possano limitare il traffico in ingresso a un set selezionato di indirizzi IP.

Ricevi gli ultimi aggiornamenti di sicurezza per le immagini personalizzate

Per utilizzare immagini personalizzate con Amazon EMR su EKS, è possibile installare qualsiasi tipo di dati binari e librerie sull'immagine. L'utente è responsabile dell'applicazione di patch di sicurezza dei dati binari aggiunti all'immagine. Alle immagini di Amazon EMR su EKS vengono regolarmente applicate i patch di sicurezza più recenti. Per ottenere l'immagine più recente, occorre ricostruire le immagini personalizzate ogni volta che è disponibile una nuova versione dell'immagine di base del rilascio di Amazon EMR. Per ulteriori informazioni, consultare Rilasci di Amazon EMR su EKS e Dettagli per la selezione dell'URI di un'immagine di base.

Limitazione dell'accesso alle credenziali del pod

Kubernetes supporta diversi metodi di assegnazione delle credenziali a un pod. Il provisioning di più provider di credenziali può aumentare la complessità del modello di sicurezza. Amazon EMR su EKS ha adottato l'utilizzo di Ruoli IAM per gli account dei servizi (IRSA) come provider di credenziali standard all'interno di uno spazio dei nomi EKS registrato. Altri metodi non sono supportati, tra cui kube2iam, kiam e l'utilizzo di un profilo di istanza dell' EC2 istanza in esecuzione sul cluster.

Isolamento di un codice dell'applicazione non attendibile

Amazon EMR su EKS non controlla l'integrità del codice dell'applicazione inviato dagli utenti del sistema. Se si esegue un cluster virtuale multi-tenant configurato utilizzando più ruoli di esecuzione che possono essere utilizzati per inviare processi da tenant non attendibili che eseguono un codice arbitrario, esiste il rischio che un'applicazione dannosa incrementi i propri privilegi. In questo caso, è consigliabile isolare i ruoli di esecuzione con privilegi simili in un cluster virtuale diverso.

Autorizzazioni per il controllo degli accessi basato su ruoli (RBAC)

Gli amministratori devono controllare rigorosamente le autorizzazioni per il controllo degli accessi basato su ruoli (RBAC) per gli spazi dei nomi gestiti da Amazon EMR su EKS. Come misura minima, le seguenti autorizzazioni non dovrebbero essere concesse agli autori di processi negli spazi dei nomi gestiti da Amazon EMR su EKS.

- Le autorizzazioni Kubernetes RBAC per modificare configmap sono necessarie in quanto Amazon EMR su EKS utilizza Kubernetes configmaps per generare modelli di pod gestiti con il nome dell'account del servizio gestito. Questo attributo non deve essere mutato.

- Kubernetes RBAC consente di accedere ai pod Amazon EMR su EKS per evitare l'accesso a modelli di pod gestiti con il nome SA gestito. Questo attributo non deve essere mutato. Questa autorizzazione può anche dare accesso al token JWT montato nel pod che può quindi essere utilizzato per recuperare le credenziali del ruolo di esecuzione.
- Autorizzazioni RBAC di Kubernetes per creare pod - per impedire agli utenti di creare pod utilizzando un Kubernetes ServiceAccount che può essere mappato a un ruolo IAM con più privilegi rispetto all'utente. AWS
- Autorizzazioni RBAC di Kubernetes per implementare webhook mutanti - per impedire agli utenti di utilizzare il webhook mutante per modificare il nome Kubernetes per i pod creati da Amazon EMR su EKS. ServiceAccount
- Le autorizzazioni Kubernetes RBAC per leggere i segreti di Kubernetes servono a impedire agli utenti di leggere i dati riservati memorizzati in questi segreti.

Limitare l'accesso alle credenziali del ruolo IAM o del profilo dell'istanza del nodegroup

- Ti consigliamo di assegnare AWS autorizzazioni minime ai ruoli IAM di nodegroup. Ciò consente di evitare l'escalation dei privilegi in base al codice che può essere eseguito utilizzando le credenziali del profilo dell'istanza dei nodi di lavoro EKS.
- Per bloccare completamente l'accesso alle credenziali del profilo dell'istanza a tutti i pod eseguiti negli spazi dei nomi gestiti da Amazon EMR su EKS, ti consigliamo di eseguire i comandi `iptables` sui nodi EKS. Per ulteriori informazioni, consulta Limitazione dell'accesso alle credenziali del profilo di EC2 istanza Amazon. In ogni caso, è importante includere correttamente i ruoli IAM degli account di servizio in modo che i pod dispongano di tutte le autorizzazioni necessarie. Ad esempio, al ruolo IAM del nodo vengono assegnate autorizzazioni per estrarre le immagini di container da Amazon ECR. Se a un pod non sono assegnate tali autorizzazioni, il pod non può estrarre le immagini di container da Amazon ECR. È necessario aggiornare anche il plug-in VPC CNI. Per ulteriori informazioni, consulta Spiegazione passo per passo: aggiornamento del plug-in VPC CNI per l'utilizzo dei ruoli IAM per gli account di servizio.

Protezione dei dati

Il modello di responsabilità AWS condivisa si applica alla protezione dei dati in Amazon EMR su EKS. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutto il AWS cloud. L'utente è responsabile del controllo dei contenuti ospitati su questa

infrastruttura. Questo contenuto include la configurazione della protezione e le attività di gestione per i servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, consulta Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il modello di responsabilità AWS condivisa e il post sul blog sul GDPR sul AWS Security Blog.

Ai fini della protezione dei dati, ti consigliamo di proteggere le credenziali degli AWS account e di configurare singoli account con AWS Identity and Access Management (IAM). In questo modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere il proprio lavoro. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È consigliabile TLS 1.2 o versioni successive.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno AWS dei servizi.
- Utilizza i servizi di sicurezza gestiti avanzati, ad esempio Amazon Macie, che aiutano a individuare e proteggere i dati personali archiviati in Amazon S3.
- Utilizza le opzioni di crittografia di Amazon EMR su EKS per crittografare i dati a riposo e in transito.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Consigliamo di non inserire mai informazioni identificative sensibili, ad esempio i numeri di account dei clienti, in campi a formato libero come un campo Nome. Ciò include quando lavori con Amazon EMR su EKS o altri AWS servizi utilizzando la console, l'API o. AWS CLI AWS SDKs Gli eventuali dati immessi in Amazon EMR su EKS o altri servizi potrebbero essere prelevati per l'inserimento nei log di diagnostica. Quando fornisci un URL a un server esterno, non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta a tale server.

Crittografia a riposo

La crittografia dei dati consente di impedire agli utenti non autorizzati di leggere dati su un cluster e sui sistemi di archiviazione di dati associati. Sono inclusi i dati salvati su supporti persistenti, noti

come dati a riposo, e i dati che possono essere intercettati quando circolano in rete, noti come dati in transito.

La crittografia dei dati richiede chiavi e certificati. Puoi scegliere tra diverse opzioni, tra cui chiavi gestite da AWS Key Management Service, chiavi gestite da Amazon S3 e chiavi e certificati di fornitori personalizzati da te forniti. Quando lo utilizzi AWS KMS come fornitore di chiavi, vengono addebitati costi per l'archiviazione e l'uso delle chiavi di crittografia. Per ulteriori informazioni, consultare AWS KMS Prezzi.

Prima di specificare le opzioni di crittografia, scegli i sistemi di gestione di chiavi e certificati che intendi utilizzare. Quindi, crea le chiavi e i certificati per i provider personalizzati specificati come parte delle impostazioni di crittografia.

Crittografia dei dati a riposo per dati EMRFS in Amazon S3

La crittografia di Amazon S3 funziona con gli oggetti del file system EMR (EMRFS) letti da e scritti su Amazon S3. Puoi specificare la crittografia lato server (SSE) o la crittografia lato client (CSE) di Amazon S3 come modalità di crittografia predefinita quando abiliti la crittografia dei dati a riposo. Facoltativamente, è possibile specificare diversi metodi di crittografia per singoli bucket utilizzando override di crittografia per bucket. Indipendentemente dall'abilitazione o meno della crittografia di Amazon S3, il protocollo Transport Layer Security (TLS) esegue la crittografia degli oggetti EMRFS in transito tra i nodi cluster EMR e Amazon S3. Per ulteriori informazioni sulla crittografia Amazon S3, consulta Protezione dei dati tramite la crittografia nella Guida per gli sviluppatori di Amazon Simple Storage Service.

> **Note**
>
> Quando si utilizza AWS KMS, vengono addebitati costi per l'archiviazione e l'uso delle chiavi di crittografia. Per ulteriori informazioni, consultare AWS KMS Prezzi.

Crittografia lato server di Amazon S3

Quando configuri la crittografia lato server Amazon S3, Amazon S3 esegue la crittografia dei dati a livello di oggetto, tramite la scrittura dei dati su disco, e ne esegue la decrittografia al momento dell'accesso. Per ulteriori informazioni sulla SEE, consulta Protezione dei dati con la crittografia lato server nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Puoi scegliere tra due diversi sistemi di gestione delle chiavi quando specifichi la SSE in Amazon EMR su EKS:

- SSE-S3: Amazon S3 gestisce le chiavi per te.
- SSE-KMS - Si utilizza una AWS KMS key configurazione con politiche adatte per Amazon EMR su EKS.

La SSE con chiavi fornite dal cliente (SSE-C) non è disponibile per l'uso con Amazon EMR su EKS.

Crittografia lato client Amazon S3

Con la crittografia lato client di Amazon S3, la crittografia e la decrittografia Amazon S3 avvengono nel client EMRFS nel tuo cluster. Gli oggetti vengono crittografati prima di essere caricati su Amazon S3 e decrittati dopo il loro download. Il provider specificato fornisce la chiave di crittografia utilizzata dal client. Il client può utilizzare le chiavi fornite da AWS KMS (CSE-KMS) o una classe Java personalizzata che fornisce la chiave principale lato client (CSE-C). Le specifiche di crittografia sono leggermente diverse tra CSE-KMS e CSE-C, a seconda del provider specificato e dei metadati dell'oggetto da decrittare o crittografare. Per ulteriori informazioni su queste differenze, consulta Protezione dei dati tramite la crittografia lato client nella Guida per gli sviluppatori di Amazon Simple Storage Service.

> **Note**
>
> Amazon S3 CSE garantisce solo che i dati EMRFS scambiati con Amazon S3 siano crittografati; non tutti i dati sui volumi delle istanze del cluster sono crittografati. Inoltre, poiché Hue non utilizza EMRFS, gli oggetti che il browser di file Hue S3 scrive su Amazon S3 non vengono crittografati.

Crittografia disco locale

Apache Spark supporta la crittografia dei dati temporanei scritti su dischi locali. Questo copre i file shuffle, gli spill shuffle e i blocchi di dati memorizzati sul disco per le variabili di caching e broadcast. Non copre la crittografia dei dati di output generati da applicazioni con caratteristiche come o. APIs `saveAsHadoopFile` `saveAsTable` Inoltre, potrebbe non coprire i file temporanei creati esplicitamente dall'utente. Per ulteriori informazioni, consulta Crittografia archiviazione locale nella documentazione di Spark. Spark non supporta i dati crittografati sul disco locale, ad esempio i dati intermedi scritti su un disco locale da un processo executor quando i dati non entrano nella memoria. I dati persistenti su disco vengono esaminati in base al runtime del processo e la chiave utilizzata per crittografare i dati viene generata dinamicamente da Spark per ogni esecuzione di processo. Una volta terminato il processo Spark, nessun altro processo può decrittografare i dati.

Per il pod driver ed executor, è possibile crittografare i dati a riposo che vengono mantenuti al volume montato. Esistono tre diverse opzioni di storage AWS nativo che puoi utilizzare con Kubernetes: EBS, FSx EFS e for Lustre. Tutti e tre offrono la crittografia di dati a riposo utilizzando una chiave gestita dal servizio o una AWS KMS key. Per ulteriori informazioni, consulta la Guida alle best practice EKS. Con questo approccio, tutti i dati persistenti sul volume montato vengono crittografati.

Gestione delle chiavi

È possibile configurare KMS per ruotare in automatico le chiavi KMS. Questa impostazione effettua una rotazione delle chiavi una volta all'anno e allo stesso tempo salva le vecchie chiavi a tempo indeterminato, in modo che i dati possano ancora essere decrittati. Per ulteriori informazioni, consulta Rotazione. AWS KMS keys

Crittografia dei dati in transito

Diversi meccanismi di crittografia sono abilitati con la crittografia in transito. Si tratta di funzionalità open source specifiche dell'applicazione che possono variare a seconda della versione Amazon EMR su EKS. Con Amazon EMR su EKS, è possibile abilitare le seguenti funzionalità di crittografia specifiche dell'applicazione:

- Spark
  - Le comunicazioni RPC interne tra componenti Spark, ad esempio il servizio di trasferimento dei blocchi e il servizio shuffle esterno, sono crittografate utilizzando la cifratura AES-256 in Amazon EMR versione 5.9.0 e versioni successive. Nelle versioni precedenti, la comunicazione RPC interna viene crittografata utilizzando SASL con DIGEST- MD5 come codice.
  - Le comunicazioni HTTP con interfacce utente come Spark History Server e file server HTTPS sono crittografate mediante la configurazione SSL di Spark. Per ulteriori informazioni, consulta Configurazione SSL nella documentazione di Spark.

  Per ulteriori informazioni, consulta le impostazioni di sicurezza Spark.
- Dovresti consentire solo connessioni crittografate su HTTPS (TLS) utilizzando le policy IAM di aws: SecureTransport condition on Amazon S3 bucket.
- I risultati delle query inviati ai client JDBC o ODBC sono crittografati con TLS.

Identity and Access Management

AWS Identity and Access Management (IAM) è un servizio Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (effettuare l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare risorse Amazon EMR su EKS. IAM è un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all’amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi relativi all'identità e all'accesso di Amazon EMR su EKS)
- Amministratore del servizio: determina l’accesso degli utenti e invia le richieste di autorizzazione (consulta Come funziona Amazon EMR su EKS con IAM)
- Amministratore IAM: scrivi policy per gestire l’accesso (consulta Esempi di policy basate su identità per Amazon EMR su EKS)

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/ Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali dell'utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come procedura ottimale, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare AWS IAM Identity Center. Per ulteriori informazioni, consulta Che cos'è il Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la

federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un ruolo utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso federato degli utenti, le autorizzazioni utente IAM temporanee, l'accesso tra account, l'accesso tra servizi e le applicazioni in esecuzione su Amazon. EC2 Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e collegandole a identità o risorse. AWS Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come

creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere policy in linea (con embedding direttamente in una singola identità) o policy gestite (policy autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in. AWS Organizations Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta Politiche di controllo delle risorse (RCPs) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come funziona Amazon EMR su EKS con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon EMR su EKS, è necessario comprendere quali funzionalità IAM sono disponibili per l'uso con Amazon EMR su EKS.

Funzionalità IAM utilizzabili con Amazon EMR su EKS

Funzionalità IAM	Supporto Amazon EMR su EKS
Policy basate sull'identità	Sì
Policy basate su risorse	No
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione delle policy	Sì
ACLs	No
ABAC (tag nelle policy)	Sì
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	No
Ruoli collegati al servizio	Sì

Per avere una visione di alto livello di come Amazon EMR su EKS e AWS altri servizi funzionano con la maggior parte delle funzionalità IAM, AWS consulta i servizi che funzionano con IAM nella IAM User Guide.

Policy basate su identità per Amazon EMR su EKS

Supporta le policy basate sull'identità: sì

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente di IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Esempi di policy basate su identità per Amazon EMR su EKS

Per visualizzare esempi di policy basate su identità Amazon EMR su EKS, consulta Esempi di policy basate su identità per Amazon EMR su EKS.

Policy basate su risorse all'interno di Amazon EMR su EKS

Supporta le policy basate su risorse: no

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Azioni di policy per Amazon EMR su EKS

Supporta le operazioni di policy: si

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di azioni Amazon EMR su EKS, consulta Operazioni, risorse e chiavi di condizione per Amazon EMR su EKS in Guida di riferimento all'autorizzazione del servizio.

Le azioni di policy in Amazon EMR su EKS utilizzano il seguente prefisso prima dell'operazione:

```
emr-containers
```

Per specificare più azioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "emr-containers:action1",
      "emr-containers:action2"
         ]
```

Per visualizzare esempi di policy basate su identità Amazon EMR su EKS, consulta Esempi di policy basate su identità per Amazon EMR su EKS.

Risorse di policy per Amazon EMR su EKS

Supporta le risorse relative alle policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse Amazon EMR su EKS e relativi ARNs, consulta Resources defined by Amazon EMR on EKS nel Service Authorization Reference. Per informazioni sulle operazioni per cui è possibile specificare l'ARN di ciascuna risorsa, consulta Operazioni, risorse e chiavi di condizione per Amazon EMR su EKS.

Per visualizzare esempi di policy basate su identità Amazon EMR su EKS, consulta Esempi di policy basate su identità per Amazon EMR su EKS.

Chiavi di condizione delle policy per Amazon EMR su EKS

Supporta le chiavi di condizione delle policy specifiche del servizio: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Per visualizzare un elenco di chiavi di condizione Amazon EMR su EKS e scoprire quali operazioni e risorse è possibile utilizzare con una chiave di condizione, consulta Operazioni, risorse e chiavi di condizione per Amazon EMR su EKS in Guida di riferimento all'autorizzazione del servizio.

Per visualizzare esempi di policy basate su identità Amazon EMR su EKS, consulta Esempi di policy basate su identità per Amazon EMR su EKS.

Liste di controllo degli accessi (ACLs) in Amazon EMR su EKS

Supporti ACLs: no

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Controllo degli accessi basato su attributi (ABAC) con Amazon EMR su EKS

Supporta ABAC (tag nelle policy)	Sì

Il controllo degli accessi basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base ad attributi chiamati tag. È possibile allegare tag a entità e AWS risorse IAM, quindi progettare politiche ABAC per consentire operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consulta Definizione delle autorizzazioni con autorizzazione ABAC nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con Amazon EMR su EKS

Supporta le credenziali temporanee: sì

Le credenziali temporanee forniscono l'accesso a breve termine alle AWS risorse e vengono create automaticamente quando si utilizza la federazione o si cambia ruolo. AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza temporanee in IAM e Servizi AWS compatibili con IAM nella Guida per l'utente IAM.

Autorizzazioni delle entità principali tra servizi per Amazon EMR su EKS

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale che chiama e, in combinazione con la richiesta Servizio AWS, Servizio AWS per effettuare richieste ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.

Ruoli di servizio per Amazon EMR su EKS

Supporta i ruoli di servizio	No

Ruoli collegati ai servizi per Amazon EMR su EKS

Supporta i ruoli collegati ai servizi	Sì

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Utilizzo di ruoli collegati ai servizi per Amazon EMR su EKS

Amazon EMR su EKS utilizza ruoli collegati ai servizi AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EMR su EKS. I ruoli collegati ai servizi sono predefiniti da Amazon EMR su EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri servizi per tuo conto. AWS

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EMR su EKS, perché ti permette di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EMR su EKS definisce le autorizzazioni del ruolo collegato ai servizi e, salvo diversamente definito, solo Amazon EMR su EKS può assumere il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato al servizio solo dopo avere eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EMR su EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consulta la sezione Servizi AWS che funzionano con IAM e cerca i servizi che riportano Sì nella colonna Ruolo associato ai servizi. Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon EMR su EKS

Amazon EMR su EKS utilizza il ruolo collegato al servizio denominato. AWSServiceRoleForAmazonEMRContainers

Ai fini dell'assunzione del ruolo, il ruolo collegato al servizio `AWSServiceRoleForAmazonEMRContainers` considera attendibili i seguenti servizi:

- `emr-containers.amazonaws.com`

La policy delle autorizzazioni di ruolo `AmazonEMRContainersServiceRolePolicy` consente ad Amazon EMR su EKS di completare una serie di operazioni sulle risorse specificate, come dimostra la seguente istruzione di policy.

> **Note**
>
> Il contenuto delle policy gestite cambia, pertanto la politica mostrata qui potrebbe essere. out-of-date Visualizza la maggior parte della documentazione up-to-date sulle policy di Amazon EMRContainers ServiceRolePolicy nella AWS Managed Policy Reference Guide.

JSON

```
{
  "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",
        "eks:ListPodIdentityAssociations",
        "eks:DescribePodIdentityAssociation"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEKSDescribecluster"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "acm:ImportCertificate",
        "acm:AddTagsToCertificate"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/emr-container:endpoint:managed-certificate": "true"
        }
      },
      "Sid": "AllowACMImportcertificate"
    },
    {
      "Effect": "Allow",
      "Action": [
        "acm:DeleteCertificate"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/emr-container:endpoint:managed-certificate": "true"
        }
      },
      "Sid": "AllowACMDeletecertificate"
    }
  ]
}
```

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon EMR su EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un cluster virtuale, Amazon EMR su EKS crea il ruolo collegato ai servizi per te.

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando crei un cluster virtuale, Amazon EMR su EKS crea di nuovo il ruolo collegato ai servizi per te.

È possibile utilizzare la console IAM anche per creare un ruolo collegato ai servizi con il caso d'uso Amazon EMR su EKS. Nella AWS CLI o nell' AWS API, crea un ruolo collegato al servizio con il nome del `emr-containers.amazonaws.com` servizio. Per ulteriori informazioni, consulta Creazione di un ruolo collegato ai servizi nella Guida per l'utente IAM. Se elimini il ruolo collegato ai servizi, è possibile utilizzare lo stesso processo per crearlo nuovamente.

Modifica di un ruolo collegato ai servizi per Amazon EMR su EKS

Amazon EMR su EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEMRContainers`. Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta la sezione Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EMR su EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato al servizio prima di poterlo eliminare manualmente.

> **Note**
>
> Se il servizio Amazon EMR su EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non vada a buon fine. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Eliminazione delle risorse Amazon EMR su EKS utilizzate da **`AWSServiceRoleForAmazonEMRContainers`**

1. Apri la console Amazon EMR.
2. Scegli un cluster virtuale.
3. Nella pagina `Virtual Cluster`, scegli Delete (Elimina).
4. Ripeti questa procedura per tutti gli altri cluster virtuali nell'account.

Per eliminare manualmente il ruolo collegato ai servizi mediante IAM

Utilizza la console IAM AWS CLI, l'o l' AWS API per eliminare il ruolo collegato al `AWSServiceRoleForAmazonEMRContainers` servizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato al servizio nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EMR su EKS

Amazon EMR su EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta Endpoint di servizio e Service Quotas Amazon EMR su EKS.

Policy gestite per Amazon EMR su EKS

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Amazon EMR su EKS dal 1° marzo 2021.

Modifica	Descrizione	Data
`AmazonEMRContainersServiceRolePolicy` - Aggiunte autorizzazioni di lettura per elencare le associazioni di identità dei pod EKS in un cluster e un'altra autorizzazione di lettura per restituire informazioni descrittive sulle associazioni di identità dei pod in un cluster. Per ulteriori informazioni, consulta Amazon EMRContainers ServiceRolePolicy.	Le seguenti autorizzazioni vengono aggiunte alla politica:`eks:ListPodIdentityAssociations` ,`eks:DescribePodIdentityAssociation` .	3 febbraio 2023
`AmazonEMRContainersServiceRolePolicy` : aggiunte autorizzazioni	Sono state aggiunte le seguenti autorizzazioni alla policy: `eks:ListNodeGroups` , `eks:Descr`	13 marzo 2023

Modifica	Descrizione	Data
per descrivere ed elencare i gruppi di nodi Amazon EKS, descrivere i gruppi di destinazione dei sistemi di bilanciamento del carico e descrivere lo stato di destinazione del sistema di bilanciamento del carico.	`ibeNodeGroup` , `elasticloadbalancing:DescribeTargetGroups` , `elasticloadbalancing:DescribeTargetHealth` .	
`AmazonEMRContainersServiceRolePolicy` - Sono state aggiunte le autorizzazioni per importare ed eliminare i certificati in. AWS Certificate Manager	Le seguenti autorizzazioni vengono aggiunte alla policy: `acm:ImportCertificate` , `acm:AddTagsToCertificate` e `acm:DeleteCertificate` .	3 dicembre 2021
Amazon EMR su EKS ha iniziato a monitorare le modifiche	Amazon EMR su EKS ha iniziato a tracciare le modifiche per le sue politiche AWS gestite.	1° marzo 2021

Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS

Se intendi utilizzare il comando `StartJobRun` per inviare un processo eseguito in un cluster EKS, dovrai prima integrare un ruolo di esecuzione di processo da utilizzare con un cluster virtuale. Per ulteriori informazioni, consulta Creazione di un ruolo di esecuzione di processo in Configurazione di Amazon EMR su EKS. È inoltre possibile seguire le istruzioni riportate nella sezione di Amazon EMR su EKS Workshop Creazione di un ruolo IAM per l'esecuzione del processo.

Le seguenti autorizzazioni devono essere incluse nella policy di affidabilità per il ruolo di esecuzione di processo.

JSON

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

```
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRoleWithWebIdentity"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:userid": "system:serviceaccount:NAMESPACE:emr-containers-sa-*-
*-AWS_ACCOUNT_ID-BASE36_ENCODED_ROLE_NAME"
        }
      },
      "Sid": "AllowSTSAssumerolewithwebidentity"
    }
  ]
}
```

La policy di affidabilità nell'esempio precedente concede le autorizzazioni solo a un account di servizio Kubernetes gestito da Amazon EMR con un nome che corrisponde al pattern `emr-containers-sa-*-*-`*`AWS_ACCOUNT_ID`*`-`*`BASE36_ENCODED_ROLE_NAME`. Gli account di servizio con tale pattern verranno creati in automatico all'invio del processo e assegnati all'ambito dello spazio dei nomi in cui invii il processo. La policy di affidabilità consente a questi account di servizio di assumere il ruolo di esecuzione e di ottenerne le credenziali temporanee. Agli account di servizi di un cluster Amazon EKS diverso o di uno spazio dei nomi diverso all'interno del medesimo cluster EKS è impedito di assumere il ruolo di esecuzione.

Per aggiornare in automatico la policy di affidabilità nel formato indicato sopra, puoi eseguire il comando seguente.

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

Controllo dell'accesso al ruolo di esecuzione

Un amministratore del cluster Amazon EKS può creare un cluster virtuale Amazon EMR su EKS multi-tenant a cui un amministratore IAM può aggiungere più ruoli di esecuzione. Poiché i tenant non affidabili possono utilizzare questi ruoli di esecuzione per inviare processi che eseguono un codice arbitrario, è consigliabile limitare tali tenant in modo che non possano eseguire un codice che ottiene

le autorizzazioni assegnate a uno o più dei suddetti ruoli di esecuzione. Per limitare la policy IAM allegata a un'identità IAM, l'amministratore IAM può utilizzare la chiave di condizione del nome della risorsa Amazon (ARN) `emr-containers:ExecutionRoleArn`. Questa condizione accetta un elenco di ruoli di esecuzione ARNs che dispongono delle autorizzazioni per il cluster virtuale, come dimostra la seguente politica di autorizzazione.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:StartJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:*:/virtualclusters/VIRTUAL_CLUSTER_ID"
      ],
      "Condition": {
        "ArnEquals": {
          "emr-containers:ExecutionRoleArn": [
            "arn:aws:iam::*:role/execution_role_name_1",
            "arn:aws:iam::*:role/execution_role_name_2"
          ]
        }
      },
      "Sid": "AllowEMRCONTAINERSStartjobrun"
    }
  ]
}
```

Se desideri autorizzare tutti i ruoli di esecuzione che iniziano con un prefisso particolare, ad esempio `MyRole`, puoi sostituire l'operatore di condizione `ArnEquals` con l'operatore `ArnLike` e il valore `execution_role_arn` nella condizione con il carattere jolly `*`. Ad esempio, `arn:aws:iam::AWS_ACCOUNT_ID:role/MyRole*`. Sono supportate anche tutte le altre chiavi di condizione ARN.

> **Note**
>
> Con Amazon EMR su EKS, non puoi concedere autorizzazioni a ruoli di esecuzione basati su tag o attributi. Amazon EMR su EKS non supporta il controllo degli accessi basato su tag (TBAC) né il controllo degli accessi basato su attributi (ABAC) per i ruoli di esecuzione.

Esempi di policy basate su identità per Amazon EMR su EKS

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse Amazon EMR su EKS. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM (console) nella Guida per l'utente di IAM.

Per dettagli sulle azioni e sui tipi di risorse definiti da Amazon EMR su EKS, incluso il formato di ARNs per ogni tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per Amazon EMR su EKS nel Service Authorization Reference.

Argomenti

Best practice per le policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon EMR su EKS nell'account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le politiche AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.

- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Uso della console di Amazon EMR su EKS

Per accedere alla console di Amazon EMR su EKS, è necessario disporre di un set di autorizzazioni minimo. Queste autorizzazioni devono consentire di elencare e visualizzare i dettagli relativi alle risorse di Amazon EMR su EKS nell' Account AWS. Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso o l' AWS CLI API. AWS Al contrario, concedi l'accesso solo alle operazioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli possano continuare a utilizzare Amazon EMR sulla console EKS, collega anche Amazon EMR su EKS `ConsoleAccess` o la policy `ReadOnly` AWS gestita alle entità. Per ulteriori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente IAM.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa policy include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o in modo programmatico. AWS CLI AWS

```
{
    "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": "*"
        }
    ]
}
```

Policy per il controllo degli accessi basato su tag

È possibile utilizzare le condizioni nella policy basata su identità per controllare l'accesso ai cluster virtuali e alle esecuzioni di processo basati su tag. Per ulteriori informazioni sull'assegnazione di tag, consulta Assegnazione di tag alle risorse Amazon EMR su EKS.

I seguenti esempi illustrano differenti scenari e i modi per utilizzare gli operatori di condizione con chiavi di condizione Amazon EMR su EKS. Queste dichiarazioni di policy IAM sono concepite unicamente per scopi dimostrativi e non devono essere utilizzate negli ambienti di produzione. Esistono vari modi di combinare dichiarazioni di policy per concedere o negare autorizzazioni in base alle tue esigenze. Per ulteriori informazioni sulla pianificazione e sul test di policy IAM, consulta la Guida per l'utente IAM.

> **Important**
>
> Negare esplicitamente l'autorizzazione per operazioni di assegnazione di tag è una possibilità da tenere in debita considerazione. Ciò impedisce agli utenti di concedersi personalmente autorizzazioni tramite tag di una risorsa che non avevi intenzione di accordare. Se le operazioni di assegnazione di tag per una risorsa non vengono negate, un utente può modificare i tag e aggirare l'intenzione delle policy basate su tag. Per un esempio di policy che nega le operazioni di tag, consulta Negazione dell'accesso per aggiungere ed eliminare tag.

Gli esempi seguenti mostrano le policy di autorizzazioni basate su identità utilizzate per controllare le operazioni che sono consentite con cluster virtuali Amazon EMR su EKS.

Autorizzazione di operazioni solo su risorse con specifici valori di tag

Nel seguente esempio di policy, l'operatore StringEquals condition tenta di abbinare dev al valore per il reparto tag. Se il tag reparto non è stato aggiunto al cluster virtuale, oppure non contiene il

valore dev, la policy non viene applicata e le operazioni non sono consentite da questa policy. Se nessun'altra istruzione di policy consente le operazioni, l'utente può utilizzare solo i cluster virtuali che hanno questo tag con tale valore.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeVirtualCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      },
      "Sid": "AllowEMRCONTAINERDescribevirtualcluster"
    }
  ]
}
```

Puoi anche specificare più valori di tag utilizzando un operatore di condizione. Ad esempio, per consentire le operazioni su cluster virtuali in cui il tag `department` contiene il valore `dev` o `test`, puoi sostituire il blocco di condizione dell'esempio precedente con quanto segue.

```
"Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": ["dev", "test"]
        }
      }
```

Richiesta dell'assegnazione di tag alla creazione di una risorsa

Nell'esempio seguente, il tag deve essere applicato quando si crea il cluster virtuale.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/department": "dev"
        }
      },
      "Sid": "AllowEMRCONTAINERSCreatevirtualcluster"
    }
  ]
}
```

L'istruzione di policy seguente consente a un utente di creare un cluster virtuale solo se il cluster ha un tag `department`, indipendentemente dal relativo valore.

JSON

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

```
        "Null": {
          "aws:RequestTag/department": "false"
        }
      },
      "Sid": "AllowEMRCONTAINERSCreatevirtualcluster"
    }
  ]
}
```

Negazione dell'accesso per aggiungere ed eliminare tag

L'effetto di questa policy è negare a un utente l'autorizzazione di aggiungere o rimuovere qualsiasi tag sui cluster virtuali contrassegnati con un tag `department` contenente il valore `dev`.

JSON

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

Risoluzione dei problemi relativi all'identità e all'accesso di Amazon EMR su EKS

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di Amazon EMR su EKS e IAM.

Argomenti

Non sono autorizzato a eseguire un'operazione in Amazon EMR su EKS

Se ti Console di gestione AWS dice che non sei autorizzato a eseguire un'azione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è la persona da cui si sono ricevuti il nome utente e la password.

Il seguente esempio di errore si verifica quando l'utente `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia, ma non dispone di autorizzazioni `emr-containers:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
containers:GetWidget on resource: my-example-widget
```

In questo caso, Mateo richiede al suo amministratore di aggiornare le policy per poter accedere alla risorsa *`my-example-widget`* utilizzando l'operazione `emr-containers:`*`GetWidget`*.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, è necessario aggiornare le policy affinché tu possa trasmettere un ruolo ad Amazon EMR su EKS.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Amazon EMR su EKS. Tuttavia, l'azione richiede

che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse Amazon EMR su EKS

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per consentire alle persone di accedere alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon EMR su EKS supporta queste funzionalità, consulta Come funziona Amazon EMR su EKS con IAM.
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

Utilizzo di Amazon EMR su EKS con AWS Lake Formation per un controllo granulare degli accessi

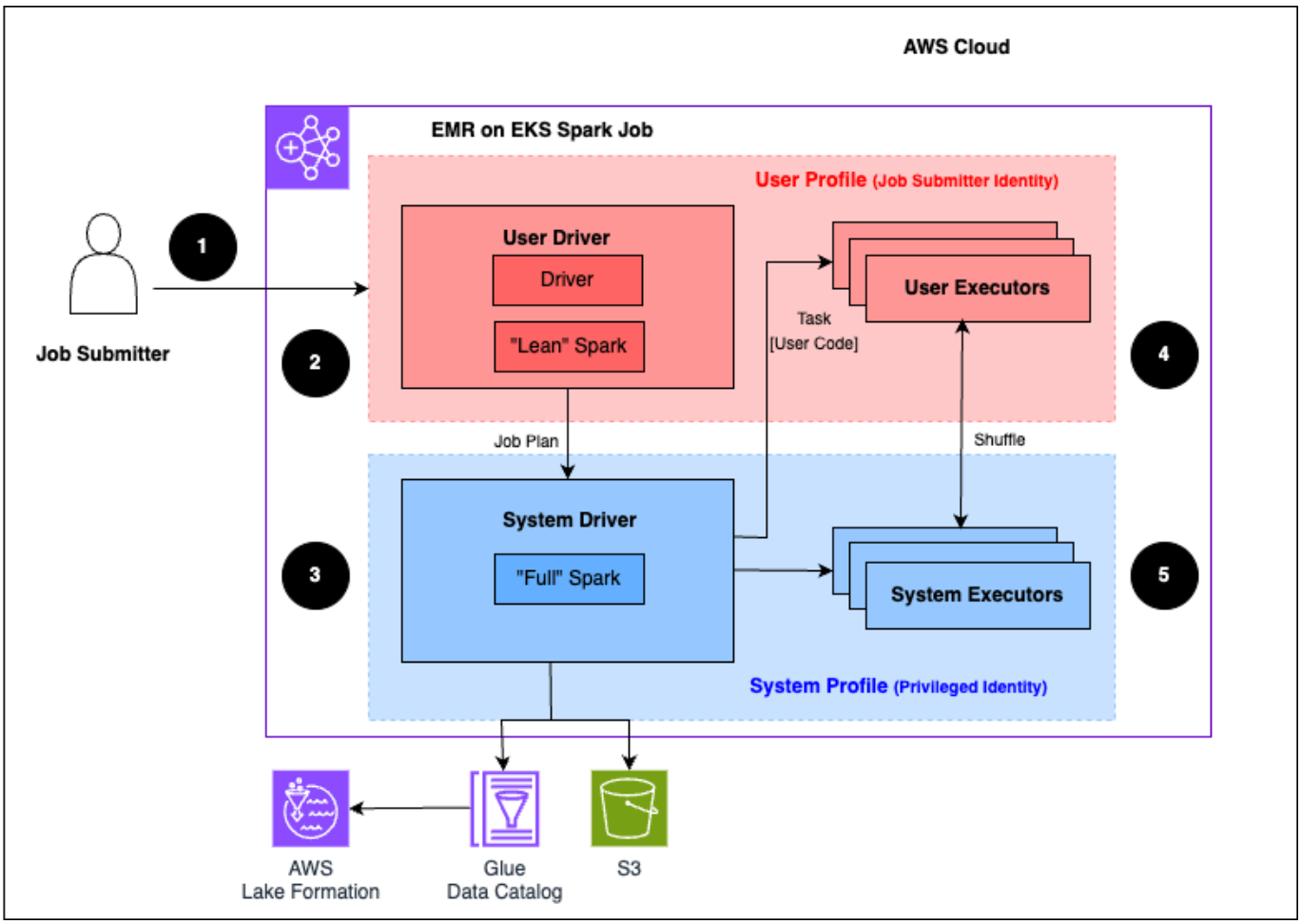
Con Amazon EMR versione 7.7 e successive, puoi sfruttare Lake AWS Formation per applicare controlli AWS di accesso dettagliati alle tabelle di Glue Data Catalog supportate da bucket Amazon S3. Questa funzionalità consente di configurare i controlli di accesso a livello di tabella, riga, colonna e cella per le query di lettura all'interno di Amazon EMR su EKS Spark Jobs.

Argomenti

Come funziona Amazon EMR su EKS con Lake Formation AWS

L'utilizzo di Amazon EMR su EKS con Lake Formation ti consente di applicare un livello di autorizzazioni su ogni lavoro Spark per applicare il controllo delle autorizzazioni di Lake Formation quando Amazon EMR su EKS esegue lavori. Amazon EMR su EKS utilizza i profili di risorse Spark per creare due profili per eseguire i lavori in modo efficace. Il profilo utente esegue il codice fornito dall'utente, mentre il profilo di sistema applica le politiche di Lake Formation. Ogni Job abilitato per Lake Formation utilizza due driver Spark, uno per il profilo utente e l'altro per il profilo System. Per ulteriori informazioni, vedi What is AWS Lake Formation.

Di seguito è riportata una panoramica di alto livello su come Amazon EMR su EKS ottiene l'accesso ai dati protetti dalle politiche di sicurezza di Lake Formation.



I seguenti passaggi descrivono questo processo:

1. Un utente invia uno Spark Job a un cluster virtuale Amazon EMR su EKS abilitato per AWS Lake Formation.
2. Il servizio Amazon EMR on EKS configura il driver utente ed esegue il processo nel profilo utente. Lo User Driver esegue una versione snella di Spark che non è in grado di avviare attività, richiedere esecutori, accedere ad Amazon S3 o al Glue Data Catalog. Costruisce solo un Job plan.
3. Il servizio Amazon EMR on EKS configura un secondo driver chiamato System Driver e lo esegue nel profilo di sistema (con un'identità privilegiata). Amazon EKS configura un canale TLS crittografato tra i due driver per la comunicazione. Lo User Driver utilizza il canale per inviare i piani di lavoro al driver di sistema. Il driver di sistema non esegue il codice inviato dall'utente. Funziona con Spark completo e comunica con Amazon S3 e il Data Catalog per l'accesso ai dati. Richiede esecutori e compila il Job Plan in una sequenza di fasi di esecuzione.

4. Amazon EMR sul servizio EKS esegue quindi le fasi relative agli esecutori. Il codice utente in qualsiasi fase viene eseguito esclusivamente sugli esecutori dei profili utente.
5. Le fasi che leggono i dati dalle tabelle del Data Catalog protette da Lake Formation o quelle che applicano filtri di sicurezza sono delegate agli esecutori di sistema.

Abilita Lake Formation con Amazon EMR su EKS

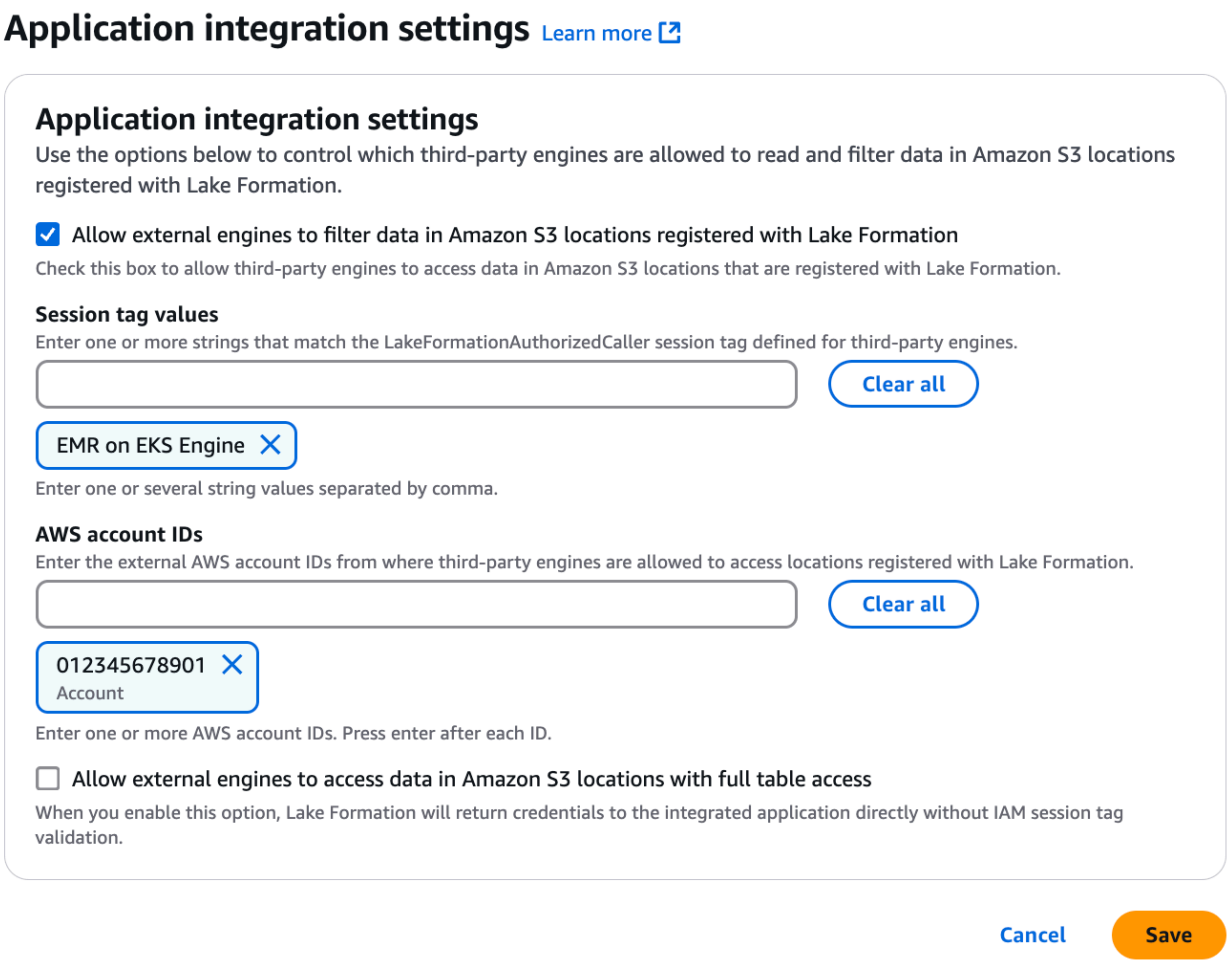
Con Amazon EMR versione 7.7 e successive, puoi sfruttare Lake AWS Formation per applicare controlli di accesso granulari alle tabelle di Data Catalog supportate da Amazon S3. Questa funzionalità consente di configurare i controlli di accesso a livello di tabella, riga, colonna e cella per le query di lettura all'interno di Amazon EMR su EKS Spark Jobs.

Questa sezione spiega come creare una configurazione di sicurezza e configurare Lake Formation per l'utilizzo con Amazon EMR. Descrive inoltre come creare un cluster virtuale con la configurazione di sicurezza creata per Lake Formation. Queste sezioni devono essere completate in sequenza.

Passaggio 1: configurare le autorizzazioni a livello di colonna, riga o cella basate su Lake Formation

Innanzitutto, per applicare le autorizzazioni a livello di riga e colonna con Lake Formation, l'amministratore del data lake per Lake Formation deve impostare il tag di LakeFormationAuthorizedCallersessione. Lake Formation utilizza questo tag di sessione per autorizzare i chiamanti e fornire l'accesso al data lake.

Vai alla console di AWS Lake Formation e seleziona l'opzione Impostazioni di integrazione delle applicazioni dalla sezione Amministrazione nella barra laterale. Quindi, seleziona la casella Consenti ai motori esterni di filtrare i dati nelle località Amazon S3 registrate con Lake Formation. Aggiungi l'AWS account IDs in cui verranno eseguiti gli Spark Jobs e i valori del tag di sessione.



Nota che il tag di LakeFormationAuthorizedCallersessione passato qui viene passato in un SecurityConfigurationsecondo momento quando configuri i ruoli IAM, nella sezione 3.

Fase 2: Configurazione delle autorizzazioni EKS RBAC

In secondo luogo, si impostano le autorizzazioni per il controllo degli accessi basato sui ruoli.

Fornisci le autorizzazioni del cluster EKS al servizio Amazon EMR su EKS

Il servizio Amazon EMR su EKS deve disporre delle autorizzazioni EKS Cluster Role in modo da poter creare autorizzazioni cross-namespace per consentire al driver di sistema di inserire gli esecutori degli utenti nello spazio dei nomi utente.

Crea un ruolo del cluster

Questo esempio definisce le autorizzazioni per una raccolta di risorse.

```
vim emr-containers-cluster-role.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: emr-containers
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: ["clusterroles","clusterrolebindings","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"]
  - apiGroups: ["kyverno.io"]
    resources: ["clusterpolicies"]
    verbs: ["create", "delete"]
---
```

```
kubectl apply -f emr-containers-cluster-role.yaml
```

Crea associazioni di ruoli del cluster

```
vim emr-containers-cluster-role-binding.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: emr-containers
subjects:
- kind: User
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
---
```

```
kubectl apply -f emr-containers-cluster-role-binding.yaml
```

Fornisci l'accesso Namespace al servizio Amazon EMR on EKS

Crea due namespace Kubernetes, uno per driver utente ed esecutori e un altro per driver ed esecutori di sistema, e abilita l'accesso al servizio Amazon EMR su EKS per inviare lavori sia nei namespace utente che in quelli di sistema. Segui la guida esistente per fornire l'accesso a ogni spazio dei nomi, disponibile in Abilita l'accesso al cluster utilizzando. `aws-auth`

Fase 3: Configurazione dei ruoli IAM per i componenti del profilo utente e di sistema

In terzo luogo, si impostano i ruoli per componenti specifici. Uno Spark Job abilitato per Lake Formation ha due componenti, utente e sistema. Il driver User e gli executor vengono eseguiti nello spazio dei nomi utente e sono legati a ciò che viene passato nell' JobExecutionRole API. StartJobRun Il driver e gli executor di sistema vengono eseguiti nello spazio dei nomi System e sono legati al ruolo. QueryEngine

Configura il ruolo di Query Engine

Il QueryEngine ruolo è legato ai componenti dello spazio di sistema e avrebbe le autorizzazioni per assumere il tag JobExecutionRolewith LakeFormationAuthorizedCallerSession. Il ruolo IAM Permissions Policy of Query Engine è il seguente:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeJobRoleWithSessionTagAccessForSystemDriver",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Resource": [
        "arn:aws:iam::*:role/JobExecutionRole"
      ],
      "Condition": {
        "StringLike": {
          "aws:RequestTag/LakeFormationAuthorizedCaller": "EMR on EKS Engine"
        }
      }
    },
    {
      "Sid": "AssumeJobRoleWithSessionTagAccessForSystemExecutor",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/JobExecutionRole"
      ]
    },
    {
      "Sid": "CreateCertificateAccessForTLS",
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateCertificate"
      ],
```

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

Configura la politica di attendibilità del ruolo Query Engine per considerare attendibile lo spazio dei nomi del sistema Kubernetes.

```
aws emr-containers update-role-trust-policy \
    --cluster-name eks cluster \
    --namespace eks system namespace \
    --role-name query_engine_iam_role_name
```

Per ulteriori informazioni, consulta Aggiornamento della policy di attendibilità dei ruoli.

Configurazione del ruolo Job Execution

Le autorizzazioni di Lake Formation controllano l'accesso alle risorse di AWS Glue Data Catalog, alle sedi Amazon S3 e ai dati sottostanti in tali sedi. Le autorizzazioni IAM controllano l'accesso a Lake Formation and AWS Glue APIs e alle risorse. Sebbene tu possa avere l'autorizzazione Lake Formation per accedere a una tabella nel Data Catalog (SELECT), l'operazione fallisce se non disponi dell'autorizzazione IAM sulle operazioni `glue:Get*` API.

Politica sulle autorizzazioni IAM di JobExecutionRole: Il JobExecutionruolo dovrebbe avere le dichiarazioni politiche nella sua politica sulle autorizzazioni.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
```

```
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CreateCertificateAccessForTLS",
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateCertificate"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

IAM Trust Policy per: JobExecutionRole

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "TrustQueryEngineRoleForSystemDriver",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
```

```
      ],
      "Resource": [
        "arn:aws:iam::*:role/QueryExecutionRole"
      ],
      "Condition": {
        "StringLike": {
          "aws:RequestTag/LakeFormationAuthorizedCaller": "EMR on EKS Engine"
        }
      }
    },
    {
      "Sid": "TrustQueryEngineRoleForSystemExecutor",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/QueryEngineRole"
      ]
    }
  ]
}
```

Configura la Trust Policy of Job execution Role per considerare attendibile lo spazio dei nomi utente Kubernetes:

```
aws emr-containers update-role-trust-policy \
    --cluster-name eks cluster \
    --namespace eks User namespace \
    --role-name job_execution_role_name
```

Per ulteriori informazioni, consulta Aggiornare la politica di fiducia del ruolo di esecuzione del lavoro.

Fase 4: Configurazione della configurazione di sicurezza

Per eseguire un job abilitato per Lake Formation, è necessario creare una configurazione di sicurezza.

```
aws emr-containers create-security-configuration \
    --name 'security-configuration-name' \
    --security-configuration '{
```

```
        "authorizationConfiguration": {
            "lakeFormationConfiguration": {
                "authorizedSessionTagValue": "SessionTag configured in LakeFormation",
                "secureNamespaceInfo": {
                    "clusterId": "eks-cluster-name",
                    "namespace": "system-namespace-name"
                },
                "queryEngineRoleArn": "query-engine-IAM-role-ARN"
            }
        }
    }'
```

Assicurati che il tag di sessione passato nel campo authorizedSessionTagValue possa autorizzare Lake Formation. Imposta il valore su quello configurato in Lake Formation, inPassaggio 1: configurare le autorizzazioni a livello di colonna, riga o cella basate su Lake Formation.

Fase 5: Creare un cluster virtuale

Crea un cluster virtuale Amazon EMR su EKS con una configurazione di sicurezza.

```
aws emr-containers create-virtual-cluster \
--name my-lf-enabled-vc \
--container-provider '{
    "id": "eks-cluster",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "user-namespace"
        }
    }
}' \
--security-configuration-id SecurityConfiguraionId
```

Assicurati che venga passato l'SecurityConfigurationID del passaggio precedente, in modo che la configurazione di autorizzazione di Lake Formation venga applicata a tutti i lavori in esecuzione sul cluster virtuale. Per ulteriori informazioni, consulta Registrare il cluster Amazon EKS con Amazon EMR.

Fase 6: Inviare un Job in FGAC Enabled VirtualCluster

Il processo di invio dei lavori è lo stesso sia per i lavori non Lake Formation che per quelli di Lake Formation. Per ulteriori informazioni, consulta Inviare un lavoro eseguito con `StartJobRun`.

Il driver Spark, l'Executor e i registri degli eventi del driver di sistema sono archiviati nel bucket S3 di AWS Service Account per il debug. Consigliamo di configurare una chiave KMS gestita dal cliente nel Job Run per crittografare tutti i log archiviati nel bucket di servizi. AWS Per ulteriori informazioni sull'attivazione della crittografia dei log, consulta Encrypting Amazon EMR sui log EKS.

Considerazioni e limitazioni

Tieni presente le seguenti considerazioni e limitazioni quando utilizzi Lake Formation con Amazon EMR su EKS:

- Amazon EMR su EKS supporta il controllo granulare degli accessi tramite Lake Formation solo per i formati di tabella Apache Hive, Apache Iceberg, Apache Hudi e Delta. I formati Apache Hive includono Parquet, ORC e XSv.
- `DynamicResourceAllocation`è abilitato per impostazione predefinita e non è possibile disattivarlo `DynamicResourceAllocation` per i lavori di Lake Formation. Poiché il valore predefinito della `spark.dynamicAllocation.maxExecutors` configurazione DRA è infinito, configurate un valore appropriato in base al carico di lavoro.
- I job abilitati per Lake Formation non supportano l'utilizzo di EMR personalizzato su immagini EKS in System Driver e System Executors.
- Si può usare Lake Formation solo con i processi Spark.
- EMR su EKS with Lake Formation supporta solo una singola sessione Spark per tutta la durata di un job.
- EMR su EKS with Lake Formation supporta solo le query tabellari tra account condivise tramite link alle risorse.
- I seguenti elementi non sono supportati:
  - Resilient Distributed Dataset (RDD)
  - Streaming di Spark
  - Scrivere con le autorizzazioni concesse da Lake Formation
  - Controllo degli accessi per colonne annidate
- L'EMR su EKS blocca le funzionalità che potrebbero compromettere il completo isolamento dei driver di sistema, tra cui:
  - UDTs, Hive UDFs e qualsiasi funzione definita dall'utente che coinvolga classi personalizzate
  - Origini dati personalizzate
  - Fornitura di jar aggiuntivi per l'estensione Spark, il connettore o il comando metastore `ANALYZE TABLE`

- Per applicare i controlli di accesso, `EXPLAIN PLAN` e le operazioni DDL, come `DESCRIBE TABLE` non espongono informazioni riservate.
- Amazon EMR su EKS limita l'accesso ai log Spark dei driver di sistema sui job abilitati per Lake Formation. Poiché il driver di sistema viene eseguito con più accessi, gli eventi e i log generati da quest'ultimo possono includere informazioni riservate. Per impedire a utenti o codici non autorizzati di accedere a questi dati sensibili, EMR su EKS ha disabilitato l'accesso ai registri dei driver di sistema. Per la risoluzione dei problemi, contatta l'assistenza. AWS
- Se hai registrato una posizione in una tabella con Lake Formation, il percorso di accesso ai dati passa attraverso le credenziali archiviate di Lake Formation, indipendentemente dall'autorizzazione IAM per il ruolo di esecuzione del lavoro EMR on EKS. Se configuri erroneamente il ruolo registrato con la posizione della tabella, i lavori inviati che utilizzano il ruolo con l'autorizzazione S3 IAM per la posizione della tabella avranno esito negativo.
- La scrittura su una tabella Lake Formation utilizza l'autorizzazione IAM anziché le autorizzazioni concesse da Lake Formation. Se il tuo ruolo di esecuzione del lavoro dispone delle autorizzazioni S3 necessarie, puoi utilizzarlo per eseguire operazioni di scrittura.

Di seguito sono riportate considerazioni e limitazioni per l'utilizzo di Apache Iceberg:

- È possibile utilizzare Apache Iceberg solo con il catalogo delle sessioni e non con i cataloghi con nomi arbitrari.
- Le tabelle Iceberg registrate in Lake Formation supportano solo le tabelle di metadati`history`,`metadata_log_entries`,, `snapshots` `filesmanifests`, e. `refs` Amazon EMR nasconde le colonne che potrebbero contenere dati sensibili, ad esempio`partitions`, `path` e. `summaries` Questa limitazione non si applica alle tabelle Iceberg che non sono registrate in Lake Formation.
- Le tabelle non registrate in Lake Formation supportano tutte le stored procedure di Iceberg. Le procedure di `register_table` e `migrate` non sono supportate per nessuna tabella.
- Ti consigliamo di utilizzare Iceberg DataFrameWriter V2 anziché V1.

Per ulteriori informazioni, consulta Comprendere i concetti e la terminologia di Amazon EMR su EKS e Abilitare l'accesso ai cluster per Amazon EMR su EKS.

Dichiarazione di non responsabilità per gli amministratori di dati

> **Note**
>
> Quando concedi l'accesso alle risorse di Lake Formation a un ruolo IAM per EMR su EKS, devi assicurarti che l'amministratore o l'operatore del cluster EMR sia un amministratore affidabile. Ciò è particolarmente importante per le risorse di Lake Formation condivise tra più organizzazioni e AWS account.

Responsabilità degli amministratori EKS

- Il `System` namespace deve essere protetto. A nessun utente, risorsa o entità o strumento sarebbe consentito disporre di autorizzazioni RBAC Kubernetes sulle risorse Kubernetes nel namespace. `System`
- Nessun utente, risorsa o entità tranne il servizio EMR su EKS dovrebbe avere `CREATE` accesso a POD, CONFIG_MAP e SECRET nello spazio dei nomi. `User`
- `System`i driver e `System` gli esecutori contengono dati sensibili. Pertanto, gli eventi Spark, i registri dei driver Spark e i registri degli esecutori Spark presenti nel `System` namespace non devono essere inoltrati a sistemi di archiviazione dei log esterni.

Risoluzione dei problemi

Registrazione dei log

EMR su EKS utilizza i profili di risorse Spark per suddividere l'esecuzione del lavoro. Amazon EMR su EKS utilizza il profilo utente per eseguire il codice fornito, mentre il profilo di sistema applica le politiche di Lake Formation. Puoi accedere ai log dei contenitori eseguiti come profilo utente configurando la richiesta con. StartJobRun MonitoringConfiguration

Spark History Server

Lo Spark History Server contiene tutti gli eventi Spark generati dal profilo utente e gli eventi oscurati generati dal driver di sistema. Puoi vedere tutti i contenitori dei driver utente e di sistema nella scheda Executors. Tuttavia, i link di registro sono disponibili solo per il profilo utente.

Il processo ha avuto esito negativo con autorizzazioni Lake Formation insufficienti

Assicurati che il tuo ruolo di esecuzione del lavoro disponga delle autorizzazioni necessarie per l'esecuzione `SELECT` e sia presente `DESCRIBE` sulla tabella a cui stai accedendo.

Processo con esecuzione RDD non riuscita

EMR su EKS attualmente non supporta le operazioni di set di dati distribuiti resilienti (RDD) sui lavori abilitati a Lake Formation.

Impossibile accedere ai file di dati in Amazon S3

Assicurarsi di aver registrato la posizione del data lake in Lake Formation.

Eccezione di convalida della sicurezza

EMR su EKS ha rilevato un errore di convalida della sicurezza. Contatta l' AWS assistenza per ricevere assistenza.

Condivisione del catalogo dati e delle tabelle di AWS Glue tra account

È possibile condividere database e tabelle tra account e continuare a utilizzare Lake Formation. Per ulteriori informazioni, consulta Condivisione dei dati tra account in Lake Formation e Come posso condividere AWS Glue Data Catalog e tabelle tra account utilizzando AWS Lake Formation? .

Errore di inizializzazione generato da Iceberg Job che non imposta la regione AWS

Il messaggio è il seguente:

```
25/02/25 13:33:19 ERROR SparkFGACExceptionSanitizer: Client received error with id =
 b921f9e6-f655-491f-b8bd-b2842cdc20c7,
reason = IllegalArgumentException, message = Cannot initialize
LakeFormationAwsClientFactory, please set client.region to a valid aws region
```

Assicurati che la configurazione di Spark `spark.sql.catalog.`*`catalog_name`*`.client.region` sia impostata su una regione valida.

Iceberg Job - Lancio SparkUnsupportedOperationException

Il messaggio è il seguente:

```
25/02/25 13:53:15 ERROR SparkFGACExceptionSanitizer: Client received error with id =
 921fef42-0800-448b-bef5-d283d1278ce0,
reason = SparkUnsupportedOperationException, message = Either glue.id or glue.account-
id is set with non-default account.
Cross account access with fine-grained access control is only supported with AWS
 Resource Access Manager.
```

Assicurati che la configurazione Spark `spark.sql.catalog.catalog_name.glue.account-id` sia impostata su un ID account valido.

Iceberg Job fallisce con «403 Access Denied» durante l'operazione MERGE

Il messaggio è il seguente:

```
software.amazon.awssdk.services.s3.model.S3Exception: Access Denied (Service: S3,
 Status Code: 403,
...
 at
 software.amazon.awssdk.services.s3.DefaultS3Client.deleteObject(DefaultS3Client.java:3365)
 at org.apache.iceberg.aws.s3.S3FileIO.deleteFile(S3FileIO.java:162)
 at org.apache.iceberg.io.FileIO.deleteFile(FileIO.java:86)
 at
 org.apache.iceberg.io.RollingFileWriter.closeCurrentWriter(RollingFileWriter.java:129)
```

Disabilita le operazioni di eliminazione di S3 in Spark aggiungendo la seguente proprietà. `--conf spark.sql.catalog.s3-table-name.s3.delete-enabled=false`.

Registrazione di log e monitoraggio

Per rilevare incidenti, ricevere avvisi quando si verificano incidenti e rispondere a essi, utilizza queste opzioni con Amazon EMR su EKS:

- Monitora Amazon EMR su EKS con AWS CloudTrail - AWS CloudTrailfornisce un registro delle azioni intraprese da un utente, ruolo o AWS servizio in Amazon EMR su EKS. Consente di acquisire le chiamate dalla console di Amazon EMR e le chiamate di codice alle operazioni API di Amazon EMR su EKS come eventi. Questo consente di determinare la richiesta effettuata ad Amazon EMR su EKS, l'indirizzo IP da cui è stata eseguita la richiesta, l'autore della richiesta, il momento in cui è stata eseguita e altri dettagli. Per ulteriori informazioni, consulta Registrazione di Amazon EMR su chiamate API EKS utilizzando AWS CloudTrail.

- Use CloudWatch Events with Amazon EMR su EKS - CloudWatch Events offre un flusso quasi in tempo reale di eventi di sistema che descrivono i cambiamenti nelle AWS risorse. CloudWatch Events viene a conoscenza dei cambiamenti operativi man mano che si verificano, risponde ad essi e intraprende le azioni correttive necessarie, inviando messaggi per rispondere all'ambiente, attivando funzioni, apportando modifiche e acquisendo informazioni sullo stato. Per utilizzare CloudWatch Events with Amazon EMR su EKS, crea una regola che si attiva su una chiamata API Amazon EMR su EKS tramite. CloudTrail Per ulteriori informazioni, consulta Monitora i lavori con Amazon CloudWatch Events.

Crittografia di Amazon EMR sui log EKS con storage gestito

Le sezioni che seguono mostrano come configurare la crittografia per i log.

Enable Encryption (Abilita crittografia)

Per crittografare i log nello storage gestito con la tua chiave KMS, usa la seguente configurazione quando invii un job run.

```
"monitoringConfiguration": {
            "managedLogs": {
                "allowAWSToRetainLogs":"ENABLED",
                "encryptionKeyArn":"KMS key arn"
            },
            "persistentAppUI": "ENABLED"
        }
```

La `allowAWSToRetainLogs` configurazione consente di AWS conservare i registri dello spazio dei nomi di sistema durante l'esecuzione di un lavoro utilizzando Native FGAC. La `persistentAppUI` configurazione consente di AWS salvare i registri degli eventi utilizzati per generare l'interfaccia utente Spark. `encryptionKeyArn`viene utilizzato per specificare l'ARN della chiave KMS che si desidera utilizzare per crittografare i registri archiviati da. AWS

Autorizzazioni richieste per la crittografia dei log

L'utente che invia il lavoro o visualizza l'interfaccia utente di Spark deve disporre dell'autorizzazione alle azioni `kms:DescribeKey` e alla `kms:GenerateDataKey` chiave di `kms:Decrypt` crittografia. Queste autorizzazioni vengono utilizzate per verificare la validità della chiave e verificare che l'utente disponga delle autorizzazioni necessarie per leggere e scrivere registri crittografati con la chiave

KMS. Se l'utente che invia il lavoro non dispone delle autorizzazioni chiave necessarie, Amazon EMR su EKS rifiuta l'invio dell'esecuzione del lavoro.

Esempio di policy IAM per Role Used to Call StartJobRun

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "emr-containers:StartJobRun"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "AllowEMRCONTAINERSStartjobrun"
    },
    {
      "Action": [
        "kms:DescribeKey",
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:*:*:key/key-id"
      ],
      "Effect": "Allow",
      "Sid": "AllowKMSDescribekey"
    }
  ]
}
```

È inoltre necessario configurare la chiave KMS per consentire a `persistentappui.elasticmapreduce.amazonaws.com` and `elasticmapreduce.amazonaws.com` Service Principal di effettuare la e. `kms:GenerateDataKey` `kms:Decrypt` Ciò consente a EMR di leggere e scrivere registri crittografati con la chiave KMS per lo storage gestito.

Esempio di politica chiave KMS

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:DescribeKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "kms:viaService": "emr-containers.*.amazonaws.com"
        }
      },
      "Sid": "AllowKMSDescribekey"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "kms:viaService": "emr-containers.*.amazonaws.com",
          "kms:EncryptionContext:aws:emr-containers:virtualClusterId": "virtual
 cluster id"
        }
      },
      "Sid": "AllowKMSDecryptGenerate"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:emr-containers:virtualClusterId": "virtual
 cluster id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:emr-containers:*:*:/
virtualclusters/virtual_cluster_id"
        }
      },
      "Sid": "AllowKMSDecryptService"
    }
  ]
}
```

Come best practice di sicurezza, ti consigliamo di aggiungere le `aws:SourceArn` condizioni `kms:viaServicekms:EncryptionContext`, e. Queste condizioni aiutano a garantire che la chiave venga utilizzata solo da Amazon EMR su EKS e utilizzata solo per i log generati dai lavori in esecuzione in uno specifico cluster virtuale.

Registrazione di Amazon EMR su chiamate API EKS utilizzando AWS CloudTrail

Amazon EMR su EKS è integrato con AWS CloudTrail un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o AWS servizio in Amazon EMR su EKS. CloudTrail acquisisce tutte le chiamate API per Amazon EMR su EKS come eventi. Le chiamate acquisite includono le chiamate dalla console Amazon EMR su EKS e le chiamate di codice alle operazioni API di Amazon EMR su EKS. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon EMR su EKS. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console in Cronologia eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare la richiesta che è stata effettuata ad Amazon EMR su EKS, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta la Guida per l'AWS CloudTrail utente.

Amazon EMR sulle informazioni EKS in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in Amazon EMR su EKS, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare eventi recenti nel tuo AWS account. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Amazon EMR su EKS, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando crei un percorso nella console, il percorso si applica a tutte le AWS regioni. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un percorso
- CloudTrail servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte le azioni di Amazon EMR su EKS vengono registrate CloudTrail e documentate nella documentazione dell'API Amazon EMR su EKS. Ad esempio, le chiamate a `StartJobRun` e le `CreateVirtualCluster` `ListJobRuns` azioni generano voci nei file di registro. CloudTrail

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente root o AWS Identity and Access Management (IAM).
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, vedete l'elemento Identity CloudTrail dell'utente.

Informazioni sulle voci dei file di log Amazon EMR su EKS

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'`ListJobRuns`azione.


```
{
  "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"
}
```

Utilizzo di Amazon S3 Access Grants con Amazon EMR su EKS

Panoramica di S3 Access Grants per Amazon EMR su EKS

Con Amazon EMR rilascio 6.15.0 e successivi, Amazon S3 Access Grants fornisce una soluzione di controllo degli accessi scalabile che puoi utilizzare per aumentare l'accesso ai tuoi dati Amazon S3 da Amazon EMR su EKS. Se hai una configurazione di autorizzazioni complessa o di grandi dimensioni per i dati S3, puoi utilizzare Access Grants per dimensionare le autorizzazioni relative ai dati S3 per utenti, ruoli e applicazioni.

Utilizza S3 Access Grants per aumentare l'accesso ai dati di Amazon S3 oltre alle autorizzazioni concesse dal ruolo di runtime o dai ruoli IAM collegati alle identità con accesso al tuo cluster Amazon EMR su EKS.

Per ulteriori informazioni, consulta Gestione dell'accesso con S3 Access Grants per Amazon EMR nella Guida alla gestione di Amazon EMR e Gestione dell'accesso con S3 Access Grants nella Guida per l'utente di Amazon Simple Storage Service.

Questa pagina descrive i requisiti per eseguire un processo Spark in Amazon EMR su EKS con l'integrazione di S3 Access Grants. Con Amazon EMR su EKS, S3 Access Grants richiede una dichiarazione di policy IAM aggiuntiva nel ruolo di esecuzione del processo e un'ulteriore configurazione sostitutiva per l'API `StartJobRun`. Per i passaggi per configurare S3 Access Grants con altre implementazioni di Amazon EMR, consulta la seguente documentazione:

- Utilizzo di S3 Access Grants con Amazon EMR
- Utilizzo di S3 Access Grants con EMR serverless

Avvio di un cluster Amazon EMR su EKS con S3 Access Grants per la gestione dei dati

Puoi abilitare S3 Access Grants su Amazon EMR su EKS e avviare un processo Spark. Quando l'applicazione effettua una richiesta di dati S3, Amazon S3 fornisce credenziali temporanee che rientrano nell'ambito del bucket, del prefisso o dell'oggetto specifico.

1. Configura un ruolo di esecuzione dei processi per il cluster Amazon EMR su EKS. Includi le autorizzazioni IAM necessarie per eseguire i processi Spark, `s3:GetDataAccess` e `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**
>
> Se specifichi ruoli IAM destinati all'esecuzione del processo che dispongono di autorizzazioni aggiuntive per accedere direttamente a S3, gli utenti potrebbero essere in grado di accedere ai dati indipendentemente dalle autorizzazioni definite in S3 Access Grants

2. Invia un processo al tuo cluster Amazon EMR su EKS con un'etichetta di rilascio di Amazon EMR 6.15 o successiva e la classificazione `emrfs-site`, come illustrato nell'esempio seguente. Sostituisci i valori in *`red text`* con i valori appropriati per il tuo scenario di utilizzo.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-7.12.0-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"
         }
      }
    ],
  }
}
```

Considerazioni su S3 Access Grants con Amazon EMR su EKS

Per informazioni importanti su supporto, compatibilità e comportamento quando usi Amazon S3 Access Grants con Amazon EMR su EKS, consulta S3 Access Grants considerations with Amazon EMR nella Guida alla gestione di Amazon EMR.

Convalida della conformità per Amazon EMR su EKS

I revisori di terze parti valutano la sicurezza e la conformità di Amazon EMR su EKS nell'ambito di AWS diversi programmi di conformità. Questi includono SOC, PCI, FedRAMP, HIPAA e altri.

Resilienza in Amazon EMR su EKS

L'infrastruttura AWS globale è costruita attorno a AWS regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta Global Infrastructure.AWS

Oltre all'infrastruttura AWS globale, Amazon EMR su EKS offre l'integrazione con Amazon S3 tramite EMRFS per supportare le tue esigenze di resilienza e backup dei dati.

Sicurezza dell'infrastruttura in Amazon EMR su EKS

In quanto servizio gestito, Amazon EMR è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi di AWS sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi chiamate API AWS pubblicate per accedere ad Amazon EMR attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Analisi della configurazione e delle vulnerabilità

AWS gestisce attività di sicurezza di base come l'applicazione di patch al sistema operativo guest (OS) e al database, la configurazione del firewall e il disaster recovery. Queste procedure sono state riviste e certificate dalle terze parti appropriate. Per ulteriori dettagli, consulta le seguenti risorse :

- Convalida della conformità per Amazon EMR su EKS
- Modello di responsabilità condivisa
- Amazon Web Services: panoramica dei processi di sicurezza (whitepaper)

Connessione ad Amazon EMR su EKS con un endpoint VPC di interfaccia

Puoi connetterti direttamente ad Amazon EMR su EKS utilizzando gli endpoint Interface VPC (AWS PrivateLink) nel tuo Virtual Private Cloud (VPC) invece di connetterti tramite Internet. Quando utilizzi

un endpoint VPC di interfaccia, la comunicazione tra il tuo VPC e Amazon EMR su EKS viene condotta interamente all'interno della rete. AWS Ogni endpoint VPC è rappresentato da una o più interfacce di rete elastiche (ENIs) con indirizzi IP privati nelle sottoreti VPC.

L'interfaccia VPC endpoint collega il tuo VPC direttamente ad Amazon EMR su EKS senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione Direct Connect. AWS Le istanze presenti nel tuo VPC non richiedono indirizzi IP pubblici per comunicare con l'API Amazon EMR su EKS.

Puoi creare un endpoint VPC di interfaccia per connetterti ad Amazon EMR su EKS utilizzando i Console di gestione AWS comandi or (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consulta Creazione di un endpoint di interfaccia.

Se dopo aver creato un endpoint VPC di interfaccia abiliti nomi host DNS privati per l'endpoint, l'endpoint Amazon EMR su EKS predefinito restituisce il tuo endpoint VPC. L'endpoint del nome del servizio predefinito per Amazon EMR su EKS è nel formato seguente.

```
emr-containers.Region.amazonaws.com
```

Se non abiliti nomi host DNS privati, Amazon VPC fornisce un nome di endpoint DNS che puoi utilizzare nel formato seguente:

```
VPC_Endpoint_ID.emr-containers.Region.vpce.amazonaws.com
```

Per ulteriori informazioni, consulta Interface VPC Endpoints (AWS PrivateLink) nella Amazon VPC User Guide. Amazon EMR su EKS supporta l'esecuzione di chiamate a tutte le sue operazioni API all'interno del VPC.

Puoi collegare le policy di endpoint VPC a un endpoint VPC per controllare l'accesso per le entità principali IAM. Puoi inoltre associare i gruppi di sicurezza a un endpoint VPC per controllare l'accesso in ingresso e in uscita in base all'origine e alla destinazione del traffico di rete, ad esempio un intervallo di indirizzi IP. Per ulteriori informazioni, consulta Controllo dell'accesso ai servizi con endpoint VPC.

Creazione di una policy di endpoint VPC per Amazon EMR su EKS

Puoi creare una policy per gli endpoint VPC di Amazon per Amazon EMR su EKS per specificare quanto segue:

- Il principale che può o non può eseguire azioni
- Le azioni che possono essere eseguite
- Le risorse sui cui si possono eseguire le azioni

Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC nella Guida per l'utente di Amazon VPC.

Example Policy degli endpoint VPC per negare tutti gli accessi da un account specificato AWS

La seguente politica degli endpoint VPC nega all' AWS account *123456789012* tutti gli accessi alle risorse che utilizzano l'endpoint.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example Policy di endpoint VPC per consentire l'accesso VPC solo a un'entità principale IAM (utente) specificata

La seguente policy sugli endpoint VPC consente l'accesso completo solo all'utente *lijuan* IAM nell'account. AWS *123456789012* A tutte le altre entità principali IAM viene negato l'accesso utilizzando l'endpoint.

```
{
```

```
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
                ]
            }
        }
    ]
}
```

Example Policy di endpoint VPC per consentire azioni Amazon EMR su EKS di sola lettura

La seguente policy sugli endpoint VPC consente solo *123456789012* all' AWS account di eseguire le azioni Amazon EMR su EKS specificate.

Le azioni specificate forniscono l'accesso di sola lettura equivalente per Amazon EMR su EKS. Tutte le altre azioni sul VPC vengono negate per l'account specificato. A tutti gli altri account viene negato l'accesso. Per l'elenco delle operazioni Amazon EMR su EKS, consulta Operazioni, risorse e chiavi di condizione per Amazon EMR su 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 Policy di endpoint VPC per negare l'accesso a un cluster virtuale specificato

La seguente politica degli endpoint VPC consente l'accesso completo a tutti gli account e i principali, ma nega qualsiasi accesso AWS dell'account *123456789012* alle azioni eseguite sul cluster virtuale con l'ID del cluster. *A1B2CD34EF5G* Altre azioni Amazon EMR su EKS che non supportano le autorizzazioni a livello di risorsa per i cluster virtuali sono comunque consentite. Per un elenco delle azioni Amazon EMR su EKS e dei tipi di risorse corrispondenti, consulta Operazioni, risorse e chiavi di condizione per Amazon EMR su EKS nella Guida per l'utente di AWS Identity and Access Management .

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:emr-containers:us-west-2:123456789012:/
virtualclusters/A1B2CD34EF5G",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Configurazione dell'accesso multi-account per Amazon EMR su EKS

Puoi impostare l'accesso multi-account per Amazon EMR su EKS. L'accesso su più account consente agli utenti di un AWS account di eseguire Amazon EMR su job EKS e accedere ai dati sottostanti che appartengono a AWS un altro account.

Prerequisiti

Per configurare l'accesso tra più account per Amazon EMR su EKS, completerai le attività accedendo ai seguenti AWS account:

- `AccountA`- Un AWS account in cui hai creato un cluster virtuale Amazon EMR su EKS registrando Amazon EMR con uno spazio dei nomi su un cluster EKS.
- `AccountB`- Un AWS account che contiene un bucket Amazon S3 o una tabella DynamoDB a cui desideri che accedano i tuoi job Amazon EMR on EKS.

Prima di configurare l'accesso tra più account, devi avere a portata di mano quanto segue nei tuoi AWS account:

- Un cluster virtuale Amazon EMR su EKS in `AccountA` in cui eseguire i tuoi processi.
- Un ruolo di esecuzione di processo in `AccountA` che dispone delle autorizzazioni necessarie per eseguire processi nel cluster virtuale. Per ulteriori informazioni, consultare Creazione di un ruolo di esecuzione di processo e Uso dei ruoli di esecuzione di processo con Amazon EMR su EKS.

Come accedere a un bucket Amazon S3 o a una tabella DynamoDB su più account

Per impostare l'accesso multi-account per Amazon EMR su EKS, completa la procedura seguente.

1. Crea un bucket Amazon S3, `cross-account-bucket`, in `AccountB`. Per ulteriori informazioni, consulta Creazione di un bucket. Se si desidera avere un accesso multi-account a DynamoDB, è anche possibile creare una tabella DynamoDB in `AccountB`. Per ulteriori informazioni, consulta Creazione di una tabella DynamoDB.
2. Crea un ruolo IAM `Cross-Account-Role-B` in `AccountB` per accedere a `cross-account-bucket`.
    1. Accedi alla console IAM.
    2. Scegli Roles (Ruoli), quindi crea un nuovo ruolo: `Cross-Account-Role-B`. Per ulteriori informazioni su come creare ruoli IAM, consulta Creazione di ruoli IAM nella Guida per l'utente IAM.
    3. Crea una policy IAM che specifichi le autorizzazioni per `Cross-Account-Role-B` per accedere al bucket S3 `cross-account-bucket`, come dimostra la seguente istruzione di

policy. Quindi, allega la policy IAM a `Cross-Account-Role-B`. Per ulteriori informazioni, consulta Creazione di una nuova policy nella Guida per l'utente IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::cross-account-bucket",
        "arn:aws:s3:::cross-account-bucket/*"
      ],
      "Sid": "AllowS3"
    }
  ]
}
```

Se è richiesto l'accesso a DynamoDB, crea una policy IAM che specifichi le autorizzazioni per accedere alla tabella DynamoDB multi-account. Quindi, allega la policy IAM a `Cross-Account-Role-B`. Per ulteriori informazioni, consulta Creazione di una tabella DynamoDB nella Guida per l'utente IAM.

Di seguito è riportata una policy per accedere a una tabella DynamoDB, `CrossAccountTable`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:*"
      ],
      "Resource": [
        "arn:aws:dynamodb:us-east-1:*:table/CrossAccountTable"
```

```
      ],
      "Sid": "AllowDYNAMODB"
    }
  ]
}
```

3. Modifica la relazione di fiducia per il ruolo `Cross-Account-Role-B`.

   1. Per configurare la relazione di fiducia per il ruolo, seleziona la scheda Trust Relationships (Relazioni di fiducia) nella console IAM per il ruolo creato nel Passaggio 2: `Cross-Account-Role-B`.
   2. Seleziona Edit Trust Relationship (Modifica relazione di fiducia).
   3. Aggiungi il seguente documento di policy, che consente a `Job-Execution-Role-A` in `AccountA` di assumere questo ruolo `Cross-Account-Role-B`.

      JSON

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

4. Concedi `Job-Execution-Role-A` in `AccountA` con autorizzazione del ruolo STS Assume per l'assunzione di `Cross-Account-Role-B`.

   1. Nella console IAM per l' AWS account`AccountA`, seleziona`Job-Execution-Role-A`.
   2. Aggiungi la seguente istruzione di policy a `Job-Execution-Role-A` per autorizzare l'operazione `AssumeRole` nel ruolo `Cross-Account-Role-B`.

JSON

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

5. Per accedere ad Amazon S3, imposta i seguenti parametri `spark-submit` (`spark conf`) durante l'invio del processo ad Amazon EMR su EKS.

> **Note**
>
> Per impostazione predefinita, EMRFS utilizza il ruolo di esecuzione di processo per accedere al bucket S3 dal processo. Ma quando `customAWSCredentialsProvider` è impostato su `AssumeRoleAWSCredentialsProvider`, EMRFS utilizza il ruolo corrispondente specificato con `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` invece di `Job-Execution-Role-A` per accedere ad Amazon S3.

- `--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**
>
> È necessario impostare `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` sia per l'executor che per il driver `env` nella configurazione del processo Spark.

Per l'accesso multi-account DynamoDB, è necessario impostare `--conf spark.dynamodb.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCr`

6. Esegui il processo Amazon EMR su EKS con accesso multi-account, come illustrato nell'esempio seguente.

```
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" }}}'
```

Assegnazione di tag alle risorse Amazon EMR su EKS

Per semplificare la gestione delle risorse Amazon EMR su EKS, puoi assegnare metadati personalizzati a ciascuna risorsa utilizzando dei tag. In questo argomento viene fornita una panoramica della funzione dei tag e viene illustrato come creare i tag.

Argomenti

Nozioni di base sui tag

Un tag è un'etichetta che si assegna a una AWS risorsa. Ogni tag è composto da una chiave e da un valore opzionale, entrambi personalizzabili.

I tag consentono di classificare le AWS risorse in base ad attributi quali scopo, proprietario o ambiente. Se disponi di un numero elevato di risorse, puoi individuare rapidamente una risorsa specifica in base ai tag assegnati. Ad esempio, puoi definire un set di tag per i cluster Amazon EMR su EKS per monitorare il proprietario di ogni cluster e il livello di pila. Consigliamo di definire un set coerente di chiavi di tag per ciascun tipo di risorsa. È possibile cercare e filtrare le risorse in base ai tag aggiunti.

I tag non vengono assegnati in automatico alle risorse. Dopo aver aggiunto un tag, puoi modificarne le chiavi e i valori oppure rimuovere i tag da una risorsa in qualsiasi momento. Se elimini una risorsa, verranno eliminati anche tutti i tag a essa associati.

I tag non hanno alcun significato semantico per Amazon EMR su EKS e vengono interpretati rigorosamente come una stringa di caratteri.

Un valore di tag può essere una stringa vuota, ma non nullo. Una chiave di tag non può essere una stringa vuota. Se aggiungi un tag con la stessa chiave di un tag esistente a una risorsa specifica, il nuovo valore sovrascrive quello precedente.

Se utilizzi AWS Identity and Access Management (IAM), puoi controllare quali utenti del tuo AWS account sono autorizzati a gestire i tag.

Per esempi di policy di controllo dell'accesso basate su tag, consulta Policy per il controllo degli accessi basato su tag.

Assegnazione di tag alle risorse

Puoi applicare tag a cluster virtuali nuovi o esistenti e ad esecuzioni di processo il cui stato è attivo. Gli stati attivi per le esecuzioni di processo includono: `PENDING`, `SUBMITTED`, `RUNNING` e `CANCEL_PENDING`. Gli stati attivi per i cluster virtuali includono: `RUNNING`, `TERMINATING` e `ARRESTED`. Per ulteriori informazioni, consultare Stati delle esecuzioni di processi e Stati dei cluster virtuali.

Quando un cluster virtuale viene terminato, i tag vengono eliminati e non sono più accessibili.

Se utilizzi l'API Amazon EMR on EKS, o un AWS SDK AWS CLI, puoi applicare tag a nuove risorse utilizzando il parametro tags nell'azione API pertinente. Puoi applicare tag a risorse esistenti utilizzando l'operazione API `TagResource`.

Puoi utilizzare alcune operazioni per la creazione di risorse per specificare tag per una risorsa durante la sua creazione. In questo caso, se i tag non possono essere applicati durante la creazione della risorsa, la risorsa non viene creata. Mediante questo meccanismo, le risorse a cui desideri applicare tag al momento della creazione vengono create con tag specifici o non vengono create affatto. Se aggiungi tag alle risorse al momento della creazione, non è necessario eseguire script di assegnazione di tag personalizzati dopo la creazione di una risorsa.

Nella seguente tabella sono descritte le risorse Amazon EMR su EKS a cui puoi aggiungere un tag.

Risorsa	Supporta tag	Supporta la propagazione di tag	Supporta l'etichet tatura alla creazione (Amazon EMR su EKS API e AWS CLI SDK) AWS	API per la creazione (i tag possono essere aggiunti durante la creazione)
Cluster virtuale	Sì	No. I tag associati a un cluster virtuale	Sì	CreateVir tualCluster

Risorsa	Supporta tag	Supporta la propagazione di tag	Supporta l'etichettatura alla creazione (Amazon EMR su EKS API e AWS CLI SDK) AWS	API per la creazione (i tag possono essere aggiunti durante la creazione)
		non vengono propagati alle esecuzioni di processo inviate a tale cluster virtuale.		
Esecuzioni di processo	Sì	No	Sì	StartJobRun

Limitazioni applicate ai tag

Si applicano le seguenti limitazioni di base ai tag:

- Numero massimo di tag per risorsa: 50
- Per ciascuna risorsa, ogni chiave del tag deve essere univoca e ogni chiave del tag può avere un solo valore.
- Lunghezza massima della chiave: 128 caratteri Unicode in formato UTF-8
- Lunghezza massima del valore: 256 caratteri Unicode in formato UTF-8
- Se il tuo schema di tagging viene utilizzato su più AWS servizi e risorse, ricorda che altri servizi potrebbero avere restrizioni sui caratteri consentiti. I caratteri generalmente consentiti sono: lettere, numeri, spazi rappresentabili in formato UTF-8 e i seguenti caratteri speciali: + - = . _ : / @.
- I valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole.
- Un valore di tag può essere una stringa vuota, ma non nullo. Una chiave di tag non può essere una stringa vuota.
- Non utilizzare `aws:`, `AWS:` o qualsiasi combinazione di maiuscole o minuscole di un tale prefisso per chiavi o valori. Questi sono riservati solo all' AWS uso.

Lavora con i tag utilizzando AWS CLI l'API Amazon EMR on EKS

Utilizza i seguenti AWS CLI comandi o Amazon EMR sulle operazioni API EKS per aggiungere, aggiornare, elencare ed eliminare i tag per le tue risorse.

Attività	AWS CLI	Azione API
Aggiunta o sovrascrittura di uno o più tag	tag-resource	TagResource
Elencazione dei tag associati a una risorsa	list-tags-for-resource	ListTagsForResource
Eliminazione di uno o più tag	untag-resource	UntagResource

I seguenti esempi mostrano come aggiungere o rimuovere tag alle o dalle risorse utilizzando la AWS CLI.

Esempio 1: assegnazione di un tag a un cluster virtuale esistente

Il comando seguente aggiunge un tag a un cluster virtuale esistente.

```
aws emr-containers tag-resource --resource-arn resource_ARN --tags team=devs
```

Esempio 2: rimozione di un tag da un cluster virtuale esistente

Il comando seguente elimina un tag da un cluster virtuale esistente.

```
aws emr-containers untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Esempio 3: elencazione dei tag di una risorsa

Il comando seguente elenca i tag associati a una risorsa esistente.

```
aws emr-containers list-tags-for-resource --resource-arn resource_ARN
```

Risoluzione dei problemi di Amazon EMR su EKS

La presente sezione descrive come risolvere i problemi con Amazon EMR su EKS. Per informazioni sulla risoluzione di problemi generali con Amazon EMR, consulta Risoluzione dei problemi di un cluster nella Guida per la gestione di Amazon EMR.

Argomenti

Risoluzione dei problemi relativi ai lavori che utilizzano PersistentVolumeClaims (PVC)

Se devi creare, elencare o eliminare PersistentVolumeClaims (PVC) per un lavoro ma non aggiungi le autorizzazioni PVC al ruolo Kubernetes predefinito emr-containers, il lavoro fallisce quando lo invii. Senza queste autorizzazioni, il ruolo emr-containers non può creare i ruoli necessari per il driver Spark o il client Spark. Non è sufficiente aggiungere autorizzazioni ai ruoli driver o client Spark, come suggerito dai messaggi di errore. Il ruolo primario emr-containers deve includere anche le autorizzazioni richieste. In questa sezione viene descritto come aggiungere le autorizzazioni richieste al ruolo primario emr-containers.

Verifica

Per verificare se il tuo ruolo emr-containers dispone delle autorizzazioni necessarie, imposta la variabile NAMESPACE con il tuo valore e quindi esegui il seguente comando:

```
export NAMESPACE=YOUR_VALUE
kubectl describe role emr-containers -n ${NAMESPACE}
```

Inoltre, per verificare se i ruoli Spark e client dispongono delle autorizzazioni necessarie, esegui il seguente comando:

```
kubectl describe role emr-containers-role-spark-driver -n ${NAMESPACE}
kubectl describe role emr-containers-role-spark-client -n ${NAMESPACE}
```

Se le autorizzazioni non sono disponibili, procedi con la patch, come segue.

Patch

1. Se i processi senza le autorizzazioni sono attualmente in esecuzione, arrestarli.
2. Crea un file denominato RBAC_Patch.py come segue:

```
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", "patch"]
         }

    ]

    driverRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch", "deletecollection"]
        },
        {
            "apiGroups": [""],
            "resources": ["services"],
            "verbs": ["get", "list", "describe", "create", "delete", "watch",
 "deletecollection"]
        },
        {
            "apiGroups": [""],
            "resources": ["configmaps", "pods"],
            "verbs": ["deletecollection"]
        }
```

```
    ]

    clientRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch"]
        }
    ]

    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. Esegui lo script Python:

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

4. Viene visualizzata una differenza kubectl tra le nuove autorizzazioni e quelle vecchie. Premi y per applicare la patch al ruolo.
5. Verifica i tre ruoli con autorizzazioni aggiuntive come segue:

```
kubectl describe role -n ${NAMESPACE}
```

6. Esegui lo script python:

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

7. Dopo aver eseguito il comando, verrà visualizzata una differenza kubectl tra le nuove autorizzazioni e quelle vecchie. Premi y per applicare la patch al ruolo.
8. Verifica i tre ruoli con autorizzazioni aggiuntive:

```
kubectl describe role -n ${NAMESPACE}
```

9. Invia nuovamente il processo.

Patch manuale

Se l'autorizzazione richiesta dalla tua applicazione si applica a qualcosa di diverso dalle regole PVC, puoi aggiungere manualmente le autorizzazioni Kubernetes per il tuo cluster virtuale Amazon EMR secondo necessità.

> **Note**
>
> Il ruolo emr-containers è un ruolo primario. Ciò significa che deve fornire tutte le autorizzazioni necessarie prima di poter modificare i ruoli driver o client sottostanti.

1. Scarica le autorizzazioni attuali nei file yaml eseguendo i comandi seguenti:

```
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. In base all'autorizzazione richiesta dalla tua applicazione, modifica ogni file e aggiungi regole aggiuntive come le seguenti:
   - emr-containers-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
```

   - driver-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
```

```
  verbs:
  - list
  - create
  - delete
  - patch
  - deletecollection
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - describe
  - create
  - delete
  - watch
  - deletecollection
- apiGroups:
  - ""
  resources:
  - configmaps
  - pods
  verbs:
  - deletecollection
```

- client-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
```

3. Rimuovi i seguenti attributi con i relativi valori. Ciò è necessario per applicare l'aggiornamento.
   - creationTimestamp
   - resourceVersion
   - uid

4. Infine, esegui la patch:

```
kubectl apply -f emr-containers-role-patch.yaml
kubectl apply -f driver-role-patch.yaml
kubectl apply -f client-role-patch.yaml
```

Risoluzione dei problemi relativi al dimensionamento automatico verticale di Amazon EMR su EKS

Fai riferimento alle sezioni seguenti se riscontri problemi durante la configurazione dell'operatore di dimensionamento automatico verticale di Amazon EMR su EKS in un cluster Amazon EKS con Operator Lifecycle Manager. Per ulteriori informazioni, compresa la procedura per completare l'installazione, consulta Uso del dimensionamento automatico verticale con i processi Spark di Amazon EMR.

Errore 403 Accesso negato

Se hai seguito le fasi indicate in Installazione di Operator Lifecycle Manager (OLM) sul cluster Amazon EKS, eseguito il comando `olm status` e quest'ultimo ha restituito un errore `403 Forbidden` come quello riportato di seguito, potresti non aver ottenuto i token di autenticazione al repository Amazon ECR per l'operatore.

Per risolvere il problema, ripeti la procedura indicata in Installazione dell'operatore di dimensionamento automatico verticale di Amazon EMR su EKS per ottenere i token. Quindi, riprova l'installazione.

```
Error: FATA[0002] Failed to run bundle: pull bundle image: error pulling image IMAGE.
error resolving name : unexpected status code [manifests latest]: 403 Forbidden
```

Impossibile trovare lo spazio dei nomi Kubernetes

Quando configuri l'operatore di dimensionamento automatico verticale di Amazon EMR su EKS in un cluster Amazon EKS, potresti ricevere un errore `namespaces not found` come il seguente:

```
FATA[0020] Failed to run bundle: create catalog: error creating catalog source:
 namespaces "NAME" not found.
```

Se lo spazio dei nomi specificato non esiste, OLM non installerà l'operatore di dimensionamento automatico verticale. Per risolvere questo problema, utilizza il comando seguente per creare lo spazio dei nomi. Quindi, riprova l'installazione.

```
kubectl create namespace NAME
```

Errore durante il salvataggio delle credenziali Docker

Per configurare il dimensionamento automatico verticale, devi autenticare e recuperare le immagini Docker relative al dimensionamento automatico verticale di Amazon EMR su EKS. Quando esegui questa operazione, potresti ricevere un errore simile al seguente se Docker non è in esecuzione:

```
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'
```

Per risolvere questo problema, verifica che Docker sia in esecuzione o apri Docker Desktop. Quindi, prova a salvare nuovamente le credenziali.

Risoluzione dei problemi relativi all'operatore Amazon EMR su EKS

Fai riferimento alle seguenti sezioni se riscontri problemi con l'operatore Spark di Amazon EMR su EKS. Per ulteriori informazioni, compresa la procedura per completare l'installazione, consulta Esecuzione dei processi Spark con l'operatore Spark.

Errore nell'installazione del grafico Helm

Se hai seguito le fasi indicate in Installazione dell'operatore Spark e hai ricevuto un errore `INSTALLATION FAILED` come quello riportato di seguito quando hai provato a installare o verificare il grafico Helm, potresti non aver ottenuto i token di autenticazione al repository Amazon ECR per l'operatore.

Per risolvere il problema, ripeti la fase indicata in Installazione dell'operatore Spark per autenticare il client Helm nel registro Amazon ECR. Quindi, ritenta la fase di installazione.

```
Error: INSTALLATION FAILED: Kubernetes cluster unreachable: the server has asked for
 the client to provide credentials
```

UnsupportedFileSystemException: No FileSystem per lo schema «s3"

Potresti riscontrare la seguente eccezione nel thread "main":

```
org.apache.hadoop.fs.UnsupportedFileSystemException: No FileSystem for scheme "s3"
```

In tal caso, aggiungi le seguenti eccezioni alle specifiche `SparkApplication`:

```
 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
```

Endpoint di servizio e Service Quotas Amazon EMR su EKS

Di seguito sono riportati gli endpoint di servizio e le Service Quotas di Amazon EMR su EKS. Per connettersi a livello di codice a un AWS servizio, si utilizza un endpoint. Oltre agli AWS endpoint standard, alcuni AWS servizi offrono endpoint FIPS in regioni selezionate. Per ulteriori informazioni, consulta Endpoint del servizio AWS. Le Service Quotas, conosciute anche come limiti, rappresentano il numero massimo di risorse o operazioni di servizio dell'account AWS . Per ulteriori informazioni, consulta Service Quotas di AWS.

Endpoint di servizio

Regione AWS nome	Codice	Endpoint	Protocollo
Stati Uniti orientali (Virginia settentrionale)	us-east-1	`emr-containers.us-east-1.amazonaws.com`	HTTPS
Stati Uniti orientali (Ohio)	us-east-2	`emr-containers.us-east-2.amazonaws.com`	HTTPS
Stati Uniti occidentali (California settentrionale)	us-west-1	`emr-containers.us-west-1.amazonaws.com`	HTTPS
US West (Oregon)	us-west-2	`emr-containers.us-west-2.amazonaws.com`	HTTPS
Asia Pacifico (Tokyo)	ap-northeast-1	`emr-containers.ap-northeast-1.amazonaws.com`	HTTPS
Asia Pacifico (Seoul)	ap-northeast-2	`emr-containers.ap-northeast-2.amazonaws.com`	HTTPS
Asia Pacifico (Osaka-Locale)	ap-northeast-3	`emr-containers.ap-northeast-3.amazonaws.com`	HTTPS

Regione AWS nome	Codice	Endpoint	Protocollo
Asia Pacifico (Mumbai)	ap-south-1	emr-containers.ap-south-1.amazonaws.com	HTTPS
Asia Pacific (Hyderabad)	ap-south-2	emr-containers.ap-south-2.amazonaws.com	HTTPS
Asia Pacifico (Singapore)	ap-southeast-1	emr-containers.ap-southeast-1.amazonaws.com	HTTPS
Asia Pacifico (Sydney)	ap-southeast-2	emr-containers.ap-southeast-2.amazonaws.com	HTTPS
Asia Pacifico (Giacarta)	ap-southeast-3	emr-containers.ap-southeast-3.amazonaws.com	HTTPS
Asia Pacifico (Hong Kong)	ap-east-1	emr-containers.ap-east-1.amazonaws.com	HTTPS
Africa (Città del Capo)	af-south-1	emr-containers.af-south-1.amazonaws.com	HTTPS
Canada (Centrale)	ca-central-1	emr-containers.ca-central-1.amazonaws.com	HTTPS
Cina (Ningxia)	cn-northwest-1	emr-containers.cn-northwest-1.amazonaws.com.cn	HTTPS
Cina (Pechino)	cn-north-1	emr-containers.cn-north-1.amazonaws.com.cn	HTTPS
Europa (Francoforte)	eu-central-1	emr-containers.eu-central-1.amazonaws.com	HTTPS
Europa (Zurigo)	eu-central-2	emr-containers.eu-central-2.amazonaws.com	HTTPS

Regione AWS nome	Codice	Endpoint	Protocollo
Europa (Irlanda)	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 (Paris)	eu-west-3	emr-containers.eu-west-3.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	emr-containers.eu-north-1.amazonaws.com	HTTPS
Europa (Milano)	eu-south-1	emr-containers.eu-south-1.amazonaws.com	HTTPS
Europa (Spagna)	eu-south-2	emr-containers.eu-south-2.amazonaws.com	HTTPS
Israele (Tel Aviv)	il-central-1	emr-containers.il-central-1.amazonaws.com	HTTPS
Sud America (San Paolo)	sa-east-1	emr-containers.sa-east-1.amazonaws.com	HTTPS
Medio Oriente (Emirati Arabi Uniti)	me-central-1	emr-containers.me-central-1.amazonaws.com	HTTPS
Medio Oriente (Bahrein)	me-south-1	emr-containers.me-south-1.amazonaws.com	HTTPS
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	emr-containers.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	emr-containers.us-gov-west-1.amazonaws.com	HTTPS

Quote del servizio

Amazon EMR su EKS limita le seguenti richieste API per ogni AWS account in base alla regione. Per ulteriori informazioni su come viene applicata la limitazione, consulta API Request Throttling nell'Amazon API Reference. EC2 Puoi richiedere un aumento delle quote di limitazione delle API per il tuo AWS account seguendo la guida che segue.

Azione API	Capacità massima bucket	Tasso di riempimento bucket (al secondo)
CancelJobRun	25	1
CreateJobTemplate	25	1
CreateManagedEndpoint	25	1
CreateSecurityConfiguration	25	1
CreateVirtualCluster	25	1
DeleteJobTemplate	25	1
DeleteManagedEndpoint	25	1
DeleteVirtualCluster	25	1
DescribeJobRun	100	20
DescribeJobTemplate	25	1
DescribeManagedEndpoint	100	5
DescribeSecurityConfiguration	25	1
DescribeVirtualCluster	100	5
GetManagedEndpoint SessionCredentials	25	1
ListJobRuns	100	5

Azione API	Capacità massima bucket	Tasso di riempimento bucket (al secondo)
ListJobTemplates	25	1
ListManagedEndpoints	25	1
ListSecurityConfigurations	25	1
ListVirtualClusters	100	5
StartJobRun	25	1
Throttle quota for all EMR on EKS API requests	200	20

Quando crei il tuo AWS account, impostiamo delle quote predefinite (dette anche limiti) sulle tue AWS risorse in base alla regione. Se tenti di superare la quota per una risorsa, la richiesta ha esito negativo. In questo caso, puoi ridurre l'utilizzo delle risorse o richiedere un aumento delle quote.

La console Service Quotas è una posizione centrale in cui è possibile visualizzare e gestire le quote per AWS i servizi e richiedere aumenti delle quote per molte delle risorse utilizzate. Utilizza le informazioni sulle quote fornite qui per gestire la tua AWS infrastruttura. Pianifica le richieste di incremento delle quote con un certo anticipo rispetto a quando ne avrai effettivamente bisogno.

Visualizza le quote e richiedi aumenti delle quote

Puoi visualizzare le quote di servizio correnti per ogni regione utilizzando la console Service Quotas. Per istruzioni dettagliate, vedere Visualizzazione delle quote di servizio nella Guida per l'utente di Service Quotas.

Puoi richiedere un aumento della quota dalla console di AWS gestione o utilizzando la AWS CLI. I passaggi sono descritti in dettaglio in Richiesta di aumento della quota.

Rilasci di Amazon EMR su EKS

Un rilascio di Amazon EMR è un insieme di applicazioni open source dell'ecosistema di big data. Ogni rilascio comprende diverse applicazioni, componenti e funzionalità di big data che selezioni di avere per l'implementazione e la configurazione di Amazon EMR su EKS quando esegui un processo.

A partire dai rilasci 5.32.0 e 6.2.0 di Amazon EMR, puoi implementare Amazon EMR su EKS. Questa opzione di implementazione non è disponibile con le versioni precedenti di Amazon EMR. È necessario specificare una versione supportata quando si invia il processo.

Amazon EMR su EKS utilizza la seguente forma di etichetta di rilascio: `emr-x.x.x-latest` o `emr-x.x.x-yyyymmdd` con una data di rilascio specifica. Ad esempio `emr-7.12.0-latest` o `emr-7.12.0-20210129`. Quando utilizzi il suffisso `-latest`, garantisci che la tua versione di Amazon EMR includa sempre gli aggiornamenti di sicurezza più recenti.

> **Note**
>
> Per un confronto tra Amazon EMR su EKS e Amazon EMR in esecuzione su, EC2 consulta Amazon EMR sul sito Web. FAQs AWS

Argomenti

- Rilasci 7.0.0 di Amazon EMR su EKS
- Rilasci 6.15.0 di Amazon EMR su EKS
- Rilasci 6.14.0 di Amazon EMR su EKS
- Rilasci 6.13.0 di Amazon EMR su EKS
- Rilasci 6.12.0 di Amazon EMR su EKS
- Rilasci 6.11.0 di Amazon EMR su EKS
- Rilasci 6.10.0 di Amazon EMR su EKS
- Rilasci 6.9.0 di Amazon EMR su EKS
- Rilasci 6.8.0 di Amazon EMR su EKS
- Rilasci 6.7.0 di Amazon EMR su EKS
- Rilasci 6.6.0 di Amazon EMR su EKS
- Rilasci 6.5.0 di Amazon EMR su EKS
- Rilasci 6.4.0 di Amazon EMR su EKS
- Rilasci 6.3.0 di Amazon EMR su EKS
- Rilasci 6.2.0 di Amazon EMR su EKS
- Rilasci 5.36.0 di Amazon EMR su EKS
- Rilasci 5.35.0 di Amazon EMR su EKS
- Rilasci 5.34.0 di Amazon EMR su EKS
- Rilasci 5.33.0 di Amazon EMR su EKS
- Rilasci 5.32.0 di Amazon EMR su EKS

Amazon EMR nelle versioni EKS 7.12.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.12.0 in generale, consulta Amazon EMR 7.12.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.12

Le seguenti versioni di Amazon EMR 7.12.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.12.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag con l'immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.12.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.12.0-flink-latest
- emr-7.12.0-flink-20251111

Spark releases

Le seguenti versioni di Amazon EMR 7.12.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.12.0-più recente
- emr-7.12.0-20251111
- emr-7.12.0-spark-rapids-latest
- emr-7.12.0-spark-rapids-20251111
- emr-7.12.0-java11-latest
- emr-7.12.0-java11-20251111
- emr-7.12.0-java8-latest
- emr-7.12.0-java8-20251111
- emr-7.12.0-spark-rapids-java8-latest
- emr-7.12.0-spark-rapids-java8-20251111
- notebook-spark/emr-7.12.0-latest
- notebook-spark/emr-7.12.0-20251111
- notebook-spark/emr-7.12.0-spark-rapids-latest
- notebook-spark/emr-7.12.0-spark-rapids-20251111
- notebook-spark/emr-7.12.0-java11-latest
- notebook-spark/emr-7.12.0-java11-20251111
- notebook-spark/emr-7.12.0-java8-latest
- notebook-spark/emr-7.12.0-java8-20251111
- notebook-spark/emr-7.12.0-spark-rapids-java8-latest
- notebook-spark/emr-7.12.0-spark-rapids-java8-20251111
- notebook-python/emr-7.12.0-latest

- notebook-python/emr-7.12.0-20251111
- notebook-python/emr-7.12.0-spark-rapids-latest
- notebook-python/emr-7.12.0-spark-rapids-20251111
- notebook-python/emr-7.12.0-java11-latest
- notebook-python/emr-7.12.0-java11-20251111
- notebook-python/emr-7.12.0-java8-latest
- notebook-python/emr-7.12.0-java8-20251111
- notebook-python/emr-7.12.0-spark-rapids-java8-latest
- notebook-python/emr-7.12.0-spark-rapids-java8-20251111
- livy/emr-7.12.0-latest
- livy/emr-7.12.0-20251111
- livy/emr-7.12.0-java11-latest
- livy/emr-7.12.0-java11-20251111
- livy/emr-7.12.0-java8-latest
- livy/emr-7.12.0-java8-20251111

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.12.0:

- Applicazioni supportate - AWS SDK per Java 2.35.5 and 1.12.792, Apache Spark 3.5.6-amzn-1, Apache Hudi 1.0.2-amzn-1, Apache Iceberg 1.10.0-amzn-0, Delta 3.3.2-amzn-1, Apache Spark RAPIDS 25.04.0-amzn-0, Apache Flink 1.20.0-amzn-6
- Componenti supportati -`emr-ddb`,,`emr-goodies`,`emr-s3-select`,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`. `iceberg` `spark-kubernetes`
- Classificazioni di configurazione supportate

  Da usare con StartJobRune CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .

Classificazioni	Descrizioni
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Modifiche e funzionalità

Le seguenti funzionalità sono incluse nella versione 7.12.0 di Amazon EMR su EKS:

- Iceberg Materialized Views — A partire da EMR 7.12.0, EMR Spark supporta la creazione e la gestione di Iceberg Materialized Views (MV).
- Hudi Full Table Access — A partire da EMR 7.12.0, EMR supporta ora il controllo Full Table Access (FTA) per Apache Hudi in Apache Spark in base alle politiche definite in Lake Formation. Questa funzionalità consente le operazioni di lettura e scrittura dai job di Amazon EMR Spark sulle tabelle registrate di Lake Formation quando il ruolo lavorativo ha accesso completo alla tabella.
- Aggiornamento della versione Iceberg — EMR 7.12.0 supporta la versione 1.10 di Apache Iceberg.
- Registrazione per carichi di lavoro interattivi Livy: a partire da EMR 7.12.0, EMR supporta la registrazione completa dei componenti chiave del sistema per migliorare la risoluzione dei problemi relativi agli errori dei job Livy Spark. Questa funzionalità fornirà al servizio EMR l'accesso a Livy e SecretAgent registri aggiuntivi per semplificare la risoluzione dei problemi.

emr-7.12.0-più recente

Note di rilascio: `emr-7.12.0-latest` attualmente indica `emr-7.12.0-20251111`.

Regioni: `emr-7.12.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.12.0:latest`

emr-7.12.0-20251111

Note sulla versione: è stato rilasciato nel novembre 2025. `emr-7.12.0-20251111` Questa è la versione iniziale di Amazon EMR 7.12.0 (Spark).

Regioni: `emr-emr-7.12.0-20251111` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.12.0-20251111`

emr-7.12.0-flink-latest

Note di rilascio: attualmente punta a `emr-7.12.0-flink-latest` `emr-7.12.0-flink-20251111`

Regioni: `emr-7.12.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.12.0-flink:latest`

emr-7.12.0-flink-20251111

Note sulla versione: è stato rilasciato nel novembre 2025. `7.12.0-flink-20251111` Questa è la versione iniziale di Amazon EMR 7.12.0 (Flink).

Regioni: `emr-7.12.0-flink-20251111` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.12.0-flink:20251111`

Amazon EMR nelle versioni EKS 7.11.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.11.0 in generale, consulta Amazon EMR 7.11.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.11

Le seguenti versioni di Amazon EMR 7.11.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.11.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.11.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.11.0-flink-latest
- emr-7.11.0-flink-20251020

Spark releases

Le seguenti versioni di Amazon EMR 7.11.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.11.0-più recente
- emr-7.11.0-20251020
- emr-7.11.0-spark-rapids-latest
- emr-7.11.0-spark-rapids-20251020
- emr-7.11.0-java11-latest
- emr-7.11.0-java11-20251020
- emr-7.11.0-java8-latest
- emr-7.11.0-java8-20251020
- emr-7.11.0-spark-rapids-java8-latest
- emr-7.11.0-spark-rapids-java8-20251020
- notebook-spark/emr-7.11.0-latest
- notebook-spark/emr-7.11.0-20251020
- notebook-spark/emr-7.11.0-spark-rapids-latest
- notebook-spark/emr-7.11.0-spark-rapids-20251020
- notebook-spark/emr-7.11.0-java11-latest
- notebook-spark/emr-7.11.0-java11-20251020
- notebook-spark/emr-7.11.0-java8-latest
- notebook-spark/emr-7.11.0-java8-20251020
- notebook-spark/emr-7.11.0-spark-rapids-java8-latest
- notebook-spark/emr-7.11.0-spark-rapids-java8-20251020
- notebook-python/emr-7.11.0-latest
- notebook-python/emr-7.11.0-20251020
- notebook-python/emr-7.11.0-spark-rapids-latest
- notebook-python/emr-7.11.0-spark-rapids-20251020
- notebook-python/emr-7.11.0-java11-latest
- notebook-python/emr-7.11.0-java11-20251020
- notebook-python/emr-7.11.0-java8-latest
- notebook-python/emr-7.11.0-java8-20251020

- notebook-python/emr-7.11.0-spark-rapids-java8-latest
- notebook-python/emr-7.11.0-spark-rapids-java8-20251020
- livy/emr-7.11.0-latest
- livy/emr-7.11.0-20251020
- livy/emr-7.11.0-java11-latest
- livy/emr-7.11.0-java11-20251020
- livy/emr-7.11.0-java8-latest
- livy/emr-7.11.0-java8-20251020

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.11.0:

- Applicazioni supportate - AWS SDK per Java 2.35.5 and 1.12.792, Apache Spark 3.5.6-amzn-0, Apache Hudi 1.0.2-amzn-0, Apache Iceberg 1.9.1-amzn-0, Delta 3.3.2-amzn-0, Apache Spark RAPIDS 25.04.0-amzn-0, Apache Flink 1.20.0-amzn-5
- Componenti supportati -`emr-ddb`,,`emr-goodies`,`emr-s3-select`,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`. `iceberg` `spark-kubernetes`
- Classificazioni di configurazione supportate

  Da usare con StartJobRune CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.

Classificazioni	Descrizioni
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Modifiche e funzionalità

Le seguenti modifiche sono incluse nella versione 7.11.0 di Amazon EMR su EKS:

Amazon EMR su EKS ora supporta l'integrazione con SageMaker Unified Studio tramite una soluzione Livy gestita che include:

- Sessioni gestite: nuovo tipo di risorsa di sessione interattiva che fornisce un endpoint HTTPS personalizzato per l'esecuzione di sessioni Spark sul cluster EKS tramite Unified Studio SageMaker

- Lake Formation Integration: supporta il controllo dell'accesso ai dati con due modalità a) Controllo degli accessi granulare b) Accesso completo alla tabella (modalità compatibilità)
- Gestione delle identità: opzioni di autenticazione flessibili a) Controllo degli accessi basato sui ruoli IAM b) controllo degli accessi basato sui ruoli.
- Sessioni utente in background con integrazione: supporta carichi di lavoro Spark di lunga durata per continuare a funzionare anche dopo la disconnessione degli utenti da SageMaker Unified Studio, supportando sessioni fino a 90 giorni

emr-7.11.0-più recente

Note di rilascio: `emr-7.11.0-latest` attualmente indica `emr-7.11.0-20251020`.

Regioni: `emr-7.11.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.11.0:latest`

emr-7.11.0-20251020

Note sulla versione: è stato rilasciato nel novembre 2025. `emr-7.11.0-20251020` Questa è la versione iniziale di Amazon EMR 7.11.0 (Spark).

Regioni: `emr-emr-7.11.0-20251020` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.11.0-20251020`

emr-7.11.0-flink-latest

Note di rilascio: attualmente punta a `emr-7.11.0-flink-latest` `emr-7.11.0-flink-20251020`

Regioni: `emr-7.11.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.11.0-flink:latest`

emr-7.11.0-flink-20251020

Note di rilascio: è stato rilasciato nel novembre 2025. `7.11.0-flink-20251020` Questa è la versione iniziale di Amazon EMR 7.11.0 (Flink).

Regioni: `emr-7.11.0-flink-20251020` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.11.0-flink:20251020`

Amazon EMR nelle versioni EKS 7.10.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.10.0 in generale, consulta Amazon EMR 7.10.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.10

Le seguenti versioni di Amazon EMR 7.10.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.10.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.10.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.10.0-flink-latest
- emr-7.10.0-flink-20250801

Spark releases

Le seguenti versioni di Amazon EMR 7.10.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.10.0-latest
- emr-7.10.0-20250801

- emr-7.10.0-spark-rapids-latest
- emr-7.10.0-spark-rapids-20250801
- emr-7.10.0-java11-latest
- emr-7.10.0-java11-20250801
- emr-7.10.0-java8-latest
- emr-7.10.0-java8-20250801
- emr-7.10.0-spark-rapids-java8-latest
- emr-7.10.0-spark-rapids-java8-20250801
- notebook-spark/emr-7.10.0-latest
- notebook-spark/emr-7.10.0-20250801
- notebook-spark/emr-7.10.0-spark-rapids-latest
- notebook-spark/emr-7.10.0-spark-rapids-20250801
- notebook-spark/emr-7.10.0-java11-latest
- notebook-spark/emr-7.10.0-java11-20250801
- notebook-spark/emr-7.10.0-java8-latest
- notebook-spark/emr-7.10.0-java8-20250801
- notebook-spark/emr-7.10.0-spark-rapids-java8-latest
- notebook-spark/emr-7.10.0-spark-rapids-java8-20250801
- notebook-python/emr-7.10.0-latest
- notebook-python/emr-7.10.0-20250801
- notebook-python/emr-7.10.0-spark-rapids-latest
- notebook-python/emr-7.10.0-spark-rapids-20250801
- notebook-python/emr-7.10.0-java11-latest
- notebook-python/emr-7.10.0-java11-20250801
- notebook-python/emr-7.10.0-java8-latest
- notebook-python/emr-7.10.0-java8-20250801
- notebook-python/emr-7.10.0-spark-rapids-java8-latest
- notebook-python/emr-7.10.0-spark-rapids-java8-20250801

- livy/emr-7.10.0-latest
- livy/emr-7.10.0-20250801
- livy/emr-7.10.0-java11-latest
- livy/emr-7.10.0-java11-20250801
- livy/emr-7.10.0-java8-latest
- livy/emr-7.10.0-java8-20250801

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.10.0:

- Applicazioni supportate - AWS SDK per Java 2.31.48 and 1.12.782, Apache Spark 3.5.5-amzn-1, Apache Hudi 0.15.0-amzn-7, Apache Iceberg 1.8.1-amzn-0, Delta 3.3.0-amzn-2, Apache Spark RAPIDS 25.04.0-amzn-0, Apache Flink 1.20.0-amzn-4, Flink Kubernetes Operator 1.10.0-amzn-4
- Componenti supportati -`emr-ddb`,,`emr-goodies`,`emr-s3-select`,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`. `iceberg` `spark-kubernetes`
- Classificazioni di configurazione supportate

  Da usare con StartJobRune CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .

Classificazioni	Descrizioni
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con `CreateManagedEndpoint` APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Modifiche e funzionalità

Le seguenti funzionalità sono incluse nella versione 7.10.0 di Amazon EMR su EKS:

- Filesystem S3A — A partire dalla versione 7.10.0, il filesystem S3A ha sostituito EMRFS come connettore EMR S3 predefinito. Per ulteriori informazioni, vedere EMR File System (EMRFS).

emr-7.10.0-latest

Note di rilascio: `emr-7.10.0-latest` attualmente indica `emr-7.10.0-20250801`.

Regioni: `emr-7.10.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.10.0:latest`

emr-7.10.0-20250801

Note sulla versione: è stato rilasciato nel febbraio 2025. `emr-7.10.0-20250801` Questa è la versione iniziale di Amazon EMR 7.10.0 (Spark).

Regioni: `emr-emr-7.10.0-20250801` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.10.0-20250801`

emr-7.10.0-flink-latest

Note di rilascio: attualmente punta a `emr-7.10.0-flink-latest` `emr-7.10.0-flink-20250801`

Regioni: `emr-7.10.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.10.0-flink:latest`

emr-7.10.0-flink-20250801

Note di rilascio: è stato rilasciato nel febbraio 2025. `7.10.0-flink-20250801` Questa è la versione iniziale di Amazon EMR 7.10.0 (Flink).

Regioni: `emr-7.10.0-flink-20250801` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.10.0-flink:20250801`

Amazon EMR nelle versioni EKS 7.9.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.9.0 in generale, consulta Amazon EMR 7.9.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.9

Le seguenti versioni di Amazon EMR 7.9.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.9.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.9.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.9.0-flink-latest
- emr-7.9.0-flink-20250425

Spark releases

Le seguenti versioni di Amazon EMR 7.9.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.9.0-più recente
- emr-7.9.0-20250425
- emr-7.9.0-spark-rapids-latest
- emr-7.9.0-spark-rapids-20250425
- emr-7.9.0-java11-latest
- emr-7.9.0-java11-20250425
- emr-7.9.0-java8-latest
- emr-7.9.0-java8-20250425
- emr-7.9.0-spark-rapids-java8-latest
- emr-7.9.0-spark-rapids-java8-20250425
- notebook-spark/emr-7.9.0-latest
- notebook-spark/emr-7.9.0-20250425
- notebook-spark/emr-7.9.0-spark-rapids-latest
- notebook-spark/emr-7.9.0-spark-rapids-20250425
- notebook-spark/emr-7.9.0-java11-latest
- notebook-spark/emr-7.9.0-java11-20250425

- notebook-spark/emr-7.9.0-java8-latest
- notebook-spark/emr-7.9.0-java8-20250425
- notebook-spark/emr-7.9.0-spark-rapids-java8-latest
- notebook-spark/emr-7.9.0-spark-rapids-java8-20250425
- notebook-python/emr-7.9.0-latest
- notebook-python/emr-7.9.0-20250425
- notebook-python/emr-7.9.0-spark-rapids-latest
- notebook-python/emr-7.9.0-spark-rapids-20250425
- notebook-python/emr-7.9.0-java11-latest
- notebook-python/emr-7.9.0-java11-20250425
- notebook-python/emr-7.9.0-java8-latest
- notebook-python/emr-7.9.0-java8-20250425
- notebook-python/emr-7.9.0-spark-rapids-java8-latest
- notebook-python/emr-7.9.0-spark-rapids-java8-20250425
- livy/emr-7.9.0-latest
- livy/emr-7.9.0-20250425
- livy/emr-7.9.0-java11-latest
- livy/emr-7.9.0-java11-20250425
- livy/emr-7.9.0-java8-latest
- livy/emr-7.9.0-java8-20250425

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.9.0

- Applicazioni supportate - AWS SDK per Java 2.31.16 and 1.12.782, Apache Spark 3.5.5, Apache Hudi 0.15.0-amzn-6, Apache Iceberg 1.7.1-amzn-2, Delta 3.3.0-amzn-1, Apache Spark RAPIDS 25.02.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-3, Flink Operator 1.10.0-amzn-3
- Componenti supportati -`emr-ddb`,,`emr-goodies`,`emr-s3-select`,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`. `iceberg` `spark-kubernetes`
- Classificazioni di configurazione supportate

Da usare con StartJobRune CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
core-site	Modifica i valori nel file Hadoop core-site.xml .
emrfs-site	Modifica le impostazioni EMRFS.
spark-metrics	Modifica i valori nel file Spark metrics.properties .
spark-defaults	Modifica i valori nel file Spark spark-defaults.conf .
spark-env	Modifica i valori nell'ambiente Spark.
spark-hive-site	Modifica i valori nel file Spark hive-site.xml .
spark-log4j2	Modifica i valori nel file Spark log4j2.properties .
emr-job-submitter	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
jeg-config	Modifica i valori nel file jupyter_enterprise_gateway_config.py Jupyter Enterprise Gateway.
jupyter-kernel-overrides	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Modifiche

Le seguenti modifiche sono incluse nella versione 7.9.0 di Amazon EMR su EKS:

- Nessuna modifica per la versione.

emr-7.9.0-più recente

Note di rilascio: `emr-7.9.0-latest` attualmente indica `emr-7.9.0-20250425`.

Regioni: `emr-7.9.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.9.0:latest`

emr-7.9.0-20250425

Note sulla versione: è stato rilasciato nel febbraio 2025. `emr-7.9.0-20250425` Questa è la versione iniziale di Amazon EMR 7.9.0 (Spark).

Regioni: `emr-emr-7.9.0-20250425` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.9.0-20250425`

emr-7.9.0-flink-latest

Note di rilascio: attualmente punta a `emr-7.9.0-flink-latest` `emr-7.9.0-flink-20250425`

Regioni: `emr-7.9.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.9.0-flink:latest`

emr-7.9.0-flink-20250425

Note di rilascio: è stato rilasciato nel febbraio 2025. `7.9.0-flink-20250425` Questa è la versione iniziale di Amazon EMR 7.9.0 (Flink).

Regioni: `emr-7.9.0-flink-20250425` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.9.0-flink:20250425`

Amazon EMR nelle versioni EKS 7.8.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.8.0 in generale, consulta Amazon EMR 7.8.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.8

Le seguenti versioni di Amazon EMR 7.8.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.8.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.8.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.8.0-flink-latest
- emr-7.8.0-flink-20250228

Spark releases

Le seguenti versioni di Amazon EMR 7.8.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.8.0-più recente
- emr-7.8.0-20250228

- emr-7.8.0-spark-rapids-latest
- emr-7.8.0-spark-rapids-20250228
- emr-7.8.0-java11-latest
- emr-7.8.0-java11-20250228
- emr-7.8.0-java8-latest
- emr-7.8.0-java8-20250228
- emr-7.8.0-spark-rapids-java8-latest
- emr-7.8.0-spark-rapids-java8-20250228
- notebook-spark/emr-7.8.0-latest
- notebook-spark/emr-7.8.0-20250228
- notebook-spark/emr-7.8.0-spark-rapids-latest
- notebook-spark/emr-7.8.0-spark-rapids-20250228
- notebook-spark/emr-7.8.0-java11-latest
- notebook-spark/emr-7.8.0-java11-20250228
- notebook-spark/emr-7.8.0-java8-latest
- notebook-spark/emr-7.8.0-java8-20250228
- notebook-spark/emr-7.8.0-spark-rapids-java8-latest
- notebook-spark/emr-7.8.0-spark-rapids-java8-20250228
- notebook-python/emr-7.8.0-latest
- notebook-python/emr-7.8.0-20250228
- notebook-python/emr-7.8.0-spark-rapids-latest
- notebook-python/emr-7.8.0-spark-rapids-20250228
- notebook-python/emr-7.8.0-java11-latest
- notebook-python/emr-7.8.0-java11-20250228
- notebook-python/emr-7.8.0-java8-latest
- notebook-python/emr-7.8.0-java8-20250228
- notebook-python/emr-7.8.0-spark-rapids-java8-latest
- notebook-python/emr-7.8.0-spark-rapids-java8-20250228
- livy/emr-7.8.0-latest

- livy/emr-7.8.0-20250228
- livy/emr-7.8.0-java11-latest
- livy/emr-7.8.0-java11-20250228
- livy/emr-7.8.0-java8-latest
- livy/emr-7.8.0-java8-20250228

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.8.0

- Applicazioni supportate - AWS SDK per Java 2.29.52 and 1.12.780, Apache Spark 3.5.4, Apache Hudi 0.15.0-amzn-5, Apache Iceberg 1.7.1-amzn-1, Delta 3.3.0-amzn-0, Apache Spark RAPIDS 24.12.0-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-2, Flink Operator 1.10.0-amzn-2
- Componenti supportati -`emr-ddb`,,`emr-goodies`,`emr-s3-select`,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`. `iceberg` `spark-kubernetes`
- Classificazioni di configurazione supportate

  Da usare con StartJobRune CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .

Classificazioni	Descrizioni
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Modifiche

Le seguenti modifiche sono incluse nella versione 7.8.0 di Amazon EMR su EKS:

- Funzionalità FGAC native, tra cui:
  - Supporto Iceberg per l'esecuzione di lavori che eseguono azioni su Non-Lake Formation Tables in un cluster virtuale FGAC (controllo degli accessi a grana fine). (Esiste un fallback su IAM.)
  - Supporto per tabelle S3
- Connessione Spark

emr-7.8.0-più recente

Note di rilascio: `emr-7.8.0-latest` attualmente indica `emr-7.8.0-20250228`.

Regioni: `emr-7.8.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.8.0:latest`

emr-7.8.0-20250228

Note di rilascio: è stato rilasciato nel febbraio 2025. `emr-7.8.0-20250228` Questa è la versione iniziale di Amazon EMR 7.8.0 (Spark).

Regioni: `emr-emr-7.8.0-20250228` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.8.0-20250228`

emr-7.8.0-flink-latest

Note di rilascio: attualmente punta a `emr-7.8.0-flink-latest` `emr-7.8.0-flink-20250228`

Regioni: `emr-7.8.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.8.0-flink:latest`

emr-7.8.0-flink-20250228

Note di rilascio: è stato rilasciato nel febbraio 2025. `7.8.0-flink-20250228` Questa è la versione iniziale di Amazon EMR 7.8.0 (Flink).

Regioni: `emr-7.8.0-flink-20250228` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.8.0-flink:20250228`

Amazon EMR nelle versioni EKS 7.7.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon

EC2 e sulla versione Amazon EMR 7.7.0 in generale, consulta Amazon EMR 7.7.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.7

Le seguenti versioni di Amazon EMR 7.7.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.7.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.7.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.7.0-flink-latest
- emr-7.7.0-flink-20250131

Spark releases

Le seguenti versioni di Amazon EMR 7.7.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.7.0-più recente
- emr-7.7.0-20250131
- emr-7.7.0-spark-rapids-latest
- emr-7.7.0-spark-rapids-20250131
- emr-7.7.0-java11-latest
- emr-7.7.0-java11-20250131
- emr-7.7.0-java8-latest
- emr-7.7.0-java8-20250131
- emr-7.7.0-spark-rapids-java8-latest
- emr-7.7.0-spark-rapids-java8-20250131
- notebook-spark/emr-7.7.0-latest
- notebook-spark/emr-7.7.0-20250131
- notebook-spark/emr-7.7.0-spark-rapids-latest

- notebook-spark/emr-7.7.0-spark-rapids-20250131
- notebook-spark/emr-7.7.0-java11-latest
- notebook-spark/emr-7.7.0-java11-20250131
- notebook-spark/emr-7.7.0-java8-latest
- notebook-spark/emr-7.7.0-java8-20250131
- notebook-spark/emr-7.7.0-spark-rapids-java8-latest
- notebook-spark/emr-7.7.0-spark-rapids-java8-20250131
- notebook-python/emr-7.7.0-latest
- notebook-python/emr-7.7.0-20250131
- notebook-python/emr-7.7.0-spark-rapids-latest
- notebook-python/emr-7.7.0-spark-rapids-20250131
- notebook-python/emr-7.7.0-java11-latest
- notebook-python/emr-7.7.0-java11-20250131
- notebook-python/emr-7.7.0-java8-latest
- notebook-python/emr-7.7.0-java8-20250131
- notebook-python/emr-7.7.0-spark-rapids-java8-latest
- notebook-python/emr-7.7.0-spark-rapids-java8-20250131
- livy/emr-7.7.0-latest
- livy/emr-7.7.0-20250131
- livy/emr-7.7.0-java11-latest
- livy/emr-7.7.0-java11-20250131
- livy/emr-7.7.0-java8-latest
- livy/emr-7.7.0-java8-20250131

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.7.0

- Applicazioni supportate - AWS SDK per Java 2.29.25 and 1.12.779, Apache Spark 3.5.3-amzn-0, Apache Hudi 0.15.0-amzn-3, Apache Iceberg 1.6.1-amzn-2, Delta 3.2.1-amzn-1, Apache Spark RAPIDS 24.10.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-0, Flink Operator 1.10.0-amzn-0

- Componenti supportati -`emr-ddb`,,`emr-goodies`,`emr-s3-select`,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`. `iceberg` `spark-kubernetes`
- Classificazioni di configurazione supportate

  Da usare con StartJobRune CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

  Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.

Classificazioni	Descrizioni
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Modifiche

Le seguenti modifiche sono incluse nella versione 7.7.0 di Amazon EMR su EKS:

- La versione Iceberg in uso a partire da EMR 7.7.0 non supporta più Java 8. Inoltre, Iceberg è escluso dalle seguenti immagini Java 8: e. `emr-7.7.0-java8-latest` `emr-7.7.0-spark-rapids-java8-latest`

emr-7.7.0-più recente

Note di rilascio: `emr-7.7.0-latest` attualmente indica `emr-7.7.0-20250131`.

Regioni: `emr-7.7.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.7.0:latest`

emr-7.7.0-20250131

Note di rilascio: è stato rilasciato nel febbraio 2025. `emr-7.7.0-20250131` Questa è la versione iniziale di Amazon EMR 7.7.0 (Spark).

Regioni: `emr-emr-7.7.0-20250131` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.7.0-20250131`

emr-7.7.0-flink-latest

Note di rilascio: attualmente punta a `emr-7.7.0-flink-latest` `emr-7.7.0-flink-20250131`

Regioni: `emr-7.7.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.7.0-flink:latest`

emr-7.7.0-flink-20250131

Note di rilascio: è stato rilasciato nel febbraio 2025. `7.7.0-flink-20250131` Questa è la versione iniziale di Amazon EMR 7.7.0 (Flink).

Regioni: `emr-7.7.0-flink-20250131` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.7.0-flink:20250131`

Amazon EMR nelle versioni EKS 7.6.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.6.0 in generale, consulta Amazon EMR 7.6.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.6

Le seguenti versioni di Amazon EMR 7.6.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.6.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.6.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.6.0-flink-latest
- emr-7.6.0-flink-20241213

Spark releases

Le seguenti versioni di Amazon EMR 7.6.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.6.0-più recente
- emr-7.6.0-20241213
- emr-7.6.0-spark-rapids-latest
- emr-7.6.0-spark-rapids-20241213
- emr-7.6.0-java11-latest
- emr-7.6.0-java11-20241213
- emr-7.6.0-java8-latest
- emr-7.6.0-java8-20241213
- emr-7.6.0-spark-rapids-java8-latest
- emr-7.6.0-spark-rapids-java8-20241213
- notebook-spark/emr-7.6.0-latest
- notebook-spark/emr-7.6.0-20241213
- notebook-spark/emr-7.6.0-spark-rapids-latest
- notebook-spark/emr-7.6.0-spark-rapids-20241213
- notebook-spark/emr-7.6.0-java11-latest
- notebook-spark/emr-7.6.0-java11-20241213
- notebook-spark/emr-7.6.0-java8-latest
- notebook-spark/emr-7.6.0-java8-20241213
- notebook-spark/emr-7.6.0-spark-rapids-java8-latest
- notebook-spark/emr-7.6.0-spark-rapids-java8-20241213
- notebook-python/emr-7.6.0-latest
- notebook-python/emr-7.6.0-20241213
- notebook-python/emr-7.6.0-spark-rapids-latest
- notebook-python/emr-7.6.0-spark-rapids-20241213
- notebook-python/emr-7.6.0-java11-latest
- notebook-python/emr-7.6.0-java11-20241213
- notebook-python/emr-7.6.0-java8-latest
- notebook-python/emr-7.6.0-java8-20241213
- notebook-python/emr-7.6.0-spark-rapids-java8-latest

- notebook-python/emr-7.6.0-spark-rapids-java8-20241213
- livy/emr-7.6.0-latest
- livy/emr-7.6.0-20241213
- livy/emr-7.6.0-java11-latest
- livy/emr-7.6.0-java11-20241213
- livy/emr-7.6.0-java8-latest
- livy/emr-7.6.0-java8-20241213

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.6.0

- Applicazioni supportate - AWS SDK per Java 2.29.25 and 1.12.779, Apache Spark 3.5.3-amzn-0, Apache Hudi 0.15.0-amzn-3, Apache Iceberg 1.6.1-amzn-2, Delta 3.2.1-amzn-1, Apache Spark RAPIDS 24.10.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-0, Flink Operator 1.10.0-amzn-0
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  Da utilizzare con e: StartJobRun CreateManagedEndpoint APIs

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.

Classificazioni	Descrizioni
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Le seguenti funzionalità sono incluse nella versione 7.6.0 di Amazon EMR su EKS:

- Supporto alla configurazione di monitoraggio per Apache Spark Operator: la configurazione di monitoraggio consente di configurare facilmente l'archiviazione dei log dell'applicazione Spark e dei log dell'operatore su Amazon S3 o Amazon. CloudWatch Puoi sceglierne uno o entrambi. In questo modo viene aggiunto un sidecar di log agent ai pod operatore, driver ed executor di Spark e successivamente inoltra i log di questi componenti ai sink configurati. Per maggiori informazioni,

consulta Usare la configurazione di monitoraggio per monitorare l'operatore Spark Kubernetes e i job Spark.

Modifiche

Le seguenti modifiche sono incluse nella versione 7.6.0 di Amazon EMR su EKS:

- Nessuna modifica per la versione.

emr-7.6.0-più recente

Note di rilascio: `emr-7.6.0-latest` attualmente indica `emr-7.6.0-20241213`.

Regioni: `emr-7.6.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.6.0:latest`

emr-7.6.0-20241213

Note di rilascio: è stato rilasciato a gennaio 2024. `7.6.0-20241213` Questa è la versione iniziale di Amazon EMR 7.6.0 (Spark).

Regioni: `emr-7.6.0-20241213` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.6.0:20241213`

emr-7.6.0-flink-latest

Note di rilascio: attualmente punta a `emr-7.6.0-flink-latest` `emr-7.6.0-flink-20241213`

Regioni: `emr-7.6.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.6.0-flink:latest`

emr-7.6.0-flink-20241213

Note di rilascio: è stato rilasciato nel gennaio 2024. `7.6.0-flink-20241213` Questa è la versione iniziale di Amazon EMR 7.6.0 (Flink).

Regioni: `emr-7.6.0-flink-20241213` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.6.0-flink:20241213`

Amazon EMR nelle versioni EKS 7.5.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.5.0 in generale, consulta Amazon EMR 7.5.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.5

Le seguenti versioni di Amazon EMR 7.5.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.5.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.5.0

- Applicazioni supportate - AWS SDK per Java 2.28.8 and 1.12.772, Apache Spark 3.5.2-amzn-1, Apache Hudi 0.15.0-amzn-1, Apache Iceberg 1.6.1-amzn-0, Delta 3.2.0-amzn-1, Apache Spark RAPIDS 24.08.1-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.19.1-amzn-1, Flink Operator 1.9.0-amzn-0
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.

Amazon EMR nelle versioni EKS 7.4.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.4.0 in generale, consulta Amazon EMR 7.4.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.4

Le seguenti versioni di Amazon EMR 7.4.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.4.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.4.0

- Applicazioni supportate - AWS SDK per Java 2.25.70 and 1.12.772, Apache Spark 3.5.2-amzn-0, Apache Hudi 0.15.0-amzn-1, Apache Iceberg 1.6.1-amzn-0, Delta 3.2.0-amzn-1, Apache Spark RAPIDS 24.08.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.19.1-amzn-0, Flink Operator 1.9.0-amzn-1
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.

Amazon EMR nelle versioni EKS 7.3.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.3.0 in generale, consulta Amazon EMR 7.3.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.3

Le seguenti versioni di Amazon EMR 7.3.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.3.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.3.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.3.0-flink-latest
- emr-7.3.0-flink-29240920

Spark releases

Le seguenti versioni di Amazon EMR 7.3.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.3.0-più recente
- emr-7.3.0-20240920
- emr-7.3.0-spark-rapids-latest
- emr-7.3.0-spark-rapids-29240920
- emr-7.3.0-java11-latest
- emr-7.3.0-java11-29240920
- emr-7.3.0-java8-latest
- emr-7.3.0-java8-29240920
- emr-7.3.0-spark-rapids-java8-latest
- emr-7.3.0-spark-rapids-java8-29240920
- notebook-spark/emr-7.3.0-latest
- notebook-spark/emr-7.3.0-20240920
- notebook-spark/emr-7.3.0-spark-rapids-latest
- notebook-spark/emr-7.3.0-spark-rapids-29240920
- notebook-spark/emr-7.3.0-java11-latest
- notebook-spark/emr-7.3.0-java11-29240920
- notebook-spark/emr-7.3.0-java8-latest
- notebook-spark/emr-7.3.0-java8-29240920
- notebook-spark/emr-7.3.0-spark-rapids-java8-latest
- notebook-spark/emr-7.3.0-spark-rapids-java8-29240920
- notebook-python/emr-7.3.0-latest
- notebook-python/emr-7.3.0-20240920
- notebook-python/emr-7.3.0-spark-rapids-latest
- notebook-python/emr-7.3.0-spark-rapids-29240920
- notebook-python/emr-7.3.0-java11-latest
- notebook-python/emr-7.3.0-java11-29240920

- notebook-python/emr-7.3.0-java8-latest
- notebook-python/emr-7.3.0-java8-29240920
- notebook-python/emr-7.3.0-spark-rapids-java8-latest
- notebook-python/emr-7.3.0-spark-rapids-java8-29240920
- livy/emr-7.3.0-latest
- livy/emr-7.3.0-20240920
- livy/emr-7.3.0-java11-latest
- livy/emr-7.3.0-java11-29240920
- livy/emr-7.3.0-java8-latest
- livy/emr-7.3.0-java8-29240920

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.3.0

- Applicazioni supportate - AWS SDK per Java 2.25.70 and 1.12.747, Apache Spark 3.5.1-amzn-1, Apache Hudi 0.15.0-amzn-0, Apache Iceberg 1.5.2-amzn-0, Delta 3.2.0-amzn-0, Apache Spark RAPIDS 24.06.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-2, Flink Operator 1.9.0-amzn-0
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  Da utilizzare con StartJobRune: CreateManagedEndpoint APIs

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .

Classificazioni	Descrizioni
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Le seguenti funzionalità sono incluse nella versione 7.3.0 di Amazon EMR su EKS.

- Aggiornamenti delle applicazioni: Amazon EMR su EKS ora include Flink Operator 1.9.0. Oltre ad altre funzionalità, Flink Kubernetes ora consente di impostare quote di CPU e memoria per l'autoscaler.
- Supporto Apache Iceberg per Apache Flink — Apache Iceberg è un formato open source ad alte prestazioni per grandi tabelle analitiche. A partire da Amazon EMR 7.3.0, puoi utilizzare le tabelle Apache Iceberg quando esegui Apache Flink su Amazon EMR su EKS. Per ulteriori informazioni, consulta Amazon EMR su EKS Using Apache Iceberg with Amazon EMR on EKS.
- Supporto Delta Lake per Apache Flink: Delta Lake è un framework a livello di storage per architetture Lakehouse comunemente costruito su Amazon S3. Con Amazon EMR 7.3.0 e versioni successive, puoi usare le tabelle Delta quando esegui Apache Flink su Amazon EMR su EKS. Per ulteriori informazioni, consulta Usare Delta Lake con Amazon EMR su EKS.

Modifiche

Le seguenti modifiche sono incluse nella versione 7.3.0 di Amazon EMR su EKS.

- Con Amazon EMR su EKS 7.3.0 e versioni successive, Apache Flink ora utilizza il runtime Java 17 per impostazione predefinita.

emr-7.3.0-più recente

Note di rilascio: `emr-7.3.0-latest` attualmente indica `emr-7.3.0-20240920`.

Regioni: `emr-7.3.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.3.0:latest`

emr-7.3.0-20240920

Note di rilascio: `7.3.0-20240920` è stato rilasciato nel mese di dicembre 2023. Questa è la versione iniziale di Amazon EMR 7.3.0 (Spark).

Regioni: `emr-7.3.0-20240920` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.3.0:29240920`

emr-7.3.0-flink-latest

Note di rilascio: `emr-7.3.0-flink-latest` attualmente indica `emr-7.3.0-flink-29240920`.

Regioni: `emr-7.3.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.3.0-flink:latest`

emr-7.3.0-flink-29240920

Note di rilascio: `7.3.0-flink-29240920` è stato rilasciato nel mese di dicembre 2023. Questa è la versione iniziale di Amazon EMR 7.3.0 (Flink).

Regioni: `emr-7.3.0-flink-29240920` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.3.0-flink:29240920`

Amazon EMR nelle versioni EKS 7.2.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.2.0 in generale, consulta Amazon EMR 7.2.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.2

Le seguenti versioni di Amazon EMR 7.2.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.2.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.2.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.2.0-flink-latest
- emr-7.2.0-flink-20240610

Spark releases

Le seguenti versioni di Amazon EMR 7.2.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.2.0-più recente
- emr-7.2.0-20240610
- emr-7.2.0-spark-rapids-latest
- emr-7.2.0-spark-rapids-20240610
- emr-7.2.0-java11-latest
- emr-7.2.0-java11-20240610
- emr-7.2.0-java8-latest
- emr-7.2.0-java8-20240610
- emr-7.2.0-spark-rapids-java8-latest
- emr-7.2.0-spark-rapids-java8-20240610
- notebook-spark/emr-7.2.0-latest
- notebook-spark/emr-7.2.0-20240610
- notebook-spark/emr-7.2.0-spark-rapids-latest
- notebook-spark/emr-7.2.0-spark-rapids-20240610
- notebook-spark/emr-7.2.0-java11-latest
- notebook-spark/emr-7.2.0-java11-20240610
- notebook-spark/emr-7.2.0-java8-latest
- notebook-spark/emr-7.2.0-java8-20240610
- notebook-spark/emr-7.2.0-spark-rapids-java8-latest
- notebook-spark/emr-7.2.0-spark-rapids-java8-20240610
- notebook-python/emr-7.2.0-latest
- notebook-python/emr-7.2.0-20240610
- notebook-python/emr-7.2.0-spark-rapids-latest
- notebook-python/emr-7.2.0-spark-rapids-20240610
- notebook-python/emr-7.2.0-java11-latest
- notebook-python/emr-7.2.0-java11-20240610

- notebook-python/emr-7.2.0-java8-latest
- notebook-python/emr-7.2.0-java8-20240610
- notebook-python/emr-7.2.0-spark-rapids-java8-latest
- notebook-python/emr-7.2.0-spark-rapids-java8-20240610
- livy/emr-7.2.0-latest
- livy/emr-7.2.0-20240610
- livy/emr-7.2.0-java11-latest
- livy/emr-7.2.0-java11-20240610
- livy/emr-7.2.0-java8-latest
- livy/emr-7.2.0-java8-20240610

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.2.0

- Applicazioni supportate - AWS SDK per Java 2.23.18 and 1.12.705, Apache Spark 3.5.1-amzn-1, Apache Hudi 0.14.1-amzn-0, Apache Iceberg 1.5.0-amzn-0, Delta 3.1.0, Apache Spark RAPIDS 24.02.0-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-0, Flink Operator 1.8.0-amzn-1
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  Da utilizzare con StartJobRune: CreateManagedEndpoint APIs

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .

Classificazioni	Descrizioni
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Le seguenti funzionalità sono incluse nella versione 7.2.0 di Amazon EMR su EKS.

- Aggiornamenti delle applicazioni: gli aggiornamenti delle applicazioni Amazon EMR su EKS 7.2.0 includono Spark 3.5.1, Flink 1.18.1 e Flink Operator 1.8.0.

- Aggiornamenti di Autoscaler for Flink: la versione 7.2.0 utilizza la configurazione open source per consentire la stima del tempo di ridimensionamento, in modo da non dover più assegnare manualmente valori empirici `job.autoscaler.restart.time-tracking.enabled` all'ora di riavvio. Se utilizzi la versione 7.1.0 o una versione precedente, puoi comunque utilizzare Amazon EMR autoscaling.
- Integrazione con Apache Hudi Apache Flink su Amazon EMR su EKS: questa versione aggiunge un'integrazione tra Apache Hudi e Apache Flink, in modo da poter utilizzare l'operatore Flink Kubernetes per eseguire job Hudi. Hudi ti consente di utilizzare operazioni a livello di record che puoi utilizzare per semplificare la gestione dei dati e lo sviluppo di pipeline di dati.
- Integrazione di Amazon S3 Express One Zone con Amazon EMR su EKS: con la versione 7.2.0 e versioni successive, puoi caricare dati in S3 Express One Zone con Amazon EMR su EKS. S3 Express One Zone è una classe di storage Amazon S3 a zona singola ad alte prestazioni che offre un accesso ai dati coerente a una cifra in millisecondi per la maggior parte delle applicazioni sensibili alla latenza. Al momento del suo rilascio, S3 Express One Zone offre lo storage di oggetti cloud con la latenza più bassa e le prestazioni più elevate in Amazon S3.
- Supporto per le configurazioni predefinite nell'operatore Spark — L'operatore Spark su Amazon EKS ora supporta le stesse configurazioni predefinite del modello start job run su Amazon EMR su EKS per 7.2.0 e versioni successive. Ciò significa che funzionalità come Amazon S3 ed EMRFS non richiedono più configurazioni manuali nel file yaml.

emr-7.2.0-più recente

Note di rilascio: `emr-7.2.0-latest` attualmente indica `emr-7.2.0-20240610`.

Regioni: `emr-7.2.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.2.0:latest`

emr-7.2.0-20240610

Note di rilascio: `7.2.0-20240610` è stato rilasciato nel mese di dicembre 2023. Questa è la versione iniziale di Amazon EMR 7.2.0 (Spark).

Regioni: `emr-7.2.0-20240610` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.2.0:20240610`

emr-7.2.0-flink-latest

Note di rilascio: `emr-7.2.0-flink-latest` attualmente indica `emr-7.2.0-flink-20240610`.

Regioni: `emr-7.2.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.2.0-flink:latest`

emr-7.2.0-flink-20240610

Note di rilascio: `7.2.0-flink-20240610` è stato rilasciato nel mese di dicembre 2023. Questa è la versione iniziale di Amazon EMR 7.2.0 (Flink).

Regioni: `emr-7.2.0-flink-20240610` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.2.0-flink:20240610`

Amazon EMR nelle versioni EKS 7.1.0

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.1.0 in generale, consulta Amazon EMR 7.1.0 nella Amazon EMR Release Guide.

Amazon EMR nelle versioni EKS 7.1

Le seguenti versioni di Amazon EMR 7.1.0 sono disponibili per Amazon EMR su EKS. Seleziona una versione specifica di EMR-7.1.0-xxxx per visualizzare ulteriori dettagli, come il relativo tag di immagine del contenitore.

Flink releases

Le seguenti versioni di Amazon EMR 7.1.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Flink.

- emr-7.1.0-flink-latest
- emr-7.1.0-flink-20240321

Spark releases

Le seguenti versioni di Amazon EMR 7.1.0 sono disponibili per Amazon EMR su EKS quando esegui applicazioni Spark.

- emr-7.1.0-più recente
- 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

Note di rilascio

Note di rilascio per Amazon EMR su EKS 7.1.0

- Applicazioni supportate - AWS SDK per 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
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  Da utilizzare con StartJobRune: CreateManagedEndpoint APIs

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .

Classificazioni	Descrizioni
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j2`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Le seguenti funzionalità sono incluse nella versione 7.1.0 di Amazon EMR su EKS.

- Supporto Apache Livy per Amazon EMR su EKS: con le versioni 7.1.0 e successive di Amazon EMR su EKS, puoi utilizzare Apache Livy su un cluster Amazon EKS per creare un'interfaccia

REST di Apache Livy per inviare lavori Spark o frammenti di codice Spark. In questo modo puoi recuperare i risultati in modo sincrono e asincrono, sfruttando al contempo i vantaggi di Amazon EMR su EKS, come il runtime Spark ottimizzato per Amazon EMR, gli endpoint Livy abilitati per SSL e un'esperienza di configurazione programmatica.

emr-7.1.0-più recente

Note di rilascio: `emr-7.1.0-latest` attualmente indica `emr-7.1.0-20240321`.

Regioni: `emr-7.1.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.1.0:latest`

emr-7.1.0-20240321

Note di rilascio: `7.1.0-20240321` è stato rilasciato nel mese di dicembre 2023. Questa è la versione iniziale di Amazon EMR 7.1.0 (Spark).

Regioni: `emr-7.1.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.1.0:20240321`

emr-7.1.0-flink-latest

Note di rilascio: `emr-7.1.0-flink-latest` attualmente indica `emr-7.1.0-flink-20240321`.

Regioni: `emr-7.1.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.1.0-flink:latest`

emr-7.1.0-flink-20240321

Note di rilascio: `7.1.0-flink-20240321` è stato rilasciato nel mese di dicembre 2023. Questa è la versione iniziale di Amazon EMR 7.1.0 (Flink).

Regioni: `emr-7.1.0-flink-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.1.0-flink:20240321`

Rilasci 7.0.0 di Amazon EMR su EKS

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 7.0.0 in generale, consulta Amazon EMR 7.0.0 nella Amazon EMR Release Guide.

Rilasci 7.0 di Amazon EMR su EKS

I seguenti rilasci 7.0.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-7.0.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

Flink releases

I seguenti rilasci 7.0.0 di Amazon EMR sono disponibili per Amazon EMR su EKS quando vengono eseguite le applicazioni Flink.

- emr-7.0.0-flink-latest
- emr-7.0.0-flink-2024321
- emr-7.0.0-flink-20231211

Spark releases

I seguenti rilasci 7.0.0 di Amazon EMR sono disponibili per Amazon EMR su EKS quando vengono eseguite le applicazioni Spark.

- 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

Note di rilascio

Note di rilascio di Amazon EMR su EKS 7.0.0

- Applicazioni supportate - AWS SDK per 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
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.

- Classificazioni di configurazione supportate

  Da usare con e: StartJobRun CreateManagedEndpoint APIs

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

  Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Nel rilascio 7.0 di Amazon EMR su EKS sono incluse le funzionalità elencate di seguito.

- Aggiornamenti dell'applicazione: gli aggiornamenti dell'applicazione Amazon EMR su EKS 7.0.0 includono Spark 3.5, Flink 1.18 e Flink Operator 1.6.1.
- Regolazione automatica dei parametri Flink Autoscaler: i parametri predefiniti utilizzati da Flink Autoscaler per i calcoli di dimensionamento potrebbero non essere il valore ottimale per un determinato processo. Amazon EMR su EKS 7.0.0 utilizza le tendenze storiche di specifiche metriche acquisite per calcolare il parametro ottimale personalizzato per il processo.

Modifiche

Nel rilascio 7.0 di Amazon EMR su EKS sono incluse le modifiche elencate di seguito.

- Amazon Linux 2023: con Amazon EMR su EKS 7.0.0 e rilasci successivi, tutte le immagini dei container sono basate su Amazon Linux 2023.
- Spark utilizza Java 17 come runtime predefinito: Amazon EMR su EKS 7.0.0 Spark utilizza Java 17 come runtime predefinito. Se necessario, puoi passare a utilizzare Java 8 o Java 11 con l'etichetta di rilascio corrispondente, come indicato nell'elenco Rilasci 7.0 di Amazon EMR su EKS.

emr-7.0.0-latest

Note di rilascio: `emr-7.0.0-latest` attualmente indica `emr-7.0.0-2024321`.

Regioni: `emr-7.0.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.0.0:latest`

emr-7.0.0-2024321

Note sulla versione: `7.0.0-2024321` è stato rilasciato l'11 marzo 2024. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-7.0.0-2024321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.0.0:2024321`

emr-7.0.0-20231211

Note di rilascio: `7.0.0-20231211` è stato rilasciato nel mese di dicembre 2023. Questo è il rilascio iniziale di Amazon EMR 7.0.0 (Spark).

Regioni: `emr-7.0.0-20231211` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.0.0:20231211`

emr-7.0.0-flink-latest

Note di rilascio: `emr-7.0.0-flink-latest` attualmente indica `emr-7.0.0-flink-2024321`.

Regioni: `emr-7.0.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.0.0-flink:latest`

emr-7.0.0-flink-2024321

Note di rilascio: è stato rilasciato l'11 marzo 2024. `7.0.0-flink-2024321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-7.0.0-flink-2024321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.0.0-flink:2024321`

emr-7.0.0-flink-20231211

Note di rilascio: `7.0.0-flink-20231211` è stato rilasciato nel mese di dicembre 2023. Questo è il rilascio iniziale di Amazon EMR 7.0.0 (Flink).

Regioni: `emr-7.0.0-flink-20231211` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-7.0.0-flink:20231211`

Rilasci 6.15.0 di Amazon EMR su EKS

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 6.15.0 in generale, consulta Amazon EMR 6.15.0 nella Amazon EMR Release Guide.

Rilasci 6.15 di Amazon EMR su EKS

I seguenti rilasci 6.15.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.15.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

Flink releases

I seguenti rilasci 6.15.0 di Amazon EMR sono disponibili per Amazon EMR su EKS quando vengono eseguite le applicazioni Flink.

- emr-6.15.0-flink-latest
- emr-6.15.0-flink-20240105
- emr-6.15.0-flink-20231109

Spark releases

I seguenti rilasci 6.15.0 di Amazon EMR sono disponibili per Amazon EMR su EKS quando vengono eseguite le applicazioni Spark.

- 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

Note di rilascio

Note di rilascio di Amazon EMR su EKS 6.15.0

- Applicazioni supportate - AWS SDK per 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
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  Da usare con e: StartJobRun CreateManagedEndpoint APIs

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j`	Modifica i valori nel file Spark `log4j2.properties` .

Classificazioni	Descrizioni
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Nel rilascio 6.15 di Amazon EMR su EKS sono incluse le funzionalità elencate di seguito.

- Amazon EMR su EKS con Apache Flink: con Amazon EMR su EKS 6.15.0, puoi eseguire la tua applicazione basata su Apache Flink insieme ad altri tipi di applicazioni sullo stesso cluster Amazon EKS. Ciò consente di migliorare l'utilizzo delle risorse e di semplificare la gestione dell'infrastruttura. Puoi sfruttare le istanze spot in un'applicazione Flink con una disattivazione graduale e ottenere tempi di riavvio più rapidi con il ripristino granulare e il ripristino locale delle attività con Amazon EBS. Le funzionalità di accessibilità e monitoraggio includono la possibilità di avviare un'applicazione Flink con jar archiviati in Amazon S3, l'accesso al AWS Glue Data Catalog, il monitoraggio dell'integrazione con Amazon S3 e Amazon CloudWatch e la rotazione dei log dei container.

emr-6.15.0-latest

Note di rilascio: `emr-6.15.0-latest` attualmente indica `emr-6.15.0-20240105`.

Regioni: `emr-6.15.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.15.0:latest`

emr-6.15.0-20240105

Note di rilascio: `6.15.0-20240105` è stato rilasciato il 17 gennaio 2024. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.15.0-20240105` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.15.0:20240105`

emr-6.15.0-20231109

Note di rilascio: `6.15.0-20231109` è stato rilasciato il 17 novembre 2023. Questo è il rilascio iniziale di Amazon EMR 6.15.0.

Regioni: `emr-6.15.0-20231109` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.15.0:20231109`

emr-6.15.0-flink-latest

Note di rilascio: `emr-6.15.0-flink-latest` attualmente indica `emr-6.15.0-flink-20240105`.

Regioni: `emr-6.15.0-flink-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.15.0-flink:latest`

emr-6.15.0-flink-20240105

Note di rilascio: è stato rilasciato il 17 gennaio 2024. `6.15.0-flink-20240105` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.15.0-flink-20240105` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.15.0-flink:20240105`

emr-6.15.0-flink-20231109

Note di rilascio: `6.15.0-flink-20231109` è stato rilasciato il 17 novembre 2023. Questo è il rilascio iniziale di Amazon EMR 6.15.0.

Regioni: `emr-6.15.0-flink-20231109` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.15.0-flink:20231109`

Rilasci 6.14.0 di Amazon EMR su EKS

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 6.14.0 in generale, consulta Amazon EMR 6.14.0 nella Amazon EMR Release Guide.

Rilasci 6.14 di Amazon EMR su EKS

I seguenti rilasci 6.14.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.14.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio

Note di rilascio di Amazon EMR su EKS 6.14.0

- Applicazioni supportate - AWS SDK per 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
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  StartJobRunDa CreateManagedEndpoint APIsutilizzare con e:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

  Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Nel rilascio 6.14 di Amazon EMR su EKS sono incluse le funzionalità elencate di seguito.

- Supporto per Apache Livy: Amazon EMR su EKS ora supporta Apache Livy con `spark-submit`.

emr-6.14.0-latest

Note di rilascio: `emr-6.14.0-latest` attualmente indica `emr-6.14.0-20231005`.

Regioni: `emr-6.14.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.14.0:latest`

emr-6.14.0-20231005

Note di rilascio: `6.14.0-20231005` è stato rilasciato il 17 ottobre 2023. Questo è il rilascio iniziale di Amazon EMR 6.14.0.

Regioni: `emr-6.14.0-20231005` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.14.0:20231005`

Rilasci 6.13.0 di Amazon EMR su EKS

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 6.13.0 in generale, consulta Amazon EMR 6.13.0 nella Amazon EMR Release Guide.

Rilasci 6.13 di Amazon EMR su EKS

I seguenti rilasci 6.13.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.13.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio

Note di rilascio di Amazon EMR su EKS 6.13.0

- Applicazioni supportate - AWS SDK per 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
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  StartJobRunDa CreateManagedEndpoint APIsutilizzare con e:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.

Classificazioni	Descrizioni
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Nel rilascio 6.13 di Amazon EMR su EKS sono incluse le funzionalità elencate di seguito.

- Amazon Linux 2023 - Con Amazon EMR su EKS 6.13 e versioni successive, puoi avviare Spark con AL2 023 come sistema operativo insieme al runtime Java 17. Per farlo, utilizza l'etichetta di rilascio con `al2023` nel nome. Ad esempio: `emr-6.13.0-java17-al2023-latest`. Ti consigliamo di convalidare ed eseguire test delle prestazioni prima di spostare i carichi di lavoro di produzione su 023 e Java 17. AL2
- Amazon EMR su EKS con Apache Flink (anteprima pubblica): i rilasci 6.13 e successivi di Amazon EMR su EKS supportano Apache Flink, disponibile in anteprima pubblica. Con questo lancio, puoi eseguire la tua applicazione basata su Apache Flink insieme ad altri tipi di applicazioni sullo stesso cluster Amazon EKS. Ciò consente di migliorare l'utilizzo delle risorse e di semplificare la gestione dell'infrastruttura. Se stai già eseguendo framework di big data su Amazon EKS, ora puoi utilizzare Amazon EMR per automatizzare il provisioning e la gestione.

emr-6.13.0-latest

Note di rilascio: `emr-6.13.0-latest` attualmente indica `emr-6.13.0-20230814`.

Regioni: `emr-6.13.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.13.0:latest`

emr-6.13.0-20230814

Note di rilascio: `6.13.0-20230814` è stato rilasciato il 7 settembre 2023. Questo è il rilascio iniziale di Amazon EMR 6.13.0.

Regioni: `emr-6.13.0-20230814` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.13.0:20230814`

Rilasci 6.12.0 di Amazon EMR su EKS

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon

EC2 e sulla versione Amazon EMR 6.12.0 in generale, consulta Amazon EMR 6.12.0 nella Amazon EMR Release Guide.

Rilasci 6.12 di Amazon EMR su EKS

I seguenti rilasci 6.12.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.12.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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- spark-rapids-java 17 - più recente
- emr-6.12.0- spark-rapids-java 17-20230701
- notebook-spark/emr-6.12.0-latest
- notebook-spark/emr-6.12.0-20230701
- taccuino - 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

Note di rilascio

Note di rilascio di Amazon EMR su EKS 6.12.0

- Applicazioni supportate - AWS SDK per 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
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  CreateManagedEndpoint APIsDa utilizzare con e: StartJobRun

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j`	Modifica i valori nel file Spark `log4j2.properties` .
`emr-job-submitter`	Configurazione per il pod del mittente di processi.

  Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Nel rilascio 6.12 di Amazon EMR su EKS sono incluse le funzionalità elencate di seguito.

- Java 17: con Amazon EMR su EKS 6.12 e versioni successive, puoi avviare Spark con il runtime di Java 17. Per farlo, trasmetti `emr-6.12.0-java17-latest` come etichetta di rilascio. Consigliamo di convalidare ed eseguire test delle prestazioni prima di trasferire i carichi di lavoro di produzione da versioni precedenti dell'immagine Java all'immagine Java 17.

emr-6.12.0-latest

Note di rilascio: `emr-6.12.0-latest` attualmente indica `emr-6.12.0-20240321`.

Regioni: `emr-6.12.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.12.0:latest`

emr-6.12.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `6.12.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.12.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.12.0:20240321`

emr-6.12.0-20230701

Note di rilascio: `6.12.0-20230701` è stato rilasciato il 1° luglio 2023. Questo è il rilascio iniziale di Amazon EMR 6.12.0.

Regioni: `emr-6.12.0-20230701` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.12.0:20230701`

Rilasci 6.11.0 di Amazon EMR su EKS

Questa pagina descrive la funzionalità nuova e aggiornata di Amazon EMR specifica per l'implementazione di Amazon EMR su EKS. Per dettagli su Amazon EMR in esecuzione su Amazon EC2 e sulla versione Amazon EMR 6.11.0 in generale, consulta Amazon EMR 6.11.0 nella Amazon EMR Release Guide.

Rilasci 6.11 di Amazon EMR su EKS

I seguenti rilasci 6.11.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.11.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio

Note di rilascio di Amazon EMR su EKS 6.11.0

- Applicazioni supportate - AWS SDK per 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
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate

  StartJobRunDa CreateManagedEndpoint APIsutilizzare con e:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file Hadoop `core-site.xml` .
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file Spark `metrics.properties` .
`spark-defaults`	Modifica i valori nel file Spark `spark-defaults.conf` .
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file Spark `hive-site.xml` .
`spark-log4j`	Modifica i valori nel file Spark `log4j.properties` .

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

Nel rilascio 6.11 di Amazon EMR su EKS sono incluse le funzionalità elencate di seguito.

- Immagine di base di Amazon EMR su EKS in Amazon ECR Public Gallery: se utilizzi la funzionalità immagine personalizzata, la nostra immagine di base fornisce i jar, la configurazione e le librerie indispensabili per interagire con Amazon EMR su EKS. Ora puoi trovare l'immagine di base in Amazon ECR Public Gallery.
- Rotazione dei log di container Spark: Amazon EMR su EKS 6.11 supporta la rotazione dei log di container Spark. Puoi abilitare la funzionalità con `containerLogRotationConfiguration` all'interno dell'operazione `MonitoringConfiguration` dell'API `StartJobRun`. Puoi configurare i valori di `rotationSize` e `maxFilestoKeep` per specificare il numero e la dimensione dei file di log che desideri che Amazon EMR su EKS conservi nei pod di driver ed executor Spark. Per ulteriori informazioni, consulta Uso della rotazione dei log di container Spark.
- Supporto di Volcano nell'operatore Spark e in spark-submit: Amazon EMR su EKS 6.11 supporta l'esecuzione di processi Spark con Volcano come pianificatore personalizzato Kubernetes nell'operatore Spark e in spark-submit. Puoi utilizzare funzionalità come la pianificazione di gruppo, la gestione delle code, l'azione preventiva e la pianificazione della quota equa per ottenere una velocità di trasmissione effettiva di pianificazione elevata e una capacità ottimizzata. Per ulteriori

informazioni, consulta Uso di Volcano come pianificatore personalizzato per Apache Spark in Amazon EMR su EKS.

emr-6.11.0-latest

Note di rilascio: `emr-6.11.0-latest` attualmente indica `emr-20230905`.

Regioni: `emr-6.11.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.11.0:latest`

emr-6.11.0-20230905

Note di rilascio: è stato rilasciato il 29 settembre 2023. `6.11.0-20230905` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.11.0-20230509` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.11.0:20230509`

emr-6.11.0-20230509

Note di rilascio: `6.11.0-20230509` è stato rilasciato il 9 maggio 2023. Questo è il rilascio iniziale di Amazon EMR 6.11.0.

Regioni: `emr-6.11.0-20230509` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.11.0:20230509`

Rilasci 6.10.0 di Amazon EMR su EKS

I seguenti rilasci 6.10.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.10.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 6.10.0

- Applicazioni supportate - AWS SDK per 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.
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate:

  StartJobRunDa usare CreateManagedEndpoint APIscon e:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file `core-site.xml` di Hadoop.

Classificazioni	Descrizioni
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file `metrics.properties` di Spark.
`spark-defaults`	Modifica i valori nel file `spark-defaults.conf` di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file `hive-site.xml` di Spark.
`spark-log4j`	Modifica i valori nel file `log4j.properties` di Spark.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

- Operatore Spark: con Amazon EMR su EKS 6.10.0 e versioni successive, puoi utilizzare l'operatore Kubernetes per Apache Spark, ovvero l'operatore Spark, per implementare e gestire

applicazioni Spark con il runtime di rilascio di Amazon EMR sui tuoi cluster Amazon EKS. Per ulteriori informazioni, consulta Esecuzione dei processi Spark con l'operatore Spark.

- Java 11: con Amazon EMR su EKS 6.10 e versioni successive, puoi avviare Spark con il runtime di Java 11. Per farlo, trasmetti `emr-6.10.0-java11-latest` come etichetta di rilascio. Consigliamo di convalidare ed eseguire test delle prestazioni prima di trasferire i carichi di lavoro di produzione dall'immagine Java 8 all'immagine Java 11.
- Per l'integrazione di Amazon Redshift per Apache Spark, Amazon EMR su EKS 6.10.0 rimuove la dipendenza da `minimal-json.jar` e aggiunge in automatico i jar relativi a `spark-redshift` necessari al percorso di classe dell'executor per Spark: `spark-redshift.jar`, `spark-avro.jar` e `RedshiftJDBC.jar`.

Modifiche

- Il committer ottimizzato per EMRFS S3 è ora abilitato per impostazione predefinita per formati parquet, ORC e basati su testo (inclusi CSV e JSON).

emr-6.10.0-latest

Note di rilascio: `emr-6.10.0-latest` attualmente indica `emr-6.10.0-20230905`.

Regioni: `emr-6.10.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.10.0:latest`

emr-6.10.0-20230905

Note di rilascio: è stato rilasciato il 29 settembre 2023. `6.10.0-20230905` Rispetto al rilascio precedente, questa versione è stata aggiornata con pacchetti Amazon Linux recentemente aggiornati e correzioni importanti.

Regioni: `emr-6.10.0-20230905` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.10.0:20230905`

emr-6.10.0-20230624

Note di rilascio: `6.10.0-20230624` è stato rilasciato il 7 luglio 2023. Rispetto al rilascio precedente, questa versione è stata aggiornata con pacchetti Amazon Linux recentemente aggiornati e correzioni importanti.

Regioni: `emr-6.10.0-20230624` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.10.0:20230624`

emr-6.10.0-20230421

Note di rilascio: `6.10.0-20230421` è stato rilasciato il 28 aprile 2023. Rispetto al rilascio precedente, questa versione è stata aggiornata con pacchetti Amazon Linux recentemente aggiornati e correzioni importanti.

Regioni: `emr-6.10.0-20230421` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.10.0:20230421`

emr-6.10.0-20230403

Note di rilascio: `6.10.0-20230403` è stato rilasciato il 12 aprile 2023. Rispetto al rilascio precedente, questa versione è stata aggiornata con pacchetti Amazon Linux recentemente aggiornati e correzioni importanti.

Regioni: `emr-6.10.0-20230403` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.10.0:20230403`

emr-6.10.0-20230220

Note di rilascio: `emr-6.10.0-20230220` è stato rilasciato il 20 febbraio 2023. Questo è il rilascio iniziale di Amazon EMR 6.10.0.

Regioni: `emr-6.10.0-20230220` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.10.0:20230220`

Rilasci 6.9.0 di Amazon EMR su EKS

I seguenti rilasci 6.9.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.9.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- emr-6.9.0-latest
- emr-6.9.0-20230905
- 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

Note di rilascio di Amazon EMR 6.9.0

- Applicazioni supportate - AWS SDK per 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.
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate:

  StartJobRunDa CreateManagedEndpoint APIsusare con e:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.

Classificazioni	Descrizioni
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Da utilizzare specificamente con CreateManagedEndpoint APIs:

Classificazioni	Descrizioni
`jeg-config`	Modifica i valori nel file `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifica il valore per l'immagine del kernel nel file Jupyter Kernel Spec.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Funzionalità significative

- Nvidia RAPIDS Accelerator for Apache Spark - Amazon EMR su EKS per accelerare Spark utilizzando tipi di istanze di unità di elaborazione EC2 grafica (GPU). Per utilizzare l'immagine

Spark con RAPIDS Accelerator, specifica l'etichetta di rilascio come emr-6.9.0-. spark-rapids-latest Per maggiori informazioni, consulta la pagina della documentazione.

- Connettore Spark-Redshift: l'integrazione di Amazon Redshift per Apache Spark è inclusa in Amazon EMR rilascio 6.9.0 e successivi. In precedenza uno strumento open source, l'integrazione nativa è un connettore Spark che è possibile utilizzare per creare applicazioni Apache Spark in grado di leggere e scrivere dati in Amazon Redshift e Amazon Redshift Serverless. Per ulteriori informazioni, consulta Uso dell'integrazione di Amazon Redshift per Apache Spark in Amazon EMR su EKS.
- Delta Lake: Delta Lake è un formato di archiviazione open source che consente di creare data lake con coerenza transazionale, definizione coerente di set di dati, modifiche all'evoluzione dello schema e supporto per le mutazioni dei dati. Per maggiori informazioni, consulta la sezione Uso di Delta Lake.
- Modifica PySpark parametri - Gli endpoint interattivi ora supportano la modifica dei parametri Spark associati alle PySpark sessioni in EMR Studio Jupyter Notebook. Visita Modificare i parametri della sessione per saperne di più. PySpark

Problemi risolti

- Quando utilizzi il connettore DynamoDB con Spark nelle versioni 6.6.0, 6.7.0 e 6.8.0 di Amazon EMR, tutte le letture della tabella restituiscono un risultato vuoto, anche se la divisione di input fa riferimento a dati non vuoti. Amazon EMR rilascio 6.9.0 risolve questo problema.
- Amazon EMR su EKS 6.8.0 popola in modo errato l'hash di compilazione nei metadati dei file Parquet generati con Apache Spark. Questo problema può causare errori negli strumenti che analizzano la stringa della versione dei metadati dai file Parquet generati da Amazon EMR su EKS 6.8.0.

Problema noto

- Se utilizzi l'integrazione Amazon Redshift per Apache Spark e disponi di un orario, timetz, timestamp o timestamptz con precisione al microsecondo in formato Parquet, il connettore arrotonda i valori temporali al valore in millisecondi più vicino. Come soluzione alternativa, utilizza il parametro `unload_s3_format` del formato di scaricamento del testo.

emr-6.9.0-latest

Note di rilascio: `emr-6.9.0-latest` attualmente indica `emr-6.9.0-20230905`.

Regioni: `emr-6.9.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.9.0:latest`

emr-6.9.0-20230905

Note di rilascio: `emr-6.9.0-20230905`. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.9.0-20230905` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.9.0:20230905`

emr-6.9.0-20230624

Note di rilascio: `emr-6.9.0-20230624` è stato rilasciato il 7 luglio 2023.

Regioni: `emr-6.9.0-20230624` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.9.0:20230624`

emr-6.9.0-20221108

Note di rilascio: `emr-6.9.0-20221108` è stato rilasciato l'8 dicembre 2022. Questo è il rilascio iniziale di Amazon EMR 6.9.0.

Regioni: `emr-6.9.0-20221108` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.9.0:20221108`

Rilasci 6.8.0 di Amazon EMR su EKS

I seguenti rilasci 6.8.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.8.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 6.8.0

- Applicazioni supportate - AWS SDK per Java 1.12.170, Spark 3.3.0-amzn-0, Hudi 0.11.1-amzn-0, Iceberg 0.14.0-amzn-0.
- Componenti supportati: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.

Classificazioni	Descrizioni
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Caratteristiche da tenere in considerazione

- Spark3.3.0 - Amazon EMR su EKS 6.8 include Spark 3.3.0, che supporta l'uso di etichette di selezione del nodo separate per i pod Spark Driver Executor. Queste nuove etichette consentono di definire i tipi di nodi per i pod driver ed executor separatamente nell'API, senza utilizzare modelli di pod. StartJobRun
  - Proprietà del selettore del nodo driver: spark.kubernetes.driver.node.selector.[labelKey]
  - Proprietà dell'executor del nodo driver: spark.kubernetes.executor.node.selector.[labelKey]
- Messaggio di errore dei processi migliorato: in questo rilascio è stata introdotta la funzione configurazione `spark.stage.extraDetailsOnFetchFailures.enabled` e `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` per tenere traccia degli errori delle attività dovuti al codice utente. Questi dettagli verranno utilizzati per migliorare il messaggio di errore visualizzato nel log del driver quando una fase viene interrotta a causa di un errore di recupero casuale.

Nome proprietà	Valore predefinito	Significato	Dalla versione
`spark.stage.extraDetailsOnFetchFailures.enabled`	false	Se impostato su `true`, questa proprietà viene utilizzata per migliorare il messaggio di errore visualizzato nel log del driver quando una fase viene interrotta	emr-6.8

Nome proprietà	Valore predefinito	Significato	Dalla versione
		a causa di un errore di recupero casuale. Per impostazione predefinita, vengono tracciati gli ultimi 5 errori causati dal codice utente e il messaggio di errore viene aggiunto nei registri dei driver. Per aumentare il numero di errori delle attività con le eccezioni degli utenti da monitorare, consulta la configurazione `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` .	

Nome proprietà	Valore predefinito	Significato	Dalla versione
`spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude`	5	Numero di operazioni non riuscite per monitorare per fase e tentativo. Questa proprietà viene utilizzata per migliorare il messaggio di errore con eccezioni utente visualizzato nel registro del log quando una fase viene interrotta a causa di un errore di recupero casuale. Questa proprietà funziona solo se Config spark.stage.extraDetailsOnFetchFailures.enabled è impostato su true.	emr-6.8

Per ulteriori informazioni, documentazione di configurazione di Apache Spark.

Problema noto

- Amazon EMR su EKS 6.8.0 popola erroneamente l'hash di compilazione nei metadati dei file Parquet generati con Apache Spark. Questo problema può causare il fallimento degli strumenti che analizzano la stringa della versione dei metadati dai file Parquet generati da Amazon EMR su EKS 6.8.0. I clienti che analizzano la stringa della versione dai metadati di Parquet e dipendono dall'hash di compilazione devono passare a una versione diversa di Amazon EMR e riscrivere il file.

Problema risolto

- Funzionalità Interrupt Kernel per i kernel PySpark: i carichi di lavoro interattivi in corso che vengono attivati dall'esecuzione di celle in un notebook possono essere interrotti utilizzando la funzionalità `Interrupt Kernel`. È stata introdotta una correzione in modo che questa funzionalità funzioni per i kernel pySpark. È disponibile anche in versione open source all'indirizzo Changes for handling interrupts for PySpark Kubernetes Kernel #1115.

emr-6.8.0-latest

Note di rilascio: `emr-6.8.0-latest` attualmente indica `emr-6.8.0-20230624`.

Regioni: `emr-6.8.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.8.0:latest`

emr-6.8.0-20230905

Note di rilascio: è stato rilasciato il 29 settembre 2023. `emr-6.8.0-20230905` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.8.0-20230905` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.8.0:20230905`

emr-6.8.0-20230624

Note di rilascio: `emr-6.8.0-20230624` è stato rilasciato il 7 luglio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.8.0-20230624` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.8.0:20230624`

emr-6.8.0-20221219

Note di rilascio: `emr-6.8.0-20221219` è stato rilasciato il 19 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.8.0-20221219` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.8.0:20221219`

emr-6.8.0-20220802

Note di rilascio: `emr-6.8.0-20220802` è stato rilasciato il 27 settembre 2022. Questo è il rilascio iniziale di Amazon EMR 6.8.0.

Regioni: `emr-6.8.0-20220802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.8.0:20220802`

Rilasci 6.7.0 di Amazon EMR su EKS

I seguenti rilasci 6.7.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.7.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 6.7.0

- Applicazioni supportate: Spark 3.2.1-amzn-0, Jupyter Enterprise Gateway 2.6, Hudi 0.11-amzn-0, Iceberg 0.13.1.
- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.

- Con l'aggiornamento a JEG 2.6, la gestione del kernel è ora asincrona, il che significa che JEG non blocca le transazioni quando è in corso un avvio del kernel. Ciò migliora notevolmente l'esperienza dell'utente fornendo quanto segue:
  - capacità di eseguire comandi nei notebook attualmente in esecuzione quando sono in corso altri avvii del kernel
  - capacità di avviare più kernel contemporaneamente senza influire sui kernel già in esecuzione
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file `core-site.xml` Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file `metrics.properties` Spark.
`spark-defaults`	Modifica i valori nel file `spark-defaults.conf` Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file `hive-site.xml` Spark.
`spark-log4j`	Modifica i valori nel file `log4j.properties` Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio `spark-hive-site.xml`. Per ulteriori informazioni, consulta Configurazione delle applicazioni.

Problemi risolti

- Amazon EMR su EKS 6.7 risolve un problema nella versione 6.6 legato all'utilizzo della funzionalità dei modelli di pod di Apache Spark con gli endpoint interattivi. Il problema era presente nei rilasci

6.4, 6.5 e 6.6 di Amazon EMR su EKS. Ora puoi utilizzare i modelli di pod per definire la modalità di avvio dei pod di driver ed executor Spark quando utilizzi gli endpoint interattivi per eseguire analisi interattive.

- Nei rilasci precedenti di Amazon EMR su EKS, Jupyter Enterprise Gateway bloccava le transazioni quando l'avvio del kernel era in corso e ciò impediva l'esecuzione delle sessioni del notebook attualmente in esecuzione. Ora è possibile eseguire comandi nei notebook attualmente in esecuzione quando sono in corso altri avvii del kernel. Inoltre, è possibile avviare più kernel contemporaneamente senza il rischio di perdere la connettività con i kernel già in esecuzione.

emr-6.7.0-latest

Note di rilascio: `emr-6.7.0-latest` attualmente indica `emr-6.7.0-20240321`.

Regioni: `emr-6.7.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.7.0:latest`

emr-6.7.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-6.7.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.7.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.7.0:20240321`

emr-6.7.0-20230624

Note di rilascio: `emr-6.7.0-20230624` è stato rilasciato il 7 luglio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.7.0-20230624` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.7.0:20230624`

emr-6.7.0-20221219

Note di rilascio: `emr-6.7.0-20221219` è stato rilasciato il 19 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.7.0-20221219` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.7.0:20221219`

emr-6.7.0-20220630

Note di rilascio: `emr-6.7.0-20220630` è stato rilasciato il 12 luglio 2022. Questo è il rilascio iniziale di Amazon EMR 6.7.0.

Regioni: `emr-6.7.0-20220630` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.7.0:20220630`

Rilasci 6.6.0 di Amazon EMR su EKS

I seguenti rilasci 6.6.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.6.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 6.6.0

- Applicazioni supportate: Spark 3.2.0-amzn-0, Jupyter Enterprise Gateway (endpoint, anteprima pubblica), Hudi 0.10.1-amzn-0, Iceberg 0.13.1.

- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio.xml. spark-hive-site Per ulteriori informazioni, consulta Configurazione delle applicazioni.

Problema noto

- La funzionalità del modello di pod Spark con endpoint interattivi non funziona nei rilasci 6.4, 6.5 e 6.6 di Amazon EMR su EKS.

Problema risolto

- I log degli endpoint interattivi vengono caricati su Cloudwatch e S3.

emr-6.6.0-latest

Note di rilascio: `emr-6.6.0-latest` attualmente indica `emr-6.6.0-20240321`.

Regioni: `emr-6.6.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.6.0:latest`

emr-6.6.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-6.6.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.6.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.6.0:20240321`

emr-6.6.0-20230624

Note di rilascio: `emr-6.6.0-20230624` è stato rilasciato il 27 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.6.0-20230624` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.6.0:20230624`

emr-6.6.0-20221219

Note di rilascio: `emr-6.6.0-20221219` è stato rilasciato il 27 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.6.0-20221219` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.6.0:20221219`

emr-6.6.0-20220411

Note di rilascio: `emr-6.6.0-20220411` è stato rilasciato il 20 maggio 2022. Questo è il rilascio iniziale di Amazon EMR 6.6.0.

Regioni: `emr-6.6.0-20220411` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.6.0:20220411`

Rilasci 6.5.0 di Amazon EMR su EKS

I seguenti rilasci 6.5.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.5.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 6.5.0

- Applicazioni supportate: Spark 3.1.2-amzn-1, Jupyter Enterprise Gateway (endpoint, anteprima pubblica).
- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.

Classificazioni	Descrizioni
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio.xml. spark-hive-site Per ulteriori informazioni, consulta Configurazione delle applicazioni.

Problema noto

- La funzionalità del modello di pod Spark con endpoint interattivi non funziona nei rilasci 6.4 e 6.5 di Amazon EMR su EKS.

emr-6.5.0-latest

Note di rilascio: `emr-6.5.0-latest` attualmente indica `emr-6.5.0-20240321`.

Regioni: `emr-6.5.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.5.0:latest`

emr-6.5.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-6.5.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.5.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.5.0:20240321`

emr-6.5.0-20221219

Note di rilascio: `emr-6.5.0-20221219` è stato rilasciato il 19 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.5.0-20221219` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.5.0:20221219`

emr-6.5.0-20220802

Note di rilascio: `emr-6.5.0-20220802` è stato rilasciato il 24 agosto 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-6.5.0-20220802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.5.0:20220802`

emr-6.5.0-20211119

Note di rilascio: `emr-6.5.0-20211119` è stato rilasciato il 20 gennaio 2022. Questo è il rilascio iniziale di Amazon EMR 6.5.0.

Regioni: `emr-6.5.0-20211119` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.5.0:20211119`

Rilasci 6.4.0 di Amazon EMR su EKS

I seguenti rilasci 6.4.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.4.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- emr-6.4.0-latest
- emr-6.4.0-20240321
- emr-6.4.0-20221219
- emr-6.4.0-20210830

Note di rilascio di Amazon EMR 6.4.0

- Applicazioni supportate: Spark 3.1.2-amzn-0, Jupyter Enterprise Gateway (endpoint, anteprima pubblica).
- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Questi spesso corrispondono a un file XML di configurazione per l'applicazione, ad spark-hive-site esempio.xml. Per ulteriori informazioni, consulta Configurazione delle applicazioni.

Problema noto

- La funzionalità del modello di pod Spark con endpoint interattivi non funziona nel rilascio 6.4 di Amazon EMR su EKS.

emr-6.4.0-latest

Note di rilascio: `emr-6.4.0-latest` attualmente indica `emr-6.4.0-20240321`.

Regioni: `emr-6.4.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.4.0:latest`

emr-6.4.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-6.4.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.4.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.4.0:20240321`

emr-6.4.0-20221219

Note di rilascio: `emr-6.4.0-20221219` è stato rilasciato il 27 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiunti di recente.

Regioni: `emr-6.4.0-20221219` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.4.0:20221219`

emr-6.4.0-20210830

Note di rilascio: `emr-6.4.0-20210830` è stato rilasciato il 9 dicembre 2021. Questo è il rilascio iniziale di Amazon EMR 6.4.0.

Regioni: `emr-6.4.0-20210830` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.4.0:20210830`

Rilasci 6.3.0 di Amazon EMR su EKS

I seguenti rilasci 6.3.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.3.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 6.3.0

- Nuove funzionalità: a partire da Amazon EMR 6.3.0 nella serie di rilasci 6.x, Amazon EMR su EKS supporta la funzione del modello pod di Spark. È anche possibile attivare la funzionalità di rotazione dei log di eventi Spark per Amazon EMR su EKS. Per ulteriori informazioni, consultare Uso dei modelli di pod e Uso della rotazione dei log di eventi Spark.
- Applicazioni supportate: Spark 3.1.1-amzn-0, Jupyter Enterprise Gateway (endpoint, anteprima pubblica).
- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.

Classificazioni	Descrizioni
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Questi spesso corrispondono a un file XML di configurazione per l'applicazione, ad spark-hive-site esempio.xml. Per ulteriori informazioni, consulta Configurazione delle applicazioni.

emr-6.3.0-latest

Note di rilascio: `emr-6.3.0-latest` attualmente punta a `emr-6.3.0-20240321`.

Regioni: `emr-6.3.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.3.0:latest`

emr-6.3.0-20240321

Note di rilascio: `emr-6.3.0-20240321` è stato rilasciato l'11 marzo 2024. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.3.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.3.0:20240321`

emr-6.3.0-20220802

Note di rilascio: `emr-6.3.0-20220802` è stato rilasciato il 27 settembre 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-6.3.0-20220802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.3.0:20220802`

emr-6.3.0-20211008

Note di rilascio: `emr-6.3.0-20211008` è stato rilasciato il 9 dicembre 2021. Rispetto alla versione precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-6.3.0-20211008` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.3.0:20211008`

emr-6.3.0-20210802

Note di rilascio: `emr-6.3.0-20210802` è stato rilasciato il 2 agosto 2021. Rispetto al rilascio precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-6.3.0-20210802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.3.0:20210802`

emr-6.3.0-20210429

Note di rilascio: `emr-6.3.0-20210429` è stato rilasciato il 29 aprile 2021. Questo è il rilascio iniziale di Amazon EMR 6.3.0.

Regioni: `emr-6.3.0-20210429` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.3.0:20210429`

Rilasci 6.2.0 di Amazon EMR su EKS

I seguenti rilasci 6.2.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-6.2.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 6.2.0

- Applicazioni supportate: Spark 3.0.1-amzn-0, Jupyter Enterprise Gateway (endpoint, anteprima pubblica).
- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.

Classificazioni	Descrizioni
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Questi spesso corrispondono a un file XML di configurazione per l'applicazione, ad spark-hive-site esempio.xml. Per ulteriori informazioni, consulta Configurazione delle applicazioni.

emr-6.2.0-latest

Note di rilascio: `emr-6.2.0-latest` attualmente indica `emr-6.2.0-20240321`.

Regioni: `emr-6.2.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.2.0:20240321`

emr-6.2.0-20240321

Note di rilascio: `emr-6.2.0-20240321` è stato rilasciato l'11 marzo 2024. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-6.2.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.2.0:20240321`

emr-6.2.0-20220802

Note di rilascio: `emr-6.2.0-20220802` è stato rilasciato il 27 settembre 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-6.2.0-20220802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-6.2.0:20220802`

emr-6.2.0-20211008

Note di rilascio: `emr-6.2.0-20211008` è stato rilasciato il 9 dicembre 2021. Rispetto alla versione precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-6.2.0-20211008` è disponibile nelle seguenti Regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-6.2.0:20211008`

emr-6.2.0-20210802

Note di rilascio: `emr-6.2.0-20210802` è stato rilasciato il 2 agosto 2021. Rispetto al rilascio precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-6.2.0-20210802` è disponibile nelle seguenti Regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-6.2.0:20210802`

emr-6.2.0-20210615

Note di rilascio: `emr-6.2.0-20210615` è stato rilasciato il 15 giugno 2021. Rispetto al rilascio precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-6.2.0-20210615` è disponibile nelle seguenti Regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-6.2.0:20210615`

emr-6.2.0-20210129

Note di rilascio: `emr-6.2.0-20210129` è stato rilasciato il 29 gennaio 2021. Rispetto a `emr-6.2.0-20201218`, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-6.2.0-20210129` è disponibile nelle seguenti regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-6.2.0-20210129`

emr-6.2.0-20201218

Note di rilascio: `emr-6.2.0-20201218` è stato rilasciato il 18 dicembre 2020. Rispetto a `emr-6.2.0-20201201`, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-6.2.0-20201218` è disponibile nelle seguenti regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-6.2.0-20201218`

emr-6.2.0-20201201

Note di rilascio: `emr-6.2.0-20201201` è stato rilasciato il 1° dicembre 2020. Questo è il rilascio iniziale di Amazon EMR 6.2.0.

Regioni: `emr-6.2.0-20201201` è disponibile nelle seguenti Regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-6.2.0-20201201`

Rilasci 5.36.0 di Amazon EMR su EKS

I seguenti rilasci 5.36.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-5.36.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 5.36.0

- Risolti i problemi di sicurezza log4j2.
- Applicazioni supportate: Spark 2.4.8-amzn-2, Jupyter Enterprise Gateway (endpoint, anteprima pubblica; kernel Scala non supportato), livy-0.7.1, fluentd-4.0.0.
- Componenti supportati - aws-hm-client, emr-ddb aws-sagemaker-spark-sdk, emr-goodies, emr-kinesis, kerberos-server.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.

Classificazioni	Descrizioni
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Questi spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio.xml. spark-hive-site Per ulteriori informazioni, consulta Configurazione delle applicazioni.

emr-5.36.0-latest

Note di rilascio: `emr-5.36.0-latest` attualmente indica `emr-5.36.0-20240321`.

Regioni: `emr-5.36.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.36.0:latest`

emr-5.36.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-5.36.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-5.36.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.36.0:20240321`

emr-5.36.0-20221219

Note di rilascio: `emr-5.36.0-20221219` è stato rilasciato il 27 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.36.0-20221219` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.36.0:20221219`

emr-5.36.0-20220620

Note di rilascio: `emr-5.36.0-20220620` è stato rilasciato il 27 luglio 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.36.0-20220620` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.36.0:20220620`

emr-5.36.0-20220525

Note di rilascio: `emr-5.36.0-20220525` è stato rilasciato il 16 giugno 2022. Questo è il rilascio iniziale di Amazon EMR 5.36.0.

Regioni: `emr-5.36.0-20220525` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.36.0:20220525`

Rilasci 5.35.0 di Amazon EMR su EKS

I seguenti rilasci 5.35.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-5.35.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 5.35.0

- Risolti i problemi di sicurezza log4j2.

- Applicazioni supportate: Spark 2.4.8-amzn-1, Hudi 0.9.0-amzn-2, Jupyter Enterprise Gateway (endpoint, anteprima pubblica; kernel Scala non supportato).
- Componenti supportati - aws-hm-client (connettore Glue), emr-s3-select aws-sagemaker-spark-sdk, emrfs, emr-ddb, hudi-spark.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Questi spesso corrispondono a un file XML di configurazione per l'applicazione, ad esempio.xml. spark-hive-site Per ulteriori informazioni, consulta Configurazione delle applicazioni.

emr-5.35.0-latest

Note di rilascio: `emr-5.35.0-latest` attualmente indica `emr-5.35.0-20240321`.

Regioni: `emr-5.35.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.35.0:latest`

emr-5.35.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-5.35.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-5.35.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.35.0:20240321`

emr-5.35.0-20221219

Note di rilascio: `emr-5.35.0-20221219` è stato rilasciato il 27 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.35.0-20221219` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.35.0:20221219`

emr-5.35.0-20220802

Note di rilascio: `emr-5.35.0-20220802` è stato rilasciato il 27 settembre 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.35.0-20220802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.35.0:20220802`

emr-5.35.0-20220307

Note di rilascio: `emr-5.35.0-20220307` è stato rilasciato il 30 marzo 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.35.0-20220307` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.35.0:20220307`

Rilasci 5.34.0 di Amazon EMR su EKS

I seguenti rilasci 5.34.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-5.34.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- emr-5.34.0-latest
- emr-5.34.0-20240321
- emr-5.34.0-20220802

Note di rilascio di Amazon EMR 5.34.0

- Applicazioni supportate: Spark 2.4.8-amzn-0, Jupyter Enterprise Gateway (endpoint, anteprima pubblica; kernel Scala non supportato).
- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.

Classificazioni	Descrizioni
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Questi spesso corrispondono a un file XML di configurazione per l'applicazione, ad spark-hive-site esempio.xml. Per ulteriori informazioni, consulta Configurazione delle applicazioni.

emr-5.34.0-latest

Note di rilascio: `emr-5.34.0-latest` attualmente indica `emr-5.34.0-20220802`.

Regioni: `emr-5.34.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.34.0:latest`

emr-5.34.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-5.34.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-5.34.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.34.0:20240321`

emr-5.34.0-20220802

Note di rilascio: `emr-5.34.0-20220802` è stato rilasciato il 24 agosto 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.34.0-20220802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.34.0:20220802`

emr-5.34.0-20211208

Note di rilascio: `emr-5.34.0-20211208` è stato rilasciato il 20 gennaio 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.34.0-20211208` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.34.0:20211208`

Rilasci 5.33.0 di Amazon EMR su EKS

I seguenti rilasci 5.33.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-5.33.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 5.33.0

- Nuova funzionalità: a partire da Amazon EMR 5.33.0 nella serie di rilasci 5.x, Amazon EMR su EKS supporta la funzione del modello pod di Spark. Per ulteriori informazioni, consulta Uso dei modelli di pod.
- Applicazioni supportate: Spark 2.4.7-amzn-1, Jupyter Enterprise Gateway (endpoint, anteprima pubblica; kernel Scala non supportato).
- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Questi spesso corrispondono a un file XML di configurazione per l'applicazione, ad spark-hive-site esempio.xml. Per ulteriori informazioni, consulta Configurazione delle applicazioni.

emr-5.33.0-latest

Note di rilascio: `emr-5.33.0-latest` attualmente punta a `emr-5.33.0-20240321`.

Regioni: `emr-5.33.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.33.0:latest`

emr-5.33.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-5.33.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-5.33.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.33.0:20240321`

emr-5.33.0-20221219

Note di rilascio: `emr-5.33.0-20221219` è stato rilasciato il 19 gennaio 2023. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-5.33.0-20221219` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.33.0:20221219`

emr-5.33.0-20220802

Note di rilascio: `emr-5.33.0-20220802` è stato rilasciato il 24 agosto 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.33.0-20220802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.33.0:20220802`

emr-5.33.0-20211008

Note di rilascio: `emr-5.33.0-20211008` è stato rilasciato il 9 dicembre 2021. Rispetto alla versione precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-5.33.0-20211008` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.33.0:20211008`

emr-5.33.0-20210802

Note di rilascio: `emr-5.33.0-20210802` è stato rilasciato il 2 agosto 2021. Rispetto al rilascio precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-5.33.0-20210802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.33.0:20210802`

emr-5.33.0-20210615

Note di rilascio: `emr-5.33.0-20210615` è stato rilasciato il 15 giugno 2021. Rispetto al rilascio precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-5.33.0-20210615` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag immagine del container: `emr-5.33.0:20210615`

emr-5.33.0-20210323

Note di rilascio: `emr-5.33.0-20210323` è stato rilasciato il 23 marzo 2021. Questo è il rilascio iniziale di Amazon EMR 5.33.0.

Regioni: `emr-5.33.0-20210323` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.33.0-20210323`

Rilasci 5.32.0 di Amazon EMR su EKS

I seguenti rilasci 5.32.0 di Amazon EMR sono disponibili per Amazon EMR su EKS. Seleziona un rilascio emr-5.32.0-XXXX specifico per visualizzare ulteriori dettagli, come il relativo tag dell'immagine di container.

- 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

Note di rilascio di Amazon EMR 5.32.0

- Applicazioni supportate: Spark 2.4.7-amzn-0, Jupyter Enterprise Gateway (endpoint, anteprima pubblica; kernel Scala non supportato).
- Componenti supportati: `aws-hm-client` (connettore Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classificazioni di configurazione supportate:

Classificazioni	Descrizioni
`core-site`	Modifica i valori nel file core-site.xml di Hadoop.
`emrfs-site`	Modifica le impostazioni EMRFS.
`spark-metrics`	Modifica i valori nel file metrics.properties di Spark.
`spark-defaults`	Modifica i valori nel file spark-defaults.conf di Spark.
`spark-env`	Modifica i valori nell'ambiente Spark.
`spark-hive-site`	Modifica i valori nel file hive-site.xml di Spark.
`spark-log4j`	Modifica i valori nel file log4j.properties di Spark.

Le classificazioni di configurazione consentono di personalizzare le applicazioni. Questi spesso corrispondono a un file XML di configurazione per l'applicazione, ad spark-hive-site esempio.xml. Per ulteriori informazioni, consulta Configurazione delle applicazioni.

emr-5.32.0-latest

Note di rilascio: `emr-5.32.0-latest` attualmente indica `emr-5.32.0-20240321`.

Regioni: `emr-5.32.0-latest` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.32.0:latest`

emr-5.32.0-20240321

Note sulla versione: è stato rilasciato l'11 marzo 2024. `emr-5.32.0-20240321` Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux aggiornati di recente e le correzioni critiche.

Regioni: `emr-5.32.0-20240321` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.32.0:20240321`

emr-5.32.0-20220802

Note di rilascio: `emr-5.32.0-20220802` è stato rilasciato il 24 agosto 2022. Rispetto alla versione precedente, questa versione è stata aggiornata con i pacchetti Amazon Linux recentemente aggiornati.

Regioni: `emr-5.32.0-20220802` è disponibile in tutte le Regioni supportate da Amazon EMR su EKS. Per ulteriori informazioni, consulta Endpoint del servizio di Amazon EMR su EKS.

Tag dell'immagine di container: `emr-5.32.0:20220802`

emr-5.32.0-20211008

Note di rilascio: `emr-5.32.0-20211008` è stato rilasciato il 9 dicembre 2021. Rispetto alla versione precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-5.32.0-20211008` è disponibile nelle seguenti Regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-5.32.0:20211008`

emr-5.32.0-20210802

Note di rilascio: `emr-5.32.0-20210802` è stato rilasciato il 2 agosto 2021. Rispetto al rilascio precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-5.32.0-20210802` è disponibile nelle seguenti Regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-5.32.0:20210802`

emr-5.32.0-20210615

Note di rilascio: `emr-5.32.0-20210615` è stato rilasciato il 15 giugno 2021. Rispetto al rilascio precedente, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-5.32.0-20210615` è disponibile nelle seguenti Regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-5.32.0:20210615`

emr-5.32.0-20210129

Note di rilascio: `emr-5.32.0-20210129` è stato rilasciato il 29 gennaio 2021. Rispetto a `emr-5.32.0-20201218`, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-5.32.0-20210129` è disponibile nelle seguenti regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-5.32.0-20210129`

emr-5.32.0-20201218

Note di rilascio: `5.32.0-20201218` è stato rilasciato il 18 dicembre 2020. Rispetto a `5.32.0-20201201`, questa versione contiene correzioni dei problemi e aggiornamenti di sicurezza.

Regioni: `emr-5.32.0-20201218` è disponibile nelle seguenti regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-5.32.0-20201218`

emr-5.32.0-20201201

Note di rilascio: `5.32.0-20201201` è stato rilasciato l'1 dicembre 2020. Questo è il rilascio iniziale di Amazon EMR 5.32.0.

Regioni: `5.32.0-20201201` è disponibile nelle seguenti Regioni: Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo), Europa (Irlanda), Sud America (San Paolo).

Tag immagine del container: `emr-5.32.0-20201201`

Cronologia dei documenti

Nella tabella seguente vengono descritte le modifiche importanti apportate alla documentazione dall'ultima versione di Amazon EMR su EKS. Per ulteriori informazioni sugli aggiornamenti di questa documentazione, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Aggiunte informazioni sulla versione EMR 7.12.0	Aggiunte informazioni sulla versione EMR 7.12.0 con Iceberg Materialized Views e Hudi Full Table Access	Novembre 2025
Aggiunte informazioni sulla versione EMR 7.11.0	Aggiunte informazioni sulla versione EMR 7.11.0 con l'integrazione di Unified Studio SageMaker	Novembre 2025
Contenuti aggiornati	Policy gestite per Amazon EMR su EKS— Autorizzazioni aggiuntive per. `AmazonEMRContainersServiceRolePolicy`	3 febbraio 2025
Nuovo rilascio	Amazon EMR nelle versioni EKS 7.6.0	10 gennaio 2025
Nuovo rilascio	Amazon EMR nelle versioni EKS 7.5.0	21 novembre 2024
Nuovo rilascio	Amazon EMR nelle versioni EKS 7.4.0	13 novembre 2024
Nuovo rilascio	Amazon EMR nelle versioni EKS 7.3.0	16 ottobre 2024
Nuovo rilascio	Amazon EMR nelle versioni EKS 7.2.0	25 luglio 2024
Nuovo rilascio	Amazon EMR nelle versioni EKS 7.1.0	17 aprile 2024
Nuovo rilascio	Rilasci 7.0.0 di Amazon EMR su EKS	22 dicembre 2023
Nuovo rilascio	Rilasci 6.15.0 di Amazon EMR su EKS	17 novembre 2023
Nuovo rilascio	Rilasci 6.14.0 di Amazon EMR su EKS	17 ottobre 2023

Modifica	Descrizione	Data
Contenuti aggiornati	Rinomina degli "endpoint gestiti" in endpoint interattivi; disponibilità generale degli endpoint interattivi	29 settembre 2023
Nuovo rilascio	Rilasci 6.13.0 di Amazon EMR su EKS e documenti in anteprima pubblica per Esecuzione di processi Flink con Amazon EMR su EKS	12 settembre 2023
Nuovo rilascio	Rilasci 6.12.0 di Amazon EMR su EKS	21 luglio 2023
Nuovo contenuto	Aggiunto Uso di Volcano come pianificatore personalizzato per Apache Spark in Amazon EMR su EKS	13 giugno 2023
Nuovo contenuto	Aggiunto Uso di Volcano come pianificatore personalizzato per Apache Spark in Amazon EMR su EKS	13 giugno 2023
Nuovo contenuto	Aggiunto Uso della rotazione dei log di container Spark	12 giugno 2023
Contenuti aggiornati	Aggiornata la documentazione sulle immagini personalizzate per trovare le informazioni sulle immagini di base in Amazon ECR Public Gallery.	8 giugno 2023
Nuovo rilascio	Rilasci 6.11.0 di Amazon EMR su EKS	8 giugno 2023
Nuovo contenuto	Aggiunta la sezione Esecuzione dei processi Spark con l'operatore Spark e riorganizzate le sezioni Esecuzioni di processo in Esecuzione di job Spark con Amazon EMR su EKS.	5 giugno 2023

Modifica	Descrizione	Data
Nuovo contenuto	Aggiunte due sezioni: Uso del dimensionamento automatico verticale con i processi Spark di Amazon EMR e Uso di notebook Jupyter in hosting autonomo	4 maggio 2023
Pagina con la cronologia dei documenti	È stata creata una pagina con la cronologia dei documenti per Amazon EMR su EKS.	13 marzo 2023
Pagina delle policy gestite	È stata creata una pagina delle policy gestite per Amazon EMR su EKS.	13 marzo 2023

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.