Guida per l'utente di Amazon EMR Serverless

Amazon EMR

Amazon EMR: Guida per l'utente di Amazon EMR Serverless

Table of Contents

Cos'è Amazon EMR Serverless?

Amazon EMR Serverless è un'opzione di distribuzione per Amazon EMR che fornisce un ambiente di runtime serverless. Ciò semplifica il funzionamento delle applicazioni di analisi che utilizzano i più recenti framework open source, come Apache Spark e Apache Hive. Con EMR Serverless, non è necessario configurare, ottimizzare, proteggere o utilizzare i cluster per eseguire applicazioni con questi framework.

EMRServerless consente di evitare un approvvigionamento eccessivo o insufficiente delle risorse per le attività di elaborazione dei dati. EMRServerless determina automaticamente le risorse necessarie all'applicazione, ottiene queste risorse per elaborare i lavori e rilascia le risorse al termine dei processi. Nei casi d'uso in cui le applicazioni richiedono una risposta entro pochi secondi, come l'analisi interattiva dei dati, è possibile preinizializzare le risorse necessarie all'applicazione al momento della creazione dell'applicazione.

Con EMR Serverless, continuerai a ottenere i vantaggi di AmazonEMR, come la compatibilità open source, la concorrenza e le prestazioni di runtime ottimizzate per i framework più diffusi.

EMRServerless è adatto ai clienti che desiderano semplificare il funzionamento delle applicazioni che utilizzano framework open source. Offre un avvio rapido dei processi, una gestione automatica della capacità e un controllo diretto dei costi.

Concetti

In questa sezione, trattiamo i termini e i concetti relativi al EMR sistema serverless presenti nella nostra Guida per l'utente EMR Serverless.

Versione di rilascio

Una EMR versione di Amazon è un insieme di applicazioni open source dell'ecosistema dei big data. Ogni versione include diverse applicazioni, componenti e funzionalità per i big data che scegli EMR Serverless per la distribuzione e la configurazione in modo che possano eseguire le tue applicazioni. Quando si crea un'applicazione, è necessario specificarne la versione di rilascio. Scegli la versione di EMR rilascio di Amazon e la versione del framework open source che desideri utilizzare nella tua applicazione. Per ulteriori informazioni sulle versioni preliminari, consultaVersioni di rilascio di Amazon EMR Serverless.

Applicazione

Con EMR Serverless, puoi creare una o più applicazioni EMR Serverless che utilizzano framework di analisi open source. Per creare un'applicazione, è necessario specificare i seguenti attributi:

- La versione di EMR rilascio di Amazon per la versione del framework open source che desideri utilizzare. Per determinare la tua versione di rilascio, consultaVersioni di rilascio di Amazon EMR Serverless.
- Il runtime specifico che desideri venga utilizzato dall'applicazione, ad esempio Apache Spark o Apache Hive.

Dopo aver creato un'applicazione, puoi inviare lavori di elaborazione dati o richieste interattive all'applicazione.

Ogni applicazione EMR Serverless viene eseguita su un Amazon Virtual Private Cloud (VPC) sicuro e distinto dalle altre applicazioni. Inoltre, puoi usare AWS Identity and Access Management (IAM) politiche per definire quali utenti e ruoli possono accedere all'applicazione. È inoltre possibile specificare limiti per controllare e tenere traccia dei costi di utilizzo sostenuti dall'applicazione.

Prendi in considerazione la possibilità di creare più applicazioni quando devi fare quanto segue:

- Utilizza diversi framework open source
- Utilizza versioni diverse di framework open source per diversi casi d'uso
- Esegui test A/B durante l'aggiornamento da una versione all'altra
- Mantieni ambienti logici separati per scenari di test e produzione
- Fornisci ambienti logici separati per diversi team con controlli dei costi e monitoraggio dell'utilizzo indipendenti
- Separa diverse line-of-business applicazioni

EMRServerless è un servizio regionale che semplifica il modo in cui i carichi di lavoro vengono eseguiti su più zone di disponibilità in una regione. Per ulteriori informazioni su come utilizzare le applicazioni con EMR Serverless, consulta. Interazione con un'applicazione

Esecuzione del processo

L'esecuzione di un processo è una richiesta inviata a un'applicazione EMR Serverless che l'applicazione esegue in modo asincrono e ne tiene traccia fino al completamento. Esempi di lavori

includono una query HiveQL che invii a un'applicazione Apache Hive o uno script di elaborazione dati che invii a PySpark un'applicazione Apache Spark. Quando inviate un lavoro, dovete specificare un ruolo di runtime, creato inIAM, che il lavoro utilizza per accedere AWS risorse, come oggetti Amazon S3. È possibile inviare più richieste di esecuzione di lavoro a un'applicazione e ogni esecuzione di lavoro può utilizzare un ruolo di runtime diverso per l'accesso AWS risorse. Un'applicazione EMR serverless inizia a eseguire i lavori non appena li riceve ed esegue più richieste di lavoro contemporaneamente. Per ulteriori informazioni su come EMR Serverless esegue i job, consulta. Esecuzione di processi

Worker

Un'applicazione EMR serverless utilizza internamente i lavoratori per eseguire i carichi di lavoro. Le dimensioni predefinite di questi worker si basano sul tipo di applicazione e sulla versione di EMR rilascio di Amazon. Quando pianifichi l'esecuzione di un lavoro, puoi sostituire queste dimensioni.

Quando si invia un lavoro, EMR Serverless calcola le risorse necessarie all'applicazione per il lavoro e pianifica i lavoratori. EMRServerless suddivide i carichi di lavoro in attività, scarica immagini, provvede e organizza i lavoratori e li disattiva al termine del lavoro. EMRServerless aumenta o riduce automaticamente i dipendenti in base al carico di lavoro e al parallelismo richiesti in ogni fase del lavoro. Questa scalabilità automatica elimina la necessità di stimare il numero di lavoratori necessari all'applicazione per eseguire i carichi di lavoro.

Capacità preinizializzata

EMRServerless offre una funzionalità di capacità preinizializzata che mantiene gli operatori inizializzati e pronti a rispondere in pochi secondi. Questa capacità crea in modo efficace un pool di lavoratori accogliente per un'applicazione. Per configurare questa funzionalità per ogni applicazione, impostate il `initial-capacity` parametro di un'applicazione. Quando si configura la capacità preinizializzata, i lavori possono iniziare immediatamente in modo da poter implementare applicazioni iterative e lavori urgenti. Per ulteriori informazioni sui worker preinizializzati, consulta. Configurazione di un'applicazione

EMRStudio

EMRStudio è la console utente che puoi utilizzare per gestire le tue applicazioni EMR Serverless. Se nel tuo account non è presente EMR uno Studio quando crei la tua prima applicazione EMR Serverless, ne creiamo automaticamente uno per te. Puoi accedere a EMR Studio dalla EMR console Amazon oppure puoi attivare l'accesso federato dal tuo provider di identità (IdP) IAM tramite IAM

Identity Center. In questo modo, gli utenti possono accedere a Studio e gestire le applicazioni EMR Serverless senza accesso diretto alla EMR console Amazon. Per saperne di più su come le applicazioni EMR Serverless funzionano con EMR Studio, consulta Interazione con l'applicazione dalla console Studio EMR e. Esecuzione di processi dalla console EMR Studio

Prerequisiti per iniziare a usare Serverless EMR

Argomenti

Registrati per un Account AWS

Se non disponi di un Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la https://portal.aws.amazon.com/billing/registrazione.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, un Utente root dell'account AWSviene creato. L'utente root ha accesso a tutti Servizi AWS e le risorse presenti nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. In qualsiasi momento, puoi visualizzare l'attività corrente del tuo account e gestirlo accedendo a https://aws.amazon.com/e scegliendo Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato per un Account AWS, proteggi il tuo Utente root dell'account AWS, abilita AWS IAM Identity Center e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi a AWS Management Consolecome proprietario dell'account selezionando Utente root e inserendo il Account AWS indirizzo email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso tramite utente root, consulta Accesso come utente root in Accedi ad AWS Guida per l'utente.
2. Attiva l'autenticazione a più fattori (MFA) per il tuo utente root.

   Per istruzioni, consulta Abilitare un MFA dispositivo virtuale per il Account AWS utente root (console) nella Guida per l'IAMutente.

Crea un utente con accesso amministrativo

1. Abilita IAM Identity Center.

   Per istruzioni, vedi Abilitazione AWS IAM Identity Center nella AWS IAM Identity Center Guida per l'utente.
2. In IAM Identity Center, concedi l'accesso amministrativo a un utente.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, vedi Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory nella AWS IAM Identity Center Guida per l'utente.

Accesso come utente amministratore

- Per accedere con il tuo utente IAM Identity Center, utilizza l'accesso URL che ti è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso tramite un utente di IAM Identity Center, vedi Accesso a AWS accedere al portale in Accedi ad AWS Guida per l'utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni che segua la migliore pratica di applicazione delle autorizzazioni con privilegi minimi.

Per istruzioni, vedere Creare un set di autorizzazioni nella AWS IAM Identity Center Guida per l'utente.

2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

   Per istruzioni, consulta Aggiungere gruppi nella AWS IAM Identity Center Guida per l'utente.

Concessione delle autorizzazioni

Negli ambienti di produzione, si consiglia di utilizzare politiche più dettagliate. Per esempi di tali politiche, vedere. Esempi di policy di accesso degli utenti per Serverless EMR Per ulteriori informazioni sulla gestione degli accessi, vedere Gestione degli accessi per AWS risorse nella Guida IAM per l'utente.

Per gli utenti che devono iniziare a usare EMR Serverless in un ambiente sandbox, utilizza una policy simile alla seguente:

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

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

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in AWS IAM Identity Center:

  Crea un set di autorizzazioni. Segui le istruzioni riportate in Creare un set di autorizzazioni nella AWS IAM Identity Center Guida per l'utente.

- Utenti gestiti IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate in Creazione di un ruolo per un provider di identità di terze parti (federazione) nella Guida per l'IAMutente.

- IAMutenti:
  - Crea un ruolo che l'utente possa assumere. Segui le istruzioni riportate nella sezione Creazione di un ruolo per un IAM utente nella Guida per l'IAMutente.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate in Aggiungere autorizzazioni a un utente (console) nella Guida per l'IAMutente.

Concessione dell'accesso programmatico

Gli utenti hanno bisogno dell'accesso programmatico se vogliono interagire con AWS al di fuori del AWS Management Console. Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede AWS.

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti in IAM Identity Center)	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, oppure AWS APIs.	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per il AWS CLI, vedi Configurazione di AWS CLI da usare AWS IAM Identity Center nella AWS Command Line Interface Guida per l'utente. • In AWS SDKs, strumenti e AWS APIs, vedi Autentica zione in IAM Identity Center nella AWS SDKse Guida di riferimento agli strumenti.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, oppure AWS APIs.	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con AWS risorse nella Guida per l'IAMutente.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI, AWS SDKs, oppure AWS APIs.	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per il AWS CLI, vedere Autenticazione tramite credenziali IAM utente

Quale utente necessita dell'accesso programmatico?	Per	Come
		nella AWS Command Line Interface Guida per l'utente. • In AWS SDKse strumenti, vedi Autenticazione tramite credenziali a lungo termine nel AWS SDKse Guida di riferimento agli strumenti. • In AWS APIs, vedi Gestione delle chiavi di accesso per IAM gli utenti nella Guida per l'IAMutente.

Installa e configura il AWS CLI

Se si desidera utilizzare EMR ServerlessAPIs, è necessario installare la versione più recente di AWS Command Line Interface (AWS CLI). Non hai bisogno del AWS CLI puoi utilizzare EMR Serverless dalla console EMR Studio e puoi iniziare senza di esso CLI seguendo la procedura riportata di seguito. Guida introduttiva a EMR Serverless dalla console

Per configurare il AWS CLI

1. Per installare la versione più recente di AWS CLI per macOS, Linux o Windows, vedi Installazione o aggiornamento della versione più recente di AWS CLI.
2. Per configurare il AWS CLI e configurazione sicura del tuo accesso a Servizi AWS, incluso EMR Serverless, vedi Configurazione rapida con `aws configure`.
3. Per verificare la configurazione, immettere il seguente DataBrew comando al prompt dei comandi.

   ```
   aws emr-serverless help
   ```

   AWS CLI i comandi utilizzano l'impostazione predefinita Regione AWS dalla tua configurazione, a meno che tu non la imposti con un parametro o un profilo. Per impostare il Regione AWS con un parametro, è possibile aggiungere il `--region` parametro a ciascun comando.

Per impostare il Regione AWS con un profilo, aggiungi prima un profilo denominato nel `~/.aws/config` file o nel `%UserProfile%/.aws/config` file (per Microsoft Windows). Segui la procedura descritta in Profili denominati per AWS CLI. Quindi, imposta il tuo Regione AWS e altre impostazioni con un comando simile a quello dell'esempio seguente.

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

Aprire la console

La maggior parte degli argomenti relativi alla console in questa sezione inizia dalla console Amazon EMR. Se non hai già effettuato l'accesso al tuo Account AWS, accedi, quindi apri la EMRconsole Amazon e passa alla sezione successiva per continuare a usare AmazonEMR.

Guida introduttiva ad Amazon EMR Serverless

Questo tutorial ti aiuta a iniziare a usare EMR Serverless quando distribuisci un esempio di carico di lavoro Spark o Hive. Potrai creare, eseguire ed eseguire il debug della tua applicazione. Mostriamo le opzioni predefinite nella maggior parte di questo tutorial.

Prima di avviare un'applicazione EMR Serverless, completa le seguenti attività.

Argomenti

Concedi le autorizzazioni per utilizzare Serverless EMR

Per utilizzare EMR Serverless, è necessario un utente o un IAM ruolo con una policy allegata che conceda le autorizzazioni per Serverless. EMR Per creare un utente e allegare la politica appropriata a quell'utente, segui le istruzioni riportate in. Concessione delle autorizzazioni

Prepara lo storage per EMR Serverless

In questo tutorial, utilizzerai un bucket S3 per archiviare i file di output e i log del carico di lavoro Spark o Hive di esempio che eseguirai utilizzando un'applicazione Serverless. EMR Per creare un bucket, segui le istruzioni in Creazione di un bucket nella Guida per l'utente della console di Amazon Simple Storage Service. Sostituisci ogni ulteriore riferimento a *DOC-EXAMPLE-BUCKET* con il nome del bucket appena creato.

Crea uno EMR Studio per eseguire carichi di lavoro interattivi

Se desideri utilizzare EMR Serverless per eseguire query interattive tramite notebook ospitati in EMR Studio, devi specificare un bucket S3 e il ruolo di servizio minimo per Serverless per creare

un workspace. EMR Per la procedura di configurazione, consulta Configurare uno EMR Studio nella Amazon EMR Management Guide. Per ulteriori informazioni sui carichi di lavoro interattivi, consultaEsegui carichi di lavoro interattivi con EMR Serverless tramite Studio EMR.

Creare un ruolo di job runtime

Job viene eseguito in EMR modalità Serverless, utilizza un ruolo di runtime che fornisce autorizzazioni granulari per specifici Servizi AWS e risorse in fase di esecuzione. In questo tutorial, un bucket S3 pubblico ospita i dati e gli script. Il bucket *DOC-EXAMPLE-BUCKET* memorizza l'output.

Per impostare un ruolo di job runtime, create innanzitutto un ruolo di runtime con una policy di fiducia in modo che EMR Serverless possa utilizzare il nuovo ruolo. Quindi, collega la politica di accesso S3 richiesta a quel ruolo. I seguenti passaggi ti guidano attraverso il processo.

Console

1. Passare alla console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra seleziona Ruoli.
3. Scegliere Crea ruolo.
4. Per il tipo di ruolo, scegli Politica di fiducia personalizzata e incolla la seguente politica di fiducia. Ciò consente ai lavori inviati alle tue applicazioni Amazon EMR Serverless di accedere ad altre Servizi AWS per tuo conto.

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

5. Scegli Avanti per accedere alla pagina Aggiungi autorizzazioni, quindi scegli Crea politica.
6. La pagina Crea policy si apre in una nuova scheda. Incolla la politica JSON qui sotto.

> **Important**
>
> Sostituisci *DOC-EXAMPLE-BUCKET* la politica seguente con il nome effettivo del bucket creato inPrepara lo storage per EMR Serverless. Questa è una politica di base per l'accesso a S3. Per altri esempi di ruoli di job runtime, consultaRuoli Job Runtime per Amazon EMR Serverless.

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

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

7. Nella pagina Rivedi la politica, inserisci un nome per la tua politica, ad esempio`EMRServerlessS3AndGlueAccessPolicy`.
8. Aggiorna la pagina Allega criteri di autorizzazione e scegli. `EMRServerlessS3AndGlueAccessPolicy`
9. Nella pagina Nome, rivedi e crea, in Nome ruolo, inserisci un nome per il tuo ruolo, ad esempio. `EMRServerlessS3RuntimeRole` Per creare questo IAM ruolo, scegli Crea ruolo.

CLI

1. Crea un file denominato `emr-serverless-trust-policy.json` che contenga la politica di fiducia da utilizzare per il IAM ruolo. Il file deve contenere la seguente politica.

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

```
}
```

2. Crea un IAM ruolo denominato`EMRServerlessS3RuntimeRole`. Utilizza la politica di fiducia che hai creato nel passaggio precedente.

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

Annota il valore ARN nell'output. Durante l'invio ARN del lavoro, si utilizza il nuovo ruolo, denominato successivamente. *`job-role-arn`*

3. Crea un file denominato `emr-sample-access-policy.json` che definisca la IAM politica per il tuo carico di lavoro. Ciò fornisce l'accesso in lettura allo script e ai dati archiviati nei bucket S3 pubblici e l'accesso in lettura/scrittura a. *`DOC-EXAMPLE-BUCKET`*

> **Important**
>
> *`DOC-EXAMPLE-BUCKET`*Sostituiscilo nella policy seguente con il nome effettivo del bucket creato in.. Prepara lo storage per EMR Serverless Questa è una politica di base per AWS Glue e accesso S3. Per altri esempi di ruoli in Job Runtime, consultaRuoli Job Runtime per Amazon EMR Serverless.

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

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

4. Create una IAM policy denominata `EMRServerlessS3AndGlueAccessPolicy` con il file di policy creato nel passaggio 3. Prendi nota dell'ARNoutput, poiché utilizzerai ARN la nuova politica nel passaggio successivo.

```
aws iam create-policy \
    --policy-name EMRServerlessS3AndGlueAccessPolicy \
```

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

Nota che la nuova politica è ARN nell'output. La sostituirai *policy-arn* nel passaggio successivo.

5. Allega la IAM policy `EMRServerlessS3AndGlueAccessPolicy` al ruolo `EMRServerlessS3RuntimeRole` di job runtime.

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

Guida introduttiva a EMR Serverless dalla console

Passaggi da completare

- Fase 1: Creare un'applicazione EMR serverless
- Fase 2: Inviare un job run o un carico di lavoro interattivo
- Passaggio 3: Visualizza l'interfaccia utente e i registri dell'applicazione
- Fase 4: pulizia

Fase 1: Creare un'applicazione EMR serverless

Crea una nuova applicazione con EMR Serverless come segue.

1. Accedi a AWS Management Console e apri la EMR console Amazon all'indirizzo https://console.aws.amazon.com/emr.
2. Nel riquadro di navigazione a sinistra, scegli EMRServerless per accedere alla landing page EMR Serverless.
3. Per creare o gestire applicazioni EMR Serverless, è necessaria l'EMRinterfaccia utente di Studio.
   - Se hai già uno EMR Studio in Regione AWS dove desideri creare un'applicazione, seleziona Gestisci applicazioni per accedere a EMR Studio oppure seleziona lo studio che desideri utilizzare.
   - Se non disponi di uno EMR Studio in Regione AWS dove vuoi creare un'applicazione, scegli Inizia, quindi scegli Crea e avvia Studio. EMRServerless crea uno EMR Studio per te in modo che tu possa creare e gestire applicazioni.

4. Nell'interfaccia utente di Create studio che si apre in una nuova scheda, inserisci il nome, il tipo e la versione di rilascio dell'applicazione. Se desideri eseguire solo processi batch, seleziona Usa le impostazioni predefinite solo per i processi batch. Per i carichi di lavoro interattivi, seleziona Usa le impostazioni predefinite per i carichi di lavoro interattivi. Con questa opzione puoi anche eseguire lavori in batch su applicazioni abilitate all'interattività. Se necessario, è possibile modificare queste impostazioni in un secondo momento.

   Per ulteriori informazioni, consulta Creare uno studio.
5. Seleziona Crea applicazione per creare la tua prima applicazione.

Passa alla sezione successiva Fase 2: Inviare un job run o un carico di lavoro interattivo per inviare un job run o un carico di lavoro interattivo.

Fase 2: Inviare un job run o un carico di lavoro interattivo

Spark job run

In questo tutorial, utilizziamo uno PySpark script per calcolare il numero di occorrenze di parole uniche in più file di testo. Un bucket S3 pubblico di sola lettura memorizza sia lo script che il set di dati.

Per eseguire un job Spark

1. Carica lo script di esempio `wordcount.py` nel tuo nuovo bucket con il seguente comando.

   ```
   aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
   scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
   ```
2. Fase 1: Creare un'applicazione EMR serverlessIl completamento porta alla pagina dei dettagli dell'applicazione in EMR Studio. Qui, scegli l'opzione Invia lavoro.
3. Nella pagina Invia lavoro, completa quanto segue.
   - Nel campo Nome, inserisci il nome con cui vuoi chiamare il job run.
   - Nel campo Runtime role, inserisci il nome del ruolo in cui hai creatoCreare un ruolo di job runtime.
   - Nel campo Posizione dello script, inserisci `s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py` come S3URI.

- Nel campo Argomenti dello script, inserisci`["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/output"]`.
- Nella sezione delle proprietà di Spark, scegli Modifica come testo e inserisci le seguenti configurazioni.

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

4. Per avviare l'esecuzione del lavoro, scegli Invia lavoro.
5. Nella scheda Job run, dovresti vedere il tuo nuovo job eseguito con lo stato Running.

Hive job run

In questa parte del tutorial, creiamo una tabella, inseriamo alcuni record ed eseguiamo una query di aggregazione del conteggio. Per eseguire il job Hive, devi prima creare un file che contenga tutte le query Hive da eseguire come parte di un singolo job, carica il file su S3 e specifica questo percorso S3 all'avvio del job Hive.

Per eseguire un job Hive

1. Crea un file chiamato `hive-query.ql` che contenga tutte le query che desideri eseguire nel tuo job Hive.

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

2. Carica `hive-query.ql` nel tuo bucket S3 con il seguente comando.

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

3. Fase 1: Creare un'applicazione EMR serverlessIl completamento porta alla pagina dei dettagli dell'applicazione in EMR Studio. Qui, scegli l'opzione Invia lavoro.
4. Nella pagina Invia lavoro, completa quanto segue.

- Nel campo Nome, inserisci il nome con cui vuoi chiamare il job run.
- Nel campo Runtime role, inserisci il nome del ruolo in cui hai creatoCreare un ruolo di job runtime.
- Nel campo Posizione dello script, inserisci `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql` come S3URI.
- Nella sezione delle proprietà di Hive, scegli Modifica come testo e inserisci le seguenti configurazioni.

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

- Nella sezione Configurazione Job, scegli Modifica come JSON e inserisci quanto segueJSON.

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

5. Per avviare l'esecuzione del lavoro, scegli Invia lavoro.
6. Nella scheda Job run, dovresti vedere il tuo nuovo job eseguito con lo stato Running.

Interactive workload

Con Amazon EMR 6.14.0 e versioni successive, puoi utilizzare notebook ospitati in EMR Studio per eseguire carichi di lavoro interattivi per Spark in modalità Serverless. EMR Per ulteriori informazioni, tra cui autorizzazioni e prerequisiti, consulta. Esegui carichi di lavoro interattivi con EMR Serverless tramite Studio EMR

Dopo aver creato l'applicazione e impostato le autorizzazioni richieste, utilizzate i seguenti passaggi per eseguire un taccuino interattivo con Studio: EMR

1. Passa alla scheda Workspaces in EMR Studio. Se devi ancora configurare una posizione di archiviazione di Amazon S3 e il ruolo del servizio EMR Studio, seleziona il pulsante Configura studio nel banner nella parte superiore dello schermo.
2. Per accedere a un notebook, seleziona un Workspace o crea un nuovo Workspace. Usa Avvio veloce per aprire il tuo spazio di lavoro in una nuova scheda.
3. Vai alla scheda appena aperta. Seleziona l'icona Compute dalla barra di navigazione a sinistra. Seleziona EMR Serverless come tipo di elaborazione.
4. Seleziona l'applicazione interattiva che hai creato nella sezione precedente.
5. Nel campo Runtime role, inserisci il nome del IAM ruolo che l'applicazione EMR Serverless può assumere per l'esecuzione del job. Per ulteriori informazioni sui ruoli di runtime, consulta Job runtime roles nella Amazon EMR Serverless User Guide.
6. Seleziona Allega. Questa operazione potrebbe richiedere fino a un minuto. La pagina verrà aggiornata una volta allegata.
7. Scegli un kernel e avvia un notebook. Puoi anche sfogliare taccuini di esempio su EMR Serverless e copiarli nel tuo Workspace. Per accedere ai taccuini di esempio, accedi al **`{...}`**menu nella barra di navigazione a sinistra e sfoglia i taccuini che contengono il nome del file del taccuino. `serverless`
8. Nel notebook, puoi accedere al collegamento al registro dei driver e a un collegamento all'interfaccia utente di Apache Spark, un'interfaccia in tempo reale che fornisce metriche per monitorare il tuo lavoro. Per ulteriori informazioni, consulta Monitoring EMR Serverless Applications and Job nella Amazon EMR Serverless User Guide.

Quando colleghi un'applicazione a un'area di lavoro di Studio, l'avvio dell'applicazione si attiva automaticamente se non è già in esecuzione. È inoltre possibile preavviare l'applicazione e tenerla pronta prima di collegarla all'area di lavoro.

Passaggio 3: Visualizza l'interfaccia utente e i registri dell'applicazione

Per visualizzare l'interfaccia utente dell'applicazione, identificate innanzitutto il job eseguito. Un'opzione per l'interfaccia utente Spark o l'interfaccia utente Hive Tez è disponibile nella prima riga di opzioni per l'esecuzione del processo, in base al tipo di lavoro. Seleziona l'opzione appropriata.

Se hai scelto l'interfaccia utente Spark, scegli la scheda Executors per visualizzare i log dei driver e degli executors. Se hai scelto l'interfaccia utente di Hive Tez, scegli la scheda Tutte le attività per visualizzare i log.

Una volta che lo stato di esecuzione del processo risulta completato, puoi visualizzare l'output del processo nel tuo bucket S3.

Fase 4: pulizia

Sebbene l'applicazione che hai creato dovrebbe interrompersi automaticamente dopo 15 minuti di inattività, ti consigliamo comunque di rilasciare risorse che non intendi utilizzare più.

Per eliminare l'applicazione, vai alla pagina Elenco applicazioni. Seleziona l'applicazione che hai creato e scegli Azioni → Stop per interrompere l'applicazione. Dopo che l'applicazione è nello `STOPPED` stato, seleziona la stessa applicazione e scegli Azioni → Elimina.

Per altri esempi di esecuzione dei job Spark e Hive, consulta Offerte di lavoro Spark e. Offerte di lavoro Hive

Iniziare da AWS CLI

Passaggio 1: creare un'applicazione EMR serverless

Usa il `emr-serverless create-application`comando per creare la tua prima applicazione EMR serverless. Devi specificare il tipo di applicazione e l'etichetta di EMR rilascio Amazon associata alla versione dell'applicazione che desideri utilizzare. Il nome dell'applicazione è facoltativo.

Spark

Per creare un'applicazione Spark, esegui il comando seguente.

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

Hive

Per creare un'applicazione Hive, esegui il comando seguente.

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

Annotate l'ID dell'applicazione restituito nell'output. Utilizzerai l'ID per avviare la candidatura e durante l'invio del lavoro, in seguito denominato. *application-id*

Prima di passare aFase 2: Inviare un job eseguito all'applicazione EMR Serverless, assicurati che la tua candidatura abbia raggiunto lo CREATED stato con. get-applicationAPI

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

EMRServerless crea lavoratori per soddisfare i lavori richiesti. Per impostazione predefinita, questi vengono creati su richiesta, ma è anche possibile specificare una capacità preinizializzata impostando il initialCapacity parametro al momento della creazione dell'applicazione. È inoltre possibile limitare la capacità massima totale che un'applicazione può utilizzare con il maximumCapacity parametro. Per ulteriori informazioni su queste opzioni, consulta Configurazione di un'applicazione.

Fase 2: Inviare un job eseguito all'applicazione EMR Serverless

Ora la tua applicazione EMR Serverless è pronta per eseguire i lavori.

Spark

In questo passaggio, utilizziamo uno PySpark script per calcolare il numero di occorrenze di parole uniche in più file di testo. Un bucket S3 pubblico di sola lettura memorizza sia lo script che il set di dati. L'applicazione invia il file di output e i dati di registro dal runtime Spark /output e alle /logs directory del bucket S3 che hai creato.

Per eseguire un job Spark

1. Usa il seguente comando per copiare lo script di esempio che eseguiremo nel tuo nuovo bucket.

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

2. Nel comando seguente, sostituiscilo *application-id* con l'ID dell'applicazione. Sostituiscilo *job-role-arn* con il ruolo di runtime in ARN cui hai creato. Creare un ruolo di job runtime Sostituto *job-run-name* con il nome che vuoi chiamare job run. Sostituisci tutte *DOC-EXAMPLE-BUCKET* le stringhe con il bucket Amazon S3 che hai creato e `/output` aggiungile al percorso. Questo crea una nuova cartella nel bucket in cui EMR Serverless può copiare i file di output dell'applicazione.

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

3. Annota l'ID di esecuzione del processo restituito nell'output. Sostituiscilo *job-run-id* con questo ID nei passaggi seguenti.

Hive

In questo tutorial, creiamo una tabella, inseriamo alcuni record ed eseguiamo una query di aggregazione del conteggio. Per eseguire il job Hive, devi prima creare un file che contenga tutte le query Hive da eseguire come parte di un singolo job, carica il file su S3 e specifica questo percorso S3 all'avvio del job Hive.

Per eseguire un lavoro Hive

1. Crea un file chiamato `hive-query.ql` che contenga tutte le query che desideri eseguire nel tuo job Hive.

```
create database if not exists emrserverless;
use emrserverless;
create table if not exists test_table(id int);
drop table if exists Values__Tmp__Table__1;
```

```
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;
```

2. Carica `hive-query.ql` nel tuo bucket S3 con il seguente comando.

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

3. Nel comando seguente, sostituiscilo *`application-id`* con il tuo ID dell'applicazione. Sostituiscilo *`job-role-arn`* con il ruolo di runtime in ARN cui hai creato. Creare un ruolo di job runtime Sostituisci tutte *`DOC-EXAMPLE-BUCKET`* le stringhe con il bucket Amazon S3 che hai creato e `/output` aggiungi `/logs` e al percorso. In questo modo vengono create nuove cartelle nel bucket, in cui EMR Serverless può copiare i file di output e di registro dell'applicazione.

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

```
            }
        }
    }'
```

4. Annotate l'ID di esecuzione del processo restituito nell'output. Sostituiscilo *job-run-id* con questo ID nei passaggi seguenti.

Fase 3: Esaminate l'output dell'esecuzione del job

Il completamento dell'esecuzione del processo dovrebbe in genere richiedere 3-5 minuti.

Spark

Puoi controllare lo stato del tuo job Spark con il seguente comando.

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

Con la destinazione dei log impostata sus3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs, puoi trovare i log relativi a questo specifico job in cui viene eseguito. s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs/applications/*application-id*/jobs/*job-run-id*

Per le applicazioni Spark, EMR Serverless invia i registri degli eventi ogni 30 secondi sparklogs nella cartella nella destinazione dei registri S3. Al termine del processo, i log di runtime di Spark relativi al driver e agli esecutori vengono caricati in cartelle denominate in modo appropriato in base al tipo di lavoratore, ad esempio o. driver executor L'output del lavoro viene caricato su. PySpark s3://*DOC-EXAMPLE-BUCKET*/output/

Hive

Puoi controllare lo stato del tuo lavoro in Hive con il seguente comando.

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

Con la destinazione del registro impostata sus3://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs, puoi trovare i log relativi a questo specifico processo in

cui viene eseguito. `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs/applications/application-id/jobs/job-run-id`

Per le applicazioni Hive, EMR Serverless carica continuamente il driver Hive `HIVE_DRIVER` nella cartella e i registri delle attività Tez nella `TEZ_TASK` cartella della destinazione del registro S3. Dopo che l'esecuzione del processo raggiunge `SUCCEEDED` lo stato, l'output della tua query Hive diventa disponibile nella posizione Amazon S3 che hai specificato `monitoringConfiguration` nel campo di. `configurationOverrides`

Fase 4: pulizia

Quando hai finito di lavorare con questo tutorial, valuta la possibilità di eliminare le risorse che hai creato. Ti consigliamo di rilasciare risorse che non intendi riutilizzare.

Elimina la tua applicazione

Per eliminare un'applicazione, utilizzare il seguente comando.

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

Elimina il tuo bucket di log S3

Per eliminare il bucket di registrazione e output S3, usa il seguente comando. Sostituisci *DOC-EXAMPLE-BUCKET* con il nome effettivo del bucket S3 creato in.. Prepara lo storage per EMR Serverless

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

Elimina il ruolo Job Runtime

Per eliminare il ruolo di runtime, scollega la policy dal ruolo. È quindi possibile eliminare sia il ruolo che la politica.

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

Per eliminare il ruolo, utilizzare il comando seguente.

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

Per eliminare la politica allegata al ruolo, utilizzare il comando seguente.

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

Per altri esempi di esecuzione dei job Spark e Hive, consulta Offerte di lavoro Spark e. Offerte di lavoro Hive

Interazione con un'applicazione

Questa sezione spiega come interagire con la tua applicazione Amazon EMR Serverless con AWS CLI e le impostazioni predefinite per i motori Spark e Hive.

Argomenti

Stati dell'applicazione

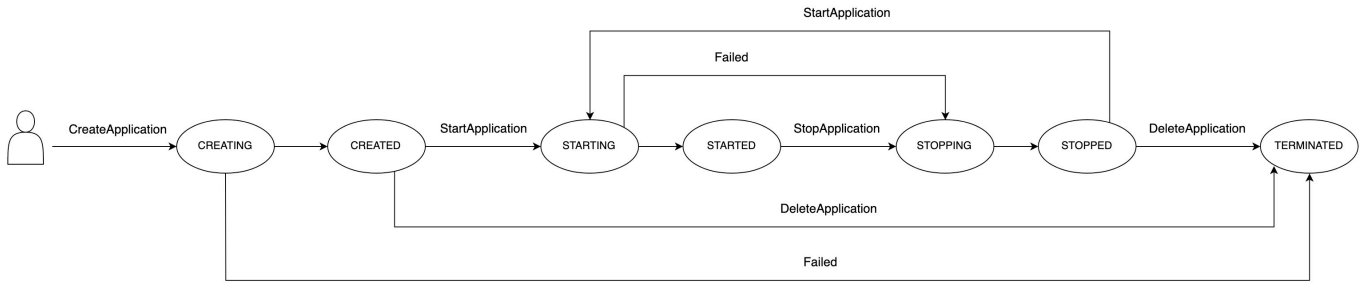
Quando si crea un'applicazione con EMR Serverless, l'esecuzione dell'applicazione entra nello stato. `CREATING` Dopodiché passa per i seguenti stati fino a quando non ha esito positivo (codice di uscita `0`) o negativo (codice di uscita diverso da zero).

Le applicazioni possono avere i seguenti stati:

Stato	Descrizione
Creating (Creazione in corso)	L'applicazione è in fase di preparazione e non è ancora pronta per l'uso.
Creato	L'applicazione è stata creata ma non ha ancora fornito capacità. È possibile modificare l'applica zione per modificare la configurazione iniziale della capacità.
Avvio di	L'applicazione si sta avviando e sta fornendo capacità.

Stato	Descrizione
Avviato	L'applicazione è pronta per accettare nuovi lavori. L'applicazione accetta offerte di lavoro solo quando si trova in questo stato.
Stopping (In arresto)	Tutti i lavori sono stati completati e l'applica zione sta esaurendo la sua capacità.
Arrestate	L'applicazione è arrestata e nessuna risorsa è in esecuzione sull'applicazione. È possibile modificare l'applicazione per cambiarne la configurazione iniziale della capacità.
Terminata	L'applicazione è stata chiusa e non appare nell'elenco delle applicazioni.

Il diagramma seguente mostra la traiettoria degli stati delle applicazioni Serverless. EMR



Interazione con l'applicazione dalla console Studio EMR

Dalla console EMR Studio, puoi creare, visualizzare e gestire applicazioni EMR serverless. Per accedere alla console EMR Studio, segui le istruzioni riportate in Guida introduttiva alla console.

Creazione di un'applicazione

Con la pagina Crea applicazione, puoi creare un'applicazione EMR Serverless seguendo questi passaggi.

1. Nel campo Nome, inserisci il nome con cui vuoi chiamare l'applicazione.
2. Nel campo Tipo, scegli Spark o Hive come tipo di applicazione.
3. Nel campo Versione di rilascio, scegli il numero della EMR release.
4. Nelle opzioni Architettura, scegli l'architettura del set di istruzioni da utilizzare. Per ulteriori informazioni, consulta Opzioni di architettura Amazon EMR Serverless.

   - arm64 — ARM architettura a 64 bit; per usare i processori Graviton
   - x86_64 — architettura x86 a 64 bit; per usare processori basati su x86
5. Esistono due opzioni di configurazione dell'applicazione per i campi rimanenti: impostazioni predefinite e impostazioni personalizzate. Questi campi sono opzionali.

   Impostazioni predefinite: le impostazioni predefinite consentono di creare rapidamente un'applicazione con capacità preinizializzata. Ciò include un driver e un executor per Spark e un driver e un Tez Task per Hive. Le impostazioni predefinite non abilitano la connettività di rete al tuo. VPCs L'applicazione è configurata per arrestarsi se inattiva per 15 minuti e si avvia automaticamente all'invio del lavoro.

   Impostazioni personalizzate: le impostazioni personalizzate consentono di modificare le seguenti proprietà.

   - Capacità preinizializzata: il numero di driver ed esecutori o di Hive Tez Task worker e le dimensioni di ciascun lavoratore.
   - Limiti delle applicazioni: la capacità massima di un'applicazione.
   - Comportamento dell'applicazione: il comportamento di avvio e arresto automatico dell'applicazione.
   - Connessioni di rete: connettività di rete alle risorse. VPC
   - Tag: tag personalizzati che è possibile assegnare all'applicazione.

   Per ulteriori informazioni sulla capacità preinizializzata, sui limiti delle applicazioni e sul comportamento delle applicazioni, consulta. Configurazione di un'applicazione Per ulteriori informazioni sulla connettività di rete, vedere. Configurazione dell'accesso VPC
6. Per creare l'applicazione, scegli Crea applicazione.

Elenca le applicazioni

È possibile visualizzare tutte le applicazioni EMR Serverless esistenti dalla pagina Elenco applicazioni. È possibile scegliere il nome di un'applicazione per accedere alla pagina dei dettagli dell'applicazione.

Gestione delle applicazioni

È possibile eseguire le seguenti azioni su un'applicazione dalla pagina Elenco applicazioni o dalla pagina Dettagli di un'applicazione specifica.

Avvia l'applicazione

Scegliete questa opzione per avviare manualmente un'applicazione.

Interrompi l'applicazione

Scegliete questa opzione per interrompere manualmente un'applicazione. Un'applicazione non deve avere processi in esecuzione per poter essere interrotta. Per ulteriori informazioni sulle transizioni di stato delle applicazioni, consultaStati dell'applicazione.

Configurare l'applicazione

Modifica le impostazioni opzionali per un'applicazione dalla pagina Configura applicazione. È possibile modificare la maggior parte delle impostazioni dell'applicazione. Ad esempio, puoi modificare l'etichetta di rilascio per un'applicazione per aggiornarla a una versione diversa di Amazon EMR oppure puoi cambiare l'architettura da x86_64 a arm64. Le altre impostazioni opzionali sono le stesse presenti nella sezione Impostazioni personalizzate nella pagina Crea applicazione. Per ulteriori informazioni sulle impostazioni dell'applicazione, vedereCreazione di un'applicazione.

Eliminare l'applicazione

Scegliete questa opzione per eliminare manualmente un'applicazione. È necessario interrompere un'applicazione per eliminarla. Per ulteriori informazioni sulle transizioni di stato delle applicazioni, consultaStati dell'applicazione.

Interagire con l'applicazione su AWS CLI

Da AWS CLI, è possibile creare, descrivere ed eliminare singole applicazioni. Puoi anche elencare tutte le tue applicazioni in modo da poterle visualizzare a colpo d'occhio. Questa sezione descrive

come eseguire queste azioni. Per ulteriori operazioni sull'applicazione, come l'avvio, l'arresto e l'aggiornamento dell'applicazione, consulta il EMRServerless API Reference. Per esempi di come utilizzare EMR Serverless API utilizzando il AWS SDK for Java, consulta gli esempi di Java nel nostro GitHub repository. Per esempi di come utilizzare EMR Serverless API utilizzando il AWS SDK for Python (Boto), vedi esempi di Python nel nostro GitHub repository.

Per creare un'applicazione, usa. `create-application` È necessario specificare `SPARK` o `HIVE` come applicazione`type`. Questo comando restituisce l'applicazione`ARN`, il nome e l'ID.

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

Per descrivere un'applicazione, usala `get-application` e forniscila`application-id`. Questo comando restituisce le configurazioni relative allo stato e alla capacità dell'applicazione.

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

Per elencare tutte le tue applicazioni, chiama. `list-applications` Questo comando restituisce le stesse proprietà `get-application` ma include tutte le applicazioni.

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

Per eliminare la tua applicazione, chiama `delete-application` e fornisci il tuo`application-id`.

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

Configurazione di un'applicazione

Con EMR Serverless, puoi configurare le applicazioni che utilizzi. Ad esempio, è possibile impostare la capacità massima fino a cui un'applicazione può scalare, configurare la capacità preinizializzata per mantenere il conducente e gli operatori pronti a rispondere e specificare un set comune di configurazioni di runtime e monitoraggio a livello di applicazione. Le pagine seguenti descrivono come configurare le applicazioni quando si utilizza Serverless. EMR

Argomenti

Comprensione del comportamento delle applicazioni

Comportamento predefinito dell'applicazione

Avvio automatico: per impostazione predefinita, un'applicazione è configurata per l'avvio automatico all'invio del lavoro. È possibile disattivare questa funzionalità.

Arresto automatico: per impostazione predefinita, un'applicazione è configurata per l'arresto automatico quando è inattiva per 15 minuti. Quando un'applicazione passa allo `STOPPED` stato, rilascia qualsiasi capacità preinizializzata configurata. È possibile modificare la quantità di tempo di inattività prima dell'arresto automatico di un'applicazione oppure disattivare questa funzionalità.

Capacità massima

È possibile configurare la capacità massima fino alla quale un'applicazione può scalare. È possibile specificare la capacità massima in termini di CPU memoria (GB) e disco (GB).

> **Note**
>
> Si consiglia di configurare la capacità massima in modo proporzionale alle dimensioni dei lavoratori supportate moltiplicando il numero di lavoratori per le loro dimensioni. Ad esempio, se desiderate limitare l'applicazione a 50 worker con vCPUs 2.16 GB di memoria e 20 GB per disco, impostate la capacità massima su vCPUs 100.800 GB per la memoria e 1000 GB per il disco.

Configurazioni dei lavoratori supportate

La tabella seguente mostra le configurazioni e le dimensioni dei worker supportate che è possibile specificare per EMR Serverless. È possibile configurare dimensioni diverse per driver ed esecutori in base alle esigenze del carico di lavoro.

CPU	Memoria	Archiviazione temporanea predefinita
1 v CPU	Minimo 2 GB, massimo 8 GB, con incrementi di 1 GB	20 GB - 200 GB
2 v CPU	Minimo 4 GB, massimo 16 GB, con incrementi di 1 GB	20 GB - 200 GB
4 v CPU	Minimo 8 GB, massimo 30 GB, con incrementi di 1 GB	20 GB - 200 GB
8 v CPU	Minimo 16 GB, massimo 60 GB, con incrementi di 4 GB	20 GB - 200 GB
16 v CPU	Minimo 32 GB, massimo 120 GB, con incrementi di 8 GB	20 GB - 200 GB

CPU— Ogni lavoratore può avere 1, 2, 4, 8 o 16vCPUs.

Memoria: ogni lavoratore dispone di memoria, specificata in GB, entro i limiti elencati nella tabella precedente. I job Spark hanno un sovraccarico di memoria, il che significa che la memoria che usano è superiore alle dimensioni del contenitore specificate. Questo sovraccarico è specificato con le proprietà e. `spark.driver.memory0verhead` `spark.executor.memory0verhead` L'overhead ha un valore predefinito del 10% della memoria del contenitore, con un minimo di 384 MB. È necessario considerare questo sovraccarico quando si scelgono le dimensioni dei lavoratori.

Ad esempio, se scegli 4 vCPUs per l'istanza di lavoro e una capacità di storage preinizializzata di 30 GB, dovresti impostare un valore di circa 27 GB come memoria executor per il tuo job Spark. Ciò massimizza l'utilizzo della capacità preinizializzata. La memoria utilizzabile sarebbe di 27 GB, più il 10% di 27 GB (2,7 GB), per un totale di 29,7 GB.

Disco: è possibile configurare ogni lavoratore con dischi di archiviazione temporanei con una dimensione minima di 20 GB e un massimo di 200 GB. Paghi solo lo spazio di archiviazione aggiuntivo oltre i 20 GB configurato per lavoratore.

Capacità preinizializzata

EMRServerless offre una funzionalità opzionale che mantiene conducente e addetti preinizializzati e pronti a rispondere in pochi secondi. In questo modo si crea in modo efficace un pool di lavoratori accogliente per un'applicazione. Questa funzionalità è denominata capacità preinizializzata. Per configurare questa funzionalità, è possibile impostare il `initialCapacity` parametro di un'applicazione sul numero di lavoratori che si desidera preinizializzare. Con la capacità dei lavoratori preinizializzata, i lavori iniziano immediatamente. Questa soluzione è ideale quando si desidera implementare applicazioni iterative e lavori urgenti.

Quando si invia un lavoro, se `initialCapacity` sono disponibili lavoratori di, il lavoro utilizza tali risorse per avviarne l'esecuzione. Se tali lavoratori sono già utilizzati da altri lavori o se il lavoro richiede più risorse di quelle disponibili`initialCapacity`, l'applicazione richiede e ottiene lavoratori aggiuntivi, fino ai limiti massimi di risorse impostati per l'applicazione. Al termine dell'esecuzione, un processo rilascia i worker utilizzati e il numero di risorse disponibili per l'applicazione torna a essere lo stesso`initialCapacity`. Un'applicazione mantiene le `initialCapacity` risorse anche dopo che i job hanno terminato l'esecuzione. L'applicazione rilascia risorse in eccesso oltre il periodo in `initialCapacity` cui i job non ne hanno più bisogno per l'esecuzione.

La capacità preinizializzata è disponibile e pronta all'uso all'avvio dell'applicazione. La capacità preinizializzata diventa inattiva quando l'applicazione viene arrestata. Un'applicazione passa allo `STARTED` stato solo se la capacità preinizializzata richiesta è stata creata ed è pronta per l'uso. Per tutto il tempo in cui l'applicazione è nello `STARTED` stato, EMR Serverless mantiene la capacità preinizializzata disponibile per l'uso o l'uso da parte di job o carichi di lavoro interattivi. La funzionalità ripristina la capacità dei contenitori rilasciati o guasti. Ciò mantiene il numero di lavoratori specificato dal `InitialCapacity` parametro. Lo stato di un'applicazione senza capacità preinizializzata può cambiare immediatamente da a. `CREATED` `STARTED`

È possibile configurare l'applicazione per rilasciare la capacità preinizializzata se non viene utilizzata per un determinato periodo di tempo, con un valore predefinito di 15 minuti. Una candidatura interrotta si avvia automaticamente quando invii un nuovo lavoro. È possibile impostare queste configurazioni di avvio e arresto automatiche quando si crea l'applicazione oppure modificarle quando l'applicazione è in uno `STOPPED` stato `CREATED` or.

È possibile modificare i `InitialCapacity` conteggi e specificare configurazioni di calcoloCPU, ad esempio memoria e disco, per ogni lavoratore. Poiché non è possibile apportare modifiche parziali, è necessario specificare tutte le configurazioni di calcolo quando si modificano i valori. È possibile modificare le configurazioni solo quando l'applicazione è nello stato o. `CREATED` `STOPPED`

> **Note**
>
> Per ottimizzare l'uso delle risorse da parte dell'applicazione, consigliamo di allineare le dimensioni dei container alle dimensioni dei dipendenti con capacità preinizializzata. Ad esempio, se configuri la dimensione dell'esecutore Spark su 2 CPUs e la memoria su 8 GB, ma la capacità preinizializzata del worker è 4 CPUs con 16 GB di memoria, gli esecutori Spark utilizzano solo la metà delle risorse dei lavoratori quando vengono assegnati a questo lavoro.

Personalizzazione della capacità preinizializzata per Spark e Hive

Puoi personalizzare ulteriormente la capacità preinizializzata per i carichi di lavoro eseguiti su specifici framework di big data. Ad esempio, quando un carico di lavoro viene eseguito su Apache Spark, puoi specificare quanti lavoratori iniziano come driver e quanti come esecutori. Allo stesso modo, quando si utilizza Apache Hive, è possibile specificare quanti lavoratori si avviano come driver Hive e quanti devono eseguire le attività Tez.

Configurazione di un'applicazione che esegue Apache Hive con capacità preinizializzata

La seguente API richiesta crea un'applicazione che esegue Apache Hive basata sulla EMR versione Amazon emr-6.6.0. L'applicazione inizia con 5 driver Hive preinizializzati, ciascuno con 2 v CPU e 4 GB di memoria, e 50 task worker Tez preinizializzati, ciascuno con 4 v e 8 GB di memoria. CPU Quando le query Hive vengono eseguite su questa applicazione, utilizzano innanzitutto i worker preinizializzati e iniziano l'esecuzione immediatamente. Se tutti i worker preinizializzati sono occupati e vengono inviati più lavori Hive, l'applicazione può scalare fino a un totale di 400 v e 1024 GB di memoria. CPU Facoltativamente, è possibile omettere la capacità per il lavoratore o per il lavoratore. `DRIVER` `TEZ_TASK`

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB"
```

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

Configurazione di un'applicazione che esegue Apache Spark con capacità preinizializzata

La seguente API richiesta crea un'applicazione che esegue Apache Spark 3.2.0 basata sulla versione 6.6.0 di AmazonEMR. L'applicazione inizia con 5 driver Spark preinizializzati, ciascuno con 2 v CPU e 4 GB di memoria, e 50 executor preinizializzati, ciascuno con 4 v e 8 GB di memoria. CPU Quando i job Spark vengono eseguiti su questa applicazione, utilizzano innanzitutto i worker preinizializzati e iniziano a essere eseguiti immediatamente. Se tutti i worker preinizializzati sono occupati e vengono inviati più job Spark, l'applicazione può scalare fino a un totale di 400 v CPU e 1024 GB di memoria. Facoltativamente, puoi omettere la capacità per il o il. `DRIVER` `EXECUTOR`

> **Note**
>
> Spark aggiunge un sovraccarico di memoria configurabile, con un valore predefinito del 10%, alla memoria richiesta per driver ed esecutori. Affinché i job utilizzino worker preinizializzati, la configurazione della memoria con capacità iniziale deve essere maggiore della memoria richiesta dal job e dall'overhead.

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
```

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

Configurazione predefinita dell'applicazione per Serverless EMR

È possibile specificare un set comune di configurazioni di runtime e monitoraggio a livello di applicazione per tutti i lavori inviati nella stessa applicazione. Ciò riduce il sovraccarico aggiuntivo associato alla necessità di inviare le stesse configurazioni per ogni lavoro.

È possibile modificare le configurazioni nei seguenti momenti:

- Dichiarare le configurazioni a livello di applicazione al momento dell'invio del lavoro.
- Sostituisci le configurazioni predefinite durante l'esecuzione del lavoro.

Le sezioni seguenti forniscono maggiori dettagli e un esempio per un ulteriore contesto.

Dichiarazione delle configurazioni a livello di applicazione

È possibile specificare le proprietà di registrazione a livello di applicazione e di configurazione di runtime per i lavori inviati tramite l'applicazione.

`monitoringConfiguration`

Per specificare le configurazioni di registro per i lavori inviati con l'applicazione, utilizza il campo. `monitoringConfiguration` Per ulteriori informazioni sulla registrazione per EMR Serverless, vedere. Archiviazione dei log

`runtimeConfiguration`

Per specificare proprietà di configurazione in fase di esecuzione`spark-defaults`, ad esempio, fornire un oggetto di configurazione nel `runtimeConfiguration` campo. Ciò influisce sulle configurazioni predefinite per tutti i lavori inviati con l'applicazione. Per ulteriori informazioni, consulta Parametro di sovrascrittura della configurazione Hive e La configurazione di Spark sovrascrive il parametro.

Le classificazioni di configurazione disponibili variano in base alla specifica versione EMR Serverless. Ad esempio, le classificazioni per Log4j personalizzate `spark-executor-log4j2` sono disponibili solo con le versioni 6.8.0 `spark-driver-log4j2` e successive. Per un elenco delle proprietà specifiche dell'applicazione, vedere e. Proprietà del lavoro Spark Proprietà del lavoro di Hive

Puoi anche configurare le proprietà di Apache Log4j2, AWS Secrets Manager per la protezione dei dati e il runtime di Java 17 a livello di applicazione.

Per trasmettere i segreti di Secrets Manager a livello di applicazione, allega la seguente policy agli utenti e ai ruoli che devono creare o aggiornare applicazioni EMR Serverless con segreti.

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

Per ulteriori informazioni sulla creazione di policy personalizzate per i segreti, consulta Esempi di policy di autorizzazione per AWS Secrets Manager nella AWS Secrets Manager Guida per l'utente.

> **Note**
>
> `runtimeConfiguration`Quello specificato a livello di applicazione è `applicationConfiguration` mappato in `StartJobRun`API.

Dichiarazione di esempio

L'esempio seguente mostra come dichiarare le configurazioni predefinite con. `create-application`

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

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

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

Sovrascrivere le configurazioni durante l'esecuzione di un lavoro

È possibile specificare sostituzioni di configurazione per la configurazione dell'applicazione e la configurazione di monitoraggio con. `StartJobRun`API EMRServerless unisce quindi le configurazioni specificate a livello di applicazione e a livello di processo per determinare le configurazioni per l'esecuzione del lavoro.

Il livello di granularità al momento dell'unione è il seguente:

- **`ApplicationConfiguration`**- Tipo di classificazione, ad esempio. `spark-defaults`
- **`MonitoringConfiguration`**- Tipo di configurazione, ad esempio`s3MonitoringConfiguration`.

> **Note**
>
> La priorità delle configurazioni fornite in `StartJobRun`sostituisce le configurazioni fornite a livello di applicazione.

Per ulteriori informazioni sulla classificazione delle priorità, vedere e. Parametro di sovrascrittura della configurazione Hive La configurazione di Spark sovrascrive il parametro

Quando si avvia un lavoro, se non si specifica una configurazione particolare, questa verrà ereditata dall'applicazione. Se si dichiarano le configurazioni a livello di processo, è possibile eseguire le seguenti operazioni:

- Sostituisci una configurazione esistente: fornisci lo stesso parametro di configurazione nella `StartJobRun` richiesta con i tuoi valori di override.

- Aggiungi una configurazione aggiuntiva: aggiungi il nuovo parametro di configurazione nella `StartJobRun` richiesta con i valori che desideri specificare.
- Rimuovere una configurazione esistente: per rimuovere una configurazione di runtime dell'applicazione, fornite la chiave per la configurazione che desiderate rimuovere e passate una dichiarazione vuota `{}` per la configurazione. Non è consigliabile rimuovere le classificazioni che contengono parametri necessari per l'esecuzione di un processo. Ad esempio, se si tenta di rimuovere le proprietà richieste per un lavoro Hive, il processo avrà esito negativo.

  Per rimuovere una configurazione di monitoraggio dell'applicazione, utilizzate il metodo appropriato per il tipo di configurazione pertinente:

  - **`cloudWatchLoggingConfiguration`**- Per rimuoverla`cloudWatchLogging`, passate il flag enabled as`false`.
  - **`managedPersistenceMonitoringConfiguration`**- Per rimuovere le impostazioni di persistenza gestita e tornare allo stato abilitato predefinito, passa una dichiarazione vuota `{}` per la configurazione.
  - **`s3MonitoringConfiguration`**- Per rimuovere`s3MonitoringConfiguration`, passa una dichiarazione vuota `{}` per la configurazione.

Esempio: override

L'esempio seguente mostra diverse operazioni che è possibile eseguire durante l'invio di un lavoro all'indirizzo. `start-job-run`

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

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

Al momento dell'esecuzione del lavoro, verranno applicate le seguenti classificazioni e configurazioni in base alla classificazione di priorità prioritaria descritta in and. Parametro di sovrascrittura della configurazione Hive La configurazione di Spark sovrascrive il parametro

- La classificazione `spark-defaults` verrà aggiornata con le proprietà specificate a livello di mansione. Ai fini di questa classificazione `StartJobRun` verranno prese in considerazione solo le proprietà incluse in.
- La classificazione `spark-executor-log4j2` verrà aggiunta all'elenco di classificazioni esistente.
- La classificazione `spark-driver-log4j2` verrà rimossa.
- Le configurazioni di `managedPersistenceMonitoringConfiguration` verranno aggiornate con le configurazioni a livello di processo.
- Le configurazioni di `s3MonitoringConfiguration` verranno rimosse.
- Le configurazioni di `cloudWatchLoggingConfiguration` verranno aggiunte alle configurazioni di monitoraggio esistenti.

Personalizzazione di un'immagine EMR Serverless

A partire da Amazon EMR 6.9.0, puoi utilizzare immagini personalizzate per impacchettare le dipendenze delle applicazioni e gli ambienti di runtime in un unico contenitore con Amazon Serverless. EMR Ciò semplifica la gestione delle dipendenze del carico di lavoro e rende i pacchetti più portabili. La personalizzazione dell'immagine EMR Serverless offre i seguenti vantaggi:

- Installa e configura pacchetti ottimizzati per i carichi di lavoro. Questi pacchetti potrebbero non essere ampiamente disponibili nella distribuzione pubblica degli ambienti di EMR runtime Amazon.
- Integra EMR Serverless con gli attuali processi di compilazione, test e distribuzione stabiliti all'interno dell'organizzazione, inclusi sviluppo e test locali.
- Applica processi di sicurezza consolidati, come la scansione delle immagini, che soddisfano i requisiti di conformità e governance all'interno dell'organizzazione.
- Consente di utilizzare le proprie versioni di Python JDK e Python per le applicazioni.

EMRServerless fornisce immagini che è possibile utilizzare come base per creare immagini personalizzate. L'immagine di base fornisce i jar, la configurazione e le librerie essenziali per l'interazione dell'immagine con EMR Serverless. Puoi trovare l'immagine di base nella Amazon ECR Public Gallery. Usa l'immagine che corrisponde al tipo di applicazione (Spark o Hive) e alla versione di rilascio. Ad esempio, se crei un'applicazione sulla EMR versione 6.9.0 di Amazon, utilizza le seguenti immagini.

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

Prerequisiti

Prima di creare un'immagine personalizzata EMR Serverless, completa questi prerequisiti.

1. Crea un ECR repository Amazon nello stesso Regione AWS che usi per avviare applicazioni EMR Serverless. Per creare un archivio ECR privato Amazon, consulta Creazione di un repository privato.
2. Per concedere agli utenti l'accesso al tuo ECR repository Amazon, aggiungi le seguenti politiche agli utenti e ai ruoli che creano o aggiornano applicazioni EMR Serverless con immagini da questo repository.

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

Per ulteriori esempi di policy basate sull'ECRidentità di Amazon, consulta Esempi di policy basate sull'identità di Amazon Elastic Container Registry.

Passaggio 1: crea un'immagine personalizzata da immagini di base Serverless EMR

Innanzitutto, crea un Dockerfile che inizi con un'FROMistruzione che utilizza l'immagine di base preferita. Dopo l'FROMistruzione, puoi includere qualsiasi modifica che desideri apportare all'immagine. L'immagine di base imposta automaticamente USER suhadoop. Questa impostazione potrebbe non disporre delle autorizzazioni per tutte le modifiche incluse. Come soluzione alternativa, imposta suroot, modifica USER l'immagine e quindi reimpostala su. USER hadoop:hadoop Per visualizzare esempi di casi d'uso comuni, consultaUtilizzo di immagini personalizzate con EMR Serverless.

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

USER root
# MODIFICATIONS GO HERE

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

Dopo aver creato il Dockerfile, crea l'immagine con il seguente comando.

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

Passaggio 2: convalida l'immagine localmente

EMRServerless fornisce uno strumento offline in grado di controllare staticamente l'immagine personalizzata per convalidare file di base, variabili di ambiente e correggere le configurazioni dell'immagine. Per informazioni su come installare ed eseguire lo strumento, consulta Amazon EMR Serverless Image CLI GitHub.

Dopo aver installato lo strumento, esegui il seguente comando per convalidare un'immagine:

```
amazon-emr-serverless-image \
validate-image -r emr-6.9.0 -t spark \
-i aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

Dovresti vedere un output simile al seguente.

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

Passaggio 3: carica l'immagine nel tuo ECR repository Amazon

Invia la tua ECR immagine Amazon al tuo ECR repository Amazon con i seguenti comandi. Assicurati di disporre delle IAM autorizzazioni corrette per inviare l'immagine al tuo repository. Per ulteriori informazioni, consulta Pushing an image nella Amazon ECR User Guide.

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

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

Passaggio 4: creare o aggiornare un'applicazione con immagini personalizzate

Seleziona AWS Management Console scheda o AWS CLI seleziona in base alla modalità in cui desideri avviare l'applicazione, quindi completa i seguenti passaggi.

Console

1. Accedi alla console di EMR Studio all'indirizzo https://console.aws.amazon.com/emr. Accedi alla tua applicazione o crea una nuova applicazione seguendo le istruzioni riportate in Creare un'applicazione.
2. Per specificare immagini personalizzate quando crei o aggiorni un'applicazione EMR Serverless, seleziona Impostazioni personalizzate nelle opzioni di configurazione dell'applicazione.
3. Nella sezione Impostazioni dell'immagine personalizzata, seleziona la casella di controllo Usa l'immagine personalizzata con questa applicazione.
4. Incolla l'ECRimmagine Amazon URI nel URI campo Immagine. EMRServerless utilizza questa immagine per tutti i tipi di lavoratori dell'applicazione. In alternativa, puoi scegliere Immagini personalizzate diverse e incollare ECR immagini Amazon diverse URIs per ogni tipo di lavoratore.

CLI

- Crea un'applicazione con il `image-configuration` parametro. EMRServerless applica questa impostazione a tutti i tipi di lavoratore.

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

  Per creare un'applicazione con impostazioni di immagine diverse per ogni tipo di lavoratore, utilizzate il `worker-type-specifications` parametro.

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

Per aggiornare un'applicazione, utilizzate il `image-configuration` parametro. EMRServerless applica questa impostazione a tutti i tipi di lavoratore.

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

Passaggio 5: consentire a EMR Serverless di accedere all'archivio di immagini personalizzato

Aggiungi la seguente politica sulle risorse al ECR repository Amazon per consentire al responsabile del servizio EMR Serverless di utilizzare `getdescribe`, e `download` le richieste da questo repository.

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

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

Come best practice di sicurezza, aggiungi una chiave di `aws:SourceArn` condizione alla policy del repository. La chiave di condizione IAM globale `aws:SourceArn` garantisce che EMR Serverless utilizzi il repository solo per un'applicazione. ARN Per ulteriori informazioni sulle politiche relative ai ECR repository di Amazon, consulta Creazione di un repository privato.

Considerazioni e limitazioni

Quando lavori con immagini personalizzate, considera quanto segue:

- Usa l'immagine di base corretta che corrisponda al tipo (Spark o Hive) e all'etichetta di rilascio (ad esempio`emr-6.9.0`) dell'applicazione.
- EMRServerless ignora le `[ENTRYPOINT]` istruzioni `[CMD]` o le istruzioni contenute nel file Docker. Usa istruzioni comuni nel file Docker, ad esempio, e`[COPY]`. `[RUN]` `[WORKDIR]`
- Non è necessario modificare le variabili di ambiente `JAVA_HOME`SPARK_HOME,`HIVE_HOME`, `TEZ_HOME` quando si crea un'immagine personalizzata.
- Le immagini personalizzate non possono superare le dimensioni di 5 GB.
- Se modifichi i file binari o i jar nelle immagini di EMR base di Amazon, ciò potrebbe causare errori nell'avvio dell'applicazione o del processo.
- Il ECR repository Amazon dovrebbe trovarsi nello stesso Regione AWS che usi per avviare applicazioni EMR Serverless.

Configurazione dell'accesso VPC

Puoi configurare applicazioni EMR Serverless per connetterti ai tuoi archivi di dati all'interno del tuoVPC, come cluster Amazon Redshift, database Amazon o RDS bucket Amazon S3 con endpoint. VPC La tua applicazione EMR Serverless dispone di connettività in uscita agli archivi di dati all'interno del tuo. VPC Per impostazione predefinita, EMR Serverless blocca l'accesso in entrata alle applicazioni per migliorare la sicurezza.

> **Note**
>
> È necessario configurare VPC l'accesso se si desidera utilizzare un database metastore Hive esterno per l'applicazione. Per informazioni su come configurare un metastore Hive esterno, consulta Configurazione di Metastore.

Crea applicazione

Nella pagina Crea applicazione, puoi scegliere impostazioni personalizzate e specificare le sottoreti e i gruppi di sicurezza che VPC le applicazioni Serverless possono utilizzare. EMR

VPCs

Scegli il nome del cloud privato virtuale (VPC) che contiene i tuoi archivi di dati. La pagina Crea applicazione elenca tutti VPCs i dati prescelti Regione AWS.

Sottoreti

Scegli le sottoreti all'interno dell'archivio dati VPC che contiene il tuo archivio dati. La pagina Crea applicazione elenca tutte le sottoreti per gli archivi dati presenti nel tuo. VPC

Le sottoreti selezionate devono essere sottoreti private. Ciò significa che le tabelle di routing associate per le sottoreti non devono disporre di gateway Internet.

Per la connettività in uscita a Internet, le sottoreti devono disporre di percorsi in uscita che utilizzano un gateway. NAT Per configurare un NAT gateway, consulta Lavorare con i gateway. NAT

Per la connettività Amazon S3, le sottoreti devono avere un NAT gateway o un endpoint configurato. VPC Per configurare un endpoint S3, consulta Creare un VPC endpoint gateway.

Per la connettività ad altri Servizi AWS al di fuori diVPC, come Amazon DynamoDB, è necessario configurare gli endpoint VPC o un gateway. NAT Per configurare gli endpoint per VPC Servizi AWS, vedi Lavorare con gli VPC endpoint.

I lavoratori possono connettersi agli archivi dati all'interno dell'azienda VPC tramite il traffico in uscita. Per impostazione predefinita, EMR Serverless blocca l'accesso in entrata ai lavoratori per migliorare la sicurezza.

Quando si utilizza AWS Config, EMR Serverless crea un record di elementi dell'interfaccia di rete elastica per ogni lavoratore. Per evitare i costi legati a questa risorsa, prendi in considerazione la possibilità di disattivarla `AWS::EC2::NetworkInterface` AWS Config.

> **Note**
>
> Ti consigliamo di selezionare più sottoreti in più zone di disponibilità. Questo perché le sottoreti scelte determinano le zone di disponibilità disponibili per l'avvio di un'applicazione EMR serverless. Ogni lavoratore utilizzerà un indirizzo IP nella sottorete in cui viene avviato. Assicurati che le sottoreti specificate abbiano indirizzi IP sufficienti per il numero di worker che intendi avviare. Per ulteriori informazioni sulla pianificazione delle sottoreti, vedere. the section called “Procedure consigliate per la pianificazione delle sottoreti”

Gruppi di sicurezza

Scegli uno o più gruppi di sicurezza in grado di comunicare con i tuoi archivi di dati. La pagina Crea applicazione elenca tutti i gruppi di sicurezza presenti nel tuoVPC. EMRServerless associa questi gruppi di sicurezza a interfacce di rete elastiche collegate alle sottoreti. VPC

> **Note**
>
> Si consiglia di creare un gruppo di sicurezza separato per le applicazioni Serverless. EMR Ciò rende più efficienti l'isolamento e la gestione delle regole di rete. Ad esempio, per comunicare con i cluster Amazon Redshift, puoi definire le regole del traffico tra i gruppi di sicurezza Redshift e EMR Serverless, come illustrato nell'esempio seguente.

Example Esempio: comunicazione con i cluster Amazon Redshift

1. Aggiungi una regola per il traffico in entrata al gruppo di sicurezza Amazon Redshift da uno dei gruppi di EMR sicurezza Serverless.

Tipo	Protocollo	Intervallo porte	Origine
Tutti TCP	TCP	5439	`emr-serverless-security-group`

2. Aggiungi una regola per il traffico in uscita da uno dei gruppi di sicurezza EMR Serverless. Ci sono due modi per farlo. Innanzitutto, puoi aprire il traffico in uscita verso tutte le porte.

Tipo	Protocollo	Intervallo porte	Destinazione
Tutto il traffico	TCP	ALL	0.0.0.0/0

In alternativa, puoi limitare il traffico in uscita ai cluster Amazon Redshift. Ciò è utile solo quando l'applicazione deve comunicare con i cluster Amazon Redshift e nient'altro.

Tipo	Protocollo	Intervallo porte	Origine
Tutto TCP	TCP	5439	`redshift-security-group`

Configura l'applicazione

È possibile modificare la configurazione di rete per un'applicazione EMR Serverless esistente dalla pagina Configura applicazione.

Visualizza i dettagli dell'esecuzione del processo

Nella pagina dei dettagli del Job run, è possibile visualizzare la sottorete utilizzata dal job per un'esecuzione specifica. Si noti che un processo viene eseguito solo in una sottorete selezionata dalle sottoreti specificate.

Procedure consigliate per la pianificazione delle sottoreti

AWS le risorse vengono create in una sottorete che è un sottoinsieme di indirizzi IP disponibili in Amazon. VPC Ad esempio, una maschera di rete VPC con una maschera di rete /16 ha fino a 65.536 indirizzi IP disponibili che possono essere suddivisi in più reti più piccole utilizzando maschere di sottorete. Ad esempio, è possibile suddividere questo intervallo in due sottoreti, ognuna delle quali utilizza la maschera /17 e 32.768 indirizzi IP disponibili. Una sottorete si trova all'interno di una zona di disponibilità e non può estendersi su più zone.

Le sottoreti devono essere progettate tenendo conto dei limiti di scalabilità delle applicazioni Serverless. EMR Ad esempio, se un'applicazione richiede 4 vCpu lavoratori ed è possibile scalare fino a 4.000vCpu, l'applicazione richiederà al massimo 1.000 lavoratori per un totale di 1.000 interfacce di rete. Ti consigliamo di creare sottoreti su più zone di disponibilità. Ciò consente a EMR Serverless di riprovare il lavoro o di fornire capacità preinizializzata in un'altra zona di disponibilità nell'improbabile eventualità di un guasto in una zona di disponibilità. Pertanto, ogni sottorete in almeno due zone di disponibilità deve avere più di 1.000 indirizzi IP disponibili.

Sono necessarie sottoreti con una dimensione della maschera inferiore o uguale a 22 per effettuare il provisioning di 1.000 interfacce di rete. Qualsiasi maschera superiore a 22 non soddisferà il requisito. Ad esempio, una subnet mask di /23 fornisce 512 indirizzi IP, mentre una maschera di /22 fornisce 1024 e una maschera di /21 fornisce 2048 indirizzi IP. Di seguito è riportato un esempio di 4 sottoreti con una maschera di rete /22 in una maschera di rete /16 che possono essere allocate a VPC diverse zone di disponibilità. Esiste una differenza di cinque tra gli indirizzi IP disponibili e quelli utilizzabili perché i primi quattro indirizzi IP e l'ultimo indirizzo IP in ogni sottorete sono riservati da AWS.

ID sottorete	Indirizzo di sottorete	Maschera di sottorete	Intervallo di indirizzi IP	Indirizzi IP disponibili	Indirizzi IP utilizzabili
1	10.0.0.0	255,255,252,0/22	10.0.0.0 - 10.0.3.255	1,024	1.019

ID sottorete	Indirizzo di sottorete	Maschera di sottorete	Intervallo di indirizzi IP	Indirizzi IP disponibili	Indirizzi IP utilizzabili
2	10,04,0	255,255,252,0/22	10.0.4.0 - 10.0.7.255	1,024	1.019
3	10,08.0	255,255,252,0/22	10.0.4.0 - 10.0.7.255	1,024	1.019
4	10,012,0	255,255,252,0/22	10.0.12.0 - 10.0.15.255	1,024	1.019

Dovresti valutare se il tuo carico di lavoro è più adatto per lavoratori di grandi dimensioni. L'utilizzo di lavoratori di dimensioni maggiori richiede un minor numero di interfacce di rete. Ad esempio, l'utilizzo di 16 vCpu worker con un limite di scalabilità delle applicazioni di 4.000 vCpu richiederà al massimo 250 lavoratori per un totale di 250 indirizzi IP disponibili per fornire le interfacce di rete. Per effettuare il provisioning di 250 interfacce di rete sono necessarie sottoreti in più zone di disponibilità con dimensioni della maschera inferiori o uguali a 24. Qualsiasi maschera di dimensioni superiori a 24 offre meno di 250 indirizzi IP.

Se condividi sottoreti tra più applicazioni, ogni sottorete deve essere progettata tenendo conto dei limiti di scalabilità collettivi di tutte le applicazioni. Ad esempio, se avete 3 applicazioni che richiedono 4 vCpu lavoratori e ciascuna può essere scalata fino a 4000 vCpu con una quota di servizio a vCpu livello di account di 12.000, ogni sottorete richiederà 3000 indirizzi IP disponibili. Se VPC quello che desideri utilizzare non ha un numero sufficiente di indirizzi IP, prova ad aumentare il numero di indirizzi IP disponibili. Puoi farlo associando blocchi Classless Inter-Domain Routing () CIDR aggiuntivi al tuo. VPC Per ulteriori informazioni, consulta Associare IPv4 CIDR blocchi aggiuntivi ai tuoi VPC nella Amazon VPC User Guide.

Puoi utilizzare uno dei tanti strumenti disponibili online per generare rapidamente definizioni di sottorete e rivedere la gamma di indirizzi IP disponibili.

Opzioni di architettura Amazon EMR Serverless

L'architettura del set di istruzioni della tua applicazione Amazon EMR Serverless determina il tipo di processori utilizzati dall'applicazione per eseguire il processo. Amazon EMR offre due opzioni di architettura per la tua applicazione: x86_64 e arm64. EMRServerless si aggiorna automaticamente

alle istanze di ultima generazione non appena sono disponibili, in modo che le applicazioni possano utilizzare le istanze più recenti senza richiedere ulteriori sforzi da parte tua.

Argomenti

Utilizzo dell'architettura x86_64

L'architettura x86_64 è anche nota come x86 64-bit o x64. x86_64 è l'opzione predefinita per le applicazioni serverless. EMR Questa architettura utilizza processori basati su x86 ed è compatibile con la maggior parte degli strumenti e delle librerie di terze parti.

La maggior parte delle applicazioni è compatibile con la piattaforma hardware x86 e può essere eseguita correttamente sull'architettura x86_64 predefinita. Tuttavia, se l'applicazione è compatibile con la tecnologia a 64 bitARM, è possibile passare ad arm64 per utilizzare i processori Graviton per migliorare prestazioni, potenza di calcolo e memoria. L'esecuzione di istanze sull'architettura arm64 costa meno rispetto a quando si eseguono istanze di uguali dimensioni sull'architettura x86.

Utilizzo dell'architettura arm64 (Graviton)

AWS I processori Graviton sono progettati su misura da AWS con core ARM Neoverse a 64 bit e sfrutta l'architettura arm64 (nota anche come Arch64 o 64-bit). ARM Il AWS La linea di processori Graviton disponibile su Serverless include i processori Graviton3 e EMR Graviton2. Questi processori offrono un rapporto prezzo/prestazioni superiore per i carichi di lavoro Spark e Hive rispetto ai carichi di lavoro equivalenti eseguiti sull'architettura x86_64. EMRServerless utilizza automaticamente i processori di ultima generazione, se disponibili, senza alcuno sforzo da parte dell'utente per eseguire l'aggiornamento ai processori di ultima generazione.

Lancio di nuove applicazioni con supporto Graviton

Utilizzate uno dei seguenti metodi per avviare un'applicazione che utilizza l'architettura arm64.

AWS CLI

Per avviare un'applicazione utilizzando processori Graviton da AWS CLI, specificare `ARM64` come `architecture` parametro in. `create-application` API Specificate i valori appropriati per l'applicazione negli altri parametri.

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

EMR Studio

Per avviare un'applicazione utilizzando i processori Graviton di EMR Studio, scegli arm64 come opzione Architettura quando crei o aggiorni un'applicazione.

Configurazione delle applicazioni esistenti per l'utilizzo di Graviton

Puoi configurare le tue applicazioni Amazon EMR Serverless esistenti per utilizzare l'architettura Graviton (arm64) con, SDK AWS CLI o Studio. EMR

Per convertire un'applicazione esistente da x86 a arm64

1. Confermate che state utilizzando la versione principale più recente di AWS CLI/SDKche supporta il `architecture` parametro.
2. Verificate che non vi siano lavori in esecuzione, quindi arrestate l'applicazione.

   ```
   aws emr-serverless stop-application \
    --application-id application-id \
    --region us-west-2
   ```
3. Per aggiornare l'applicazione per utilizzare Graviton, specificate `ARM64` il `architecture` parametro in. `update-application` API

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

4. Per verificare che l'CPUarchitettura dell'applicazione sia aggiornataARM64, utilizzate il `get-application`API.

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

5. Quando sei pronto, riavvia l'applicazione.

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

Considerazioni sull'utilizzo di Graviton

Prima di avviare un'applicazione EMR Serverless utilizzando arm64 per il supporto Graviton, conferma quanto segue.

Compatibilità con le librerie

Quando selezioni Graviton (arm64) come opzione di architettura, assicurati che i pacchetti e le librerie di terze parti siano compatibili con l'architettura a 64 bit. ARM Per informazioni su come impacchettare le librerie Python in un ambiente virtuale Python compatibile con l'architettura selezionata, vedere. Utilizzo delle librerie Python con Serverless EMR

Per ulteriori informazioni su come configurare un carico di lavoro Spark o Hive per l'utilizzo a 64 bit, consulta il ARM AWS Repository Graviton Getting Started attivo. GitHub Questo repository contiene risorse essenziali che possono aiutarti a iniziare con Graviton basato su base. ARM

Trasferisci i dati in S3 Express One Zone con Serverless EMR

Con le EMR versioni 7.2.0 e successive di Amazon, puoi usare EMR Serverless 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

- Autorizzazioni S3 Express One Zone: quando S3 Express One Zone esegue inizialmente un'azione come o su un oggetto S3`GET`, `LIST` la classe di storage chiama per tuo conto. PUT `CreateSession` La tua IAM politica deve consentire l'autorizzazione in modo che il `s3express:CreateSession` S3A il connettore può richiamare il. `CreateSession` API Per un esempio di policy con questa autorizzazione, consulta Nozioni di base su S3 Express One Zone.
- S3A connettore: per configurare Spark per accedere ai dati da un bucket Amazon S3 che utilizza la classe di storage S3 Express One Zone, devi utilizzare il connettore Apache Hadoop S3A. Per utilizzare il connettore, assicurati che tutti gli S3 URIs utilizzino lo `s3a` schema. In caso contrario, puoi modificare l'implementazione del file system che utilizzi per gli schemi `s3` e `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"
    }
  }
]
```

Nozioni di base su S3 Express One Zone

Segui questi passaggi per iniziare a usare S3 Express One Zone.

1. Crea un VPC endpoint. Aggiungi l'endpoint `com.amazonaws.us-west-2.s3express` all'endpoint. VPC
2. Segui Guida introduttiva ad Amazon EMR Serverless per creare un'applicazione con Amazon EMR Release Label 7.2.0 o versione successiva.
3. Configura la tua applicazione per utilizzare l'VPCendpoint appena creato, un gruppo di sottoreti privato e un gruppo di sicurezza.
4. Aggiungi l'`CreateSession`autorizzazione al tuo ruolo di esecuzione del lavoro.

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

5. Esegui il tuo lavoro. Tieni presente che devi utilizzare lo `S3A` schema per accedere ai bucket S3 Express One Zone.

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

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


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

Esecuzione di processi

Dopo aver fornito la candidatura, è possibile inviare offerte di lavoro all'applicazione. In questa sezione viene illustrato come utilizzare AWS CLI per eseguire questi lavori. Questa sezione identifica anche i valori predefiniti per ogni tipo di applicazione disponibile su EMR Serverless.

Argomenti

Stati delle esecuzioni di processi

Quando invii un'esecuzione di lavoro a una coda di processi Amazon EMR Serverless, l'esecuzione del processo entra nello `SUBMITTED` stato. Lo stato di un processo `SUBMITTED` passa da «a» `RUNNING` fino a raggiungere `FAILED``SUCCESS`, o. `CANCELLING`

Le esecuzioni di processi possono avere i seguenti stati:

Stato	Descrizione
Inviato	Lo stato iniziale del processo quando si invia un processo eseguito a EMR Serverless. Il lavoro attende di essere pianificato per l'applicazione. EMRServerless inizia a stabilire le priorità e a pianificare l'esecuzione del lavoro.

Stato	Descrizione
Pending (In attesa)	Lo scheduler sta valutando l'esecuzione del job per stabilire le priorità e pianificare l'esecuzione dell'applicazione.
Pianificato	EMRServerless ha pianificato l'esecuzione del lavoro per l'applicazione e sta allocando risorse per eseguire il lavoro.
In esecuzione	EMRServerless ha allocato le risorse inizialmente necessarie al processo e il processo è in esecuzione nell'applicazione. Nelle applicazioni Spark, ciò significa che il processo del driver Spark si trova nello stato `running`.
Failed (Non riuscito)	EMRServerless non è riuscito a inviare il job run all'applicazione oppure il job è stato completato senza successo. `StateDetails` Per ulteriori informazioni su questo errore del processo, vedere.
Riuscito	L'esecuzione del processo è stata completata correttamente.
Annullamento in corso	`CancelJobRun` APIHa richiesto l'annullamento dell'esecuzione del lavoro oppure l'esecuzione del lavoro è scaduta. EMRServerless sta tentando di annullare il processo nell'applicazione e di rilasciare le risorse.
Annullato	L'esecuzione del processo è stata annullata correttamente e le risorse utilizzate sono state rilasciate.

Esecuzione di processi dalla console EMR Studio

È possibile inviare le esecuzioni dei job alle applicazioni EMR Serverless e visualizzarle dalla console EMR Studio. Per creare o accedere all'applicazione EMR Serverless sulla console EMR Studio, segui le istruzioni in Guida introduttiva alla console.

Invio di un processo

Nella pagina Invia un lavoro, puoi inviare un lavoro a un'applicazione EMR Serverless come segue.

Spark

1. Nel campo Nome, inserisci un nome per l'esecuzione del job.
2. Nel campo Runtime role, inserisci il nome del IAM ruolo che l'applicazione EMR Serverless può assumere per l'esecuzione del job. Per ulteriori informazioni sui ruoli di runtime, consultaRuoli Job Runtime per Amazon EMR Serverless.
3. Nel campo Posizione dello script, inserisci la posizione Amazon S3 dello script o JAR che desideri eseguire. Per i lavori Spark, lo script può essere un file Python `.py` () o JAR un file `.jar` ().
4. Se la posizione dello script è un JAR file, inserisci il nome della classe che è il punto di ingresso per il lavoro nel campo Classe principale.
5. (Facoltativo) Inserite i valori per i campi rimanenti.

   - Argomenti dello script: inserisci gli argomenti che desideri passare allo script principale JAR o in Python. Il codice legge questi parametri. Separare ogni argomento dell'array con una virgola.
   - Proprietà Spark: espandi la sezione delle proprietà Spark e inserisci qualsiasi parametro di configurazione Spark in questo campo.

     > **Note**
     >
     > Se specifichi le dimensioni del driver e dell'executor Spark, devi tenere conto del sovraccarico di memoria. Specificate i valori del sovraccarico di memoria nelle proprietà e. `spark.driver.memoryOverhead` `spark.executor.memoryOverhead` Il sovraccarico di memoria ha un valore predefinito del 10% della memoria del contenitore, con un minimo di 384 MB. La memoria dell'esecutore e il sovraccarico di memoria insieme non possono superare

la memoria di lavoro. Ad esempio, il massimo `spark.executor.memory` per un worker da 30 GB deve essere di 27 GB.

- Job configuration: specifica qualsiasi configurazione del lavoro in questo campo. È possibile utilizzare queste configurazioni di lavoro per sovrascrivere le configurazioni predefinite per le applicazioni.
- Impostazioni aggiuntive: attiva o disattiva il AWS Glue Data Catalog come metastore e modifica le impostazioni del registro dell'applicazione. Per ulteriori informazioni sulle configurazioni dei metastore, consulta. Configurazione Metastore Per ulteriori informazioni sulle opzioni di registrazione delle applicazioni, consulta. Archiviazione dei log
- Tag: assegna tag personalizzati all'applicazione.

6. Seleziona Submit job (Invia processo).

Hive

1. Nel campo Nome, inserisci un nome per l'esecuzione del job.
2. Nel campo Runtime role, inserisci il nome del IAM ruolo che l'applicazione EMR Serverless può assumere per l'esecuzione del job.
3. Nel campo Posizione dello script, inserisci la posizione Amazon S3 dello script o JAR che desideri eseguire. Per i lavori Hive, lo script deve essere un file Hive (`.sql`.
4. (Facoltativo) Immettete i valori per i campi rimanenti.

   - Posizione dello script di inizializzazione: immettere la posizione dello script che inizializza le tabelle prima dell'esecuzione dello script Hive.
   - Proprietà Hive: espandi la sezione delle proprietà di Hive e inserisci qualsiasi parametro di configurazione Hive in questo campo.
   - Configurazione del lavoro: specifica qualsiasi configurazione del lavoro. È possibile utilizzare queste configurazioni di lavoro per sovrascrivere le configurazioni predefinite per le applicazioni. Per i lavori Hive, `hive.exec.scratchdir` e `hive.metastore.warehouse.dir` sono proprietà obbligatorie nella configurazione. `hive-site`

```
{
    "applicationConfiguration": [
        {
            "classification": "hive-site",
```

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

- Impostazioni aggiuntive: attiva o disattiva il AWS Glue Data Catalog come metastore e modifica le impostazioni del registro dell'applicazione. Per ulteriori informazioni sulle configurazioni dei metastore, consulta. Configurazione Metastore Per ulteriori informazioni sulle opzioni di registrazione delle applicazioni, consulta. Archiviazione dei log
- Tag: assegna qualsiasi tag personalizzato all'applicazione.

5. Seleziona Submit job (Invia processo).

Visualizza esecuzioni dei processi

Dalla scheda Job run nella pagina Dettagli di un'applicazione, è possibile visualizzare le esecuzioni dei job ed eseguire le seguenti azioni per le esecuzioni di job.

Annulla processo: per annullare l'esecuzione di un processo in questo `RUNNING` stato, scegliete questa opzione. Per ulteriori informazioni sulle transizioni Job Run, consultaStati delle esecuzioni di processi.

Clona processo: per clonare un processo precedente eseguito e inviarlo nuovamente, scegli questa opzione.

Esecuzione di lavori da AWS CLI

È possibile creare, descrivere ed eliminare singoli lavori su AWS CLI. Puoi anche elencare tutti i tuoi lavori per visualizzarli a colpo d'occhio.

Per inviare un nuovo lavoro, usa`start-job-run`. Fornisci l'ID dell'applicazione che desideri eseguire, insieme alle proprietà specifiche del lavoro. Per gli esempi di Spark, consulta. Offerte di lavoro Spark Per gli esempi di Hive, vedi. Offerte di lavoro Hive Questo comando restituisce il tuo `application-id`ARN, e nuovo`job-id`.

Ogni esecuzione di un processo ha una durata di timeout impostata. Se l'esecuzione del processo supera tale durata, EMR Serverless la annullerà automaticamente. Il timeout predefinito è di 12 ore. Quando si avvia l'esecuzione del job, è possibile configurare questa impostazione di timeout su un valore che soddisfi i requisiti del job. Configura il valore con la `executionTimeoutMinutes` proprietà.

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

Per descrivere un lavoro, usa`get-job-run`. Questo comando restituisce le configurazioni specifiche del job e la capacità impostata per il nuovo job.

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

Per elencare i tuoi lavori, usa. `list-job-runs` Questo comando restituisce un insieme abbreviato di proprietà che include il tipo di lavoro, lo stato e altri attributi di alto livello. Se non desideri visualizzare tutti i tuoi lavori, puoi specificare il numero massimo di lavori che desideri visualizzare, fino a 50. L'esempio seguente specifica che desideri vedere i tuoi ultimi due job eseguiti.

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

```
--application-id application-id
```

Per annullare un lavoro, usa`cancel-job-run`. Fornisci il nome `application-id` e il nome `job-id` del lavoro che desideri annullare.

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

Per ulteriori informazioni su come eseguire i lavori da AWS CLI, consulta il EMRServerless API Reference.

Utilizzo di dischi ottimizzati per la riproduzione casuale

Con le EMR versioni 7.1.0 e successive di Amazon, puoi usare dischi ottimizzati per lo shuffle quando esegui i job Apache Spark o Hive per migliorare le prestazioni per carichi di lavoro a uso intensivo di I/O. Rispetto ai dischi standard, i dischi ottimizzati per lo shuffle offrono prestazioni superiori IOPS (operazioni di I/O al secondo) per uno spostamento più rapido dei dati e una latenza ridotta durante le operazioni di shuffle. I dischi ottimizzati per lo shuffle consentono di collegare dischi di dimensioni fino a 2 TB per lavoratore, in modo da poter configurare la capacità appropriata per i requisiti del carico di lavoro.

Vantaggi principali

I dischi ottimizzati per lo shuffle offrono i seguenti vantaggi.

- IOPSPrestazioni elevate: i dischi ottimizzati per lo shuffle offrono prestazioni superiori IOPS rispetto ai dischi standard, il che consente uno shuffling dei dati più efficiente e rapido durante i job Spark e Hive e altri carichi di lavoro che richiedono un uso intensivo dello shuffle.
- Dimensioni del disco più grandi: i dischi ottimizzati per Shuffle supportano dimensioni del disco da 20 GB a 2 TB per lavoratore, in modo da poter scegliere la capacità appropriata in base ai carichi di lavoro.

Nozioni di base

Consulta i passaggi seguenti per utilizzare dischi ottimizzati per lo shuffle nei flussi di lavoro.

Spark

1. Crea un'applicazione EMR Serverless release 7.1.0 con il seguente comando.

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

2. Configura il tuo job Spark per includere i parametri `spark.emr-serverless.driver.disk.type` e/o per l'esecuzione con dischi ottimizzati `spark.emr-serverless.executor.disk.type` per lo shuffle. Puoi utilizzare uno o entrambi i parametri, a seconda del caso d'uso.

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

Per maggiori informazioni, consulta Spark job properties.

Hive

1. Crea un'applicazione EMR Serverless release 7.1.0 con il seguente comando.

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

```
  --release-label emr-7.1.0 \
  --region <AWS_REGION>
```

2. Configura il tuo job Hive per includere i parametri `hive.driver.disk.type` e/o per l'esecuzione con dischi ottimizzati `hive.tez.disk.type` per lo shuffle. È possibile utilizzare uno o entrambi i parametri, a seconda del caso d'uso.

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

Per ulteriori informazioni, consulta Hive job properties.

Configurazione di un'applicazione con capacità preinizializzata

Guarda i seguenti esempi per creare applicazioni basate sulla EMR release 7.1.0 di Amazon. Queste applicazioni hanno le seguenti proprietà:

- 5 driver Spark preinizializzati, ciascuno con 2 vCPU, 4 GB di memoria e 50 GB di disco ottimizzato per lo shuffle.
- 50 executor preinizializzati, ciascuno con 4 vCPU, 8 GB di memoria e 500 GB di disco ottimizzato per lo shuffle.

Quando questa applicazione esegue i job Spark, utilizza innanzitutto i worker preinizializzati e quindi ridimensiona i worker su richiesta fino alla capacità massima di 400 v e 1024 GB di memoria. CPU Facoltativamente, puoi omettere la capacità per uno o. `DRIVER EXECUTOR`

Spark

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

Hive

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

Offerte di lavoro in streaming

Un processo di streaming in EMR Serverless è una modalità di lavoro che consente di analizzare ed elaborare i dati di streaming quasi in tempo reale. Questi processi di lunga durata analizzano i dati in streaming ed elaborano continuamente i risultati non appena arrivano i dati. I lavori in streaming sono più adatti per attività che richiedono l'elaborazione dei dati in tempo reale, come analisi quasi in tempo reale, rilevamento delle frodi e motori di suggerimenti. EMRI processi di streaming senza server offrono ottimizzazioni, come la resilienza integrata dei processi, il monitoraggio in tempo reale, la gestione avanzata dei log e l'integrazione con i connettori di streaming.

Di seguito sono riportati alcuni casi d'uso con i lavori di streaming:

- Analisi quasi in tempo reale: i lavori di streaming in Amazon EMR Serverless ti consentono di elaborare i dati in streaming quasi in tempo reale, in modo da poter eseguire analisi in tempo reale su flussi di dati continui, come dati di log, dati di sensori o dati clickstream per ricavare informazioni e prendere decisioni tempestive sulla base delle informazioni più recenti.
- Rilevamento delle frodi: puoi utilizzare i processi di streaming per eseguire il rilevamento delle frodi quasi in tempo reale nelle transazioni finanziarie, nelle operazioni con carte di credito o nelle attività online quando analizzi i flussi di dati e identifichi modelli o anomalie sospetti man mano che si verificano.
- Motori di raccomandazione: i processi di streaming possono elaborare i dati sulle attività degli utenti e aggiornare i modelli di suggerimenti. In questo modo si aprono possibilità di consigli personalizzati e in tempo reale basati su comportamenti e preferenze.
- Analisi dei social media: le offerte di lavoro in streaming possono elaborare i dati dei social media, come tweet, commenti e post, in modo che le organizzazioni possano monitorare le tendenze, l'analisi del sentiment e gestire la reputazione del marchio quasi in tempo reale.
- Analisi dell'Internet of Things (IoT): i job in streaming possono gestire e analizzare flussi di dati ad alta velocità provenienti da dispositivi IoT, sensori e macchinari connessi, in modo da poter eseguire il rilevamento delle anomalie, la manutenzione predittiva e altri casi d'uso di analisi IoT.
- Analisi clickstream: i job di streaming possono elaborare e analizzare i dati clickstream provenienti da siti Web o applicazioni mobili. Le aziende che utilizzano tali dati possono eseguire analisi per saperne di più sul comportamento degli utenti, personalizzare le esperienze degli utenti e ottimizzare le campagne di marketing.
- Monitoraggio e analisi dei log: i processi di streaming possono anche elaborare i dati di registro da server, applicazioni e dispositivi di rete. Ciò consente il rilevamento delle anomalie, la risoluzione dei problemi e lo stato e le prestazioni del sistema.

Principali vantaggi

I lavori di streaming in EMR Serverless forniscono automaticamente la resilienza del lavoro, che è una combinazione dei seguenti fattori:

- Riprova automatica: EMR Serverless riprova automaticamente tutti i lavori non riusciti senza alcun input manuale da parte dell'utente.
- Resilienza della zona di disponibilità (AZ): EMR Serverless sposta automaticamente i job di streaming su una zona AZ integra se la zona di disponibilità originale presenta problemi.

- Gestione dei registri:
  - Rotazione dei log: per una gestione più efficiente dello storage su disco, EMR Serverless ruota periodicamente i log per lunghi lavori di streaming. In questo modo si evita l'accumulo di log che potrebbe consumare tutto lo spazio su disco.
  - Compattazione dei log: consente di gestire e ottimizzare in modo efficiente i file di registro in modalità di persistenza gestita. La compattazione migliora anche l'esperienza di debug quando si utilizza il server di cronologia Spark gestito.

Fonti di dati e data sink supportati

EMRServerless funziona con una serie di sorgenti di dati di input e data sink di output:

- Fonti di dati di input supportate: Amazon Kinesis Data Streams, Amazon Managed Streaming for Apache Kafka e cluster Apache Kafka autogestiti. Per impostazione predefinita, le EMR versioni di Amazon 7.1.0 e successive includono il connettore Amazon Kinesis Data Streams, quindi non è necessario creare o scaricare pacchetti aggiuntivi.
- Dissipatori di dati di output supportati: AWS Tabelle Glue Data Catalog, Amazon S3, Amazon Redshift, SQL My, SQL Postgre Oracle, Oracle, SQL Microsoft, Apache Iceberg, Delta Lake e Apache Hudi.

Considerazioni e limitazioni

Quando utilizzi i lavori di streaming, tieni presente le seguenti considerazioni e limitazioni.

- I lavori di streaming sono supportati dalle EMRversioni di Amazon 7.1.0 e successive.
- EMRServerless prevede che i processi di streaming durino a lungo, quindi non è possibile impostare il timeout di esecuzione per limitare il tempo di esecuzione del processo.
- I job di streaming sono compatibili solo con il motore Spark, che si basa sul framework di streaming strutturato.
- EMRServerless riprova i processi di streaming a tempo indeterminato e non puoi personalizzare il numero massimo di tentativi. La prevenzione degli errori viene inclusa automaticamente per interrompere il nuovo tentativo di lavoro se il numero di tentativi falliti supera una soglia impostata su una finestra oraria. La soglia predefinita è di cinque tentativi falliti nell'arco di un'ora. È possibile configurare questa soglia in modo che sia compresa tra 1 e 10 tentativi. Per ulteriori informazioni, consulta Job resiliency.

- I lavori di streaming dispongono di punti di controllo per salvare lo stato di runtime e l'avanzamento, quindi EMR Serverless può riprendere il processo di streaming dal checkpoint più recente. Per ulteriori informazioni, consulta Recovery from failures with Checkpointing nella documentazione di Apache Spark.

Guida introduttiva allo streaming di lavori

Consulta le seguenti istruzioni per scoprire come iniziare a utilizzare i lavori di streaming.

1. Segui Guida introduttiva ad Amazon EMR Serverless per creare un'applicazione. Tieni presente che la tua applicazione deve eseguire la EMRversione Amazon 7.1.0 o successiva.
2. Una volta che l'applicazione è pronta, imposta il `mode` parametro `STREAMING` per inviare un lavoro in streaming, simile al seguente AWS CLI esempio.

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

Connettori di streaming supportati

I connettori di streaming facilitano la lettura dei dati da una fonte di streaming e possono anche scrivere dati su un sink di streaming.

I seguenti sono i connettori di streaming supportati:

Connettore Amazon Kinesis Data Streams

Il connettore Amazon Kinesis Data Streams per Apache Spark consente di creare applicazioni e pipeline di streaming che consumano e scrivono dati da Amazon Kinesis Data Streams. Il connettore supporta un maggiore consumo di fan-out con una velocità di trasmissione dedicata di lettura fino a 2 MB/secondo per shard. Per impostazione predefinita, Amazon EMR Serverless 7.1.0 e versioni successive includono il connettore, quindi non è necessario creare o scaricare pacchetti aggiuntivi. Per ulteriori informazioni sul connettore, consulta la spark-sql-kinesis-connector pagina su. GitHub

Di seguito è riportato un esempio di come avviare l'esecuzione di un processo con la dipendenza del connettore Kinesis Data Streams.

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

Per connettersi a Kinesis Data Streams, è necessario EMR configurare l'VPCapplicazione Serverless con accesso e VPC utilizzare un endpoint per consentire l'accesso privato oppure NAT utilizzare un gateway per ottenere l'accesso pubblico. Per ulteriori informazioni, consulta Configurazione dell'accesso. VPC È inoltre necessario assicurarsi che il ruolo di job runtime disponga delle autorizzazioni di lettura e scrittura necessarie per accedere ai flussi di dati richiesti. Per ulteriori informazioni su come configurare un ruolo Job runtime, consulta Job runtime roles for Amazon EMR Serverless. Per un elenco completo di tutte le autorizzazioni richieste, consulta la spark-sql-kinesis-connector pagina su. GitHub

Connettore Apache Kafka

Il connettore Apache Kafka per lo streaming strutturato di Spark è un connettore open source della community Spark ed è disponibile in un repository Maven. Questo connettore consente alle

applicazioni di streaming strutturato Spark di leggere e scrivere dati su Apache Kafka e Amazon Managed Streaming for Apache Kafka autogestiti. Per ulteriori informazioni sul connettore, consulta la Structured Streaming + Kafka Integration Guide nella documentazione di Apache Spark.

L'esempio seguente mostra come includere il connettore Kafka nella richiesta di esecuzione del lavoro.

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

La versione del connettore Apache Kafka dipende dalla versione EMR Serverless in uso e dalla versione Spark corrispondente. Per trovare la versione corretta di Kafka, consulta la Guida all'integrazione di Structured Streaming + Kafka.

Per utilizzare Amazon Managed Streaming for Apache IAM Kafka con autenticazione, devi includere un'altra dipendenza con cui consentire al connettore Kafka di connettersi ad Amazon. MSK IAM Per ulteriori informazioni, consulta il repository su. aws-msk-iam-auth GitHub È inoltre necessario assicurarsi che il ruolo Job Runtime disponga delle IAM autorizzazioni necessarie. L'esempio seguente mostra come utilizzare il connettore con IAM l'autenticazione.

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

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

Per utilizzare il connettore Kafka e la libreria di IAM autenticazione di Amazon, MSK devi configurare l'applicazione EMR Serverless con access. VPC Le sottoreti devono avere accesso a Internet e utilizzare un NAT gateway per accedere alle dipendenze Maven. Per ulteriori informazioni, vedere Configurazione dell'accesso. VPC Le sottoreti devono disporre di connettività di rete per accedere al cluster Kafka. Questo vale indipendentemente dal fatto che il cluster Kafka sia autogestito o che utilizzi Amazon Managed Streaming for Apache Kafka.

Gestione dei registri dei lavori in streaming

Rotazione del registro

I lavori di streaming supportano la rotazione dei log per i log delle applicazioni Spark e i log degli eventi. La rotazione dei log impedisce che i lavori di streaming lunghi generino file di registro di grandi dimensioni che potrebbero occupare tutto lo spazio disponibile su disco. La rotazione dei log consente di risparmiare spazio su disco e previene gli errori dei lavori dovuti allo scarso spazio su disco. Per ulteriori informazioni, vedere Rotazione dei log.

Compattazione dei tronchi

I lavori di streaming supportano anche la compattazione dei log dei log degli eventi Spark ogni volta che è disponibile la registrazione gestita. Per maggiori dettagli sulla registrazione gestita, consulta Registrazione con storage gestito. I processi di streaming possono durare a lungo e la quantità di dati sugli eventi può accumularsi nel tempo e aumentare notevolmente le dimensioni dei file di registro. Lo Spark History Server legge e carica questi eventi in memoria per l'interfaccia utente dell'applicazione Spark. Questo processo può causare latenze e costi elevati, soprattutto se i log degli eventi archiviati in Amazon S3 sono molto grandi.

La compattazione dei log riduce le dimensioni del registro degli eventi, quindi lo Spark History Server non deve caricare più di 1 GB di registri degli eventi in qualsiasi momento. Per ulteriori informazioni, consulta Monitoraggio e strumentazione nella documentazione di Apache Spark.

Offerte di lavoro Spark

È possibile eseguire i job Spark su un'applicazione con il `type` parametro impostato su. `SPARK` I lavori devono essere compatibili con la versione Spark compatibile con la versione di EMR rilascio di Amazon. Ad esempio, quando esegui lavori con Amazon EMR release 6.6.0, il tuo lavoro deve essere compatibile con Apache Spark 3.2.0. Per informazioni sulle versioni delle applicazioni per ogni versione, consulta. Versioni di rilascio di Amazon EMR Serverless

Parametri del job Spark

Quando si utilizza il `StartJobRun`APIper eseguire un job Spark, è possibile specificare i seguenti parametri.

Parametri obbligatori

- Ruolo di runtime del job Spark
- Parametro Spark Job Driver
- La configurazione di Spark sovrascrive il parametro
- Ottimizzazione dinamica dell'allocazione delle risorse Spark

Ruolo di runtime del job Spark

`executionRoleArn`Utilizzalo ARN per specificare il IAM ruolo utilizzato dall'applicazione per eseguire i job Spark. Questo ruolo deve contenere le seguenti autorizzazioni:

- Leggi dai bucket S3 o da altre fonti di dati in cui risiedono i tuoi dati
- Leggi i bucket o i prefissi S3 in cui risiede lo script o il file PySpark JAR
- Scrivi nei bucket S3 dove intendi scrivere l'output finale
- Scrivi i log in un bucket o prefisso S3 che specifica `S3MonitoringConfiguration`
- Accesso alle KMS chiavi se utilizzi KMS le chiavi per crittografare i dati nel tuo bucket S3
- Accesso a AWS Glue Data Catalog se usi Spark SQL

Se il tuo job Spark legge o scrive dati da o verso altre fonti di dati, specifica le autorizzazioni appropriate per questo ruolo. IAM Se non fornisci queste autorizzazioni al IAM ruolo, il lavoro potrebbe fallire. Per ulteriori informazioni, consulta Ruoli Job Runtime per Amazon EMR Serverless e Archiviazione dei log.

Parametro Spark Job Driver

`jobDriver`Utilizzato per fornire input al lavoro. Il parametro job driver accetta solo un valore per il tipo di processo che si desidera eseguire. Per un job Spark, il valore del parametro è`sparkSubmit`. Puoi usare questo tipo di lavoro per eseguire Scala, Java PySpark, SparkR e qualsiasi altro lavoro supportato tramite Spark submit. I job Spark hanno i seguenti parametri:

- **`sparkSubmitParameters`**— Questi sono i parametri Spark aggiuntivi che desideri inviare al job. Utilizzate questo parametro per sovrascrivere le proprietà predefinite di Spark, come la memoria del driver o il numero di esecutori, come quelle definite negli argomenti or. `--conf` `--class`
- **`entryPointArguments`**— Questa è una serie di argomenti che vuoi passare al tuo file principale JAR o Python. Dovrai gestire la lettura di questi parametri con il tuo codice di entrypoint. Separare ogni argomento dell'array con una virgola.
- **`entryPoint`**— Questo è il riferimento in Amazon S3 al file principale o JAR Python che desideri eseguire. Se stai usando Scala o JavaJAR, specifica la classe di ingresso principale nell'argomento `SparkSubmitParameters` using the`--class`.

Per ulteriori informazioni, consulta Avvio delle applicazioni con spark-submit.

La configurazione di Spark sovrascrive il parametro

Utilizzato **`configurationOverrides`**per sovrascrivere le proprietà di configurazione a livello di monitoraggio e a livello di applicazione. Questo parametro accetta un JSON oggetto con i due campi seguenti:

- **`monitoringConfiguration`**- Utilizza questo campo per specificare Amazon S3 URL (`s3MonitoringConfiguration`) in cui desideri che il job EMR Serverless memorizzi i log del tuo lavoro Spark. Assicurati di aver creato questo bucket con lo stesso Account AWS che ospita la tua applicazione e nella stessa Regione AWS dove si svolge il tuo lavoro.
- **`applicationConfiguration`**— Per sovrascrivere le configurazioni predefinite per le applicazioni, è possibile fornire un oggetto di configurazione in questo campo. È possibile 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. È possibile specificare più classificazioni per più applicazioni in un unico oggetto. JSON

> **Note**
>
> Le classificazioni di configurazione disponibili variano in base alla specifica versione EMR Serverless. Ad esempio, le classificazioni per Log4j personalizzate `spark-executor-log4j2` sono disponibili solo con le versioni 6.8.0 `spark-driver-log4j2` e successive.

Se usi la stessa configurazione in un'applicazione override e nei parametri di invio di Spark, i parametri di invio di Spark hanno la priorità. Le configurazioni hanno la priorità seguente, dalla più alta alla più bassa:

- Configurazione fornita da EMR Serverless al momento della creazione. `SparkSession`
- Configurazione fornita come parte dell'`sparkSubmitParameters--conf`argomento.
- La configurazione fornita come parte dell'applicazione ha la precedenza quando si avvia un lavoro.
- La configurazione fornita come parte della configurazione `runtimeConfiguration` quando si crea un'applicazione.
- Configurazioni ottimizzate EMR utilizzate da Amazon per il rilascio.
- Configurazioni open source predefinite per l'applicazione.

Per ulteriori informazioni sulla dichiarazione delle configurazioni a livello di applicazione e sulla sovrascrittura delle configurazioni durante l'esecuzione del processo, consulta. Configurazione predefinita dell'applicazione per Serverless EMR

Ottimizzazione dinamica dell'allocazione delle risorse Spark

Utilizza `dynamicAllocationOptimization` per ottimizzare l'utilizzo delle risorse in modalità EMR Serverless. L'impostazione di questa proprietà `true` nella classificazione della configurazione di Spark indica a EMR Serverless di ottimizzare l'allocazione delle risorse degli esecutori per allineare meglio la velocità con cui Spark richiede e annulla gli esecutori con la velocità con cui Serverless crea e rilascia i lavoratori. EMR In questo modo, EMR Serverless riutilizza in modo più ottimale i lavoratori su più fasi, con conseguente riduzione dei costi di esecuzione di lavori con più fasi mantenendo le stesse prestazioni.

Questa proprietà è disponibile in tutte le versioni di EMR rilascio di Amazon.

Di seguito è riportato un esempio di classificazione della configurazione con`dynamicAllocationOptimization`.

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

Considerate quanto segue se utilizzate l'ottimizzazione dinamica dell'allocazione:

- Questa ottimizzazione è disponibile per i job Spark per i quali hai abilitato l'allocazione dinamica delle risorse.
- Per ottenere la massima efficienza in termini di costi, consigliamo di configurare una scalabilità superiore limitata ai lavoratori utilizzando l'impostazione a livello di job `spark.dynamicAllocation.maxExecutors` o l'impostazione della capacità massima a livello di applicazione in base al carico di lavoro.
- Potresti non vedere un miglioramento dei costi nei lavori più semplici. Ad esempio, se il job viene eseguito su un piccolo set di dati o termina l'esecuzione in un'unica fase, Spark potrebbe non aver bisogno di un numero maggiore di executor o di più eventi di scalabilità.
- I lavori con una sequenza composta da una fase grande, fasi più piccole e poi ancora una fase grande potrebbero subire una regressione nella fase di esecuzione del processo. Poiché EMR Serverless utilizza le risorse in modo più efficiente, potrebbe comportare un minor numero di lavoratori disponibili per fasi più ampie, con conseguente maggiore autonomia.

Proprietà del lavoro Spark

La tabella seguente elenca le proprietà Spark opzionali e i relativi valori predefiniti che puoi sostituire quando invii un lavoro Spark.

Chiave	Descrizione	Valore predefinito
`spark.archives`	Un elenco di archivi separati da virgole che Spark estrae nella directory di lavoro di ogni esecutore. I tipi di file supportati includono, e. `.jar` `.tar.gz` `.tgz` `.zip` Per specificare il nome della directory da estrarre, aggiungil o # dopo il nome del file che desideri estrarre. Ad esempio `file.zip#directory` .	`NULL`
`spark.authenticate`	Opzione che attiva l'autenti cazione delle connessioni interne di Spark.	`TRUE`
`spark.driver.cores`	Il numero di core utilizzati dal driver.	4
`spark.driver.extra JavaOptions`	Opzioni Java aggiuntive per il driver Spark.	`NULL`
`spark.driver.memory`	La quantità di memoria utilizzata dal driver.	14 G
`spark.dynamicAlloc ation.enabled`	Opzione che attiva l'allocaz ione dinamica delle risorse. Questa opzione aumenta o riduce il numero di esecutori registrati con l'applicazione, in base al carico di lavoro.	`TRUE`
`spark.dynamicAlloc ation.executorIdle Timeout`	Il periodo di tempo in cui un executor può rimanere inattivo prima che Spark lo rimuova.	anni '60

Chiave	Descrizione	Valore predefinito
	Questo vale solo se attivi l'allocazione dinamica.	
`spark.dynamicAllocation.initialExecutors`	Il numero iniziale di esecutori da eseguire se si attiva l'allocazione dinamica.	`3`
`spark.dynamicAllocation.maxExecutors`	Il limite superiore per il numero di esecutori se si attiva l'allocazione dinamica.	Per 6.10.0 e versioni successive, `infinity` Per 6.9.0 e versioni precedenti, `100`
`spark.dynamicAllocation.minExecutors`	Il limite inferiore per il numero di esecutori se si attiva l'allocazione dinamica.	`0`
`spark.emr-serverless.allocation.batch.size`	Il numero di contenitori da richiedere in ogni ciclo di allocazione degli esecutori. C'è un intervallo di un secondo tra ogni ciclo di allocazione.	20
`spark.emr-serverless.driver.disk`	Il disco del driver Spark.	20G
`spark.emr-serverless.driverEnv.` *`[KEY]`*	Opzione che aggiunge variabili di ambiente al driver Spark.	`NULL`
`spark.emr-serverless.executor.disk`	Il disco Spark Executor.	20G
`spark.emr-serverless.memoryOverheadFactor`	Imposta il sovraccarico di memoria da aggiungere alla memoria del contenitore del driver e dell'executor.	0.1

Chiave	Descrizione	Valore predefinito
`spark.emr-serverless.driver.disk.type`	Il tipo di disco collegato al driver Spark.	Standard
`spark.emr-serverless.executor.disk.type`	Il tipo di disco collegato agli esecutori Spark.	Standard
`spark.executor.cores`	Il numero di core utilizzati da ogni executor.	4
`spark.executor.extraJavaOptions`	Opzioni Java aggiuntive per l'esecutore Spark.	`NULL`
`spark.executor.instances`	Il numero di contenitori Spark Executor da allocare.	3
`spark.executor.memory`	La quantità di memoria utilizzata da ogni executor.	14 G
`spark.executorEnv.`*`[KEY]`*	Opzione che aggiunge variabili di ambiente agli esecutori Spark.	`NULL`
`spark.files`	Un elenco di file separati da virgole da inserire nella directory di lavoro di ogni executor. È possibile accedere ai percorsi di questi file nell'esecutore con. `SparkFiles.get(`*`fileName`*`)`	`NULL`
`spark.hadoop.hive.metastore.client.factory.class`	La classe di implementazione del metastore Hive.	`NULL`

Chiave	Descrizione	Valore predefinito
`spark.jars`	Jar aggiuntivi da aggiungere al classpath di runtime del driver e degli executor.	`NULL`
`spark.network.crypto.enabled`	Opzione che attiva la crittografia basata. AES RPC Ciò include il protocollo di autenticazione aggiunto in Spark 2.2.0.	`FALSE`
`spark.sql.warehouse.dir`	La posizione predefinita per i database e le tabelle gestiti.	Il valore di `$PWD/spark-warehouse`
`spark.submit.pyFiles`	Un elenco separato da virgole di `.zip.egg`, o `.py` file da inserire nelle app per `PYTHONPATH` Python.	`NULL`

La tabella seguente elenca i parametri di invio Spark predefiniti.

Chiave	Descrizione	Valore predefinito
`archives`	Un elenco di archivi separati da virgole che Spark estrae nella directory di lavoro di ogni esecutore.	`NULL`
`class`	La classe principale dell'applicazione (per le app Java e Scala).	`NULL`
`conf`	Una proprietà di configurazione arbitraria di Spark.	`NULL`

Chiave	Descrizione	Valore predefinito
`driver-cores`	Il numero di core utilizzati dal driver.	4
`driver-memory`	La quantità di memoria utilizzata dal driver.	14 G
`executor-cores`	Il numero di core utilizzati da ogni esecutore.	4
`executor-memory`	La quantità di memoria utilizzata dall'esecutore.	14 G
`files`	Un elenco di file separati da virgole da inserire nella directory di lavoro di ogni esecutore. È possibile accedere ai percorsi di questi file nell'esec utore con. `SparkFile s.get( fileName)`	`NULL`
`jars`	Un elenco di jar separati da virgole da includere nei percorsi di classe del driver e dell'executor.	`NULL`
`num-executors`	Il numero di esecutori da avviare.	3
`py-files`	Un elenco separato da virgole di `.zip.egg`, o `.py` file da inserire nelle app per `PYTHONPATH` Python.	`NULL`
`verbose`	Opzione che attiva un output di debug aggiuntivo.	`NULL`

Esempi Spark

L'esempio seguente mostra come usare `StartJobRun` API per eseguire uno script Python. Per un end-to-end tutorial che utilizza questo esempio, vediGuida introduttiva ad Amazon EMR Serverless. Puoi trovare altri esempi su come eseguire PySpark lavori e aggiungere dipendenze Python nel repository EMRServerless Samples. GitHub

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

L'esempio seguente mostra come utilizzare per eseguire uno Spark `StartJobRun`API. JAR

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

Offerte di lavoro Hive

È possibile eseguire lavori Hive su un'applicazione con il `type` parametro impostato su. `HIVE` I lavori devono essere compatibili con la versione di Hive compatibile con la versione di EMR rilascio di Amazon. Ad esempio, quando esegui lavori su un'applicazione con Amazon EMR release

6.6.0, il tuo lavoro deve essere compatibile con Apache Hive 3.1.2. Per informazioni sulle versioni dell'applicazione per ogni versione, consulta. Versioni di rilascio di Amazon EMR Serverless

Parametri del lavoro Hive

Quando si utilizza il `StartJobRun`APIper eseguire un processo Hive, è necessario specificare i seguenti parametri.

Parametri obbligatori

- Ruolo di esecuzione del job Hive
- Parametro Hive job driver
- Parametro di sovrascrittura della configurazione Hive

Ruolo di esecuzione del job Hive

`executionRoleArn`Utilizzalo ARN per specificare il IAM ruolo utilizzato dall'applicazione per eseguire i job Hive. Questo ruolo deve contenere le seguenti autorizzazioni:

- Leggi dai bucket S3 o da altre fonti di dati in cui risiedono i tuoi dati
- Leggi dai bucket o dai prefissi S3 in cui risiedono il file di query Hive e il file di query init
- Leggi e scrivi nei bucket S3 in cui risiedono la directory Hive Scratch e la directory di magazzino Hive Metastore
- Scrivi nei bucket S3 in cui intendi scrivere l'output finale
- Scrivi i log in un bucket o prefisso S3 che specifica `S3MonitoringConfiguration`
- Accesso alle KMS chiavi se utilizzi KMS le chiavi per crittografare i dati nel tuo bucket S3
- Accesso a AWS Catalogo dati Glue

Se il tuo job Hive legge o scrive dati da o verso altre fonti di dati, specifica le autorizzazioni appropriate per questo ruolo. IAM Se non fornisci queste autorizzazioni al IAM ruolo, il tuo lavoro potrebbe fallire. Per ulteriori informazioni, consulta Ruoli Job Runtime per Amazon EMR Serverless.

Parametro Hive job driver

`jobDriver`Utilizzato per fornire input al lavoro. Il parametro job driver accetta solo un valore per il tipo di processo che si desidera eseguire. Quando specificate `hive` come tipo di lavoro, EMR Serverless passa una query Hive al `jobDriver` parametro. I job Hive hanno i seguenti parametri:

- **`query`**— Questo è il riferimento in Amazon S3 al file di query Hive che desideri eseguire.
- **`parameters`**— Queste sono le proprietà di configurazione aggiuntive di Hive che desideri sovrascrivere. Per sovrascrivere le proprietà, passale a questo parametro come. `--hiveconf` *`property=value`* Per sovrascrivere le variabili, passale a questo parametro come. `--hivevar` *`key=value`*
- **`initQueryFile`**— Questo è il file di query init Hive. Hive esegue questo file prima della query e può usarlo per inizializzare le tabelle.

Parametro di sovrascrittura della configurazione Hive

Utilizzato **`configurationOverrides`**per sovrascrivere le proprietà di configurazione a livello di monitoraggio e di applicazione. Questo parametro accetta un JSON oggetto con i due campi seguenti:

- **`monitoringConfiguration`**— Utilizza questo campo per specificare Amazon S3 URL (`s3MonitoringConfiguration`) in cui desideri che il job EMR Serverless memorizzi i log del tuo processo Hive. Assicurati di creare questo bucket con lo stesso Account AWS che ospita la tua applicazione e nella stessa Regione AWS dove si svolge il tuo lavoro.
- **`applicationConfiguration`**— È possibile fornire un oggetto di configurazione in questo campo per sovrascrivere le configurazioni predefinite per le applicazioni. È possibile 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. È possibile specificare più classificazioni per più applicazioni in un unico oggetto. JSON

> **Note**
>
> Le classificazioni di configurazione disponibili variano in base alla specifica versione EMR Serverless. Ad esempio, le classificazioni per Log4j personalizzate `spark-executor-log4j2` sono disponibili solo con le versioni 6.8.0 `spark-driver-log4j2` e successive.

Se si passa la stessa configurazione in un'applicazione override e nei parametri Hive, i parametri Hive hanno la priorità. L'elenco seguente classifica le configurazioni dalla priorità più alta alla priorità più bassa.

- Configurazione fornita come parte dei parametri Hive con. `--hiveconf property=value`
- La configurazione fornita come parte dell'applicazione ha la precedenza quando si avvia un lavoro.
- La configurazione fornita come parte della configurazione `runtimeConfiguration` quando si crea un'applicazione.
- Configurazioni ottimizzate che Amazon EMR assegna per il rilascio.
- Configurazioni open source predefinite per l'applicazione.

Per ulteriori informazioni sulla dichiarazione delle configurazioni a livello di applicazione e sulla sovrascrittura delle configurazioni durante l'esecuzione del processo, vedere. Configurazione predefinita dell'applicazione per Serverless EMR

Proprietà del lavoro di Hive

La tabella seguente elenca le proprietà obbligatorie che è necessario configurare quando si invia un lavoro Hive.

Impostazione	Descrizione
`hive.exec.scratchdir`	La posizione Amazon S3 in cui EMR Serverless crea file temporanei durante l'esecuzione del job Hive.
`hive.metastore.warehouse.dir`	La posizione Amazon S3 dei database per le tabelle gestite in Hive.

La tabella seguente elenca le proprietà opzionali di Hive e i relativi valori predefiniti che puoi sostituire quando invii un lavoro Hive.

Impostazione	Descrizione	Valore predefinito
`fs.s3.customAWSCredentialsProvider`	Il AWS Provider di credenziali che desideri utilizzare.	com.amazonaws.auth.d efaultAWSCredentialsProviderChain

Impostazione	Descrizione	Valore predefinito
`fs.s3a.aws.credentials.provider`	Il AWS Provider di credenziali che desideri utilizzare con un file system S3A.	com.amazonaws.auth.d efaultAWSCredentials ProviderChain
`hive.auto.convert.join`	Opzione che attiva la conversione automatica dei join comuni in mapjoin, in base alla dimensione del file di input.	`TRUE`
`hive.auto.convert.join.noconditionaltask`	Opzione che attiva l'ottimizzazione quando Hive converte un common join in un mapjoin in base alla dimensione del file di input.	`TRUE`
`hive.auto.convert.join.noconditionaltask.size`	Un join si converte direttamente in un mapjoin di dimensioni inferiori a questa dimensione.	Il valore ottimale viene calcolato in base alla memoria delle attività Tez
`hive.cbo.enable`	Opzione che attiva ottimizzazioni basate sui costi con il framework Calcite.	`TRUE`
`hive.cli.tez.session.async`	Opzione per avviare una sessione Tez in background durante la compilazione della query Hive. Se impostato su`false`, Tez AM si avvia dopo la compilazione della query Hive.	`TRUE`

Impostazione	Descrizione	Valore predefinito
`hive.compute.query.using.stats`	Opzione che attiva Hive per rispondere a determinate domande con statistiche memorizzate nel metastore. Per le statistiche di base, imposta su. `hive.stats.autogather TRUE` Per una raccolta di interrogazioni più avanzata, `analyze table queries` esegui.	`TRUE`
`hive.default.fileformat`	Il formato di file predefinito per le `CREATE TABLE` istruzioni. È possibile sovrascriverlo esplicitamente se lo si specifica `STORED AS [FORMAT]` nel comando. `CREATE TABLE`	`TEXTFILE`
`hive.driver.cores`	Il numero di core da utilizzare per il processo del driver Hive.	2
`hive.driver.disk`	La dimensione del disco per il driver Hive.	20G
`hive.driver.disk.type`	Il tipo di disco per il driver Hive.	Standard
`hive.tez.disk.type`	La dimensione del disco per i tez worker.	Standard

Impostazione	Descrizione	Valore predefinito
`hive.driver.memory`	La quantità di memoria da utilizzare per ogni processo del driver Hive. Hive CLI e Tez Application Master condivido no questa memoria in parti uguali con il 20% di margine di crescita.	6 G
`hive.emr-serverless.launch.env.[KEY]`	Opzione per impostare la variabile di *KEY* ambiente in tutti i processi specifici di Hive, come il driver Hive, Tez AM e l'attività Tez.	
`hive.exec.dynamic.partition`	Opzioni che attivano le partizioni dinamiche in/. DML DDL	`TRUE`
`hive.exec.dynamic.partition.mode`	Opzione che specifica se si desidera utilizzare la modalità rigorosa o la modalità non rigorosa. In modalità rigorosa, è necessario specificare almeno una partizione statica in caso di sovrascrittura accidentale di tutte le partizioni. In modalità non rigorosa, tutte le partizioni possono essere dinamiche.	`strict`
`hive.exec.max.dynamic.partitions`	Il numero massimo di partizioni dinamiche create da Hive in totale.	1000

Impostazione	Descrizione	Valore predefinito
`hive.exec.max.dynamic.partitions.per node`	Numero massimo di partizioni dinamiche che Hive crea in ogni nodo mapper e reducer.	100
`hive.exec.orc.split.strategy`	Prevede uno dei seguenti valori:, o. `BI` `ETL` `HYBRID` Questa non è una configurazione a livello utente. `BI`specifica che si desidera dedicare meno tempo alla generazione frazionata rispetto all'esecuzione delle query. `ETL`specifica che si desidera dedicare più tempo alla generazione frazionata. `HYBRID`specifica una scelta delle strategie di cui sopra in base all'euristica.	`HYBRID`
`hive.exec.reducers.bytes.per.reducer`	La dimensione di ogni riduttore. L'impostazione predefinita è 256 MB. Se la dimensione di input è 1G, il job utilizza 4 riduttori.	256000000
`hive.exec.reducers.max`	Il numero massimo di riduttori.	256

Impostazione	Descrizione	Valore predefinito
`hive.exec.stagingdir`	Il nome della directory che memorizza i file temporane i creati da Hive all'interno delle posizioni delle tabelle e nella posizione della directory scratch specificata nella proprietà. `hive.exec .scratchdir`	`.hive-staging`
`hive.fetch.task.co nversion`	Prevede uno dei seguenti valori: `NONEMINIMAL`, o. `MORE` Hive può convertire le interrogazioni di selezione in una singola attività. `FETCH` Ciò riduce al minimo la latenza.	`MORE`
`hive.groupby.posit ion.alias`	Opzione che fa sì che Hive utilizzi un alias di posizione della colonna nelle istruzioni. `GROUP BY`	`FALSE`
`hive.input.format`	Il formato di input predefini to. Imposta su `HiveInput Format` se riscontri problemi con`CombineHiveInputFo rmat` .	`org.apache.hadoop. hive.ql.io.Combine HiveInputFormat`
`hive.log.explain.o utput`	Opzione che attiva le spiegazioni dell'output esteso per qualsiasi query nel registro di Hive.	`FALSE`
`hive.log.level`	Il livello di registrazione di Hive.	`INFO`

Impostazione	Descrizione	Valore predefinito
`hive.mapred.reduce.tasks.speculative.execution`	Opzione che attiva il lancio speculativo dei riduttori. Supportato solo con Amazon EMR 6.10.x e versioni precedenti.	`TRUE`
`hive.max-task-containers`	Il numero massimo di contenitori simultanei. La memoria del mappatore configurata viene moltiplicata per questo valore per determinare la memoria disponibile utilizzata per il calcolo diviso e la priorità delle attività.	1000
`hive.merge.mapfiles`	Opzione che causa l'unione di file di piccole dimensioni alla fine di un lavoro basato sulla sola mappa.	`TRUE`
`hive.merge.size.per.task`	La dimensione dei file uniti alla fine del lavoro.	256000000
`hive.merge.tezfiles`	Opzione che attiva l'unione di file di piccole dimensioni alla fine di un Tez. DAG	`FALSE`
`hive.metastore.client.factory.class`	Il nome della classe factory che produce oggetti che implementano l'`IMetaStoreClient` interfaccia.	`com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory`

Impostazione	Descrizione	Valore predefinito
hive.metastore.glue.catalogid	Se il file AWS Glue Data Catalog funge da metastore ma viene eseguito in un altro Account AWS rispetto ai lavori, l'ID del Account AWS dove vengono eseguiti i lavori.	NULL
hive.metastore.uris	La parsimonia URI che il client metastore utilizza per connettersi al metastore remoto.	NULL
hive.optimize.ppd	Opzione che attiva il predicate pushdown.	TRUE
hive.optimize.ppd.storage	Opzione che attiva il pushdown dei predicati verso i gestori di archiviazione.	TRUE
hive.orderby.position.alias	Opzione che fa sì che Hive utilizzi un alias di posizione della colonna nelle istruzioni. ORDER BY	TRUE
hive.prewarm.enabled	Opzione che attiva il preriscaldamento del contenitore per Tez.	FALSE
hive.prewarm.numcontainers	Il numero di contenitori da preriscaldare per Tez.	10
hive.stats.autogather	Opzione che fa sì che Hive raccolga automaticamente le statistiche di base durante il comando. INSERT OVERWRITE	TRUE

Impostazione	Descrizione	Valore predefinito
`hive.stats.fetch.column.stats`	Opzione che disattiva il recupero delle statistiche delle colonne dal metastore. Il recupero delle statistiche sulle colonne può essere costoso quando il numero di colonne è elevato.	`FALSE`
`hive.stats.gather.num.threads`	Il numero di thread utilizzati dai comandi `partialscan` and `noscan` analysis per le tabelle partizionate. Questo vale solo per i formati di file che implementano `StatsProvidingRecordReader` (come). ORC	10
`hive.strict.checks.cartesian.product`	Opzioni che attivano controlli di join cartesiani rigorosi. Questi controlli non consentono un prodotto cartesiano (un cross join).	`FALSE`
`hive.strict.checks.type.safety`	Opzione che attiva rigorosi controlli di sicurezza tipografici e disattiva il confronto tra entrambi `string` e. `bigint` `double`	`TRUE`

Impostazione	Descrizione	Valore predefinito
`hive.support.quoted.identifiers`	Si aspetta un valore di `NONE` o`COLUMN`. `NONE`implica che solo i caratteri alfanumerici e i caratteri di sottolineatura siano validi negli identificatori. `COLUMN`implica che i nomi delle colonne possono contenere qualsiasi carattere.	`COLUMN`
`hive.tez.auto.reducer.parallelism`	Opzione che attiva la funzione di parallelismo del riduttore automatico Tez. Hive stima ancora le dimensioni dei dati e imposta le stime di parallelismo. Tez campiona le dimensioni di output dei vertici della sorgente e regola le stime in fase di esecuzione, se necessario.	`TRUE`
`hive.tez.container.size`	La quantità di memoria da utilizzare per il processo di operazione Tez.	6144
`hive.tez.cpu.vcores`	Il numero di core da utilizzare per ogni attività Tez.	2
`hive.tez.disk.size`	La dimensione del disco per ogni contenitore di attività.	20G
`hive.tez.input.format`	Il formato di input per la generazione di split in Tez AM.	`org.apache.hadoop.hive.ql.io.HiveInputFormat`

Impostazione	Descrizione	Valore predefinito
`hive.tez.min.partition.factor`	Limite inferiore di riduttori specificato da Tez quando si attiva il parallelismo del riduttore automatico.	0.25
`hive.vectorized.execution.enabled`	Opzione che attiva la modalità vettoriale di esecuzione delle query.	`TRUE`
`hive.vectorized.execution.reduce.enabled`	Opzione che attiva la modalità vettoriale del lato ridotto dell'esecuzione di una query.	`TRUE`
`javax.jdo.option.ConnectionDriverName`	Il nome della classe di driver per un metastore. JDBC	`org.apache.derby.jdbc.EmbeddedDriver`
`javax.jdo.option.ConnectionPassword`	La password associata a un database metastore.	`NULL`
`javax.jdo.option.ConnectionURL`	La stringa di JDBC connessione per un JDBC metastore.	`jdbc:derby:;databaseName=metastore_db;create=true`
`javax.jdo.option.ConnectionUserName`	Il nome utente associato a un database metastore.	`NULL`
`mapreduce.input.fileinputformat.split.maxsize`	La dimensione massima di una divisione durante il calcolo delle suddivisioni quando il formato di input è. `org.apache.hadoop.hive.ql.io.CombineHiveInputFormat` Il valore 0 indica l'assenza di limiti.	0

Impostazione	Descrizione	Valore predefinito
`tez.am.dag.cleanup.on.completion`	Opzione che attiva la pulizia casuale dei dati al termine. DAG	`TRUE`
`tez.am.emr-serverless.launch.env.[ KEY]`	Opzione per impostare la variabile di *KEY* ambiente nel processo Tez AM. Per Tez AM, questo valore sostituisce il valore. `hive.emr-serverless.launch.env.[ KEY]`	
`tez.am.log.level`	Il livello di registrazione root che EMR Serverless passa all'app master Tez.	`INFO`
`tez.am.sleep.time.before.exit.millis`	EMRServerless dovrebbe inviare ATS gli eventi dopo questo periodo di tempo dopo la richiesta di spegnimento di AM.	0
`tez.am.speculation.enabled`	Opzione che causa l'avvio speculativo di attività più lente. Ciò può aiutare a ridurre la latenza del lavoro quando alcune attività sono più lente a causa di macchine danneggiate o lente. Supportato solo con Amazon EMR 6.10.x e versioni precedenti.	`FALSE`

Impostazione	Descrizione	Valore predefinito
`tez.am.task.max.failed.attempts`	Il numero massimo di tentativi che possono fallire per una determinata attività prima che l'operazione abbia esito negativo. Questo numero non include i tentativi terminati manualmente.	3
`tez.am.vertex.cleanup.height`	Una distanza alla quale, se tutti i vertici dipendenti sono completi, Tez AM eliminerà i dati relativi allo shuffle dei vertici. Questa funzione è disattivata quando il valore è 0. EMRLe versioni di Amazon 6.8.0 e successive supportano questa funzionalità.	0
`tez.client.asynchronous-stop`	Opzione che fa sì che EMR Serverless invii gli ATS eventi prima che termini il driver Hive.	`FALSE`
`tez.grouping.max-size`	Il limite massimo di dimensione (in byte) di una divisione raggruppata. Questo limite impedisce suddivisioni eccessivamente grandi.	1073741824
`tez.grouping.min-size`	Il limite di dimensione inferiore (in byte) di una divisione raggruppata. Questo limite impedisce troppe suddivisioni di piccole dimensioni.	16777216

Impostazione	Descrizione	Valore predefinito
`tez.runtime.io.sort.mb`	La dimensione del soft buffer quando Tez ordina l'output viene ordinata.	Il valore ottimale viene calcolato in base alla memoria delle attività di Tez
`tez.runtime.unordered.output.buffer.size-mb`	La dimensione del buffer da usare se Tez non scrive direttamente su disco.	Il valore ottimale viene calcolato in base alla memoria delle attività di Tez
`tez.shuffle-vertex-manager.max-src-fraction`	La frazione di attività di origine che deve essere completata prima che EMR Serverless pianifichi tutte le attività per il vertice corrente (in caso di `ScatterGather` connessione). Il numero di attività pronte per la pianificazione sul vertice corrente varia linearmente tra e. `min-fraction` `max-fraction` Il valore predefinito è il valore predefinito o, a seconda del valore maggiore. `tez.shuffle-vertex-manager.min-src-fraction`	0.75
`tez.shuffle-vertex-manager.min-src-fraction`	La frazione di attività di origine che deve essere completata prima che EMR Serverless pianifichi le attività per il vertice corrente (in caso di connessione). `ScatterGather`	0.25

Impostazione	Descrizione	Valore predefinito
`tez.task.emr-serverless.launch.env.[ KEY]`	Opzione per impostare la variabile di *KEY* ambiente nel processo di attività Tez. Per le attività Tez, questo valore ha la precedenza sul valore. `hive.emr-serverless.launch.env.[ KEY]`	
`tez.task.log.level`	Il livello di registrazione principale che EMR Serverless passa alle attività Tez.	`INFO`
`tez.yarn.ats.event.flush.timeout.millis`	Il periodo massimo di attesa che AM deve attendere prima che gli eventi vengano eliminati prima dello spegnimento.	300000

Esempi di lavoro in Hive

Il seguente esempio di codice mostra come eseguire una query Hive con. `StartJobRun` API

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

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

Puoi trovare altri esempi di come eseguire i job Hive nel repository EMRServerless Samples. GitHub

EMRResilienza di Job serverless

EMRLe versioni Serverless 7.1.0 e successive includono il supporto per la resilienza dei job, in modo da riprovare automaticamente tutti i job non riusciti senza alcun intervento manuale da parte dell'utente. Un altro vantaggio della resilienza dei processi è che EMR Serverless sposta i job run in diverse zone di disponibilità (AZ) in caso di problemi in un'AZ.

Per abilitare la resilienza dei job, imposta la policy di riprova per il tuo job. Una politica di riprova assicura che EMR Serverless riavvii automaticamente un processo in caso di errore in qualsiasi momento. Le politiche di riprova sono supportate sia per i processi in batch che per quelli in streaming, quindi puoi personalizzare la resilienza dei processi in base al tuo caso d'uso. La tabella seguente confronta i comportamenti e le differenze di resilienza dei processi tra processi in batch e in streaming.

	Processi batch	Lavori in streaming
Comportamento predefinito	Non riesegue il lavoro.	Riprova sempre a eseguire il processo man mano che l'applicazione crea checkpoin t durante l'esecuzione del lavoro.
Punto di riprova	I processi Batch non hanno checkpoint, quindi EMR Serverless riesegue sempre il processo dall'inizio.	I processi di streaming supportano i checkpoin t, quindi puoi configurare la query di streaming per

	Processi batch	Lavori in streaming
		salvare lo stato di runtime e l'avanzamento verso una posizione di checkpoint in Amazon S3. EMRServerless riprende l'esecuzione del processo dal checkpoint. Per ulteriori informazioni, consulta Recovery from failure with Checkpointing nella documentazione di Apache Spark.
Numero massimo di tentativi	Consente un massimo di 10 nuovi tentativi.	I job di streaming dispongono di un sistema integrato di prevenzione degli errori, pertanto l'applicazione smette di riprovare i lavori se questi continuano a fallire dopo un'ora. Il numero predefinito di tentativi entro un'ora è di cinque tentativi. È possibile configurare questo numero di tentativi in modo che sia compreso tra 1 o 10. Non è possibile personalizzare il numero massimo di tentativi. Il valore 1 indica che non sono stati effettuati nuovi tentativi.

Quando EMR Serverless tenta di rieseguire un processo, lo indicizza anche con un numero di tentativo, in modo da poter tenere traccia del ciclo di vita di un processo in tutti i suoi tentativi.

È possibile utilizzare le operazioni Serverless o EMR API AWS CLI per modificare la resilienza del lavoro o visualizzare le informazioni relative alla resilienza del lavoro. Per ulteriori informazioni, consulta la guida EMRServerless API.

Per impostazione predefinita, EMR Serverless non esegue nuovamente i processi batch. Per abilitare i nuovi tentativi per i processi batch, configura il `maxAttempts` parametro quando si avvia l'esecuzione di un processo batch. Il `maxAttempts` parametro è applicabile solo ai processi batch. L'impostazione predefinita è 1, il che significa che non viene eseguito nuovamente il processo. I valori accettati sono compresi tra 1 e 10.

L'esempio seguente mostra come specificare un numero massimo di 10 tentativi all'avvio di un job run.

```
aws emr-serverless start-job-run
 --application-id <APPLICATION_ID> \
 --execution-role-arn <JOB_EXECUTION_ROLE> \
 --mode 'BATCH' \
 --retry-policy '{
    "maxAttempts": 10
 }' \
 --job-driver '{
    "sparkSubmit": {
         "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
         "entryPointArguments": ["1"],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
     }
}'
```

EMRServerless riprova a tempo indeterminato i processi di streaming se falliscono. Per evitare che si verifichino problemi dovuti a errori ripetuti e irrecuperabili, utilizzate il controllo di prevenzione delle interruzioni `maxFailedAttemptsPerHour` per lo streaming di nuovi tentativi di lavoro. Questo parametro consente di specificare il numero massimo di tentativi falliti consentiti entro un'ora prima che Serverless interrompa i nuovi tentativi. EMR L'impostazione predefinita è cinque. I valori accettati sono compresi tra 1 e 10.

```
aws emr-serverless start-job-run
 --application-id <APPPLICATION_ID> \
 --execution-role-arn <JOB_EXECUTION_ROLE> \
 --mode 'STREAMING' \
 --retry-policy '{
    "maxFailedAttemptsPerHour": 7
 }' \
 --job-driver '{
    "sparkSubmit": {
```

```
        "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
        "entryPointArguments": ["1"],
        "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
    }
}'
```

È inoltre possibile utilizzare le altre API operazioni di esecuzione dei lavori per ottenere informazioni sui lavori. Ad esempio, è possibile utilizzare il `attempt` parametro con l'`GetJobRun`operazione per ottenere dettagli su uno specifico tentativo di lavoro. Se non si include il `attempt` parametro, l'operazione restituisce informazioni sull'ultimo tentativo.

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

L'`ListJobRunAttempts`operazione restituisce informazioni su tutti i tentativi relativi all'esecuzione di un processo.

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

L'`GetDashboardForJobRun`operazione crea e restituisce un URL file che è possibile utilizzare per accedere all'applicazione UIs per l'esecuzione di un processo. Il `attempt` parametro consente di ottenere un valore URL per un tentativo specifico. Se non si include il `attempt` parametro, l'operazione restituisce informazioni sull'ultimo tentativo.

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

Monitoraggio di un processo con una policy di ripetizione

Il supporto per la resilienza dei lavori aggiunge anche il nuovo evento EMRServerless job run retry. EMRServerless pubblica questo evento a ogni nuovo tentativo del processo. È possibile utilizzare questa notifica per tenere traccia dei nuovi tentativi di lavoro. Per ulteriori informazioni sugli eventi, consulta Amazon EventBridge events.

Registrazione con politica di riprova

Ogni volta che EMR Serverless riprova un processo, il tentativo genera il proprio set di log. Per garantire che EMR Serverless possa inviare correttamente questi log ad Amazon S3 e CloudWatch Amazon senza EMR sovrascriverli, Serverless aggiunge un prefisso al formato del CloudWatch percorso e del nome del flusso di log di S3 per includere il numero del tentativo del processo.

Di seguito è riportato un esempio di come si presenta il formato.

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

Questo formato garantisce che EMR Serverless pubblichi tutti i log per ogni tentativo di lavoro nella propria posizione designata in Amazon S3 e. CloudWatch Per maggiori dettagli, consulta Archiviazione dei log.

> **Note**
>
> EMRServerless utilizza questo formato di prefisso solo con tutti i processi di streaming e tutti i processi batch con riprova abilitato.

Configurazione Metastore

Un metastore Hive è una posizione centralizzata che memorizza le informazioni strutturali sulle tabelle, inclusi schemi, nomi delle partizioni e tipi di dati. Con EMR Serverless, puoi mantenere i metadati di questa tabella in un metastore che ha accesso ai tuoi lavori.

Hai due opzioni per un metastore Hive:

- Il AWS Catalogo dati Glue
- Un metastore Apache Hive esterno

Utilizzo di AWS Glue Data Catalog come metastore

Puoi configurare i job Spark e Hive per utilizzare il AWS Glue Data Catalog come metastore. Consigliamo questa configurazione quando è necessario un metastore persistente o un metastore condiviso da diverse applicazioni, servizi o Account AWS. Per ulteriori informazioni sul Data Catalog,

vedere Population the AWS Catalogo dati Glue. Per informazioni su AWS Prezzi di Glue, vedi AWS Prezzi della colla.

Puoi configurare il tuo job EMR Serverless per utilizzare il AWS Glue Data Catalog nello stesso Account AWS come applicazione o in un'altra Account AWS.

Configura il AWS Catalogo dati Glue

Per configurare il Data Catalog, scegli il tipo di applicazione EMR Serverless che desideri utilizzare.

Spark

Quando utilizzi EMR Studio per eseguire i tuoi lavori con le applicazioni EMR Serverless Spark, AWS Glue Data Catalog è il metastore predefinito.

Quando si utilizza o SDKs AWS CLI, è possibile impostare la `spark.hadoop.hive.metastore.client.factory.class` configurazione su `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` nei `sparkSubmit` parametri dell'esecuzione del processo. L'esempio seguente mostra come configurare il Data Catalog con AWS CLI.

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

In alternativa, puoi impostare questa configurazione quando ne crei una nuova `SparkSession` nel codice Spark.

```
from pyspark.sql import SparkSession

spark = (
    SparkSession.builder.appName("SparkSQL")
```

```
    .config(
        "spark.hadoop.hive.metastore.client.factory.class",
        "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
    )
    .enableHiveSupport()
    .getOrCreate()
)

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

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

Hive

Per le applicazioni EMR Serverless Hive, il Data Catalog è il metastore predefinito. Cioè, quando si eseguono lavori su un'applicazione EMR Serverless Hive, Hive registra le informazioni sui metastore nel Data Catalog nella stessa Account AWS come applicazione. Non è necessario un cloud privato virtuale (VPC) per utilizzare Data Catalog come metastore.

Per accedere alle tabelle dei metastore di Hive, aggiungi il file richiesto AWS Le politiche di Glue descritte in Configurazione delle IAM autorizzazioni per AWS Glue.

Configura l'accesso tra più account per EMR Serverless e AWS Catalogo dati Glue

Per configurare l'accesso tra più account per EMR Serverless, devi prima accedere a quanto segue Account AWS:

- `AccountA`— Un Account AWS dove è stata creata un'applicazione EMR serverless.
- `AccountB`— Un Account AWS che contiene un AWS Glue Data Catalog a cui desideri che EMR acceda il tuo job Serverless.

1. Assicurati che un amministratore o un'altra identità autorizzata `AccountB` alleghi una politica delle risorse al Data Catalog in. `AccountB` Questa politica concede autorizzazioni `AccountA` specifiche per diversi account per eseguire operazioni sulle risorse del catalogo. `AccountB`

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

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

2. Aggiungi una IAM policy al ruolo EMR Serverless Job Runtime in `AccountA` modo che tale ruolo possa accedere alle risorse del Data Catalog in. `AccountB`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
```

```
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": ["arn:aws:glue:region:AccountB:catalog"]
    }
  ]
}
```

3. Inizia il tuo job run. Questo passaggio è leggermente diverso a seconda `AccountA` del tipo di applicazione EMR Serverless.

   Spark

   Impostate la `spark.hadoop.hive.metastore.glue.catalogid` proprietà nella `hive-site` classificazione come illustrato nell'esempio seguente. Replace (Sostituisci) *`Account-catalog-ID`* con l'ID del catalogo dati in. `AccountB`

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

Hive

Imposta la `hive.metastore.glue.catalogid` proprietà nella `hive-site` classificazione come illustrato nell'esempio seguente. Replace (Sostituisci) *`Account-catalog-ID`* con l'ID del catalogo dati in. `AccountB`

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

Considerazioni sull'utilizzo di AWS Catalogo dati Glue

Puoi aggiungere elementi ausiliari `ADD JAR` negli JARs script di Hive. Per ulteriori considerazioni, vedi Considerazioni sull'uso AWS Catalogo dati Glue.

Utilizzo di un metastore Hive esterno

Puoi configurare i job EMR Serverless Spark e Hive per connetterti a un metastore Hive esterno, come Amazon Aurora o Amazon for My. RDS SQL Questa sezione descrive come configurare un metastore Amazon RDS Hive, configurare e configurare i VPC job EMR Serverless per utilizzare un metastore esterno.

Crea un metastore Hive esterno

1. Crea un Amazon Virtual Private Cloud (AmazonVPC) con sottoreti private seguendo le istruzioni in Crea un. VPC
2. Crea la tua applicazione EMR Serverless con le tue nuove sottoreti Amazon VPC e private. Quando configuri un'applicazione EMR Serverless con aVPC, effettua innanzitutto il provisioning di un'interfaccia di rete elastica per ogni sottorete specificata. Quindi collega il gruppo di sicurezza specificato a quell'interfaccia di rete. Ciò consente il controllo dell'accesso all'applicazione. Per ulteriori dettagli su come configurare il tuoVPC, consultaConfigurazione dell'accesso VPC.
3. Crea un SQL database My SQL o Aurora Postgre in una sottorete privata in Amazon. VPC Per informazioni su come creare un RDS database Amazon, consulta Creazione di un'istanza Amazon RDS DB.
4. Modifica il gruppo di sicurezza del tuo database My SQL o Aurora per consentire JDBC le connessioni dal tuo gruppo di sicurezza EMR Serverless seguendo i passaggi descritti in Modificare un'istanza Amazon RDS DB. Aggiungi una regola per il traffico in entrata al gruppo di RDS sicurezza da uno dei tuoi EMR gruppi di sicurezza Serverless.

Tipo	Protocollo	Intervallo porte	Origine
Tutti TCP	TCP	3306	`emr-serverless-security-group`

Configura le opzioni Spark

Usando JDBC

Per configurare la tua applicazione EMR Serverless Spark per connettersi a un metastore Hive basato su un'istanza Amazon for My RDS o Amazon SQL Aurora My, usa una connessione. SQL JDBC Inserisci i parametri del `mariadb-connector-java.jar` `--jars` job run. `spark-submit`

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

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

Il seguente esempio di codice è uno script entrypoint Spark che interagisce con un metastore Hive su Amazon. RDS

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

```
spark.sql("SELECT count(*) FROM sampledb.sparknyctaxi").show()
spark.stop()
```

Utilizzo del servizio Thrift

Puoi configurare la tua applicazione EMR Serverless Hive per connettersi a un metastore Hive basato su un'istanza Amazon for My RDS o Amazon SQL Aurora My. SQL A tale scopo, esegui un server Thrift sul nodo master di un EMR cluster Amazon esistente. Questa opzione è ideale se disponi già di un EMR cluster Amazon con un server Thrift che desideri utilizzare per semplificare le configurazioni dei processi EMR Serverless.

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

Il seguente esempio di codice è uno script entrypoint (`thriftscript.py`) che utilizza il protocollo thrift per connettersi a un metastore Hive. Nota che la `hive.metastore.uris` proprietà deve essere impostata per la lettura da un metastore Hive esterno.

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
```

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

Configura le opzioni Hive

Usando JDBC

Se desideri specificare una posizione di database Hive esterna su un'istanza Amazon RDS My SQL o Amazon Aurora, puoi sovrascrivere la configurazione predefinita del metastore.

> **Note**
>
> In Hive, puoi eseguire più scritture su tabelle metastore contemporaneamente. Se condividete le informazioni sui metastore tra due job, assicuratevi di non scrivere contemporaneamente sulla stessa tabella di metastore a meno che non scriviate su partizioni diverse della stessa tabella di metastore.

Imposta le seguenti configurazioni nella classificazione per attivare il metastore Hive esternohive-site.

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

```
    }
}
```

Utilizzo di un server parsimonioso

Puoi configurare la tua applicazione EMR Serverless Hive per connettersi a un metastore Hive basato su Amazon for My RDS o Amazon Aurora M. SQL ySQLinstance A tale scopo, esegui un server Thrift sul nodo principale di un EMR cluster Amazon esistente. Questa opzione è ideale se disponi già di un EMR cluster Amazon che esegue un server Thrift e desideri utilizzare le configurazioni di lavoro EMR Serverless.

Imposta le seguenti configurazioni nella `hive-site` classificazione in modo che EMR Serverless possa accedere al thrift metastore remoto. Si noti che è necessario impostare la `hive.metastore.uris` proprietà per la lettura da un metastore Hive esterno.

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

Considerazioni sull'utilizzo di un metastore esterno

- Puoi configurare database compatibili con JDBC MariadB come metastore. Esempi di questi database sono RDS per MariadB, SQL My e Amazon Aurora.
- I metastore non vengono inizializzati automaticamente. Se il tuo metastore non è inizializzato con uno schema per la tua versione di Hive, usa lo strumento Hive Schema.
- EMRServerless non supporta l'autenticazione Kerberos. Non è possibile utilizzare un server metastore Thrift con autenticazione Kerberos con i job Serverless Spark o Hive. EMR

Accesso ai dati S3 in un altro AWS account di EMR Serverless

Puoi eseguire job Amazon EMR Serverless da un unico AWS account e configurali per accedere ai dati nei bucket Amazon S3 che appartengono a un altro AWS conto. Questa pagina descrive come configurare l'accesso tra account a S3 da EMR Serverless.

I lavori eseguiti su EMR Serverless possono utilizzare una policy di bucket S3 o un ruolo assunto per accedere ai dati in Amazon S3 da un altro AWS conto.

Prerequisiti

Per configurare l'accesso tra più account per Amazon EMR Serverless, devi completare le attività dopo aver effettuato l'accesso a due AWS account:

- **`AccountA`**— Questa è la AWS account in cui hai creato un'applicazione Amazon EMR Serverless. Prima di configurare l'accesso tra più account, devi avere a disposizione quanto segue in questo account:
  - Un'applicazione Amazon EMR Serverless in cui eseguire lavori.
  - Un ruolo di esecuzione del lavoro che dispone delle autorizzazioni necessarie per eseguire i lavori nell'applicazione. Per ulteriori informazioni, consulta Ruoli Job Runtime per Amazon EMR Serverless.
- **`AccountB`**— Questa è la AWS account che contiene il bucket S3 a cui desideri che i tuoi job Amazon EMR Serverless accedano.

Utilizza una policy sui bucket S3 per accedere ai dati S3 tra account

Per accedere al bucket S3 in account B from account A, collega la seguente policy al bucket S3 in account B.

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

```
      {
         "Sid": "Example permissions 2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB/*"
         ]
      }
   ]
}
```

Per ulteriori informazioni sull'accesso a più account S3 con le policy dei bucket S3, consulta l'Esempio 2: Il proprietario del bucket concede le autorizzazioni per i bucket tra account nella Guida per l'utente di Amazon Simple Storage Service.

Usa un ruolo presunto per accedere ai dati S3 tra più account

Un altro modo per configurare l'accesso tra più account per Amazon EMR Serverless è con l'`AssumeRole`azione di AWS Security Token Service (AWS STS). AWS STS è un servizio web globale che consente di richiedere credenziali temporanee con privilegi limitati per gli utenti. Puoi effettuare API chiamate verso EMR Serverless e Amazon S3 con le credenziali di sicurezza temporanee con cui crei. `AssumeRole`

I passaggi seguenti illustrano come utilizzare un ruolo presunto per accedere ai dati S3 tra account diversi da Serverless: EMR

1. Crea un bucket Amazon S3, *`cross-account-bucket`*, nel. `AccountB` Per ulteriori informazioni, consulta Creare un bucket nella Guida per l'utente di Amazon Simple Storage Service. 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 nella Amazon DynamoDB Developer Guide.
2. Crea un `Cross-Account-Role-B` IAM ruolo in grado di accedere a `AccountB` *`cross-account-bucket`*.

a. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.

b. Scegli Roles (Ruoli), quindi crea un nuovo ruolo: `Cross-Account-Role-B`. Per ulteriori informazioni su come creare IAM ruoli, consulta Creazione di IAM ruoli nella Guida per l'IAMutente.

c. Crea una IAM politica che specifichi le autorizzazioni per accedere `Cross-Account-Role-B` a *`cross-account-bucket`* Bucket S3, come dimostra la seguente dichiarazione politica. Quindi allega la IAM politica a. `Cross-Account-Role-B` Per ulteriori informazioni, consulta Creazione IAM di politiche nella Guida IAM per l'utente.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::cross-account-bucket",
                "arn:aws:s3:::cross-account-bucket/*"
            ]
        }
    ]
}
```

Se hai bisogno dell'accesso a DynamoDB, crea IAM una policy che specifichi le autorizzazioni per accedere alla tabella DynamoDB tra account. IAMQuindi `Cross-Account-Role-B` allega la policy a. Per ulteriori informazioni, consulta Amazon DynamoDB: consente l'accesso a una tabella specifica nella IAM Guida per l'utente.

Di seguito è riportata una politica per consentire l'accesso alla tabella DynamoDB`CrossAccountTable`.

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

```
        }
    ]
}
```

3. Modifica la relazione di fiducia per il ruolo `Cross-Account-Role-B`.
   a. Per configurare la relazione di fiducia per il ruolo, scegli la scheda Relazioni di fiducia nella IAM console relativa al ruolo `Cross-Account-Role-B` che hai creato nel passaggio 2.
   b. Seleziona Edit Trust Relationship (Modifica relazione di fiducia).
   c. Aggiungi il seguente documento di policy. Ciò consente `Job-Execution-Role-A` `AccountA` di assumere il `Cross-Account-Role-B` ruolo.

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

4. Grant `Job-Execution-Role-A` `AccountA` nel AWS STS `AssumeRole`permesso di presumere`Cross-Account-Role-B`.
   a. Nella IAM console per AWS account`AccountA`, seleziona`Job-Execution-Role-A`.
   b. Aggiungi la seguente istruzione di policy a `Job-Execution-Role-A` per autorizzare l'operazione `AssumeRole` nel ruolo `Cross-Account-Role-B`.

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

Esempi di ruoli presunti

Puoi utilizzare un singolo ruolo assunto per accedere a tutte le risorse S3 in un account oppure, con Amazon EMR 6.11 e versioni successive, puoi configurare più IAM ruoli da assumere quando accedi a diversi bucket S3 tra account diversi.

Argomenti

Accedi alle risorse S3 con un ruolo presunto

> **Note**
>
> Quando configuri un lavoro per utilizzare un singolo ruolo assunto, tutte le risorse S3 del job utilizzano quel ruolo, incluso lo `entryPoint` script.

Se desideri utilizzare un singolo ruolo assunto per accedere a tutte le risorse S3 nell'account B, specifica le seguenti configurazioni:

1. Specificare la EMRFS configurazione `fs.s3.customAWSCredentialsProvider` per. `spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAW`
2. Per Spark, usa `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` e specifica `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` le variabili di ambiente su driver ed executor.
3. Per Hive `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARNtez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`, usa e specifica le variabili `tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` di ambiente nei contenitori di attività Hive driver, Tez application master e Tez.

Gli esempi seguenti mostrano come utilizzare un ruolo presunto per avviare un processo EMR Serverless eseguito con accesso tra account.

Spark

L'esempio seguente mostra come utilizzare un ruolo presunto per avviare un job EMR Serverless Spark eseguito con accesso a S3 su più account.

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

Hive

L'esempio seguente mostra come utilizzare un ruolo assunto per avviare un processo EMR Serverless Hive eseguito con accesso a S3 su più account.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
```

```
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.task.emr-
serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

Accedi alle risorse S3 con più ruoli presunti

Con le versioni EMR Serverless 6.11.0 e successive, puoi configurare più IAM ruoli da assumere quando accedi a diversi bucket tra più account. Se desideri accedere a diverse risorse S3 con diversi ruoli assunti nell'account B, usa le seguenti configurazioni all'avvio del job run:

1. Specificare la EMRFS configurazione `fs.s3.customAWSCredentialsProvider` per. `com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredenti`
2. Specificate la EMRFS configurazione `fs.s3.bucketLevelAssumeRoleMapping` per definire la mappatura dal nome del bucket S3 al IAM ruolo da assumere nell'account B. Il valore deve essere nel formato di. `bucket1->role1;bucket2->role2`

Ad esempio, è possibile utilizzare `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` per accedere al bucket `bucket1` e utilizzare `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` per accedere al `bucket2` bucket. Gli esempi seguenti mostrano come avviare un processo EMR Serverless con accesso a più account tramite più ruoli presunti.

Spark

L'esempio seguente mostra come utilizzare più ruoli presunti per creare un'esecuzione di job Spark EMR Serverless.

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

Hive

Gli esempi seguenti mostrano come utilizzare più ruoli presunti per creare un'esecuzione di job EMR Serverless Hive.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
```

```
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "fs.s3.bucketLevelAssumeRoleMapping": "bucket1-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"
            }
        }]
    }'
```

Risoluzione degli errori in EMR Serverless

Utilizza le seguenti informazioni per diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con Amazon EMR Serverless.

Argomenti

Errore: limite superato per la capacità massima consentita.

Questo errore indica che EMR Serverless non è riuscito a inviare il lavoro perché l'applicazione ha superato i limiti di capacità massima configurati. Aumenta i limiti di capacità massima per l'applicazione.

Errore: la capacità massima configurata è stata superata. Please try again later. (Per favore, riprova più tardi)

Questo errore indica che EMR Serverless non è riuscito ad avviare un nuovo processo perché l'applicazione ha superato i limiti di capacità massima configurati. Aumenta i limiti di capacità massima per l'applicazione.

Errore: l'accesso a S3 è negato. Controlla le autorizzazioni di accesso S3 del ruolo Job Runtime sulle risorse S3 richieste.

Questo errore indica che il tuo job non ha accesso alle tue risorse S3. Verifica che il ruolo Job Runtime sia autorizzato ad accedere alle risorse S3 che il job deve utilizzare. Per ulteriori informazioni sui ruoli di runtime, consultaRuoli Job Runtime per Amazon EMR Serverless.

Errore ModuleNotFoundError: nessun modulo denominato<module>. Fai riferimento alla guida per l'utente su come usare le librerie python con EMR Serverless.

Questo errore indica che un modulo Python non era disponibile per il job Spark. Verifica che le librerie Python dipendenti siano disponibili per il lavoro. Per ulteriori informazioni su come impacchettare le librerie Python, vedere. Utilizzo delle librerie Python con Serverless EMR

Errore: impossibile assumere il ruolo di esecuzione <role name>perché non esiste o non è configurato con la relazione di trust richiesta.

Questo errore indica che il ruolo di runtime del job specificato per il job non esiste o che il ruolo non ha una relazione di trust per le autorizzazioni EMR Serverless. Per verificare che il IAM ruolo esista e confermare di aver impostato correttamente la politica di fiducia del ruolo, consulta le istruzioni in. Ruoli Job Runtime per Amazon EMR Serverless

Esegui carichi di lavoro interattivi con EMR Serverless tramite Studio EMR

Panoramica

Un'applicazione interattiva è un'applicazione EMR serverless con funzionalità interattive abilitate. Con le applicazioni interattive Amazon EMR Serverless, puoi eseguire carichi di lavoro interattivi con notebook Jupyter gestiti in Amazon Studio. EMR Questo aiuta i data engineer, i data scientist e gli analisti di dati a utilizzare EMR Studio per eseguire analisi interattive con set di dati in archivi di dati come Amazon S3 e Amazon DynamoDB.

I casi d'uso per applicazioni interattive in EMR Serverless includono i seguenti:

- I data engineer utilizzano l'IDEesperienza di EMR Studio per creare uno ETL script. Lo script acquisisce i dati dall'ambiente locale, li trasforma per l'analisi e li archivia in Amazon S3.
- I data scientist utilizzano i notebook per esplorare i set di dati e addestrare modelli di apprendimento automatico (ML) per rilevare anomalie nei set di dati.
- Gli analisti di dati esplorano i set di dati e creano script che generano report giornalieri per aggiornare applicazioni come i dashboard aziendali.

Prerequisiti

Per utilizzare carichi di lavoro interattivi con EMR Serverless, devi soddisfare i seguenti requisiti:

- EMRLe applicazioni interattive serverless sono supportate con Amazon EMR 6.14.0 e versioni successive.
- Per accedere alla tua applicazione interattiva, eseguire i carichi di lavoro inviati ed eseguire notebook interattivi da EMR Studio, hai bisogno di autorizzazioni e ruoli specifici. Per ulteriori informazioni, consulta Autorizzazioni richieste per i carichi di lavoro interattivi.

Autorizzazioni richieste per i carichi di lavoro interattivi

Oltre alle autorizzazioni di base necessarie per accedere a EMR Serverless, devi configurare autorizzazioni aggiuntive per la tua identità o il tuo ruolo: IAM

Per accedere alla tua applicazione interattiva

Configura le autorizzazioni utente e Workspace per EMR Studio. Per ulteriori informazioni, consulta Configurare le autorizzazioni utente di EMR Studio nella Amazon EMR Management Guide.

Per eseguire i carichi di lavoro inviati con Serverless EMR

Imposta un ruolo di job runtime. Per ulteriori informazioni, consulta Creare un ruolo di job runtime.

Per eseguire i taccuini interattivi di Studio EMR

Aggiungi le seguenti autorizzazioni aggiuntive alla IAM politica per gli utenti di Studio:

- **`emr-serverless:AccessInteractiveEndpoints`**- Concede l'autorizzazione per accedere e connettersi all'applicazione interattiva specificata come. `Resource` Questa autorizzazione è necessaria per collegarsi a un'applicazione EMR Serverless da uno EMR Studio Workspace.
- **`iam:PassRole`**- Concede l'autorizzazione ad accedere al ruolo di IAM esecuzione che intendi utilizzare quando ti colleghi a un'applicazione. È richiesta l'`PassRole`autorizzazione appropriata per collegarsi a un'applicazione EMR Serverless da uno EMR Studio Workspace.

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

Configurazione di applicazioni interattive

Utilizza i seguenti passaggi di alto livello per creare un'applicazione EMR Serverless con funzionalità interattive di Amazon EMR Studio nel AWS Management Console.

1. Segui i passaggi indicati Guida introduttiva ad Amazon EMR Serverless per creare un'applicazione.
2. Quindi, avvia un'area di lavoro da EMR Studio e collegala a un'applicazione EMR Serverless come opzione di elaborazione. Per ulteriori informazioni, consulta la scheda Carico di lavoro interattivo nella fase 2 della documentazione EMRServerless Getting Started.

Quando si collega un'applicazione a Studio Workspace, l'avvio dell'applicazione si attiva automaticamente se non è già in esecuzione. È inoltre possibile preavviare l'applicazione e tenerla pronta prima di collegarla al Workspace.

Considerazioni sulle applicazioni interattive

- EMRLe applicazioni interattive serverless sono supportate con Amazon EMR 6.14.0 e versioni successive.
- EMRStudio è l'unico client integrato con le applicazioni interattive EMR Serverless. Le seguenti funzionalità di EMR Studio non sono supportate dalle applicazioni interattive EMR Serverless: collaborazione in Workspace, SQL Explorer ed esecuzione programmatica di notebook.
- Le applicazioni interattive sono supportate solo per il motore Spark.
- Le applicazioni interattive supportano i kernel Python 3 PySpark e Spark Scala.
- È possibile eseguire fino a 25 notebook simultanei su una singola applicazione interattiva.
- Non esiste un endpoint o un'APIinterfaccia che supporti notebook Jupyter ospitati autonomamente con applicazioni interattive.
- Per un'esperienza di avvio ottimizzata, si consiglia di configurare la capacità preinizializzata per driver ed esecutori e di preavviare l'applicazione. Quando preavvii l'applicazione, ti assicuri che sia pronta quando desideri collegarla al tuo Workspace.

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

- Per impostazione predefinita, `autoStopConfig` è abilitato per le applicazioni. Questa operazione chiude l'applicazione dopo 30 minuti di inattività. Puoi modificare questa configurazione come parte della tua richiesta`create-application`. `update-application`
- Quando si utilizza un'applicazione interattiva, si consiglia di configurare una capacità preinizializzata di kernel, driver ed esecutori per far funzionare i notebook. Ogni sessione interattiva Spark richiede un kernel e un driver, quindi EMR Serverless mantiene un kernel worker preinizializzato per ogni driver preinizializzato. Per impostazione predefinita, EMR Serverless mantiene la capacità preinizializzata di un kernel worker per l'intera applicazione anche se non si specifica alcuna capacità preinizializzata per i driver. Ogni kernel worker utilizza 4 v e 16 GB di memoria. CPU Per informazioni aggiornate sui prezzi, consulta la pagina EMRdei prezzi di Amazon.
- Devi avere una quota di CPU servizio v sufficiente nel tuo Account AWS per eseguire carichi di lavoro interattivi. Se non esegui carichi di lavoro compatibili con Lake Formation, ti consigliamo almeno 24 v. CPU Se lo fai, ti consigliamo almeno 28 v. CPU
- EMRServerless interrompe automaticamente i kernel dai notebook se sono rimasti inattivi per più di 60 minuti. EMRServerless calcola il tempo di inattività del kernel a partire dall'ultima attività completata durante la sessione del notebook. Al momento non è possibile modificare l'impostazione del timeout di inattività del kernel.
- Per abilitare Lake Formation con carichi di lavoro interattivi, imposta la configurazione `spark.emr-serverless.lakeformation.enabled` su `true` sotto la `spark-defaults` classificazione nell'`runtime-configuration`oggetto quando crei un'applicazione EMR Serverless. Per ulteriori informazioni sull'abilitazione di Lake Formation in EMR Serverless, consulta Enabling Lake Formation in Amazon EMR.

Esegui carichi di lavoro interattivi con EMR Serverless tramite un endpoint Apache Livy

Con le EMR versioni 6.14.0 e successive di Amazon, puoi creare e abilitare un endpoint Apache Livy mentre crei un'applicazione EMR Serverless ed eseguire carichi di lavoro interattivi tramite notebook ospitati autonomamente o con un client personalizzato. Un endpoint Apache Livy offre i seguenti vantaggi:

- Puoi connetterti in modo sicuro a un endpoint Apache Livy tramite i notebook Jupyter e gestire i carichi di lavoro Apache Spark con l'interfaccia di Apache Livy. REST

- Usa le operazioni Apache Livy per applicazioni web interattive che utilizzano i dati dei carichi di lavoro Apache Spark. REST API

Prerequisiti

Per utilizzare un endpoint Apache Livy con EMR Serverless, devi soddisfare i seguenti requisiti:

- Completa la procedura descritta in Guida introduttiva ad Amazon EMR Serverless.
- Per eseguire carichi di lavoro interattivi tramite gli endpoint Apache Livy, sono necessari determinati permessi e ruoli. Per ulteriori informazioni, consulta Autorizzazioni richieste per carichi di lavoro interattivi.

Autorizzazioni richieste

Oltre alle autorizzazioni richieste per accedere a EMR Serverless, devi aggiungere anche le seguenti autorizzazioni al tuo IAM ruolo per accedere a un endpoint Apache Livy ed eseguire applicazioni:

- `emr-serverless:AccessLivyEndpoints`— concede l'autorizzazione per accedere e connettersi all'applicazione abilitata per Livy specificata come. `Resource` È necessaria questa autorizzazione per eseguire REST API le operazioni disponibili dall'endpoint Apache Livy.
- `iam:PassRole`— concede l'autorizzazione ad accedere al ruolo di IAM esecuzione durante la creazione della sessione Apache Livy. EMRServerless utilizzerà questo ruolo per eseguire i tuoi carichi di lavoro.
- `emr-serverless:GetDashboardForJobRun`— concede l'autorizzazione a generare l'interfaccia utente di Spark Live e i link ai registri dei driver e fornisce l'accesso ai log come parte dei risultati della sessione di Apache Livy.

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

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

Nozioni di base

1. Per creare un'applicazione compatibile con Apache Livy, esegui il comando seguente.

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

2. Dopo che EMR Serverless ha creato l'applicazione, avviatela per rendere disponibile l'endpoint Apache Livy.

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

   Utilizzate il seguente comando per verificare lo stato della vostra applicazione. Una volta raggiunto lo statoSTARTED, puoi accedere all'endpoint Apache Livy.

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

3. Usa quanto segue URL per accedere all'endpoint:

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

Una volta che l'endpoint è pronto, puoi inviare carichi di lavoro in base al tuo caso d'uso. È necessario firmare ogni richiesta all'endpoint con il SIGv4 protocollo e inserire un'intestazione di autorizzazione. È possibile utilizzare i seguenti metodi per eseguire carichi di lavoro:

- HTTPclient: è necessario inviare le API operazioni relative agli endpoint Apache Livy con un client personalizzato. HTTP
- Kernel Sparkmagic: è necessario eseguire localmente il kernel sparkmagic e inviare query interattive con i notebook Jupyter.

HTTPclienti

Per creare una sessione di Apache Livy, devi inserire `emr-serverless.session.executionRoleArn` il `conf` parametro del corpo della richiesta. L'esempio seguente è un esempio `POST /sessions` di richiesta.

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

La tabella seguente descrive tutte le operazioni di Apache Livy API disponibili.

APIoperazione	Descrizione
GET/sessioni	Restituisce un elenco di tutte le sessioni interattive attive.
POST/sessioni	Crea una nuova sessione interattiva tramite spark o pyspark.
GET/sessioni/ <*sessionId* >	Restituisce le informazioni sulla sessione.

APIoperazione	Descrizione
GET/sessions/ <*sessionId* /sessions//stato	Restituisce lo stato della sessione.
DELETE/sessions/ <*sessionId* >	Interrompe ed elimina la sessione.
GET/sessioni/ <*sessionId* /sessions//dichiarazioni	Restituisce tutte le istruzioni di una sessione.
POST/sessioni/ <*sessionId* /sessions//dichiarazioni	Esegue un'istruzione in una sessione.
GET/sessioni/ <*sessionId* /sessions//dichiarazioni/<*statementId* >	Restituisce i dettagli dell'istruzione specificata in una sessione.
POST/sessions/ <*sessionId* /sessions//dichiarazioni/<*statementId* >/dichiarazioni//annulla	Annulla l'istruzione specificata in questa sessione.

Invio di richieste all'endpoint Apache Livy

È inoltre possibile inviare richieste direttamente all'endpoint Apache Livy da un client. HTTP In questo modo è possibile eseguire in remoto il codice per i casi d'uso al di fuori di un notebook.

Prima di iniziare a inviare richieste all'endpoint, assicurati di aver installato le seguenti librerie:

```
pip3 install botocore awscrt requests
```

Di seguito è riportato un esempio di script Python per inviare HTTP richieste direttamente a un endpoint:

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

endpoint = 'https://<application_id>.livy.emr-serverless-
services-<AWS_REGION>.amazonaws.com'
```

```
headers = {'Content-Type': 'application/json'}

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


### Create session request

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

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

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

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

pprint.pprint(r.json())


### List Sessions Request

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

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

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


### Get session state

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

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

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

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

pprint.pprint(r3.json())


### Submit Statement

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

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

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

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

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

pprint.pprint(r4.json())

### Check statements results

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

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

request.context["payload_signing_enabled"] = False

signer.add_auth(request)
```

```
prepped = request.prepare()

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

pprint.pprint(r5.json())


### Delete session


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

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

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

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

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

Kernel Sparkmagic

Prima di installare sparkmagic, assicurati di aver configurato AWS credenziali nell'istanza in cui vuoi installare sparkmagic

1. Installa sparkmagic seguendo i passaggi di installazione. Nota che devi solo eseguire i primi quattro passaggi.
2. Il kernel sparkmagic supporta autenticatori personalizzati, quindi puoi integrare un autenticatore con il kernel sparkmagic in modo che ogni richiesta venga firmata. SIGv4
3. EMRInstalla l'autenticatore personalizzato Serverless.

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

4. Ora fornisci il percorso dell'autenticatore personalizzato e dell'endpoint URL Apache Livy nel file json di configurazione sparkmagic. Usa il seguente comando per aprire il file di configurazione.

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

Di seguito è riportato un `config.json` file di esempio.

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

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

5. Avvia Jupyter lab. Dovrebbe utilizzare l'autenticazione personalizzata che hai impostato nell'ultimo passaggio.
6. Puoi quindi eseguire i seguenti comandi del notebook e il tuo codice per iniziare.

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

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

```
    }
}
```

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

Internamente, ogni istruzione richiama ciascuna delle API operazioni di Apache Livy tramite l'endpoint Apache Livy configurato. URL È quindi possibile scrivere le istruzioni in base al caso d'uso.

Considerazioni

Prendi in considerazione le seguenti considerazioni quando esegui carichi di lavoro interattivi tramite endpoint Apache Livy.

- EMRServerless mantiene l'isolamento a livello di sessione utilizzando il caller principal. Il principale chiamante che crea la sessione è l'unico che può accedere a tale sessione. Per un isolamento più granulare, puoi configurare un'identità di origine quando assumi le credenziali. In questo caso, EMR Serverless applica l'isolamento a livello di sessione in base sia al principale chiamante che all'identità di origine. Per ulteriori informazioni sull'identità di origine, consulta Monitoraggio e controllo delle azioni intraprese con i ruoli presunti.
- Gli endpoint Apache Livy sono supportati dalle versioni EMR Serverless 6.14.0 e successive.
- Gli endpoint Apache Livy sono supportati solo per il motore Apache Spark.
- Gli endpoint Apache Livy supportano Scala Spark e. PySpark
- Per impostazione predefinita, `autoStopConfig` è abilitato nelle tue applicazioni. Ciò significa che le applicazioni si chiudono dopo 15 minuti di inattività. Puoi modificare questa configurazione come parte della tua `create-application` `update-application` richiesta.
- È possibile eseguire fino a 25 sessioni simultanee su una singola applicazione abilitata per endpoint Apache Livy.
- Per una migliore esperienza di avvio, si consiglia di configurare la capacità preinizializzata per driver ed esecutori.
- È necessario avviare manualmente l'applicazione prima di connettersi all'endpoint Apache Livy.
- È necessario disporre di una quota di CPU servizio v sufficiente nel Account AWS per eseguire carichi di lavoro interattivi con l'endpoint Apache Livy. Consigliamo almeno 24 v. CPU
- Il timeout di sessione predefinito di Apache Livy è di 1 ora. Se non si eseguono istruzioni per un'ora, Apache Livy elimina la sessione e rilascia il driver e gli esecutori. Non è possibile modificare questa configurazione.

- Solo le sessioni attive possono interagire con un endpoint Apache Livy. Una volta terminata, annullata o terminata la sessione, non è possibile accedervi tramite l'endpoint Apache Livy.

Registrazione di log e monitoraggio

Il monitoraggio è un elemento importante per mantenere l'affidabilità, la disponibilità e le prestazioni delle applicazioni e dei EMR lavori serverless. È necessario raccogliere i dati di monitoraggio da tutte le parti delle soluzioni EMR Serverless in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica uno.

Argomenti

Archiviazione dei log

Per monitorare l'avanzamento del lavoro su EMR Serverless e risolvere i problemi relativi ai processi, è possibile scegliere in che modo EMR Serverless archivia e pubblica i log delle applicazioni. Quando invii un job run, puoi specificare storage gestito, Amazon S3 e Amazon CloudWatch come opzioni di registrazione.

Con CloudWatch, puoi specificare i tipi e le posizioni di registro che desideri utilizzare o accettare i tipi e le posizioni predefiniti. Per ulteriori informazioni sui CloudWatch log, vederethe section called “Amazon CloudWatch”. Per quanto riguarda lo storage gestito e la registrazione S3, la tabella seguente mostra le posizioni dei log e la disponibilità dell'interfaccia utente che puoi aspettarti se scegli lo storage gestito, i bucket Amazon S3 o entrambi.

Opzione	Registri degli eventi	Registri di container	Interfaccia utente dell'applicazione
Archiviazione gestita	Archiviata in uno storage gestito	Archiviata in un sistema di storage gestito	Supportato

Opzione	Registri degli eventi	Registri di container	Interfaccia utente dell'applicazione
Storage gestito e bucket S3	Archiviati in entrambi i posti	Memorizzato in un bucket S3	Supportato
Bucket Amazon S3	Memorizzato nel bucket S3	Memorizzato nel bucket S3	Non supportato 1

[1] Si consiglia di mantenere selezionata l'opzione di archiviazione gestita. In caso contrario, non è possibile utilizzare l'applicazione integrataUIs.

Registrazione per EMR Serverless con storage gestito

Per impostazione predefinita, EMR Serverless archivia i log delle applicazioni in modo sicuro nello storage EMR gestito di Amazon per un massimo di 30 giorni.

> **Note**
>
> Se disattivi l'opzione predefinita, Amazon non EMR può risolvere i tuoi lavori per tuo conto.

Per disattivare questa opzione da EMR Studio, deseleziona Consenti AWS per conservare i log per 30 giorni nella sezione Impostazioni aggiuntive della pagina Invia offerta di lavoro.

Per disattivare questa opzione dal AWS CLI, utilizza la `managedPersistenceMonitoringConfiguration` configurazione quando invii l'esecuzione di un job.

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

Registrazione per EMR Serverless con bucket Amazon S3

Prima che i tuoi lavori possano inviare dati di log ad Amazon S3, devi includere le seguenti autorizzazioni nella politica di autorizzazione per il ruolo di esecuzione del lavoro. Sostituisci *DOC-EXAMPLE-BUCKET-LOGGING* con il nome del bucket di accesso.

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

Per configurare un bucket Amazon S3 per archiviare i log provenienti da AWS CLI, usa la `s3MonitoringConfiguration` configurazione quando avvii l'esecuzione di un job. A tale scopo, fornite quanto segue `--configuration-overrides` nella configurazione.

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

Per i processi batch che non hanno i nuovi tentativi abilitati, EMR Serverless invia i log al seguente percorso:

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

EMRLe versioni Serverless 7.1.0 e successive supportano i tentativi di ripetizione per processi di streaming e processi in batch. Se si esegue un processo con i tentativi abilitati, EMR Serverless

aggiunge automaticamente un numero di tentativo al prefisso del percorso di registro, in modo da poter distinguere e tenere traccia meglio dei log.

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

Registrazione per sistemi EMR Serverless con Amazon CloudWatch

Quando invii un lavoro a un'applicazione EMR Serverless, puoi scegliere Amazon CloudWatch come opzione per archiviare i log delle tue applicazioni. Ciò consente di utilizzare funzionalità di analisi dei CloudWatch log come CloudWatch Logs Insights e Live Tail. Puoi anche trasmettere i log da altri sistemi, CloudWatch ad esempio OpenSearch per ulteriori analisi.

EMRServerless fornisce la registrazione in tempo reale dei registri dei driver. È possibile visualizzare i log in tempo reale con la funzionalità CloudWatch live tail o tramite i comandi tail. CloudWatch CLI

Per impostazione predefinita, la CloudWatch registrazione è disabilitata per EMR Serverless. Per abilitarlo, consulta la configurazione in. AWS CLI

> **Note**
>
> Amazon CloudWatch pubblica i log in tempo reale, quindi richiede più risorse ai lavoratori. Se scegli una capacità di manodopera ridotta, l'impatto sulla durata del lavoro potrebbe aumentare. Se abiliti CloudWatch la registrazione, ti consigliamo di scegliere una maggiore capacità di lavoro. È anche possibile che la pubblicazione dei log rallenti se la frequenza delle transazioni al secondo (TPS) è troppo bassa per. `PutLogEvents` La configurazione di CloudWatch limitazione è globale per tutti i servizi, incluso Serverless. EMR Per ulteriori informazioni, vedi Come posso determinare la limitazione nei miei log? CloudWatch su AWS re:post.

Autorizzazioni richieste per la registrazione con CloudWatch

Prima che i tuoi lavori possano inviare dati di log ad Amazon CloudWatch, devi includere le seguenti autorizzazioni nella politica di autorizzazione per il ruolo Job Runtime.

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

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

AWS CLI

Per configurare Amazon CloudWatch per archiviare i log per EMR Serverless da AWS CLI, usa la `cloudWatchLoggingConfiguration` configurazione quando avvii l'esecuzione di un job. A tale scopo, fornite le seguenti sostituzioni di configurazione. Facoltativamente, puoi anche fornire il nome del gruppo di log, il nome del prefisso del flusso di log, i tipi di registro e una chiave di crittografia. ARN

Se non specifichi i valori opzionali, CloudWatch pubblica i log in un gruppo di log predefinito/`aws/emr-serverless`, con il flusso di log predefinito. `/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`*

EMRLe versioni serverless 7.1.0 e successive supportano i tentativi di ripetizione per processi di streaming e processi in batch. Se hai abilitato i nuovi tentativi per un processo, EMR Serverless aggiunge automaticamente un numero di tentativo al prefisso del percorso di registro, in modo da poter distinguere e tracciare meglio i log.

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

Di seguito viene mostrata la configurazione minima richiesta per attivare Amazon CloudWatch Logging con le impostazioni predefinite per EMR Serverless:

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

L'esempio seguente mostra tutte le configurazioni obbligatorie e opzionali che puoi specificare quando attivi Amazon CloudWatch Logging for EMR Serverless. I `logTypes` valori supportati sono elencati anche sotto questo esempio.

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

Per impostazione predefinita, EMR Serverless pubblica solo i log dei driver su cui stdout e stderr. CloudWatch Se desideri altri log, puoi specificare un ruolo del contenitore e i tipi di log corrispondenti con il campo. `logTypes`

L'elenco seguente mostra i tipi di worker supportati che è possibile specificare per la `logTypes` configurazione:

Spark

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

Hive

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

Registri rotanti

Amazon EMR Serverless può ruotare i log delle applicazioni Spark e i log degli eventi. La rotazione dei log aiuta a risolvere il problema dei processi di lunga durata che generano file di registro di grandi dimensioni che possono occupare tutto lo spazio su disco. La rotazione dei log consente di risparmiare spazio su disco e riduce il numero di errori dei processi, poiché non rimane più spazio sul disco.

La rotazione dei log è abilitata per impostazione predefinita ed è disponibile solo per i job Spark.

Registri degli eventi di Spark

> **Note**
>
> La rotazione dei registri degli eventi Spark è disponibile su tutte le etichette di EMR rilascio di Amazon.

Invece di generare un singolo file di registro degli eventi, EMR Serverless ruota il registro degli eventi a intervalli di tempo regolari e rimuove i file di registro degli eventi più vecchi. La rotazione dei log non influisce sui log caricati nel bucket S3.

Registri delle applicazioni Spark

> **Note**
>
> La rotazione dei log delle applicazioni Spark è disponibile su tutte le etichette di EMR rilascio di Amazon.

EMRServerless ruota anche i log delle applicazioni Spark per driver ed executor, ad esempio file e. `stdout` `stderr` Puoi accedere ai file di registro più recenti scegliendo i link di registro in Studio utilizzando i collegamenti Spark History Server e Live UI. I file di registro sono le versioni troncate dei log più recenti. Per visualizzare i log ruotati precedenti, è necessario specificare una posizione

Amazon S3 durante l'archiviazione dei log. Per ulteriori informazioni, consulta Logging for EMR Serverless con bucket Amazon S3.

Puoi trovare i file di log più recenti nella seguente posizione. EMRServerless aggiorna i file ogni 15 secondi. Questi file possono variare da 0 MB a 128 MB.

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

La seguente posizione contiene i file ruotati più vecchi. Ogni file è di 128 MB.

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

Lo stesso comportamento si applica anche agli esecutori Spark. Questa modifica è applicabile solo alla registrazione di S3. La rotazione dei log non introduce alcuna modifica ai flussi di log caricati su Amazon CloudWatch.

EMRLe versioni serverless 7.1.0 e successive supportano i tentativi di ripetizione per lo streaming e i processi in batch. Se nel processo sono stati abilitati i tentativi di riprovare, EMR Serverless aggiunge un prefisso al percorso di registro per tali processi in modo da poter tracciare e distinguere meglio i log l'uno dall'altro. Questo percorso contiene tutti i log ruotati.

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

Crittografia dei log

Crittografia dei log EMR Serverless con storage gestito

Per crittografare i log nello storage gestito con la tua KMS chiave, usa la `managedPersistenceMonitoringConfiguration` configurazione quando invii un job run.

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

Crittografia dei log EMR serverless con bucket Amazon S3

Per crittografare i log nel tuo bucket Amazon S3 con la KMS tua chiave, usa la configurazione quando `s3MonitoringConfiguration` invii un job run.

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

Crittografia dei log EMR Serverless con Amazon CloudWatch

Per crittografare i log in Amazon CloudWatch con la tua KMS chiave, usa la `cloudWatchLoggingConfiguration` configurazione quando invii un job run.

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

Autorizzazioni richieste per la crittografia dei log

In questa sezione

- Autorizzazioni utente richieste
- Autorizzazioni delle chiavi di crittografia per Amazon S3 e storage gestito
- Autorizzazioni per le chiavi di crittografia per Amazon CloudWatch

Autorizzazioni utente richieste

L'utente che invia il lavoro o visualizza i log dell'applicazione UIs deve disporre delle autorizzazioni per utilizzare la chiave. È possibile specificare le autorizzazioni nella politica KMS chiave o nella IAM

politica per l'utente, il gruppo o il ruolo. Se l'utente che invia il job non dispone delle autorizzazioni KMS chiave, EMR Serverless rifiuta l'invio dell'esecuzione del job.

Esempio di politica chiave

La seguente politica chiave fornisce le autorizzazioni per `kms:GenerateDataKey` e `kms:Decrypt`:

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

Politica di esempio IAM

La seguente IAM politica fornisce le autorizzazioni per `kms:GenerateDataKey` e `kms:Decrypt`:

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

Per avviare l'interfaccia utente di Spark o Tez, devi concedere ai tuoi utenti, gruppi o ruoli le autorizzazioni per accedere a quanto segue: `emr-serverless:GetDashboardForJobRun` API

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

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

Autorizzazioni delle chiavi di crittografia per Amazon S3 e storage gestito

Quando crittografi i log con la tua chiave di crittografia nello storage gestito o nei bucket S3, devi configurare le autorizzazioni delle chiavi come segue. KMS

Il `emr-serverless.amazonaws.com` principale deve disporre delle seguenti autorizzazioni nella politica per la chiave: KMS

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

Come procedura consigliata in materia di sicurezza, si consiglia di aggiungere una chiave di `aws:SourceArn` condizione alla policy KMS chiave. La chiave di condizione IAM globale `aws:SourceArn` aiuta a garantire che EMR Serverless utilizzi la KMS chiave solo per un'applicazioneARN.

Il ruolo di job runtime deve disporre delle seguenti autorizzazioni nella sua IAM politica:

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

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

Autorizzazioni per le chiavi di crittografia per Amazon CloudWatch

Per associare la KMS chiave ARN al tuo gruppo di log, utilizza la seguente IAM politica per il ruolo Job Runtime.

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

Configura la politica KMS chiave per concedere KMS le autorizzazioni ad Amazon CloudWatch:

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

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

Configurazione delle proprietà di Apache Log4j2 per Amazon Serverless EMR

Questa pagina descrive come configurare le proprietà personalizzate di Apache Log4j 2.x per i lavori Serverless su. EMR `StartJobRun` Se desideri configurare le classificazioni Log4j a livello di applicazione, consulta. Configurazione predefinita dell'applicazione per Serverless EMR

Configurazione delle proprietà di Spark Log4j2 per Amazon Serverless EMR

Con le EMR versioni 6.8.0 e successive di Amazon, puoi personalizzare le proprietà di Apache Log4j 2.x per specificare configurazioni di log dettagliate. Questo semplifica la risoluzione dei problemi dei job Spark su Serverless. EMR Per configurare queste proprietà, usa le classificazioni `espark-driver-log4j2`. `spark-executor-log4j2`

Argomenti

- Classificazioni Log4j2 per Spark
- Esempio di configurazione Log4j2 per Spark
- Log4j2 negli esempi di job Spark
- Considerazioni su Log4j2 per Spark

Classificazioni Log4j2 per Spark

Per personalizzare le configurazioni dei log di Spark, usa le seguenti classificazioni con. `applicationConfiguration` Per configurare le proprietà di Log4j 2.x, usa quanto segue. `properties`

`spark-driver-log4j2`

Questa classificazione imposta i valori nel `log4j2.properties` file per il driver.

`spark-executor-log4j2`

Questa classificazione imposta i valori nel `log4j2.properties` file per l'esecutore.

Esempio di configurazione Log4j2 per Spark

L'esempio seguente mostra come inviare un job Spark con per personalizzare le configurazioni di Log4j2 `applicationConfiguration` per il driver e l'esecutore Spark.

Per configurare le classificazioni Log4j a livello di applicazione anziché al momento dell'invio del lavoro, consulta. Configurazione predefinita dell'applicazione per Serverless EMR

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

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

```
    ]
  }'
```

Log4j2 negli esempi di job Spark

I seguenti esempi di codice mostrano come creare un'applicazione Spark mentre inizializzi una configurazione Log4j2 personalizzata per l'applicazione.

Python

Example - Utilizzo di Log4j2 per un lavoro Spark con Python

```
import os
import sys

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

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

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

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

    // your code here

    spark.stop()
```

Per personalizzare Log4j2 per il driver quando esegui un job Spark, puoi usare la seguente configurazione:

```
{
```

```
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.PySparkApp.level": "info",
          "logger.PySparkApp.name": "PySparkApp"
      }
}
```

Scala

Example - Usare Log4j2 per un job Spark con Scala

```
import org.apache.log4j.Logger
import org.apache.spark.sql.SparkSession

object ExampleClass {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession
    .builder
    .appName(this.getClass.getName)
    .getOrCreate()

    val logger = Logger.getLogger(this.getClass);
    logger.info("script logging info logs")
    logger.warn("script logging warn logs")
    logger.error("script logging error logs")

// your code here
    spark.stop()
  }
}
```

Per personalizzare Log4j2 per il driver quando esegui un job Spark, puoi usare la seguente configurazione:

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.ExampleClass.level": "info",
          "logger.ExampleClass.name": "ExampleClass"
      }
```

```
}
```

Considerazioni su Log4j2 per Spark

Le seguenti proprietà Log4j2.x non sono configurabili per i processi Spark:

- `rootLogger.appenderRef.stdout.ref`
- `appender.console.type`
- `appender.console.name`
- `appender.console.target`
- `appender.console.layout.type`
- `appender.console.layout.pattern`

Per informazioni dettagliate sulle proprietà di Log4j2.x che puoi configurare, consulta il file su. `log4j2.properties.template` GitHub

Monitoraggio senza server EMR

Questa sezione illustra i modi in cui puoi monitorare le tue applicazioni e i tuoi lavori Amazon EMR Serverless.

Argomenti

- Monitoraggio di applicazioni e EMR lavori serverless
- Monitora i parametri di Spark con Amazon Managed Service for Prometheus
- EMRParametri di utilizzo serverless

Monitoraggio di applicazioni e EMR lavori serverless

Con Amazon CloudWatch Metrics for EMR Serverless, puoi ricevere parametri in 1 minuto CloudWatch e accedere a CloudWatch dashboard per visualizzare near-real-time le operazioni e le prestazioni delle tue applicazioni Serverless. EMR

EMRServerless invia metriche a ogni minuto. CloudWatch EMRServerless emette queste metriche a livello di applicazione, nonché a livello di mansione, tipo di lavoratore e livelli. capacity-allocation-type

Per iniziare, utilizza il modello di CloudWatch dashboard EMR Serverless fornito nel repository Serverless e distribuiscilo. EMR GitHub

> **Note**
>
> EMRNei carichi di lavoro interattivi serverless è abilitato solo il monitoraggio a livello di applicazione e hanno una nuova dimensione di tipo di lavoratore,. `Spark_Kernel` Per monitorare ed eseguire il debug dei carichi di lavoro interattivi, puoi visualizzare i log e l'interfaccia utente di Apache Spark dall'interno di Studio Workspace. EMR

La tabella seguente descrive le dimensioni EMR Serverless disponibili all'interno dello spazio dei nomi. `AWS/EMRServerless`

Dimensioni per metriche Serverless EMR

Dimensione	Descrizione
`ApplicationId`	Filtri per tutte le metriche di un'EMRapplicazione serverless.
`JobId`	Filtri per tutte le metriche dell'esecuzione di un processo EMR Serverless.
`WorkerType`	Filtri per tutte le metriche di un determinato tipo di lavoratore. Ad esempio, puoi filtrare per `SPARK_DRIVER` e `SPARK_EXECUTORS` per i job Spark.
`CapacityAllocationType`	Filtri per tutte le metriche di un determinato tipo di allocazione della capacità. Ad esempio, puoi filtrare per la capacità preinizializzata e `PreInitCapacity`

Dimensione	Descrizione	
	`OnDemandCapacity` per tutto il resto.	

Monitoraggio a livello di applicazione

Puoi monitorare l'utilizzo della capacità a livello di applicazione EMR Serverless con i CloudWatch parametri di Amazon. Puoi anche configurare una vista singola per monitorare l'utilizzo della capacità delle applicazioni in una CloudWatch dashboard.

EMRMetriche delle applicazioni serverless

Parametro	Descrizione	Dimensione principale	Dimensione secondaria
`CPUAllocated`	Il numero totale di vCPUs risorse allocate.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`IdleWorkerCount`	Il numero totale di lavoratori inattivi.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`MaxCPUAllowed`	Il numero massimo CPU consentito per l'applicazione.	`ApplicationId`	N/D
`MaxMemoryAllowed`	La memoria massima in GB consentita per l'applicazione.	`ApplicationId`	N/D
`MaxStorageAllowed`	La capacità massima di archiviazione in GB consentita per l'applicazione.	`ApplicationId`	N/D

Parametro	Descrizione	Dimensione principale	Dimensione secondaria
`MemoryAllocated`	Memoria totale in GB allocata.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`PendingCreationWorkerCount`	Il numero totale di lavoratori in attesa di creazione.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`RunningWorkerCount`	Il numero totale di lavoratori utilizzati dall'applicazione.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`StorageAllocated`	Lo spazio di archiviazione totale su disco in GB allocato.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`TotalWorkerCount`	Il numero totale di lavoratori disponibili.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`

Monitoraggio a livello di mansione

Amazon EMR Serverless invia i seguenti parametri a livello di processo a Amazon CloudWatch ogni minuto. È possibile visualizzare i valori delle metriche per le esecuzioni aggregate dei processi in base allo stato di esecuzione del processo. L'unità per ciascuna metrica è il conteggio.

EMRMetriche a livello di job serverless

Parametro	Descrizione	Dimensione principale
`SubmittedJobs`	Il numero di lavori in uno stato Inviato.	`ApplicationId`
`PendingJobs`	Il numero di lavori in uno stato In sospeso.	`ApplicationId`
`ScheduledJobs`	Il numero di lavori in uno stato Pianificato.	`ApplicationId`
`RunningJobs`	Il numero di lavori in uno stato In esecuzione.	`ApplicationId`
`SuccessJobs`	Il numero di lavori in uno stato di successo.	`ApplicationId`
`FailedJobs`	Il numero di lavori in uno stato Non riuscito.	`ApplicationId`
`CancellingJobs`	Il numero di lavori in uno stato di annullamento.	`ApplicationId`
`CancelledJobs`	Il numero di lavori in uno stato annullato.	`ApplicationId`

È possibile monitorare le metriche specifiche del motore sia per i job EMR Serverless in esecuzione che per quelli completati con un'applicazione specifica del motore. UIs Quando si visualizza l'interfaccia utente di un job in esecuzione, viene visualizzata l'interfaccia utente live dell'applicazione con aggiornamenti in tempo reale. Quando si visualizza l'interfaccia utente di un lavoro completato, viene visualizzata l'interfaccia utente persistente dell'app.

Esecuzione di processi

Per eseguire lavori EMR Serverless, puoi visualizzare un'interfaccia in tempo reale che fornisce metriche specifiche del motore. Puoi utilizzare l'interfaccia utente di Apache Spark o l'interfaccia utente di Hive Tez per monitorare ed eseguire il debug dei tuoi lavori. Per accederviUIs, usa la console EMR Studio o richiedi un endpoint sicuro con URL AWS Command Line Interface.

Lavori completati

Per i lavori EMR Serverless completati, puoi utilizzare lo Spark History Server o l'interfaccia utente Persistent Hive Tez per visualizzare i dettagli dei lavori, le fasi, le attività e le metriche relative all'esecuzione dei job Spark o Hive. Per accederviUIs, usa la console EMR Studio o richiedi un endpoint sicuro con URL AWS Command Line Interface.

Monitoraggio a livello di Job Worker

Amazon EMR Serverless invia ad Amazon i seguenti parametri a livello di job worker disponibili nel `AWS/EMRServerless` namespace e nel gruppo di `Job Worker Metrics` metrici. CloudWatch EMRServerless raccoglie punti dati dai singoli lavoratori durante le esecuzioni dei lavori a livello di mansione, tipo di lavoratore e livello. capacity-allocation-type È possibile utilizzarlo `ApplicationId` come dimensione per monitorare più lavori che appartengono alla stessa applicazione.

EMRMetriche serverless a livello di job worker

Parametro	Descrizione	Unità	Dimensione principale	Dimensione secondaria
`WorkerCpuAllocated`	Il numero totale di v CPU core assegnati ai lavoratori in un determinato periodo di lavoro.	Nessuno	`JobId`	`ApplicationId` , `WorkerType` e `CapacityAllocationType`
`WorkerCpuUsed`	Il numero totale di v CPU core utilizzati dai lavoratori in un ciclo di lavoro.	Nessuno	`JobId`	`ApplicationId` , `WorkerType` e `CapacityAllocationType`
`WorkerMemoryAllocated`	Memoria totale in GB allocata per i lavoratori durante l'esecuzi	Gigabyte (GB)	`JobId`	`ApplicationId` , `WorkerType` e `CapacityA`

Parametro	Descrizione	Unità	Dimensione principale	Dimensione secondaria
	one di un processo.			llocation Type
WorkerMemoryUsed	Memoria totale in GB utilizzata dai lavoratori in un processo.	Gigabyte (GB)	JobId	ApplicationId , WorkerType e CapacityAllocation Type
WorkerEphemeralStorageAllocated	Il numero di byte di storage temporaneo allocato ai lavoratori durante l'esecuzione di un job.	Gigabyte (GB)	JobId	ApplicationId , WorkerType e CapacityAllocation Type
WorkerEphemeralStorageUsed	Il numero di byte di storage temporaneo utilizzati dai lavoratori durante l'esecuzione di un processo.	Gigabyte (GB)	JobId	ApplicationId , WorkerType e CapacityAllocation Type
WorkerStorageReadBytes	Il numero di byte letti dallo storage dai lavoratori durante l'esecuzione di un job.	Byte	JobId	ApplicationId , WorkerType e CapacityAllocation Type

Parametro	Descrizione	Unità	Dimensione principale	Dimensione secondaria
`WorkerStorageWriteBytes`	Il numero di byte scritti nello storage dai lavoratori durante l'esecuzione di un job.	Byte	`JobId`	`ApplicationId` , `WorkerType` e `CapacityAllocationType`

I passaggi seguenti descrivono come visualizzare i vari tipi di metriche.

Console

Per accedere all'interfaccia utente dell'applicazione con la console

1. Passa all'applicazione EMR Serverless su EMR Studio con le istruzioni in Guida introduttiva dalla console.
2. Per visualizzare l'applicazione UIs e i registri specifici del motore per un processo in esecuzione:
   a. Scegli un lavoro con uno stato. `RUNNING`
   b. Seleziona il lavoro nella pagina dei dettagli della candidatura o vai alla pagina dei dettagli del lavoro relativa al tuo lavoro.
   c. Nel menu a discesa Display UI, scegli Spark UI o Hive Tez UI per accedere all'interfaccia utente dell'applicazione per il tuo tipo di lavoro.
   d. Per visualizzare i log del motore Spark, vai alla scheda Executors nell'interfaccia utente Spark e scegli il link Logs per il driver. Per visualizzare i log del motore Hive, scegli il link Logs corrispondente nell'interfaccia utente di Hive Tez. DAG
3. Per visualizzare l'applicazione e i registri specifici del motore per un lavoro completato: UIs
   a. Scegli un lavoro con uno stato. `SUCCESS`
   b. Seleziona il lavoro nella pagina dei dettagli della candidatura o vai alla pagina dei dettagli del lavoro.
   c. Nel menu a discesa Display UI, scegli Spark History Server o Persistent Hive Tez UI per accedere all'interfaccia utente dell'applicazione per il tuo tipo di lavoro.

d. Per visualizzare i log del motore Spark, vai alla scheda Executors nell'interfaccia utente Spark e scegli il link Logs per il driver. Per visualizzare i log del motore Hive, scegli il link Logs corrispondente nell'interfaccia utente di Hive Tez. DAG

AWS CLI

Per accedere all'interfaccia utente dell'applicazione con il AWS CLI

- Per generare un'URLinterfaccia da utilizzare per accedere all'interfaccia utente dell'applicazione sia per i lavori in esecuzione che per quelli completati, chiama il `GetDashboardForJobRun`API.

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

URLQuella generata è valida per un'ora.

Monitora i parametri di Spark con Amazon Managed Service for Prometheus

Con le EMR versioni 7.1.0 e successive di Amazon, puoi integrare EMR Serverless con Amazon Managed Service for Prometheus per raccogliere i parametri di Apache Spark per lavori e applicazioni Serverless. EMR Questa integrazione è disponibile quando invii un lavoro o crei un'applicazione utilizzando uno dei AWS console, EMR Serverless API o AWS CLI.

Prerequisiti

Prima di poter fornire i parametri Spark ad Amazon Managed Service for Prometheus, devi completare i seguenti prerequisiti.

- Crea un'area di lavoro Amazon Managed Service per Prometheus. Questo Workspace funge da endpoint di acquisizione. Prendi nota di quanto URL visualizzato per Endpoint: scrittura remota. URL Dovrai specificare URL quando crei l'applicazione EMR Serverless.
- Per concedere l'accesso dei tuoi lavori ad Amazon Managed Service for Prometheus a scopo di monitoraggio, aggiungi la seguente politica al tuo ruolo di esecuzione del lavoro.

```
{
    "Sid": "AccessToPrometheus",
    "Effect": "Allow",
    "Action": ["aps:RemoteWrite"],
    "Resource": "arn:aws:aps:<AWS_REGION>:<AWS_ACCOUNT_ID>:workspace/<WORKSPACE_ID>"
}
```

Installazione

Per utilizzare nuovamente il plugin AWS console per creare un'applicazione integrata con Amazon Managed Service for Prometheus

1. Vedi Guida introduttiva ad Amazon EMR Serverless per creare un'applicazione.
2. Durante la creazione di un'applicazione, scegli Usa impostazioni personalizzate, quindi configura l'applicazione specificando le informazioni nei campi che desideri configurare.
3. In Application logs and metrics, scegli Deliver engine metrics to Amazon Managed Service for Prometheus, quindi specifica la scrittura remota. URL
4. Specificate le altre impostazioni di configurazione desiderate, quindi scegliete Crea e avvia l'applicazione.

Utilizzate il AWS CLI o EMR Serverless API

Puoi anche usare il AWS CLI o EMR Serverless API per integrare la tua applicazione EMR Serverless con Amazon Managed Service for Prometheus quando esegui i comandi o. `create-application` `start-job-run`

create-application

```
aws emr-serverless create-application \
--release-label emr-7.1.0 \
--type "SPARK" \
--monitoring-configuration '{
    "prometheusMonitoringConfiguration": {
        "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
    }
}'
```

start-job-run

```
aws emr-serverless start-job-run \
--application-id <APPPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
        "entryPointArguments": ["10000"],
        "sparkSubmitParameters": "--conf spark.dynamicAllocation.maxExecutors=10"
    }
}' \
--configuration-overrides '{
     "monitoringConfiguration": {
        "prometheusMonitoringConfiguration": {
            "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
        }
    }
}'
```

L'inclusione `prometheusMonitoringConfiguration` nel comando indica che EMR Serverless deve eseguire il job Spark con un agente che raccoglie le metriche Spark e le scrive sul tuo endpoint `remoteWriteUrl` per Amazon Managed Service for Prometheus. Puoi quindi utilizzare le metriche Spark in Amazon Managed Service for Prometheus per la visualizzazione, gli avvisi e l'analisi.

Proprietà di configurazione avanzate

EMRServerless utilizza un componente all'interno di Spark denominato `PrometheusServlet` per raccogliere i parametri Spark e traduce i dati sulle prestazioni in dati compatibili con Amazon Managed Service for Prometheus. Per impostazione predefinita, EMR Serverless imposta i valori predefiniti in Spark e analizza le metriche del driver e dell'esecutore quando invii un lavoro utilizzando. `PrometheusMonitoringConfiguration`

La tabella seguente descrive tutte le proprietà che puoi configurare quando invii un job Spark che invia metriche ad Amazon Managed Service for Prometheus.

Proprietà Spark	Valore predefinito	Descrizione
`spark.metrics.conf.*.sink.prometheusServlet.class`	org.apache.spark.metrics.sink.PrometheusServlet	La classe utilizzata da Spark per inviare metriche ad Amazon Managed Service for Prometheus. Per ignorare il comportamento predefinito, specifica la tua classe personalizzata.
`spark.metrics.conf.*.source.jvm.class`	org.apache.spark.metrics.source. JvmSource	La classe utilizzata da Spark per raccogliere e inviare metriche cruciali dalla macchina virtuale Java sottostante. Per interrompere la raccolta JVM delle metriche, disabilita questa proprietà impostandola su una stringa vuota, ad esempio. "" Per ignorare il comportamento predefinito, specificate la vostra classe personalizzata.
`spark.metrics.conf.driver.sink.prometheusServlet.path`	/metrics/prometheus	La distinzione URL che Amazon Managed Service for Prometheus utilizza per raccogliere le metriche dal driver. Per ignorare il comportamento predefinito, specifica il tuo percorso. Per interrompere la raccolta delle metriche dei driver, disattiva te questa proprietà impostand ola su una stringa vuota, ad esempio. ""

Proprietà Spark	Valore predefinito	Descrizione
`spark.metrics.conf.executor.sink.prometheusServlet.path`	/metrics/executor/prometheus	La distinzione URL che Amazon Managed Service for Prometheus utilizza per raccogliere i parametri dall'esecutore. Per ignorare il comportamento predefinito, specifica il tuo percorso. Per interrompere la raccolta delle metriche degli esecutori, disabilitate questa proprietà impostandola su una stringa vuota, ad esempio. ""

Per ulteriori informazioni sulle metriche Spark, consulta Metriche di Apache Spark.

Considerazioni e limitazioni

Quando utilizzi Amazon Managed Service for Prometheus per raccogliere metriche EMR da Serverless, considera le seguenti considerazioni e limitazioni.

- Il supporto per l'utilizzo di Amazon Managed Service for Prometheus EMR con Serverless è disponibile solo in Regioni AWS dove Amazon Managed Service for Prometheus è generalmente disponibile.
- Far sì che l'agente raccolga i parametri Spark su Amazon Managed Service for Prometheus richiede più risorse da parte dei lavoratori. Se scegli un lavoratore di dimensioni inferiori, ad esempio uno v CPU worker, il tempo di esecuzione del lavoro potrebbe aumentare.
- Il supporto per l'utilizzo di Amazon Managed Service for Prometheus EMR con Serverless è disponibile solo per EMR le versioni di Amazon 7.1.0 e successive.

EMRParametri di utilizzo serverless

Puoi utilizzare i parametri di CloudWatch utilizzo di Amazon per fornire visibilità sulle risorse utilizzate dal tuo account. Utilizza queste metriche per visualizzare l'utilizzo del servizio su CloudWatch grafici e dashboard.

EMRLe metriche di utilizzo serverless corrispondono a Service Quotas. È possibile configurare gli allarmi che avvisano quando l'uso si avvicina a una quota di servizio. Per ulteriori informazioni, consulta Service Quotas e Amazon CloudWatch alarms nella Service Quotas User Guide.

Per ulteriori informazioni sulle quote di servizio EMR Serverless, consulta. Endpoint e quote per EMR Serverless

Metriche di utilizzo delle quote di servizio per Serverless EMR

EMRServerless pubblica le seguenti metriche di utilizzo delle quote di servizio nel namespace. `AWS/Usage`

Parametro	Descrizione
`ResourceCount`	Il numero totale della risorsa specificata in esecuzione sul tuo account. La risorsa è definita dalle dimensioni associate alla metrica.

Dimensioni per le metriche di utilizzo delle quote di servizio EMR Serverless

Puoi utilizzare le seguenti dimensioni per perfezionare le metriche di utilizzo pubblicate da Serverless. EMR

Dimensione	Valore	Descrizione
`Service`	EMRServerless	Il nome del Servizio AWS che contiene la risorsa.
`Type`	Risorsa	Il tipo di entità segnalata da EMR Serverless.
`Resource`	v CPU	Il tipo di risorsa monitorata da EMR Serverless.
`Class`	Nessuno	La classe di risorse monitorata da EMR Serverless.

Automazione Serverless con EMR Amazon EventBridge

È possibile utilizzare… Amazon EventBridge per automatizzare il tuo Servizi AWS e rispondono automaticamente agli eventi di sistema, come problemi di disponibilità delle applicazioni o modifiche delle risorse. EventBridge fornisce un flusso quasi in tempo reale di eventi di sistema che descrivono i cambiamenti avvenuti AWS risorse. Puoi compilare regole semplici che indichino quali eventi sono considerati di interesse per te e quali azioni automatizzate intraprendere quando un evento corrisponde a una regola. Con EventBridge, puoi automaticamente:

- Invocare un AWS Lambda funzione
- Inoltrare un evento ad Amazon Kinesis Data Streams
- Attiva un AWS Step Functions macchina a stati
- Notifica un SNS argomento Amazon o una SQS coda Amazon

Ad esempio, quando utilizzi EventBridge EMR Serverless, puoi attivare un AWS Lambda funziona quando un ETL lavoro ha esito positivo o notifica un SNS argomento di Amazon quando un ETL lavoro fallisce.

EMRServerless emette tre tipi di eventi:

- Eventi di modifica dello stato dell'applicazione: eventi che generano ogni modifica di stato di un'applicazione. Per ulteriori informazioni sugli stati delle applicazioni, vedereStati dell'applicazione.
- Eventi di modifica dello stato del job run: eventi che generano ogni modifica di stato di un job run. Per ulteriori informazioni su, vedereStati delle esecuzioni di processi.
- Eventi di ripetizione del processo: eventi che generano ogni nuovo tentativo di un processo eseguito dalle versioni 7.1.0 e successive di Amazon EMR Serverless.

EventBridgeEsempi di eventi Serverless EMR

Agli eventi segnalati da EMR Serverless è `aws.emr-serverless` assegnato un valore`source`, come illustrato negli esempi seguenti.

Evento di modifica dello stato dell'applicazione

L'evento di esempio seguente mostra un'applicazione nello `CREATING` stato.

```
{
```

```
    "version": "0",
    "id": "9fd3cf79-1ff1-b633-4dd9-34508dc1e660",
    "detail-type": "EMR Serverless Application State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:16:31Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "applicationId": "00f1cbsc6anuij25",
        "applicationName": "3965ad00-8fba-4932-a6c8-ded32786fd42",
        "arn": "arn:aws:emr-serverless:us-east-1:111122223333:/
applications/00f1cbsc6anuij25",
        "releaseLabel": "emr-6.6.0",
        "state": "CREATING",
        "type": "HIVE",
        "createdAt": "2022-05-31T21:16:31.547953Z",
        "updatedAt": "2022-05-31T21:16:31.547970Z",
        "autoStopConfig": {
            "enabled": true,
            "idleTimeout": 15
        },
        "autoStartConfig": {
            "enabled": true
        }
    }
}
```

Evento di modifica dello stato di Job run

L'evento di esempio seguente mostra l'esecuzione di un processo che passa da `SCHEDULED` uno `RUNNING` stato all'altro.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
```

```
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "state": "RUNNING",
        "previousState": "SCHEDULED",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
        "createdAt": "2022-05-31T21:07:25.325900Z"
    }
}
```

Evento Job run Retry

Di seguito è riportato un esempio di evento Job Run Retry.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run Retry",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
        "createdAt": "2022-05-31T21:07:25.325900Z",
        //Attempt Details
        "previousAttempt": 1,
        "previousAttemptState": "FAILED",
        "previousAttemptCreatedAt": "2022-05-31T21:07:25.325900Z",
        "previousAttemptEndedAt": "2022-05-31T21:07:30.325900Z",
        "newAttempt": 2,
        "newAttemptCreatedAt": "2022-05-31T21:07:30.325900Z"
```

```
    }
}
```

Applicazione di tag alle risorse

Puoi assegnare i tuoi metadati a ciascuna risorsa utilizzando tag per aiutarti a gestire le tue EMR risorse Serverless. Questa sezione fornisce una panoramica delle funzioni dei tag e mostra come creare tag.

Argomenti

Che cos'è un tag?

Un tag è un'etichetta che si assegna a un AWS risorsa. Ogni tag consiste di una chiave e di un valore, entrambi personalizzabili. I tag ti consentono di classificare i tuoi AWS risorse per attributi come scopo, proprietario e 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 le tue applicazioni Amazon EMR Serverless per aiutarti a monitorare il proprietario e il livello di stack di ciascuna applicazione. Consigliamo di definire un set coerente di chiavi di tag per ciascun tipo di risorsa.

I tag non vengono assegnati in automatico alle risorse. Dopo aver aggiunto un tag a una risorsa, puoi modificare il valore di un tag o rimuoverlo dalla risorsa in qualsiasi momento. I tag non hanno alcun significato semantico per Amazon EMR Serverless e vengono interpretati rigorosamente come stringhe di caratteri. Se aggiungi un tag con la stessa chiave di un tag esistente a una risorsa specifica, il nuovo valore sovrascrive quello precedente.

Se lo utilizziIAM, puoi controllare quali utenti del tuo AWS l'account è autorizzato 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.

Tagging delle risorse

È possibile contrassegnare applicazioni ed esecuzioni di lavori nuove o esistenti. Se utilizzi Amazon EMR ServerlessAPI, AWS CLI, o un AWS SDK, è possibile applicare tag a nuove risorse utilizzando

il `tags` parametro sull'APIazione pertinente. È possibile applicare tag a risorse esistenti utilizzando l'`TagResource`APIazione.

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 taggate le risorse al momento della creazione, non è necessario eseguire script di etichettatura personalizzati dopo aver creato una risorsa.

La tabella seguente descrive le risorse Amazon EMR Serverless che possono essere taggate.

Risorsa	Supporta tag	Supporta la propagazione di tag	Supporta l'etichettatura alla creazione (Amazon EMR ServerlessAPI, AWS CLI e AWS SDK)	APIper la creazione (i tag possono essere aggiunti durante la creazione)
Applicazione	Sì	No. I tag associati a un'applic azione non si propagano alle esecuzioni di job inviate a tale applicazione.	Sì	`CreateApp lication`
Esecuzione del processo	Sì	No	Sì	`StartJobRun`

Limitazioni relative all'etichettatura

Le seguenti limitazioni di base si applicano ai tag:

- Ogni risorsa può avere un massimo di 50 tag creati dall'utente.

- Per ciascuna risorsa, ogni chiave del tag deve essere univoca e ogni chiave del tag può avere un solo valore.
- La lunghezza massima della chiave è di 128 caratteri Unicode in UTF -8.
- La lunghezza massima del valore è di 256 caratteri Unicode in UTF -8.
- I caratteri consentiti sono lettere, numeri, spazi rappresentabili in UTF -8 e i seguenti caratteri: _.:/= + - @.
- Una chiave di tag non può essere una stringa vuota. Un valore di tag può essere una stringa vuota, ma non nullo.
- I valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole.
- Non utilizzare `AWS:` alcuna combinazione maiuscola o minuscola, ad esempio un prefisso per chiavi o valori. Questi sono riservati solo a AWS uso.

Lavorare con i tag utilizzando il AWS CLI e Amazon EMR Serverless API

Usa quanto segue AWS CLI comandi o API operazioni Amazon EMR Serverless per aggiungere, aggiornare, elencare ed eliminare i tag per le tue risorse.

Risorsa	Supporta tag	Supporta la propagazione di tag
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 etichettare o rimuovere i tag dalle risorse utilizzando AWS CLI.

Etichettare un'applicazione esistente

Il comando seguente contrassegna un'applicazione esistente.

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

Rimuovere i tag da un'applicazione esistente

Il comando seguente elimina un tag da un'applicazione esistente.

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Elenca i tag per una risorsa

Il comando seguente elenca i tag associati a una risorsa esistente.

```
aws emr-serverless list-tags-for-resource --resource-arn resource_ARN
```

Tutorial per Serverless EMR

Questa sezione descrive i casi d'uso comuni quando si lavora con applicazioni EMR Serverless.

Argomenti

Utilizzo di Java 17 con Amazon EMR Serverless

Con le EMR versioni 6.11.0 e successive di Amazon, puoi configurare i job Spark EMR Serverless per utilizzare il runtime Java 17 per la Java Virtual Machine (). JVM Usa uno dei seguenti metodi per configurare Spark con Java 17.

`JAVA_HOME`

Per sovrascrivere l'JVMimpostazione per EMR Serverless 6.11.0 e versioni successive, puoi fornire l'`JAVA_HOME`impostazione alle relative classificazioni e a quelle di ambiente. `spark.emr-serverless.driverEnv` `spark.executorEnv`

x86_64

Imposta le proprietà richieste per specificare Java 17 come `JAVA_HOME` configurazione per il driver e gli esecutori Spark:

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-
corretto.x86_64/
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64/
```

arm_64

Imposta le proprietà richieste per specificare Java 17 come JAVA_HOME configurazione per il driver e gli esecutori Spark:

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-
corretto.aarch64/
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.aarch64/
```

spark-defaults

In alternativa, puoi specificare Java 17 nella `spark-defaults` classificazione per sovrascrivere l'JVMimpostazione per EMR Serverless 6.11.0 e versioni successive.

x86_64

Specificare Java 17 nella classificazione: `spark-defaults`

```
{
"applicationConfiguration": [
         {
            "classification": "spark-defaults",
            "properties": {
               "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.x86_64/",
               "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.x86_64/"
            }
         }
      ]
}
```

arm_64

Specificare Java 17 nella `spark-defaults` classificazione:

```
{
```

```
"applicationConfiguration": [
         {
            "classification": "spark-defaults",
            "properties": {
               "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.aarch64/",
               "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.aarch64/"
            }
         }
      ]
}
```

Utilizzo di Apache Hudi con Serverless EMR

Per utilizzare Apache Hudi con applicazioni Serverless EMR

1. Imposta le proprietà Spark richieste nell'esecuzione del job Spark corrispondente.

   ```
   spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
   spark.serializer=org.apache.spark.serializer.KryoSerializer
   ```

2. Per sincronizzare una tabella Hudi con il catalogo configurato, designate uno dei seguenti AWS Glue Data Catalog come metastore o configura un metastore esterno. EMRServerless supporta la modalità di sincronizzazione per `hms` le tabelle Hive per i carichi di lavoro Hudi. EMRServerless attiva questa proprietà come impostazione predefinita. Per ulteriori informazioni su come configurare il metastore, consulta. Configurazione Metastore

   > **Important**
   >
   > EMRServerless non supporta `HIVEQL` né `JDBC` fornisce opzioni di modalità di sincronizzazione per le tabelle Hive per gestire i carichi di lavoro Hudi. Per ulteriori informazioni, consulta Modalità di sincronizzazione.

   Quando usi il AWS Glue Data Catalog come metastore, puoi specificare le seguenti proprietà di configurazione per il tuo lavoro Hudi.

   ```
   --conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,
   ```

```
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer,
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

Per ulteriori informazioni sulle versioni di Apache Hudi di AmazonEMR, consulta la cronologia delle versioni di Hudi.

Utilizzo di Apache Iceberg con Serverless EMR

Per utilizzare Apache Iceberg con applicazioni Serverless EMR

1. Imposta le proprietà Spark richieste nell'esecuzione del job Spark corrispondente.

   ```
   spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
   ```

2. Designare uno dei due AWS Glue Data Catalog come metastore o configura un metastore esterno. Per ulteriori informazioni sulla configurazione del metastore, consulta. Configurazione Metastore

   Configura le proprietà del metastore che desideri utilizzare per Iceberg. Ad esempio, se si desidera utilizzare il AWS Glue Data Catalog, imposta le seguenti proprietà nella configurazione dell'applicazione.

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

   Quando si utilizza il AWS Glue Data Catalog come metastore, puoi specificare le seguenti proprietà di configurazione per il tuo lavoro Iceberg.

   ```
   --conf spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar,
   --conf
    spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions,
   --conf spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog,
   --conf spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog,
   --conf spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
   ```

```
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

Per ulteriori informazioni sulle versioni di Apache Iceberg di AmazonEMR, consulta la cronologia delle versioni di Iceberg.

Utilizzo delle librerie Python con Serverless EMR

Quando esegui PySpark lavori su applicazioni Amazon EMR Serverless, puoi impacchettare varie librerie Python come dipendenze. Per fare ciò, puoi usare le funzionalità native di Python, creare un ambiente virtuale o configurare direttamente i tuoi PySpark lavori per utilizzare le librerie Python. Questa pagina descrive ogni approccio.

Utilizzo delle funzionalità native di Python

Quando imposti la seguente configurazione, puoi usarla PySpark per caricare file Python (`.py`), pacchetti Python compressi () e file `.egg` Egg `.zip` () negli esecutori Spark.

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

Per maggiori dettagli su come usare gli ambienti virtuali Python per i PySpark lavori, vedi Uso delle funzionalità PySpark native.

Creazione di un ambiente virtuale Python

Per impacchettare più librerie Python per un PySpark lavoro, puoi creare ambienti virtuali Python isolati.

1. Per creare l'ambiente virtuale Python, usa i seguenti comandi. L'esempio mostrato installa i pacchetti `scipy` e `matplotlib` in un pacchetto di ambiente virtuale e copia l'archivio in una posizione Amazon S3.

   > **Important**
   >
   > È necessario eseguire i seguenti comandi in un ambiente Amazon Linux 2 simile con la stessa versione di Python utilizzata in EMR Serverless, ovvero Python 3.7.10 per Amazon release 6.6.0. EMR Puoi trovare un Dockerfile di esempio nel repository Serverless Samples. EMR GitHub

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

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

# install the python packages
pip3 install scipy
pip3 install matplotlib

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

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

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

2. Invia il job Spark con le proprietà impostate per utilizzare l'ambiente virtuale Python.

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

Nota che se non sovrascrivi il binario Python originale, la seconda configurazione nella sequenza di impostazioni precedente sarà. `--conf spark.executorEnv.PYSPARK_PYTHON=python`

Per ulteriori informazioni su come utilizzare gli ambienti virtuali Python per i PySpark lavori, vedere Using Virtualenv. Per altri esempi su come inviare lavori Spark, vedi. Offerte di lavoro Spark

Configurazione dei PySpark lavori per l'utilizzo delle librerie Python

Con le EMR versioni 6.12.0 e successive di Amazon, puoi configurare direttamente i PySpark job EMR Serverless per utilizzare le librerie Python più diffuse per la scienza dei dati come panda e senza alcuna configurazione aggiuntiva. NumPyPyArrow

Gli esempi seguenti mostrano come impacchettare ogni libreria Python per un PySpark lavoro.

NumPy

NumPy è una libreria Python per il calcolo scientifico che offre array e operazioni multidimensionali per matematica, ordinamento, simulazione casuale e statistiche di base. NumPyPer utilizzarla, esegui il seguente comando:

```
import numpy
```

pandas

pandas è una libreria Python basata su. NumPy La libreria pandas fornisce ai data scientist strutture di dati e strumenti di analisi DataFramedei dati. Per usare pandas, esegui il seguente comando:

```
import pandas
```

PyArrow

PyArrow è una libreria Python che gestisce i dati colonnari in memoria per migliorare le prestazioni lavorative. PyArrow si basa sulla specifica di sviluppo multilingue di Apache Arrow, che è un modo standard per rappresentare e scambiare dati in un formato colonnare. Per utilizzarlo PyArrow, esegui il seguente comando:

```
import pyarrow
```

Utilizzo di diverse versioni di Python con Serverless EMR

Oltre allo use case inUtilizzo delle librerie Python con Serverless EMR, puoi anche utilizzare ambienti virtuali Python per lavorare con versioni di Python diverse rispetto alla versione inclusa nella EMR release Amazon per la tua applicazione Amazon Serverless. EMR Per fare ciò, devi creare un ambiente virtuale Python con la versione di Python che desideri utilizzare.

Per inviare un lavoro da un ambiente virtuale Python

1. Crea il tuo ambiente virtuale con i comandi nell'esempio seguente. Questo esempio installa Python 3.9.9 in un pacchetto di ambiente virtuale e copia l'archivio in una posizione Amazon S3.

> **Important**
>
> Se utilizzi le EMR versioni di Amazon 7.0.0 e successive, devi eseguire i comandi in un ambiente Amazon Linux 2023 simile a quello che usi per le tue applicazioni EMR Serverless.
> Se utilizzi la versione 6.15.0 o precedente, devi eseguire i seguenti comandi in un ambiente Amazon Linux 2 simile.

```
# install Python 3.9.9 and activate the venv
yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
wget https://www.python.org/ftp/python/3.9.9/Python-3.9.9.tgz && \
tar xzf Python-3.9.9.tgz && cd Python-3.9.9 && \
./configure --enable-optimizations && \
make altinstall

# create python venv with Python 3.9.9
python3.9 -m venv pyspark_venv_python_3.9.9 --copies
source pyspark_venv_python_3.9.9/bin/activate

# copy system python3 libraries to venv
cp -r /usr/local/lib/python3.9/* ./pyspark_venv_python_3.9.9/lib/python3.9/

# package venv to archive.
# **Note** that you have to supply --python-prefix option
# to make sure python starts with the path where your
# copied libraries are present.
# Copying the python binary to the "environment" directory.
pip3 install venv-pack
venv-pack -f -o pyspark_venv_python_3.9.9.tar.gz --python-prefix /home/hadoop/
environment

# stage the archive in S3
aws s3 cp pyspark_venv_python_3.9.9.tar.gz s3://<path>

# optionally, remove the virtual environment directory
```

```
rm -fr pyspark_venv_python_3.9.9
```

2. Imposta le tue proprietà per utilizzare l'ambiente virtuale Python e invia il job Spark.

```
# note that the archive suffix "environment" is the same as the directory where you
 copied the Python binary.
--conf spark.archives=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
pyspark_venv_python_3.9.9.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

Per ulteriori informazioni su come utilizzare gli ambienti virtuali Python per i PySpark lavori, vedere Using Virtualenv. Per altri esempi su come inviare lavori Spark, vedi. Offerte di lavoro Spark

Utilizzo di Delta Lake OSS con EMR Serverless

Amazon EMR versioni 6.9.0 e successive

> **Note**
>
> Amazon EMR 7.0.0 e versioni successive utilizzano Delta Lake 3.0.0, che rinomina il file in. `delta-core.jar` `delta-spark.jar` Se usi Amazon EMR 7.0.0 o versioni successive, assicurati di specificarlo `delta-spark.jar` nelle configurazioni.

Amazon EMR 6.9.0 e versioni successive includono Delta Lake, quindi non devi più impacchettare Delta Lake da solo o fornire il `--packages` flag per i tuoi lavori EMR Serverless.

1. Quando invii lavori EMR Serverless, assicurati di avere le seguenti proprietà di configurazione e di includere i seguenti parametri nel campo. `sparkSubmitParameters`

```
--conf spark.jars=/usr/share/aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/
delta-storage.jar
    --conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension
    --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
```

2. Crea un locale `delta_sample.py` per testare la creazione e la lettura di una tabella Delta.

```
# delta_sample.py
    from pyspark.sql import SparkSession

    import uuid

    url = "s3://DOC-EXAMPLE-BUCKET/delta-lake/output/%s/" % str(uuid.uuid4())
    spark = SparkSession.builder.appName("DeltaSample").getOrCreate()

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

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

3. Utilizzo di AWS CLI, carica il `delta_sample.py` file nel tuo bucket Amazon S3. Quindi usa il `start-job-run` comando per inviare un lavoro a un'applicazione EMR Serverless esistente.

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

    aws emr-serverless start-job-run \
        --application-id application-id \
        --execution-role-arn job-role-arn \
        --name emr-delta \
        --job-driver '{
            "sparkSubmit": {
                "entryPoint": "s3://DOC-EXAMPLE-BUCKET/code/delta_sample.py",
                "sparkSubmitParameters": "--conf spark.jars=/usr/share/
aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/delta-storage.jar --
conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
            }
        }'
```

Per usare le librerie Python con Delta Lake, puoi aggiungere la `delta-core` libreria impacchettandola come dipendenza o usandola come immagine personalizzata.

In alternativa, puoi usare `SparkContext.addPyFile` per aggiungere le librerie Python dal `delta-core` JAR file:

```
import glob
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
spark.sparkContext.addPyFile(glob.glob("/usr/share/aws/delta/lib/delta-core_*.jar")[0])
```

Amazon EMR versioni 6.8.0 e precedenti

Se utilizzi Amazon EMR 6.8.0 o versioni precedenti, segui questi passaggi per utilizzare Delta Lake OSS con le tue applicazioni EMR Serverless.

1. Per creare una versione open source di Delta Lake compatibile con la versione di Spark sulla tua applicazione Amazon EMR Serverless, accedi a Delta GitHub e segui le istruzioni.
2. Carica le librerie Delta Lake in un bucket Amazon S3 nel tuo Account AWS.
3. Quando invii lavori EMR Serverless nella configurazione dell'applicazione, includi JAR i file Delta Lake che ora sono nel tuo bucket.

   ```
   --conf spark.jars=s3://DOC-EXAMPLE-BUCKET/jars/delta-core_2.12-1.1.0.jar
   ```

4. Per assicurarti di poter leggere e scrivere da una tabella Delta, esegui un test di esempio PySpark.

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

       import uuid

       conf = SparkConf()
       sc = SparkContext(conf=conf)
       sqlContext = HiveContext(sc)

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

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

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

Invio di lavori EMR Serverless da Airflow

L'Amazon Provider di Apache Airflow fornisce operatori EMR Serverless. Per ulteriori informazioni sugli operatori, consulta Amazon EMR Serverless Operators nella documentazione di Apache Airflow.

Puoi usarlo `EmrServerlessCreateApplicationOperator` per creare un'applicazione Spark o Hive. Puoi anche utilizzarla `EmrServerlessStartJobOperator` per avviare uno o più lavori con la tua nuova applicazione.

Per utilizzare l'operatore con Amazon Managed Workflows for Apache Airflow (MWAA) con Airflow 2.2.2, aggiungi la riga seguente al `requirements.txt` file e aggiorna l'MWAAambiente per utilizzare il nuovo file.

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

Tieni presente che il supporto EMR Serverless è stato aggiunto alla versione 5.0.0 del provider Amazon. La release 6.0.0 è l'ultima versione compatibile con Airflow 2.2.2. È possibile utilizzare versioni successive con Airflow 2.4.3 attivo. MWAA

Il seguente esempio abbreviato mostra come creare un'applicazione, eseguire più job Spark e quindi arrestare l'applicazione. Un esempio completo è disponibile nel repository EMRServerless Samples. GitHub Per ulteriori dettagli sulla `sparkSubmit` configurazione, vedere. Offerte di lavoro Spark

```
from datetime import datetime

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

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

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

```
}

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

    application_id = create_app.output

    job1 = EmrServerlessStartJobOperator(
        task_id="start_job_1",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/
pi_fail.py",
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

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

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

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

Utilizzo di funzioni Hive definite dall'utente con Serverless EMR

Le funzioni definite dall'utente di Hive (UDFs) consentono di creare funzioni personalizzate per elaborare record o gruppi di record. In questo tutorial, utilizzerai un esempio UDF con un'applicazione Amazon EMR Serverless preesistente per eseguire un processo che genera un risultato di query. Per informazioni su come configurare un'applicazione, consulta. Guida introduttiva ad Amazon EMR Serverless

Per utilizzare un UDF con EMR Serverless

1. Vai a GitHubper un esempioUDF. Clona il repository e passa al ramo git che desideri utilizzare. Aggiorna il `pom.xml` file `maven-compiler-plugin` nel repository per avere una fonte. Aggiorna anche la configurazione della versione java di destinazione su`1.8`. Esegui `mvn package -DskipTests` per creare il JAR file che contiene il tuo campioneUDFs.
2. Dopo aver creato il JAR file, caricalo nel tuo bucket S3 con il seguente comando.

   ```
   aws s3 cp brickhouse-0.8.2-JS.jar s3://DOC-EXAMPLE-BUCKET/jars/
   ```

3. Crea un file di esempio per utilizzare una delle funzioni di esempioUDF. Salva questa query con nome `udf_example.q` e caricala nel tuo bucket S3.

   ```
   add jar s3://DOC-EXAMPLE-BUCKET/jars/brickhouse-0.8.2-JS.jar;
   CREATE TEMPORARY FUNCTION from_json AS 'brickhouse.udf.json.FromJsonUDF';
   select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
    array(cast(0 as int))));
   select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
    array(cast(0 as int))))["key1"][2];
   ```

4. Invia il seguente lavoro Hive.

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

```
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://'$BUCKET'/emr-
serverless-hive/warehouse"
    }
}' --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.driver.cores": "2",
            "hive.driver.memory": "6G"
        }
    }],
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/logs/"
        }
    }
}'
```

5. Usa il `get-job-run` comando per controllare lo stato del tuo lavoro. Attendi che lo stato cambi in`SUCCESS`.

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

6. Scarica i file di output con il seguente comando.

```
aws s3 cp --recursive s3://DOC-EXAMPLE-BUCKET/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

Il `stdout.gz` file è simile al seguente.

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

Utilizzo di immagini personalizzate con EMR Serverless

Argomenti

Usa una versione Python personalizzata

Puoi creare un'immagine personalizzata per usare una versione diversa di Python. Per usare la versione 3.10 di Python per i job Spark, ad esempio, esegui il seguente comando:

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

USER root

# install python 3
RUN yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
RUN wget https://www.python.org/ftp/python/3.10.0/Python-3.10.0.tgz && \
tar xzf Python-3.10.0.tgz && cd Python-3.10.0 && \
./configure --enable-optimizations && \
make altinstall

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

Prima di inviare il job Spark, imposta le proprietà per utilizzare l'ambiente virtuale Python, come segue.

```
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=/usr/local/bin/python3.10
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
--conf spark.executorEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
```

Usa una versione Java personalizzata

L'esempio seguente mostra come creare un'immagine personalizzata per utilizzare Java 11 per i lavori Spark.

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

USER root
```

```
# install JDK 11
RUN sudo amazon-linux-extras install java-openjdk11

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

Prima di inviare il job Spark, imposta le proprietà di Spark per utilizzare Java 11, come segue.

```
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-1.amzn2.0.1.x86_64
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-
```

Crea un'immagine di data science

L'esempio seguente mostra come includere pacchetti Python comuni per la scienza dei dati, come Pandas e. NumPy

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

USER root

# python packages
RUN pip3 install boto3 pandas numpy
RUN pip3 install -U scikit-learn==0.23.2 scipy
RUN pip3 install sk-dist
RUN pip3 install xgboost

# EMR Serverless will run the image as hadoop
USER hadoop:hadoop
```

Elaborazione di dati geospaziali con Apache Sedona

L'esempio seguente mostra come creare un'immagine per includere Apache Sedona per l'elaborazione geospaziale.

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

USER root

RUN yum install -y wget
```

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

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

Utilizzo dell'integrazione di Amazon Redshift per Apache Spark su Amazon Serverless EMR

Con la EMR versione 6.9.0 di Amazon e successive, ogni immagine di versione include un connettore tra Apache Spark e Amazon Redshift. Con questo connettore, puoi utilizzare Spark su Amazon EMR Serverless per elaborare i dati archiviati in Amazon Redshift. L'integrazione si basa sul connettore `spark-redshift` open source. Per Amazon EMR Serverless, l'integrazione Amazon Redshift per Apache Spark è inclusa come integrazione nativa.

Argomenti

- Avvio di un'applicazione Spark con l'integrazione 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 con l'integrazione Amazon Redshift per Apache Spark

Per utilizzare l'integrazione con EMR Serverless 6.9.0, devi passare le dipendenze Spark-Redshift richieste con il tuo job Spark. `--jars`Da utilizzare 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`

EMRLe versioni di Amazon 6.10.0 e successive non richiedono la `minimal-json.jar` dipendenza e installano automaticamente le altre dipendenze in ciascun cluster per impostazione predefinita. Gli esempi seguenti mostrano come avviare un'applicazione Spark con l'integrazione di Amazon Redshift per Apache Spark.

Amazon EMR 6.10.0 +

Avvia un job Spark su Amazon EMR Serverless con l'integrazione Amazon Redshift per Apache Spark EMR nella versione Serverless 6.10.0 e successive.

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

Per avviare un job Spark su Amazon EMR Serverless con l'integrazione Amazon Redshift per Apache Spark EMR nella versione Serverless 6.9.0, utilizza l'`--jars`opzione come illustrato nell'esempio seguente. Tieni presente che i percorsi elencati con l'`--jars`opzione sono i percorsi predefiniti per i file. JAR

```
--jars
    /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,
    /usr/share/aws/redshift/spark-redshift/lib/spark-redshift.jar,
    /usr/share/aws/redshift/spark-redshift/lib/spark-avro.jar,
    /usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar
```

```
spark-submit \
  --jars /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,/usr/share/aws/redshift/
spark-redshift/lib/spark-redshift.jar,/usr/share/aws/redshift/spark-redshift/lib/
spark-avro.jar,/usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar \
  my_script.py
```

Autenticazione con l'integrazione di Amazon Redshift per Apache Spark

Utilizzo AWS Secrets Manager per recuperare le credenziali e connettersi ad Amazon Redshift

Puoi autenticarti in modo sicuro su Amazon Redshift archiviando le credenziali in Secrets Manager e facendo in modo che il job `GetSecretValue` API Spark chiami per recuperarle:

```
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
```

Effettua l'autenticazione su Amazon Redshift con un driver JDBC

Imposta nome utente e password all'interno di JDBC URL

Puoi autenticare un job Spark in un cluster Amazon Redshift specificando il nome e la password del database Amazon Redshift nel. JDBC URL

> **Note**
>
> Se trasmetti le credenziali del database inURL, chiunque abbia accesso ad esse può accedere anche alle credenziali. URL Questo metodo non è generalmente consigliato perché non è un'opzione sicura.

Se la sicurezza non è un problema per la vostra applicazione, potete utilizzare il seguente formato per impostare il nome utente e la password in: JDBC URL

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

Usa l'autenticazione IAM basata con il ruolo di esecuzione del lavoro di Amazon EMR Serverless

A partire dalla versione 6.9.0 di Amazon EMR Serverless, il JDBC driver Amazon Redshift 2.1 o superiore viene integrato nell'ambiente. Con JDBC il driver 2.1 e versioni successive, puoi specificare e non includere il nome utente JDBC URL e la password non elaborati.

È invece possibile specificare uno schema `jdbc:redshift:iam://`. Ciò ordina al JDBC driver di utilizzare il ruolo di esecuzione del lavoro EMR Serverless per recuperare automaticamente le credenziali. Per ulteriori informazioni, consulta Configurare una ODBC connessione JDBC or per utilizzare IAM le credenziali nella Amazon Redshift Management Guide. Un esempio di ciò èURL:

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Le seguenti autorizzazioni sono necessarie per il ruolo di esecuzione del lavoro quando sono soddisfatte le condizioni fornite:

Autorizzazione	Condizioni richieste per il ruolo di esecuzione di processo
`redshift:GetClusterCredentials`	Richiesto al JDBC conducente per recuperare le credenziali da Amazon Redshift
`redshift:DescribeCluster`	Obbligatorio se si specifica il cluster Amazon Redshift e Regione AWS al JDBC URL posto dell'endpoint
`redshift-serverless:GetCredentials`	Richiesto al JDBC driver per recuperare le credenziali da Amazon Redshift Serverless
`redshift-serverless:GetWorkgroup`	Obbligatorio se utilizzi Amazon Redshift Serverless e specifichi il nome del gruppo di lavoro e la URL regione

Connessione ad Amazon Redshift all'interno di un altro VPC

Quando configuri un cluster Amazon Redshift o un gruppo di lavoro Amazon Redshift Serverless con un provisioning nell'ambito di VPC un, devi configurare la VPC connettività per l'applicazione EMR Amazon Serverless per accedere alle risorse. Per ulteriori informazioni su come configurare la VPC connettività su un'applicazione Serverless, consulta. EMR Configurazione dell'accesso VPC

- Se il cluster Amazon Redshift o il gruppo di lavoro Serverless Amazon Redshift fornito sono accessibili al pubblico, puoi specificare una o più sottoreti private a cui è collegato un gateway quando crei applicazioni Serverless. NAT EMR
- Se il cluster Amazon Redshift o il gruppo di lavoro Serverless Amazon Redshift fornito non sono accessibili al pubblico, devi creare un VPC endpoint gestito Amazon Redshift per il tuo cluster Amazon Redshift come descritto in. Configurazione dell'accesso VPC In alternativa, puoi creare il tuo gruppo di lavoro Amazon Redshift Serverless come descritto in Connessione ad Amazon Redshift Serverless nella Amazon Redshift Management Guide. È necessario associare il cluster o il sottogruppo alle sottoreti private specificate al momento della creazione dell'applicazione Serverless. EMR

> **Note**
>
> Se utilizzi l'autenticazione IAM basata e le tue sottoreti private per l'applicazione EMR Serverless non dispongono di un NAT gateway collegato, devi anche creare un VPC endpoint su tali sottoreti per Amazon Redshift o Amazon Redshift Serverless. In questo modo, l'autista può recuperare le credenziali. JDBC

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'origine dati API e con Spark. SQL

Data source API

PySpark Utilizzalo per leggere e scrivere dati di esempio da e verso un database Amazon Redshift con origine dati. API

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
sql_context = SQLContext(sc)

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

df = sql_context.read \
```

```
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "table-name") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws-iam-role-arn") \
    .load()

df.write \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "table-name-copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws-iam-role-arn") \
    .mode("error") \
    .save()
```

SparkSQL

PySpark Utilizzalo per leggere e scrivere dati di esempio da e verso un database Amazon Redshift con Spark. SQL

```
import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

bucket = "s3://path/for/temp/data"
tableName = "table-name" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {table-name} (country string, data string)
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{table-name}', tempdir '{bucket}', url '{url}', aws_iam_role
 '{aws-iam-role-arn}' ); """
```

```
spark.sql(s)

columns = ["country" ,"data"]
data = [("test-country","test-data")]
df = spark.sparkContext.parallelize(data).toDF(columns)

# Insert data into table
df.write.insertInto(table-name, overwrite=False)
df = spark.sql(f"SELECT * FROM {table-name}")
df.show()
```

Considerazioni e limitazioni relative all'utilizzo del connettore Spark

- Ti consigliamo di attivare SSL la JDBC connessione da Spark su Amazon EMR ad Amazon Redshift.
- Ti consigliamo di gestire le credenziali per il cluster Amazon Redshift in AWS Secrets Manager come best practice. Vedi Utilizzo AWS Secrets Manager per recuperare le credenziali per la connessione ad Amazon Redshift, ad esempio.
- Ti consigliamo di passare un IAM ruolo con il parametro `aws_iam_role` per il parametro di autenticazione Amazon Redshift.
- Il parametro `tempformat` attualmente non supporta il formato Parquet.
- I `tempdir` URI punti rimandano a una sede 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 Amazon S3 utilizzati.
  - Si consiglia di utilizzare le policy del ciclo di vita di Amazon S3 per definire le regole di conservazione del bucket Amazon S3.
  - Amazon verifica EMR sempre il codice importato dall'open source nell'immagine. Per motivi di sicurezza, non supportiamo i seguenti metodi di autenticazione da Spark ad Amazon S3:

- Impostazione AWS chiavi di accesso nella classificazione della configurazione `hadoop-env`
- Encoding AWS chiavi di accesso in `tempdir` URI

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

Connessione a DynamoDB con Amazon Serverless EMR

In questo tutorial, carichi un sottoinsieme di dati dallo United States Board on Geographic Names in un bucket Amazon S3 e poi usi Hive o Spark su Amazon Serverless per copiare i dati in una tabella EMR Amazon DynamoDB su cui eseguire query.

Fase 1: caricare i dati in un bucket Amazon S3

Per creare un bucket Amazon S3, segui le istruzioni in Creazione di un bucket nella Guida per l'utente della console di Amazon Simple Storage Service. Sostituisci i riferimenti a *DOC-EXAMPLE-BUCKET* con il nome del bucket appena creato. Ora la tua applicazione EMR Serverless è pronta per eseguire i lavori.

1. Scarica l'archivio di dati di esempio `features.zip` con il seguente comando.

```
wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
features.zip
```

2. Estrai il `features.txt` file dall'archivio e visualizza le prime poche righe del file:

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

Il risultato dovrebbe essere simile al seguente.

```
1535908|Big Run|Stream|WV|38.6370428|-80.8595469|794
875609|Constable Hook|Cape|NJ|40.657881|-74.0990309|7
```

```
1217998|Gooseberry Island|Island|RI|41.4534361|-71.3253284|10
26603|Boone Moore Spring|Spring|AZ|34.0895692|-111.410065|3681
1506738|Missouri Flat|Flat|WA|46.7634987|-117.0346113|2605
1181348|Minnow Run|Stream|PA|40.0820178|-79.3800349|1558
1288759|Hunting Creek|Stream|TN|36.343969|-83.8029682|1024
533060|Big Charles Bayou|Bay|LA|29.6046517|-91.9828654|0
829689|Greenwood Creek|Stream|NE|41.596086|-103.0499296|3671
541692|Button Willow Island|Island|LA|31.9579389|-93.0648847|98
```

I campi in ogni riga qui indicano un identificatore univoco, un nome, un tipo di elemento naturale, lo stato, la latitudine in gradi, la longitudine in gradi e l'altezza in piedi.

3. Carica i tuoi dati su Amazon S3

```
aws s3 cp features.txt s3://DOC-EXAMPLE-BUCKET/features/
```

Fase 2: Creare una tabella Hive

Usa Apache Spark o Hive per creare una nuova tabella Hive che contenga i dati caricati in Amazon S3.

Spark

Per creare una tabella Hive con Spark, esegui il seguente comando.

```
import org.apache.spark.sql.SparkSession

val sparkSession = SparkSession.builder().enableHiveSupport().getOrCreate()

sparkSession.sql("CREATE TABLE hive_features \
    (feature_id BIGINT, \
    feature_name STRING, \
    feature_class STRING, \
    state_alpha STRING, \
    prim_lat_dec DOUBLE, \
    prim_long_dec DOUBLE, \
    elev_in_ft BIGINT) \
    ROW FORMAT DELIMITED \
    FIELDS TERMINATED BY '|' \
    LINES TERMINATED BY '\n' \
    LOCATION 's3://DOC-EXAMPLE_BUCKET/features';")
```

Ora hai una tabella Hive popolata con i dati del file. `features.txt` Per verificare che i dati siano presenti nella tabella, esegui una SQL query Spark come mostrato nell'esempio seguente.

```
sparkSession.sql(
    "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

Per creare una tabella Hive con Hive, esegui il comando seguente.

```
CREATE TABLE hive_features
    (feature_id             BIGINT,
    feature_name            STRING ,
    feature_class           STRING ,
    state_alpha             STRING,
    prim_lat_dec            DOUBLE ,
    prim_long_dec           DOUBLE ,
    elev_in_ft              BIGINT)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '|'
    LINES TERMINATED BY '\n'
    LOCATION 's3://DOC-EXAMPLE-BUCKET/features';
```

Ora hai una tabella Hive che contiene i dati del file. `features.txt` Per verificare che i dati siano presenti nella tabella, esegui una query HiveQL, come mostrato nell'esempio seguente.

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

Fase 3: Copiare i dati su DynamoDB

Usa Spark o Hive per copiare i dati in una nuova tabella DynamoDB.

Spark

Per copiare i dati dalla tabella Hive creata nel passaggio precedente in DynamoDB, segui i passaggi 1-3 in Copiare i dati su DynamoDB. Questo crea una nuova tabella DynamoDB chiamata. `Features` È quindi possibile leggere i dati direttamente dal file di testo e copiarli nella tabella DynamoDB, come illustrato nell'esempio seguente.

```
import com.amazonaws.services.dynamodbv2.model.AttributeValue
```

```
import org.apache.hadoop.dynamodb.DynamoDBItemWritable
import org.apache.hadoop.dynamodb.read.DynamoDBInputFormat
import org.apache.hadoop.io.Text
import org.apache.hadoop.mapred.JobConf
import org.apache.spark.SparkContext

import scala.collection.JavaConverters._

object EmrServerlessDynamoDbTest {

    def main(args: Array[String]): Unit = {

        jobConf.set("dynamodb.input.tableName", "Features")
        jobConf.set("dynamodb.output.tableName", "Features")
        jobConf.set("dynamodb.region", "region")

        jobConf.set("mapred.output.format.class",
 "org.apache.hadoop.dynamodb.write.DynamoDBOutputFormat")
        jobConf.set("mapred.input.format.class",
 "org.apache.hadoop.dynamodb.read.DynamoDBInputFormat")

        val rdd = sc.textFile("s3://DOC-EXAMPLE-BUCKET/ddb-connector/")
            .map(row => {
                val line = row.split("\\|")
                val item = new DynamoDBItemWritable()

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

                item.setItem(Map(
                    "feature_id" -> new AttributeValue().withN(line(0)),
                    "feature_name" -> new AttributeValue(line(1)),
                    "feature_class" -> new AttributeValue(line(2)),
                    "state_alpha" -> new AttributeValue(line(3)),
                    "prim_lat_dec" -> new AttributeValue().withN(line(4)),
                    "prim_long_dec" -> new AttributeValue().withN(line(5)),
                    "elev_in_ft" -> elevInFt)
                    .asJava)
                (new Text(""), item)
        })
        rdd.saveAsHadoopDataset(jobConf)
```

```
    }
}
```

Hive

Per copiare i dati dalla tabella Hive creata nel passaggio precedente in DynamoDB, segui le istruzioni in Copiare i dati su DynamoDB.

Fase 4: Interrogare i dati da DynamoDB

Usa Spark o Hive per interrogare la tua tabella DynamoDB.

Spark

Per interrogare i dati dalla tabella DynamoDB creata nel passaggio precedente, puoi utilizzare Spark o SQL Spark. MapReduce API

Example — Interroga la tua tabella DynamoDB con Spark SQL

La seguente SQL query Spark restituisce un elenco di tutti i tipi di feature in ordine alfabetico.

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
    FROM ddb_features \
    ORDER BY feature_class;")
```

La seguente SQL query Spark restituisce un elenco di tutti i laghi che iniziano con la lettera M.

```
val dataFrame = sparkSession.sql("SELECT feature_name, state_alpha \
    FROM ddb_features \
    WHERE feature_class = 'Lake' \
    AND feature_name LIKE 'M%' \
    ORDER BY feature_name;")
```

La seguente SQL query Spark restituisce un elenco di tutti gli stati con almeno tre caratteristiche superiori a un miglio.

```
val dataFrame = sparkSession.dql("SELECT state_alpha, feature_class, COUNT(*) \
    FROM ddb_features \
    WHERE elev_in_ft > 5280 \
```

```
    GROUP by state_alpha, feature_class \
    HAVING COUNT(*) >= 3 \
    ORDER BY state_alpha, feature_class;")
```

Example — Interroga la tua tabella DynamoDB con Spark MapReduce API

La seguente MapReduce query restituisce un elenco di tutti i tipi di feature in ordine alfabetico.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
    .map(pair => pair._2.get("feature_class").getS)
    .distinct()
    .sortBy(value => value)
    .toDF("feature_class")
```

La seguente MapReduce query restituisce un elenco di tutti i laghi che iniziano con la lettera M.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
    .filter(pair => "Lake".equals(pair._2.get("feature_class").getS))
    .filter(pair => pair._2.get("feature_name").getS.startsWith("M"))
    .map(pair => (pair._2.get("feature_name").getS,
 pair._2.get("state_alpha").getS))
    .sortBy(_._1)
    .toDF("feature_name", "state_alpha")
```

La seguente MapReduce query restituisce un elenco di tutti gli stati con almeno tre caratteristiche superiori a un miglio.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => pair._2.getItem)
    .filter(pair => pair.get("elev_in_ft").getN != null)
    .filter(pair => Integer.parseInt(pair.get("elev_in_ft").getN) > 5280)
    .groupBy(pair => (pair.get("state_alpha").getS, pair.get("feature_class").getS))
    .filter(pair => pair._2.size >= 3)
    .map(pair => (pair._1._1, pair._1._2, pair._2.size))
    .sortBy(pair => (pair._1, pair._2))
    .toDF("state_alpha", "feature_class", "count")
```

Hive

Per interrogare i dati dalla tabella DynamoDB creata nel passaggio precedente, segui le istruzioni in Interrogare i dati nella tabella DynamoDB.

Configurazione dell'accesso multi-account

Per configurare l'accesso tra più account per EMR Serverless, completa i seguenti passaggi. Nell'esempio, `AccountA` è l'account in cui hai creato l'applicazione Amazon EMR Serverless ed `AccountB` è l'account in cui si trova Amazon DynamoDB.

1. Crea una tabella DynamoDB in. `AccountB` Per ulteriori informazioni, consulta Fase 1: Creare una tabella.
2. Crea un `Cross-Account-Role-B` IAM ruolo in `AccountB` grado di accedere alla tabella DynamoDB.

   a. Accedi al AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.

   b. Scegli Ruoli e crea un nuovo ruolo chiamato`Cross-Account-Role-B`. Per ulteriori informazioni su come creare IAM ruoli, consulta Creazione di IAM ruoli nella guida per l'utente.

   c. Crea una IAM policy che conceda le autorizzazioni per accedere alla tabella DynamoDB tra account. Quindi allega la policy a. IAM `Cross-Account-Role-B`

      Di seguito è riportata una politica che concede l'accesso a una tabella DynamoDB. `CrossAccountTable`

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

   d. Modifica la relazione di fiducia per il ruolo `Cross-Account-Role-B`.

Per configurare la relazione di trust per il ruolo, scegli la scheda Trust Relationships nella IAM console relativa al ruolo che hai creato nel Passaggio 2: Cross-Account-Role-B.

Seleziona Modifica relazione di fiducia, quindi aggiungi il seguente documento di policy. Questo documento consente `Job-Execution-Role-A` `AccountA` di assumere questo `Cross-Account-Role-B` ruolo.

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

e. Concedi `Job-Execution-Role-A` `AccountA` con `- STS Assume role` le autorizzazioni da assumere`Cross-Account-Role-B`.

Nella IAM console per Account AWS `AccountA`, seleziona`Job-Execution-Role-A`. Aggiungi la seguente istruzione di policy a `Job-Execution-Role-A` per autorizzare l'operazione `AssumeRole` nel ruolo `Cross-Account-Role-B`.

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

f. Imposta la `dynamodb.customAWSCredentialsProvider` proprietà con il valore come `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` nella classificazione del sito principale. Imposta la variabile di ambiente `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` con il ARN valore di. `Cross-Account-Role-B`

3. Esegui il job Spark o Hive utilizzando. `Job-Execution-Role-A`

Considerazioni

Considerazioni sull'utilizzo del connettore DynamoDB con Apache Spark

- Spark SQL non supporta la creazione di una tabella Hive con l'opzione storage-handler. Per ulteriori informazioni, consulta Specificare il formato di archiviazione per le tabelle Hive nella documentazione di Apache Spark.
- Spark SQL non supporta il funzionamento con il `STORED BY` gestore di archiviazione. Se desideri interagire con una tabella DynamoDB tramite una tabella Hive esterna, usa Hive per creare prima la tabella.
- Per tradurre una query in una query DynamoDB, il connettore DynamoDB utilizza il pushdown dei predicati. Predicate pushdown filtra i dati in base a una colonna mappata alla chiave di partizione di una tabella DynamoDB. Predicate pushdown funziona solo quando si utilizza il connettore con Spark e non con. SQL MapReduce API

Considerazioni sull'utilizzo del connettore DynamoDB con Apache Hive

Ottimizzazione del numero massimo di mappatori

- Se si utilizza la `SELECT` query per leggere i dati da una tabella Hive esterna mappata a DynamoDB, il numero di attività di mappa EMR su Serverless viene calcolato come il throughput di lettura totale configurato per la tabella DynamoDB, diviso per il throughput per task di mappa. La velocità effettiva predefinita per attività di mappa è 100.
- Il job Hive può utilizzare il numero di attività di mappa oltre al numero massimo di contenitori configurati per applicazione EMR Serverless, a seconda del throughput di lettura configurato per DynamoDB. Inoltre, una query Hive a esecuzione prolungata può consumare tutta la capacità di lettura assegnata alla tabella DynamoDB. Ciò ha un impatto negativo sugli altri utenti.
- È possibile utilizzare la `dynamodb.max.map.tasks` proprietà per impostare un limite superiore per le attività di mappatura. È inoltre possibile utilizzare questa proprietà per ottimizzare la quantità di dati letti da ogni attività della mappa in base alla dimensione del contenitore dell'attività.
- È possibile impostare la `dynamodb.max.map.tasks` proprietà a livello di query Hive o nella `hive-site` classificazione del start-job-run comando. Questo valore deve essere maggiore o uguale a 1. Quando Hive elabora la tua query, il job Hive risultante utilizza solo i valori di `dynamodb.max.map.tasks` quando legge dalla tabella DynamoDB.

Ottimizzazione della velocità di scrittura per attività

- Il throughput di scrittura per attività su EMR Serverless viene calcolato come il throughput di scrittura totale configurato per una tabella DynamoDB, diviso per il valore della proprietà. `mapreduce.job.maps` Per Hive, il valore predefinito di questa proprietà è 2. Pertanto, le prime due attività nella fase finale del processo Hive possono consumare tutta la velocità di scrittura. Ciò comporta una riduzione delle scritture di altre attività nello stesso lavoro o in altri lavori.
- Per evitare la limitazione della scrittura, è possibile impostare il valore della `mapreduce.job.maps` proprietà in base al numero di attività nella fase finale o alla velocità di scrittura che si desidera allocare per attività. Imposta questa proprietà nella `mapred-site` classificazione del comando su Serverless. start-job-run EMR

Sicurezza

Sicurezza nel cloud presso AWS è la massima priorità. Come un AWS cliente, trarrai vantaggio da un'architettura di data center e rete progettata per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS e tu. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud — AWS è responsabile della protezione dell'infrastruttura in esecuzione AWS servizi in AWS Cloud. AWS ti offre anche servizi che puoi utilizzare in modo sicuro. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito del AWS programmi di conformità. Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon EMR Serverless, consulta AWS servizi rientranti nell'ambito del programma di conformità.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. Inoltre, sei responsabile anche di altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e le leggi e le normative applicabili.

Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa quando usi Amazon EMR Serverless. Gli argomenti di questo capitolo mostrano come configurare Amazon EMR Serverless e utilizzare altri AWS servizi per soddisfare i tuoi obiettivi di sicurezza e conformità.

Argomenti

- Sicurezza dell'infrastruttura in Amazon EMR Serverless
- Analisi della configurazione e delle vulnerabilità in Amazon Serverless EMR

Best practice di sicurezza per Amazon EMR Serverless

Amazon EMR Serverless offre una serie di funzionalità di sicurezza da prendere in considerazione durante lo sviluppo e l'implementazione delle proprie politiche di sicurezza. Le seguenti best practice sono linee guida generali e non rappresentano una soluzione di sicurezza completa. Poiché queste best practice potrebbero non essere appropriate o sufficienti per l'ambiente, gestiscile come considerazioni utili anziché prescrizioni.

Applicazione del principio del privilegio minimo

EMRServerless fornisce una politica di accesso granulare per le applicazioni che utilizzano IAM ruoli, come i ruoli di esecuzione. 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.

Isolamento di un codice dell'applicazione non attendibile

EMRServerless crea un isolamento di rete completo tra i lavori appartenenti a diverse EMR applicazioni Serverless. Nei casi in cui si desidera l'isolamento a livello di processo, è consigliabile isolare i lavori in diverse applicazioni Serverless. EMR

Autorizzazioni di controllo degli accessi () basate sui ruoli RBAC

Gli amministratori devono controllare rigorosamente le autorizzazioni di controllo degli accessi () RBAC basate sul ruolo per le applicazioni serverless. EMR

Protezione dei dati

Il AWS il modello di responsabilità condivisa si applica alla protezione dei dati in Amazon EMR Serverless. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutte le AWS Cloud. L'utente è responsabile di mantenere il controllo sui contenuti ospitati su questa infrastruttura. Questo contenuto include le attività di configurazione e gestione della sicurezza per AWS i servizi che utilizzi. Per ulteriori informazioni sulla privacy dei dati,

consulta la pagina Privacy dei dati FAQ. Per informazioni sulla protezione dei dati in Europa, consulta il AWS Modello di responsabilità condivisa e post sul GDPR blog sul AWS Blog sulla sicurezza.

Ai fini della protezione dei dati, ti consigliamo di proteggere AWS credenziali dell'account e configura account individuali 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.
- UsaSSL/TLSper comunicare con AWS risorse. Consigliamo TLS 1.2 o versioni successive.
- Configurazione API e registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizzo AWS soluzioni di crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno AWS 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 Amazon EMR Serverless per crittografare i dati a riposo e in transito.
- Se hai bisogno di FIPS 140-2 moduli crittografici convalidati per l'accesso AWS tramite un'interfaccia a riga di comando o unAPI, utilizza un endpoint. FIPS Per ulteriori informazioni sugli FIPS endpoint disponibili, vedere 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 Serverless o altro AWS servizi che utilizzano la console, API AWS CLI, oppure AWS SDKs. Tutti i dati che inserisci in Amazon EMR Serverless o in altri servizi potrebbero essere raccolti per essere inclusi nei log di diagnostica. Quando fornisci un messaggio URL a un server esterno, non includere le informazioni sulle credenziali URL per convalidare la tua richiesta a quel 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, incluse le chiavi gestite da AWS Key Management Service, chiavi gestite da Amazon S3 e chiavi e certificati di

fornitori personalizzati da te forniti. Quando si utilizza AWS KMS in qualità di fornitore delle chiavi, vengono addebitati costi per l'archiviazione e l'uso delle chiavi di crittografia. Per ulteriori informazioni, consulta 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 a riposo per EMRFS i dati in Amazon S3

Ogni applicazione EMR Serverless utilizza una versione di rilascio specifica, che include EMRFS (EMRFile System). La crittografia Amazon S3 funziona con oggetti EMR File System (EMRFS) letti e scritti su Amazon S3. Puoi specificare la crittografia lato server di Amazon S3 (SSE) o la crittografia lato client (CSE) come modalità di crittografia predefinita quando abiliti la crittografia a riposo. Facoltativamente, è possibile specificare diversi metodi di crittografia per singoli bucket utilizzando override di crittografia per bucket. Indipendentemente dal fatto che la crittografia Amazon S3 sia abilitata, Transport Layer Security (TLS) crittografa EMRFS gli oggetti in transito tra i nodi del EMR cluster e Amazon S3. Se utilizzi Amazon S3 CSE con chiavi gestite dal cliente, il ruolo di esecuzione utilizzato per eseguire lavori in un'applicazione EMR Serverless deve avere accesso alla chiave. Per informazioni approfondite sulla crittografia di Amazon S3, consulta Protection data using encryption nella Amazon Simple Storage Service Developer Guide.

> **Note**
>
> Quando usi AWS KMS, l'archiviazione e l'uso delle chiavi di crittografia sono a pagamento. Per ulteriori informazioni, consulta 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 informazioniSSE, consulta la sezione Protezione dei dati utilizzando la crittografia lato server nella Amazon Simple Storage Service Developer Guide.

Puoi scegliere tra due diversi sistemi di gestione delle chiavi quando lo specifichi SSE in Amazon EMR Serverless:

- SSE-S3 - Amazon S3 gestisce le chiavi per te. Non è richiesta alcuna configurazione aggiuntiva su Serverless. EMR
- SSE- KMS - Si utilizza un AWS KMS key da configurare con politiche adatte a EMR Serverless. Non è richiesta alcuna configurazione aggiuntiva su EMR Serverless.

Per utilizzare AWS KMS crittografia per i dati che scrivi su Amazon S3, hai due opzioni quando usi la. `StartJobRun` API Puoi abilitare la crittografia per tutto ciò che scrivi su Amazon S3 oppure puoi abilitare la crittografia per i dati che scrivi in un bucket specifico. Per ulteriori informazioni su `StartJobRun`API, consulta il Serverless Reference. EMR API

Per accendere AWS KMS crittografia per tutti i dati che scrivi su Amazon S3, usa i seguenti comandi quando chiami il. `StartJobRun` API

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

Per accendere AWS KMS crittografia per i dati che scrivi in un bucket specifico, usa i seguenti comandi quando chiami il `StartJobRun`API.

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

SSEcon chiavi fornite dal cliente (SSE-C) non è disponibile per l'uso con Serverless. EMR

Crittografia lato client Amazon S3

Con la crittografia lato client di Amazon S3, la crittografia e la decrittografia di Amazon S3 avvengono nel client disponibile in ogni versione di Amazon. EMRFS EMR 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 sul 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 decrittografare o crittografare. Se utilizzi Amazon S3 CSE con chiavi gestite dal cliente, il ruolo di esecuzione utilizzato per eseguire lavori in un'applicazione EMR Serverless deve avere accesso alla chiave. Potrebbero essere applicati costi aggiuntiviKMS. Per ulteriori informazioni su queste differenze, consulta la sezione Protezione dei dati utilizzando la crittografia lato client nella Amazon Simple Storage Service Developer Guide.

Crittografia del disco locale

I dati archiviati nello storage temporaneo sono crittografati con chiavi di proprietà del servizio utilizzando l'algoritmo crittografico -256 standard del settore. AES

Gestione delle chiavi

È possibile configurare la rotazione automatica delle KMS chiavi. KMS Verrà effettuata una rotazione delle chiavi una volta all'anno salvando le vecchie chiavi a tempo indeterminato in modo che i dati possano ancora essere decrittografati. Per ulteriori informazioni, consulta Rotazione delle chiavi cliente principali.

Crittografia in transito

Le seguenti funzionalità di crittografia specifiche dell'applicazione sono disponibili con Amazon Serverless: EMR

- Spark
  - Per impostazione predefinita, la comunicazione tra i driver Spark e gli esecutori è autenticata e interna. RPCla comunicazione tra driver ed esecutori è crittografata.
- Hive
  - Comunicazione tra AWS Le applicazioni Glue metastore e EMR Serverless avvengono tramite. TLS

Dovresti consentire solo connessioni crittografate su HTTPS (TLS) utilizzando le policy dei IAM bucket aws: SecureTransport condition on Amazon S3.

Identity and Access Management (IAM) in Amazon EMR Serverless

AWS Identity and Access Management (IAM) è un Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso a AWS risorse. IAMgli amministratori controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse Amazon EMR Serverless. IAMè un Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità

- Gestione dell'accesso con policy
- Come funziona EMR Serverless con IAM
- Utilizzo di ruoli collegati ai servizi per Serverless EMR
- Ruoli Job Runtime per Amazon EMR Serverless
- Esempi di policy di accesso degli utenti per Serverless EMR
- Policy per il controllo degli accessi basato su tag
- Esempi di policy basate sull'identità per Serverless EMR
- Aggiornamenti Amazon EMR Serverless a AWS policy gestite
- Risoluzione dei problemi di identità e accesso ad Amazon EMR Serverless

Destinatari

Come si usa AWS Identity and Access Management (IAM) differisce, a seconda del lavoro svolto in Amazon EMR Serverless.

Utente del servizio: se utilizzi il servizio Amazon EMR Serverless per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più funzionalità di Amazon EMR Serverless per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di Amazon EMR Serverless, consultaRisoluzione dei problemi di identità e accesso ad Amazon EMR Serverless.

Amministratore del servizio: se sei responsabile delle risorse Amazon EMR Serverless della tua azienda, probabilmente hai pieno accesso ad Amazon EMR Serverless. È tuo compito determinare a quali funzionalità e risorse Amazon EMR Serverless devono accedere gli utenti del servizio. Devi quindi inviare richieste all'IAMamministratore per modificare le autorizzazioni degli utenti del servizio. Consulta le informazioni contenute in questa pagina per comprendere i concetti di base diIAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM Amazon EMR Serverless, consultaIdentity and Access Management (IAM) in Amazon EMR Serverless.

IAMamministratore: se sei un IAM amministratore, potresti voler saperne di più su come scrivere policy per gestire l'accesso ad Amazon EMR Serverless. Per visualizzare esempi di policy basate sull'identità di Amazon EMR Serverless che puoi utilizzare, consulta. IAM Esempi di policy basate sull'identità per Serverless EMR

Autenticazione con identità

L'autenticazione è il modo in cui accedi a AWS utilizzando le tue credenziali di identità. Devi essere autenticato (aver effettuato l'accesso a AWS) come Utente root dell'account AWS, come IAM utente o assumendo un IAM ruolo.

Puoi accedere a AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAMIdentity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali Google o Facebook sono esempi di identità federate. Quando accedi come identità federata, l'amministratore aveva precedentemente configurato la federazione delle identità utilizzando i ruoli. IAM Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente che sei, puoi accedere a AWS Management Console o il AWS portale di accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Accedi ad AWS Guida per l'utente.

Se accedi AWS programmaticamente, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le richieste utilizzando le credenziali dell'utente. Se non usi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, vedi Firma AWS APIrichieste nella Guida IAM per l'utente.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del proprio account. Per ulteriori informazioni, consulta Autenticazione a più fattori nel AWS IAM Identity Center Guida per l'utente e utilizzo dell'autenticazione a più fattori () MFA in AWS nella Guida per l'utente di IAM.

Account AWS utente root

Quando crei un Account AWS, inizi con un'unica identità di accesso con accesso completo a tutti Servizi AWS e le risorse presenti nell'account. Questa identità è denominata Account AWS utente root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per l'elenco completo delle attività che richiedono l'accesso come utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'IAMutente.

Identità federata

Come procedura ottimale, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, un provider di identità Web, il AWS Directory Service, la directory Identity Center o qualsiasi utente che accede Servizi AWS utilizzando le credenziali fornite tramite una fonte di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare AWS IAM Identity Center. È possibile creare utenti e gruppi in IAM Identity Center oppure connettersi e sincronizzarsi con un set di utenti e gruppi nella propria fonte di identità per utilizzarli su tutti i Account AWS e applicazioni. Per informazioni su IAM Identity Center, vedi Cos'è IAM Identity Center? nel AWS IAM Identity Center Guida per l'utente.

IAM users and groups

Un IAMutente è un'identità all'interno del tuo Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Laddove possibile, consigliamo di fare affidamento su credenziali temporanee anziché creare IAM utenti con credenziali a lungo termine come password e chiavi di accesso. Tuttavia, se hai casi d'uso specifici che richiedono credenziali a lungo termine con IAM gli utenti, ti consigliamo di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta Ruotare regolarmente le chiavi di accesso per i casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente. IAM

Un IAMgruppo è un'identità che specifica un insieme di utenti. IAM Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, potresti avere un gruppo denominato IAMAdminse concedere a quel gruppo le autorizzazioni per IAM amministrare le risorse.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un IAM utente (anziché un ruolo) nella Guida per l'IAMutente.

IAMruoli

Un IAMruolo è un'identità all'interno del tuo Account AWS che dispone di autorizzazioni specifiche. È simile a un IAM utente, ma non è associato a una persona specifica. È possibile assumere temporaneamente un IAM ruolo nel AWS Management Console cambiando ruolo. Puoi assumere un ruolo chiamando un AWS CLI oppure AWS APIoperazione o utilizzando un comando personalizzatoURL. Per ulteriori informazioni sui metodi di utilizzo dei ruoli, vedere Utilizzo IAM dei ruoli nella Guida per l'IAMutente.

IAMi ruoli con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per informazioni sui ruoli per la federazione, vedere Creazione di un ruolo per un provider di identità di terze parti nella Guida per l'IAMutente. Se utilizzi IAM Identity Center, configuri un set di autorizzazioni. Per controllare a cosa possono accedere le identità dopo l'autenticazione, IAM Identity Center correla il set di autorizzazioni a un ruolo in. IAM Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nel AWS IAM Identity Center Guida per l'utente.
- Autorizzazioni IAM utente temporanee: un IAM utente o un ruolo può assumere un IAM ruolo per assumere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso su più account: puoi utilizzare un IAM ruolo per consentire a qualcuno (un responsabile fidato) di un altro account di accedere alle risorse del tuo account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per conoscere la differenza tra i ruoli e le politiche basate sulle risorse per l'accesso tra più account, consulta l'accesso alle risorse tra account IAM nella Guida per l'utente. IAM
- Accesso tra servizi: alcuni Servizi AWS usa le funzionalità in altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è normale che quel servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso diretto (FAS): quando utilizzi un IAM utente o un ruolo per eseguire azioni in AWS, sei considerato un preside. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FASutilizza le autorizzazioni del principale che chiama un Servizio AWS, in combinazione con la richiesta Servizio AWS per effettuare richieste ai servizi a valle. FASle richieste vengono effettuate solo quando un servizio

riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse da completare. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli FAS delle politiche relative alle richieste, consulta Forward access sessions.

- Ruolo di servizio: un ruolo di servizio è un IAMruolo che un servizio assume per eseguire azioni per conto dell'utente. Un IAM amministratore può creare, modificare ed eliminare un ruolo di servizio dall'internoIAM. Per ulteriori informazioni, vedere Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
- Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo di eseguire un'azione per conto dell'utente. I ruoli collegati ai servizi vengono visualizzati nel Account AWS e sono di proprietà del servizio. Un IAM amministratore può visualizzare, ma non modificare le autorizzazioni per i ruoli collegati al servizio.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un IAM ruolo per gestire le credenziali temporanee per le applicazioni in esecuzione su un'EC2istanza e in fase di creazione AWS CLI oppure AWS APIrichieste. Ciò è preferibile alla memorizzazione delle chiavi di accesso all'interno dell'EC2istanza. Per assegnare un AWS assegnare un ruolo a un'EC2istanza e renderlo disponibile a tutte le relative applicazioni, è necessario creare un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull'EC2istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Usare un IAM ruolo per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella Guida per l'IAMutente.

Per sapere se utilizzare IAM ruoli o IAM utenti, consulta Quando creare un IAM ruolo (anziché un utente) nella Guida per l'IAMutente.

Gestione dell'accesso con policy

Puoi controllare l'accesso in AWS creando politiche e allegandole a AWS identità o risorse. Una politica è un oggetto in AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata in AWS come JSON documenti. Per ulteriori informazioni sulla struttura e il contenuto dei documenti relativi alle JSON politiche, vedere Panoramica delle JSON politiche nella Guida per l'IAMutente.

Gli amministratori possono utilizzare AWS JSONpolitiche per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti il permesso di eseguire azioni sulle risorse di cui hanno bisogno, un IAM amministratore può creare IAM politiche. L'amministratore può quindi aggiungere le IAM politiche ai ruoli e gli utenti possono assumerli.

IAMle politiche definiscono le autorizzazioni per un'azione indipendentemente dal metodo utilizzato per eseguire l'operazione. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale criterio può ottenere informazioni sul ruolo da AWS Management Console, il AWS CLI, o il AWS API.

Policy basate su identità

I criteri basati sull'identità sono documenti relativi ai criteri di JSON autorizzazione che è possibile allegare a un'identità, ad esempio un IAM utente, un gruppo di utenti o un ruolo. Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una politica basata sull'identità, consulta Creazione di politiche nella Guida per l'utente. IAM IAM

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono AWS politiche gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una politica gestita o una politica in linea, consulta Scelta tra politiche gestite e politiche in linea nella Guida per l'IAMutente.

Policy basate su risorse

Le politiche basate sulle risorse sono documenti di JSON policy allegati a una risorsa. Esempi di politiche basate sulle risorse sono le policy di trust dei IAM ruoli 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. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi usare AWS politiche gestite da IAM una politica basata sulle risorse.

Liste di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLssono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy. JSON

Amazon S3, AWS WAF e Amazon VPC sono esempi di servizi che supportanoACLs. Per ulteriori informazioniACLs, consulta la panoramica di Access control list (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite di autorizzazioni è una funzionalità avanzata in cui si impostano le autorizzazioni massime che una politica basata sull'identità può concedere a un'entità (utente o ruolo). IAM IAM È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal`sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità nella Guida per l'utente. IAM IAM
- Politiche di controllo del servizio (SCPs): SCPs sono JSON politiche che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in AWS Organizations. AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di cui è proprietaria la tua azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. I SCP limiti e le autorizzazioni per le entità presenti negli account dei membri, inclusi tutti Utente root dell'account AWS. Per ulteriori informazioni su Organizations andSCPs, vedere Service control policies nel AWS Organizations Guida per l'utente.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta le politiche di sessione nella Guida IAM per l'utente.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella Guida per l'IAMutente.

Come funziona EMR Serverless con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon EMR Serverless, scopri quali IAM funzionalità sono disponibili per l'uso con Amazon EMR Serverless.

IAMfunzionalità che puoi usare con Serverless EMR

IAMcaratteristica	Supporto Amazon EMR Serverless
Policy basate su identità	Sì
Policy basate su risorse	No
Azioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione delle policy	No
ACLs	No
ABAC(tag nelle politiche)	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 EMR Serverless e altro AWS i servizi funzionano con la maggior parte delle IAM funzionalità, vedi AWS servizi compatibili con IAM la Guida per l'IAMutente.

Politiche basate sull'identità per Serverless EMR

Supporta le policy basate su identità: sì

Le politiche basate sull'identità sono documenti relativi alle politiche di JSON autorizzazione che è possibile allegare a un'identità, ad esempio un IAM utente, un gruppo di utenti o un ruolo. Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una politica basata sull'identità, consulta Creazione di politiche nella Guida per l'utente. IAM IAM

Con le politiche IAM basate sull'identità, puoi specificare azioni e risorse consentite o negate, nonché le condizioni in base alle quali le azioni sono consentite o negate. Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per ulteriori informazioni su tutti gli elementi che è possibile utilizzare in una JSON politica, vedere il riferimento agli elementi IAM JSON della politica nella Guida per l'IAMutente.

Esempi di policy basate sull'identità per Serverless EMR

Per visualizzare esempi di policy basate sull'identità di Amazon EMR Serverless, consulta. Esempi di policy basate sull'identità per Serverless EMR

Politiche basate sulle risorse all'interno di Serverless EMR

Supporta le policy basate su risorse: no

Le politiche basate sulle risorse sono documenti di policy allegati a JSON una risorsa. Esempi di politiche basate sulle risorse sono le policy di trust dei IAM ruoli 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. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Per abilitare l'accesso tra più account, puoi specificare un intero account o IAM entità in un altro account come principale in una politica basata sulle risorse. L'aggiunta di un principale multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando il principale e la risorsa sono diversi Account AWS, un IAM amministratore dell'account affidabile deve inoltre concedere all'entità principale (utente o ruolo) l'autorizzazione ad accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una

policy basata su risorse concede l'accesso a un principale nello stesso account, non sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta Cross Account Resource Access IAM nella Guida IAM per l'utente.

Azioni politiche per EMR Serverless

Supporta le operazioni di policy: si

Gli amministratori possono utilizzare AWS JSONpolitiche per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'`Action`elemento di una JSON policy descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a una policy. Le azioni politiche in genere hanno lo stesso nome di quelle associate AWS APIoperazione. Esistono alcune eccezioni, come le azioni di sola autorizzazione che non hanno un'operazione corrispondente. API Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di azioni EMR Serverless, consulta Azioni, risorse e chiavi di condizione per Amazon EMR Serverless nel Service Authorization Reference.

Le azioni politiche in EMR Serverless utilizzano il seguente prefisso prima dell'azione.

```
emr-serverless
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "emr-serverless:action1",
      "emr-serverless:action2"
         ]
```

Per visualizzare esempi di policy basate sull'identità di Amazon EMR Serverless, consulta. Esempi di policy basate sull'identità per Serverless EMR

Risorse politiche per Serverless EMR

Supporta le risorse di policy: sì

Gli amministratori possono utilizzare AWS JSONpolitiche per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento `Resource` JSON policy specifica l'oggetto o gli oggetti a cui si applica l'azione. Le istruzioni devono includere un elemento `Resource`o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il relativo Amazon Resource Name (ARN). Puoi eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, 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 Serverless e relativiARNs, consulta Resources defined by Amazon EMR Serverless nel Service Authorization Reference. Per sapere quali azioni puoi specificare per ogni risorsa, consulta Azioni, risorse e chiavi di condizione per Amazon EMR Serverless. ARN

Per visualizzare esempi di policy basate sull'identità di Amazon EMR Serverless, consulta. Esempi di policy basate sull'identità per Serverless EMR

Chiavi delle condizioni delle policy per Serverless EMR

Supporta le chiavi di condizione delle policy specifiche del servizio	No

Gli amministratori possono utilizzare AWS JSONpolitiche per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition`(o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition`è facoltativo. Puoi 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.

Se specificate più `Condition` elementi in un'istruzione o più chiavi in un singolo `Condition` elemento, AWS li valuta utilizzando un'ANDoperazione logica. Se specificate più valori per una

singola chiave di condizione, AWS valuta la condizione utilizzando un'0Roperazione logica. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, è possibile concedere a un IAM utente l'autorizzazione ad accedere a una risorsa solo se è contrassegnata con il relativo nome IAM utente. Per ulteriori informazioni, consulta gli elementi IAM della politica: variabili e tag nella Guida IAM per l'utente.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche del servizio. Per vedere tutto AWS chiavi di condizione globali, vedi AWS chiavi di contesto della condizione globale nella Guida IAM per l'utente.

Per visualizzare un elenco di chiavi di condizione di Amazon EMR Serverless e per scoprire quali azioni e risorse è possibile utilizzare una chiave di condizione, consulta Azioni, risorse e chiavi di condizione per Amazon EMR Serverless nel Service Authorization Reference.

Tutte le EC2 azioni di Amazon supportano le chiavi `aws:RequestedRegion` e `ec2:Region` condition. Per ulteriori informazioni, consulta Esempio: limitazione dell'accesso a una regione specifica.

Accedi agli elenchi di controllo (ACLs) in Serverless EMR

SupportiACLs: 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. ACLssono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy. JSON

Controllo degli accessi basato sugli attributi () con Serverless ABAC EMR

Supporti ABAC (tag nelle politiche)	Sì

Il controllo degli accessi basato sugli attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, questi attributi sono chiamati tag. È possibile allegare tag a IAM entità (utenti o ruoli) e a molte AWS risorse. L'etichettatura di entità e risorse è il primo passo diABAC. Quindi si progettano ABAC politiche per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa a cui sta tentando di accedere.

ABACè utile in ambienti in rapida crescita e aiuta in situazioni in cui la gestione delle politiche diventa complicata.

Per controllare l'accesso basato su tag, fornisci 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 Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per ulteriori informazioni suABAC, vedere Cos'è? ABAC nella Guida IAM per l'utente. Per visualizzare un tutorial con i passaggi per la configurazioneABAC, consulta Utilizzare il controllo di accesso basato sugli attributi (ABAC) nella Guida per l'IAMutente.

Utilizzo di credenziali temporanee con Serverless EMR

Supporta le credenziali temporanee: sì

Medio Servizi AWS non funzionano quando accedi utilizzando credenziali temporanee. Per ulteriori informazioni, tra cui Servizi AWS lavorare con credenziali temporanee, vedere Servizi AWS che funzionano con IAM la Guida per l'IAMutente.

Stai utilizzando credenziali temporanee se accedi a AWS Management Console utilizzando qualsiasi metodo tranne il nome utente e la password. Ad esempio, quando accedi AWS utilizzando il link Single Sign-On (SSO) della vostra azienda, tale processo crea automaticamente credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sul cambio di ruolo, consulta Passare a un ruolo (console) nella Guida per l'IAMutente.

È possibile creare manualmente credenziali temporanee utilizzando il AWS CLI oppure AWS API. È quindi possibile utilizzare tali credenziali temporanee per accedere AWS. AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, vedere Credenziali di sicurezza temporanee in. IAM

Autorizzazioni principali multiservizio per Serverless EMR

Supporta sessioni di accesso diretto () FAS: Sì

Quando si utilizza un IAM utente o un ruolo per eseguire azioni in AWS, sei considerato un preside. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione

in un servizio diverso. FASutilizza le autorizzazioni del principale che chiama un Servizio AWS, in combinazione con la richiesta Servizio AWS per effettuare richieste ai servizi a valle. FASle richieste vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse da completare. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli FAS delle politiche relative alle richieste, consulta Forward access sessions.

Ruoli di servizio per EMR Serverless

Supporta i ruoli di servizio	No

Ruoli collegati ai servizi per Serverless EMR

Supporta i ruoli collegati ai servizi	Sì

Per informazioni dettagliate sulla creazione o la gestione di ruoli collegati ai servizi, consulta AWS servizi che funzionano con. 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 Serverless EMR

Usi di Amazon EMR Serverless AWS Identity and Access Management (IAM) ruoli collegati ai servizi. Un ruolo collegato ai servizi è un tipo di IAM ruolo unico collegato direttamente a Serverless. EMR I ruoli collegati ai servizi sono predefiniti da EMR Serverless e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di EMR Serverless perché non è necessario aggiungere manualmente le autorizzazioni necessarie. EMRServerless definisce le autorizzazioni dei suoi ruoli collegati ai servizi e, se non diversamente definito, solo Serverless può assumerne i ruoli. EMR Le autorizzazioni definite includono la politica di fiducia e la politica di autorizzazione e tale politica di autorizzazione non può essere associata a nessun'altra entità. IAM

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questo protegge le tue risorse EMR Serverless perché non puoi rimuovere inavvertitamente l'autorizzazione ad accedere alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, vedi AWS Servizi compatibili IAM e cerca i servizi con Sì nella colonna Ruoli collegati ai servizi. Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni di ruolo collegate ai servizi per Serverless EMR

EMRServerless utilizza il ruolo collegato al servizio denominato per consentirgli di chiamare AWSServiceRoleForAmazonEMRServerless AWS APIsper tuo conto.

Il ruolo AWSServiceRoleForAmazonEMRServerless collegato al servizio prevede che i seguenti servizi assumano il ruolo:

- `ops.emr-serverless.amazonaws.com`

La politica di autorizzazione dei ruoli denominata `AmazonEMRServerlessServiceRolePolicy` consente a EMR Serverless di completare le seguenti azioni sulle risorse specificate.

> **Note**
>
> Il contenuto delle policy gestite cambia, pertanto la politica mostrata qui potrebbe non essere aggiornata. Visualizza la maggior parte delle up-to-date politiche A mazonEMRServerless ServiceRolePolicy nel AWS Management Console.

- Operazione: `ec2:CreateNetworkInterface`
- Operazione: `ec2:DeleteNetworkInterface`
- Operazione: `ec2:DescribeNetworkInterfaces`
- Operazione: `ec2:DescribeSecurityGroups`
- Operazione: `ec2:DescribeSubnets`
- Operazione: `ec2:DescribeVpcs`
- Operazione: `ec2:DescribeDhcpOptions`
- Operazione: `ec2:DescribeRouteTables`
- Operazione: `cloudwatch:PutMetricData`

Di seguito è riportata la `AmazonEMRServerlessServiceRolePolicy` politica completa.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2PolicyStatement",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchPolicyStatement",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/EMRServerless",
                        "AWS/Usage"
                    ]
                }
            }
        }
    ]
}
```

La seguente politica di fiducia è allegata a questo ruolo per consentire al principale EMR Serverless di assumere questo ruolo.

```
{
```

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

È necessario configurare le autorizzazioni per consentire a un'IAMentità (ad esempio un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio. Per ulteriori informazioni, consulta Autorizzazioni dei ruoli collegati ai servizi nella Guida per l'utente. IAM

Creazione di un ruolo collegato ai servizi per Serverless EMR

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando si crea una nuova applicazione EMR Serverless in AWS Management Console (utilizzando EMR Studio), AWS CLI, oppure AWS API, EMR Serverless crea automaticamente il ruolo collegato ai servizi. È necessario configurare le autorizzazioni per consentire a un'IAMentità (ad esempio un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio.

Per creare il ruolo collegato al servizio utilizzando AWSServiceRoleForAmazonEMRServerless IAM

Aggiungi la seguente dichiarazione alla politica delle autorizzazioni per l'IAMentità che deve creare il ruolo collegato al servizio.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

Se elimini questo ruolo collegato ai servizi, puoi ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando crei una nuova applicazione Serverless, EMR EMR Serverless crea nuovamente il ruolo collegato al servizio per te.

Puoi anche utilizzare la IAM console per creare un ruolo collegato al servizio con lo use case Serverless. EMR Nel AWS CLI o il AWS API, crea un ruolo collegato al servizio con il nome del `ops.emr-serverless.amazonaws.com` servizio. Per ulteriori informazioni, vedere Creazione di un ruolo collegato al servizio nella Guida per l'utente. IAM Se elimini il ruolo collegato ai servizi, puoi utilizzare lo stesso processo per crearlo nuovamente.

Modifica di un ruolo collegato al servizio per Serverless EMR

EMRServerless non consente di modificare il ruolo AWSServiceRoleForAmazonEMRServerless collegato al servizio perché diverse entità potrebbero fare riferimento al ruolo. Non è possibile modificare il AWS IAM-policy di proprietà utilizzata dal ruolo EMR Serverless collegato al servizio, in quanto contiene tutte le autorizzazioni necessarie di cui Serverless ha bisogno. EMR Tuttavia, è possibile modificare la descrizione del ruolo utilizzando. IAM

Per modificare la descrizione del ruolo AWSServiceRoleForAmazonEMRServerless collegato al servizio utilizzando IAM

Aggiungi la seguente dichiarazione alla politica delle autorizzazioni per l'IAMentità che deve modificare la descrizione di un ruolo collegato al servizio.

```
{
    "Effect": "Allow",
    "Action": [
        "iam: UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente. IAM

Eliminazione di un ruolo collegato al servizio per Serverless EMR

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 avrai un'entità inutilizzata che non viene monitorata o gestita attivamente. Tuttavia, è necessario eliminare tutte le applicazioni EMR Serverless in tutte le regioni prima di poter eliminare il ruolo collegato al servizio.

> **Note**
>
> Se il servizio EMR Serverless utilizza il ruolo quando si tenta di eliminare le risorse associate al ruolo, l'eliminazione potrebbe non riuscire. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Per eliminare il ruolo collegato al AWSServiceRoleForAmazonEMRServerless servizio utilizzando IAM

Aggiungi la seguente dichiarazione alla politica delle autorizzazioni per l'IAMentità che deve eliminare un ruolo collegato al servizio.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

Per eliminare manualmente il ruolo collegato al servizio utilizzando IAM

Usa la IAM console, AWS CLI, o AWS APIper eliminare il ruolo AWSServiceRoleForAmazonEMRServerless collegato al servizio. Per ulteriori informazioni, vedere Eliminazione di un ruolo collegato al servizio nella Guida per l'utente. IAM

Regioni supportate per i ruoli Serverless collegati ai servizi EMR

EMRServerless supporta l'utilizzo di ruoli collegati ai servizi in tutte le regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta AWS Regioni ed endpoint.

Ruoli Job Runtime per Amazon EMR Serverless

È possibile specificare le autorizzazioni di IAM ruolo che l'esecuzione di un job EMR Serverless può assumere quando si chiamano altri servizi per conto dell'utente. Ciò include l'accesso ad Amazon S3 per qualsiasi fonte di dati, destinazione e altro AWS risorse come i cluster Amazon Redshift e le tabelle DynamoDB. Per ulteriori informazioni su come creare un ruolo, consulta. Creare un ruolo di job runtime

Esempi di politiche di runtime

È possibile allegare una policy di runtime, come la seguente, a un ruolo di job runtime. La seguente politica di esecuzione dei processi consente di:

- Accesso in lettura ai bucket Amazon S3 con esempi. EMR
- Accesso completo ai bucket S3.
- Crea e leggi l'accesso a AWS Catalogo dati Glue.

Per aggiungere l'accesso ad altri AWS per risorse come DynamoDB, dovrai includere le relative autorizzazioni nella policy quando crei il ruolo di runtime.

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

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

Passa i privilegi di ruolo

Puoi allegare politiche di IAM autorizzazione al ruolo di un utente per consentire all'utente di assegnare solo ruoli approvati. Ciò consente agli amministratori di controllare quali utenti possono trasferire ruoli di job runtime specifici ai job EMR Serverless. Per ulteriori informazioni sull'impostazione delle autorizzazioni, vedere Concessione a un utente delle autorizzazioni per passare un ruolo a un AWS servizio.

Di seguito è riportato un esempio di policy che consente di passare un ruolo di job runtime al responsabile del servizio EMR Serverless.

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

Esempi di policy di accesso degli utenti per Serverless EMR

È possibile impostare politiche granulari per gli utenti in base alle azioni che si desidera che ciascun utente esegua durante l'interazione con le applicazioni Serverless. EMR I seguenti criteri sono esempi che potrebbero aiutare a configurare le autorizzazioni corrette per gli utenti. Questa sezione si concentra solo sulle politiche EMR Serverless. Per esempi di politiche utente di EMR Studio, consulta Configurare le autorizzazioni utente di EMR Studio. Per informazioni su come allegare le politiche agli IAM utenti (principali), consulta Gestione delle IAM politiche nella Guida per l'IAMutente.

Politica per i Power User

Per concedere tutte le azioni richieste per EMR Serverless, crea e allega una `AmazonEMRServerlessFullAccess` policy all'IAMutente, al ruolo o al gruppo richiesti.

Di seguito è riportato un esempio di policy che consente agli utenti esperti di creare e modificare applicazioni EMR Serverless, nonché di eseguire altre azioni come l'invio e il debug di job. Rivela tutte le azioni richieste da EMR Serverless per altri servizi.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CreateApplication",
```

```
                "emr-serverless:UpdateApplication",
                "emr-serverless:DeleteApplication",
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StopApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

Quando abiliti la connettività di rete alle tue VPC applicazioni EMR serverless creano interfacce di rete EC2 elastiche Amazon (ENIs) per comunicare con VPC le risorse. La seguente politica garantisce che tutte le nuove EC2 ENIs vengano create solo nel contesto di applicazioni EMR Serverless.

> **Note**
>
> Consigliamo vivamente di impostare questa politica per garantire che gli utenti non possano creare EC2 ENIs se non nel contesto del lancio di applicazioni EMR Serverless.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEC2ENICreationWithEMRTags",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
```

```
                }
            }
        }
}
```

Se desideri limitare l'accesso EMR Serverless a determinate sottoreti, puoi etichettare ogni sottorete con una condizione di tag. Questa IAM politica garantisce che le applicazioni EMR Serverless possano creare solo all'interno di sottoreti consentite. EC2 ENIs

```
{
    "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/KEY": "VALUE"
        }
    }
}
```

Important

Se sei un amministratore o un utente esperto che crea la tua prima applicazione, devi configurare le tue politiche di autorizzazione per consentirti di creare un ruolo EMR Serverless collegato ai servizi. Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Serverless EMR.

La seguente IAM politica ti consente di creare un ruolo collegato ai servizi EMR Serverless per il tuo account.

```
{
   "Sid":"AllowEMRServerlessServiceLinkedRoleCreation",
   "Effect":"Allow",
   "Action":"iam:CreateServiceLinkedRole",
```

```
   "Resource":"arn:aws:iam::account-id:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless"
}
```

Politica del tecnico dei dati

Di seguito è riportato un esempio di politica che consente agli utenti le autorizzazioni di sola lettura sulle applicazioni EMR Serverless, nonché la possibilità di inviare ed eseguire il debug dei lavori. Tieni presente che poiché questa policy non nega esplicitamente le operazioni, un'altra dichiarazione di policy può essere utilizzata per concedere l'accesso alle operazioni specificate.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

Utilizzo di tag per il controllo degli accessi

È possibile utilizzare le condizioni dei tag per un controllo granulare degli accessi. Ad esempio, puoi limitare gli utenti di un team in modo che possano inviare lavori solo alle applicazioni EMR Serverless contrassegnate con il nome del team.

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

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

Policy per il controllo degli accessi basato su tag

È possibile utilizzare le condizioni della policy basata sull'identità per controllare l'accesso alle applicazioni e le esecuzioni di processi in base ai tag.

Gli esempi seguenti mostrano diversi scenari e modi di utilizzare gli operatori di condizione con chiavi di condizione EMR Serverless. Queste dichiarazioni IAM politiche sono destinate esclusivamente a scopi dimostrativi e non devono essere utilizzate in ambienti di produzione. Esistono vari modi di combinare dichiarazioni di policy per concedere o negare autorizzazioni in base alle tue esigenze. Per ulteriori informazioni sulle IAM politiche di pianificazione e test, consulta la Guida per l'IAMutente.

> **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 illustrano le politiche di autorizzazione basate sull'identità utilizzate per controllare le azioni consentite con le applicazioni Serverless. EMR

Autorizzazione di operazioni solo su risorse con specifici valori di tag

Nel seguente esempio di policy, l'operatore `StringEquals` condition tenta di corrispondere al valore del `dev` reparto tag. Se il reparto tag non è stato aggiunto all'applicazione o non contiene il valore`dev`, la politica non si applica e le azioni non sono consentite da questa politica. Se nessun'altra dichiarazione politica consente le azioni, l'utente può lavorare solo con applicazioni che hanno questo tag con questo valore.

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

Puoi anche specificare più valori di tag utilizzando un operatore di condizione. Ad esempio, per consentire azioni sulle applicazioni in cui il `department` tag contiene il valore `dev` oppure`test`, è possibile sostituire il blocco delle condizioni nell'esempio precedente con il seguente.

```
"Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": ["dev", "test"]
        }
      }
```

Richiesta dell'assegnazione di tag alla creazione di una risorsa

Nell'esempio seguente, il tag deve essere applicato durante la creazione dell'applicazione.

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

La seguente dichiarazione politica consente a un utente di creare un'applicazione solo se l'applicazione ha un `department` tag, che può contenere qualsiasi valore.

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

Negazione dell'accesso per aggiungere ed eliminare tag

Questa politica impedisce a un utente di aggiungere o rimuovere tag su applicazioni EMR Serverless con un `department` tag il cui valore non `dev` è.

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

Esempi di policy basate sull'identità per Serverless EMR

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse Amazon EMR Serverless. Inoltre, non possono eseguire attività utilizzando il AWS Management Console, AWS Command Line Interface (AWS CLI), oppure AWS API. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un IAM amministratore può creare IAM politiche. L'amministratore può quindi aggiungere le IAM politiche ai ruoli e gli utenti possono assumerli.

Per informazioni su come creare una politica IAM basata sull'identità utilizzando questi documenti di esempioJSON, consulta Creazione di IAM politiche nella Guida per l'IAMutente.

Per dettagli sulle azioni e sui tipi di risorse definiti da Amazon EMR Serverless, incluso il formato di ARNs per ogni tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per Amazon EMR Serverless nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Consentire agli utenti di visualizzare le loro autorizzazioni

Best practice per le policy

> **Note**
>
> EMRServerless non supporta le policy gestite, quindi la prima pratica elencata di seguito non si applica.

Le policy basate sull'identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon EMR Serverless nel tuo account. Queste azioni possono comportare costi per Account AWS. Quando crei o modifichi politiche basate sull'identità, segui queste linee guida e consigli:

- Inizia con AWS politiche gestite e passaggio alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza il AWS politiche gestite che concedono autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo Account AWS. Si consiglia di ridurre ulteriormente le autorizzazioni definendo AWS politiche gestite dai clienti specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta AWS politiche gestite o AWS politiche gestite per le funzioni lavorative nella Guida per IAM l'utente.
- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le IAM politiche, concedi solo le autorizzazioni necessarie per eseguire un'attività. Puoi farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo per applicare le autorizzazioni, consulta Politiche e autorizzazioni nella Guida IAM per l'utente. IAM IAM
- Utilizza le condizioni nelle IAM politiche per limitare ulteriormente l'accesso: puoi aggiungere una condizione alle tue politiche per limitare l'accesso ad azioni e risorse. Ad esempio, puoi scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzandoSSL. È inoltre possibile utilizzare le condizioni per concedere l'accesso alle azioni di servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta Elementi IAM JSON della politica: Condizione nella Guida IAM per l'utente.
- Usa IAM Access Analyzer per convalidare IAM le tue policy e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio delle IAM policy () e alle best practice. JSON IAM IAMAccess Analyzer fornisce più di 100 controlli delle politiche e consigli pratici per aiutarti a creare policy sicure e funzionali.

Per ulteriori informazioni, vedere Convalida delle policy di IAM Access Analyzer nella Guida per l'utente. IAM

- Richiedi l'autenticazione a più fattori (MFA): se si dispone di uno scenario che richiede IAM utenti o un utente root nel Account AWS, attivala MFA per una maggiore sicurezza. Per richiedere MFA quando vengono richiamate API le operazioni, aggiungi MFA delle condizioni alle tue politiche. Per ulteriori informazioni, vedere Configurazione dell'APIaccesso MFA protetto nella Guida per l'IAMutente.

Per ulteriori informazioni sulle procedure consigliate inIAM, consulta la sezione Procedure consigliate in materia di sicurezza IAM nella Guida per l'IAMutente.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra come è possibile creare una politica che consenta IAM agli utenti di visualizzare le politiche in linea e gestite allegate alla loro identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando programmaticamente il AWS CLI oppure AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
```

```
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Aggiornamenti Amazon EMR Serverless a AWS policy gestite

Visualizza i dettagli sugli aggiornamenti di AWS politiche gestite per Amazon EMR Serverless da quando questo servizio ha iniziato a tracciare queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al RSS feed nella pagina della cronologia dei documenti di Amazon EMR Serverless.

Modifica	Descrizione	Data
A mazonEMRServerless ServiceRolePolicy — Aggiornamento a una policy esistente	Amazon EMR Serverless ha aggiunto la nuova `Sid` `CloudWatchPolicySt atement` e `EC2Policy Statement` alla mazonEMRServerless ServiceRolePolicy politica A.	25 gennaio 2024
A mazonEMRServerless ServiceRolePolicy — Aggiornamento a una politica esistente	Amazon EMR Serverless ha aggiunto nuove autorizzazioni per consentire ad Amazon EMR Serverless di pubblicar e parametri aggregati degli account per l'utilizzo di v CPU nello spazio dei nomi. `"AWS/ Usage"`	20 aprile 2023

Modifica	Descrizione	Data
Amazon EMR Serverless ha iniziato a tracciare le modifiche	Amazon EMR Serverless ha iniziato a tracciare le modifiche per il suo AWS politiche gestite.	20 aprile 2023

Risoluzione dei problemi di identità e accesso ad Amazon EMR Serverless

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con Amazon EMR Serverless e. IAM

Argomenti

- Non sono autorizzato a eseguire un'azione in Amazon EMR Serverless
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire l'accesso a persone esterne al mio AWS account per accedere alle mie risorse Amazon EMR Serverless

Non sono autorizzato a eseguire un'azione in Amazon EMR Serverless

Se il file AWS Management Console ti dice che non sei autorizzato a eseguire un'azione, quindi 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-serverless:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
serverless: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-serverless:`*`GetWidget`*.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue politiche devono essere aggiornate per consentirti di trasferire un ruolo ad Amazon EMR Serverless.

Medio Servizi AWS ti consentono di trasferire un ruolo esistente a quel servizio anziché 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 IAM utente denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in Amazon EMR Serverless. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare 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 AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire l'accesso a persone esterne al mio AWS account per accedere alle mie risorse Amazon EMR Serverless

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 policy per consentire alle persone di accedere alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon EMR Serverless supporta queste funzionalità, consultaIdentity and Access Management (IAM) in Amazon EMR Serverless.
- Per scoprire come fornire l'accesso alle tue risorse in tutto il mondo Account AWS di cui sei proprietario, vedi Fornire l'accesso a un IAM utente in un altro Account AWS che possiedi nella Guida per l'IAMutente.

- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, vedi Fornire l'accesso a Account AWS di proprietà di terzi nella Guida per l'IAMutente.
- 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'IAMutente.
- Per conoscere la differenza tra l'utilizzo di ruoli e politiche basate sulle risorse per l'accesso tra account diversi, consulta la sezione Accesso alle risorse tra account nella Guida per l'utente. IAM IAM

Utilizzo di EMR Serverless con AWS Lake Formation per un controllo granulare degli accessi

Panoramica

Con le EMR versioni 7.2.0 e successive di Amazon, puoi sfruttare AWS Lake Formation per applicare controlli di accesso dettagliati alle tabelle di Data Catalog supportate da S3. Questa funzionalità consente di configurare i controlli di accesso a livello di tabella, riga, colonna e cella per read domande all'interno dei tuoi job Amazon EMR Serverless Spark. Per configurare un controllo granulare degli accessi per i processi batch e le sessioni interattive di Apache Spark, usa Studio. EMR Consulta le sezioni seguenti per saperne di più su Lake Formation e su come usarlo con EMR Serverless.

Utilizzo di Amazon EMR Serverless con AWS Lake Formation comporta costi aggiuntivi. Per ulteriori informazioni, consulta i EMRprezzi di Amazon.

Come funziona EMR Serverless con AWS Lake Formation

L'utilizzo di EMR Serverless with Lake Formation ti consente di applicare un livello di autorizzazioni su ogni lavoro Spark per applicare il controllo delle autorizzazioni di Lake Formation quando Serverless esegue i lavori. EMR EMRServerless 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. Per ulteriori informazioni, consulta Cos'è AWS Lake Formatione Considerazioni e limitazioni.

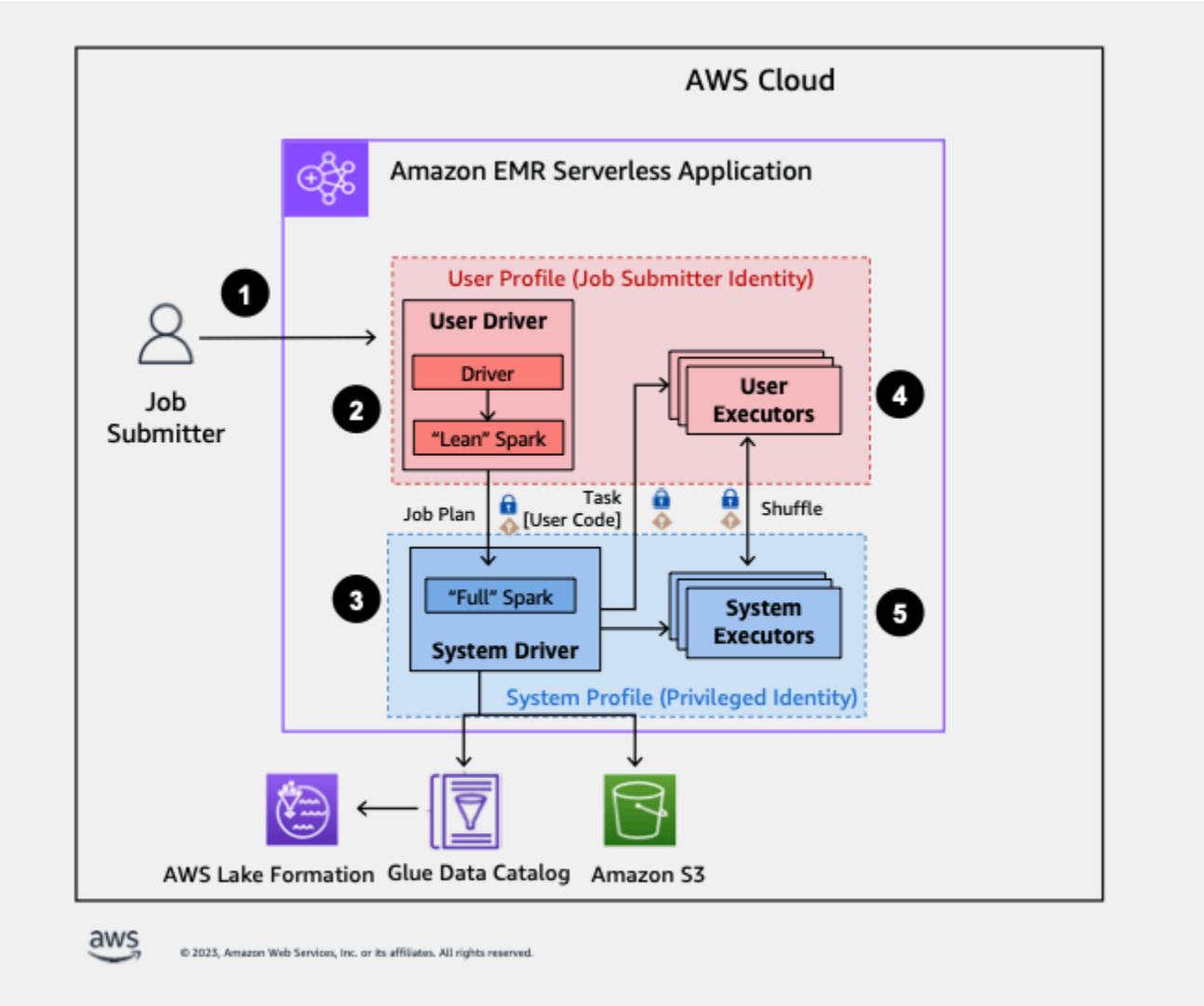
Quando utilizzi la capacità preinizializzata con Lake Formation, ti consigliamo di avere almeno due driver Spark. Ogni job abilitato per Lake Formation utilizza due driver Spark, uno per il profilo utente e uno per il profilo di sistema. Per ottenere prestazioni ottimali, è necessario utilizzare il doppio del numero di driver per i lavori abilitati a Lake Formation rispetto a chi non utilizza Lake Formation.

Quando esegui i job Spark su EMR Serverless, devi anche considerare l'impatto dell'allocazione dinamica sulla gestione delle risorse e sulle prestazioni del cluster. La configurazione `spark.dynamicAllocation.maxExecutors` del numero massimo di esecutori per profilo di risorsa si applica sia agli esecutori utente che a quelli di sistema. Se si configura tale numero in modo che sia uguale al numero massimo consentito di executor, l'esecuzione del job potrebbe bloccarsi a causa di un tipo di executor che utilizza tutte le risorse disponibili, impedendo all'altro executor di eseguire i job di job.

Per non esaurire le risorse, EMR Serverless imposta il numero massimo predefinito di esecutori per profilo di risorsa al 90% del valore. `spark.dynamicAllocation.maxExecutors` È possibile ignorare questa configurazione quando si specifica `spark.dynamicAllocation.maxExecutorsRatio` un valore compreso tra 0 e 1. Inoltre, è possibile configurare le seguenti proprietà per ottimizzare l'allocazione delle risorse e le prestazioni complessive:

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`
- `spark.cleaner.periodicGC.interval`

Di seguito è riportata una panoramica di alto livello su come EMR Serverless accede ai dati protetti dalle politiche di sicurezza di Lake Formation.



1. Un utente invia un job Spark a un AWS Lake Formation- applicazione Serverless abilitataEMR.
2. EMRServerless invia il lavoro a un driver utente ed esegue il processo nel profilo utente. Il driver utente esegue una versione snella di Spark che non è in grado di avviare attività, richiedere esecutori, accedere a S3 o al Glue Catalog. Costruisce un piano di lavoro.
3. EMRServerless configura un secondo driver chiamato driver di sistema e lo esegue nel profilo di sistema (con un'identità privilegiata). EMRServerless configura un TLS canale crittografato tra i due driver per la comunicazione. Il driver utente utilizza il canale per inviare i piani di lavoro al driver di sistema. Il driver di sistema non esegue il codice inviato dall'utente. Esegue Spark completo e comunica con 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. EMRServerless esegue quindi le fasi sugli executor con il driver utente o il driver di sistema. Il codice utente in qualsiasi fase viene eseguito esclusivamente sugli esecutori dei profili utente.

5. Fasi che leggono i dati dalle tabelle del Data Catalog protette da AWS Lake Formation oppure quelle che applicano filtri di sicurezza sono delegate agli esecutori di sistema.

Abilitare Lake Formation in Amazon EMR

Per abilitare Lake Formation, è necessario impostare `spark.emr-serverless.lakeformation.enabled` la `true` `spark-defaults` sottoclassificazione per il parametro di configurazione di runtime durante la creazione di un'applicazione EMR Serverless.

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

Puoi anche abilitare Lake Formation quando crei una nuova applicazione in EMR Studio. Scegli Use Lake Formation per un controllo granulare degli accessi, disponibile in Configurazioni aggiuntive.

La crittografia tra lavoratori è abilitata per impostazione predefinita quando si utilizza Lake Formation con EMR Serverless, quindi non è necessario abilitare nuovamente in modo esplicito la crittografia tra lavoratori.

Attivazione dei lavori di Lake Formation per Spark

Per abilitare Lake Formation per i singoli job Spark, imposta su `spark.emr-serverless.lakeformation.enabled` true durante l'utilizzo`spark-submit`.

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

IAMAutorizzazioni del ruolo Job Runtime

Le autorizzazioni di Lake Formation controllano l'accesso a AWS Le risorse di Glue Data Catalog, le sedi Amazon S3 e i dati sottostanti in tali sedi. IAMle autorizzazioni controllano l'accesso al Lake Formation e AWS APIsGlue e 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'IAMautorizzazione per l'`glue:Get*`APIoperazione.

Di seguito è riportato un esempio di politica su come fornire IAM le autorizzazioni per accedere a uno script in S3, caricando i log su S3, AWS Glue i API permessi e il permesso di accedere a Lake Formation.

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

```
            "Effect": "Allow",
            "Action": [
             "lakeformation:GetDataAccess"
             ],
            "Resource": ["*"]
        }
    ]
}
```

Configurazione delle autorizzazioni di Lake Formation per il ruolo Job Runtime

Innanzitutto, registra la posizione del tuo tavolo Hive con Lake Formation. Quindi crea le autorizzazioni per il tuo ruolo di job runtime nella tabella desiderata. Per maggiori dettagli su Lake Formation, vedi What is AWS Lake Formation? nel AWS Lake Formation Guida per gli sviluppatori.

Dopo aver configurato le autorizzazioni di Lake Formation, puoi inviare lavori Spark su Amazon EMR Serverless. Per ulteriori informazioni sui job Spark, consulta gli esempi di Spark.

Invio di un job run

Dopo aver completato la configurazione delle sovvenzioni Lake Formation, puoi inviare lavori Spark su EMR Serverless. Per eseguire i job Iceberg, devi fornire le seguenti proprietà. `spark-submit`

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=<S3_DATA_LOCATION>
--conf spark.sql.catalog.spark_catalog.glue.account-id=<ACCOUNT_ID>
--conf spark.sql.catalog.spark_catalog.client.region=<REGION>
--conf spark.sql.catalog.spark_catalog.glue.endpoint=https://
glue.<REGION>.amazonaws.com
```

Supporto per formati a tabella aperta

La EMR versione 7.2.0 di Amazon include il supporto per il controllo granulare degli accessi basato su Lake Formation. EMRServerless supporta i tipi di tabelle Hive e Iceberg. La tabella seguente descrive tutte le operazioni supportate.

Operazioni	Hive	Iceberg
DDLcomandi	Solo con autorizzazioni di IAM ruolo	Solo con autorizzazioni di IAM ruolo
Query incrementali	Non applicabile	Completamente supportato
Query temporali	Non applicabile a questo formato di tabella	Completamente supportato
Tabelle dei metadati	Non applicabile a questo formato di tabella	Supportato, ma alcune tabelle sono nascoste. Per ulteriori informazioni, consulta Considerazioni e limitazioni.
`DML INSERT`	Solo con IAM autorizzazioni	Solo con IAM autorizzazioni
DML UPDATE	Non applicabile a questo formato di tabella	Solo con IAM autorizzazioni
`DML DELETE`	Non applicabile a questo formato di tabella	Solo con IAM autorizzazioni
Operazioni di lettura	Completamente supportato	Completamente supportato
Stored procedure	Non applicabile	Supportato con le eccezioni di `register_table` `emigrate`. Per ulteriori informazioni, consulta Considerazioni e limitazioni.

Considerazioni e limitazioni

Considerate le seguenti considerazioni e limitazioni quando utilizzate Lake Formation with EMR Serverless.

> **Note**
>
> Quando abiliti Lake Formation per un job Spark su EMR Serverless, il job avvia un driver di sistema e un driver utente. Se hai specificato la capacità preinizializzata all'avvio, i driver forniti dalla capacità preinizializzata e il numero di driver di sistema è uguale al numero di driver utente specificato. Se si sceglie la capacità On Demand, EMR Serverless avvia un driver di sistema oltre a un driver utente. Per stimare i costi associati al tuo lavoro EMR Serverless with Lake Formation, utilizza il AWS Pricing Calculator.

Amazon EMR Serverless with Lake Formation è disponibile in tutte le regioni EMRServerless supportate, tranne AWS GovCloud (Stati Uniti orientali) e AWS GovCloud (Stati Uniti occidentali).

- Amazon EMR Serverless supporta il controllo granulare degli accessi tramite Lake Formation solo per le tabelle Apache Hive e Apache Iceberg. I formati Apache Hive includono Parquet e XSv. ORC
- Le applicazioni abilitate per Lake Formation non supportano l'utilizzo di immagini Serverless personalizzate EMR.
- Non puoi smettere di lavorare `DynamicResourceAllocation` per Lake Formation.
- Puoi usare Lake Formation solo con i job Spark.
- EMRServerless with Lake Formation supporta solo una singola sessione Spark per tutta la durata di un job.
- EMRServerless with Lake Formation supporta solo le query tabellari tra account condivise tramite collegamenti alle risorse.
- Quanto segue non è supportato:
  - Set di dati distribuiti resilienti () RDD
  - Streaming Spark
  - Scrivi con le autorizzazioni concesse da Lake Formation
  - Controllo degli accessi per le colonne annidate
- EMRServerless blocca le funzionalità che potrebbero compromettere il completo isolamento dei driver di sistema, tra cui:
  - UDTsiveUDFs, H e qualsiasi funzione definita dall'utente che includa classi personalizzate
  - Origini dati personalizzate
  - Fornitura di vasetti aggiuntivi per l'estensione, il connettore o il metastore Spark
  - Comando `ANALYZE TABLE`

- Per imporre controlli di accesso `EXPLAIN PLAN` e DDL operazioni come `DESCRIBE TABLE` non esporre informazioni riservate.
- EMRServerless limita l'accesso ai registri Spark dei driver di sistema sulle applicazioni abilitate per Lake Formation. Poiché il driver di sistema viene eseguito con più accesso, gli eventi e i log generati dal driver di sistema possono includere informazioni riservate. Per impedire a utenti o codici non autorizzati di accedere a questi dati sensibili, EMR Serverless ha disabilitato l'accesso ai registri dei driver di sistema. Per la risoluzione dei problemi, contatta AWS assistenza.
- 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'IAMautorizzazione per il ruolo EMR Serverless Job Runtime. Se configuri erroneamente il ruolo registrato con la posizione della tabella, i lavori inviati che utilizzano il ruolo con IAM autorizzazione S3 per la posizione della tabella avranno esito negativo.
- La scrittura su una tabella Lake Formation utilizza IAM l'autorizzazione anziché le autorizzazioni concesse da Lake Formation. Se il ruolo Job Runtime dispone delle autorizzazioni S3 necessarie, è possibile utilizzarlo per eseguire operazioni di scrittura.

Di seguito sono riportate considerazioni e limitazioni relative all'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 `partitionspath`, `esummaries`. Questa limitazione non si applica alle tabelle Iceberg che non sono registrate in Lake Formation.
- Le tabelle che non vengono registrate in Lake Formation supportano tutte le stored procedure Iceberg. Le `migrate` procedure `register_table` and non sono supportate per nessuna tabella.
- Ti consigliamo di utilizzare Iceberg DataFrameWriter V2 anziché V1.

Risoluzione dei problemi

Consulta le seguenti sezioni per le soluzioni di risoluzione dei problemi.

Registrazione

EMRServerless utilizza i profili di risorse Spark per suddividere l'esecuzione del lavoro. EMRServerless utilizza il profilo utente per eseguire il codice fornito, mentre il profilo di sistema applica le politiche di Lake Formation. È possibile accedere ai registri delle attività eseguite come profilo utente.

Interfaccia utente live e Spark History Server

L'interfaccia utente Live e lo Spark History Server contengono tutti gli eventi Spark generati dal profilo utente e gli eventi oscurati generati dal driver di sistema.

Puoi visualizzare tutte le attività dei driver utente e di sistema nella scheda Executors. Tuttavia, i link di registro sono disponibili solo per il profilo utente. Inoltre, alcune informazioni vengono cancellate dall'interfaccia utente Live, come il numero di record di output.

Job non riuscito con permessi insufficienti per Lake Formation

Assicurati che il tuo ruolo Job Runtime disponga delle autorizzazioni necessarie per l'esecuzione SELECT e sia presente DESCRIBE sulla tabella a cui stai accedendo.

Job con RDD esecuzione non riuscita

EMRServerless attualmente non supporta operazioni resilienti di set di dati distribuiti (RDD) su lavori abilitati per Lake Formation.

Impossibile accedere ai file di dati in Amazon S3

Assicurati di aver registrato la posizione del data lake in Lake Formation.

Eccezione di convalida della sicurezza

EMRServerless ha rilevato un errore di convalida della sicurezza. Contatti AWS supporto per l'assistenza.

Condivisione AWS Glue Data Catalog e tabelle tra gli account

Puoi 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 How do I share AWS Glue Data Catalog e tabelle su più account utilizzando AWS Lake Formation?.

Crittografia tra lavoratori

Con EMR le versioni 6.15.0 e successive di Amazon, puoi abilitare la comunicazione TLS crittografata reciproca tra i lavoratori durante le esecuzioni dei job Spark. Se abilitato, EMR Serverless genera e distribuisce automaticamente un certificato univoco per ogni lavoratore assegnato nell'ambito delle tue esecuzioni di lavoro. Quando questi lavoratori comunicano per scambiare messaggi di controllo o trasferire dati casuali, stabiliscono una TLS connessione reciproca e utilizzano i certificati configurati per verificare l'identità reciproca. Se un lavoratore non è in grado di verificare un altro certificato, l'TLShandshake fallisce e EMR Serverless interrompe la connessione tra i due.

Se utilizzi Lake Formation con EMR Serverless, la TLS crittografia reciproca è abilitata per impostazione predefinita.

Abilitazione della TLS crittografia reciproca su Serverless EMR

Per abilitare la TLS crittografia reciproca sulla tua applicazione Spark, imposta su true quando `spark.ssl.internode.enabled` crei un'applicazione EMR Serverless. Se stai usando il AWS console per creare un'applicazione EMR serverless, scegli Usa impostazioni personalizzate, quindi espandi Configurazione dell'applicazione e inserisci la tua`runtimeConfiguration`.

```
aws emr-serverless create-application \
--release-label emr-6.15.0 \
--runtime-configuration '{
  "classification": "spark-defaults",
  "properties": {"spark.ssl.internode.enabled": "true"}
}' \
--type "SPARK"
```

Se desideri abilitare la TLS crittografia reciproca per le singole esecuzioni di job Spark, imposta su `spark.ssl.internode.enabled` true durante l'utilizzo. `spark-submit`

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

Secrets Manager per la protezione dei dati con EMR Serverless

AWS Secrets Manager è un servizio di archiviazione segreto che è possibile utilizzare per proteggere le credenziali, le API chiavi e altre informazioni segrete del database. Quindi, nel codice, puoi

sostituire le credenziali hardcoded con una API chiamata a Secrets Manager. Questo aiuta a garantire che il segreto non possa essere compromesso da qualcuno che esamina il codice, perché il segreto non è presente. Per una panoramica, consulta il AWS Secrets Manager Guida per l'utente.

Secrets Manager crittografa i segreti utilizzando AWS Key Management Service chiavi. Per ulteriori informazioni, vedere Crittografia e decrittografia segrete nel AWS Secrets Manager Guida per l'utente.

È possibile configurare Secrets Manager in modo che ruoti automaticamente i segreti in base a una pianificazione specificata. In questo modo puoi sostituire i segreti a lungo termine con altri a breve termine, contribuendo a ridurre notevolmente il rischio di compromissione. Per ulteriori informazioni, consulta Rotazione AWS Secrets Manager segreti nel AWS Secrets Manager Guida per l'utente.

Amazon EMR Serverless si integra con AWS Secrets Manager in modo da poter archiviare i dati in Secrets Manager e utilizzare l'ID segreto nelle configurazioni.

In che modo EMR Serverless utilizza i segreti

Quando si archiviano i dati in Secrets Manager e si utilizza l'ID segreto nelle configurazioni per EMR Serverless, non si passano dati di configurazione sensibili a EMR Serverless in testo semplice per poi esporli a fonti esterne. APIs Se indichi che una coppia chiave-valore contiene un ID segreto per un segreto archiviato in Secrets Manager, EMR Serverless recupera il segreto quando invia i dati di configurazione ai worker per l'esecuzione dei job.

Per indicare che una coppia chiave-valore per una configurazione contiene un riferimento a un segreto archiviato in Secrets Manager, aggiungete l'`EMR.secret@`annotazione al valore di configurazione. Per qualsiasi proprietà di configurazione con annotazione ID segreta, EMR Serverless chiama Secrets Manager e risolve il segreto al momento dell'esecuzione del lavoro.

Come creare un segreto

Per creare un segreto, segui i passaggi in Creare un AWS Secrets Manager segreto nel AWS Secrets Manager Guida per l'utente. Nel passaggio 3, scegli il campo Testo normale per inserire il tuo valore sensibile.

Fornisci un segreto in una classificazione di configurazione

I seguenti esempi mostrano come fornire un segreto in una classificazione di configurazione in`StartJobRun`. Se si desidera configurare le classificazioni per Secrets Manager a livello di applicazione, vedereConfigurazione predefinita dell'applicazione per Serverless EMR.

Negli esempi, *SecretName* sostituiscilo con il nome del segreto da recuperare. Includi il trattino seguito dai sei caratteri che Secrets Manager aggiunge alla fine del segretoARN. Per ulteriori informazioni, consulta Come creare un segreto.

In questa sezione

Specificate i riferimenti segreti - Spark

Example — Specificate i riferimenti segreti nella configurazione del metastore Hive esterno per Spark

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

Example — Specificate i riferimenti segreti per la configurazione del metastore Hive esterno nella classificazione **`spark-defaults`**

```
{
        "classification": "spark-defaults",
        "properties": {

 "spark.hadoop.javax.jdo.option.ConnectionDriverName":"org.mariadb.jdbc.Driver"
            "spark.hadoop.javax.jdo.option.ConnectionURL":"jdbc:mysql://db-host:db-
port/db-name"
            "spark.hadoop.javax.jdo.option.ConnectionUserName":"connection-user-name"
            "spark.hadoop.javax.jdo.option.ConnectionPassword":
 "EMR.secret@SecretName",
        }
    }
```

Specificare i riferimenti segreti - Hive

Example — Specificate i riferimenti segreti nella configurazione del metastore Hive esterno per Hive

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
    --job-driver '{
        "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/
emr-serverless-hive/hive/warehouse
                    --hiveconf javax.jdo.option.ConnectionUserName=username
                    --hiveconf
 javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
                    --hiveconf
 hive.metastore.client.factory.class=org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreCli
                    --hiveconf
 javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
                    --hiveconf javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
```

```
        "s3MonitoringConfiguration": {
            "logUri": "s3://EXAMPLE-LOG-BUCKET"
        }
    }
}'
```

Example — Specificare i riferimenti segreti per la configurazione del metastore Hive esterno nella classificazione **`hive-site`**

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

Concedi l'accesso a EMR Serverless per recuperare il segreto

Per consentire a EMR Serverless di recuperare il valore segreto da Secrets Manager, aggiungi la seguente dichiarazione di policy al tuo segreto quando lo crei. È necessario creare il segreto con la KMS chiave gestita dal cliente affinché EMR Serverless possa leggere il valore segreto. Per ulteriori informazioni, consulta Autorizzazioni per la chiave nel KMS AWS Secrets Manager Guida per l'utente.

Nella seguente politica, sostituiscilo *`applicationId`* con l'ID della tua applicazione.

Politica delle risorse per il segreto

È necessario includere le seguenti autorizzazioni nella politica delle risorse per il segreto in AWS Secrets Manager per consentire a EMR Serverless di recuperare valori segreti. Per garantire che solo un'applicazione specifica possa recuperare questo segreto, puoi facoltativamente specificare l'ID dell'applicazione EMR Serverless come condizione nella policy.

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

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

Crea il tuo segreto con la seguente politica gestita dal cliente AWS Key Management Service (AWS KMS) chiave:

Politica per la gestione da parte del cliente AWS KMS chiave

```
{
    "Sid": "Allow EMR Serverless to use the key for decrypting secrets",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "emr-serverless.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
```

```
        "StringEquals": {
            "kms:ViaService": "secretsmanager.Regione AWS.amazonaws.com"
        }
    }
}
```

Ruotare il segreto

La rotazione avviene quando si aggiorna periodicamente un segreto. È possibile configurare AWS Secrets Manager per ruotare automaticamente il segreto in base a una pianificazione specificata dall'utente. In questo modo, puoi sostituire i segreti a lungo termine con quelli a breve termine. Questo aiuta a ridurre il rischio di compromessi. EMRServerless recupera il valore segreto da una configurazione annotata quando il processo passa allo stato di esecuzione. Se tu o un processo aggiorni il valore segreto in Secrets Manager, devi inviare un nuovo lavoro in modo che il lavoro possa recuperare il valore aggiornato.

> **Note**
>
> I lavori che sono già in esecuzione non possono recuperare un valore segreto aggiornato. Ciò potrebbe causare un fallimento del lavoro.

Utilizzo di Amazon S3 Access Grants con Serverless EMR

Panoramica di S3 Access Grants per Serverless EMR

Con le EMR versioni 6.15.0 e successive di Amazon, Amazon S3 Access Grants offre una soluzione di controllo degli accessi scalabile che puoi utilizzare per aumentare l'accesso ai tuoi dati Amazon S3 da Serverless. EMR 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.

Usa S3 Access Grants per aumentare l'accesso ai dati di Amazon S3 oltre alle autorizzazioni concesse dal ruolo di runtime o ai IAM ruoli associati alle identità con accesso alla tua applicazione Serverless. EMR

Per ulteriori informazioni, consulta Managing access with S3 Access Grants for Amazon EMR nella Amazon EMR Management Guide e Managing access with S3 Access Grants nella Amazon Simple Storage Service User Guide.

Questa sezione descrive come avviare un'applicazione EMR Serverless che utilizza S3 Access Grants per fornire l'accesso ai dati in Amazon S3. Per i passaggi per utilizzare S3 Access Grants con altre EMR distribuzioni Amazon, consulta la seguente documentazione:

- Utilizzo di S3 Access Grants con Amazon EMR
- Utilizzo di S3 Access Grants con Amazon su EMR EKS

Avvia un'applicazione EMR serverless con S3 Access Grants per la gestione dei dati

Puoi abilitare S3 Access Grants su EMR Serverless e avviare un'applicazione 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. Imposta un ruolo di esecuzione del lavoro per la tua applicazione Serverless. EMR Includi IAM le autorizzazioni necessarie per eseguire i job Spark e utilizzare S3 Access Grants e: `s3:GetDataAccess` `s3:GetAccessGrantsInstanceForPrefix`

```
{
    "Effect": "Allow",
    "Action": [
    "s3:GetDataAccess",
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
}
```

> **Note**
>
> Se specifichi IAM ruoli per l'esecuzione del lavoro che dispongono di autorizzazioni aggiuntive per accedere direttamente a S3, gli utenti potranno accedere ai dati consentiti dal ruolo anche se non dispongono dell'autorizzazione di S3 Access Grants.

2. Avvia la tua applicazione EMR Serverless con un'etichetta di EMR release Amazon 6.15.0 o successiva e la `spark-defaults` classificazione, come illustrato nell'esempio seguente. Sostituisci i valori in *`red text`* con i valori appropriati per il tuo scenario di utilizzo.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/
wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g
 --conf spark.executor.instances=1"
        }
    }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.hadoop.fs.s3.s3AccessGrants.enabled": "true",
          "spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
      }]
}'
```

Considerazioni su S3 Access Grants con Serverless EMR

Per informazioni importanti su supporto, compatibilità e comportamento quando usi Amazon S3 Access Grants EMR con Serverless, consulta le considerazioni su S3 Access Grants con Amazon nella Amazon EMR Management Guide. EMR

Registrazione delle chiamate Amazon EMR Serverless tramite API AWS CloudTrail

Amazon EMR Serverless è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in modalità EMR Serverless. CloudTrail

acquisisce tutte le API chiamate per EMR Serverless come eventi. Le chiamate acquisite includono chiamate dalla console EMR Serverless e chiamate in codice alle operazioni Serverless. EMR API Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Serverless. EMR Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a EMR Serverless, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e ulteriori dettagli.

Per ulteriori informazioni CloudTrail, consulta la AWS CloudTrail Guida per l'utente.

EMRInformazioni serverless in CloudTrail

CloudTrail è abilitato sul tuo Account AWS quando crei l'account. Quando l'attività si verifica in EMR Serverless, tale attività viene registrata in un CloudTrail evento insieme ad altri AWS eventi di servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare gli eventi recenti nel tuo Account AWS. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo Account AWS, inclusi gli eventi per EMR Serverless, 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 tutti Regioni AWS. Il percorso registra gli eventi di tutte le regioni del AWS esegue il partizionamento e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurarne altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei CloudTrail log. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un percorso
- CloudTrail servizi e integrazioni supportati
- Configurazione delle SNS notifiche Amazon per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte le azioni EMR Serverless vengono registrate CloudTrail e documentate nel Serverless Reference. EMR API Ad esempio, le chiamate a `StartJobRun` e le `CreateApplication` `CancelJobRun` 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 root o AWS Identity and Access Management (IAM) credenziali utente.
- 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, vedi l'CloudTrail userIdentity elemento.

Informazioni sulle EMR voci dei file di registro serverless

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 stack ordinata delle API chiamate pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'`CreateApplication`azione.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
        "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
        "accountId": "012345678910",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::012345678910:role/Admin",
                "accountId": "012345678910",
                "userName": "Admin"
            },
            "webIdFederationData": {},
```

```
            "attributes": {
                "creationDate": "2022-06-01T23:46:52Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-06-01T23:49:28Z",
    "eventSource": "emr-serverless.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.26.10",
    "requestParameters": {
        "name": "my-serverless-application",
        "releaseLabel": "emr-6.6",
        "type": "SPARK",
        "clientToken": "0a1b234c-de56-7890-1234-567890123456"
    },
    "responseElements": {
        "name": "my-serverless-application",
        "applicationId": "1234567890abcdef0",
        "arn": "arn:aws:emr-serverless:us-west-2:555555555555:/
applications/1234567890abcdef0"
    },
    "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
    "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "012345678910",
    "eventCategory": "Management"
}
```


Convalida della conformità per Amazon Serverless EMR

La sicurezza e la conformità di EMR Serverless vengono valutate da revisori di terze parti come parte di più AWS programmi di conformità, inclusi i seguenti:

- Controlli di sistema e organizzazione (SOC)
- Standard di sicurezza dei dati del settore delle carte di pagamento (PCIDSS)
- Programma federale di gestione dei rischi e delle autorizzazioni (FedRAMP) Moderato

- Legge sulla portabilità e la responsabilità delle assicurazioni sanitarie () HIPAA

AWS fornisce un elenco aggiornato di frequente di AWS servizi nell'ambito di specifici programmi di conformità presso AWS Servizi rientranti nell'ambito del programma di conformità.

I report di audit di terze parti possono essere scaricati utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artefatto.

Per ulteriori informazioni sull' AWS programmi di conformità, vedere AWS Programmi di conformità.

La responsabilità di conformità quando si utilizza EMR Serverless è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'organizzazione e dalle leggi e dai regolamenti applicabili. Se l'uso di EMR Serverless è soggetto alla conformità a standard come Fed PCI RAMP ModerateHIPAA, AWS fornisce risorse per aiutare a:

- Guide introduttive su sicurezza e conformità che illustrano le considerazioni e i passaggi architetturali per l'implementazione di ambienti di base incentrati sulla sicurezza e la conformità su AWS.
- AWS Le guide alla conformità dei clienti possono aiutarti a comprendere il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization ()). ISO
- AWS Config è utile per valutare il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti del settore.
- AWS Compliance Resources è una raccolta di cartelle di lavoro e guide applicabili al tuo settore e alla tua località.
- AWS Security Hub ti offre una visione completa del tuo stato di sicurezza all'interno AWS e ti aiuta a verificare la tua conformità agli standard e alle migliori pratiche del settore della sicurezza.
- AWS Audit Manager— questo Servizio AWS ti aiuta a controllare continuamente i tuoi AWS utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

Resilienza in Amazon Serverless EMR

Il AWS l'infrastruttura globale è costruita attorno AWS Regioni e zone di disponibilità. AWS Le Regioni forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite

reti altamente ridondanti, a bassa latenza e velocità di trasmissione effettiva elevata. 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 sull' AWS Regioni e zone di disponibilità, vedi AWS Infrastruttura globale.

Oltre al AWS infrastruttura globale, Amazon EMR Serverless offre l'integrazione con Amazon S3 EMRFS fino a supportare le tue esigenze di resilienza e backup dei dati.

Sicurezza dell'infrastruttura in Amazon EMR Serverless

In quanto servizio gestito, Amazon EMR è protetto da AWS sicurezza di rete globale. Per informazioni su AWS servizi di sicurezza e come AWS protegge l'infrastruttura, vedi AWS Sicurezza nel cloud. Per progettare il tuo AWS ambiente che utilizza le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Framework ben architettato.

Tu usi AWS APIchiamate pubblicate per accedere ad Amazon EMR tramite la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). Richiediamo TLS 1.2 e consigliamo TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS) come (Ephemeral Diffie-Hellman) o DHE (Elliptic Curve Ephemeral Diffie-Hellman). ECDHE La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale. IAM Oppure puoi usare il AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per firmare le richieste.

Analisi della configurazione e delle vulnerabilità in Amazon Serverless EMR

AWS gestisce le 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 Serverless EMR

- Modello di responsabilità condivisa
- Amazon Web Services: panoramica dei processi di sicurezza (whitepaper)

Endpoint e quote per EMR Serverless

Endpoint di servizio

Per connettersi a livello di codice a un Servizio AWS, si utilizza un endpoint. Un endpoint è il punto URL di ingresso per un AWS servizio web. Oltre allo standard AWS punti finali, alcuni Servizi AWS offrono FIPS endpoint in regioni selezionate. La tabella seguente elenca gli endpoint di servizio per EMR Serverless. Per ulteriori informazioni, consulta Servizio AWS endpoint.

Nome Regione	Regione	Endpoint	Protocollo
Stati Uniti orientali (Ohio)	`us-east-2` (limitato alle seguenti zone di disponibilità: `use2-az1use2-az2`, `euse2-az3`)	`emr-serverless.us-east-2.amazonaws.com`	HTTPS

Stati Uniti orientali (Virginia settentrionale)	`us-east-1` (limitato alle seguenti zone di disponibilità: `use1-az1use1-az2`,`use1-az4`,`use1-az5`, `euse1-az6`)	`emr-serverless.us-east-1.amazonaws.com`
		
		`emr-serverless-fips.us-east-1.amazonaws.com`
			HTTPS

Stati Uniti occidentali (California settentrionale)	`us-west-1`	`emr-serverless.us-west-1.amazonaws.com`	HTTPS
Stati Uniti occidentali (Oregon)	`us-west-2`	`emr-serverless.us-west-2.amazonaws.com`	HTTPS

Nome Regione	Regione	Endpoint	Protocollo
		emr-serverless-fips.us-west-2.amazonaws.com	
Africa (Città del Capo)	af-south-1	emr-serverless.af-south-1.amazonaws.com	HTTPS
Asia Pacifico (Hong Kong)	ap-east-1	emr-serverless.ap-east-1.amazonaws.com	HTTPS
Asia Pacifico (Giacarta)	ap-southeast-3	emr-serverless.ap-southeast-3.amazonaws.com	HTTPS
Asia Pacifico (Mumbai)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
Asia Pacific (Osaka)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS

Nome Regione	Regione	Endpoint	Protocollo
Asia Pacific (Seul)	ap-northeast-2	emr-serverless.ap-northeast-2.amazonaws.com	HTTPS
Asia Pacifico (Singapore)	ap-southeast-1	emr-serverless.ap-southeast-1.amazonaws.com	HTTPS
Asia Pacifico (Sydney)	ap-southeast-2	emr-serverless.ap-southeast-2.amazonaws.com	HTTPS
Asia Pacifico (Tokyo)	ap-northeast-1	emr-serverless.ap-northeast-1.amazonaws.com	HTTPS
Canada (Centrale)	ca-central-1 (limitato alle seguenti zone di disponibilità: cac1-az1 ecac1-az2)	emr-serverless.ca-central-1.amazonaws.com	HTTPS
Europa (Francoforte)	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS

Nome Regione	Regione	Endpoint	Protocollo
Europa (Irlanda)	eu-west-1	emr-serverless.eu-west-1.amazonaws.com	HTTPS
Europa (Londra)	eu-west-2	emr-serverless.eu-west-2.amazonaws.com	HTTPS
Europa (Milano)	eu-south-1	emr-serverless.eu-south-1.amazonaws.com	HTTPS
Europa (Parigi)	eu-west-3	emr-serverless.eu-west-3.amazonaws.com	HTTPS
Europa (Spagna)	eu-south-2	emr-serverless.eu-south-2.amazonaws.com	HTTPS
Europa (Stoccolma)	eu-north-1	emr-serverless.eu-north-1.amazonaws.com	HTTPS
Medio Oriente (Bahrein)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS

Nome Regione	Regione	Endpoint	Protocollo
Medio Oriente (UAE)	`me-central-1`	`emr-serverless.me-central-1.amazonaws.com`	HTTPS
Sud America (San Paolo)	`sa-east-1`	`emr-serverless.sa-east-1.amazonaws.com`	HTTPS
AWS GovCloud (Stati Uniti orientali)	`us-gov-east-1`	`emr-serverless.us-gov-east-1.amazonaws.com`	HTTPS
AWS GovCloud (Stati Uniti occidentali)	`us-gov-west-1`	`emr-serverless.us-gov-west-1.amazonaws.com`	HTTPS

Quote del servizio

Le quote di servizio, note anche come limiti, sono il numero massimo di risorse o operazioni di servizio consentite Account AWS può usare. EMRServerless raccoglie i parametri di utilizzo delle quote di servizio ogni minuto e li pubblica nel namespace. `AWS/Usage`

> **Note**
>
> Novità AWS gli account potrebbero avere quote iniziali inferiori che possono aumentare nel tempo. Amazon EMR Serverless monitora l'utilizzo degli account all'interno di ciascuno Regione AWS, quindi aumenta automaticamente le quote in base all'utilizzo.

Nella tabella seguente sono elencate le quote di servizio per EMR Serverless. Per ulteriori informazioni, consulta Servizio AWS quote.

Nome	Limite predefinito	Modificabile?	Descrizione
Numero massimo di concorrenti vCPUs per account	16	Sì	Il numero massimo di file vCPUs che possono essere eseguiti contempor aneamente per l'account nella versione corrente Regione AWS.

APIlimiti

Di seguito vengono descritti i API limiti per regione per la tua Account AWS.

Risorsa	Quota predefinita
ListApplications	10 transazioni al secondo. Burst di 50 transazio ni al secondo.
CreateApplication	1 transazione al secondo. Una raffica di 25 transazioni al secondo.
DeleteApplication	1 transazione al secondo. Una raffica di 25 transazioni al secondo.
GetApplication	10 transazioni al secondo. Burst di 50 transazio ni al secondo.
UpdateApplication	1 transazione al secondo. Una raffica di 25 transazioni al secondo.
ListJobRuns	1 transazione al secondo. Una raffica di 25 transazioni al secondo.
StartJobRun	1 transazione al secondo. Una raffica di 25 transazioni al secondo.

Risorsa	Quota predefinita
GetDashboardForJobRun	1 transazione al secondo. Una raffica di 2 transazioni al secondo.
CancelJobRun	1 transazione al secondo. Una raffica di 25 transazioni al secondo.
GetJobRun	10 transazioni al secondo. Burst di 50 transazioni al secondo.
StartApplication	1 transazione al secondo. Una raffica di 25 transazioni al secondo.
StopApplication	1 transazione al secondo. Una raffica di 25 transazioni al secondo.

Altre considerazioni

L'elenco seguente contiene altre considerazioni relative a Serverless. EMR

- EMRServerless è disponibile nelle seguenti versioni Regioni AWS:
  - Stati Uniti orientali (Ohio)
  - Stati Uniti orientali (Virginia settentrionale)
  - Stati Uniti occidentali (California settentrionale)
  - Stati Uniti occidentali (Oregon)
  - Africa (Città del Capo)
  - Asia Pacifico (Hong Kong)
  - Asia Pacifico (Giacarta)
  - Asia Pacifico (Mumbai)
  - Asia Pacific (Osaka)
  - Asia Pacific (Seul)
  - Asia Pacifico (Singapore)
  - Asia Pacifico (Sydney)
  - Asia Pacifico (Tokyo)
  - Canada (Centrale)
  - Europa (Francoforte)
  - Europa (Irlanda)
  - Europa (Londra)
  - Europa (Milano)
  - Europa (Parigi)
  - Europa (Spagna)
  - Europa (Stoccolma)
  - Medio Oriente (Bahrein)
  - Medio Oriente () UAE
  - Sud America (San Paolo)
  - AWS GovCloud (Stati Uniti orientali)
  - AWS GovCloud (Stati Uniti occidentali)

Per un elenco degli endpoint associati a queste regioni, vedere. Endpoint di servizio

- Il timeout predefinito per l'esecuzione di un processo è di 12 ore. È possibile modificare questa impostazione con la `executionTimeoutMinutes` proprietà in `startJobRun` API o AWS SDK. È possibile `executionTimeoutMinutes` impostare su 0 se si desidera che l'esecuzione del processo non si esaurisca mai. Ad esempio, se si dispone di un'applicazione di streaming, è possibile `executionTimeoutMinutes` impostare su 0 per consentire l'esecuzione continua del processo di streaming.
- La `billedResourceUtilization` proprietà in `getJobRun` API mostra l'aggregato vCPU, la memoria e lo storage che AWS ha fatturato l'esecuzione del lavoro. Le risorse fatturate includono un utilizzo minimo di 1 minuto per i lavoratori, oltre a uno storage aggiuntivo di oltre 20 GB per lavoratore. Queste risorse non includono l'utilizzo per lavoratori inattivi preinizializzati.
- Senza VPC connettività, un job può accedervi Servizio AWS endpoint nello stesso Regione AWS. Questi servizi includono Amazon S3, AWS Glue, Amazon CloudWatch Logs, AWS KMS, AWS Security Token Service, Amazon DynamoDB e AWS Secrets Manager. È possibile abilitare la VPC connettività per accedere ad altri Servizi AWS attraverso AWS PrivateLink, ma non sei obbligato a farlo. Per accedere ai servizi esterni, puoi creare la tua applicazione con unVPC.
- EMRServerless non supportaHDFS. I dischi locali dei worker sono lo storage temporale che EMR Serverless utilizza per mescolare ed elaborare i dati durante l'esecuzione dei processi.
- EMRServerless non supporta l'esistente. emr-dynamodb-connector

Versioni di rilascio di Amazon EMR Serverless

Una EMR versione di Amazon è un insieme di applicazioni open source dell'ecosistema dei big data. Ogni versione include applicazioni, componenti e funzionalità per big data che scegli per far distribuire e configurare Amazon EMR Serverless durante l'esecuzione del tuo lavoro.

Con Amazon EMR 6.6.0 e versioni successive, puoi distribuire EMR Serverless. Questa opzione di distribuzione non è disponibile con le EMR versioni precedenti di Amazon. Quando invii il tuo lavoro, devi specificare una delle seguenti versioni supportate.

Argomenti

EMR Serverless 7.2.0

La tabella seguente elenca le versioni dell'applicazione disponibili con EMR Serverless 7.2.0.

Applicazione	Versione
Apache Spark	3.5.1

Applicazione	Versione
Apache Hive	3.1.3
Apache Tez	0,10,2

EMRNote di rilascio Serverless 7.2.0

- Lake Formation con EMR Serverless: ora puoi usare AWS Lake Formation per applicare controlli di accesso dettagliati alle tabelle di Data Catalog supportate da 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 dei job Serverless Spark. EMR Per ulteriori informazioni, consulta the section called “Lake Formation per FGAC” e the section called “Considerazioni”.

EMR Serverless 7.1.0

La tabella seguente elenca le versioni dell'applicazione disponibili con EMR Serverless 7.1.0.

Applicazione	Versione
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	010.2

EMR Serverless 7,0,0

La tabella seguente elenca le versioni dell'applicazione disponibili con EMR Serverless 7.0.0.

Applicazione	Versione
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	010.2

EMR Serverless 6,15,0

La tabella seguente elenca le versioni dell'applicazione disponibili con EMR Serverless 6.15.0.

Applicazione	Versione
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	010.2

EMRNote di rilascio di Serverless 6.15.0

- TLSsupporto — Con le versioni 6.15.0 e successive di Amazon EMR Serverless, puoi abilitare la comunicazione TLS criptata reciproca tra i lavoratori che eseguono i job di Spark. Se abilitato, EMR Serverless genera automaticamente un certificato univoco per ogni lavoratore, fornito nell'ambito di un job run, che i lavoratori utilizzano durante l'TLShandshake per autenticarsi a vicenda e stabilire un canale crittografato per elaborare i dati in modo sicuro. Per ulteriori informazioni sulla crittografia reciproca, vedere TLS Crittografia tra lavoratori.

EMR Serverless 6.14.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.14.0.

Applicazione	Versione
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	010.2

EMR Serverless 6,13,0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.13.0.

Applicazione	Versione
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	010.2

EMR Serverless 6,12,0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.12.0.

Applicazione	Versione
Apache Spark	3.4.0
Apache Hive	3.1.3
Apache Tez	010.2

EMR Serverless 6.11,0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.11.0.

Applicazione	Versione
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	010.2

EMRNote di rilascio di Serverless 6.11.0

- Accedi alle risorse S3 da altri account: con le versioni 6.11.0 e successive, puoi configurare più IAM ruoli da assumere quando accedi a bucket Amazon S3 in diversi modi AWS account di Serverless. EMR

EMR Serverless 6.10.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.10.0.

Applicazione	Versione
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	010.2

EMRNote di rilascio di Serverless 6.10.0

- Per le applicazioni EMR serverless con versione 6.10.0 o successiva, il valore predefinito per la proprietà è. `spark.dynamicAllocation.maxExecutors` `infinity` Le versioni precedenti hanno come impostazione predefinita. `100` Per ulteriori informazioni, consulta Proprietà del lavoro Spark.

EMR Serverless 6.9.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.9.0.

Applicazione	Versione
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	010.2

EMRNote di rilascio di Serverless 6.9.0

- L'integrazione di Amazon Redshift per Apache Spark è inclusa nelle EMR versioni di Amazon 6.9.0 e successive. 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 Utilizzo dell'integrazione di Amazon Redshift per Apache Spark su Amazon Serverless EMR.

- EMRLa versione Serverless 6.9.0 aggiunge il supporto per AWS Architettura Graviton2 (arm64). È possibile utilizzare il `architecture` parametro per `create-application` e per scegliere l'architettura `update-application` APIs arm64. Per ulteriori informazioni, consulta Opzioni di architettura Amazon EMR Serverless.
- Ora puoi esportare, importare, interrogare e unire tabelle Amazon DynamoDB direttamente dalle EMR tue applicazioni Serverless Spark e Hive. Per ulteriori informazioni, consulta Connessione a DynamoDB con Amazon Serverless EMR.

Problemi noti

- Se utilizzi l'integrazione di Amazon Redshift per Apache Spark e disponi di un'indicazione temporale time, timez, timestamp o timestamptz con una precisione di microsecondi 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 Serverless 6.8.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.8.0.

Applicazione	Versione
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless 6.7.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.7.0.

Applicazione	Versione
Apache Spark	3.2.1
Apache Hive	3.1.3
Apache Tez	0.9.2

Modifiche, miglioramenti e problemi risolti specifici del motore

La tabella seguente elenca una nuova funzionalità specifica del motore.

Modifica	Descrizione
Funzionalità	Tez scheduler ora supporta la priorità dell'atti vità Tez anziché la priorità del contenitore

EMR Serverless 6.6.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.6.0.

Applicazione	Versione
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

EMRNote di rilascio iniziali di Serverless

- EMRServerless supporta la classificazione della configurazione Spark. `spark-defaults` Questa classificazione modifica i valori nel file di Spark. `spark-defaults.conf` XML Le classificazioni di configurazione consentono di personalizzare le applicazioni. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

- EMRServerless supporta le classificazioni di configurazione Hive`hive-site`,, `tez-site` e. `emrfs-site` `core-site` Questa classificazione può modificare rispettivamente i valori nel `hive-site.xml` file di Hive, nel `tez-site.xml` file di Tez, EMRFS nelle impostazioni EMR di Amazon o nel file di `core-site.xml` Hadoop. Le classificazioni di configurazione consentono di personalizzare le applicazioni. Per ulteriori informazioni, consulta la sezione Configurazione delle applicazioni.

Modifiche, miglioramenti e problemi risolti specifici del motore

- La tabella seguente elenca i backport di Hive e Tez.

  Modifiche a Hive e Tez

Modifica	Descrizione
Backport	TEZ-4430: problema risolto con la proprietà `tez.task.launch.cmd-opts`
Backport	HIVE-25971: Risolti i ritardi di chiusura delle attività Tez dovuti all'apertura del pool di thread memorizzato nella cache

Cronologia dei documenti

La tabella seguente descrive le modifiche importanti alla documentazione dall'ultima versione di EMR Serverless. Per ulteriori informazioni sugli aggiornamenti di questa documentazione, puoi iscriverti a un RSS feed.

Modifica	Descrizione	Data
Nuova versione	EMR Serverless 7.2.0	25 luglio 2024
Nuova versione	EMR Serverless 7.1.0	17 aprile 2024
Aggiornamento a una politica esistente.	È stata aggiunta la nuova politica `Sid` `CloudWatchPolicyStatement` e `EC2PolicyStatement` alla mazonEMRServerless ServiceRolePolicy politica A.	25 gennaio 2024
Nuova versione	EMR Serverless 7,0,0	29 dicembre 2023
Nuova versione	EMR Serverless 6,15,0	17 novembre 2023
Nuova caratteristica	Configura più IAM ruoli da assumere quando accedi ai bucket Amazon S3 in account diversi da EMR Serverless (6.11 e versioni successive)	18 ottobre 2023
Nuova versione	EMR Serverless 6.14.0	17 ottobre 2023
Nuova caratteristica	Configurazione predefinita dell'applicazione per Serverless EMR	25 settembre 2023
Aggiornamento alle proprietà Hive predefinite	Sono stati aggiornati i valori predefiniti per`hive.driver.disk` , `hive.tez.disk.size` `hive.tez.`	12 settembre 2023

	`auto.reducer.parallelism` , e le proprietà del lavoro `tez.grouping.min-size` Hive.	
Nuova versione	EMR Serverless 6,13,0	11 settembre 2023
Nuova versione	EMR Serverless 6,12,0	21 luglio 2023
Nuova versione	EMR Serverless 6.11,0	8 giugno 2023
Aggiornamento della politica relativa ai ruoli collegati ai servizi	È stato aggiornato il `AmazonEMRServerlessServiceRolePolicy` _SLRruolo per pubblicare l'utilizzo a livello di account nel namespace. `"AWS/Usage"`	20 aprile 2023
EMR Serverless disponibilità generale (GA)	Questa è la prima versione pubblica di EMR Serverless.	1 giugno 2022

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.