



Guida per l'utente serverless di Amazon EMR

Amazon EMR



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

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Cos'è Amazon EMR Serverless?	1
Concetti	1
Versione di rilascio	1
Applicazione	2
Esecuzione del processo	3
Worker	3
Capacità preinizializzata	3
EMR Studio	4
Prerequisiti per iniziare	5
Iscriviti per un Account AWS	5
Concessione delle autorizzazioni	5
Concessione dell'accesso programmatico	7
Configura il AWS CLI	9
Aprire la console	10
Nozioni di base	11
Permissions	11
Archiviazione	11
Carichi di lavoro interattivi	11
Creare un ruolo di job runtime	12
Guida introduttiva alla console	18
Fase 1: creazione di un'applicazione	18
Fase 2: Inviare un job run o un carico di lavoro interattivo	19
Passaggio 3: Visualizza l'interfaccia utente e i registri dell'applicazione	22
Fase 4: pulizia	23
Iniziare da AWS CLI	23
Fase 1: creazione di un'applicazione	23
Fase 2: Inviare un job run	24
Fase 3: Esamina l'output	27
Fase 4: pulizia	28
Interazione e configurazione di un'applicazione EMR Serverless	30
Stati dell'applicazione	30
Utilizzo della console EMR Studio	31
Creazione di un'applicazione	32
Elenca le applicazioni dalla console EMR Studio	33

Gestione delle applicazioni dalla console EMR Studio	33
Utilizzando il AWS CLI	34
Configurazione di un'applicazione	35
Il comportamento delle applicazioni	35
Pre-initialized capacità di lavorare con un'applicazione in EMR Serverless	37
Configurazione predefinita dell'app	40
Personalizzazione di un'immagine	47
Prerequisiti	36
Fase 1: Creare un'immagine personalizzata da immagini di base EMR Serverless	48
Passaggio 2: convalida l'immagine localmente	49
Fase 3: carica l'immagine nel tuo repository Amazon ECR	50
Passaggio 4: creare o aggiornare un'applicazione con immagini personalizzate	50
Fase 5: consentire a EMR Serverless di accedere all'archivio di immagini personalizzato	52
Considerazioni e limitazioni	53
Configurazione dell'accesso VPC per le applicazioni EMR Serverless per la connessione ai dati	54
Crea applicazione	54
Configura l'applicazione	57
Procedure consigliate per la pianificazione delle sottoreti	58
Opzioni di architettura	59
Utilizzo dell'architettura x86_64	60
Utilizzo dell'architettura arm64 (Graviton)	60
Avvia nuove app con Graviton	60
Converti le app esistenti in Graviton	61
Considerazioni	62
Concorrenza e attesa dei lavori	62
Principali vantaggi della concorrenza e dell'accodamento	62
Guida introduttiva alla concorrenza e alla messa in coda	63
Considerazioni sulla concorrenza e sull'accodamento	64
Caricamento dei dati	65
Prerequisiti	65
Nozioni di base su S3 Express One Zone	66
Esecuzione di processi	68
Stati delle esecuzioni di processi	68
Annullamento del job run con periodo di grazia	70
Periodo di grazia per i lavori in batch	71

Periodo di grazia per i lavori di streaming	72
Considerazioni	53
Utilizzo della console EMR Studio	75
Invia un lavoro	75
I job di Access vengono eseguiti	78
Usando il AWS CLI	79
Esecuzione della politica IAM	80
Nozioni di base	80
Esempi di comandi CLI	81
Note importanti	82
Intersezione delle politiche	83
Utilizzo di dischi ottimizzati per la riproduzione casuale	86
Vantaggi principali	86
Nozioni di base	86
Utilizzo dello storage serverless	90
Vantaggi principali	90
Nozioni di base	91
Considerazioni e limitazioni	93
Supportato Regioni AWS	93
Lavori di streaming per l'elaborazione di dati in streaming continuo	94
Considerazioni e limitazioni	96
Nozioni di base	97
Connettori di streaming	97
Gestione dei log	100
Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless	101
Parametri Spark	101
Proprietà Spark	104
Best practice per la configurazione delle risorse	110
Esempi di Spark	111
Utilizzo delle configurazioni Hive quando si eseguono job EMR Serverless	112
Parametri Hive	112
Proprietà Hive	114
Esempi di Hive	128
Resilienza del processo	129
Monitoraggio di un processo con una policy di ripetizione	133
Registrazione con politica di riprova	133

Configurazione Metastore per EMR Serverless	133
Utilizzo del AWS Glue Data Catalog come metastore	134
Utilizzo di un metastore Hive esterno	139
Utilizzo della gerarchia AWS multicatalogo di Glue su EMR Serverless	144
Considerazioni sull'utilizzo di un metastore esterno	145
Cross-account Accesso S3	146
Prerequisiti	146
Utilizza una policy sui bucket S3	146
Usa un ruolo presunto	147
Esempi di ruoli presunti	150
Risoluzione degli errori	155
Errore: Job non riuscito poiché l'account ha raggiunto il limite di servizio sulla vCPU massima che può utilizzare contemporaneamente.	155
Errore: Job non riuscito perché l'applicazione ha superato le impostazioni di MaximumCapacity.	155
Errore: Job non riuscito a causa dell'impossibilità di allocare Worker poiché l'applicazione ha superato la capacità massima.	156
Errore: l'accesso a S3 è negato. Controlla le autorizzazioni di accesso S3 del ruolo Job Runtime sulle risorse S3 richieste.	156
Errore ModuleNotFoundError: nessun modulo denominato<module>. Consulta la guida per l'utente su come utilizzare le librerie python con EMR Serverless.	156
Errore: impossibile assumere il ruolo di esecuzione <role name>perché non esiste o non è configurato con la relazione di trust richiesta.	156
Allocazione dei costi a livello di Job	156
Comportamento predefinito	157
Come abilitare o disabilitare la funzionalità	157
Considerazioni e limitazioni	158
Esecuzione di sessioni interattive	159
Autorizzazioni richieste	159
Lavorare con sessioni interattive	161
Considerazioni e limitazioni	165
Esecuzione di carichi di lavoro interattivi	167
Panoramica di	167
Prerequisiti	167
Permissions	168
Configurazione	169

Considerazioni	169
Esecuzione di carichi di lavoro interattivi tramite l'endpoint Apache Livy	171
Prerequisiti	171
Autorizzazioni richieste	171
Nozioni di base	173
Considerazioni	179
Registrazione di log e monitoraggio	182
Archiviazione dei registri	182
Storage gestito	183
Simple Storage Service (Amazon S3)	184
Amazon CloudWatch	186
Registri rotanti	189
Crittografia dei log	191
Storage gestito	191
Bucket Amazon S3	191
Amazon CloudWatch	191
Autorizzazioni richieste	192
Configurazione di Log4j2	196
Log4j2 e Spark	196
Monitoraggio	200
Applicazioni e lavori	201
Metriche del motore Spark	210
Parametri di utilizzo	215
Automazione con EventBridge	216
Esempi di eventi EMR Serverless EventBridge	217
Applicazione di tag alle risorse	221
Che cos'è un tag?	221
Applicazione di tag alle risorse	221
Limitazioni relative all'etichettatura	222
Lavorare con i tag	223
Esercitazioni	225
Utilizzo di Java 17	225
JAVA_HOME	225
spark-defaults	226
Usare Hudi	227
Utilizzo di Iceberg	228

Usare le librerie Python	229
Utilizzo delle funzionalità native di Python	229
Creazione di un ambiente virtuale Python	230
Configurazione dei PySpark lavori per l'utilizzo delle librerie Python	231
Usare diverse versioni di Python	232
Utilizzo di Delta Lake OSS	234
Amazon EMR versioni 6.9.0 e successive	234
Amazon EMR versioni 6.8.0 e precedenti	235
Invio di lavori da Airflow	236
Utilizzo delle funzioni definite dall'utente di Hive	238
Utilizzo di immagini personalizzate	240
Usa una versione Python personalizzata	240
Usa una versione Java personalizzata	241
Crea un'immagine di data science	242
Elaborazione di dati geospaziali con Apache Sedona	242
Informazioni sulle licenze per l'utilizzo di immagini personalizzate	243
Utilizzo di Spark su Amazon Redshift	243
Avvio di un'applicazione Spark	244
Autenticazione su Amazon Redshift	245
Lettura e scrittura su Amazon Redshift	248
Considerazioni	250
Connessione a DynamoDB	251
Fase 1: Caricamento su Amazon S3	251
Fase 2: Creare una tabella Hive	252
Fase 3: Copia su DynamoDB	253
Fase 4: Interrogazione da DynamoDB	255
Configurazione dell'accesso multi-account	256
Considerazioni	258
Sicurezza	261
Best practice di sicurezza	262
Applicazione del principio del privilegio minimo	262
Isolamento di un codice dell'applicazione non attendibile	262
Autorizzazioni per il controllo degli accessi basato su ruoli (RBAC)	262
Protezione dei dati	262
Crittografia a riposo	263
Crittografia dei dati in transito	266

Identity and Access Management (IAM)	267
Destinatari	267
Autenticazione con identità	268
Gestione dell'accesso tramite policy	269
Come funziona EMR Serverless con IAM	271
Uso di ruoli collegati ai servizi	276
Ruoli Job Runtime per Amazon EMR Serverless	282
Politiche di accesso degli utenti	285
Policy per il controllo degli accessi basato su tag	289
Identity-based politiche	293
Aggiornamenti delle policy	296
Risoluzione dei problemi	296
Propagazione attendibile delle identità	299
Panoramica di	299
Funzionalità e vantaggi	300
Come funziona	300
Guida introduttiva a Trusted-Identity Propagation	301
Trusted Identity Propagation per carichi di lavoro interattivi	304
Sessioni in background degli utenti	305
Considerazioni sull'integrazione EMR Serverless Trusted-Identity-Propagation	310
Utilizzo di Lake Formation con EMR Serverless	311
Disponibilità delle funzionalità	311
Accesso completo alla tabella di Lake Formation per EMR Serverless	312
Lake Formation per FGAC	329
Crittografia tra lavoratori	360
Abilitazione della crittografia Mutual-TLS su EMR Serverless	361
Crittografia del disco con KMS CMK	361
Utilizzo del contesto di crittografia	362
Configurazione della crittografia del disco con chiavi gestite dal cliente	362
Autorizzazioni richieste per la crittografia del disco	364
Monitoraggio dell'utilizzo delle chiavi	367
Ulteriori informazioni	369
Secrets Manager per la protezione dei dati	370
Come funzionano i segreti	370
Creazione di un segreto	370
Specificare i riferimenti segreti	371

Concedi l'accesso al segreto	373
Ruota il segreto	375
S3 Access Grants per il controllo dell'accesso ai dati	375
Panoramica di	375
Avvia un'applicazione	376
Considerazioni	378
CloudTrail per la registrazione	378
Informazioni EMR Serverless in CloudTrail	378
Informazioni sulle voci dei file di registro EMR Serverless	379
Convalida della conformità	381
Resilienza	382
Sicurezza dell'infrastruttura	382
Analisi della configurazione e delle vulnerabilità	383
Punti finali e quote	384
Endpoint di servizio	384
Service Quotas	389
Limiti di API	390
Altre considerazioni	53
Versioni di rilascio	394
Runtime AWS per Apache Spark (emr-spark-8.0.0)	395
Runtime AWS per Apache Spark (emr-spark-8.0-preview)	396
EMR Serverless 7.13.0	398
EMR Serverless 7.12.0	398
EMR Serverless 7.11.0	399
EMR Serverless 7.10.0	400
EMR Serverless 7.9.0	400
EMR Serverless 7,80	401
EMR Serverless 7,7,0	401
EMR Serverless 7,6,0	401
EMR Serverless 7,5,0	402
EMR Serverless 7,40	402
EMR Serverless 7.3.0	402
EMR Serverless 7.2.0	403
EMR Serverless 7.1.0	404
EMR Serverless 7,0,0	404
EMR Serverless 6,15,0	404

EMR Serverless 6.14.0	405
EMR Serverless 6,13,0	405
EMR Serverless 6,12,0	405
EMR Serverless 6.11,0	406
EMR Serverless 6.10.0	406
EMR Serverless 6.9.0	407
EMR Serverless 6.8.0	408
EMR Serverless 6.7.0	408
Engine-specific modifiche	409
EMR Serverless 6.6.0	409
Cronologia dei documenti	411
.....	cdxiii

Cos'è Amazon EMR Serverless?

Amazon EMR Serverless è un'opzione di implementazione 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 gestire i cluster per eseguire applicazioni con questi framework.

EMR Serverless consente di evitare un approvvigionamento eccessivo o insufficiente delle risorse per i processi di elaborazione dei dati. EMR Serverless 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 in pochi secondi, come l'analisi interattiva dei dati, è possibile preinizializzare le risorse di cui l'applicazione ha bisogno al momento della creazione dell'applicazione.

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

EMR Serverless 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 di EMR Serverless che compaiono nella nostra Guida per l'utente EMR Serverless.

Versione di rilascio

Una versione di Amazon EMR 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 l'utente seleziona per l'implementazione e la configurazione di EMR Serverless in modo che possano eseguire le applicazioni. Quando create un'applicazione, specificate la versione di rilascio. Scegli la versione di rilascio di Amazon EMR e la versione del framework open source che desideri utilizzare nella tua applicazione. Per ulteriori informazioni sulle versioni preliminari, consulta. [Versioni di rilascio di Amazon EMR Serverless](#)

Applicazione

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

- La versione di rilascio di Amazon EMR per la versione del framework open source che desideri utilizzare. Per determinare la tua versione di rilascio, consulta. [Versioni 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, invia lavori di elaborazione dati o richieste interattive all'applicazione.

Ogni applicazione EMR Serverless viene eseguita su un Amazon Virtual Private Cloud (VPC) sicuro, completamente diverso dalle altre applicazioni. Inoltre, utilizza le policy AWS Identity and Access Management (IAM) 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 A/B dei test 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

EMR Serverless 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, fare riferimento a. [Interazione e configurazione di un'applicazione EMR Serverless](#)

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 invii un lavoro, devi specificare un ruolo di runtime, creato in IAM, che il job utilizza per accedere alle AWS risorse, come gli oggetti Amazon S3. È possibile inviare più richieste di esecuzione di un processo a un'applicazione e ogni esecuzione di lavoro può utilizzare un ruolo di runtime diverso per accedere AWS alle 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, fare riferimento a [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 rilascio di Amazon EMR. Quando pianifichi l'esecuzione di un lavoro, sostituisci queste dimensioni.

Quando si invia un lavoro, EMR Serverless calcola le risorse necessarie all'applicazione per il lavoro e pianifica i lavoratori. EMR Serverless suddivide i carichi di lavoro in attività, scarica immagini, provvede e organizza i lavoratori e li disattiva al termine del lavoro. EMR Serverless aumenta o riduce automaticamente i lavoratori 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

EMR Serverless 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 lavoratori preinizializzati, fare riferimento a [Configurazione di un'applicazione quando si lavora con EMR Serverless](#)

EMR Studio

EMR Studio è la console utente per la gestione delle applicazioni EMR Serverless. Se nel tuo account non esiste un EMR Studio quando crei la tua prima applicazione EMR Serverless, ne creiamo automaticamente uno per te. Accedi a EMR Studio dalla console Amazon EMR o attiva l'accesso federato dal tuo provider di identità (IdP) tramite IAM o IAM Identity Center. In questo modo, gli utenti possono accedere a Studio e gestire le applicazioni EMR Serverless senza accesso diretto alla console Amazon EMR. Per ulteriori informazioni su come le applicazioni EMR Serverless funzionano con EMR Studio, fare riferimento a e. [Creazione di un'applicazione EMR Serverless dalla console EMR Studio](#) [Esecuzione di processi dalla console EMR Studio](#)

Prerequisiti per iniziare a usare EMR Serverless

Questa sezione descrive i prerequisiti amministrativi per l'esecuzione di EMR Serverless. Questi includono la configurazione dell'account e la gestione delle autorizzazioni.

Argomenti

- [Iscriviti per un Account AWS](#)
- [Concessione delle autorizzazioni](#)
- [Installa e configura AWS CLI](#)
- [Aprire la console](#)

Iscriviti per un Account AWS

Per iniziare AWS, hai bisogno di un Account AWS. Per informazioni sulla creazione di un Account AWS, vedi Guida [introduttiva a un Account AWS](#) nella Guida Gestione dell'account AWS di riferimento.

Concessione delle autorizzazioni

Negli ambienti di produzione, ti suggeriamo di utilizzare politiche più dettagliate. Per esempi di tali politiche, fare riferimento a [Esempi di policy di accesso degli utenti per EMR Serverless](#). Per ulteriori informazioni sulla gestione degli accessi, consulta la sezione [Gestione degli accessi per AWS le risorse](#) nella IAM User Guide.

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRStudioCreate",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreateStudioPresignedUrl",

```

```

    "elasticmapreduce:DescribeStudio",
    "elasticmapreduce:CreateStudio",
    "elasticmapreduce:ListStudios"
  ],
  "Resource": [
    "*"
  ]
},
{
  "Sid": "EMRServerlessFullAccess",
  "Effect": "Allow",
  "Action": [
    "emr-serverless:*"
  ],
  "Resource": [
    "*"
  ]
},
{
  "Sid": "AllowEC2ENICreationWithEMRTags",
  "Effect": "Allow",
  "Action": [
    "ec2:CreateNetworkInterface"
  ],
  "Resource": [
    "arn:aws:ec2:*:*:network-interface/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
    }
  }
},
{
  "Sid": "AllowEMRServerlessServiceLinkedRoleCreation",
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole"
  ],
  "Resource": [
    "arn:aws:iam:*:*:role/aws-service-role/*"
  ]
}
]

```

}

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

- Utenti e gruppi in: AWS IAM Identity Center

Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina [Create a permission set](#) (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center .

- Utenti gestiti in IAM tramite un provider di identità:

Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina [Create a role for a third-party identity provider \(federation\)](#) della Guida per l'utente IAM.

- Utenti IAM:

- Crea un ruolo che l'utente possa assumere. Segui le istruzioni riportate nella pagina [Create a role for an IAM user](#) della Guida per l'utente IAM.

- (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina [Aggiunta di autorizzazioni a un utente \(console\)](#) nella Guida per l'utente IAM.

Concessione dell'accesso programmatico

Gli utenti necessitano di un accesso programmatico se desiderano interagire con utenti AWS esterni

a. Console di gestione AWS 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
IAM	(Consigliato) Utilizza le credenziali della console come credenziali temporanee per firmare le richieste programmatiche agli AWS SDK o alle AWS CLI API. AWS	Segui le istruzioni per l'interfaccia che desideri utilizzare. <ul style="list-style-type: none"> • Per la AWS CLI, consulta Login for AWS local development nella Guida

Quale utente necessita dell'accesso programmatico?	Per	Come
		<p>per l'utente.AWS Command Line Interface</p> <ul style="list-style-type: none"> Per gli AWS SDK, consulta Login per lo sviluppo AWS locale nella Guida di riferimento agli AWS SDK e agli strumenti.
<p>Identità della forza lavoro (Utenti gestiti nel centro identità IAM)</p>	<p>Utilizza credenziali temporane e per firmare le richieste programmatiche agli AWS SDK o alle AWS CLI API. AWS</p>	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> Per la AWS CLI, consulta Configurazione dell'uso AWS IAM Identity Center nella Guida AWS CLI per l'utente.AWS Command Line Interface Per AWS SDK, strumenti e AWS API, consulta l'autenticazione IAM Identity Center nella Guida di riferimento agli AWS SDK e agli strumenti.
<p>IAM</p>	<p>Utilizza credenziali temporane e per firmare le richieste programmatiche agli SDK o alle API AWS CLI. AWS AWS</p>	<p>Segui le istruzioni in Uso delle credenziali temporanee con AWS risorse nella Guida per l'utente IAM.</p>

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare le richieste programmatiche agli AWS CLI AWS SDK o alle API. AWS	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli AWS SDK e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli SDK e agli AWS strumenti. • Per le AWS API, consulta Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Installa e configura AWS CLI

Se desideri utilizzare le API EMR Serverless, installa la versione più recente di (). AWS Command Line Interface AWS CLI Non è necessario AWS CLI utilizzare EMR Serverless dalla console EMR Studio e iniziare senza la 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, consulta [Installazione o aggiornamento della versione più recente di](#). AWS CLI
2. Per configurare AWS CLI e configurare in modo sicuro l'accesso a Servizi AWS, incluso EMR Serverless, fare riferimento a 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 della configurazione, a meno che non venga impostata con un parametro o un profilo. Per impostare Regione AWS un parametro, aggiungilo a ciascun comando. `--region`

Per impostare Regione AWS 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](#). Successivamente, imposta le tue Regione AWS e le altre impostazioni con un comando simile a quello illustrato nell'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 parte dalla console Amazon [EMR](#). Se non hai già effettuato l'accesso al tuo Account AWS, accedi, quindi apri la [console Amazon EMR](#) e passa alla sezione successiva per iniziare a usare Amazon EMR.

Inizia a usare Amazon EMR Serverless

Questo tutorial ti aiuta a iniziare a usare EMR Serverless quando distribuisce un carico di lavoro Spark o Hive di esempio. 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, completare le seguenti attività.

Argomenti

- [Concedere le autorizzazioni per utilizzare EMR Serverless](#)
- [Preparazione dello storage per EMR Serverless](#)
- [Crea un EMR Studio per eseguire carichi di lavoro interattivi](#)
- [Creare un ruolo di job runtime](#)
- [Guida introduttiva a EMR Serverless dalla console](#)
- [Iniziare da AWS CLI](#)

Concedere le autorizzazioni per utilizzare EMR Serverless

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

Preparazione dello 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 EMR Serverless. 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 *amzn-s3-demo-bucket* con il nome del bucket appena creato.

Crea un 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 EMR Serverless per creare un workspace](#). Per la procedura di configurazione, consulta [Configurare un EMR Studio](#) nella Amazon

EMR Management Guide. Per ulteriori informazioni sui carichi di lavoro interattivi, consulta [Esegui carichi di lavoro interattivi con EMR Serverless tramite EMR Studio](#)

Creare un ruolo di job runtime

I job eseguiti in EMR Serverless utilizzano un ruolo di runtime che fornisce autorizzazioni granulari per risorse specifiche Servizi AWS e in fase di esecuzione. In questo tutorial, un bucket S3 pubblico ospita i dati e gli script. Il bucket *amzn-s3-demo-bucket* memorizza l'output.

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

Console

1. Accedi alla console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione sinistro, scegli Policy.
3. Scegli Crea policy.
4. La pagina Crea policy si apre in una nuova scheda. Seleziona l'editor delle politiche come Json e incolla il codice JSON della politica di seguito.

Important

Sostituisci *amzn-s3-demo-bucket* la policy seguente con il nome effettivo del bucket creato in [Preparazione dello storage per EMR Serverless](#). Questa è una politica di base per l'accesso a S3. Per altri esempi di ruoli di job runtime, consulta [Ruoli Job Runtime per Amazon EMR Serverless](#).

JSON

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

```

    "Action": [
      "s3:GetObject",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3::*.elasticmapreduce",
      "arn:aws:s3::*.elasticmapreduce/*"
    ]
  },
  {
    "Sid": "FullAccessToOutputBucket",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:ListBucket",
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws:s3::amzn-s3-demo-bucket",
      "arn:aws:s3::amzn-s3-demo-bucket/*"
    ]
  },
  {
    "Sid": "GlueCreateAndReadDataCatalog",
    "Effect": "Allow",
    "Action": [
      "glue:GetDatabase",
      "glue:CreateDatabase",
      "glue:GetDataBases",
      "glue:CreateTable",
      "glue:GetTable",
      "glue:UpdateTable",
      "glue:DeleteTable",
      "glue:GetTables",
      "glue:GetPartition",
      "glue:GetPartitions",
      "glue:CreatePartition",
      "glue:BatchCreatePartition",
      "glue:GetUserDefinedFunctions"
    ],
    "Resource": [
      "*"
    ]
  }
]

```

```

    }
  ]
}

```

5. Scegli Avanti per inserire un nome per la tua politica, ad EMRServerlessS3AndGlueAccessPolicy esempio Crea politica
6. Nel riquadro di navigazione a sinistra della console IAM, scegli Ruoli.
7. Scegli Crea ruolo.
8. Per il tipo di ruolo, scegli Custom trust policy e incolla la seguente policy di fiducia. Ciò consente ai lavori inviati alle tue applicazioni Amazon EMR Serverless di accedere ad altri per tuo Servizi AWS conto.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSTSAssumerole",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}

```

9. Scegli Avanti per accedere alla pagina Aggiungi autorizzazioni, quindi scegli S3. EMRServerless AndGlueAccessPolicy
10. Nella pagina Nome, rivedi e crea, per Nome ruolo, inserisci un nome per il tuo ruolo, ad esempio. EMRServerlessS3RuntimeRole Per creare questo ruolo IAM, scegli Crea ruolo.

CLI

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

2. Creare un ruolo IAM 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
```

Annotate l'ARN nell'output. L'ARN del nuovo ruolo viene utilizzato durante l'invio del lavoro, denominato successivamente. *job-role-arn*

3. Crea un file denominato `emr-sample-access-policy.json` che definisca la policy IAM 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. *amzn-s3-demo-bucket*

⚠ Important

amzn-s3-demo-bucket Sostituiscilo nella policy seguente con il nome effettivo del bucket creato in.. [Preparazione dello storage per EMR Serverless](#) Questa è una politica di base per l'accesso a AWS Glue e S3. Per altri esempi di ruoli in fase di job runtime, consulta [Ruoli Job Runtime per Amazon EMR Serverless](#).

JSON

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

```

    "Sid": "GlueCreateAndReadDataCatalog",
    "Effect": "Allow",
    "Action": [
      "glue:GetDatabase",
      "glue:CreateDatabase",
      "glue:GetDataBases",
      "glue:CreateTable",
      "glue:GetTable",
      "glue:UpdateTable",
      "glue>DeleteTable",
      "glue:GetTables",
      "glue:GetPartition",
      "glue:GetPartitions",
      "glue:CreatePartition",
      "glue:BatchCreatePartition",
      "glue:GetUserDefinedFunctions"
    ],
    "Resource": [
      "*"
    ]
  }
]
}

```

4. Crea una policy IAM denominata `EMRServerlessS3AndGlueAccessPolicy` con il file di policy che hai creato nel passaggio 3. Prendi nota dell'ARN nell'output, poiché utilizzerai l'ARN della nuova politica nella fase successiva.

```

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

```

Nota l'ARN della nuova politica nell'output. Lo sostituirai *policy-arn* nel passaggio successivo.

5. Allega la policy IAM `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

Questa sezione descrive l'utilizzo di EMR Serverless, inclusa la creazione di un EMR Studio. Descrive inoltre come inviare le esecuzioni dei job e visualizzare i log.

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 Console di gestione AWS e apri la console Amazon EMR su https://console.aws.amazon.com/emr.](https://console.aws.amazon.com/emr)
2. Nel riquadro di navigazione a sinistra, scegli EMR Serverless per accedere alla pagina di destinazione EMR Serverless.
3. Per creare o gestire applicazioni EMR Serverless, è necessaria l'interfaccia utente di EMR Studio.
 - Se hai già un EMR Studio nel Regione AWS luogo in cui desideri creare un'applicazione, seleziona Gestisci applicazioni per accedere a EMR Studio o seleziona lo studio che desideri utilizzare.
 - Se non disponi di uno studio EMR nel Regione AWS luogo in cui desideri creare un'applicazione, scegli Inizia, quindi scegli Crea e avvia Studio. EMR Serverless crea per te un EMR Studio 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://amzn-s3-demo-bucket/scripts/
```

2. Il completamento [Fase 1: Creare un'applicazione EMR Serverless](#) porta alla pagina dei dettagli dell'applicazione in EMR Studio. Qui, scegli l'opzione Invia lavoro.
3. Nella pagina Invia offerta di 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 creato [Creare un ruolo di job runtime](#).
 - Nel campo Posizione dello script, inserisci `s3://amzn-s3-demo-bucket/scripts/wordcount.py` come URI S3.
 - Nel campo Argomenti dello script, inserisci `["s3://amzn-s3-demo-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.sql` 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.sql` nel tuo bucket S3 con il seguente comando.

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

3. Il completamento [Fase 1: Creare un'applicazione EMR Serverless](#) porta alla pagina dei dettagli dell'applicazione in EMR Studio. Qui, scegli l'opzione Invia lavoro.
4. Nella pagina Invia offerta di 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 creato [Creare un ruolo di job runtime](#).
 - Nel campo Posizione dello script, inserisci `s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.sql` come URI S3.
 - Nella sezione delle proprietà di Hive, scegli Modifica come testo e inserisci le seguenti configurazioni.

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

- Nella sezione Configurazione del lavoro, scegli Modifica come JSON e inserisci il seguente codice JSON.

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

5. Per avviare l'esecuzione del processo, 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 EMR Serverless. Per ulteriori informazioni, tra cui autorizzazioni e prerequisiti, consulta [Esegui carichi di lavoro interattivi con EMR Serverless tramite EMR Studio](#)

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

1. Vai alla scheda Aree di lavoro in EMR Studio. Se devi ancora configurare una posizione di archiviazione 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 ruolo IAM 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 spazio di lavoro. 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 [Monitoraggio delle applicazioni e dei job EMR Serverless](#) 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 viene visualizzato come **Riuscito**, 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 [Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless](#) e [Utilizzo delle configurazioni Hive quando si eseguono job EMR Serverless](#)

Iniziare da AWS CLI

Inizia a usare EMR Serverless AWS CLI con i comandi per creare un'applicazione, eseguire lavori, controllare l'output dell'esecuzione del processo ed eliminare le tue risorse.

Fase 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 rilascio di Amazon EMR 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
```

```
--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 a [Fase 2: Invia un job run alla tua applicazione EMR Serverless](#), assicurati che la tua candidatura abbia raggiunto CREATED lo stato con l'[get-application](#) API.

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

EMR Serverless 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 quando si lavora con EMR Serverless](#).

Fase 2: Invia un job run alla tua 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://amzn-s3-demo-bucket/scripts/
```

2. Nel comando seguente, sostituiscilo *application-id* con l'ID dell'applicazione. Sostituisci *job-role-arn* con il ruolo di runtime ARN in cui hai creato. [Creare un ruolo di job runtime](#) *job-run-name* Sostituiscilo con il nome con cui vuoi chiamare il job run. Sostituisci tutte *amzn-s3-demo-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://amzn-s3-demo-bucket/scripts/wordcount.py",
      "entryPointArguments": ["s3://amzn-s3-demo-bucket/emr-serverless-
spark/output"],
      "sparkSubmitParameters": "--conf spark.executor.cores=1
--conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf
spark.driver.memory=4g --conf spark.executor.instances=1"
    }
  }'
```

3. Annotate l'ID di esecuzione del lavoro 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.q1` 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;
```

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

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

- Nel comando seguente, sostituiscilo *application-id* con il tuo ID dell'applicazione. Sostituisci *job-role-arn* con il ruolo di runtime ARN in cui hai creato. [Creare un ruolo di job runtime](#) Sostituisci tutte *amzn-s3-demo-bucket* le stringhe con il bucket Amazon S3 che hai creato e `/output` aggiungi `/logs` e al percorso. Questo crea nuove cartelle nel bucket, dove 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://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-
query.sql",
      "parameters": "--hiveconf hive.log.explain.output=false"
    }
  }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
      "classification": "hive-site",
      "properties": {
        "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-
hive/hive/scratch",
        "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/warehouse",
        "hive.driver.cores": "2",
        "hive.driver.memory": "4g",
        "hive.tez.container.size": "4096",
        "hive.tez.cpu.vcores": "1"
      }
    }],
    "monitoringConfiguration": {
      "s3MonitoringConfiguration": {
        "logUri": "s3://amzn-s3-demo-bucket/emr-serverless-hive/logs"
      }
    }
  }
```

```
}'
```

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

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

Il completamento 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 su `s3://amzn-s3-demo-bucket/emr-serverless-spark/logs`, puoi trovare i log relativi a questo specifico job in cui viene eseguito. `s3://amzn-s3-demo-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 nella cartella nella destinazione `sparklogs` 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://amzn-s3-demo-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 su `s3://amzn-s3-demo-bucket/emr-serverless-hive/logs`, puoi trovare i log relativi a questo specifico processo in cui viene eseguito. `s3://amzn-s3-demo-bucket/emr-serverless-hive/logs/applications/application-id/jobs/job-run-id`

Per le applicazioni Hive, EMR Serverless carica continuamente il driver Hive nella cartella e i registri HIVE_DRIVER delle attività Tez nella cartella della destinazione TEZ_TASK 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 *amzn-s3-demo-bucket* con il nome effettivo del bucket S3 creato in.. [Preparazione dello storage per EMR Serverless](#)

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

```
--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 [Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless](#) e [Utilizzo delle configurazioni Hive quando si eseguono job EMR Serverless](#)

Interazione e configurazione di un'applicazione EMR Serverless

Questa sezione spiega come interagire con la tua applicazione Amazon EMR Serverless con AWS CLI. Descrive inoltre la configurazione di un'applicazione, l'esecuzione delle personalizzazioni e le impostazioni predefinite per i motori Spark e Hive.

Argomenti

- [Stati dell'applicazione](#)
- [Creazione di un'applicazione EMR Serverless dalla console EMR Studio](#)
- [Interazione con l'applicazione EMR Serverless su AWS CLI](#)
- [Configurazione di un'applicazione quando si lavora con EMR Serverless](#)
- [Personalizzazione di un'immagine EMR Serverless](#)
- [Configurazione dell'accesso VPC per le applicazioni EMR Serverless per la connessione ai dati](#)
- [Opzioni di architettura Serverless Amazon EMR](#)
- [Concorrenza dei job e accodamento per un'applicazione EMR Serverless](#)

Stati dell'applicazione

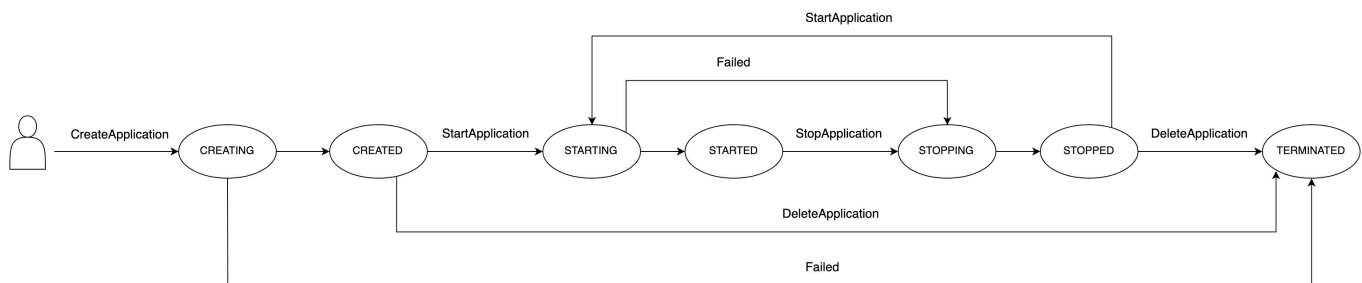
Quando si crea un'applicazione con EMR Serverless, l'esecuzione dell'applicazione entra nello stato CREATING. Dopodiché passa per i seguenti stati fino a quando non ha esito positivo (codice di uscita 0) o negativo (codice di uscita diverso da zero).

Le applicazioni possono avere i seguenti stati:

Stato	Description
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'applicazione per modificare la configurazione iniziale della capacità.

Stato	Description
Avvio di	L'applicazione si sta avviando e sta fornendo capacità.
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'applicazione 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 illustra la traiettoria degli stati delle applicazioni EMR Serverless.



Creazione di un'applicazione EMR Serverless dalla console EMR Studio

Dalla console EMR Studio, crea, accedi e gestisci applicazioni EMR Serverless. Per accedere alla console EMR Studio, segui le istruzioni in [Guida introduttiva dalla console](#).

Creazione di un'applicazione

Con la pagina Crea applicazione, crea 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 release, scegli il numero di release EMR.
4. Nelle opzioni Architettura, scegli l'architettura del set di istruzioni da utilizzare. Per ulteriori informazioni, vedi [Opzioni di architettura Serverless Amazon EMR](#).
 - arm64 — Architettura ARM a 64 bit; per usare i processori Graviton
 - x86_64 — architettura x86 a 64 bit; per utilizzare processori basati su x86
5. Esistono due opzioni di configurazione dell'applicazione per i campi rimanenti: impostazioni predefinite e impostazioni personalizzate. Questi campi sono facoltativi.

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 VPC. 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 vengono assegnati all'applicazione.

Per ulteriori informazioni sulla capacità preinizializzata, sui limiti delle applicazioni e sul comportamento dell'applicazione, fare riferimento a [Configurazione di un'applicazione quando si lavora con EMR Serverless](#). Per ulteriori informazioni sulla connettività di rete, fare riferimento a [Configurazione dell'accesso VPC per le applicazioni EMR Serverless per la connessione ai dati](#)

6. Per creare l'applicazione, scegli Crea applicazione.

Elenca le applicazioni dalla console EMR Studio

È possibile accedere a 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 dalla console EMR Studio

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 essere interrotta. Per ulteriori informazioni sulle transizioni di stato delle applicazioni, fare riferimento a [Stati 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, modifica l'etichetta di rilascio di un'applicazione per aggiornarla a una versione diversa di Amazon EMR o cambia 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, fare riferimento a [Creazione 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, fare riferimento a [Stati dell'applicazione](#).

Interazione con l'applicazione EMR Serverless su AWS CLI

Da AWS CLI, crea, descrivi ed elimina singole applicazioni. Puoi anche elencare tutte le tue applicazioni in modo da accedervi con un colpo d'occhio. Questa sezione descrive come eseguire queste azioni. Per ulteriori operazioni sulle applicazioni, come l'avvio, l'arresto e gli aggiornamenti delle applicazioni, fare riferimento all'[EMR Serverless API Reference](#). Per esempi su come utilizzare l'API EMR Serverless utilizzando il AWS SDK per Java, consulta gli [esempi Java](#) nel nostro repository. GitHub Per esempi su come utilizzare l'API EMR Serverless utilizzando il AWS SDK per Python (Boto), consulta gli esempi di [Python](#) nel nostro repository. GitHub

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

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

Per descrivere un'applicazione, usatela `get-application` e fornirla. `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 quando si lavora con EMR Serverless

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

Argomenti

- [Comprensione del comportamento delle applicazioni in EMR Serverless](#)
- [Pre-initialized capacità di lavorare con un'applicazione in EMR Serverless](#)
- [Configurazione predefinita dell'applicazione per EMR Serverless](#)

Comprensione del comportamento delle applicazioni in EMR Serverless

Questa sezione descrive il comportamento di invio dei lavori, la configurazione della capacità per la scalabilità e le impostazioni di configurazione del worker per EMR Serverless.

Comportamento predefinito dell'applicazione

Auto-start— Per impostazione predefinita, un'applicazione è configurata per l'avvio automatico all'invio del lavoro. È possibile disattivare questa funzionalità.

Auto-stop— 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

È consigliabile configurare la capacità massima in modo che sia proporzionale alle dimensioni dei lavoratori supportate moltiplicando il numero di lavoratori per le loro dimensioni. Ad

esempio, se desideri limitare l'applicazione a 50 worker con 2 vCPU, 16 GB di memoria e 20 GB per disco, imposta la capacità massima su 100 vCPU, 800 GB per memoria e 1000 GB per disco.

Configurazioni dei worker supportate

La tabella seguente elenca le configurazioni e le dimensioni dei lavoratori supportate che possono essere specificate per EMR Serverless. Configura diverse dimensioni per driver ed esecutori in base alle esigenze del carico di lavoro.

Configurazioni e dimensioni dei lavoratori

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

CPU: ogni worker può avere 1, 2, 4, 8 o 16 vCPU.

Memoria: ogni worker 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.memoryOverhead` e `spark.executor.memoryOverhead`. 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 vCPU per l'istanza di lavoro e una capacità di storage preinizializzata di 30 GB, imposta un valore di circa 27 GB come memoria esecutore per il tuo job Spark. Ciò massimizza l'utilizzo della capacità preinizializzata. La memoria utilizzabile è 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.

Pre-initialized capacità di lavorare con un'applicazione in EMR Serverless

EMR Serverless offre una funzionalità opzionale che mantiene il conducente e il personale 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, imposta il `initialCapacity` parametro di un'applicazione sul numero di lavoratori che desideri preinizializzare. Con la capacità dei lavoratori preinizializzata, i lavori iniziano immediatamente. Questa soluzione è ideale quando si desidera implementare applicazioni iterative e lavori urgenti.

Pre-initialized la capacità consente a un gruppo ristretto di lavoratori di essere pronti all'avvio di lavori e sessioni in pochi secondi. I lavoratori preinizializzati assegnati verranno pagati anche quando l'applicazione è inattiva, pertanto suggeriamo di abilitarla per i casi d'uso che traggono vantaggio dai tempi di avvio rapidi e di ridimensionarla per un utilizzo ottimale delle risorse. Le applicazioni EMR Serverless si chiudono automaticamente quando sono inattive. Suggeriamo di mantenere attiva questa funzionalità quando si utilizzano worker preinizializzati per evitare addebiti imprevisti.

Quando invii un lavoro, se `initialCapacity` sono disponibili lavoratori di, il lavoro utilizza tali risorse per iniziare la sua 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.

Pre-initialized la capacità 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 lavori 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 `CREATED` a `STARTED`.

È possibile configurare l'applicazione in modo che rilasci 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 o modificarle quando l'applicazione è in uno `STOPPED` stato `CREATED` o.

È possibile modificare i `InitialCapacity` conteggi e specificare configurazioni di calcolo come CPU, memoria e disco per ogni lavoratore. Poiché non puoi apportare modifiche parziali, specifica tutte le configurazioni di calcolo quando modifichi i valori. È possibile modificare le configurazioni solo quando l'applicazione è nello stato `CREATED` o `STOPPED`.

Note

Per ottimizzare l'uso delle risorse da parte dell'applicazione, suggeriamo di allineare le dimensioni dei container alle dimensioni dei dipendenti preinizializzate. Ad esempio, se configuri la dimensione dell'esecutore Spark su 2 CPU e la memoria su 8 GB, ma la capacità preinizializzata del worker è di 4 CPU 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, specifica quanti worker iniziano come driver e quanti come esecutori. Allo stesso modo, quando usi Apache Hive, specifica 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 richiesta API crea un'applicazione che esegue Apache Hive basata sulla release `emr-6.6.0` di Amazon EMR. L'applicazione inizia con 5 driver Hive preinizializzati, ciascuno con 2

vCPU e 4 GB di memoria, e 50 task worker Tez preinizializzati, ciascuno con 4 vCPU e 8 GB di memoria. 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 vCPU e 1024 GB di memoria. 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 richiesta API crea un'applicazione che esegue Apache Spark 3.2.0 basata sulla versione 6.6.0 di Amazon EMR. L'applicazione inizia con 5 driver Spark preinizializzati, ciascuno con 2 vCPU e 4 GB di memoria, e 50 executor preinizializzati, ciascuno con 4 vCPU e 8 GB di memoria. 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 vCPU 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 EMR Serverless

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, fare riferimento a. [Archiviazione dei registri](#)

runtimeConfiguration

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

Le classificazioni di configurazione disponibili variano in base alla specifica release EMR Serverless. Ad esempio, le classificazioni per Log4j personalizzate `spark-executor-log4j2` sono disponibili solo con le versioni `spark-driver-log4j2 6.8.0` e successive. Per un elenco delle proprietà specifiche dell'applicazione, fare riferimento a and. [Proprietà del lavoro Spark](#)
[Proprietà del lavoro di Hive](#)

È inoltre possibile configurare le [proprietà di Apache Log4j2](#), [Gestione dei segreti AWS 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.

JSON

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

```

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

```

Per ulteriori informazioni sulla creazione di policy personalizzate per i segreti, consulta gli [esempi di policy di autorizzazione disponibili](#) [Gestione dei segreti AWS nella Guida per l'utente](#) della Gestione dei segreti AWS.

Note

`runtimeConfiguration` Ciò che specifichi a livello di applicazione viene `applicationConfiguration` mappato all'[StartJobRunAPI](#).

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://amzn-s3-demo-logging-bucket/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 l'API. [StartJobRun](#) EMR Serverless 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 sulle classificazioni di priorità, fare riferimento a e. [Parametro di sovrascrittura della configurazione Hive](#) [Parametro di override della configurazione Spark](#)

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, fornisci la chiave per la configurazione che desideri rimuovere e passa 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 rimuoverelacloudWatchLogging, passate il flag `enabled` asfalse.
- **managedPersistenceMonitoringConfiguration**- Per rimuovere le impostazioni di persistenza gestita e tornare allo stato abilitato predefinito, passa una dichiarazione vuota `{}` per la configurazione.
- **s3MonitoringConfiguration**- Per rimuoveres3MonitoringConfiguration, 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://amzn-s3-demo-destination-bucket1/
wordcount_output"]
    }
  }' \
  --configuration-overrides '{
    "applicationConfiguration": [
      {
        // Override existing configuration for spark-defaults in the
application
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.cores": "2",
          "spark.executor.cores": "1",
          "spark.driver.memory": "4G",
          "spark.executor.memory": "4G"
        }
      },
      {
        // Add configuration for spark-executor-log4j2
        "classification": "spark-executor-log4j2",
        "properties": {
```

```

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

```

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](#) [Parametro di override della configurazione Spark](#)

- La classificazione `spark-defaults` verrà aggiornata con le proprietà specificate a livello di mansione. Ai fini di questa classificazione `StartJobRun` vengono 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, utilizza immagini personalizzate per impacchettare le dipendenze delle applicazioni e gli ambienti di runtime in un unico contenitore con Amazon EMR Serverless. 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 non sono ampiamente disponibili nella distribuzione pubblica degli ambienti di runtime Amazon EMR.
- Integra EMR Serverless con gli attuali processi di compilazione, test e implementazione 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 JDK e Python per le applicazioni.

EMR Serverless fornisce immagini da utilizzare come base per la creazione di 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 su Amazon EMR versione 6.9.0, utilizza le seguenti immagini.

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

Prerequisiti

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

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ECRRepositoryListGetPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:DescribeImages"
      ],
      "Resource": [
        "arn:aws:ecr:*:123456789012:repository/my-repo"
      ]
    }
  ]
}
```

Per ulteriori esempi di policy basate sull'identità di Amazon ECR, consulta gli esempi di policy basate sull'identità di [Amazon Elastic Container Registry](#).

Fase 1: Creare un'immagine personalizzata da immagini di base EMR Serverless

Innanzitutto, crea un [Dockerfile](#) che inizi con un'FROMistruzione che utilizza l'immagine di base preferita. Dopo le FROM istruzioni, includi tutte le modifiche che desideri apportare all'immagine.

L'immagine di base imposta automaticamente USER suhadoop. Questa impostazione non dispone delle autorizzazioni per tutte le modifiche incluse. Come soluzione alternativa, imposta suroot, modifica USER l'immagine e quindi reimpostala su. USER hadoop : hadoop Per fare riferimento agli esempi per i casi d'uso più comuni, fare riferimento a [Utilizzo 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 runs 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

EMR Serverless 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 [l'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
```

L'output appare 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.
-----
```

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

Invia la tua immagine Amazon ECR al tuo repository Amazon ECR con i seguenti comandi. Assicurati di disporre delle autorizzazioni IAM 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

Scegli la Console di gestione AWS scheda o la AWS CLI scheda in base a come desideri avviare l'applicazione, quindi completa i seguenti passaggi.

Console

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

CLI

- Crea un'applicazione con il `image-configuration` parametro. EMR Serverless 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, utilizzare 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. EMR Serverless 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"
}'

```

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

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

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EmrServerlessCustomImageSupport",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [

```

```

    "ecr:BatchGetImage",
    "ecr:DescribeImages",
    "ecr:GetDownloadUrlForLayer"
  ],
  "Resource": "arn:aws:ecr:*:123456789012:repository/my-repo",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/
**
    }
  }
}
]
}

```

Come best practice di sicurezza, aggiungi una chiave di `aws:SourceArn` condizione alla policy del repository. La chiave di condizione globale IAM `aws:SourceArn` garantisce che EMR Serverless utilizzi il repository solo per l'ARN di un'applicazione. Per ulteriori informazioni sulle politiche del repository Amazon ECR, 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.
- EMR Serverless ignora le nostre `[ENTRYPOINT]` istruzioni nel `[CMD]` file Docker. Usa istruzioni comuni nel file Docker, come, e. `[COPY]` `[RUN]` `[WORKDIR]`
- Non modificate le variabili di ambiente `JAVA_HOMESPARK_HOME`, `HIVE_HOME`, `TEZ_HOME` quando create un'immagine personalizzata.
- Le immagini personalizzate non possono superare le dimensioni di 10 GB.
- Se modifichi file binari o jar nelle immagini di base di Amazon EMR, ciò può causare errori di avvio dell'applicazione o del processo.
- Il repository Amazon ECR deve trovarsi nello stesso Regione AWS che usi per avviare le applicazioni EMR Serverless.

Configurazione dell'accesso VPC per le applicazioni EMR Serverless per la connessione ai dati

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

Note

È necessario configurare l'accesso VPC se si desidera utilizzare un database metastore Hive esterno per l'applicazione. [Per informazioni su come configurare un metastore Hive esterno, consulta la sezione Configurazione di Metastore.](#)

Crea applicazione

Nella pagina Crea applicazione, scegli le impostazioni personalizzate e specifica il VPC, le sottoreti e i gruppi di sicurezza che le applicazioni EMR Serverless possono utilizzare.

VPCs

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

Sottoreti

Scegli le sottoreti all'interno del VPC che contiene il tuo data store. La pagina Crea applicazione elenca tutte le sottoreti per gli archivi dati nel tuo VPC. Sono supportate sia le sottoreti pubbliche che quelle private. Puoi passare sottoreti private o pubbliche alle tue applicazioni. La scelta di disporre di una sottorete pubblica o privata comporta alcune considerazioni di cui tenere conto.

Per le sottoreti private:

- Le tabelle di routing associate non devono avere gateway Internet.
- Per la connettività in uscita a Internet, se necessario, configura i percorsi in uscita utilizzando un gateway NAT. [Per configurare un gateway NAT, fai riferimento ai gateway NAT.](#)

- Per la connettività Amazon S3, configura un gateway NAT o un endpoint VPC. Per configurare un endpoint VPC S3, consulta [Creare un endpoint gateway](#).
- Se configuri un endpoint VPC S3 e alleggi una policy endpoint per controllare l'accesso, segui le istruzioni in [Logging for EMR Serverless con storage gestito per fornire le autorizzazioni a EMR Serverless per archiviare](#) e servire i log delle applicazioni.
- Per la connettività con altri Servizi AWS dispositivi esterni al VPC, ad esempio Amazon DynamoDB, configura gli endpoint VPC o un gateway NAT. Per configurare gli endpoint VPC per Servizi AWS, consulta Lavora [con gli endpoint VPC](#).

Note

Quando configuri un'applicazione Amazon EMR Serverless in una sottorete privata, ti suggeriamo di configurare anche gli endpoint VPC per Amazon S3. Se la tua applicazione EMR Serverless si trova in una sottorete privata senza endpoint VPC per Amazon S3, dovrai sostenere costi aggiuntivi per il gateway NAT associati al traffico S3. Questo perché il traffico tra l'applicazione EMR e Amazon S3 non rimarrà all'interno del VPC quando gli endpoint VPC non sono configurati.

Per le sottoreti pubbliche:

- Questi hanno un percorso verso un Internet Gateway.
- È necessario garantire configurazioni adeguate dei gruppi di sicurezza per controllare il traffico in uscita.

I lavoratori possono connettersi agli archivi dati all'interno del tuo VPC tramite il traffico in uscita. Per impostazione predefinita, EMR Serverless blocca l'accesso in entrata ai lavoratori. Questo serve a 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 utilizza 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, fare riferimento a [the section called “Procedure consigliate per la pianificazione delle sottoreti”](#)

Considerazioni e limitazioni per le sottoreti

- EMR Serverless con sottoreti pubbliche non supporta Lake Formation. AWS
- Il traffico in entrata non è supportato per le sottoreti pubbliche.

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 nel tuo VPC. EMR Serverless 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 EMR Serverless. EMR Serverless non consente di creare, aggiornare, avviare un'applicazione se i gruppi di sicurezza hanno porte aperte alla rete Internet pubblica su 0.0.0.0/0 o nell'intervallo: :/0. Ciò offre maggiore sicurezza e isolamento e rende più efficiente la gestione delle regole di rete. Ad esempio, questo blocca il traffico imprevisto verso i lavoratori con indirizzi IP pubblici. Per comunicare con i cluster Amazon Redshift, ad esempio, definisci le regole del traffico tra i gruppi di sicurezza Serverless di Redshift ed EMR, come illustrato nell'esempio riportato nella sezione 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 sicurezza EMR Serverless.

Tipo	Protocollo	Intervallo porte	Origine
Tutte le regole TCP	TCP	5439	emr-serverless-security-group

2. Aggiungi una regola per il traffico in uscita da uno dei gruppi di sicurezza EMR Serverless. Esegui questa operazione in due modi. Innanzitutto, apri 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
Tutte le regole 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.

Accedi ai dettagli dell'esecuzione del processo

Nella pagina dei dettagli del Job run, accedi alla 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 un Amazon VPC. Ad esempio, un VPC con 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 EMR Serverless. Ad esempio, se un'applicazione richiede 4 vCPU worker ed è possibile scalare fino a 4.000 vCPU, l'applicazione richiede al massimo 1.000 worker 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 una zona di disponibilità diversa 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 soddisfa 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 maschera /22 in un VPC di /16 netmask che possono essere allocate a 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
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	10.0.8.0 - 10.0.11.255	1,024	1.019

ID sottorete	Indirizzo di sottorete	Maschera di sottorete	Intervallo di indirizzi IP	Indirizzi IP disponibili	Indirizzi IP utilizzabili
4	10,012,0	255,255,252	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 worker a 16 vCPU con un limite di scalabilità delle applicazioni di 4.000 vCPU richiede al massimo 250 lavoratori per un totale di 250 indirizzi IP disponibili per il provisioning delle interfacce di rete. Per effettuare il provisioning di 250 interfacce di rete sono necessarie sottoreti in più zone di disponibilità con una dimensione della maschera inferiore o uguale 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 hai 3 applicazioni che richiedono 4 vCPU worker e ciascuna può scalare fino a 4000 vCPU con una quota di servizio basata su 12.000 vCPU a livello di account, ogni sottorete richiede 3000 indirizzi IP disponibili. In caso contrario, è possibile provare ad aumentare il numero di indirizzi IP disponibili. Tale operazione può essere effettuata associando i blocchi di instradamento interdominio senza classi (CIDR) secondari al VPC. Per ulteriori informazioni, consulta [Associare blocchi IPv4 CIDR aggiuntivi al tuo 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 Serverless Amazon EMR

L'architettura del set di istruzioni dell'applicazione Amazon EMR Serverless determina il tipo di processori utilizzati dall'applicazione per eseguire il processo. Amazon EMR offre due opzioni di architettura per l'applicazione: x86_64 e arm64. EMR Serverless 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 dell'utente.

Argomenti

- [Utilizzo dell'architettura x86_64](#)
- [Utilizzo dell'architettura arm64 \(Graviton\)](#)

- [Lancio di nuove applicazioni con supporto Graviton](#)
- [Configurazione delle applicazioni esistenti per l'utilizzo di Graviton](#)
- [Considerazioni sull'utilizzo di Graviton](#)

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 EMR Serverless. 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 ARM a 64 bit, passa 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 all'esecuzione di istanze di uguali dimensioni sull'architettura x86.

Utilizzo dell'architettura arm64 (Graviton)

AWS I processori Graviton sono progettati su misura AWS con core ARM Neoverse a 64 bit e sfruttano l'architettura arm64 (nota anche come Arch64 o ARM a 64 bit). La linea di processori AWS Graviton disponibile su EMR Serverless include processori Graviton3 e 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. EMR Serverless utilizza automaticamente i processori di ultima generazione quando disponibili senza alcuno sforzo da parte dell'utente per l'aggiornamento alla generazione di 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 che utilizza i processori Graviton da AWS CLI, specifica ARM64 come `architecture` parametro nell'`create-applicationAPI`. Fornisci i valori appropriati per la tua applicazione negli altri parametri.

```
aws emr-serverless create-application \  
  --name my-graviton-app \  
  --release-label emr-6.8.0 \  
  --architecture arm64
```

```
--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 l'SDK o EMR Studio. AWS CLI

Per convertire un'applicazione esistente da x86 a arm64

1. Conferma di utilizzare la versione principale più recente di [AWS CLI/SDK che supporta](#) il parametro `architecture`
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, specifica ARM64 il `architecture` parametro nell'`update-applicationAPI`.

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

4. Per verificare che l'architettura della CPU dell'applicazione sia aggiornata ARM64, utilizza l'`get-applicationAPI`.

```
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, confermare 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 ARM a 64 bit. Per informazioni su come impacchettare le librerie Python in un ambiente virtuale Python compatibile con l'architettura selezionata, fare riferimento a [Utilizzo delle librerie Python con EMR Serverless](#)

Per saperne di più, consulta il repository [AWS Graviton Getting Started](#) su GitHub. Questo repository contiene risorse essenziali che possono aiutarti a iniziare con Graviton basato su ARM.

Concorrenza dei job e accodamento per un'applicazione EMR Serverless

A partire dalla versione 7.0.0 e successive di Amazon EMR, specifica il timeout della coda di esecuzione del lavoro e la configurazione di concorrenza per la tua applicazione. Quando specifichi questa configurazione, Amazon EMR Serverless inizia mettendo in coda il lavoro e inizia l'esecuzione in base all'utilizzo simultaneo dell'applicazione. Ad esempio, se la contemporanea esecuzione di un processo è pari a 10, sull'applicazione vengono eseguiti solo dieci processi alla volta. I job rimanenti vengono messi in coda fino al termine di uno dei job in esecuzione. Se il timeout della coda viene raggiunto prima, il job scade. Per ulteriori informazioni, fare riferimento a [Job run states](#).

Principali vantaggi della concorrenza e dell'accodamento

La concorrenza e l'accodamento dei lavori offrono i seguenti vantaggi quando sono necessari molti invii di lavoro:

- Aiuta a controllare i processi di esecuzione simultanei per utilizzare in modo efficiente i limiti di capacità a livello di applicazione.

- La coda può contenere una serie improvvisa di invii di lavori, con un'impostazione di timeout configurabile.

Guida introduttiva alla concorrenza e alle code

Le procedure seguenti illustrano un paio di modi diversi per implementare la concorrenza e l'accodamento.

Usando il AWS CLI

1. Crea un'applicazione Amazon EMR Serverless con timeout di coda ed esecuzioni di job simultanee:

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

2. Aggiorna un'applicazione per modificare il timeout e la concorrenza della coda dei lavori:

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

Note

È possibile aggiornare l'applicazione esistente per abilitare la concorrenza e l'accodamento dei lavori. A tale scopo, l'applicazione deve avere un'etichetta di rilascio emr-7.0.0 o successiva.

Usando il Console di gestione AWS

Nei passaggi seguenti viene illustrato come iniziare a utilizzare la concorrenza e l'accodamento dei lavori, utilizzando: Console di gestione AWS

1. Vai a EMR Studio e scegli di creare un'applicazione con etichetta di rilascio EMR-7.0.0 o superiore.

2. In Opzioni di configurazione dell'applicazione, seleziona l'opzione Usa impostazioni personalizzate.
3. In Configurazioni aggiuntive c'è una sezione per le impostazioni del Job Run. Seleziona l'opzione Abilita la concorrenza dei lavori per abilitare la funzionalità.
4. Dopo la selezione, seleziona Esecuzioni di processi simultanee e Timeout di coda per configurare rispettivamente il numero di esecuzioni di processi simultanee e il timeout della coda. Se non si inseriscono valori per queste impostazioni, vengono utilizzati i valori predefiniti.
5. Scegliete Crea applicazione e l'applicazione verrà creata con questa funzionalità abilitata. Per verificare, vai alla dashboard, seleziona l'applicazione e controlla nella scheda delle proprietà per determinare se la funzionalità è abilitata.

Dopo la configurazione, invia lavori con questa funzionalità abilitata.

Considerazioni sulla concorrenza e sull'accodamento

Quando implementate la concorrenza e l'accodamento, tenete in considerazione quanto segue:

- Job concurrency and queuing è supportato dalla release 7.0.0 e successive di Amazon EMR.
- La concorrenza e l'accodamento dei job sono abilitati per impostazione predefinita su Amazon EMR versione 7.3.0 e successive.
- Non è possibile aggiornare la concorrenza per un'applicazione nello stato STARTED.
- L'intervallo valido per `maxConcurrentRuns` è compreso tra 1 e 1000 e compreso tra 15 e 720. `queueTimeoutMinutes`
- Lo stato IN CODA può essere impostato su un massimo di 2000 lavori per account.
- La concorrenza e l'accodamento si applicano ai lavori in batch e in streaming. Non può essere utilizzato per lavori interattivi. Per ulteriori informazioni, consulta [Esegui carichi di lavoro interattivi con EMR Serverless tramite EMR Studio](#).

Trasferisci i dati in S3 Express One Zone con EMR Serverless

Con le versioni 7.2.0 e successive di Amazon EMR, usa 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 S3GET, LIST la classe di storage chiama per tuo conto. PUT CreateSession La policy IAM deve consentire l'autorizzazione s3express:CreateSession, in modo che il connettore S3A possa richiamare l'API CreateSession. Per un esempio di politica con questa autorizzazione, consulta. [Nozioni di base su S3 Express One Zone](#)
- **S3Aconnettore:** per configurare Spark per accedere ai dati da un bucket Amazon S3 che utilizza la classe di storage S3 Express One Zone, usa il connettore Apache Hadoop. S3A Per utilizzare il connettore, assicurati che tutti gli S3 utilizzino lo schema. URIs s3a In caso contrario, modificate l'implementazione e gli schemi del filesystem che utilizzate per. s3 s3n

Per modificare lo schema s3, specifica le seguenti configurazioni del cluster:

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

Per modificare lo schema s3n, specifica le seguenti configurazioni del cluster:

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

Nozioni di base su S3 Express One Zone

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

1. [Crea un endpoint VPC](#). 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 etichetta di rilascio Amazon EMR 7.2.0 o superiore.
3. [Configura la tua applicazione](#) per utilizzare l'endpoint VPC appena creato, un gruppo di sottoreti privato e un gruppo di sicurezza.
4. Aggiungi l'`CreateSession` autorizzazione al tuo ruolo di esecuzione del lavoro.

JSON

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

5. Esegui il tuo lavoro. Nota che usa 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 tua candidatura, invia le offerte di lavoro alla candidatura. Questa sezione spiega 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](#)
- [Annullamento dell'esecuzione di un processo EMR Serverless con periodo di grazia](#)
- [Esecuzione di processi dalla console EMR Studio](#)
- [Esecuzione di lavori da AWS CLI](#)
- [Esecuzione della politica IAM](#)
- [Utilizzo di dischi ottimizzati per la riproduzione casuale](#)
- [Utilizzo dello storage serverless per Amazon EMR Serverless](#)
- [Lavori di streaming per l'elaborazione di dati in streaming continuo](#)
- [Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless](#)
- [Utilizzo delle configurazioni Hive quando si eseguono job EMR Serverless](#)
- [Resilienza EMR Serverless Job](#)
- [Configurazione Metastore per EMR Serverless](#)
- [Accesso ai dati S3 in un altro AWS account da EMR Serverless](#)
- [Risoluzione degli errori in EMR Serverless](#)
- [Abilitazione dell'allocazione dei costi a livello di Job](#)

Stati delle esecuzioni di processi

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

Le esecuzioni di processi possono avere i seguenti stati:

Stato	Description
Inviato	Lo stato iniziale del processo quando si invia un'esecuzione di lavoro a EMR Serverless. Il lavoro attende di essere pianificato per l'applicazione. EMR Serverless inizia a stabilire le priorità e a pianificare l'esecuzione del lavoro.
In coda	L'esecuzione del processo attende in questo stato quando la concorrenza di esecuzione e del processo a livello di applicazione è completamente occupata. Per ulteriori informazioni sull'accodamento e sulla concorrenza, fare riferimento a Concorrenza dei job e accodamento per un'applicazione EMR Serverless
Pending (In attesa)	Lo scheduler sta valutando l'esecuzione del job per stabilire le priorità e pianificare l'esecuzione dell'applicazione.
Pianificato	EMR Serverless ha pianificato l'esecuzione del lavoro per l'applicazione e sta allocando risorse per eseguire il lavoro.
In esecuzione	EMR Serverless ha allocato le risorse di cui il lavoro inizialmente necessita e il lavoro è in esecuzione nell'applicazione. Nelle applicazioni Spark, ciò significa che il processo del driver Spark si trova nello stato <code>running</code> .
Non riuscito	EMR Serverless non è riuscito a inviare il job run all'applicazione oppure è stato completato senza successo. <code>StateDetails</code> Per ulteriori informazioni su questo errore di lavoro, fare riferimento a.

Stato	Description
Riuscito	L'esecuzione del processo è stata completata correttamente.
Annullamento in corso	L'CancelJobRun API ha richiesto l'annullamento dell'esecuzione del lavoro oppure l'esecuzione del processo è scaduta. EMR Serverless 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.

Annullamento dell'esecuzione di un processo EMR Serverless con periodo di grazia

Nei sistemi di elaborazione dati, le interruzioni improvvise possono portare a sprechi di risorse, operazioni incomplete e potenziali incongruenze nei dati. Amazon EMR Serverless consente di specificare un periodo di prova quando si annullano le esecuzioni dei job. Questa funzionalità consente di dedicare tempo alla pulizia e al completamento adeguati dei lavori in corso prima della cessazione del lavoro.

Note

Questa funzionalità è supportata con le versioni 7.9.0 e successive di Amazon EMR.

Quando annulli l'esecuzione di un lavoro, specifica un periodo di prova (in secondi) utilizzando il parametro `shutdownGracePeriodInSeconds` durante il quale il lavoro può eseguire operazioni di pulizia prima della chiusura finale. Il comportamento e le impostazioni predefinite variano tra i processi in batch e quelli in streaming.

Periodo di grazia per i lavori in batch

Per i lavori in batch, EMR Serverless consente di implementare operazioni di pulizia personalizzate che vengono eseguite durante il periodo di prova. È possibile registrare queste operazioni di pulizia come parte dell'hook di spegnimento della JVM nel codice dell'applicazione.

Comportamento predefinito

Il comportamento predefinito per l'arresto è quello di non avere un periodo di tolleranza. Consiste nelle due azioni seguenti:

- Cessazione immediata
- Le risorse vengono rilasciate immediatamente

Opzioni di configurazione

È possibile specificare le impostazioni che comportano un arresto regolare:

- Intervallo valido per il periodo di tolleranza dello spegnimento: 15-1800 secondi (opzionale)
- Interruzione immediata (senza alcun periodo di grazia): 0 secondi

Abilita lo spegnimento graduale

Per implementare Graceful Shutdown per i lavori in batch, procedi nel seguente modo:

1. Aggiungi shutdown hook nel codice dell'applicazione contenente una logica di spegnimento personalizzata.

Example in Scala

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

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

Uso di [ShutdownHookManager](#)

Example in PySpark

```
import atexit

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

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

2. Specificate un periodo di tolleranza quando annullate il lavoro per consentire l'esecuzione degli hook aggiunti in precedenza

Esempio

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

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

Periodo di grazia per i lavori di streaming

In Spark Structured Streaming, dove i calcoli implicano la lettura o la scrittura su fonti di dati esterne, arresti improvvisi possono portare a risultati indesiderati. I processi di streaming elaborano i dati in microbatch e l'interruzione di queste operazioni a metà processo può comportare un'elaborazione duplicata nei tentativi successivi. Ciò si verifica quando l'ultimo checkpoint del microbatch precedente non è stato scritto, causando la rielaborazione degli stessi dati al riavvio del processo di streaming. Tale elaborazione duplicata non solo comporta uno spreco di risorse informatiche, ma può anche influire sulle operazioni aziendali, il che rende fondamentale evitare arresti improvvisi.

EMR Serverless offre un supporto integrato per un corretto spegnimento tramite un listener di query in streaming. Ciò garantisce il corretto completamento dei microbatch in corso prima della fine del lavoro. Il servizio gestisce automaticamente lo shutdown corretto tra i microbatch per le applicazioni

di streaming, assicurando che il microbatch corrente completi l'elaborazione, i checkpoint siano scritti correttamente e che il contesto di streaming venga terminato in modo corretto senza dover importare nuovi dati durante il processo di spegnimento.

Comportamento predefinito

- Il periodo di grazia di 120 secondi è abilitato per impostazione predefinita.
- Il listener di query in streaming integrato gestisce lo spegnimento automatico.

Opzioni di configurazione

- Intervallo valido per il periodo di tolleranza dello spegnimento: 15-1800 secondi (opzionale)
- Terminazione immediata: 0 secondi

Abilita Graceful Shutdown

Per implementare Graceful Shutdown per i lavori di streaming:

Specificate un periodo di tolleranza al momento dell'annullamento del lavoro per consentire il completamento del microbatch in corso.

Esempio

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

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

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

Aggiungi ganci di spegnimento personalizzati (opzionale)

Sebbene EMR Serverless gestisca per impostazione predefinita lo spegnimento graduale tramite il suo listener di query di streaming integrato, è possibile implementare opzionalmente una logica di spegnimento personalizzata per singole query di streaming. EMR Serverless registra il suo grazioso listener di spegnimento con priorità 60 (utilizzo). ShutdownHookManager Poiché gli hook con priorità più alta vengono eseguiti per primi, è possibile registrare le operazioni di pulizia personalizzate con una priorità superiore a 60 per garantire che vengano eseguite prima dell'inizio del processo di spegnimento di EMR Serverless.

Per aggiungere un hook personalizzato, fate riferimento al primo esempio di questo argomento che mostra come aggiungere un hook di spegnimento nel codice dell'applicazione. Qui, 100 è la priorità, che è maggiore di 60. Quindi un tale gancio di spegnimento viene eseguito per primo.

Note

Gli hook di spegnimento personalizzati sono opzionali e non necessari per una corretta funzionalità di spegnimento, che viene gestita automaticamente da EMR Serverless.

Costi del periodo di grazia e durata del Batch

Se viene utilizzato il valore predefinito per il periodo di grazia (120 secondi):

- Se la durata del batch è inferiore a 120 secondi, ti verrà addebitato solo il tempo effettivo necessario per completare il batch.
- Se la durata del batch supera i 120 secondi, ti verrà addebitato il periodo di prova massimo (120 secondi), ma la query potrebbe non essere chiusa correttamente in quanto verrà interrotta forzatamente.

Per ottimizzare i costi e garantire un arresto corretto:

- Per lotti di durata superiore a 120 secondi: valuta la possibilità di aumentare il periodo di tolleranza in modo che corrisponda alla durata del batch
- Per lotti di durata inferiore a 120 secondi: non è necessario modificare il periodo di prova, in quanto ti verrà addebitato solo il tempo di elaborazione effettivo

Considerazioni

Comportamento del periodo di grazia

- Il periodo di grazia prevede il completamento degli hook di spegnimento registrati.
- Il lavoro termina non appena il gancio di spegnimento termina, anche se è ben prima del periodo di grazia.
- Se le operazioni di pulizia superano il periodo di prova, il lavoro verrà interrotto con forza.

Comportamento del servizio

- L'arresto durante il periodo di prova è disponibile solo per i lavori nello stato RUNNING.
- Le successive richieste di annullamento durante lo stato CANCELLING vengono ignorate.
- Se EMR Serverless non riesce ad avviare l'arresto del periodo di prova a causa di errori interni del servizio:
 - Il servizio riproverà per un massimo di 2 minuti.
 - Se i nuovi tentativi non hanno esito positivo, il processo verrà interrotto con forza.

Fatturazione

I lavori vengono fatturati in base alle risorse di elaborazione utilizzate fino alla chiusura completa del lavoro, incluso il tempo impiegato durante il periodo di prova.

Esecuzione di processi dalla console EMR Studio

È possibile inviare esecuzioni di job alle applicazioni EMR Serverless e accedere ai job dalla console EMR Studio. Per creare o accedere alla tua applicazione EMR Serverless sulla console EMR Studio, segui le istruzioni in [Guida introduttiva](#) dalla console.


Invia un lavoro

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

Spark

1. Nel campo Nome, inserisci un nome per l'esecuzione del lavoro.

2. Nel campo Runtime role, inserisci il nome del ruolo IAM che l'applicazione EMR Serverless può assumere per l'esecuzione del job. Per ulteriori informazioni sui ruoli di runtime, fare riferimento a [Ruoli Job Runtime per Amazon EMR Serverless](#)
3. Nel campo Posizione dello script, inserisci la posizione Amazon S3 per lo script o il JAR che desideri eseguire. Per i lavori Spark, lo script può essere un file Python `.py ()` o un file JAR `.jar ()`.
4. Se la posizione dello script è un file JAR, 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 JAR o Python principale. 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 specificate le dimensioni del driver e dell'executor Spark, tenete conto del sovraccarico di memoria. Specificate i valori del sovraccarico di memoria nelle proprietà `spark.driver.memoryOverhead` e `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. L'esempio seguente mostra come sovrascrivere le impostazioni predefinite di Spark come la memoria dell'esecutore e del driver.

```
{
  "applicationConfiguration": [
    {
      "classification": "spark-defaults",
      "configurations": [],
    }
  ]
}
```

```
        "properties": {
            "spark.executor.memory": "8G",
            "spark.driver.memory": "6G",
            "spark.driver.cores": "2",
            "spark.executor.cores": "4"
        }
    }
]
```

- 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, fare riferimento a [Configurazione Metastore per EMR Serverless](#) Per ulteriori informazioni sulle opzioni di registrazione delle applicazioni, fare riferimento a [Archiviazione dei registri](#)
 - 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 ruolo IAM che l'applicazione EMR Serverless può assumere per l'esecuzione del job.
3. Nel campo Posizione dello script, inserisci la posizione Amazon S3 per lo script o il 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, fare riferimento a [Configurazione Metastore per EMR Serverless](#) Per ulteriori informazioni sulle opzioni di registrazione delle applicazioni, fare riferimento a [Archiviazione dei registri](#)
- Tag: assegna qualsiasi tag personalizzato all'applicazione.

5. Seleziona Submit job (Invia processo).

I job di Access vengono eseguiti

Dalla scheda Job run nella pagina Dettagli di un'applicazione, accedi alle esecuzioni dei job ed esegui le seguenti azioni per le esecuzioni dei job.

Annulla processo: per annullare l'esecuzione di un processo in questo RUNNING stato, scegliete questa opzione. Per ulteriori informazioni sulle transizioni Job Run, fare riferimento a [Stati 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 accedervi 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 [Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless](#). Per esempi su Hive, consulta [Utilizzo delle configurazioni Hive quando si eseguono job EMR Serverless](#). Questo comando restituisce il tuo `application-id`, ARN e `new.job-id`.

Ogni esecuzione di un processo ha una durata di timeout impostata. Se l'esecuzione del job supera questa durata, EMR Serverless la annulla automaticamente. Il timeout predefinito è di 12 ore. Quando avvii l'esecuzione del job, configura 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://amzn-s3-demo-bucket/scripts/create_table.sql",  
      "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/  
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/hive/  
warehouse"  
    }  
  }' \  
  --configuration-overrides '{  
    "applicationConfiguration": [{  
      "classification": "hive-site",  
      "properties": {  
        "hive.client.cores": "2",  
        "hive.client.memory": "4GIB"  
      }  
    }  
  ]  
}'
```

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 accedere a tutti i tuoi lavori, specifica il numero massimo di lavori a cui desideri accedere, fino a 50. L'esempio seguente specifica che si desidera accedere alle ultime due esecuzioni di job.

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

Per annullare un lavoro, utilizzare `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 job da AWS CLI, fare riferimento all'[EMR Serverless API Reference](#).

Esecuzione della politica IAM

È possibile specificare una policy IAM di esecuzione, oltre a un ruolo di esecuzione, quando si inviano job run su EMR Serverless. Le autorizzazioni risultanti assunte dall'esecuzione del job sono l'intersezione delle autorizzazioni nell'Execution Role e nella Execution IAM Policy specificata.

Nozioni di base

Passaggi per utilizzare la politica Execution IAM:

Crea un'emr-serverless applicazione o usane una esistente, quindi esegui la seguente riga di comando aws per avviare un job run con una policy IAM in linea:

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

```
--application-id application-id \  
--execution-role-arn execution-role-arn \  
--job-driver job-driver-options \  
--execution-iam-policy '{"policy": "inline-policy"}'
```

Esempi di comandi CLI

Se abbiamo la seguente politica memorizzata nel `policy.json` file sul computer:

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetObject",  
        "s3:ListBucket"  
      ],  
      "Resource": [  
        "arn:aws:s3::my-test-bucket",  
        "arn:aws:s3::my-test-bucket/*"  
      ],  
      "Sid": "AllowS3GetObject"  
    }  
  ]  
}
```

Quindi possiamo iniziare un lavoro con questa politica usando il seguente AWS CLI comando:

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

Puoi anche utilizzare entrambe AWS le politiche gestite dal cliente, specificandole ARNs tramite:

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

È possibile specificare sia una policy IAM in linea che una policy gestita ARNs nella stessa richiesta:

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

Note importanti

- Il policy campo `execution-role-policy's` può avere una lunghezza massima di 2048 caratteri.
- La stringa di policy IAM in linea specificata nel policy campo `execution-iam-policy's` deve essere conforme allo standard delle stringhe json, senza che vengano evitate nuove righe e virgolette come nell'esempio precedente.
- È ARNs possibile specificare un elenco di un massimo di 10 policy gestite come valore nel campo's. `execution-iam-policy policyArns`
- La policy gestita ARNs deve essere un elenco di ARN di policy gestite dal cliente valide AWS o gestite dal cliente. Quando viene specificato un ARN di policy gestita dal cliente, la policy deve appartenere allo stesso AWS account dell'EMR-S. JobRun
- Quando vengono utilizzate sia la policy IAM in linea che le policy gestite, il testo semplice utilizzato per la combinazione delle policy inline e gestite non può superare i 2.048 caratteri.

- Le autorizzazioni risultanti assunte da sono l'intersezione delle JobRun autorizzazioni nell'Execution Role e nella Execution IAM Policy specificata.

Intersezione delle politiche

Le autorizzazioni risultanti assunte dall'esecuzione del job sono l'intersezione delle autorizzazioni nell'Execution Role e nella Execution IAM Policy specificata. Ciò significa che qualsiasi autorizzazione richiesta dovrà essere specificata in entrambi i luoghi per JobRun funzionare. Tuttavia, è possibile specificare un'ulteriore dichiarazione di autorizzazione generale nella politica in linea per tutte le autorizzazioni che non intendi aggiornare o sovrascrivere.

Esempio

Data la seguente politica di esecuzione del ruolo IAM:

JSON

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

```

    "Effect": "Allow",
    "Action": [
      "dynamodb:DescribeTable"
    ],
    "Resource": [
      "arn:aws:dynamodb:*:*:table/MyCompany1table"
    ],
    "Sid": "AllowDYNAMODBDescribetable"
  }
]
}

```

E la seguente politica IAM in linea:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3::my-test-bucket/tenant1",
        "arn:aws:s3::my-test-bucket/tenant1/*"
      ],
      "Sid": "AllowS3GetObject"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:*",
        "dynamodb:*"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowLOGS"
    }
  ]
}

```

```
]
}
```

Le autorizzazioni risultanti assunte da JobRun sono:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket/tenant1",
        "arn:aws:s3:::my-test-bucket/tenant1/*"
      ],
      "Sid": "AllowS3GetObject"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:123456789012:log-group::log-stream"
      ],
      "Sid": "AllowLOGSDescribeLogGroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable"
      ],
      "Resource": [
        "arn:aws:dynamodb:*:*:table/MyCompany1table"
      ],
      "Sid": "AllowDYNAMODescribeTable"
    }
  ]
}
```

```
]
}
```

Utilizzo di dischi ottimizzati per la riproduzione casuale

Con le versioni 7.1.0 e successive di Amazon EMR, usa dischi ottimizzati per lo shuffle quando esegui processi Apache Spark o Hive per migliorare le prestazioni (per le operazioni al secondo) per uno spostamento più rapido dei dati e una latenza ridotta durante I/O-intensive workloads. Compared to standard disks, shuffle-optimized disks provide higher IOPS (I/O le operazioni di shuffle. I dischi ottimizzati per Shuffle ti consentono di collegare dischi di dimensioni fino a 2 TB per lavoratore, quindi configura la capacità appropriata per i tuoi requisiti di carico di lavoro.

Vantaggi principali

I dischi ottimizzati per lo shuffle offrono i seguenti vantaggi.

- Prestazioni IOPS elevate: i dischi ottimizzati per lo shuffle offrono IOPS più elevati 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, quindi scegli la capacità appropriata in base ai tuoi carichi di lavoro.

Nozioni di base

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

Spark

1. Creare 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 da eseguire con dischi ottimizzati `spark.emr-serverless.driver.disk.type` and/or `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 le [proprietà del lavoro di Spark](#).

Hive

1. Creare 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 da eseguire con dischi ottimizzati `hive.driver.disk.type` and/or `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.q1",
        "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 release 7.1.0 di Amazon EMR. 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 vCPU e 1024 GB di memoria. 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"
}'
```

Utilizzo dello storage serverless per Amazon EMR Serverless

Con le versioni 7.12 e successive di Amazon EMR, utilizza lo storage serverless quando esegui i job Apache Spark per eliminare il provisioning locale del disco, ridurre i costi di elaborazione dei dati e prevenire gli errori dei lavori dovuti ai vincoli di capacità del disco. Lo storage serverless gestisce automaticamente le operazioni di shuffle, perdita del disco e memorizzazione nella cache del disco per i processi senza richiedere la configurazione della capacità e archivia i dati intermedi senza alcun costo. Amazon EMR Serverless archivia i dati intermedi in uno storage serverless completamente gestito che si ridimensiona automaticamente in base alle richieste del carico di lavoro e consente a Spark di rilasciare gli elaboratori immediatamente quando sono inattivi, riducendo i costi di elaborazione.

Vantaggi principali

Lo storage serverless per EMR Serverless offre i seguenti vantaggi.

- **Zero-configuration storage:** lo storage serverless elimina la necessità di configurare il tipo e la dimensione del disco locale per ogni applicazione o processo. EMR Serverless gestisce automaticamente le operazioni intermedie sui dati senza pianificazione della capacità.

- Previene gli errori dei processi grazie alla scalabilità automatica: la capacità di storage si ridimensiona automaticamente in base alla richiesta del carico di lavoro, evitando errori di lavoro dovuti all'insufficiente capacità del disco.
- Costi di elaborazione dei dati ridotti: lo storage serverless riduce i costi di elaborazione attraverso due meccanismi. Innanzitutto, lo storage intermedio dei dati viene fornito gratuitamente: si paga solo per le risorse di calcolo e di memoria. In secondo luogo, lo storage disaccoppiato con l'allocazione dinamica delle risorse di Spark consente a Spark di rilasciare immediatamente i dipendenti quando sono inattivi anziché conservarli per conservare i dati intermedi sui dischi locali. Ciò consente una scalabilità orizzontale e orizzontale più rapide per fase Spark, riducendo i costi di elaborazione per i lavori in cui le fasi successive richiedono meno lavoratori rispetto alle fasi iniziali.
- Storage crittografato con isolamento a livello di processo: tutti i dati intermedi vengono crittografati in transito e a riposo con un rigoroso isolamento a livello di processo.
- Fine-grained supporto per il controllo degli accessi: lo storage serverless supporta il controllo granulare degli accessi tramite l'integrazione di Lake Formation AWS .

Nozioni di base

Consulta i seguenti passaggi per utilizzare lo storage serverless per EMR Serverless nei flussi di lavoro Spark.

1. Creare un'applicazione EMR Serverless

Crea un'applicazione EMR Serverless release 7.12 (o successiva) con storage serverless abilitato impostando la proprietà spark su true nella classificazione spark-defaults **spark.aws.serverlessStorage.enabled**.

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

2. Avvia un job Spark

Avvia un job eseguito sulla tua applicazione. Storage serverless per EMR Serverless gestisce automaticamente le operazioni intermedie sui dati, ad esempio lo shuffle per il lavoro.

```
aws emr-serverless start-job-run \
  --application-id <application-id> \
  --execution-role-arn <job-role-arn> \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://<bucket>/script.py",
      "sparkSubmitParameters": "--conf spark.executor.cores=4
        --conf spark.executor.memory=20g
        --conf spark.driver.cores=4
        --conf spark.driver.memory=8g
        --conf spark.executor.instances=10"
    }
  }'
```

È inoltre possibile abilitare lo storage serverless per EMR Serverless a livello di processo anche quando non è abilitato a livello di applicazione. In questo modo verranno avviati nodi di lavoro dotati di storage serverless per elaborare i lavori. Puoi anche disabilitare lo storage serverless per un lavoro specifico impostando la stessa proprietà **spark.aws.serverlessStorage.enabled** Spark su false.

```
# Turn on serverless storage for EMR serverless for a specific job
aws emr-serverless start-job-run \
  --application-id <application-id> \
  --execution-role-arn <job-role-arn> \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
      "entryPointArguments": ["1"],
      "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi
        --conf spark.aws.serverlessStorage.enabled": "true"
    }
  }'
```

Note

Per continuare a utilizzare il tradizionale provisioning locale del disco, ometti la **spark.aws.serverlessStorage.enabled** configurazione o impostala su false.

Considerazioni e limitazioni

- **Versione di rilascio:** lo storage serverless è supportato su Amazon EMR versione 7.12 e successive.
- **Limiti del volume di dati:** ogni processo può leggere e scrivere fino a un totale di 200 GB di dati intermedi per esecuzione del processo. I lavori che superano questo limite avranno esito negativo e verrà visualizzato un messaggio di errore che indica che è stato raggiunto il limite di storage serverless.
- **Job Execution Timeout:** lo storage serverless supporta lavori con timeout di esecuzione fino a 24 ore. I lavori configurati per timeout di esecuzione più lunghi falliranno e verrà visualizzato un messaggio di errore.
- **Pre-initialized capacità:** i Pre-initialized capacity worker non supportano lo storage senza server. Quando si configura la capacità preinizializzata, questa verrà utilizzata solo dai processi che disabilitano esplicitamente lo storage serverless a livello di processo. I lavori con storage serverless abilitato forniranno sempre nuovi lavoratori su richiesta e non utilizzeranno alcuna capacità preinizializzata, indipendentemente dalla configurazione a livello di applicazione.
- **Tipi di carichi di lavoro:** lo storage serverless non è supportato per lo streaming e i lavori interattivi.
- **Configurazione Worker:** lo storage serverless non è supportato per i lavoratori con 1 o 2 vCPU.

Supportato Regioni AWS

EMR Serverless supporta lo storage serverless nelle seguenti regioni:

- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti orientali (Ohio)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Africa (Città del Capo)

- Asia Pacifico (Hong Kong)
- Asia Pacifico (Giacarta)
- Asia Pacifico (Melbourne)
- Asia Pacifico (Mumbai)
- Asia Pacifico (Osaka)
- Asia Pacifico (Seoul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Canada occidentale (Calgary)
- Europa (Francoforte)
- Europa (Irlanda)
- Europa (Londra)
- Europa (Milano)
- Europa (Parigi)
- Europa (Spagna)
- Europa (Stoccolma)
- Europa (Zurigo)
- Sud America (San Paolo)

Lavori di streaming per l'elaborazione di dati in streaming continuo

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 man mano che i dati arrivano. 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. I job di streaming EMR Serverless 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:** utilizza 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 lavori 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, quindi esegui 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.

Vantaggi principali

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

- **Auto-retry**— 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 commuta automaticamente i job di streaming su una zona di disponibilità 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

EMR Serverless 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 versioni di Amazon EMR 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:** tabelle AWS Glue Data Catalog, Amazon S3, Amazon Redshift, MySQL, PostgreSQL Oracle, Oracle, Microsoft SQL, 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 con le [versioni 7.1.0 e successive di Amazon EMR](#).
- EMR Serverless 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 lavoro.
- [I job di streaming sono compatibili solo con il motore Spark, che si basa sul framework di streaming strutturato.](#)
- EMR Serverless riprova a tempo indeterminato i processi di streaming e non è possibile 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 processi di streaming hanno punti di controllo per salvare lo stato di esecuzione 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 [Amazon EMR versione 7.1.0](#) o successiva.
2. Una volta che l'applicazione è pronta, imposta il mode parametro per STREAMING inviare un lavoro in streaming, simile all'esempio seguente AWS CLI .

```
aws emr-serverless start-job-run \  
--application-id <APPLICATION_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 throughput di lettura dedicata fino

a per shard. 2MB/second 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 pagina [spark-sql-kinesis-connector](#) 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 connetterti a Kinesis Data Streams, configura l'applicazione EMR Serverless con accesso VPC e utilizza un endpoint VPC per consentire l'accesso privato oppure utilizza un gateway NAT per ottenere l'accesso pubblico. Per ulteriori informazioni, consulta [Configurazione dell'accesso al 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 pagina [spark-sql-kinesis-connector](#) 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 Kafka con l'autenticazione IAM, includi un'altra dipendenza per consentire al connettore Kafka di connettersi ad Amazon MSK con IAM. [Per ulteriori informazioni, consulta il repository aws-msk-iam-auth su GitHub](#) È inoltre necessario assicurarsi che il ruolo di job runtime disponga delle autorizzazioni IAM necessarie. L'esempio seguente mostra come utilizzare il connettore con l'autenticazione IAM.

```
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 autenticazione IAM di Amazon MSK, configura l'applicazione EMR Serverless con accesso VPC. Le sottoreti devono avere accesso a Internet e utilizzare un gateway NAT per accedere alle dipendenze Maven. Per ulteriori informazioni, consulta [Configurazione dell'accesso al 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

I job di streaming supportano la rotazione dei log per i log delle applicazioni Spark e i log degli eventi e la compattazione dei log per i log degli eventi Spark. Questo ti aiuta a gestire le tue risorse in modo efficace.

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, fare riferimento a [Rotazione dei registri](#).

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 significativamente 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.

Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless

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 rilascio di Amazon EMR. Ad esempio, quando esegui job con Amazon EMR release 6.6.0, il job 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 usi l'[StartJobRunAPI](#) per eseguire un job Spark, specifica i seguenti parametri.

Parametri obbligatori

- [Ruolo di runtime del job Spark](#)
- [Parametro Spark Job Driver](#)
- [Parametro di override della configurazione Spark](#)
- [Ottimizzazione dinamica dell'allocazione delle risorse Spark](#)

Ruolo di runtime del job Spark

executionRoleArn Da utilizzare per specificare l'ARN per il ruolo IAM 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 dai bucket o dai prefissi S3 in cui risiede lo script o il file JAR PySpark
- Scrivi nei bucket S3 dove intendi scrivere l'output finale
- Scrivi i log in un bucket o prefisso S3 che specifica `S3MonitoringConfiguration`
- Accesso alle chiavi KMS se utilizzi le chiavi KMS per crittografare i dati nel tuo bucket S3
- Accesso al AWS Glue Data Catalog se usi SparkSQL

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

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 utilizzare questo tipo di lavoro per eseguire Scala PySpark, Java 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 JAR o Python principale. 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 JAR o Python principale che desideri eseguire. Se stai usando un JAR Scala o Java, specifica la classe di ingresso principale nell'argomento `SparkSubmitParameters` using the `--class`.

Per ulteriori informazioni, consulta [Avvio di applicazioni con spark-submit](#).

Parametro di override della configurazione Spark

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

- **monitoringConfiguration**- Utilizza questo campo per specificare l'URL di Amazon S3 (`s3MonitoringConfiguration`) in cui desideri che il job EMR Serverless memorizzi i log del tuo job Spark. Assicurati di aver creato questo bucket con lo stesso Account AWS che ospita l'applicazione e nello stesso luogo in cui è in esecuzione il processo. Regione AWS
- **applicationConfiguration**— Per sovrascrivere le configurazioni predefinite per le applicazioni, puoi fornire un oggetto di configurazione in questo campo. Puoi utilizzare una sintassi abbreviata per fornire la configurazione oppure fare riferimento all'oggetto di configurazione in un file JSON. Gli oggetti di configurazione sono composti da una classificazione, proprietà e

configurazioni nidificate opzionali. Le proprietà sono costituite dalle impostazioni che desideri ignorare in un dato file. Puoi specificare diverse classificazioni per più applicazioni in un singolo oggetto JSON.

Note

Le classificazioni di configurazione disponibili variano in base alla specifica release EMR Serverless. Ad esempio, le classificazioni per Log4j personalizzate `spark-executor-log4j2` sono disponibili solo con le versioni `spark-driver-log4j2 6.8.0` 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 utilizzate da Amazon EMR 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 lavoro, consulta [Configurazione predefinita dell'applicazione per EMR Serverless](#)

Ottimizzazione dinamica dell'allocazione delle risorse Spark

Utilizzalo `dynamicAllocationOptimization` per ottimizzare l'utilizzo delle risorse in EMR Serverless. L'impostazione di questa proprietà `true` nella classificazione della configurazione 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 EMR Serverless crea e rilascia i lavoratori. 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 rilascio di Amazon EMR.

Di seguito è riportato un esempio di classificazione della configurazione `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, suggeriamo 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 notare 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 portare a un minor numero di lavoratori disponibili per fasi più grandi, 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.

Proprietà Spark opzionali e valori predefiniti

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

Chiave	Description	Valore predefinito
	Questo vale solo se attivi l'allocazione dinamica.	
<code>spark.dynamicAllocation.initialExecutors</code>	Il numero iniziale di esecutori da eseguire se si attiva l'allocazione dinamica.	3
<code>spark.dynamicAllocation.maxExecutors</code>	Il limite superiore per il numero di esecutori se si attiva l'allocazione dinamica.	Per 6.10.0 e versioni successive, <code>infinity</code> Per 6.9.0 e versioni precedenti, <code>100</code>
<code>spark.dynamicAllocation.minExecutors</code>	Il limite inferiore per il numero di esecutori se si attiva l'allocazione dinamica.	0
<code>spark.emr-serverless.allocation.batch.size</code>	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
<code>spark.emr-serverless.driver.disk</code>	Il disco del driver Spark.	20G
<code>spark.emr-serverless.driverEnv</code> [KEY]	Opzione che aggiunge variabili di ambiente al driver Spark.	NULL
<code>spark.emr-serverless.executor.disk</code>	Il disco Spark Executor.	20G
<code>spark.emr-serverless.memoryOverheadFactor</code>	Imposta il sovraccarico di memoria da aggiungere alla memoria del contenitore del driver e dell'executor.	0.1

Chiave	Description	Valore predefinito
<code>spark.emr-serverless.driver.disk.type</code>	Il tipo di disco collegato al driver Spark.	Standard
<code>spark.emr-serverless.executor.disk.type</code>	Il tipo di disco collegato agli esecutori Spark.	Standard
<code>spark.executor.cores</code>	Il numero di core utilizzati da ogni executor.	4
<code>spark.executor.extraJavaOptions</code>	Opzioni Java aggiuntive per l'esecutore Spark.	NULL
<code>spark.executor.instances</code>	Il numero di contenitori Spark Executor da allocare.	3
<code>spark.executor.memory</code>	La quantità di memoria utilizzata da ogni executor.	14 G
<code>spark.executorEnv. [KEY]</code>	Opzione che aggiunge variabili di ambiente agli esecutori Spark.	NULL
<code>spark.files</code>	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 <code>SparkFiles.get(<i>fileName</i>)</code>	NULL
<code>spark.hadoop.hive.metastore.client.factory.class</code>	La classe di implementazione del metastore Hive.	NULL

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

La tabella seguente elenca i parametri di invio Spark predefiniti.

Parametri di invio Spark predefiniti

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

Chiave	Description	Valore predefinito
<code>driver-cores</code>	Il numero di core utilizzati dal driver.	4
<code>driver-memory</code>	La quantità di memoria utilizzata dal driver.	14 G
<code>executor-cores</code>	Il numero di core utilizzati da ogni esecutore.	4
<code>executor-memory</code>	La quantità di memoria utilizzata dall'esecutore.	14 G
<code>files</code>	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'esecutore con <code>SparkFile s.get(<i>fileName</i>)</code>	NULL
<code>jars</code>	Un elenco di jar separati da virgole da includere nei percorsi di classe del driver e dell'executor.	NULL
<code>num-executors</code>	Il numero di esecutori da avviare.	3
<code>py-files</code>	Un elenco separato da virgole di <code>.zip.egg</code> , o <code>.py</code> file da inserire nelle app per PYTHONPATH Python.	NULL
<code>verbose</code>	Opzione che attiva un output di debug aggiuntivo.	NULL

Best practice per la configurazione delle risorse

Configurazione delle risorse dei driver e degli esecutori tramite l'API StartJobRun

Note

I core del driver e dell'esecutore Spark e le proprietà di memoria, se specificate, devono essere specificate direttamente nella richiesta API. StartJobRun

La configurazione delle risorse in questo modo garantisce che EMR Serverless possa allocare le risorse corrette prima di eseguire il lavoro. Ciò è in contrasto con le impostazioni fornite nello script utente, ad esempio nel file .py o .jar, che vengono valutate troppo tardi, poiché i driver e gli executor a volte vengono predisposti prima dell'inizio dell'esecuzione dello script. Esistono due modi supportati per configurare queste risorse durante l'invio del lavoro:

Opzione 1: usa spark SubmitParameters

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

Opzione 2: usa ConfigurationOverrides per la classificazione spark-defaults

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

```

    }
  ]
}

```

Esempi di Spark

L'esempio seguente mostra come utilizzare l'StartJobRunAPI per eseguire uno script Python. Per un tutorial completo che utilizza questo esempio, consulta [Inizia a usare Amazon EMR Serverless](#). Puoi trovare altri esempi su come eseguire PySpark lavori e aggiungere dipendenze Python nel repository [EMR Serverless 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://amzn-s3-demo-destination-bucket/
wordcount_output"],
      "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g --
conf spark.executor.instances=1"
    }
  }'

```

L'esempio seguente mostra come utilizzare l'StartJobRunAPI per eseguire uno Spark 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"
    }
  }'

```

Utilizzo delle configurazioni Hive quando si eseguono job EMR Serverless

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 rilascio di Amazon EMR. Ad esempio, quando esegui lavori su un'applicazione con Amazon EMR release 6.6.0, il processo 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 utilizzi l'[StartJobRunAPI](#) per eseguire un job Hive, specifica i seguenti parametri.

Parametri obbligatori

- [Ruolo di runtime del job Hive](#)
- [Parametro Hive Job Driver](#)
- [Parametro di sovrascrittura della configurazione Hive](#)

Ruolo di runtime del job Hive

executionRoleArn Da utilizzare per specificare l'ARN per il ruolo IAM 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 chiavi KMS se utilizzi le chiavi KMS per crittografare i dati nel tuo bucket S3
- Accesso al AWS Glue Data Catalog

Se il tuo job Hive legge o scrive dati da o verso altre fonti di dati, specifica le autorizzazioni appropriate in questo ruolo IAM. Se non fornisci queste autorizzazioni al ruolo IAM, il tuo lavoro potrebbe fallire. Per ulteriori informazioni, vedi [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 si specifica `hive` come tipo di lavoro, EMR Serverless passa una query Hive al parametro. `jobDriver` 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 oggetto JSON con i due campi seguenti:

- **monitoringConfiguration**— Utilizza questo campo per specificare l'URL di Amazon S3 (`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 l'applicazione e nello stesso luogo in cui è in esecuzione il processo. Regione AWS
- **applicationConfiguration**— È possibile fornire un oggetto di configurazione in questo campo per sovrascrivere le configurazioni predefinite per le applicazioni. Puoi utilizzare una sintassi abbreviata per fornire la configurazione oppure fare riferimento all'oggetto di configurazione in un file JSON. Gli oggetti di configurazione sono composti da una classificazione, proprietà e configurazioni nidificate opzionali. Le proprietà sono costituite dalle impostazioni che desideri ignorare in un dato file. Puoi specificare diverse classificazioni per più applicazioni in un singolo oggetto JSON.

Note

Le classificazioni di configurazione disponibili variano in base alla specifica release EMR Serverless. Ad esempio, le classificazioni per Log4j personalizzate `spark-executor-log4j2` sono disponibili solo con le versioni `spark-driver-log4j2 6.8.0` 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, fare riferimento a.

[Configurazione predefinita dell'applicazione per EMR Serverless](#)

Proprietà del lavoro di Hive

La tabella seguente elenca le proprietà obbligatorie che si configurano quando si invia un lavoro Hive.

Proprietà obbligatorie del lavoro Hive

Impostazione	Description
<code>hive.exec.scratchdir</code>	La posizione Amazon S3 in cui EMR Serverless crea file temporanei durante l'esecuzione del job Hive.
<code>hive.metastore.warehouse.dir</code>	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.

Proprietà opzionali di Hive e valori predefiniti

Impostazione	Description	Valore predefinito
<code>fs.s3.customAWSCredentialsProvider</code>	Il provider di AWS credenziali che desideri utilizzare.	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>
<code>fs.s3a.aws.credentials.provider</code>	Il provider di AWS credenziali che desideri utilizzare con un file system S3A.	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>
<code>hive.auto.convert.join</code>	Opzione che attiva la conversione automatica dei join comuni in mapjoin, in base alla dimensione del file di input.	TRUE
<code>hive.auto.convert.join.noconditionaltask</code>	Opzione che attiva l'ottimizzazione quando Hive converte un common join in un mapjoin in base alla dimensione del file di input.	TRUE
<code>hive.auto.convert.join.noconditionaltask.size</code>	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
<code>hive.cbo.enable</code>	Opzione che attiva ottimizzazioni basate sui costi con il framework Calcite.	TRUE
<code>hive.cli.tez.session.async</code>	Opzione per avviare una sessione Tez in background durante la compilazione della query Hive. Se impostato	TRUE

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

Impostazione	Description	Valore predefinito
<code>hive.driver.memory</code>	La quantità di memoria da utilizzare per ogni processo del driver Hive. Hive CLI e Tez Application Master condividono questa memoria in parti uguali con il 20% di spazio di crescita.	6 G
<code>hive.emr-serverless.launch.env.[KEY]</code>	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.	
<code>hive.exec.dynamic.partition</code>	Opzioni che attivano le partizioni dinamiche in DML/DDL.	TRUE
<code>hive.exec.dynamic.partition.mode</code>	Opzione che specifica se si desidera utilizzare la modalità rigorosa o la modalità non rigorosa. In modalità rigorosa, specificate almeno una partizione statica in caso di sovrascrittura accidentale di tutte le partizioni. In modalità non rigorosa, tutte le partizioni possono essere dinamiche.	strict
<code>hive.exec.max.dynamic.partitions</code>	Il numero massimo di partizioni dinamiche create da Hive in totale.	1000
<code>hive.exec.max.dynamic.partitions.per.node</code>	Numero massimo di partizioni dinamiche che Hive crea in ogni nodo mapper e reducer.	100

Impostazione	Description	Valore predefinito
<code>hive.exec.orc.split.strategy</code>	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 precedenti basate sull'euristica.	HYBRID
<code>hive.exec.reducers.bytes.per.reducer</code>	La dimensione di ogni riduttore . L'impostazione predefinita è 256 MB. Se la dimensione di input è 1G, il job utilizza 4 riduttori.	256000000
<code>hive.exec.reducers.max</code>	Il numero massimo di riduttori.	256
<code>hive.exec.stagingdir</code>	Il nome della directory che memorizza i file temporanei creati da Hive all'interno delle posizioni delle tabelle e nella posizione della directory scratch specificata nella proprietà. <code>hive.exec.scratchdir</code>	<code>.hive-staging</code>

Impostazione	Description	Valore predefinito
<code>hive.fetch.task.conversion</code>	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
<code>hive.groupby.position.alias</code>	Opzione che fa sì che Hive utilizzi un alias di posizione della colonna nelle istruzioni. GROUP BY	FALSE
<code>hive.input.format</code>	Il formato di input predefinito. Imposta su HiveInputFormat se riscontri problemi conCombineHiveInputFormat.	<code>org.apache.hadoop.hive.q1.io.CombineHiveInputFormat</code>
<code>hive.log.explain.output</code>	Opzione che attiva le spiegazioni dell'output esteso per qualsiasi query nel registro di Hive.	FALSE
<code>hive.log.level</code>	Il livello di registrazione di Hive.	INFO
<code>hive.mapred.reduce.tasks.speculative.execution</code>	Opzione che attiva il lancio speculativo dei riduttori. Supportato solo con Amazon EMR 6.10.x e versioni precedenti.	TRUE

Impostazione	Description	Valore predefinito
<code>hive.max-task-containers</code>	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
<code>hive.merge.mapfiles</code>	Opzione che causa l'unione di file di piccole dimensioni alla fine di un lavoro basato sulla sola mappa.	TRUE
<code>hive.merge.size.per.task</code>	La dimensione dei file uniti alla fine del lavoro.	256000000
<code>hive.merge.tezfiles</code>	Opzione che attiva l'unione di file di piccole dimensioni alla fine di un Tez DAG.	FALSE
<code>hive.metastore.client.factory.class</code>	Il nome della classe factory che produce oggetti che implementano l'interfaccia. <code>IMetaStoreClient</code>	<code>com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory</code>
<code>hive.metastore.glue.catalogid</code>	Se il AWS Glue Data Catalog funge da metastore ma viene eseguito in un ambiente Account AWS diverso dai lavori, l'ID del Account AWS luogo in cui i lavori sono in esecuzione.	NULL

Impostazione	Description	Valore predefinito
<code>hive.metastore.uris</code>	L'URI parsimonioso utilizzato dal client metastore per connettersi al metastore remoto.	NULL
<code>hive.optimize.ppd</code>	Opzione che attiva il pushdown dei predicati.	TRUE
<code>hive.optimize.ppd.storage</code>	Opzione che attiva il pushdown dei predicati verso i gestori di archiviazione.	TRUE
<code>hive.orderby.position.alias</code>	Opzione che fa sì che Hive utilizzi un alias di posizione della colonna nelle istruzioni. ORDER BY	TRUE
<code>hive.prewarm.enabled</code>	Opzione che attiva il preriscaldamento del contenitore per Tez.	FALSE
<code>hive.prewarm.numcontainers</code>	Il numero di contenitori da preriscaldare per Tez.	10
<code>hive.stats.autogather</code>	Opzione che fa sì che Hive raccolga automaticamente le statistiche di base durante il comando. INSERT OVERWRITE	TRUE

Impostazione	Description	Valore predefinito
<code>hive.stats.fetch.column.stats</code>	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
<code>hive.stats.gather.num.threads</code>	Il numero di thread utilizzati dai comandi <code>partialscan</code> e <code>noscan analysis</code> per le tabelle partizionate. Questo vale solo per i formati di file che implementano <code>StatsProvidingRecordReader</code> (come ORC).	10
<code>hive.strict.checks.cartesian.product</code>	Opzioni che attivano rigorosi controlli cartesiani di join. Questi controlli non consentono un prodotto cartesiano (un cross join).	FALSE
<code>hive.strict.checks.type.safety</code>	Opzione che attiva rigorosi controlli di sicurezza tipografici e disattiva il confronto tra entrambi <code>string</code> e <code>bigint</code> e <code>double</code> .	TRUE

Impostazione	Description	Valore predefinito
<code>hive.support.quote d.identifiers</code>	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
<code>hive.tez.auto.redu cer.parallelism</code>	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
<code>hive.tez.container .size</code>	La quantità di memoria da utilizzare per il processo di operazione Tez.	6144
<code>hive.tez.cpu.vcores</code>	Il numero di core da utilizzare per ogni attività Tez.	2
<code>hive.tez.disk.size</code>	La dimensione del disco per ogni contenitore di attività.	20G
<code>hive.tez.input.for mat</code>	Il formato di input per la generazione di split in Tez AM.	<code>org.apache.hadoop. hive.q1.io.HiveInp utFormat</code>

Impostazione	Description	Valore predefinito
<code>hive.tez.min.partition.factor</code>	Limite inferiore di riduttori specificato da Tez quando si attiva il parallelismo del riduttore automatico.	0.25
<code>hive.vectorized.execution.enabled</code>	Opzione che attiva la modalità vettoriale di esecuzione delle query.	TRUE
<code>hive.vectorized.execution.reduce.enabled</code>	Opzione che attiva la modalità vettoriale del lato ridotto dell'esecuzione di una query.	TRUE
<code>javax.jdo.option.ConnectionDriverName</code>	Il nome della classe del driver per un metastore JDBC.	<code>org.apache.derby.jdbc.EmbeddedDriver</code>
<code>javax.jdo.option.ConnectionPassword</code>	La password associata a un database metastore.	NULL
<code>javax.jdo.option.ConnectionURL</code>	La stringa di connessione JDBC per un metastore JDBC.	<code>jdbc:derby;;databaseName=metastore_db;create=true</code>
<code>javax.jdo.option.ConnectionUserName</code>	Il nome utente associato a un database metastore.	NULL
<code>mapreduce.input.fileinputformat.split.maxsize</code>	La dimensione massima di una divisione durante il calcolo della suddivisione quando il formato di input è <code>org.apache.hadoop.hive.q1.io.CombineHiveInputFormat</code> . Il valore 0 indica l'assenza di limiti.	0

Impostazione	Description	Valore predefinito
<code>tez.am.dag.cleanup.on.completion</code>	Opzione che attiva la pulizia dei dati casuali al termine del DAG.	TRUE
<code>tez.am.emr-serverless.launch.env.[KEY]</code>	Opzione per impostare la variabile di <code>KEY</code> ambiente nel processo Tez AM. Per Tez AM, questo valore sostituisce il valore <code>hive.emr-serverless.launch.env.[KEY]</code>	
<code>tez.am.log.level</code>	Il livello di registrazione root che EMR Serverless passa all'app Tez principale.	INFO
<code>tez.am.sleep.time.before.exit.millis</code>	EMR Serverless dovrebbe inviare gli eventi ATS dopo questo periodo di tempo successivo alla richiesta di spegnimento AM.	0
<code>tez.am.speculation.enabled</code>	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	Description	Valore predefinito
<code>tez.am.task.max.failed.attempts</code>	Il numero massimo di tentativi che possono fallire per una determinata attività prima che l'attività fallisca. Questo numero non include i tentativi terminati manualmente.	3
<code>tez.am.vertex.cleanup.height</code>	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. Le versioni 6.8.0 e successive di Amazon EMR supportano questa funzionalità.	0
<code>tez.client.asynchronous-stop</code>	Opzione che fa sì che EMR Serverless invii gli eventi ATS prima che termini il driver Hive.	FALSE
<code>tez.grouping.max-size</code>	Il limite massimo di dimensione (in byte) di una divisione raggruppata. Questo limite impedisce suddivisioni eccessivamente grandi.	1073741824
<code>tez.grouping.min-size</code>	Il limite di dimensione inferiore (in byte) di una divisione raggruppata. Questo limite impedisce troppe suddivisioni di piccole dimensioni.	16777216

Impostazione	Description	Valore predefinito
<code>tez.runtime.io.sort.mb</code>	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
<code>tez.runtime.unordered.output.buffer.size-mb</code>	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
<code>tez.shuffle-vertex-manager.max-src-fraction</code>	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 connessione). ScatterGather Il numero di attività pronte per la pianificazione sul vertice corrente varia linearmente tra <code>e.min-fraction</code> e <code>e.max-fraction</code> . Il valore predefinito è il valore predefinito o, a seconda del valore maggiore. <code>tez.shuffle-vertex-manager.min-src-fraction</code>	0.75
<code>tez.shuffle-vertex-manager.min-src-fraction</code>	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	Description	Valore predefinito
<code>tez.task.emr-serverless.launch.env.[<i>KEY</i>]</code>	Opzione per impostare la variabile di <i>KEY</i> ambiente nel processo di attività Tez. Per le attività Tez, questo valore ha la precedenza sul valore. <code>hive.emr-serverless.launch.env.[<i>KEY</i>]</code>	
<code>tez.task.log.level</code>	Il livello di registrazione principale che EMR Serverless passa alle attività Tez.	INFO
<code>tez.yarn.ats.event.flush.timeout.millis</code>	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 l'API StartJobRunAPI.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
      "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-
query.q1",
      "parameters": "--hiveconf hive.log.explain.output=false"
    }
  }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
      "classification": "hive-site",
      "properties": {
```

```

        "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-hive/
hive/scratch",
        "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/warehouse",
        "hive.driver.cores": "2",
        "hive.driver.memory": "4g",
        "hive.tez.container.size": "4096",
        "hive.tez.cpu.vcores": "1"
    }
}
}'

```

È possibile trovare altri esempi di come eseguire i job Hive nel repository [EMR Serverless GitHub Samples](#).

Resilienza EMR Serverless Job

Le versioni 7.1.0 e successive di EMR Serverless includono il supporto per la resilienza dei job, in modo da riprovare automaticamente tutti i job non riusciti senza alcun input 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 una zona di disponibilità.

Per abilitare la resilienza dei job, imposta la policy relativa ai nuovi tentativi 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 per i processi in batch e in streaming, quindi personalizza 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.

	Lavori in batch	Lavori in streaming
Comportamento predefinito	Non riesegue il lavoro.	Riprova sempre a eseguire il processo man mano che l'applicazione crea checkpoint durante l'esecuzione del lavoro.
Punto di riprova	I processi Batch non hanno punti di controllo, quindi EMR	I processi di streaming supportano i checkpoint, quindi configura la query

	Lavori in batch	Lavori in streaming
	Serverless riesegue sempre il lavoro dall'inizio.	di streaming per salvare lo stato di runtime e l'avanzamento verso una posizione di checkpoint in Amazon S3. EMR Serverless riprende l'esecuzione del job dal checkpoint. Per ulteriori informazioni, consulta Recovery from failures 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 eseguire nuovamente un processo, lo indicizza anche con un numero di tentativo, in modo da tenere traccia del ciclo di vita di un lavoro tra i suoi tentativi.

Utilizza le operazioni dell'API EMR Serverless o AWS CLI per modificare la resilienza del lavoro o accedere alle informazioni relative alla resilienza del lavoro. Per ulteriori informazioni, consulta la guida [EMR Serverless 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"
    }
  }'
```

EMR Serverless 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 degli errori per lo streaming di nuovi tentativi di lavoro. `maxFailedAttemptsPerHour` Questo parametro consente di specificare il numero massimo di tentativi falliti consentiti con un'ora prima che EMR Serverless interrompa i nuovi tentativi. L'impostazione predefinita è cinque. I valori accettati sono compresi tra 1 e 10.

```
aws emr-serverless start-job-run
  --application-id <APPLICATION_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"
    }
  }'

```

Puoi anche utilizzare le altre operazioni API Job Run per ottenere informazioni sui lavori. Ad esempio, utilizzate 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 che viene utilizzato per accedere all'applicazione UIs per l'esecuzione di un processo. Il `attempt` parametro consente di ottenere un 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 EMR Serverless job run retry. EMR Serverless pubblica questo evento a ogni nuovo tentativo di lavoro. 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 job, il tentativo genera il proprio set di log. Per garantire che EMR Serverless possa inviare correttamente questi log ad Amazon S3 e Amazon senza CloudWatch sovrascriverli, EMR Serverless aggiunge un prefisso al formato del percorso di log S3 e CloudWatch al nome del flusso di log per includere il numero del tentativo del processo.

Di seguito è riportato un esempio dell'aspetto del 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 la sezione Archiviazione dei log](#).

Note

EMR Serverless utilizza questo formato di prefisso solo con tutti i processi di streaming e tutti i processi batch con riprova abilitata.

Configurazione Metastore per EMR Serverless

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, conserva i metadati di questa tabella in un metastore che ha accesso ai tuoi lavori.

Hai due opzioni per un metastore Hive:

- Il catalogo dati AWS Glue
- Un metastore esterno di Apache Hive

Utilizzo del AWS Glue Data Catalog come metastore

Puoi configurare i tuoi job Spark e Hive per utilizzare il AWS Glue Data Catalog come metastore. Consigliamo questa configurazione quando hai bisogno di un metastore persistente o un metastore condiviso da diverse applicazioni, servizi o Account AWS. Per ulteriori informazioni sul Data Catalog, consulta [Populating the AWS Glue Data Catalog](#). Per informazioni sui prezzi di AWS Glue, consulta i [prezzi di AWS Glue](#).

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

Configurazione del AWS Glue Data Catalog

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, il AWS Glue Data Catalog è il metastore predefinito.

Quando si utilizzano gli SDK o AWS CLI si imposta la `spark.hadoop.hive.metastore.client.factory.class` configurazione su `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` nei parametri di esecuzione del processo. `sparkSubmit` 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://amzn-s3-demo-bucket/code/pyspark/
extreme_weather.py",
      "sparkSubmitParameters": "--conf
spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory
--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 tuo 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. In altre parole, quando si eseguono lavori su un'applicazione EMR Serverless Hive, Hive registra le informazioni sui metastore nel Data Catalog nello stesso modo in cui viene eseguita l'applicazione. Account AWS Non è necessario un cloud privato virtuale (VPC) per utilizzare il Data Catalog come metastore.

Per accedere alle tabelle dei metastore di Hive, aggiungi le politiche AWS Glue richieste descritte in [Configurazione delle autorizzazioni IAM per Glue](#). AWS

Configurazione dell'accesso tra account per EMR Serverless AWS e Glue Data Catalog

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

- AccountA— E 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 i tuoi job EMR Serverless possano accedere.
1. Assicurati che un amministratore o un'altra identità autorizzata AccountB alleggi 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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue>DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "arn:aws:glue:*:123456789012:catalog"
      ],
      "Sid": "AllowGLUEGetdatabase"
    }
  ]
}
```

2. Aggiungi una policy IAM al ruolo di job runtime EMR Serverless in AccountA modo che quel ruolo possa accedere alle risorse del Data Catalog in AccountB

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue>DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "arn:aws:glue:*:123456789012:catalog"
      ],
      "Sid": "AllowGLUEGetdatabase"
    }
  ]
}
```

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

Spark

Passate la `spark.hadoop.hive.metastore.glue.catalogid` proprietà `sparkSubmitParameters` come illustrato nell'esempio seguente. Sostituisci *AccountB-catalog-id* con l'ID del Data Catalog in AccountB.

```
aws emr-serverless start-job-run \
--application-id "application-id" \
```

```

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

```

Hive

Imposta la `hive.metastore.glue.catalogid` proprietà nella `hive-site` classificazione come illustrato nell'esempio seguente. Sostituisci *AccountB-catalog-id* con l'ID del Data Catalog in `AccountB`.

```

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

```

```
}]
}'
```

Considerazioni sull'utilizzo del AWS Glue Data Catalog

Puoi aggiungere JAR ausiliari ADD JAR negli script Hive. Per ulteriori considerazioni, consulta [Considerazioni sull'utilizzo di AWS Glue Data Catalog](#).

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 RDS for MySQL. Questa sezione descrive come configurare un metastore Amazon RDS Hive, configurare il VPC e configurare i job EMR Serverless per utilizzare un metastore esterno.

Crea un metastore Hive esterno

1. [Crea un Amazon Virtual Private Cloud \(Amazon VPC\) con sottoreti private seguendo le istruzioni in Creare un VPC](#).
2. Crea la tua applicazione EMR Serverless con il tuo nuovo Amazon VPC e le sottoreti private. Quando si configura un'applicazione EMR Serverless con un VPC, viene prima fornita 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 maggiori dettagli su come configurare il tuo VPC, consulta [Configurazione dell'accesso VPC per le applicazioni EMR Serverless per la connessione ai dati](#)
3. Crea un database MySQL o Aurora PostgreSQL in una sottorete privata nel tuo Amazon VPC. Per informazioni su come creare un database Amazon RDS, consulta [Creazione di un'istanza database Amazon RDS](#).
4. [Modifica il gruppo di sicurezza del tuo database MySQL o Aurora per consentire le connessioni JDBC dal tuo gruppo di sicurezza EMR Serverless seguendo i passaggi descritti in Modificare un'istanza database Amazon RDS](#). Aggiungi una regola per il traffico in entrata al gruppo di sicurezza RDS da uno dei tuoi gruppi di sicurezza EMR Serverless.

Tipo	Protocollo	Intervallo porte	Origine
Tutte le regole TCP	TCP	3306	emr-serverless-security-group

Configura le opzioni Spark

Usare JDBC

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

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

```

        "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
    }
}
}'

```

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 RDS for MySQL o Amazon Aurora MySQL. A tale scopo, esegui un server Thrift sul nodo primario di un cluster Amazon EMR esistente. Questa opzione è ideale se disponi già di un cluster Amazon EMR con un server Thrift che desideri utilizzare per semplificare le configurazioni di lavoro EMR Serverless.

```

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

```

```

        --conf spark.driver.cores=2
        --conf spark.executor.memory=10G
        --conf spark.driver.memory=6G
        --conf spark.executor.cores=4"
    }
}' \
--configuration-overrides '{
    "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
        "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
    }
}
}'

```

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:thrift-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

Usare JDBC

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

Note

In Hive, puoi eseguire più scritture su tabelle metastore contemporaneamente. Se condivide 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 esterno `hive-site`.

```
{
  "classification": "hive-site",
  "properties": {
    "hive.metastore.client.factory.class":
"org.apache.hadoop.hive.q1.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 RDS for MySQL o Amazon Aurora MySQL Instance. A tale scopo, esegui un server Thrift sul nodo principale di un cluster Amazon EMR esistente. Questa opzione è ideale se disponi già di un cluster Amazon EMR 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:thrift-server-port"
}
}

```

Utilizzo della gerarchia AWS multicatalogo di Glue su EMR Serverless

È possibile configurare le applicazioni EMR Serverless in modo che funzionino con la gerarchia multicatalogo AWS Glue. L'esempio seguente mostra come usare EMR-S Spark con la gerarchia multicatalogo di AWS Glue.

Per ulteriori informazioni sulla gerarchia multicatalogo, consulta [Lavorare con una gerarchia multicatalogo in AWS Glue Data Catalog with Spark su Amazon EMR](#).

Utilizzo di Redshift Managed Storage (RMS) con Iceberg e Glue Data Catalog AWS

Di seguito viene illustrato come configurare Spark per l'integrazione con un AWS Glue Data Catalog con Iceberg:

```

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

```

Una query di esempio da una tabella del catalogo, dopo l'integrazione:

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

Utilizzo di Redshift Managed Storage (RMS) con l'API REST di Iceberg e Glue Data Catalog AWS

Di seguito viene illustrato come configurare Spark per funzionare con il catalogo REST di Iceberg:

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

Un esempio di query da una tabella del catalogo:

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

Considerazioni sull'utilizzo di un metastore esterno

- Puoi configurare database compatibili con Mariadb JDBC come metastore. Esempi di questi database sono RDS per Mariadb, MySQL 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.](#)
- EMR Serverless non supporta l'autenticazione Kerberos. Non è possibile utilizzare un server metastore Thrift con autenticazione Kerberos con i job EMR Serverless Spark o Hive.
- È necessario configurare l'accesso VPC per utilizzare la gerarchia multicatalogo.

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

Puoi eseguire job Serverless di Amazon EMR da un AWS account e configurarli per accedere ai dati nei bucket Amazon S3 che appartengono a un altro account. AWS 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 presunto per accedere ai dati in Amazon S3 da un altro account. AWS

Prerequisiti

Per configurare l'accesso a più account per Amazon EMR Serverless, completa le attività accedendo a due account: AWS

- **AccountA**— Questo è l'AWS account su cui hai creato un'applicazione Amazon EMR Serverless. Prima di configurare l'accesso tra più account, tieni a portata di mano quanto segue in questo account:
 - Un'applicazione Serverless Amazon EMR in cui eseguire lavori.
 - Un ruolo di esecuzione del lavoro che dispone delle autorizzazioni necessarie per eseguire i lavori nell'applicazione. Per ulteriori informazioni, vedi [Ruoli Job Runtime per Amazon EMR Serverless](#).
- **AccountB**— Questo è l'AWS account che contiene il bucket S3 a cui desideri accedere per i tuoi job Serverless di Amazon EMR.

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

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePermissions1",
      "Effect": "Allow",
      "Principal": {
```

```

    "AWS": "arn:aws:iam::123456789012:root"
  },
  "Action": [
    "s3:ListBucket"
  ],
  "Resource": [
    "arn:aws:s3:::my-bucket-name"
  ]
},
{
  "Sid": "ExamplePermissions2",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:root"
  },
  "Action": [
    "s3:PutObject",
    "s3:GetObject",
    "s3:DeleteObject"
  ],
  "Resource": [
    "arn:aws:s3:::my-bucket-name/*"
  ]
}
]
}

```

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 account per Amazon EMR Serverless consiste nell'eseguire `AssumeRole` l'azione di AWS Security Token Service (`sts:AssumeRole`). AWS STS è un servizio web globale che consente di richiedere credenziali temporanee con privilegi limitati per gli utenti. Puoi effettuare chiamate API 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 EMR Serverless:

1. Crea un bucket Amazon S3, *cross-account-bucket*, in AccountB. Per ulteriori informazioni, consulta la sezione [Creazione di un bucket](#) nella Guida per l'utente di Amazon Simple Storage Service. Se desideri avere accesso a DynamoDB su più account, crea anche una tabella DynamoDB in AccountB. Per ulteriori informazioni, consulta [Creating a DynamoDB table nella Amazon DynamoDB Developer Guide](#).
2. Crea un ruolo IAM Cross-Account-Role-B in AccountB per accedere a *cross-account-bucket*.
 - a. Accedi Console di gestione AWS e apri la console IAM 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 ruoli IAM, consulta [Creating IAM roles](#) nella IAM User Guide.
 - c. Crea una policy IAM che specifichi le autorizzazioni per Cross-Account-Role-B per accedere al bucket S3 *cross-account-bucket*, come dimostra la seguente istruzione di policy. Quindi, allega la policy IAM a Cross-Account-Role-B. Per ulteriori informazioni, consulta [Creating IAM policies](#) nella IAM User Guide.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::cross-account-bucket",
        "arn:aws:s3:::cross-account-bucket/*"
      ],
      "Sid": "AllowS3"
    }
  ]
}
```

Se hai bisogno dell'accesso a DynamoDB, crea una policy IAM che specifichi le autorizzazioni per accedere alla tabella DynamoDB tra account. Quindi, allega la policy IAM a Cross-Account-Role-B. Per ulteriori informazioni, consulta [Amazon DynamoDB: consente l'accesso a una tabella specifica](#) nella IAM User Guide.

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

JSON

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

3. Modifica la relazione di fiducia per il ruolo `Cross-Account-Role-B`.

- a. Per configurare la relazione di trust per il ruolo, scegli la scheda `Trust Relationships` nella console IAM per il 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.

JSON

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

```
]
}
```

4. Concedi Job-Execution-Role-A AccountA il AWS STS AssumeRole permesso di assumereCross-Account-Role-B.

- Nella console IAM per l' AWS accountAccountA, selezionaJob-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.

JSON

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

Esempi di ruoli presunti

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

Argomenti

- [Accedi alle risorse S3 con un ruolo presunto](#)
- [Accedi alle risorse S3 con più ruoli presunti](#)

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 configurazione EMRFS per. `fs.s3.customAWSCredentialsProvider` `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider`
2. Per Spark, usa `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` e specifica le variabili `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` di ambiente su driver ed executor.
3. Per Hive `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` `tez.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 Hive driver, Tez application primary e Tez task.

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

Spark

L'esempio seguente mostra come utilizzare un ruolo assunto per avviare un job EMR Serverless Spark eseguito con accesso multiaccount a S3.

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

```

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

```

Hive

L'esempio seguente mostra come utilizzare un ruolo presunto per avviare un job EMR Serverless Hive eseguito con accesso multiaccount a S3.

```

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

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

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

Spark

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

```

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 si verificano quando si lavora con Amazon EMR Serverless.

Argomenti

- [Errore: Job non riuscito poiché l'account ha raggiunto il limite di servizio sulla vCPU massima che può utilizzare contemporaneamente.](#)
- [Errore: Job non riuscito perché l'applicazione ha superato le impostazioni di MaximumCapacity.](#)
- [Errore: Job non riuscito a causa dell'impossibilità di allocare Worker poiché l'applicazione ha superato la capacità massima.](#)
- [Errore: l'accesso a S3 è negato. Controlla le autorizzazioni di accesso S3 del ruolo Job Runtime sulle risorse S3 richieste.](#)
- [Errore ModuleNotFoundError: nessun modulo denominato <module>. Consulta la guida per l'utente su come utilizzare le librerie python con EMR Serverless.](#)
- [Errore: impossibile assumere il ruolo di esecuzione <role name> perché non esiste o non è configurato con la relazione di trust richiesta.](#)

Errore: Job non riuscito poiché l'account ha raggiunto il limite di servizio sulla vCPU massima che può utilizzare contemporaneamente.

Questo errore indica che EMR Serverless non è riuscito a inviare il lavoro poiché l'account ha superato la capacità massima. Aumentare la capacità massima dell'account. Verifica i limiti del servizio nelle quote di servizio [EMR Serverless](#).

Errore: Job non riuscito perché l'applicazione ha superato le impostazioni di MaximumCapacity.

Questo errore indica che EMR Serverless non è riuscito a inviare il lavoro poiché l'applicazione ha superato la capacità massima configurata. Aumentare la capacità massima dell'applicazione.

Errore: Job non riuscito a causa dell'impossibilità di allocare Worker poiché l'applicazione ha superato la capacità massima.

Questo errore indica che il processo non è stato completato. Impossibile allocare i lavoratori perché l'applicazione ha superato le impostazioni di MaximumCapacity.

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, consulta. [Ruoli Job Runtime per Amazon EMR Serverless](#)

Errore ModuleNotFoundError: nessun modulo denominato <module>.
Consulta la guida per l'utente su come utilizzare 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, fare riferimento a. [Utilizzo delle librerie Python con EMR Serverless](#)

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 esecuzione del lavoro specificato per il lavoro non esiste o che il ruolo non ha una relazione di trust per le autorizzazioni EMR Serverless. Per verificare l'esistenza del ruolo IAM e verificare che la policy di fiducia del ruolo sia stata impostata correttamente, consulta le istruzioni riportate in. [Ruoli Job Runtime per Amazon EMR Serverless](#)

Abilitazione dell'allocazione dei costi a livello di Job

L'allocazione dei costi a livello di processo consente l'attribuzione granulare della fatturazione per EMR Serverless a livello di singola esecuzione del processo, anziché aggregare tutti i costi a livello di applicazione. Se abilitata, è possibile filtrare e tenere traccia dei costi AWS in Cost Explorer e Cost and Usage Reports per specifiche esecuzioni di job IDs e tag associati alle esecuzioni di job, garantendo una migliore visibilità degli addebiti per le esecuzioni di job inviate.

Comportamento predefinito

L'allocazione dei costi a livello di processo non è abilitata per impostazione predefinita.

Come abilitare o disabilitare la funzionalità

È possibile configurare l'allocazione dei costi a livello di processo durante la creazione dell'applicazione o aggiornarla per le applicazioni esistenti.

Specificate il `jobLevelCostAllocation` parametro quando create una nuova applicazione:

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

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

Aggiorna il `jobLevelCostAllocationConfiguration` parametro per un'applicazione esistente:

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

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

```
"enabled": false  
'
```

Considerazioni e limitazioni

- L'abilitazione dell'allocazione dei costi a livello di processo non attribuisce retroattivamente i costi per le esecuzioni di lavoro completate prima dell'attivazione della funzionalità. Le esecuzioni dei job avviate dopo l'attivazione della funzionalità avranno un'attribuzione dei costi granulare.
- Il parametro di allocazione dei costi a livello di processo può essere aggiornato solo quando un'applicazione è nello stato CREATED o STOPPED.
- Quando è abilitata l'allocazione dei costi a livello di processo, i costi vengono attribuiti alle singole esecuzioni dei processi anziché all'applicazione. Per visualizzare i costi aggregati a livello di applicazione, è necessario applicare tag coerenti (come application-name o application-id) a tutti i job eseguiti all'interno dell'applicazione e filtrare in base a tali tag in Cost Explorer o Cost and Usage Reports.

Esegui sessioni interattive con Amazon EMR Serverless tramite Spark Connect

Con la versione di Amazon EMR `emr-7.13.0` e successive, puoi connetterti a un'applicazione Amazon EMR Serverless da PySpark client autogestiti come VS Code e notebook Jupyter utilizzando la sessione EMR Serverless con Apache Spark Connect. PyCharm APIs Spark Connect utilizza un'architettura client-server che disaccoppia il codice dell'applicazione dal processo del driver Spark. Sviluppi ed esegui il debug PySpark del codice nel tuo IDE locale mentre le operazioni Spark vengono eseguite sull'elaborazione EMR Serverless. Spark Connect offre i seguenti vantaggi:

- Connect a EMR Serverless da qualsiasi PySpark client, inclusi notebook VS Code e PyCharm Jupyter.
- Imposta punti di interruzione e analizza il PySpark codice nel tuo IDE mentre esegui in remoto su dati su scala di produzione. DataFrames

Una sessione Spark Connect è una connessione gestita tra il PySpark client locale e un driver Spark in esecuzione su Amazon EMR Serverless. Quando avvii una sessione, EMR Serverless effettua il provisioning di un driver Spark e di esecutori per tuo conto. Il client locale invia DataFrame le operazioni SQL al driver, che le esegue in remoto. La sessione persiste finché non viene terminata o non raggiunge il timeout di inattività, quindi puoi eseguire più query in modo interattivo senza riavviare Spark. Ogni sessione ha il proprio URL dell'endpoint e il token di autenticazione che usi per connetterti.

Autorizzazioni richieste

Oltre alle autorizzazioni necessarie per accedere ad Amazon EMR Serverless, aggiungi anche le seguenti autorizzazioni al tuo ruolo IAM per accedere a un endpoint Spark Connect e gestire le sessioni Spark Connect:

```
emr-serverless:StartSession
```

Concede l'autorizzazione a creare una sessione Spark Connect sull'applicazione specificata come `Resource`

`emr-serverless:GetSessionEndpoint`

Concede l'autorizzazione a recuperare l'URL dell'endpoint Spark Connect e il token di autenticazione per una sessione.

`emr-serverless:GetSession`

Concede l'autorizzazione per ottenere lo stato di una sessione.

`emr-serverless:ListSessions`

Concede l'autorizzazione a elencare le sessioni su un'applicazione.

`emr-serverless:TerminateSession`

Concede l'autorizzazione a terminare una sessione.

`iam:PassRole`

Concede l'autorizzazione ad accedere al ruolo di esecuzione IAM durante la creazione della sessione Spark Connect. Amazon EMR Serverless utilizza questo ruolo per eseguire i carichi di lavoro.

`emr-serverless:GetResourceDashboard`

Concede l'autorizzazione a generare l'URL dell'interfaccia utente Spark e fornisce l'accesso ai log della sessione.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessApplicationLevelAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:StartSession",
        "emr-serverless:ListSessions"
      ],
      "Resource": [
        "arn:aws:emr-serverless:region:account-id:/applications/application-id"
      ]
    },
    {
      "Sid": "EMRServerlessSessionLevelAccess",
      "Effect": "Allow",
```

```

    "Action": [
      "emr-serverless:GetSession",
      "emr-serverless:GetSessionEndpoint",
      "emr-serverless:TerminateSession",
      "emr-serverless:GetResourceDashboard"
    ],
    "Resource": [
      "arn:aws:emr-serverless:region:account-id:/applications/application-id/
sessions/*"
    ]
  },
  {
    "Sid": "EMRServerlessRuntimeRoleAccess",
    "Effect": "Allow",
    "Action": [
      "iam:PassRole"
    ],
    "Resource": [
      "arn:aws:iam::account-id:role/EMRServerlessExecutionRole"
    ],
    "Condition": {
      "StringLike": {
        "iam:PassedToService": "emr-serverless.amazonaws.com"
      }
    }
  }
]
}

```

Lavorare con sessioni interattive

Per creare un'applicazione compatibile con Spark Connect e connetterti ad essa, segui questi passaggi.

Per avviare una sessione Spark Connect

1. Crea un'applicazione con sessioni Spark Connect.

```

aws emr-serverless create-application \
  --type "SPARK" \
  --name "spark-connect-app" \
  --release-label emr-7.13.0 \

```

```
--interactive-configuration '{"sessionEnabled": true}'
```

2. Dopo che Amazon EMR Serverless ha creato l'applicazione, avvia la se non hai abilitato l'avvio automatico per accettare le sessioni Spark Connect.

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

3. Usa il seguente comando per verificare lo stato della tua applicazione. Dopo che lo stato diventa `STARTED`, avvia una sessione.

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

4. Inizia una sessione con un ruolo di esecuzione IAM che concede l'accesso ai tuoi dati.

```
aws emr-serverless start-session \  
--application-id APPLICATION_ID \  
--execution-role-arn arn:aws:iam::account-id:role/EMRServerlessExecutionRole
```

5. Monitora lo stato della sessione utilizzando l'`get-session` API e attendi che la sessione sia attiva `STARTED` o in `IDLE` uno stato.

```
aws emr-serverless get-session \  
--application-id APPLICATION_ID \  
--session-id SESSION_ID
```

6. Recupera l'endpoint Spark Connect e il token di autenticazione. L'URL dell'endpoint restituito da `getSessionEndpoint` non include un numero di porta. Quando si crea l'URL di `sc://` connessione, è necessario aggiungere, ad `:443` esempio. `sc://hostname:443/;use_ssl=true;x-aws-proxy-auth=token` Senza di essa, il PySpark client utilizza per impostazione predefinita la porta 15002, che non è raggiungibile su EMR Serverless.

```
aws emr-serverless get-session-endpoint \  
--application-id APPLICATION_ID \  
--session-id SESSION_ID
```

La risposta include l'URL dell'endpoint e un token di autenticazione:

```
{  
  "endpoint": "ENDPOINT_URL",
```

```

"authToken": "AUTH_TOKEN",
"authTokenExpiresAt": "AUTH_TOKEN_EXPIRY_TIME"
}

```

- Quando l'endpoint è pronto, connettiti da un PySpark client. Installa il PySpark client che corrisponde alla versione Spark sulla tua applicazione EMR Serverless e l' AWS SDK per Python.

```

# Match the PySpark version to your EMR Serverless release version (3.5.6 for
  emr-7.13.0)
pip install pyspark[connect]==3.5.6
pip install boto3

```

Di seguito è riportato un esempio di script Python per avviare una sessione e inviare richieste direttamente all'endpoint della sessione:

```

import boto3
import time
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

client = boto3.client('emr-serverless', region_name='REGION')

APPLICATION_ID = 'APPLICATION_ID'
EXECUTION_ROLE = 'arn:aws:iam::account-id:role/EMRServerlessExecutionRole'

# Start the session
response = client.start_session(
    applicationId=APPLICATION_ID,
    executionRoleArn=EXECUTION_ROLE
)
session_id = response['sessionId']
print(f"Session {session_id} starting...")

# Wait for the session to be ready
while True:
    response = client.get_session(
        applicationId=APPLICATION_ID,
        sessionId=session_id
    )
    state = response['session']['state']
    print(f"Session state: {state}")

```

```
    if state in ('STARTED', 'IDLE'):
        break
    if state in ('FAILED', 'TERMINATED'):
        raise Exception(f"Session failed: {response['session'].get('stateDetails',
'Unknown error')}")
    time.sleep(5)

# Retrieve the Spark Connect endpoint and authentication token
response = client.get_session_endpoint(
    applicationId=APPLICATION_ID,
    sessionId=session_id
)

# Construct the authenticated remote URL
auth_token = response['authToken']
endpoint_url = response['endpoint']
connect_url = endpoint_url.replace("https://", "sc://", 1) + ":443/;use_ssl=true;"
connect_url += f"x-aws-proxy-auth={auth_token}"

# Start the Spark session
spark = SparkSession.builder.remote(connect_url).getOrCreate()
print(f"Connected. Spark version: {spark.version}")

# Run SQL
spark.sql("SELECT 1+1 AS result").show()

# Run DataFrame operations
df = spark.range(100).withColumn("squared", col("id") * col("id"))
df.show(10)
print(f"Count: {df.count()}")

# Stop the Spark session (disconnects the client only)
spark.stop()

# Terminate the EMR Serverless session to stop billing.
# spark.stop() only closes the local client connection. The remote session
# continues running and incurring charges until you explicitly terminate it
# or it reaches the idle timeout.
client.terminate_session(
    applicationId=APPLICATION_ID,
    sessionId=session_id
)
print(f"Session {session_id} terminated.")
```

Per accedere all'interfaccia utente live di Spark o allo Spark History Server per una sessione, usa l'API. `GetResourceDashboard`

```
response = client.get_resource_dashboard(  
    applicationId=APPLICATION_ID,  
    resourceId=session_id,  
    resourceType='SESSION'  
)  
response['url']
```

Mentre una sessione è attiva, l'URL apre l'interfaccia utente live di Apache Spark per il monitoraggio in tempo reale di query, fasi ed esecutori. Al termine di una sessione, Spark History Server rimane disponibile per l'analisi post-sessione tramite la console Amazon EMR Serverless.

Considerazioni e limitazioni

Considera quanto segue quando esegui carichi di lavoro interattivi tramite Spark Connect.

- Spark Connect è supportato con la versione Serverless `emr-7.13.0` di Amazon EMR e successive.
- Spark Connect è supportato solo per il motore Apache Spark.
- Spark Connect supporta DataFrame e SQL APIs in PySpark. I sistemi basati su RDD APIs non sono supportati.
- I token di autenticazione sono limitati nel tempo a 1 ora. Quando un token scade, le chiamate gRPC falliscono con un errore di autenticazione. Chiama `GetSessionEndpoint` per ottenere un nuovo token e creane uno nuovo `SparkSession` con il token aggiornato.
- Le sessioni terminano dopo un timeout di inattività configurabile. Il timeout predefinito è impostato su 1 ora.
- Per impostazione predefinita, ogni sessione ha un limite rigido di 24 ore, dopodiché si interrompe automaticamente anche se sta eseguendo attivamente un'attività.
- Per impostazione predefinita, ogni applicazione EMR Serverless supporta fino a 25 sessioni simultanee. Per richiedere un aumento del limite, contatta l'AWS assistenza.
- Per impostazione predefinita, `autoStopConfig` è attiva per le applicazioni. L'applicazione si interrompe automaticamente dopo 15 minuti senza sessioni attive o esecuzioni di lavoro. Puoi modificare questa configurazione come parte della tua `create-application update-application` richiesta.

- Per una migliore esperienza di avvio, configura la capacità preinizializzata per driver ed esecutori.
- È necessario abilitare `AutoStart` o avviare manualmente l'applicazione prima di iniziare una sessione EMR Serverless.
- La PySpark versione installata localmente deve corrispondere alla versione di Apache Spark sull'applicazione Amazon EMR Serverless (3.5.6 per). `emr-7.13.0` Cause di mancata corrispondenza della versione o comportamento imprevisto. `ImportError`
- Il controllo granulare degli accessi tramite Lake Formation non è supportato per le sessioni Spark Connect.
- La Trusted Identity Propagation non è supportata per le sessioni interattive con Spark Connect.
- Lo storage serverless su EMR Serverless non è supportato per le sessioni interattive con Spark Connect.
- Non sono previsti costi aggiuntivi per l'utilizzo di Spark Connect. Pagi solo per le risorse di elaborazione EMR Serverless (vCPU, memoria e storage) consumate durante la sessione.
- La configurazione Spark `spark.connect.grpc.binding.address` è riservata da EMR Serverless e non può essere sostituita dagli utenti.
- Il PySpark pacchetto che installi localmente deve corrispondere alla versione Spark sulla tua applicazione EMR Serverless. Una mancata corrispondenza della versione causa errori di connessione. Python UDFs (`@udf,spark.udf.register`) richiede anche che la versione minore locale di Python corrisponda al worker, altrimenti fallisce. `PYTHON_VERSION_MISMATCH` Le funzioni e le DataFrame operazioni SQL integrate non richiedono una corrispondenza della versione di Python.
- Per passare le configurazioni Spark `constart-session`, impostale sotto `runtimeConfiguration` nel parametro. `--configuration-overrides L'start-job-runAPI` utilizza `applicationConfiguration` invece.

Esegui carichi di lavoro interattivi con EMR Serverless tramite EMR Studio

Con le applicazioni interattive EMR Serverless, esegui carichi di lavoro interattivi per Spark con EMR Serverless utilizzando notebook ospitati in EMR Studio.

Panoramica di

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 EMR Studio. 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 le applicazioni interattive in EMR Serverless includono quanto segue:

- I data engineer utilizzano l'esperienza IDE di EMR Studio per creare uno script ETL. 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, soddisfa i seguenti requisiti:

- Le applicazioni interattive EMR Serverless sono supportate con Amazon EMR 6.14.0 e versioni successive.
- Per accedere all'applicazione interattiva, eseguire i carichi di lavoro inviati ed eseguire notebook interattivi da EMR Studio, sono necessari permessi e ruoli specifici. Per ulteriori informazioni, vedi [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](#), configura autorizzazioni aggiuntive per la tua identità o 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 EMR Serverless

Imposta un ruolo di job runtime. Per ulteriori informazioni, vedi [Creare un ruolo di job runtime](#).

Per eseguire i taccuini interattivi da EMR Studio

Aggiungi le seguenti autorizzazioni aggiuntive alla policy IAM 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 un EMR Studio Workspace.
- **iam:PassRole**- Concede l'autorizzazione ad accedere al ruolo di esecuzione IAM che intendi utilizzare quando ti colleghi a un'applicazione. È richiesta l'`PassRole` autorizzazione appropriata per collegarsi a un'applicazione EMR Serverless da un EMR Studio Workspace.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessInteractiveAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessInteractiveEndpoints"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    },
    {
```

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

Configurazione di applicazioni interattive

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

1. Segui i passaggi indicati per creare un'applicazione [inizia a usare Amazon EMR Serverless](#).
2. Quindi, avvia un'area di lavoro da EMR Studio e collegala a un'applicazione EMR Serverless come opzione di elaborazione. Per ulteriori informazioni, fare riferimento alla scheda Carico di lavoro interattivo nella fase 2 della documentazione [EMR Serverless 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

- Le applicazioni interattive EMR Serverless sono supportate con Amazon EMR 6.14.0 e versioni successive.
- EMR Studio è l'unico client integrato con le applicazioni interattive EMR Serverless. Le seguenti funzionalità di EMR Studio non sono supportate con le applicazioni interattive EMR Serverless: collaborazione 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'interfaccia endpoint o API che supporti notebook Jupyter ospitati autonomamente con applicazioni interattive.
- Per un'esperienza di avvio ottimizzata, suggeriamo 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 eseguire 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 una 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 vCPU e 16 GB di memoria. Per informazioni aggiornate sui prezzi, consulta la pagina dei [prezzi di Amazon EMR](#).
- È necessario disporre di una quota di servizi vCPU sufficiente per Account AWS eseguire carichi di lavoro interattivi. Se non esegui Formation-enabled carichi di lavoro Lake, ti suggeriamo almeno 24 vCPU. In tal caso, suggeriamo almeno 28 vCPU.
- EMR Serverless interrompe automaticamente i kernel dai notebook se sono rimasti inattivi per più di 60 minuti. EMR Serverless calcola il tempo di inattività del kernel 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, consulta [Enabling Lake Formation in Amazon EMR](#).

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

Con le versioni 6.14.0 e successive di Amazon EMR, crea e abilita un endpoint Apache Livy durante la creazione di un'applicazione EMR Serverless ed esegui 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 REST di Apache Livy.
- Utilizza le operazioni dell'API REST di Apache Livy per applicazioni Web interattive che utilizzano i dati dei carichi di lavoro Apache Spark.

Prerequisiti

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

- Completa i passaggi descritti 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 necessarie per accedere a EMR Serverless, aggiungi anche le seguenti autorizzazioni al tuo ruolo IAM per accedere a un endpoint Apache Livy ed eseguire applicazioni:

- `emr-serverless:AccessLivyEndpoints`— concede l'autorizzazione per accedere e connettersi all'applicazione specificata come `Livy-enabled Resource`. È necessaria questa autorizzazione per eseguire le operazioni dell'API REST disponibili dall'endpoint Apache Livy.
- `iam:PassRole`— concede l'autorizzazione ad accedere al ruolo di esecuzione IAM durante la creazione della sessione Apache Livy. EMR Serverless utilizzerà questo ruolo per eseguire i 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.

JSON

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

Nozioni di base

Per creare un' Livy-enabled applicazione Apache ed eseguirla, segui questi passaggi.

1. Per creare un' Livy-enabled applicazione Apache, 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, avviala 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 stato `STARTED`, accedi all'endpoint Apache Livy.

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

3. Utilizza il seguente URL per accedere all'endpoint:

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

Una volta che l'endpoint è pronto, invia i carichi di lavoro in base al tuo caso d'uso. È necessario firmare ogni richiesta all'endpoint con [il protocollo SigV4](#) e inserire un'intestazione di autorizzazione. È possibile utilizzare i seguenti metodi per eseguire carichi di lavoro:

- Client HTTP: invia le operazioni dell'API degli endpoint Apache Livy con un client HTTP personalizzato.
- Kernel Sparkmagic: esegui il kernel sparkmagic localmente e invia query interattive con i notebook Jupyter.

Client HTTP

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

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

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

Operazione API	Description
GET /sessions	Restituisce un elenco di tutte le sessioni interattive attive.
POST /sessions	Crea una nuova sessione interattiva tramite spark o pyspark.
GET /sessions/ < > <i>sessionId</i>	Restituisce le informazioni sulla sessione.
GET /sessions/ < >/state <i>sessionId</i>	Restituisce lo stato della sessione.
ELIMINA /sessions/ < > <i>sessionId</i>	Interrompe ed elimina la sessione.
GET /sessions/ < >/statements <i>sessionId</i>	Restituisce tutte le istruzioni di una sessione.
POST /sessions/ < >/dichiarazioni <i>sessionId</i>	Esegue un'istruzione in una sessione.
GET /sessions/ < >/statements/< > <i>sessionId</i> <i>statementId</i>	Restituisce i dettagli dell'istruzione specificata in una sessione.
POST /sessions/ < >/statements/< >/cancel <i>sessionId</i> <i>statementId</i>	Annulla l'istruzione specificata in questa sessione.

Invio di richieste all'endpoint Apache Livy

Puoi anche 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 awscli requests
```

Di seguito è riportato un esempio di script Python per inviare richieste HTTP 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', 'heartbeatTimeoutInSeconds': 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 le credenziali nell'istanza in cui desideri installare sparkmagic

1. [Installa sparkmagic seguendo i passaggi di installazione.](#) Nota che esegui solo 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 sia firmata SigV4.
3. Installa l'autenticatore personalizzato EMR Serverless.

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

4. Ora fornisci il percorso dell'autenticatore personalizzato e l'URL dell'endpoint 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 tutte le operazioni dell'API Apache Livy tramite l'URL dell'endpoint Apache Livy configurato. 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.

- EMR Serverless 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, configura un'identità di origine quando assumi le credenziali. In questo caso, EMR Serverless applica l'isolamento a livello di sessione in base al principale chiamante e

all'identità di origine. Per ulteriori informazioni sull'identità di origine, fare riferimento a [Monitoraggio e controllo](#) delle azioni intraprese con i ruoli assunti.

- Gli endpoint Apache Livy sono supportati con le 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, suggeriamo 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 servizi vCPU sufficiente per Account AWS eseguire carichi di lavoro interattivi con l'endpoint Apache Livy. Suggeriamo almeno 24 vCPU.
- Il timeout di sessione predefinito di Apache Livy è di 1 ora. Se non esegui le istruzioni per un'ora, Apache Livy elimina la sessione e rilascia il driver e gli executor. Dalla versione `emr-7.8.0`, questo valore può essere impostato specificando il `ttl` parametro come parte della `/sessions` POST richiesta Livy, ad esempio `2h (ore), (minuti), (secondi), (120millisecondi). 7200s 7200000ms`

Note

Questa configurazione non può essere modificata prima di `emr-7.8.0`. Di seguito è riportato un esempio di corpo della richiesta. `POST /sessions`

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

- A partire dalla versione emr-7.8.0 di Amazon EMR per applicazioni con controllo granulare degli accessi tramite LakeFormation enabled, l'impostazione può essere disabilitata per sessione. Per ulteriori informazioni sull'abilitazione del controllo granulare degli accessi per un'applicazione EMR Serverless, fare riferimento [a Metodi per](#) il controllo granulare degli accessi.

Note

Lake Formation non può essere abilitato per una sessione se non è stato abilitato per un'applicazione. Di seguito è riportato un esempio di corpo della POST `/sessions` richiesta.

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

- 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 una parte importante del mantenimento dell'affidabilità, della disponibilità e delle prestazioni delle applicazioni e dei lavori EMR Serverless. È necessario raccogliere i dati di monitoraggio da tutte le parti delle soluzioni EMR Serverless in modo che gli errori multipoint possano essere risolti più facilmente se si verificano.

Argomenti

- [Archiviazione dei registri](#)
- [Registri rotanti](#)
- [Crittografia dei log](#)
- [Configurazione delle proprietà di Apache Log4j2 per Amazon EMR Serverless](#)
- [Monitoraggio EMR Serverless](#)
- [Automazione di EMR Serverless con Amazon EventBridge](#)

Archiviazione dei registri

Per monitorare l'avanzamento del lavoro su EMR Serverless e risolvere i problemi relativi ai lavori, scegli in che modo EMR Serverless archivia e gestisce i registri delle applicazioni. Quando invii un job run, specifica lo storage gestito, Amazon S3 e Amazon CloudWatch come opzioni di registrazione.

Con CloudWatch, specifica i tipi e le posizioni di registro che desideri utilizzare o accetta i tipi e le posizioni predefiniti. Per ulteriori informazioni sui CloudWatch log, fare riferimento [alla sezione chiamata "Amazon CloudWatch"](#). Per quanto riguarda lo storage gestito e la registrazione S3, la tabella seguente elenca 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 uno storage gestito	Supportata

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	Supportata
Bucket Amazon S3	Memorizzato nel bucket S3	Memorizzato nel bucket S3	Non supportato ¹

¹ Ti consigliamo di mantenere selezionata l'opzione Archiviazione gestita. In caso contrario, non è possibile utilizzare l'applicazione integrata UIs.

Registrazione per EMR Serverless con storage gestito

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

Note

Se disattivi l'opzione predefinita, Amazon EMR non può risolvere i tuoi lavori per tuo conto. Esempio: non è possibile accedere a Spark-UI dalla console serverless EMR.

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

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

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

Se la tua applicazione EMR Serverless si trova in una sottorete privata con endpoint VPC per Amazon S3 e alleggi una policy di endpoint per controllare l'accesso, aggiungi le seguenti autorizzazioni per EMR Serverless per archiviare e servire i log delle applicazioni. Sostituisci Resource con i AppInfo bucket della tabella delle regioni disponibili in [Politiche di esempio per sottoreti private che accedono ad Amazon S3](#).

JSON

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

Inoltre, utilizza la chiave `aws:SourceVpc` condition per assicurarti che la richiesta passi attraverso il VPC a cui è collegato l'endpoint VPC.

Registrazione per EMR Serverless con bucket Amazon S3

Prima che i job possano inviare dati di log ad Amazon S3, includi le seguenti autorizzazioni nella politica di autorizzazione per il ruolo di runtime del job. Sostituisci *amzn-s3-demo-logging-bucket* con il nome del bucket di accesso.

JSON

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

Per configurare un bucket Amazon S3 per archiviare i log di AWS CLI, utilizza la `s3MonitoringConfiguration` configurazione all'avvio di un job. A tale scopo, fornisci quanto segue `--configuration-overrides` nella configurazione.

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

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

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

I log dei driver Spark sono archiviati nel seguente percorso da EMR Serverless

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

I log di Spark Executor vengono archiviati nel seguente percorso da EMR Serverless

```
'/applications/<applicationId>/jobs/<jobId>/SPARK_EXECUTOR/<EXECUTOR-ID>'
```

<EXECUTOR-ID>Il è un numero intero.

Le versioni 7.1.0 e successive di EMR Serverless supportano i tentativi di riprovare per processi di streaming e processi batch. Se si esegue un lavoro con i nuovi tentativi abilitati, 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>/'
```

Registrazione per EMR Serverless con Amazon CloudWatch

Quando invii un lavoro a un'applicazione EMR Serverless, scegli Amazon CloudWatch come opzione per archiviare i log delle 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.

EMR Serverless fornisce la registrazione in tempo reale dei registri dei driver. Puoi accedere ai log in tempo reale con la funzionalità CloudWatch live tail o tramite CloudWatch i comandi di coda della CLI.

Per impostazione predefinita, la CloudWatch registrazione è disabilitata per EMR Serverless. Per abilitarlo, usa 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 suggeriamo 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 EMR Serverless. Per ulteriori informazioni, consulta [Come posso determinare la limitazione nei miei registri?](#) CloudWatch su [re:post.AWS](#)

Autorizzazioni richieste per la registrazione con CloudWatch

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:*"
      ],
      "Sid": "AllowLOGSDescribeLogGroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSPutLogEvents"
    }
  ]
}
```

AWS CLI

Per configurare Amazon per CloudWatch archiviare i log per EMR Serverless da, utilizza AWS CLI la configurazione `cCloudWatchLoggingConfiguration` all'avvio di un job. A tale scopo, fornisci

le seguenti modifiche di configurazione. Facoltativamente, fornisci anche 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`

Le versioni 7.1.0 e successive di EMR Serverless supportano i tentativi di riprovare per processi di streaming e processi batch. Se sono stati abilitati i nuovi tentativi per un lavoro, 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 illustrata 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 specificano quando si attiva Amazon CloudWatch Logging per EMR Serverless. I `logTypes` valori supportati sono elencati anche nel seguente 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, specifica il ruolo del contenitore e i tipi di log corrispondenti con il campo. `logTypes`

L'elenco seguente mostra i tipi di worker supportati che specificano 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 di default ed è disponibile solo per i job Spark.

Registri degli eventi di Spark

Note

La rotazione dei log degli eventi Spark è disponibile su tutte le etichette di rilascio di Amazon EMR.

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 rilascio di Amazon EMR.

EMR Serverless ruota anche i log delle applicazioni Spark per driver ed esecutori, come i file `stdout` e `stderr`. È possibile accedere ai file di registro più recenti scegliendo i collegamenti 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 fare riferimento ai log ruotati precedenti, specifica 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. EMR Serverless 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.

Le versioni 7.1.0 e successive di EMR Serverless supportano i tentativi di riprovare per lo streaming e i job in batch. Se hai abilitato i tentativi di riprovare con il tuo processo, EMR Serverless aggiunge un prefisso al percorso di registro per tali lavori 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 chiave KMS, utilizza la `managedPersistenceMonitoringConfiguration` configurazione quando invii l'esecuzione di un lavoro.

```
{
  "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 tua chiave KMS, usa la configurazione `s3MonitoringConfiguration` quando invii un job run.

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

Crittografia dei log EMR Serverless con Amazon CloudWatch

Per crittografare i log in Amazon CloudWatch con la tua chiave KMS, 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. Puoi specificare le autorizzazioni nella politica delle chiavi KMS o nella politica IAM per l'utente, il gruppo o il ruolo. Se l'utente che invia il lavoro non dispone delle autorizzazioni della chiave KMS, EMR Serverless rifiuta l'invio dell'esecuzione del lavoro.

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": "*"
}

```

Policy IAM di esempio

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ],
      "Sid": "AllowKMSGeneratedatakey"
    }
  ]
}
```

Per avviare l'interfaccia utente Spark o Tez, concedi ai tuoi utenti, gruppi o ruoli le autorizzazioni per accedere all'`emr-serverless:GetDashboardForJobRunAPI` come segue:

JSON

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

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, configura le autorizzazioni delle chiavi KMS come segue.

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 best practice di sicurezza, ti suggeriamo di aggiungere una chiave di `aws:SourceArn` condizione alla politica delle chiavi KMS. La chiave di condizione globale IAM `aws:SourceArn` aiuta a garantire che EMR Serverless utilizzi la chiave KMS solo per l'ARN di un'applicazione.

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

JSON

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

```

    "kms:Decrypt"
  ],
  "Resource": [
    "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
  ],
  "Sid": "AllowKMSGeneratedatakey"
}
]
}

```

Autorizzazioni per le chiavi di crittografia per Amazon CloudWatch

Per associare l'ARN della chiave KMS al tuo gruppo di log, utilizza la seguente politica IAM per il ruolo di job runtime.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:AssociateKmsKey"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSAssociatekmskey"
    }
  ]
}

```

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

JSON

```

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

```

```
"Id": "key-default-1",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
    ],
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:*:123456789012:*"
      }
    },
    "Sid": "AllowKMSDecrypt"
  }
]
}
```

Configurazione delle proprietà di Apache Log4j2 per Amazon EMR Serverless

Questa pagina descrive come configurare le [proprietà personalizzate di Apache Log4j 2.x](#) per i lavori EMR Serverless su. StartJobRun Se si desidera configurare le classificazioni Log4j a livello di applicazione, fare riferimento a. [Configurazione predefinita dell'applicazione per EMR Serverless](#)

Configurazione delle proprietà di Spark Log4j2 per Amazon EMR Serverless

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

Argomenti

- [Classificazioni Log4j2 per Spark](#)
- [Esempio di configurazione Log4j2 per Spark](#)
- [Log4j2 in 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 EMR Serverless](#)

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

```

        "logger.IdentifierForClass.name": "classpath for setting logger",
        "logger.IdentifierForClass.level": "info"
    }
},
{
    "classification": "spark-executor-log4j2",
    "properties": {
        "rootLogger.level": "error", // will only display Spark error logs
        "logger.IdentifierForClass.name": "classpath for setting logger",
        "logger.IdentifierForClass.level": "info"
    }
}
]
}'

```

Log4j2 in esempi di job Spark

I seguenti esempi di codice mostrano come creare un'applicazione Spark mentre iniziizzi 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, usa 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- Utilizzo di 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, usa 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 configurate, consulta il file su `log4j2.properties.template` GitHub](#)

Monitoraggio EMR Serverless

Questa sezione illustra i modi per monitorare le applicazioni e i lavori Serverless di Amazon EMR.

Argomenti

- [Monitoraggio delle applicazioni e dei lavori EMR Serverless](#)
- [Monitora i parametri di Spark con Amazon Managed Service for Prometheus](#)
- [Metriche di utilizzo EMR Serverless](#)

Monitoraggio delle applicazioni e dei lavori EMR Serverless

Con Amazon CloudWatch Metrics for EMR Serverless, puoi ricevere parametri di CloudWatch 1 minuto e accedere a dashboard per CloudWatch accedere alle near-real-time operazioni e alle prestazioni delle tue applicazioni EMR Serverless.

EMR Serverless invia metriche a ogni minuto. CloudWatch EMR Serverless emette queste metriche a livello di applicazione, nonché di mansione, tipo di lavoratore e livelli. `capacity-allocation-type`

Per iniziare, utilizza il modello di CloudWatch dashboard EMR Serverless fornito nell'archivio [EMR GitHub Serverless](#) e distribuiscilo.

Note

I [carichi di lavoro interattivi EMR Serverless](#) hanno solo il monitoraggio a livello di applicazione abilitato e hanno una nuova dimensione di tipo di lavoratore, `Spark_Kernel`. [Per monitorare ed eseguire il debug dei carichi di lavoro interattivi, accedi ai log e all'interfaccia utente di Apache Spark dall'area di lavoro di EMR Studio.](#)

Monitoraggio delle metriche

Important

Stiamo ristrutturando la visualizzazione delle nostre metriche per aggiungere `ApplicationName` e `JobName` come dimensioni. Per la versione 7.10 e successive, le metriche precedenti non verranno più aggiornate. Per le versioni EMR precedenti alla 7.10, le metriche precedenti sono ancora disponibili.

Dimensioni attuali

La tabella seguente descrive le dimensioni EMR Serverless disponibili all'interno dello spazio dei nomi. `AWS/EMR Serverless`

Dimensioni per le metriche EMR Serverless

Dimensione	Description	
ApplicationId	Filtri per tutte le metriche di un'applicazione EMR Serverless utilizzando l'ID dell'applicazione.	
ApplicationName	Filtri per tutte le metriche di un'applicazione EMR Serverless utilizzando il nome. Se il nome non viene fornito o contiene caratteri non ASCII, viene pubblicato come [Non specificato].	
JobId	Filtri per tutte le metriche di un server EMR senza l'ID di esecuzione del lavoro.	
JobName	Filtri per tutte le metriche di un job EMR Serverless eseguito utilizzando il nome. Se il nome non viene fornito o contiene caratteri non ASCII, viene pubblicato come [Non specificato].	
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.	
CapacityAllocation Type	Filtri per tutte le metriche di un determinato tipo di allocazione della capacità.	

Dimensione	Description
	Ad esempio, puoi filtrare per la capacità preinizializzata e <code>PreInitCapacity</code> e <code>OnDemandCapacity</code> per tutto il resto.

Monitoraggio a livello di applicazione

Puoi monitorare l'utilizzo della capacità a livello di applicazione EMR Serverless con i parametri di Amazon CloudWatch. Puoi anche configurare un singolo display per monitorare l'utilizzo della capacità delle applicazioni in una dashboard. CloudWatch

Metriche delle applicazioni EMR Serverless

Metrica	Description	Unità	Dimensione
<code>MaxCPUAllowed</code>	La CPU massima consentita per l'applicazione.	VPCU	<code>ApplicationId</code> , <code>ApplicationName</code>
<code>MaxMemoryAllowed</code>	La memoria massima in GB consentita per l'applicazione.	Gigabyte (GB)	<code>ApplicationId</code> , <code>ApplicationName</code>
<code>MaxStorageAllowed</code>	Lo spazio di archiviazione massimo in GB consentito per l'applicazione.	Gigabyte (GB)	<code>ApplicationId</code> , <code>ApplicationName</code>
<code>CPUAllocated</code>	Il numero totale di v CPUs allocato.	VPCU	<code>ApplicationId</code> , <code>ApplicationName</code> , <code>WorkerType</code> e <code>CapacityAllocationType</code>
<code>IdleWorkerCount</code>	Il numero totale di lavoratori inattivi.	Conteggio	<code>ApplicationId</code> , <code>ApplicationName</code> ,

Metrica	Description	Unità	Dimensione
			WorkerType , CapacityAllocationType
MemoryAllocated	Memoria totale in GB allocata.	Gigabyte (GB)	ApplicationId , ApplicationName , WorkerType , CapacityAllocationType
PendingCreationWorkerCount	Il numero totale di lavoratori in attesa di creazione.	Conteggio	ApplicationId , ApplicationName , WorkerType , CapacityAllocationType
RunningWorkerCount	Il numero totale di lavoratori utilizzati dall'applicazione.	Conteggio	ApplicationId , ApplicationName , WorkerType , CapacityAllocationType
StorageAllocated	Lo spazio di archiviazione totale su disco in GB allocato.	Gigabyte (GB)	ApplicationId , ApplicationName , WorkerType , CapacityAllocationType
TotalWorkerCount	Il numero totale di lavoratori disponibili.	Conteggio	ApplicationId , ApplicationName , WorkerType , CapacityAllocationType

Monitoraggio a livello di mansione

Amazon EMR Serverless invia i seguenti parametri a livello di processo ogni minuto. Amazon CloudWatch Puoi accedere ai 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.

Metriche a livello di job EMR Serverless

Metrica	Description	Dimensione
SubmittedJobs	Il numero di lavori in uno stato Inviato.	ApplicationId , ApplicationName
PendingJobs	Il numero di lavori in uno stato In sospeso.	ApplicationId , ApplicationName
ScheduledJobs	Il numero di lavori in uno stato pianificato.	ApplicationId , ApplicationName
RunningJobs	Il numero di lavori in uno stato In esecuzione.	ApplicationId , ApplicationName
SuccessJobs	Il numero di lavori in uno stato di successo.	ApplicationId , ApplicationName
FailedJobs	Il numero di lavori in uno stato Non riuscito.	ApplicationId , ApplicationName
CancellingJobs	Il numero di lavori in uno stato di annullamento.	ApplicationId , ApplicationName
CancelledJobs	Il numero di lavori in uno stato annullato.	ApplicationId , ApplicationName

È possibile monitorare le metriche specifiche del motore per l'esecuzione e il completamento dei job EMR Serverless con un'applicazione specifica del motore. UIs Quando si accede all'interfaccia utente per un processo in esecuzione, l'interfaccia utente live dell'applicazione viene visualizzata con aggiornamenti in tempo reale. Quando si accede all'interfaccia utente per un lavoro completato, viene visualizzata l'interfaccia utente persistente dell'app.

Esecuzione di processi

Per eseguire lavori EMR Serverless, accedi a 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 lavori. Per accedervi UIs, usa la console EMR Studio o richiedi un endpoint URL sicuro con. AWS Command Line Interface

Lavori completati

Per i job EMR Serverless completati, usa lo Spark History Server o l'interfaccia utente Persistent Hive Tez per accedere ai dettagli, alle fasi, alle attività e alle metriche dei job Spark o Hive eseguiti. Per accedervi UIs, usa la console EMR Studio o richiedi un endpoint URL sicuro con. 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 metrici. `Job Worker Metrics` CloudWatch EMR Serverless 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.

Note

Per visualizzare la CPU e la memoria totali utilizzate da un job EMR Serverless durante la visualizzazione delle metriche nella CloudWatch console Amazon, utilizza la statistica come somma e il periodo come 1 minuto.

Metriche EMR Serverless a livello di job worker

Metrica	Description	Unità	Dimensione
<code>WorkerCpuAllocated</code>	Il numero totale di core vCPU allocati per i lavoratori durante l'esecuzione di un job.	VPCU	<code>JobId</code> , <code>JobName</code> , <code>ApplicationId</code> , <code>ApplicationName</code> ,

Metrica	Description	Unità	Dimensione
			WorkerType e CapacityAllocation Type
WorkerCpuUsed	Il numero totale di core vCPU utilizzati dai lavoratori in un job run.	VPCU	JobId, JobName, ApplicationId, ApplicationName, WorkerType e CapacityAllocation Type
WorkerMemoryAllocated	Memoria totale in GB allocata per i lavoratori durante l'esecuzione di un processo.	Gigabyte (GB)	JobId, JobName, ApplicationId, ApplicationName, WorkerType e CapacityAllocation Type

Metrica	Description	Unità	Dimensione
WorkerMemoryUsed	Memoria totale in GB utilizzata dai lavoratori durante l'esecuzione di un processo.	Gigabyte (GB)	JobId, JobName, ApplicationId, ApplicationName, WorkerType e CapacityAllocationType
WorkerEphemeralStorageAllocated	Il numero di byte di storage temporaneo allocato ai lavoratori durante l'esecuzione di un job.	Gigabyte (GB)	JobId, JobName, ApplicationId, ApplicationName, WorkerType e CapacityAllocationType
WorkerEphemeralStorageUsed	Il numero di byte di storage temporaneo utilizzati dai lavoratori durante l'esecuzione di un processo.	Gigabyte (GB)	JobId, JobName, ApplicationId, ApplicationName, WorkerType e CapacityAllocationType

Metrica	Description	Unità	Dimensione
WorkerStorageReadBytes	Il numero di byte letti dallo storage dai lavoratori durante l'esecuzione di un job.	Byte	JobId, JobName, ApplicationId, ApplicationName, WorkerType e CapacityAllocationType
WorkerStorageWriteBytes	Il numero di byte scritti nello storage dai lavoratori durante l'esecuzione di un job.	Byte	JobId, JobName, ApplicationId, ApplicationName, WorkerType e CapacityAllocationType

I passaggi seguenti descrivono come accedere ai vari tipi di metriche.

Console

Per accedere all'interfaccia utente dell'applicazione con la console

1. Accedi alla tua applicazione EMR Serverless su EMR Studio con le istruzioni in [Guida introduttiva](#) dalla console.
2. Per accedere all'applicazione UIs e ai log 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 accedere ai log del motore Spark, vai alla scheda Executors nell'interfaccia utente Spark e scegli il link Logs per il driver. Per accedere ai log del motore Hive, scegli il link Logs per il DAG appropriato nell'interfaccia utente di Hive Tez.
3. Per accedere all'applicazione e ai log 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 accedere ai log del motore Spark, vai alla scheda Executors nell'interfaccia utente Spark e scegli il link Logs per il driver. Per accedere ai log del motore Hive, scegli il link Logs per il DAG appropriato nell'interfaccia utente di Hive Tez.

AWS CLI

Per accedere all'interfaccia utente dell'applicazione con AWS CLI

- Per generare un URL da utilizzare per accedere all'interfaccia utente dell'applicazione per i lavori in esecuzione e completati, chiama l'GetDashboardForJobRunAPI.

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

L'URL generato è valido per un'ora.

Monitora i parametri di Spark con Amazon Managed Service for Prometheus

Con le versioni 7.1.0 e successive di Amazon EMR, puoi integrare EMR Serverless con Amazon Managed Service for Prometheus per raccogliere i parametri di Apache Spark per lavori e applicazioni EMR Serverless. Questa integrazione è disponibile quando si invia un lavoro o si crea un'applicazione utilizzando la AWS console, l'API EMR Serverless o il. AWS CLI

Prerequisiti

Prima di poter fornire i parametri Spark ad Amazon Managed Service for Prometheus, completa i seguenti prerequisiti.

- [Crea un'area di lavoro Amazon Managed Service per Prometheus](#). Questo Workspace funge da endpoint di acquisizione. Prendi nota dell'URL visualizzato per Endpoint - URL di scrittura remota. È necessario specificare l'URL quando si crea l'applicazione EMR Serverless.
- Per concedere l'accesso delle tue offerte di lavoro 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>"
}
```

Configurazione

Per utilizzare la AWS console per creare un'applicazione integrata con Amazon Managed Service for Prometheus

1. Consulta la sezione [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 l'URL di scrittura remota.
4. Specificate le altre impostazioni di configurazione desiderate, quindi scegliete Crea e avvia l'applicazione.

Usa la nostra AWS CLI API serverless EMR

Puoi anche utilizzare l'API AWS CLI o EMR Serverless 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 <APPLICATION_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 per Amazon Managed Service for remoteWriteUrl 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

EMR Serverless utilizza un componente all'interno di Spark denominato `PrometheusServlet` per raccogliere le metriche 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 configurate quando si invia un job Spark che invia metriche ad Amazon Managed Service for Prometheus.

Proprietà Spark	Valore predefinito	Description
<code>spark.metrics.conf</code> <code>.*.sink.prometheusServlet.class</code>	<code>org.apache.spark.metrics.sink.PrometheusServlet</code>	La classe utilizzata da Spark per inviare metriche ad Amazon Managed Service for Prometheus. Per ignorare il comportamento predefinito, specifica la tua classe personalizzata.
<code>spark.metrics.conf</code> <code>.*.source.jvm.class</code>	<code>org.apache.spark.metrics.source.JvmSource</code>	La classe utilizzata da Spark per raccogliere e inviare metriche cruciali dalla macchina virtuale Java sottostante. Per interrompere la raccolta delle metriche JVM, disabilita questa proprietà impostandola su una stringa vuota, ad esempio. "" Per ignorare il comportamento predefinito, specificate la vostra classe personalizzata.
<code>spark.metrics.conf</code> <code>.driver.sink.prometheusServlet.path</code>	<code>/metrics/prometheus</code>	L'URL distinto che Amazon Managed Service for Prometheus utilizza per raccogliere le metriche

Proprietà Spark	Valore predefinito	Description
		dal driver. Per ignorare il comportamento predefinito, specifica il tuo percorso. Per interrompere la raccolta delle metriche dei driver, disattiva te questa proprietà impostandola su una stringa vuota, ad esempio. ""
<code>spark.metrics.conf.executor.sink.prometheusServlet.path</code>	<code>/metrics/executor/prometheus</code>	L'URL distinto che Amazon Managed Service for Prometheus utilizza per raccogliere le metriche dell'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 le metriche di Apache Spark.](#)

Considerazioni e limitazioni

Quando utilizzi Amazon Managed Service for Prometheus per raccogliere metriche da EMR Serverless, considera le seguenti considerazioni e limitazioni.

- Il supporto per l'utilizzo di Amazon Managed Service for Prometheus con EMR Serverless è disponibile solo Regioni AWS nei paesi in [cui 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 si sceglie un worker di dimensioni inferiori, ad esempio un lavoratore vCPU, il tempo di esecuzione del lavoro potrebbe aumentare.

- Il supporto per l'utilizzo di Amazon Managed Service for Prometheus con EMR Serverless è disponibile solo per le versioni 7.1.0 e successive di Amazon EMR.
- Amazon Managed Service for Prometheus deve essere distribuito nello stesso account in cui esegui EMR Serverless per raccogliere i parametri.

Metriche di utilizzo EMR 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.

Le metriche di utilizzo di EMR 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 dei servizi EMR Serverless, fare riferimento a [Endpoint e quote per EMR Serverless](#)

Metriche di utilizzo delle quote di servizio per EMR Serverless

EMR Serverless pubblica le seguenti metriche di utilizzo delle quote di servizio nel namespace. AWS/Usage

Metrica	Description
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

È possibile utilizzare le seguenti dimensioni per perfezionare le metriche di utilizzo pubblicate da EMR Serverless.

Dimensione	Valore	Description
Service	EMR Serverless	Il nome del file che contiene la risorsa. Servizio AWS
Type	Risorsa	Il tipo di entità segnalata da EMR Serverless.
Resource	VPCU	Il tipo di risorsa monitorata da EMR Serverless.
Class	Nessuno	La classe di risorse monitorata da EMR Serverless.

Automazione di EMR Serverless con Amazon EventBridge

È possibile Amazon EventBridge utilizzarlo per automatizzare Servizi AWS e rispondere automaticamente agli eventi di sistema, ad esempio problemi di disponibilità delle applicazioni o modifiche delle risorse. EventBridge offre un flusso quasi in tempo reale di eventi di sistema che descrivono i cambiamenti nelle 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 una funzione AWS Lambda
- Inoltrare un evento ad Amazon Kinesis Data Streams
- Attiva una macchina a stati AWS Step Functions
- Notifica un argomento Amazon SNS o una coda Amazon SQS

Ad esempio, se lo utilizzi EventBridge con EMR Serverless, puoi attivare una AWS Lambda funzione quando un processo ETL ha esito positivo o notificare un argomento di Amazon SNS quando un processo ETL fallisce.

EMR Serverless emette quattro tipi di eventi:

- Eventi di modifica dello stato dell'applicazione: eventi che generano ogni cambiamento di stato di un'applicazione. Per ulteriori informazioni sugli stati delle applicazioni, fare riferimento a [Stati 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, fare riferimento a [Stati 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.
- Eventi di aggiornamento sull'utilizzo delle risorse del lavoro: gli eventi che emettono aggiornamenti sull'utilizzo delle risorse per un processo vengono eseguiti a intervalli prossimi a 30 minuti.

Esempi di eventi EMR Serverless EventBridge

Gli eventi segnalati da EMR Serverless hanno un valore `aws.emr-serverless` assegnato a `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": "00f1cb5c6anuij25",
    "applicationName": "3965ad00-8fba-4932-a6c8-ded32786fd42",
    "arn": "arn:aws:emr-serverless:us-east-1:111122223333:/
applications/00f1cb5c6anuij25",
    "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": "00f1cbb5g4bb0c01",
    "applicationId": "00f1982r1uukb925",
    "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbb5g4bb0c01",
    "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"
}
}

```

Aggiornamento sull'utilizzo delle risorse Job

L'evento di esempio seguente mostra l'aggiornamento finale sull'utilizzo delle risorse per un processo che è passato allo stato terminale dopo l'esecuzione.

```

{
  "version": "0",
  "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
  "detail-type": "EMR Serverless Job Resource Utilization Update",
  "source": "aws.emr-serverless",
  "account": "123456789012",
  "time": "2022-05-31T21:07:42Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:emr-serverless:us-east-1:123456789012:/applications/00f1982r1uukb925/
jobruns/00f1cbn5g4bb0c01"
  ],
  "detail": {
    "applicationId": "00f1982r1uukb925",
    "jobRunId": "00f1cbn5g4bb0c01",

```

```
"attempt": 1,
"mode": "BATCH",
"createdAt": "2022-05-31T21:07:25.325900Z",
"startedAt": "2022-05-31T21:07:26.123Z",
"calculatedFrom": "2022-05-31T21:07:42.299487Z",
"calculatedTo": "2022-05-31T21:07:30.325900Z",
"resourceUtilizationFinal": true,
"resourceUtilizationForInterval": {
  "vCPUHour": 0.023,
  "memoryGBHour": 0.114,
  "storageGBHour": 0.228
},
"billedResourceUtilizationForInterval": {
  "vCPUHour": 0.067,
  "memoryGBHour": 0.333,
  "storageGBHour": 0
},
"totalResourceUtilization": {
  "vCPUHour": 0.023,
  "memoryGBHour": 0.114,
  "storageGBHour": 0.228
},
"totalBilledResourceUtilization": {
  "vCPUHour": 0.067,
  "memoryGBHour": 0.333,
  "storageGBHour": 0
}
}
```

Il campo `StartedAt` sarà presente solo nel caso in cui il lavoro sia passato allo stato di esecuzione.

Applicazione di tag alle risorse

Assegna i tuoi metadati a ciascuna risorsa utilizzando i tag per aiutarti a gestire le tue risorse EMR Serverless. Questa sezione fornisce una panoramica delle funzioni dei tag e mostra come creare tag.

Argomenti

- [Che cos'è un tag?](#)
- [Tagging delle risorse](#)
- [Limitazioni relative all'etichettatura](#)
- [Utilizzo dei tag utilizzando l'API AWS CLI Serverless di Amazon EMR](#)

Che cos'è un tag?

Un tag è un'etichetta che si assegna a una AWS risorsa. Ogni tag consiste di una chiave e di un valore, entrambi personalizzabili. I tag consentono di classificare le AWS risorse in base ad attributi quali scopo, proprietario e ambiente. Quando disponi di molte risorse dello stesso tipo, identifica rapidamente una risorsa specifica in base ai tag ad essa assegnati. Ad esempio, definisci un set di tag per le tue applicazioni Amazon EMR Serverless per monitorare il proprietario e il livello di stack di ciascuna applicazione. Ti suggeriamo di creare un set coerente di chiavi di tag per ogni tipo di risorsa.

I tag non vengono assegnati in automatico alle risorse. Dopo aver aggiunto un tag a una risorsa, modificate il valore di un tag o rimuovete il tag 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 usi IAM, puoi controllare quali utenti del tuo AWS account sono autorizzati a gestire i tag. Per esempi di policy di controllo degli accessi 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 l'API Serverless di Amazon EMR o un AWS SDK AWS CLI, puoi applicare tag a nuove risorse utilizzando il `tags` parametro sull'azione API pertinente. Puoi applicare tag a risorse esistenti utilizzando l'operazione API `TagResource`.

Puoi utilizzare alcune operazioni per la creazione di risorse per specificare tag per una risorsa durante la sua creazione. In questo caso, se i tag non possono essere applicati durante la creazione della risorsa, la risorsa non viene creata. Mediante questo meccanismo, le risorse a cui desideri applicare tag al momento della creazione vengono create con tag specifici o non vengono create affatto. Se tagghi le risorse al momento della creazione, non è necessario eseguire script di tagging personalizzati dopo aver creato una risorsa.

La tabella seguente descrive le risorse Serverless di Amazon EMR che possono essere taggate.

Risorse compatibili con l'applicazione di tag

Risorsa	Supporta tag	Supporta la propagazione di tag	Supporta l'etichettatura alla creazione (Amazon EMR Serverless API AWS CLI e SDK) AWS	API per la creazione (i tag possono essere aggiunti durante la creazione)
Applicazione	Sì	No. I tag associati a un'applicazione non si propagano alle esecuzioni di job inviate a tale applicazione.	Sì	CreateApplication
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 ogni risorsa, ogni chiave di tag deve essere unica e ogni chiave di tag può avere un solo valore.

- La lunghezza massima delle chiavi è 128 caratteri Unicode in UTF-8.
- Il valore massimo è 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. Essi sono riservati per l'utilizzo esclusivo di AWS .

Utilizzo dei tag utilizzando l'API AWS CLI Serverless di Amazon EMR

Utilizza i seguenti AWS CLI comandi o le operazioni API Serverless di Amazon EMR per aggiungere, aggiornare, elencare ed eliminare i tag per le tue risorse.

Comandi CLI e operazioni API per i tag

Risorsa	Supporta tag	Supporta la propagazione di tag
Aggiunta o sovrascrittura di uno o più tag	<code>tag-resource</code>	TagResource
Elencare i tag associati a una risorsa	<code>list-tags-for-resource</code>	ListTagsForResource
Eliminazione di uno o più tag	<code>untag-resource</code>	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 EMR Serverless

Questa sezione descrive i casi d'uso comuni quando si lavora con applicazioni EMR Serverless. Ciò include una varietà di strumenti tra cui Hudi e Iceberg per lavorare su set di dati di grandi dimensioni e utilizzare le librerie Python e Python per inviare lavori Spark.

Argomenti

- [Utilizzo di Java 17 con Amazon EMR Serverless](#)
- [Utilizzo di Apache Hudi con EMR Serverless](#)
- [Utilizzo di Apache Iceberg con EMR Serverless](#)
- [Utilizzo delle librerie Python con EMR Serverless](#)
- [Utilizzo di diverse versioni di Python con EMR Serverless](#)
- [Utilizzo di Delta Lake OSS con EMR Serverless](#)
- [Invio di lavori EMR Serverless da Airflow](#)
- [Utilizzo delle funzioni Hive definite dall'utente con EMR Serverless](#)
- [Utilizzo di immagini personalizzate con EMR Serverless](#)
- [Utilizzo dell'integrazione di Amazon Redshift per Apache Spark su Amazon EMR Serverless](#)
- [Connessione a DynamoDB con Amazon EMR Serverless](#)

Utilizzo di Java 17 con Amazon EMR Serverless

Con le versioni 6.11.0 e successive di Amazon EMR, configura i job EMR Serverless Spark per utilizzare il runtime Java 17 per la Java Virtual Machine (JVM). Utilizza uno dei seguenti metodi per configurare Spark con Java 17.

JAVA_HOME

Per sovrascrivere l'impostazione JVM per EMR Serverless 6.11.0 e versioni successive, fornire l'impostazione alle relative classificazioni e a quelle di ambiente. `JAVA_HOME spark.emr-serverless.driverEnv spark.executorEnv`

`x86_64`

Imposta le proprietà richieste per specificare Java 17 come configurazione per il driver e gli `JAVA_HOME` 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, è possibile specificare Java 17 nella spark-defaults classificazione per sovrascrivere l'impostazione JVM 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 EMR Serverless

Questa sezione descrive l'utilizzo di Apache Hudi con le applicazioni EMR Serverless. Hudi è un framework di gestione dei dati che semplifica l'elaborazione dei dati.

Per utilizzare Apache Hudi con applicazioni EMR Serverless

1. Imposta le proprietà Spark richieste nell'esecuzione del job Spark corrispondente.

```
spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,/usr/lib/hudi/hudi-utilities-  
bundle.jar,/usr/lib/hudi/hudi-aws-bundle.jar  
spark.serializer=org.apache.spark.serializer.KryoSerializer
```

2. Per sincronizzare una tabella Hudi con il catalogo configurato, designa AWS Glue Data Catalog come metastore o configura un metastore esterno. EMR Serverless supporta la modalità di sincronizzazione per hms le tabelle Hive per i carichi di lavoro Hudi. EMR Serverless attiva questa proprietà come impostazione predefinita. Per ulteriori informazioni su come configurare il metastore, fare riferimento a [Configurazione Metastore per EMR Serverless](#)

Important

EMR Serverless non supporta HIVEQL né offre opzioni di modalità di sincronizzazione per JDBC le tabelle Hive per gestire i carichi di lavoro Hudi. [Per saperne di più, consulta le modalità di sincronizzazione.](#)

Quando utilizzate il AWS Glue Data Catalog come metastore, specificate le seguenti proprietà di configurazione per il vostro job 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.AWSGlueDataCatalogHiveMetastore
```

[Per ulteriori informazioni sulle versioni di Apache Hudi di Amazon EMR, consulta la cronologia delle versioni di Hudi.](#)

Utilizzo di Apache Iceberg con EMR Serverless

Questa sezione descrive come utilizzare Apache Iceberg con le applicazioni EMR Serverless. Apache Iceberg è un formato di tabella che aiuta a lavorare con set di dati di grandi dimensioni nei data lake.

Per utilizzare Apache Iceberg con applicazioni EMR Serverless

1. Imposta le proprietà Spark richieste nell'esecuzione del job Spark corrispondente.

```
spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
```

2. Designate il AWS Glue Data Catalog come metastore o configurate un metastore esterno. Per ulteriori informazioni sulla configurazione del metastore, consulta [Configurazione Metastore per EMR Serverless](#)

Configura le proprietà del metastore che desideri utilizzare per Iceberg. Ad esempio, se desideri utilizzare il AWS Glue Data Catalog, imposta le seguenti proprietà nella configurazione dell'applicazione.

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

Quando utilizzi AWS Glue Data Catalog come metastore, specifica le seguenti proprietà di configurazione per il tuo job 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://amzn-s3-demo-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 Amazon EMR, consulta la cronologia delle versioni di Iceberg.](#)

Utilizzo delle librerie Python con EMR Serverless

Quando esegui PySpark lavori su applicazioni Amazon EMR Serverless, impacchetta diverse librerie Python come dipendenze. Per fare ciò, usa le funzionalità native di Python, crea un ambiente virtuale o configura direttamente i tuoi PySpark lavori per utilizzare le librerie Python. Questa pagina illustra ogni approccio.

Utilizzo delle funzionalità native di Python

Quando imposti la seguente configurazione, usala PySpark per caricare file Python (.py), pacchetti Python compressi (.egg) e file .zip negli esecutori Spark.

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

Per maggiori dettagli su come utilizzare gli ambienti virtuali Python per i PySpark lavori, consulta [Uso delle funzionalità PySpark native](#).

Quando usi EMR Notebook, puoi rendere disponibile la dipendenza Python nel tuo Notebook eseguendo il seguente codice:

```
%%configure -f
```

```
{
  "conf": {
    "spark.submit.pyFiles": "s3:///amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip
file>
  }
}
```

Creazione di un ambiente virtuale Python

Per impacchettare più librerie Python per un PySpark lavoro, crea 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 EMR release 6.6.0. È possibile trovare un Dockerfile di esempio nel repository [EMR Serverless Samples](#). [GitHub](#)

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

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

# install the python packages
pip3 install scipy
pip3 install matplotlib

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

# copy the archive to an S3 location
aws s3 cp pyspark_venv.tar.gz s3:///amzn-s3-demo-bucket/EXAMPLE-PREFIX/
```

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

2. Invia il job Spark con le proprietà impostate per utilizzare l'ambiente virtuale Python.

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

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, consulta [Using Virtualenv](#). Per altri esempi su come inviare lavori Spark, consulta. [Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless](#)

Configurazione dei PySpark lavori per l'utilizzo delle librerie Python

[Con le versioni 6.12.0 e successive di Amazon EMR, puoi configurare direttamente i job EMR Serverless PySpark per utilizzare le più diffuse librerie Python di data science 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 su cui si basa. NumPy La libreria pandas fornisce ai data scientist strutture di dati e strumenti di analisi [DataFrame](#)dei 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 EMR Serverless

Oltre allo use case in [Utilizzo delle librerie Python con EMR Serverless](#), puoi anche utilizzare ambienti virtuali Python per lavorare con versioni di Python diverse rispetto alla versione inclusa nella release Amazon EMR per la tua applicazione Amazon EMR Serverless. Per fare ciò, crea 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 versioni 7.0.0 e successive di Amazon EMR, esegui 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, esegui 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 --enable-shared && \
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 and shared libraries to venv
cp -r /usr/local/lib/python3.9/* ./pyspark_venv_python_3.9.9/lib/python3.9/
cp /usr/local/lib/libpython3.9* ./pyspark_venv_python_3.9.9/lib/

# 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://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
pyspark_venv_python_3.9.9.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.emr-serverless.driverEnv.LD_LIBRARY_PATH=./environment/lib
--conf spark.executorEnv.LD_LIBRARY_PATH=./environment/lib

```

Per ulteriori informazioni su come utilizzare gli ambienti virtuali Python per i PySpark lavori, consulta [Using Virtualenv](#). Per altri esempi su come inviare lavori Spark, consulta [Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless](#)

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 utilizzi Amazon EMR 7.0.0 o versioni successive, assicurati di `delta-spark.jar` specificarlo nelle configurazioni.

Amazon EMR 6.9.0 e versioni successive includono Delta Lake, quindi non devi più impacchettare Delta Lake personalmente o fornire il flag per i tuoi lavori EMR `--packages 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://amzn-s3-demo-bucket/delta-lake/output/%s/" % str(uuid.uuid4())
spark = SparkSession.builder.appName("DeltaSample").getOrCreate()

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

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

- Utilizzando AWS CLI, carica il `delta_sample.py` file nel tuo bucket Amazon S3. Quindi utilizzare il `start-job-run` comando per inviare un lavoro a un'applicazione EMR Serverless esistente.

```
aws s3 cp delta_sample.py s3://amzn-s3-demo-bucket/code/

aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --name emr-delta \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://amzn-s3-demo-bucket/code/delta_sample.py",
      "sparkSubmitParameters": "--conf spark.jars=/usr/share/
aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/delta-storage.jar --
conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
    }
  }'
```

Per usare le librerie Python con Delta Lake, aggiungi la `delta-core` libreria [impacchettandola come dipendenza](#) o [usandola come immagine personalizzata](#).

In alternativa, puoi usare `SparkContext.addPyFile` per aggiungere le librerie Python dal file `delta-core` JAR:

```
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.

- Per creare una versione open source di [Delta Lake](#) compatibile con la versione di Spark sulla tua applicazione Amazon EMR Serverless, accedi a GitHub Delta e segui [le](#) istruzioni.
- Carica le librerie Delta Lake in un bucket Amazon S3 nel tuo Account AWS

- Quando invii lavori EMR Serverless nella configurazione dell'applicazione, includi i file JAR di Delta Lake che ora sono nel tuo bucket.

```
--conf spark.jars=s3://amzn-s3-demo-bucket/jars/delta-core_2.12-1.1.0.jar
```

- 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://amzn-s3-demo-bucket/delta-lake/output/1.0.1/%s/" %
str(uuid.uuid4())

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

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

Invio di lavori EMR Serverless da Airflow

Il provider Amazon 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 file e aggiorna l'ambiente MWAA per utilizzare `requirements.txt` 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 su MWAA.

Il seguente esempio abbreviato mostra come creare un'applicazione, eseguire più job Spark e quindi arrestare l'applicazione. Un esempio completo è disponibile nell'archivio [EMR Serverless Samples](#). GitHub Per ulteriori dettagli sulla `sparkSubmit` configurazione, fare riferimento a [Utilizzo delle configurazioni Spark quando si eseguono job EMR Serverless](#)

```
from datetime import datetime

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

# Replace these with your correct values
JOB_ROLE_ARN = "arn:aws:iam::account-id:role/emr_serverless_default_role"
S3_LOGS_BUCKET = "amzn-s3-demo-bucket"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://amzn-s3-demo-bucket/logs/"}
    },
}

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

```
application_id = create_app.output

job1 = EmrServerlessStartJobOperator(
    task_id="start_job_1",
    application_id=application_id,
    execution_role_arn=JOB_ROLE_ARN,
    job_driver={
        "sparkSubmit": {
            "entryPoint": "local:///usr/lib/spark/examples/src/main/python/
pi_fail.py",
        }
    },
    configuration_overrides=DEFAULT_MONITORING_CONFIG,
)

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

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

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

Utilizzo delle funzioni Hive definite dall'utente con EMR Serverless

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 UDF di esempio 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 [Inizia a usare Amazon EMR Serverless](#)

Per utilizzare un UDF con EMR Serverless

1. Vai alla pagina UDF [GitHub](#) per un esempio. 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 file JAR che contiene il tuo campione UDFs.
2. Dopo aver creato il file JAR, caricalo nel tuo bucket S3 con il seguente comando.

```
aws s3 cp brickhouse-0.8.2-JS.jar s3://amzn-s3-demo-bucket/jars/
```

3. Crea un file di esempio per utilizzare una delle funzioni UDF di esempio. Salva questa query con nome `udf_example.q` e caricala nel tuo bucket S3.

```
add jar s3://amzn-s3-demo-bucket/jars/brickhouse-0.8.2-JS.jar;
CREATE TEMPORARY FUNCTION from_json AS 'brickhouse.udf.json.FromJsonUDF';
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
  array(cast(0 as int))));
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
  array(cast(0 as int))))["key1"][2];
```

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

```
    ]],  
    "monitoringConfiguration": {  
      "s3MonitoringConfiguration": {  
        "logUri": "s3://amzn-s3-demo-bucket/logs/"  
      }  
    }  
  }  
}'
```

5. 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://amzn-s3-demo-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](#)
- [Usa una versione Java personalizzata](#)
- [Crea un'immagine di data science](#)
- [Elaborazione di dati geospaziali con Apache Sedona](#)
- [Informazioni sulle licenze per l'utilizzo di immagini personalizzate](#)

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 runs 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 amazon-linux-extras install java-openjdk11

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

Informazioni sulle licenze per l'utilizzo di immagini personalizzate

È possibile creare immagini personalizzate con EMR Serverless per eseguire attività specifiche o utilizzare versioni specifiche di un pacchetto software. La modifica e la distribuzione di immagini personalizzate possono essere soggette a regole e termini di licenza. Il testo della licenza viene visualizzato nella sottosezione che segue.

Licenze che si applicano alle immagini personalizzate

Copyright Amazon.com e le sue affiliate; tutti i diritti riservati. Questo software è AWS contenuto ai sensi [AWS del Contratto con il cliente](#) e non può essere distribuito senza autorizzazione. Oltre alle autorizzazioni previste dalla licenza di [proprietàAWS intellettuale, il AWS Licenziante](#) concede all'utente le seguenti autorizzazioni aggiuntive:

È consentito creare, copiare e utilizzare derivati del AWS contenuto a condizione che siano soddisfatte le seguenti condizioni:

- L'utente non modifica il AWS Contenuto stesso e gli eventuali derivati sono strettamente il risultato dell'aggiunta di nuovi contenuti da parte dell'Utente.
- Le riproduzioni interne devono conservare la suddetta nota sul copyright.
- La distribuzione esterna, in formato sorgente o binario, con o senza modifiche, non è consentita dai termini di questa licenza.

Per ulteriori informazioni sull'utilizzo di immagini personalizzate, fare riferimento a [Utilizzo di immagini personalizzate con EMR](#) Serverless.

Utilizzo dell'integrazione di Amazon Redshift per Apache Spark su Amazon EMR Serverless

Con Amazon EMR rilascio 6.9.0 e successivi, ogni immagine del rilascio include un connettore tra [Apache Spark](#) e Amazon Redshift. Con questo connettore, usa 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, passa le dipendenze Spark-Redshift richieste con il tuo job Spark. `--jars` Da utilizzare per includere le librerie relative al connettore Redshift. Per accedere ad altre posizioni di file supportate dall'`--jars` opzione, consultate la sezione [Advanced Dependency Management](#) della documentazione di Apache Spark.

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

Le versioni 6.10.0 e successive di Amazon EMR non richiedono la dipendenza `minimal-json.jar` e installano automaticamente le altre dipendenze su ciascun cluster per impostazione predefinita. I seguenti esempi mostrano come avviare un'applicazione Spark con l'integrazione 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 su EMR Serverless versione 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 su EMR Serverless release 6.9.0, utilizza l'opzione come illustrato nell'esempio seguente. `--jars` Come vedrai, i percorsi elencati con l'opzione `--jars` 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

Utilizzalo Gestione dei segreti AWS per recuperare le credenziali e connetterti ad Amazon Redshift

Puoi autenticarti in modo sicuro su Amazon Redshift archiviando le credenziali in Secrets Manager e facendo in modo che il job Spark `GetSecretValue` chiami l'API 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
```

Autenticazione su Amazon Redshift con un driver JDBC

Impostazione di nome utente e password all'interno dell'URL di JDBC

Puoi autenticare un job Spark in un cluster Amazon Redshift specificando il nome e la password del database Amazon Redshift nell'URL JDBC.

Note

Se passi le credenziali del database nell'URL, anche chiunque abbia accesso all'URL può accedere alle credenziali. Questo metodo non è generalmente consigliato perché non è un'opzione sicura.

Se la sicurezza non è un problema per la tua applicazione, utilizza il seguente formato per impostare il nome utente e la password nell'URL JDBC:

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

Usa l'autenticazione basata su IAM con il ruolo di esecuzione dei job di Amazon EMR Serverless

A partire dalla versione 6.9.0 di Amazon EMR Serverless, il driver Amazon Redshift JDBC 2.1 o versione successiva viene integrato nell'ambiente. Con il driver JDBC 2.1 e versioni successive, è possibile specificare l'URL di JDBC evitando di includere il nome utente e la password non elaborati.

Specificate invece lo schema. `jdbc:redshift:iam://` Ciò ordina al driver JDBC di utilizzare il ruolo di esecuzione del lavoro EMR Serverless per recuperare automaticamente le credenziali. Per ulteriori informazioni, consulta [Configurare una connessione JDBC o ODBC per utilizzare le credenziali IAM](#) nella Amazon Redshift Management Guide. Un esempio di questo URL è:

```
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
<code>redshift:GetClusterCredentials</code>	Obbligatoria affinché il driver JDBC recuperi le credenziali da Amazon Redshift
<code>redshift:DescribeCluster</code>	Obbligatoria se specifichi il cluster Amazon Redshift e Regione AWS nell'URL di JDBC anziché nell'endpoint
<code>redshift-serverless:GetCredentials</code>	Obbligatoria affinché il driver JDBC recuperi le credenziali da Amazon Redshift Serverless
<code>redshift-serverless:GetWorkgroup</code>	Obbligatorio se utilizzi Amazon Redshift Serverless e specifichi l'URL in termini di nome e regione del gruppo di lavoro

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 provisioning sotto un VPC, configura la connettività VPC per la tua applicazione Amazon EMR Serverless per accedere alle risorse. Per ulteriori informazioni su come configurare la connettività VPC su un'applicazione EMR Serverless, fare riferimento a [Configurazione dell'accesso VPC per le applicazioni EMR Serverless per la connessione ai dati](#)

- Se il cluster Amazon Redshift o il gruppo di lavoro Serverless Amazon Redshift fornito sono accessibili al pubblico, specifica una o più sottoreti private a cui è collegato un gateway NAT quando crei applicazioni EMR Serverless.
- Se il cluster Amazon Redshift o il gruppo di lavoro Serverless Amazon Redshift fornito non sono accessibili al pubblico, devi creare un endpoint VPC gestito da Amazon Redshift per il tuo cluster Amazon Redshift come descritto in [Configurazione dell'accesso VPC per le applicazioni EMR Serverless per la connessione ai dati](#). 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 EMR Serverless.

Note

Se utilizzi l'autenticazione basata su IAM e le tue sottoreti private per l'applicazione EMR Serverless non dispongono di un gateway NAT collegato, devi anche creare un endpoint VPC su tali sottoreti per Amazon Redshift o Amazon Redshift Serverless. In questo modo, il driver JDBC può recuperare le credenziali.

Lettura e scrittura da e su Amazon Redshift

I seguenti esempi di codice consentono PySpark di leggere e scrivere dati di esempio da e verso un database Amazon Redshift con un'API di origine dati e con SparkSQL.

Data source API

PySpark Utilizzalo per leggere e scrivere dati di esempio da e verso un database Amazon Redshift con API di origine dati.

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
sql_context = SQLContext(sc)

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::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 SparkSQL.

```
import boto3  
import json  
import sys  
import os  
from pyspark.sql import SparkSession  
  
spark = SparkSession \  
    .builder \  
    .enableHiveSupport() \  
    .getOrCreate()  
  
url = "jdbc:redshift:iam://redshifthost:5439/database"  
aws_iam_role_arn = "arn:aws:iam::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 per la connessione JDBC da Spark su Amazon EMR ad Amazon Redshift.
- Ti suggeriamo di gestire le credenziali per il cluster Amazon Redshift come Gestione dei segreti AWS best practice. Per un esempio, consulta [Utilizzo Gestione dei segreti AWS per recuperare le credenziali per la connessione ad Amazon Redshift](#).
- Ti suggeriamo di passare un ruolo IAM con il parametro `aws_iam_role` per il parametro di autenticazione Amazon Redshift.
- Il parametro `tempformat` attualmente non supporta il formato Parquet.
- L'URI `tempdir` indica una posizione Amazon S3. Questa directory temporanea non viene pulita in automatico e quindi potrebbe generare costi aggiuntivi.
- Prendi in considerazione i seguenti consigli per Amazon Redshift:
 - Ti suggeriamo di bloccare l'accesso pubblico al cluster Amazon Redshift.
 - Ti consigliamo di attivare la registrazione di [controllo di Amazon Redshift](#).
 - Ti consigliamo di attivare la crittografia [at-rest di Amazon Redshift](#).
- Prendi in considerazione i seguenti consigli per Amazon S3:
 - Ti consigliamo di [bloccare l'accesso pubblico ai bucket Amazon S3](#).
 - Ti consigliamo di utilizzare la [crittografia lato server di Amazon S3 per crittografare](#) i bucket Amazon S3 utilizzati.
 - Ti suggeriamo di utilizzare le [policy del ciclo di vita di Amazon S3](#) per definire le regole di conservazione per il bucket Amazon S3.
 - Amazon EMR verifica 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 delle chiavi di AWS accesso nella classificazione della configurazione `hadoop-env`
 - Codifica delle chiavi di AWS accesso nell'URI `tempdir`

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 EMR Serverless

In questo tutorial, carichi un sottoinsieme di dati dal [Board on Geographic Names degli Stati Uniti su](#) un bucket Amazon S3 e poi usi Hive o Spark su Amazon EMR Serverless per copiare i dati in una tabella Amazon DynamoDB per l'esecuzione di 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 *amzn-s3-demo-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 accedi alle prime righe del file:

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

Il risultato dovrebbe apparire 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://amzn-s3-demo-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://amzn-s3-demo-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 query SQL 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://amzn-s3-demo-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://amzn-s3-demo-bucket/ddb-connector/")
      .map(row => {
        val line = row.split("\\|")
        val item = new DynamoDBItemWritable()

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

        item.setItem(Map(
          "feature_id" -> new AttributeValue().withN(line(0)),
          "feature_name" -> new AttributeValue(line(1)),
          "feature_class" -> new AttributeValue(line(2)),
          "state_alpha" -> new AttributeValue(line(3)),
          "prim_lat_dec" -> new AttributeValue().withN(line(4)),
          "prim_long_dec" -> new AttributeValue().withN(line(5)),
          "elev_in_ft" -> elevInFt)
          .asJava)
          (new Text(""), item)
        ))
    rdd.saveAsHadoopDataset(jobConf)
  }
}

```

Hive

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, usa Spark SQL o l'API Spark. MapReduce

Example— Interroga la tua tabella DynamoDB con Spark SQL

La seguente query SQL Spark restituisce un elenco di tutti i tipi di funzionalità in ordine alfabetico.

```
val dataframe = sparkSession.sql("SELECT DISTINCT feature_class \
FROM ddb_features \
ORDER BY feature_class;")
```

La seguente query SQL Spark restituisce un elenco di tutti i lake 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 query SQL Spark restituisce un elenco di tutti gli stati con almeno tre funzionalità 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 l'API Spark MapReduce

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, completare 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 ruolo Cross-Account-Role-B IAM in AccountB grado di accedere alla tabella DynamoDB.
 - a. Accedi Console di gestione AWS e apri la console IAM 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 ruoli IAM, consulta [Creazione di ruoli IAM](#) nella guida per l'utente.
 - c. Crea una policy IAM che conceda le autorizzazioni per accedere alla tabella DynamoDB tra account. Quindi, allega la policy IAM a Cross-Account-Role-B.

Di seguito è riportata una politica che concede l'accesso a una tabella DynamoDB.
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 console IAM per il ruolo che hai creato nel passaggio 2.: Cross-Account-Role-B

Seleziona Modifica relazione di fiducia, quindi aggiungi il seguente documento politico. Questo documento consente Job-Execution-Role-A AccountA di assumere questo Cross-Account-Role-B ruolo.

JSON

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

```
}

```

- e. Concedi Job-Execution-Role-A AccountA con - STS Assume role le autorizzazioni da assumereCross-Account-Role-B.

Nella console IAM per Account AWS AccountA, selezionaJob-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.

JSON

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

- 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 valore ARN di. Cross-Account-Role-B
3. Esegui il job Spark o Hive utilizzando. Job-Execution-Role-A

Considerazioni

Nota questi comportamenti e limitazioni quando utilizzi il connettore DynamoDB con Apache Spark o Apache Hive.

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 l'operazione con lo storage handler. STORED BY 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. Il predicate pushdown funziona solo quando si utilizza il connettore con Spark SQL e non con l'API. MapReduce

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 su EMR 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 mappatura 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, impostate il valore della `mapreduce.job.maps` proprietà in base al numero di attività nella fase finale o alla velocità di scrittura che desiderate allocare per attività. Imposta questa proprietà nella `mapred-site` classificazione del `start-job-run` comando su EMR Serverless.

Sicurezza

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. 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 che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che utilizzi in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei [programmi di conformitàAWS](#). Per maggiori informazioni sui programmi di conformità che si applicano ad Amazon EMR Serverless, consulta [AWS i servizi inclusi 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 spiegano come configurare Amazon EMR Serverless e utilizzare altri AWS servizi per raggiungere i tuoi obiettivi di sicurezza e conformità.

Argomenti

- [Best practice di sicurezza per Amazon EMR Serverless](#)
- [Protezione dei dati](#)
- [Identity and Access Management \(IAM\) in Amazon EMR Serverless](#)
- [Propagazione attendibile delle identità](#)
- [Utilizzo di Lake Formation con EMR Serverless](#)
- [Crittografia tra lavoratori](#)
- [Crittografia del disco con KMS CMK](#)
- [Secrets Manager per la protezione dei dati con EMR Serverless](#)
- [Utilizzo di Amazon S3 Access Grants con EMR Serverless](#)
- [Registrazione delle chiamate API Serverless di Amazon EMR utilizzando AWS CloudTrail](#)
- [Convalida della conformità per Amazon EMR Serverless](#)

- [Resilienza in Amazon EMR Serverless](#)
- [Sicurezza dell'infrastruttura in Amazon EMR Serverless](#)
- [Analisi della configurazione e delle vulnerabilità in Amazon EMR Serverless](#)

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, sono da considerare come considerazioni utili anziché prescrizioni.

Applicazione del principio del privilegio minimo

EMR Serverless fornisce una policy di accesso granulare per le applicazioni che utilizzano ruoli IAM, come i ruoli di esecuzione. Sugeriamo che ai ruoli di esecuzione venga concesso solo il set minimo di privilegi richiesti dal job, come la copertura dell'applicazione e l'accesso alla destinazione dei 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

EMR Serverless crea l'isolamento completo della rete tra i lavori appartenenti a diverse applicazioni EMR Serverless. Nei casi in cui si desidera l'isolamento a livello di processo, si consiglia di isolare i lavori in diverse applicazioni EMR Serverless.

Autorizzazioni per il controllo degli accessi basato su ruoli (RBAC)

Gli amministratori devono controllare rigorosamente le autorizzazioni RBAC (Role-based access control) per le applicazioni EMR Serverless.

Protezione dei dati

Il [modello di responsabilità AWS 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 tutto il AWS cloud. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Questo contenuto include le attività di configurazione e gestione della sicurezza per i AWS servizi che utilizzi. Per ulteriori informazioni sulla privacy dei dati, consulta le [Domande](#)

[frequenti sulla privacy dei dati](#). Per informazioni sulla protezione dei dati in Europa, consulta [il modello di responsabilità AWS condivisa e il post sul blog sul GDPR](#) sul AWS Security Blog.

Ai fini della protezione dei dati, ti suggeriamo di proteggere le credenziali degli AWS account e di configurare 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.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. Suggeriamo TLS 1.2 o versione successiva.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno AWS dei servizi.
- Utilizza i servizi di sicurezza gestiti avanzati, ad esempio Amazon Macie, che aiutano a individuare e proteggere i dati personali archiviati in Amazon S3.
- Utilizza le opzioni di crittografia Serverless di Amazon EMR per crittografare i dati a riposo e in transito.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, fare riferimento al [Federal](#) Information Processing Standard (FIPS) 140-2.

Ti consigliamo vivamente di non inserire mai informazioni identificative sensibili, come i numeri di conto dei tuoi clienti, in campi in formato libero come il campo Nome. Ciò include quando lavori con Amazon EMR Serverless o altri AWS servizi utilizzando la console, l'API o gli SDK. AWS CLI AWS 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 URL a un server esterno, non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta a tale server.

Crittografia a riposo

La crittografia dei dati consente di impedire agli utenti non autorizzati di leggere dati su un cluster e sui sistemi di archiviazione di dati associati. Sono inclusi i dati salvati su supporti persistenti, noti come dati a riposo, e i dati che possono essere intercettati quando circolano in rete, noti come dati in transito.

La crittografia dei dati richiede chiavi e certificati. Puoi scegliere tra diverse opzioni, tra cui chiavi gestite da AWS Key Management Service, chiavi gestite da Amazon S3 e chiavi e certificati di fornitori personalizzati da te forniti. Quando lo utilizzi AWS KMS come fornitore di chiavi, vengono addebitati costi per l'archiviazione e l'uso delle chiavi di crittografia. Per ulteriori informazioni, consulta la pagina [AWS KMS dei prezzi](#).

Prima di specificare le opzioni di crittografia, scegli i sistemi di gestione di chiavi e certificati che intendi utilizzare. Quindi, crea le chiavi e i certificati per i provider personalizzati specificati come parte delle impostazioni di crittografia.

Crittografia dei dati a riposo per dati EMRFS in Amazon S3

Ogni applicazione EMR Serverless utilizza una versione di release specifica, che include EMRFS (EMR File System). La crittografia di Amazon S3 funziona con gli oggetti del file system EMR (EMRFS) letti da e scritti su Amazon S3. Puoi specificare la crittografia lato server (SSE) o la crittografia lato client (CSE) di Amazon S3 come modalità di crittografia predefinita quando abiliti la crittografia a riposo. Facoltativamente, specifica diversi metodi di crittografia per singoli bucket utilizzando le sostituzioni di crittografia Per bucket. Indipendentemente dall'abilitazione o meno della crittografia di Amazon S3, il protocollo Transport Layer Security (TLS) esegue la crittografia degli oggetti EMRFS in transito tra i nodi cluster EMR e Amazon S3. 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 la sezione [Protezione dei dati mediante](#) crittografia nella Amazon Simple Storage Service Developer Guide.

Note

Quando si utilizza AWS KMS, vengono addebitati costi per l'archiviazione e l'uso delle chiavi di crittografia. Per ulteriori informazioni, consulta la pagina [AWS KMS dei prezzi](#).

Crittografia lato server di Amazon S3

Tutti i bucket Amazon S3 hanno la crittografia configurata di default e tutti i nuovi oggetti caricati su un bucket S3 vengono crittografati automaticamente quando sono inattivi. Amazon S3 crittografa i dati a livello di oggetto mentre scrive i dati su disco e decrittografa i dati quando vi si accede. Per ulteriori informazioni su SSE, 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 specifichi SSE in Amazon EMR Serverless:

- SSE-S3- Amazon S3 gestisce le chiavi per te. Non è richiesta alcuna configurazione aggiuntiva su EMR Serverless.
- SSE-KMS- Si utilizza un AWS KMS key per configurare policy adatte a EMR Serverless. Non è richiesta alcuna configurazione aggiuntiva su EMR Serverless.

 Tip

Per ridurre AWS KMS i costi di utilizzo SSE-KMS, prendi in considerazione la possibilità di abilitare Amazon S3 Bucket Keys sui tuoi bucket Amazon S3. Le chiavi Bucket di Amazon S3 utilizzano una chiave a livello di bucket di breve durata per ridurre le chiamate AWS KMS API fino al 99 per cento. Prima di abilitare Amazon S3 Bucket Keys, esamina il tuo IAM e le policy AWS KMS chiave: il contesto di crittografia cambia dall'ARN dell'oggetto Amazon S3 all'ARN del bucket, il che può influire sulle policy che utilizzano l'oggetto ARN per il controllo degli accessi. Per ulteriori informazioni, consulta [Ridurre il costo di SSE-KMS con Amazon S3 Bucket Keys](#) nella Guida per l'utente di Amazon Simple Storage Service.

Per utilizzare AWS KMS la crittografia per i dati che scrivi su Amazon S3, hai due opzioni quando usi l'`StartJobRunAPI`. Puoi abilitare la crittografia per tutto ciò che scrivi su Amazon S3 o abilitare la crittografia per i dati che scrivi in un bucket specifico. Per ulteriori informazioni sull'`StartJobRunAPI`, fare riferimento all'[EMR Serverless](#) API Reference.

Per attivare AWS KMS la crittografia per tutti i dati che scrivi su Amazon S3, usa i seguenti comandi quando chiami l'`StartJobRunAPI`.

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

Per attivare AWS KMS la crittografia dei dati che scrivi in un bucket specifico, usa i seguenti comandi quando chiami l'`StartJobRunAPI`.

```
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-bucket1>.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-
bucket1>.serverSideEncryption.kms.keyId=<kms-id>
```

SSE con chiavi fornite dal cliente (SSE-C) non è disponibile per l'uso con EMR Serverless.

Crittografia lato client Amazon S3

Con la crittografia lato client di Amazon S3, la crittografia e la decrittografia di Amazon S3 avvengono nel client EMRFS disponibile in ogni versione di Amazon 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 root 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 KMS aggiuntivi. 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 un algoritmo crittografico standard del settore. AES-256

Gestione delle chiavi

È possibile configurare KMS per ruotare in automatico le chiavi KMS. Questa impostazione effettua una rotazione delle chiavi una volta all'anno e allo stesso tempo salva le vecchie chiavi a tempo indeterminato, in modo che i dati possano ancora essere decrittati. [Per ulteriori informazioni, consulta Rotazione delle chiavi gestite dal cliente.](#)

Crittografia dei dati in transito

Le seguenti funzionalità di crittografia specifiche dell'applicazione sono disponibili con Amazon EMR Serverless:

- Spark
 - Per impostazione predefinita, la comunicazione tra i driver Spark e gli esecutori è autenticata e interna. La comunicazione RPC tra driver ed esecutori è crittografata.
- Hive
 - La comunicazione tra il metastore AWS Glue e le applicazioni EMR Serverless avviene tramite TLS.

Dovresti consentire solo connessioni crittografate su HTTPS (TLS) utilizzando [le policy IAM di aws: SecureTransport condition](#) on Amazon S3 bucket.

Identity and Access Management (IAM) in Amazon EMR Serverless

AWS Identity and Access Management (IAM) è un software Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse Serverless di Amazon EMR. IAM è un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- [Destinatari](#)
- [Autenticazione con identità](#)
- [Gestione dell'accesso tramite policy](#)
- [Come funziona EMR Serverless con IAM](#)
- [Utilizzo di ruoli collegati ai servizi per EMR Serverless](#)
- [Ruoli Job Runtime per Amazon EMR Serverless](#)
- [Esempi di policy di accesso degli utenti per EMR Serverless](#)
- [Policy per il controllo degli accessi basato su tag](#)
- [Identity-based esempi di policy per EMR Serverless](#)
- [Amazon EMR Serverless: aggiornamenti alle policy gestite AWS](#)
- [Risoluzione dei problemi relativi all'identità e all'accesso senza server di Amazon EMR](#)

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta [Risoluzione dei problemi relativi all'identità e all'accesso senza server di Amazon EMR](#))
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta [Identity and Access Management \(IAM\) in Amazon EMR Serverless](#))
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta [Esempi di policy basate sull'identità per EMR Serverless](#))

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/Facebook Per ulteriori informazioni sull'accesso, consulta [Come accedere all' Account AWS](#) nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta [AWS Signature Version 4 per le richieste API](#) nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali dell'utente root, consulta [Attività che richiedono le credenziali dell'utente root](#) nella Guida per l'utente IAM.

Identità federata

Come procedura ottimale, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare AWS IAM Identity Center. Per ulteriori informazioni, consulta [Che cos'è il Centro identità IAM?](#) nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un [utente IAM](#) è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta [Richiedere agli utenti umani di utilizzare la](#)

[federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee](#) nella Guida per l'utente IAM.

Un [gruppo IAM](#) specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta [Casi d'uso per utenti IAM](#) nella Guida per l'utente di IAM.

Ruoli IAM

Un [ruolo IAM](#) è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo [passando da un ruolo utente a un ruolo IAM \(console\)](#) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta [Metodi per assumere un ruolo](#) nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso degli utenti federati, le autorizzazioni utente IAM temporanee, l'accesso multi-account, l'accesso multi-servizio e le applicazioni in esecuzione su Amazon EC2. Per maggiori informazioni, consultare [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta [Panoramica delle policy JSON](#) nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Identity-based politiche

Identity-based le politiche sono documenti di policy sulle autorizzazioni JSON che alleggi a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy

basata su identità, consultare [Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente](#) nella Guida per l'utente IAM.

Identity-based le politiche possono essere politiche in linea (incorporate direttamente in una singola identità) o politiche gestite (politiche autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta [Scegliere tra policy gestite e policy in linea](#) nella Guida per l'utente di IAM.

Resource-based politiche

Resource-based le politiche sono documenti di policy JSON allegati a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio [specificare un'entità principale](#).

Resource-based le politiche sono politiche in linea che si trovano in quel servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta [Limiti delle autorizzazioni per le entità IAM](#) nella Guida per l'utente di IAM.
- Policy di controllo dei servizi (SCP): specifica il numero massimo di autorizzazioni per un'organizzazione o un'unità organizzativa (OU) in AWS Organizations. Per ulteriori informazioni, consultare [Policy di controllo dei servizi](#) nella Guida per l'utente di AWS Organizations .
- Policy di controllo delle risorse (RCP): imposta le autorizzazioni massime disponibili per le risorse degli account. Per ulteriori informazioni, consulta [Policy di controllo delle risorse \(RCP\)](#) nella Guida per l'utente di AWS Organizations .
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare [Policy di sessione](#) nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta [Logica di valutazione delle policy](#) nella IAM User Guide.

Come funziona EMR Serverless con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon EMR Serverless, scopri quali funzionalità IAM sono disponibili per l'uso con Amazon EMR Serverless.

Utilizzo delle funzionalità IAM con EMR Serverless

Funzionalità IAM	Supporto Serverless per Amazon EMR
Identity-based politiche	Sì
Resource-based politiche	No
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione delle policy	No
Liste di controllo degli accessi (ACL)	No
ABAC (tag nelle policy)	Sì
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	No
Ruoli Service-linked	Sì

Per avere una visione di alto livello di come EMR Serverless e AWS altri servizi funzionano con la maggior parte delle funzionalità IAM, consulta [AWS i servizi che funzionano con](#) IAM nella IAM User Guide.

Identity-based politiche per EMR Serverless

Supporta le policy basate sull'identità: sì

Identity-based le policy sono documenti di policy sulle autorizzazioni JSON che puoi allegare a un'identità, ad esempio un utente IAM, un gruppo di utenti o un ruolo. Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente](#) nella Guida per l'utente di IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta [Guida di riferimento agli elementi delle policy JSON IAM](#) nella Guida per l'utente IAM.

Esempi di policy basate sull'identità per EMR Serverless

Per accedere ad esempi di policy basate sull'identità di Amazon EMR Serverless, consulta. [Identity-based esempi di policy per EMR Serverless](#)

Resource-based policy all'interno di EMR Serverless

Supporta le policy basate su risorse: no

Resource-based le politiche sono documenti di policy JSON allegati a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio [specificare un'entità principale](#). I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente IAM.

Azioni politiche per EMR Serverless

Supporta le operazioni di policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni di eseguire l'operazione associata.

Per fare riferimento a 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 accedere ad esempi di policy basate sull'identità di Amazon EMR Serverless, consulta. [Identity-based esempi di policy per EMR Serverless](#)

Risorse relative alle policy per EMR Serverless

Supporta le risorse relative alle policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo [nome della risorsa Amazon \(ARN\)](#). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per fare riferimento a un elenco dei tipi di risorse Amazon EMR Serverless e dei relativi ARN, consulta Resources [defined by Amazon EMR Serverless](#) nel Service Authorization Reference. Per sapere quali azioni specificano l'ARN di ogni risorsa, consulta [Azioni, risorse e chiavi di condizione per Amazon EMR Serverless](#).

Per accedere ad esempi di policy basate sull'identità di Amazon EMR Serverless, consulta. [Identity-based esempi di policy per EMR Serverless](#)

Chiavi delle condizioni delle policy per EMR Serverless

Supporto delle chiavi relative alle condizioni delle politiche

Supporta le chiavi di condizione delle policy specifiche del servizio	No
---	----

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano [operatori di condizione](#), ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di [contesto delle condizioni AWS globali nella Guida](#) per l'utente IAM.

Per consultare 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 operazioni Amazon EC2 supportano le chiavi di condizione `aws:RequestedRegion` e `ec2:Region`. Per ulteriori informazioni, consulta [Esempio: limitazione dell'accesso](#) a una regione specifica.

Liste di controllo degli accessi (ACL) in EMR Serverless

Supporta le ACL: no

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

Attribute-based controllo degli accessi (ABAC) con EMR Serverless

Attribute-based supporto per il controllo degli accessi (ABAC)

Supporta ABAC (tag nelle policy) Sì

Attribute-based il controllo degli accessi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base ad attributi chiamati tag. È possibile allegare tag a entità e AWS risorse IAM, quindi progettare politiche ABAC per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'[elemento condizione](#) di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consulta [Definizione delle autorizzazioni con autorizzazione ABAC](#) nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta [Utilizzo del controllo degli accessi basato su attributi \(ABAC\)](#) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con EMR Serverless

Supporta le credenziali temporanee: sì

Le credenziali temporanee forniscono l'accesso a breve termine alle AWS risorse e vengono create automaticamente quando si utilizza la federazione o si cambia ruolo. AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta [Credenziali di sicurezza temporanee in IAM](#) e [Servizi AWS compatibili con IAM](#) nella Guida per l'utente IAM.

Cross-service autorizzazioni principali per EMR Serverless

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale chiamante an Servizio AWS, combinate con la richiesta di effettuare richieste Servizio AWS ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consulta [Forward access sessions](#).

Ruoli di servizio per EMR Serverless

Supporta i ruoli di servizio	No
------------------------------	----

Service-linked ruoli per EMR Serverless

Supporta i ruoli collegati ai servizi	Sì
---------------------------------------	----

Per informazioni dettagliate sulla creazione o la gestione di ruoli collegati ai servizi, consulta [AWS i servizi che funzionano con IAM](#). Trova un servizio nella tabella che include un servizio Yes nella colonna del Service-linked ruolo. Scegli il link Sì per accedere alla documentazione del ruolo collegato al servizio per quel servizio.

Utilizzo di ruoli collegati ai servizi per EMR Serverless

[Amazon EMR Serverless utilizza ruoli collegati ai servizi AWS Identity and Access Management \(IAM\)](#). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente a EMR Serverless. I ruoli collegati ai servizi sono predefiniti da EMR Serverless e includono tutte le autorizzazioni richieste dal servizio per chiamare altri servizi per conto dell'utente. AWS

Un ruolo collegato al servizio semplifica la configurazione di EMR Serverless perché non è necessario aggiungere manualmente le autorizzazioni necessarie. EMR Serverless definisce le autorizzazioni dei suoi ruoli collegati ai servizi e, se non diversamente definito, solo EMR Serverless può assumerne i ruoli. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato al servizio solo dopo avere eliminato le risorse correlate. Ciò protegge le risorse EMR Serverless perché non è possibile rimuovere inavvertitamente l'autorizzazione ad accedere alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, fai riferimento a [AWS Servizi che funzionano con IAM](#) e controlla i servizi che hanno Sì nella colonna Ruoli collegati ai servizi.

Scegli un Sì con un link per accedere alla documentazione del ruolo collegato al servizio per quel servizio.

Autorizzazioni di ruolo collegate ai servizi per EMR Serverless

EMR Serverless utilizza il ruolo collegato al servizio denominato `AWSServiceRoleForAmazonEMRServerless` per consentirgli di effettuare chiamate per conto dell'utente. AWS APIs

Il ruolo `AWSService RoleForAmazon EMRServerless` 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 della policy gestita cambia, pertanto la politica mostrata qui potrebbe non essere aggiornata. Visualizza la maggior parte delle up-to-date politiche di [Amazon EMRServerless ServiceRolePolicy](#) nel Console di gestione AWS.

- 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EC2PolicyStatement",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2>DeleteNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeRouteTables"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CloudWatchPolicyStatement",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": [
            "AWS/EMRServerless",
            "AWS/Usage"
          ]
        }
      }
    }
  ]
}
```

La seguente politica di fiducia è associata a questo ruolo per consentire al principale EMR Serverless di assumere questo ruolo.

JSON

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

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta le [autorizzazioni dei ruoli collegati ai servizi](#) nella Guida per l'utente IAM.

Creazione di un ruolo collegato ai servizi per EMR Serverless

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando si crea una nuova applicazione EMR Serverless (Console di gestione AWS utilizzando EMR Studio), o l' AWS CLI API AWS , EMR Serverless crea automaticamente il ruolo collegato al servizio. Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni.

Per creare il ruolo collegato al servizio utilizzando IAM AWSService RoleForAmazon EMRServerless

Aggiungi la seguente dichiarazione alla politica di autorizzazione per l'entità IAM 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 al servizio e poi devi crearlo di nuovo, utilizza la stessa procedura per ricreare il ruolo nel tuo account. Quando crei una nuova applicazione EMR Serverless, EMR Serverless crea nuovamente il ruolo collegato al servizio per te.

Puoi anche utilizzare la console IAM per creare un ruolo collegato ai servizi con lo use case EMR Serverless. Nella AWS CLI o nell' AWS API, crea un ruolo collegato al servizio con il nome del servizio. `ops.emr-serverless.amazonaws.com` Per ulteriori informazioni, consulta [Creating a service-linked role](#) nella IAM User Guide. Se elimini questo ruolo collegato al servizio, utilizza lo stesso processo per crearlo nuovamente.

Modifica di un ruolo collegato ai servizi per EMR Serverless

EMR Serverless non consente di modificare il ruolo `AWSService RoleForAmazon EMRServerless` collegato al servizio perché diverse entità potrebbero fare riferimento al ruolo. Non è possibile modificare la policy IAM AWS di proprietà utilizzata dal ruolo collegato al servizio EMR Serverless, poiché contiene tutte le autorizzazioni necessarie di EMR Serverless. È possibile tuttavia modificarne la descrizione utilizzando IAM.

Per modificare la descrizione del ruolo collegato al servizio utilizzando IAM `AWSService RoleForAmazon EMRServerless`

Aggiungi la seguente istruzione alla policy delle autorizzazioni per l'entità IAM che deve modificare la descrizione di un ruolo collegato ai servizi.

```

{
  "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 EMR Serverless

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, si consiglia di eliminare tale ruolo. In questo modo non avrai un'entità inutilizzata che non viene monitorata o gestita attivamente. Tuttavia, elimina tutte le applicazioni EMR Serverless in tutte le regioni prima di 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 AWSService RoleForAmazon EMRServerless servizio utilizzando IAM

Aggiungi la seguente dichiarazione alla politica di autorizzazione per l'entità IAM 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 ai servizi mediante IAM

Utilizza la console IAM AWS CLI, l'o l' AWS API per eliminare il ruolo collegato al AWSService RoleForAmazon EMRServerless servizio. Per ulteriori informazioni, consulta [Eliminazione di un ruolo collegato al servizio nella Guida per l'utente IAM](#).

Regioni supportate per i ruoli collegati ai servizi EMR Serverless

EMR Serverless supporta l'utilizzo di ruoli collegati ai servizi in tutte le regioni in cui il servizio è disponibile. [Per ulteriori informazioni, consulta Regioni ed endpoint.AWS](#)

Ruoli Job Runtime per Amazon EMR Serverless

È possibile specificare le autorizzazioni dei ruoli IAM 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 altre 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 Glue Data Catalog.

Per aggiungere l'accesso ad altre AWS risorse come DynamoDB, dovrai includere le relative autorizzazioni nella policy al momento della creazione del ruolo di runtime.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadAccessForEMRSamples",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ]
    }
  ]
}
```

```

    ],
    "Resource": [
      "arn:aws:s3::*.elasticmapreduce",
      "arn:aws:s3::*.elasticmapreduce/*"
    ]
  },
  {
    "Sid": "FullAccessToS3Bucket",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:ListBucket",
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws:s3::amzn-s3-demo-bucket",
      "arn:aws:s3::amzn-s3-demo-bucket/*"
    ]
  },
  {
    "Sid": "GlueCreateAndReadDataCatalog",
    "Effect": "Allow",
    "Action": [
      "glue:GetDatabase",
      "glue:CreateDatabase",
      "glue:GetDataBases",
      "glue:CreateTable",
      "glue:GetTable",
      "glue:UpdateTable",
      "glue>DeleteTable",
      "glue:GetTables",
      "glue:GetPartition",
      "glue:GetPartitions",
      "glue:CreatePartition",
      "glue:BatchCreatePartition",
      "glue:GetUserDefinedFunctions"
    ],
    "Resource": [
      "*"
    ]
  }
]

```

```
}
```

Passa i privilegi di ruolo

Puoi allegare le policy di autorizzazione IAM al ruolo di un utente per consentire all'utente di passare solo ruoli approvati. Ciò consente agli amministratori di controllare quali utenti possono passare ruoli di job runtime specifici ai job EMR Serverless. Per ulteriori informazioni sull'impostazione delle autorizzazioni, consulta [Concessione a un utente delle autorizzazioni per trasferire un ruolo a un servizio. AWS](#)

Di seguito è riportato un esempio di policy che consente di passare un ruolo di job runtime al principale del servizio EMR Serverless.

```
{
  "Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
  "Condition": {
    "StringLike": {
      "iam:PassedToService": "emr-serverless.amazonaws.com"
    }
  }
}
```

Politiche di autorizzazione gestite associate ai ruoli di runtime

Quando invii le esecuzioni di job a EMR serverless tramite la console EMR Studio, c'è un passaggio in cui scegli un ruolo Runtime da associare alla tua applicazione. A ciascuna selezione nella console sono associate politiche gestite sottostanti di cui è importante tenere presente. Le tre selezioni sono le seguenti:

1. Tutti i bucket: quando si sceglie questa opzione, viene specificata la policy [AmazonS3FullAccess](#) AWS gestita, che fornisce l'accesso completo a tutti i bucket.
2. Bucket specifici: specifica l'identificatore Amazon Resource Name (ARN) di ogni bucket scelto. Non è inclusa una politica gestita sottostante.
3. Nessuna: non sono incluse le autorizzazioni relative alle policy gestite.

Ti suggeriamo di aggiungere bucket specifici. Se scegli tutti i bucket, tieni presente che imposta l'accesso completo per tutti i bucket.

Esempi di policy di accesso degli utenti per EMR Serverless

È possibile impostare policy granulari per gli utenti in base alle azioni che si desidera che ciascun utente esegua quando interagisce con le applicazioni EMR Serverless. Le politiche seguenti sono esempi che possono aiutare a configurare le autorizzazioni appropriate per gli utenti. Questa sezione si concentra solo sulle politiche EMR Serverless. Per esempi di politiche utente di EMR Studio, fare riferimento a [Configurare le autorizzazioni utente di EMR Studio](#). Per informazioni su come allegare le policy agli utenti IAM (principali), consulta la sezione [Managing IAM policy nella IAM User Guide](#).

Politica per i Power User

Per concedere tutte le azioni richieste per EMR Serverless, crea e allega una `AmazonEMRServerlessFullAccess` policy all'utente, ruolo o gruppo IAM richiesto.

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 lavori. Rivela tutte le azioni che EMR Serverless richiede per altri servizi.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication",
        "emr-serverless:UpdateApplication",
        "emr-serverless>DeleteApplication",
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StopApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ]
    }
  ]
}
```

```

    ],
    "Resource": [
      "*"
    ]
  }
]
}

```

Quando abiliti la connettività di rete al tuo VPC, le applicazioni EMR Serverless creano interfacce di rete elastiche (ENI) di Amazon EC2 per comunicare con le risorse VPC. La seguente policy garantisce che tutti i nuovi ENI EC2 vengano creati solo nel contesto delle applicazioni EMR Serverless.

Note

Consigliamo vivamente di impostare questa policy per garantire che gli utenti non possano creare ENI EC2 se non nel contesto del lancio di applicazioni EMR Serverless.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEC2ENICreationWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
        }
      }
    }
  ]
}

```

}

Se desideri limitare l'accesso EMR Serverless a determinate sottoreti, puoi etichettare ogni sottorete con una condizione di tag. Questa policy IAM garantisce che le applicazioni EMR Serverless possano creare ENI EC2 solo all'interno di sottoreti consentite.

```
{
  "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 collegato al servizio EMR Serverless. Per ulteriori informazioni, fare riferimento a [Utilizzo di ruoli collegati ai servizi per EMR Serverless](#)

La seguente policy IAM ti consente di creare un ruolo collegato al servizio 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 policy 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

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 lavoro in base ai tag.

Gli esempi seguenti mostrano diversi scenari e modi di utilizzare gli operatori di condizione con le chiavi di condizione EMR Serverless. Queste dichiarazioni di policy IAM sono concepite unicamente per scopi dimostrativi e non devono essere utilizzate negli ambienti di produzione. Esistono vari modi di combinare dichiarazioni di policy per concedere o negare autorizzazioni in base alle tue esigenze. Per ulteriori informazioni sulla pianificazione e il test delle politiche IAM, consulta la Guida per l'[utente IAM](#).

Important

Negare esplicitamente l'autorizzazione per operazioni di assegnazione di tag è una possibilità da tenere in debita considerazione. Ciò impedisce agli utenti di concedersi personalmente

autorizzazioni tramite tag di una risorsa che non avevi intenzione di accordare. Se le operazioni di assegnazione di tag per una risorsa non vengono negate, un utente può modificare i tag e aggirare l'intenzione delle policy basate su tag. Per un esempio di policy che nega le azioni di tagging, 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 EMR Serverless.

Autorizzazione di operazioni solo su risorse con specifici valori di tag

Nel seguente esempio di policy, l'operatore di `StringEquals` condizione cerca 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.

JSON

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

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`, sostituire il blocco delle condizioni dell'esempio precedente con il seguente.

```
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/department": ["dev", "test"]
  }
}
```

Richiesta dell'assegnazione di tag alla creazione di una risorsa

Nell'esempio seguente, il tag deve essere applicato durante la creazione dell'applicazione.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-east-1"
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-east-1", "us-west-2"]
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}
```

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 è `dev`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
      ],
      "Resource": [
```

```
    "*"
  ],
  "Condition": {
    "StringNotEquals": {
      "aws:PrincipalTag/department": "dev"
    }
  },
  "Sid": "AllowEMRSERVERLESSTagresource"
}
]
}
```

Identity-based esempi di policy per EMR Serverless

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse Amazon EMR Serverless. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta [Creazione di policy IAM \(console\)](#) nella Guida per l'utente di IAM.

Per dettagli sulle azioni e sui tipi di risorse definiti da Amazon EMR Serverless, incluso il formato degli ARN per ciascun tipo di risorsa, consulta [Azioni, risorse e chiavi di condizione per Amazon EMR Serverless](#) nel Service Authorization Reference.

Argomenti

- [Best practice per le policy](#)
- [Consenti agli utenti di accedere alle proprie autorizzazioni](#)

Best practice per le policy

Note

EMR Serverless non supporta le policy gestite, quindi la prima pratica elencata di seguito non si applica.

Identity-based le policy determinano se qualcuno può creare, accedere o eliminare le risorse Serverless di Amazon EMR nel tuo account. Queste operazioni possono comportare costi aggiuntivi

per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo Account AWS. Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta [Policy gestite da AWS](#) o [Policy gestite da AWS per le funzioni dei processi](#) nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta [Policy e autorizzazioni in IAM](#) nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione [Elementi delle policy JSON di IAM: condizione](#) nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare [Convalida delle policy per il Sistema di analisi degli accessi IAM](#) nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare [Protezione dell'accesso API con MFA](#) nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta [Best practice di sicurezza in IAM](#) nella Guida per l'utente di IAM.

Consenti agli utenti di accedere alle proprie autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon EMR Serverless: aggiornamenti alle policy gestite AWS

Accedi ai dettagli sugli aggiornamenti delle politiche AWS 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 feed RSS nella pagina della cronologia dei documenti Serverless di Amazon EMR.](#)

Modifica	Descrizione	Data
AmazonEMRServerlessServiceRolePolicy — Aggiornamento a una policy esistente	Amazon EMR Serverless ha aggiunto il nuovo SidCloudWatchPolicyStatement e EC2PolicyStatement alla policy AmazonEMRServerlessServiceRolePolicy	25 gennaio 2024
AmazonEMRServerlessServiceRolePolicy — Aggiornamento a una politica esistente	Amazon EMR Serverless ha aggiunto nuove autorizzazioni per consentire ad Amazon EMR Serverless di pubblicare e parametri aggregati degli account per l'utilizzo di vCPU nello spazio dei nomi. "AWS/Usage"	20 aprile 2023
Amazon EMR Serverless ha iniziato a tracciare le modifiche	Amazon EMR Serverless ha iniziato a tracciare le modifiche per le sue AWS policy gestite.	20 aprile 2023

Risoluzione dei problemi relativi all'identità e all'accesso senza server di Amazon EMR

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 a persone esterne al mio AWS account di accedere alle mie risorse Amazon EMR Serverless](#)
- [Non riesco ad aprire il server di UI/Spark cronologia live da EMR Studio per eseguire il debug del mio lavoro o si verifica un errore API quando cerco di ottenere i log utilizzando get-dashboard-for-job-run](#)

Non sono autorizzato a eseguire un'azione in Amazon EMR Serverless

Se ti Console di gestione AWS dice che non sei autorizzato a eseguire un'azione, contatta il tuo amministratore per ricevere assistenza. L'amministratore è la persona da cui si sono ricevuti il nome utente e la password.

L'errore di esempio seguente si verifica quando l'utente `mateojackson` tenta di utilizzare la console per accedere ai dettagli di una `my-example-widget` risorsa fittizia ma non dispone delle autorizzazioni fittizie `emr-serverless:GetWidget`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-serverless:GetWidget on resource: my-example-widget
```

In questo caso, Mateo chiede al suo amministratore di aggiornare le policy per poter accedere alla risorsa `my-example-widget` mediante 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'azione `iam:PassRole`, le tue policy devono essere aggiornate per consentirti di trasferire un ruolo ad Amazon EMR Serverless.

Alcuni Servizi AWS consentono di trasferire un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un utente IAM denominato `marymajor` 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 trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse Amazon EMR 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 servizi che supportano policy basate su risorse o liste di controllo degli accessi (ACL), utilizzare tali policy per concedere alle persone l'accesso alle proprie risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon EMR Serverless supporta queste funzionalità, consulta [Identity and Access Management \(IAM\) in Amazon EMR Serverless](#)
- Per sapere come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta [Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà](#) nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta [Fornire l'accesso a soggetti Account AWS di proprietà di terze parti](#) nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta [Fornire l'accesso a utenti autenticati esternamente \(federazione delle identità\)](#) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente di IAM.

Non riesco ad aprire il server di UI/Spark cronologia live da EMR Studio per eseguire il debug del mio lavoro o si verifica un errore API quando cerco di ottenere i log utilizzando **get-dashboard-for-job-run**

Se utilizzi lo storage gestito EMR Serverless per la registrazione e la tua applicazione EMR Serverless si trova in una sottorete privata con endpoint VPC per Amazon S3 e alleggi una policy endpoint per controllare l'accesso, aggiungi le autorizzazioni menzionate in Logging for [EMR Serverless con storage gestito nella tua policy VPC all'endpoint gateway S3 per EMR Serverless](#) per archiviare e servire i registri delle applicazioni.

Propagazione attendibile delle identità

Con le versioni 7.8.0 e successive di Amazon EMR, puoi propagare le identità utente da AWS IAM Identity Center a carichi di lavoro interattivi con EMR Serverless tramite Apache Livy Endpoint. I carichi di lavoro interattivi Apache Livy propagheranno ulteriormente l'identità fornita a servizi downstream come Amazon S3, Lake Formation e Amazon Redshift, abilitando l'accesso sicuro ai dati tramite l'identità utente in questi downstream. Le sezioni seguenti forniscono una panoramica concettuale, i prerequisiti e i passaggi necessari per avviare e propagare l'identità ai carichi di lavoro interattivi con EMR Serverless tramite Apache Livy Endpoint.

Panoramica di

[IAM Identity Center](#) è l'approccio consigliato per l'autenticazione e l'autorizzazione della forza lavoro AWS per organizzazioni di qualsiasi dimensione e tipo. Con Identity Center, puoi creare e gestire identità utente o connettere la tua fonte di identità esistente, tra cui Microsoft Active Directory, Okta, Ping Identity JumpCloud, Google Workspace e Microsoft Entra ID (precedentemente Azure AD).
AWS

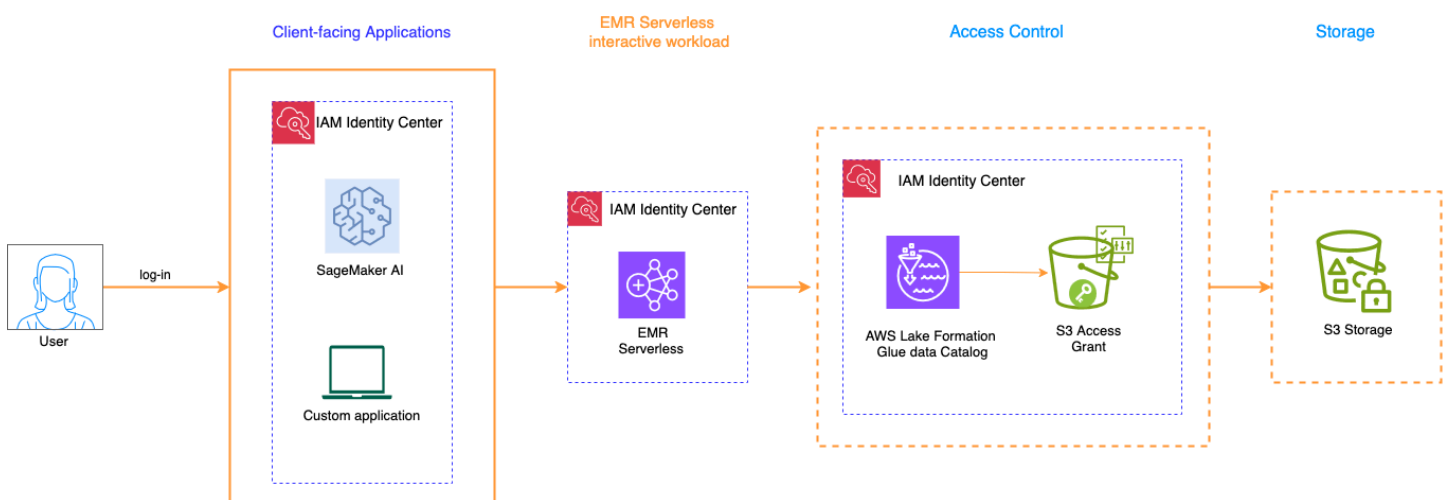
[La propagazione affidabile delle identità](#) è una funzionalità di AWS IAM Identity Center che gli amministratori dei AWS servizi connessi possono utilizzare per concedere e controllare l'accesso ai dati del servizio. L'accesso a questi dati si basa su attributi utente come le associazioni di gruppo. La configurazione di una propagazione affidabile delle identità richiede la collaborazione tra gli amministratori dei AWS servizi connessi e gli amministratori di IAM Identity Center. Per ulteriori informazioni, consulta [Prerequisiti e considerazioni](#) nella Guida per l'utente di IAM Identity Center.

Funzionalità e vantaggi

L'integrazione EMR Serverless Apache Livy Endpoint con la propagazione dell'[identità affidabile](#) di IAM Identity Center offre i seguenti vantaggi:

- La possibilità di applicare l'autorizzazione a livello di tabella con le identità di Identity Center sulle tabelle del catalogo dati AWS Glue gestite da Lake AWS Formation.
- La capacità di applicare l'autorizzazione con le identità del Centro identità sui cluster Amazon Redshift.
- Consente il monitoraggio completo delle azioni degli utenti per il controllo.
- La capacità di applicare l'autorizzazione a livello di prefisso di Amazon S3 con le identità del Centro identità sui prefissi S3 gestiti da S3 Access Grants.

Come funziona



Esempio di caso d'uso

Preparazione dei dati e ingegneria delle caratteristiche

Gli scienziati dei dati di diversi team di ricerca collaborano su progetti complessi utilizzando una piattaforma di dati unificata. Accedono all' SageMaker AI utilizzando le proprie credenziali aziendali, ottenendo immediatamente l'accesso a un vasto data lake condiviso che comprende diversi account. AWS Man mano che iniziano a progettare nuove funzionalità per nuovi modelli di apprendimento automatico, le sessioni Spark lanciate tramite EMR Serverless applicano le politiche di sicurezza a livello di colonna e riga di Lake Formation in base alle loro identità propagate. Gli scienziati possono preparare in modo efficiente i dati e progettare le funzionalità utilizzando strumenti familiari, mentre

i team addetti alla conformità hanno la certezza che ogni interazione con i dati venga tracciata e verificata automaticamente. Questo ambiente sicuro e collaborativo accelera le pipeline di ricerca mantenendo al contempo i rigorosi standard di protezione dei dati richiesti nei settori regolamentati.

Guida introduttiva a Trusted-Identity Propagation

[Questa sezione consente di configurare l'applicazione EMR-serverless con Apache Livy Endpoint per l'integrazione con AWS IAM Identity Center e abilitare la propagazione delle identità affidabili.](#)

Prerequisiti

- Un'istanza di Identity Center nella AWS regione in cui si desidera creare un endpoint EMR Serverless Apache Livy abilitato alla propagazione dell'identità affidabile. Un'istanza di Identity Center può esistere solo in una singola regione per un AWS account. Consulta [Abilita IAM Identity Center e Fornisci utenti e gruppi dalla tua fonte di identità](#) a IAM Identity Center.
- Abilita la propagazione dell'identità affidabile per servizi downstream come Lake Formation o S3 Access Grants o il cluster Amazon Redshift con cui il carico di lavoro interattivo interagisce per accedere ai dati.

Autorizzazioni per creare un'applicazione EMR Serverless abilitata alla propagazione di identità affidabili

Oltre alle [autorizzazioni di base necessarie per accedere a EMR](#) Serverless, è necessario configurare autorizzazioni aggiuntive per l'identità o il ruolo IAM utilizzati per creare un'applicazione EMR Serverless abilitata alla propagazione delle identità affidabili. Per la propagazione di identità affidabili, EMR Serverless è un'applicazione Identity Center gestita da creates/bootstraps un unico servizio nell'account che il servizio sfrutta per la convalida dell'identità e la propagazione dell'identità a valle.

```
"sso:DescribeInstance",  
"sso:CreateApplication",  
"sso>DeleteApplication",  
"sso:PutApplicationAuthenticationMethod",  
"sso:PutApplicationAssignmentConfiguration",  
"sso:PutApplicationGrant",  
"sso:PutApplicationAccessScope"
```

- `sso:DescribeInstance`— Concede l'autorizzazione a descrivere e convalidare l'IAM Identity Center InstanceARN specificato nel parametro. `identity-center-configuration`

- `sso:CreateApplication`— Concede l'autorizzazione a creare un'applicazione IAM Identity Center gestita senza server EMR che viene utilizzata per le azioni. `trusted-identity-propagation`
- `sso:DeleteApplication`— concede l'autorizzazione a pulire un'applicazione IAM Identity Center gestita da EMR Serverless
- `sso:PutApplicationAuthenticationMethod`— Concede l'autorizzazione a inserire `AuthenticationMethod` sull'applicazione IAM Identity Center gestita senza server EMR che consente al service principal di `emr-serverless` di interagire con l'applicazione IAM Identity Center.
- `sso:PutApplicationAssignmentConfiguration`— Concede l'autorizzazione a impostare l'impostazione «U» sull'applicazione IAM Identity Center. `ser-assignment-not-required`
- `sso:PutApplicationGrant`— Concede l'autorizzazione ad applicare le concessioni `token-exchange`, `IntrospectToken`, `RefreshToken` e `revokeToken` su un'applicazione IAM Identity Center.
- `sso:PutApplicationAccessScope`— Concede l'autorizzazione ad applicare l'ambito `downstream` abilitato alla propagazione dell'identità affidabile all'applicazione IAM Identity Center. Appliciamo gli ambiti «`redshift:connect`», «`lakeformation:query`» e «`s3:read_write`» per abilitare questi servizi. `trusted-identity-propagation`

Creare un'applicazione EMR Serverless abilitata alla propagazione delle identità affidabili

È necessario specificare il `-identity-center-configuration` campo con `identityCenterInstanceArn` per abilitare la propagazione dell'identità affidabile nell'applicazione. Utilizzare il comando di esempio seguente per creare un'applicazione EMR Serverless con la propagazione delle identità affidabili abilitata.

Note

È inoltre necessario specificare che la propagazione dell'identità affidabile è `--interactive-configuration '{"livyEndpointEnabled":true}'` abilitata solo per Apache Livy Endpoint.

```
aws emr-serverless create-application \
  --release-label emr-7.8.0 \
  --type "SPARK" \
  --identity-center-configuration '{"identityCenterInstanceArn" :
  "arn:aws:sso:::instance/ssoins-123456789"}' \
```

```
--interactive-configuration '{"livyEndpointEnabled":true}'
```

- `identity-center-configuration`— (opzionale) Abilita la propagazione dell'identità affidabile di Identity Center, se specificata.
- `identityCenterInstanceArn`: (obbligatorio) l'ARN dell'istanza del Centro identità.

Se non disponi delle autorizzazioni necessarie per Identity Center (menzionate in precedenza), crea prima l'applicazione EMR Serverless senza propagazione dell'identità affidabile (ad esempio, non `--identity-center-configuration` specificare il parametro) e successivamente chiedi all'amministratore dell'Identity Center di abilitare la propagazione dell'identità affidabile richiamando l'API `update-application`, vedi l'esempio seguente:

```
aws emr-serverless update-application \  
  --application-id applicationId \  
  --identity-center-configuration '{"identityCenterInstanceArn" :  
  "arn:aws:sso:::instance/ssoins-123456789"}'
```

EMR Serverless crea nel tuo account un'applicazione Identity Center gestita dal servizio che il servizio sfrutta per la convalida dell'identità e la propagazione dell'identità ai servizi downstream. L'applicazione Identity Center gestita creata da EMR Serverless è condivisa tra tutte le applicazioni `trusted-identity-propagation` EMR Serverless abilitate presenti nell'account.

Note

Non modificare manualmente le impostazioni sull'applicazione Identity Center gestita. Qualsiasi modifica potrebbe influire su tutte le applicazioni EMR Serverless `trusted-identity-propagation` abilitate nel tuo account.

Autorizzazioni Job Execution Role per propagare l'identità

Poiché EMR-Serverless sfrutta `job-execution-role` le credenziali potenziate dall'identità per propagare l'identità ai servizi downstream AWS, la policy di fiducia di Job Execution Role deve disporre di autorizzazioni aggiuntive `sts:SetContext` per migliorare le credenziali del ruolo di esecuzione del lavoro con l'identità per consentire il servizio downstream, come S3 `access-grant`, `trusted-identity-propagation` Lake Formation o Amazon Redshift. Per ulteriori informazioni su come creare [un ruolo](#), consulta [Creare un ruolo Job Runtime](#).

JSON

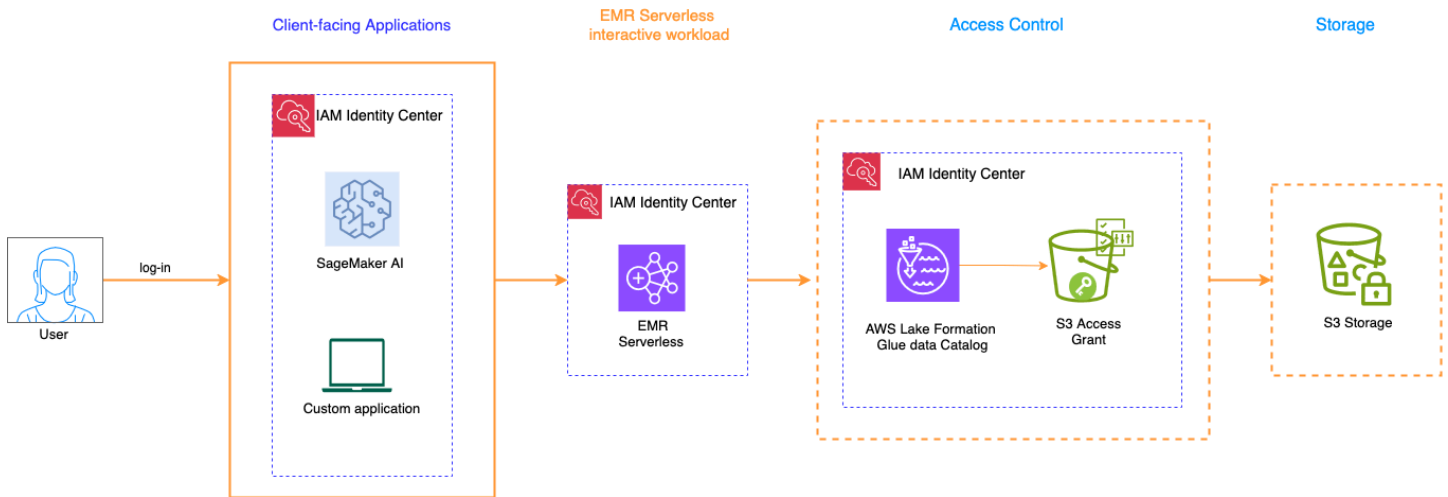
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [ "sts:AssumeRole", "sts:SetContext" ]
    }
  ]
}
```

Inoltre, JobExecutionRole necessita delle autorizzazioni per i AWS servizi downstream che job-run richiamerebbe per recuperare i dati utilizzando l'identità dell'utente. Fai riferimento ai link seguenti per configurare S3 Access Grant, Lake Formation.

- [Utilizzo di Lake Formation con EMR Serverless](#)
- [Utilizzo di Amazon S3 Access Grants con EMR Serverless](#)

Trusted Identity Propagation per carichi di lavoro interattivi

I passaggi per propagare l'identità ai carichi di lavoro interattivi tramite un endpoint Apache Livy dipendono dal fatto che gli utenti interagiscano con un ambiente di sviluppo AWS gestito, ad esempio un ambiente Notebook ospitato autonomamente come applicazione Amazon SageMaker AI rivolta al client.



AWS ambiente di sviluppo gestito

La seguente applicazione AWS gestita rivolta ai client supporta la propagazione affidabile delle identità con l'endpoint Apache Livy senza server EMR:

- [Amazon SageMaker AI](#)

Ambiente del notebook ospitato autonomamente e gestito dal cliente

Per abilitare la propagazione affidabile delle identità per gli utenti di applicazioni sviluppate su misura, consulta [Access AWS services using trusted identity propagation nel Security Blog.AWS](#)

Sessioni in background degli utenti

Le sessioni utente in background consentono ai flussi di analisi e machine learning di lunga durata di continuare anche dopo che l'utente si è disconnesso dall'interfaccia del notebook. Questa funzionalità è implementata tramite l'integrazione EMR Serverless con la funzionalità di propagazione delle identità affidabile di IAM Identity Center. Questa sezione spiega le opzioni e i comportamenti di configurazione per le sessioni utente in background.

Note

Le sessioni utente in background si applicano ai carichi di lavoro Spark avviati tramite interfacce di notebook come Amazon SageMaker Unified Studio. L'attivazione o la disabilitazione di questa funzionalità ha effetto solo sulle nuove sessioni Livy; le sessioni Livy attive esistenti non ne risentono.

Configura le sessioni utente in background

Le sessioni utente in background devono essere abilitate a due livelli per una corretta funzionalità:

1. Livello di istanza IAM Identity Center, in genere configurato dagli amministratori iDC
2. Livello di applicazione EMR Serverless: configurato dagli amministratori delle applicazioni EMR Serverless

Abilita sessioni utente in background per le applicazioni EMR Serverless

Per abilitare le sessioni utente in background per un'applicazione EMR Serverless, è necessario impostare il `userBackgroundSessionsEnabled` parametro su `true` in `identityCenterConfiguration` durante la creazione o l'aggiornamento di un'applicazione.

Prerequisiti

- Il ruolo IAM utilizzato per create/update l'applicazione EMR Serverless deve disporre dell'autorizzazione `sso:PutApplicationSessionConfiguration`. Questa autorizzazione consente a EMR Serverless di abilitare sessioni utente in background a livello di applicazione iDC gestita da EMR Serverless.
- L'applicazione EMR Serverless deve utilizzare l'etichetta di rilascio 7.8 o successiva e deve essere abilitata `Trusted-Identity Propagation`.

Per abilitare le sessioni utente in background utilizzando il AWS CLI

```
aws emr-serverless create-application \  
  --name "my-analytics-app" \  
  --type "SPARK" \  
  --release-label "emr-7.8.0" \  
  --identity-center-configuration '{"identityCenterInstanceArn":  
"arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":  
true}'
```

Per aggiornare un'applicazione esistente:

```
aws emr-serverless update-application \  
  --application-id applicationId \  
  --identity-center-configuration '{"identityCenterInstanceArn":  
"arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":  
true}'
```

```
--identity-center-configuration '{"identityCenterInstanceArn":
"arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":
true}'
```

Matrice di configurazione

L'effettiva configurazione della sessione utente in background dipende sia dall'impostazione dell'applicazione EMR Serverless che dalle impostazioni a livello di istanza di IAM Identity Center:

Matrice di configurazione della sessione utente in background

IAM Identity Center userBackgroundSession abilitato	EMR Serverless abilitato userBackgroundSessions	Comportamento
Sì	TRUE	Sessioni utente in background abilitate
Sì	FALSE	La sessione scade con il logout dell'utente
No	TRUE	L'applicazione creation/update fallisce con Exception
No	FALSE	La sessione scade con il logout dell'utente

Durata predefinita della sessione in background dell'utente

Per impostazione predefinita, tutte le sessioni utente in background hanno un limite di durata di 7 giorni in IAM Identity Center. Gli amministratori possono modificare questa durata nella console del Centro identità IAM. Questa impostazione si applica a livello di istanza IAM Identity Center e interessa tutte le applicazioni IAM Identity Center supportate all'interno di quell'istanza.

- La durata può essere impostata su qualsiasi valore da 15 minuti a 90 giorni.
- Questa impostazione è configurata nella console IAM Identity Center in Impostazioni → Autenticazione → Configura (sezione Lavori non interattivi)

Note

Le sessioni EMR Serverless Livy hanno un limite di durata massima separato di 24 ore. Le sessioni termineranno quando viene raggiunto il limite della sessione Livy o la durata della sessione in background dell'utente, a seconda dell'evento che si verifica per primo.

Impatto della disabilitazione delle sessioni utente in background

Quando le sessioni utente in background sono disabilite in IAM Identity Center:

Sessioni Livy esistenti

Continuano a funzionare senza interruzioni se sono state avviate con le sessioni utente in background abilitate. Queste sessioni continueranno a utilizzare i token di sessione in background esistenti fino a quando non termineranno naturalmente o non verranno interrotte esplicitamente.

Nuove sessioni Livy

Utilizzerà il flusso di propagazione delle identità affidabili standard e terminerà quando l'utente si disconnette o scade la sessione interattiva (ad esempio quando chiude un notebook Amazon SageMaker Unified Studio). JupyterLab

Modifica della durata delle sessioni in background degli utenti

Quando l'impostazione della durata per le sessioni utente in background viene modificata in IAM Identity Center:

Sessioni Livy esistenti

Continuate a funzionare con la stessa durata della sessione in background con cui sono state avviate.

Nuove sessioni Livy

Utilizzerà la nuova durata della sessione per le sessioni in background.

Considerazioni

Condizioni di cessazione della sessione

Quando si utilizzano sessioni utente in background, una sessione Livy continuerà a funzionare fino a quando non si verifica una delle seguenti condizioni:

- La sessione utente in background scade (in base alla configurazione iDC, fino a 90 giorni)
- La sessione in background dell'utente viene revocata manualmente da un amministratore.
- La sessione Livy raggiunge il timeout di inattività (impostazione predefinita: 1 ora dopo l'ultima istruzione eseguita)
- La sessione Livy raggiunge la sua durata massima (24 ore)
- L'utente interrompe o riavvia esplicitamente il kernel del notebook

Persistenza dei dati

Quando si utilizzano sessioni utente in background:

- Gli utenti non possono riconnettersi all'interfaccia del notebook per visualizzare i risultati dopo essersi disconnessi
- Configura le istruzioni Spark per scrivere i risultati sullo storage persistente (come Amazon S3) prima del completamento dell'esecuzione

Implicazioni sui costi

- I lavori continueranno a essere eseguiti fino al completamento anche dopo che gli utenti avranno terminato la JupyterLab sessione di Amazon SageMaker Unified Studio e verranno addebitati costi per l'intera durata dell'esecuzione completata.
- Monitora le sessioni attive in background per evitare costi inutili derivanti da sessioni dimenticate o abbandonate.

Disponibilità delle funzionalità

Le sessioni utente in background per EMR Serverless sono disponibili per:

- Solo motore Spark (il motore Hive non è supportato)

- Solo sessioni interattive Livy (i lavori in batch e i lavori in streaming non sono supportati)
- Etichette di rilascio EMR Serverless 7.8 e successive

Considerazioni sull'integrazione EMR Serverless Trusted-Identity-Propagation

Considera quanto segue quando utilizzi IAM Identity Center Trusted-Identity-Propagation con l'applicazione EMR Serverless:

- La propagazione affidabile delle identità tramite Identity Center è supportata su Amazon EMR 7.8.0 e versioni successive e solo con Apache Spark.
- Trusted Identity Propagation può essere utilizzata solo per [carichi di lavoro interattivi con EMR Serverless tramite](#) un endpoint Apache Livy. I carichi di lavoro interattivi tramite EMR Studio non supportano la propagazione delle identità affidabili
- I job in batch e i carichi di lavoro di streaming non supportano la propagazione delle identità affidabili
- I controlli di accesso granulari che utilizzano AWS Lake Formation che utilizzano Trusted Identity Propagation sono disponibili per [carichi di lavoro interattivi con EMR](#) Serverless tramite un endpoint Apache Livy.
- La propagazione affidabile delle identità con Amazon EMR è supportata nelle seguenti AWS regioni:
 - Africa (Città del Capo) – af-south-1
 - Asia Pacifico (Hong Kong) – ap-east-1
 - Asia Pacifico (Tokyo) – ap-northeast-1
 - Asia Pacifico (Seoul) – ap-northeast-2
 - Asia Pacifico (Osaka) – ap-northeast-3
 - Asia Pacifico (Mumbai) – ap-south-1
 - Asia Pacifico (Singapore) – ap-southeast-1
 - Asia Pacifico (Sydney) – ap-southeast-2
 - Asia Pacifico (Giacarta) – ap-southeast-3
 - Canada (Centrale) – ca-central-1
 - ca-west-1 — Canada (Calgary)
 - Europa (Francoforte) – eu-central-1

- Europa (Stoccolma) – eu-nord-1
- Europa (Milano) – eu-south-1
- eu-south-2 — Europa (Spagna)
- Europa (Irlanda) – eu-west-1
- Europa (Londra) – eu-west-2
- Europa (Parigi) – eu-ovest-3
- me-central-1 — Medio Oriente (EAU)
- Medio Oriente (Bahrein) – me-south-1
- Sud America (San Paolo) – sa-east-1
- Stati Uniti orientali (Virginia settentrionale) – us-est-1
- Stati Uniti orientali (Ohio) – us-est-2
- Stati Uniti occidentali (California settentrionale) – us-west-1
- Stati Uniti occidentali (Oregon) – us-west-2

Utilizzo di Lake Formation con EMR Serverless

È possibile configurare le applicazioni EMR Serverless per utilizzare Lake Formation con accesso completo alla tabella o controllo granulare degli accessi. Per i dettagli sulle funzionalità supportate in ciascuna modalità di accesso, consulta la tabella seguente.

Disponibilità delle funzionalità

Funzionalità	Disponibile da
Operazioni di lettura (SELECT, DESCRIBE) per le tabelle Hive, Iceberg	EMR 7.2+
Visualizzazioni multidialettali	EMR 7,6 +
Operazioni di lettura (SELECT, DESCRIBE) per le tabelle Delta Lake e Hudi	EMR 7,6 +
Accesso completo ai tavoli per Hive, Iceberg	EMR 7,9 +
Accesso completo ai tavoli per Delta Lake	EMR 7,11 E VERSIONI SUCCESSIVE

Funzionalità	Disponibile da
Operazioni di scrittura (DDL, DML) per tabelle Hive, Iceberg e Delta Lake	EMR 7,12+
Accesso completo ai tavoli per Hudi	EMR 7,12+

Accesso completo alla tabella di Lake Formation per EMR Serverless

Con le versioni 7.8.0 e successive di Amazon EMR, puoi sfruttare AWS Lake Formation with Glue Data Catalog, dove il ruolo di job runtime dispone di autorizzazioni complete per le tabelle senza le limitazioni del controllo granulare degli accessi. Questa funzionalità consente di leggere e scrivere su tabelle protette da Lake Formation dal batch EMR Serverless Spark e dai lavori interattivi. Consulta le seguenti sezioni per saperne di più su Lake Formation e su come utilizzarlo con EMR Serverless.

Utilizzo di Lake Formation con accesso completo ai tavoli

È possibile accedere alle tabelle del catalogo Glue Data protette da AWS Lake Formation dai job EMR Serverless Spark o dalle sessioni interattive in cui il ruolo di runtime del job ha accesso completo alla tabella. Non è necessario abilitare AWS Lake Formation sull'applicazione EMR Serverless. Quando un job Spark è configurato per Full Table Access (FTA), le credenziali di AWS Lake Formation vengono utilizzate per i dati read/write S3 per le tabelle registrate di AWS Lake Formation, mentre le credenziali del ruolo di runtime del lavoro verranno utilizzate per le read/write tabelle non registrate con Lake Formation. AWS

Important

Non abilitare AWS Lake Formation per il controllo granulare degli accessi. Un job non può eseguire contemporaneamente Full Table Access (FTA) e Fine-Grained Access Control (FGAC) sullo stesso cluster o applicazione EMR.

Passaggio 1: abilitare l'accesso completo alla tabella in Lake Formation

Per utilizzare la modalità Full Table Access (FTA), devi consentire ai motori di query di terze parti di accedere ai dati senza la convalida del tag di sessione IAM in AWS Lake Formation. Per abilitarli, seguire i passaggi descritti in [Integrazione delle applicazioni per l'accesso completo alla tabella](#).

Note

Quando si accede alle tabelle tra account diversi, è necessario abilitare l'accesso completo alla tabella sia negli account produttore che in quelli consumer. Allo stesso modo, quando si accede alle tabelle interregionali, questa impostazione deve essere abilitata sia nelle regioni produttrici che in quelle di consumo.

Fase 2: Configurazione delle autorizzazioni IAM per il ruolo di runtime dei processi

Per l'accesso in lettura o scrittura ai dati sottostanti, oltre alle autorizzazioni di Lake Formation, un ruolo di job runtime richiede l'autorizzazione `lakeformation:GetDataAccess` IAM. Con questa autorizzazione, Lake Formation concede la richiesta di credenziali temporanee per accedere ai dati.

Di seguito è riportato un esempio di politica su come fornire le autorizzazioni IAM per accedere a uno script in Amazon S3, caricare i log su S3, le autorizzazioni dell'API AWS Glue e l'autorizzazione per accedere a Lake Formation.

JSON

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

```

    ]
  },
  {
    "Sid": "GlueCatalogAccess",
    "Effect": "Allow",
    "Action": [
      "glue:Get*",
      "glue:Create*",
      "glue:Update*"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Sid": "LakeFormationAccess",
    "Effect": "Allow",
    "Action": [
      "lakeformation:GetDataAccess"
    ],
    "Resource": [
      "*"
    ]
  }
]
}

```

Passaggio 2.1 Configurare i permessi di Lake Formation

- I job Spark che leggono dati da S3 richiedono l'autorizzazione Lake Formation SELECT.
- Spark fa in modo che write/delete i dati in S3 richiedano l'autorizzazione Lake Formation ALL (SUPER).
- I lavori Spark che interagiscono con il catalogo Glue Data richiedono l'autorizzazione DESCRIBE, ALTER, DROP, a seconda dei casi.

Per ulteriori informazioni, consulta Concessione delle [autorizzazioni sulle risorse di Data Catalog](#).

Passaggio 3: inizializza una sessione Spark per l'accesso completo alla tabella utilizzando Lake Formation

Prerequisiti

AWS Glue Data Catalog deve essere configurato come metastore per accedere alle tabelle di Lake Formation.

Imposta le seguenti impostazioni per configurare il catalogo Glue come metastore:

```
--conf spark.sql.catalogImplementation=hive
--conf
  spark.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveMetastore
```

Per ulteriori informazioni sull'attivazione di Data Catalog per EMR Serverless, fare riferimento alla [configurazione Metastore](#) per EMR Serverless.

Per accedere alle tabelle registrate con AWS Lake Formation, è necessario impostare le seguenti configurazioni durante l'inizializzazione di Spark per configurare Spark per utilizzare le credenziali di Lake Formation AWS .

Hive

```
--conf
  spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationS3CredentialsResolver
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Iceberg

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=S3_DATA_LOCATION
--conf spark.sql.catalog.spark_catalog.client.region=REGION
--conf spark.sql.catalog.spark_catalog.type=glue
--conf spark.sql.catalog.spark_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.spark_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Delta Lake

```
--conf
spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialsProvider
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Hudi

```
--conf
spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialsProvider
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
--conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
--conf spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension
--conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer
```

- `spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialsProvider`: Configurare EMR Filesystem (EMRFS) o EMR S3A per utilizzare le credenziali di Lake Formation S3 per le tabelle registrate di Lake AWS Formation. Se la tabella non è registrata, utilizzare le credenziali del ruolo di runtime del processo.
- `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` e `spark.hadoop.fs.s3.folderObject.autoAction.disabled=true`: configurano EMRFS per utilizzare l'applicazione di intestazione di tipo di contenuto/directory x invece del suffisso \$folder\$ durante la creazione di cartelle S3. Questo è necessario per leggere le tabelle di Lake Formation, poiché le credenziali di Lake Formation non consentono la lettura delle cartelle delle tabelle con il suffisso \$folder\$.
- `spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true`: configura Spark per saltare la convalida del vuoto della posizione della tabella prima della creazione. Ciò è necessario per le tabelle registrate di Lake Formation, poiché le credenziali di Lake Formation per verificare la posizione vuota sono disponibili solo dopo la creazione della

tabella Glue Data Catalog. Senza questa configurazione, le credenziali del ruolo di runtime del processo convalideranno la posizione della tabella vuota.

- `spark.sql.catalog.createDirectoryAfterTable.enabled=true`: configura Spark per creare la cartella Amazon S3 dopo la creazione della tabella nel metastore Hive. Questo è necessario per le tabelle registrate di Lake Formation, poiché le credenziali di Lake Formation per creare la cartella S3 sono disponibili solo dopo la creazione della tabella Glue Data Catalog.
- `spark.sql.catalog.dropDirectoryBeforeTable.enabled=true`: Configura Spark per eliminare la cartella S3 prima dell'eliminazione della tabella nel metastore Hive. Ciò è necessario per le tabelle registrate di Lake Formation, poiché le credenziali di Lake Formation per eliminare la cartella S3 non sono disponibili dopo l'eliminazione della tabella dal Glue Data Catalog.
- `spark.sql.catalog.<catalog>.glue.lakeformation-enabled=true`: Configura il catalogo Iceberg per utilizzare le credenziali di AWS Lake Formation S3 per le tabelle registrate di Lake Formation. Se la tabella non è registrata, usare le credenziali di ambiente predefinite.

Configura la modalità di accesso completo alla tabella in Unified Studio SageMaker

Per accedere alle tabelle registrate di Lake Formation dalle sessioni interattive di Spark nei JupyterLab notebook, utilizza la modalità di autorizzazione alla compatibilità. Usa il comando `%%configure` per configurare la configurazione di Spark. Scegliere la configurazione in base al tipo di tabella:

For Hive tables

```
%%configure -f
{
  "conf": {
    "spark.hadoop.fs.s3.credentialsResolverClass":
    "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
    "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
    "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
    "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
    "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
    "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
  }
}
```

For Iceberg tables

```
%%configure -f
```

```
{
  "conf": {
    "spark.sql.catalog.spark_catalog":
"org.apache.iceberg.spark.SparkSessionCatalog",
    "spark.sql.catalog.spark_catalog.warehouse": "S3_DATA_LOCATION",
    "spark.sql.catalog.spark_catalog.client.region": "REGION",
    "spark.sql.catalog.spark_catalog.type": "glue",
    "spark.sql.catalog.spark_catalog.glue.account-id": "ACCOUNT_ID",
    "spark.sql.catalog.spark_catalog.glue.lakeformation-enabled": "true",
    "spark.sql.catalog.dropDirectoryBeforeTable.enabled": "true"
  }
}
```

For Delta Lake tables

```
%%configure -f
{
  "conf": {
    "spark.hadoop.fs.s3.credentialsResolverClass":
"com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
    "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
    "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
    "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
    "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
    "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
  }
}
```

For Hudi tables

```
%%configure -f
{
  "conf": {
    "spark.hadoop.fs.s3.credentialsResolverClass":
"com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
    "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
    "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
    "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
    "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
    "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true,
    "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
    "spark.sql.extensions":
"org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
  }
}
```

```
"spark.sql.catalog.spark_catalog":  
  "org.apache.spark.sql.hudi.catalog.HoodieCatalog",  
  "spark.serializer": "org.apache.spark.serializer.KryoSerializer"  
  }  
}
```

Sostituire i segnaposto:

- S3_DATA_LOCATION: Il percorso del tuo bucket S3
- REGION: AWS regione (ad es. us-east-1)
- ACCOUNT_ID: L'ID del tuo account AWS

Note

È necessario impostare queste configurazioni prima di eseguire qualsiasi operazione Spark sul notebook.

Operazioni supportate

Queste operazioni utilizzeranno le credenziali di AWS Lake Formation per accedere ai dati della tabella.

- CREATE TABLE
- ALTER TABLE
- INSERT INTO
- INSERT OVERWRITE
- UPDATE
- MERGE INTO
- DELETE FROM
- ANALYZE TABLE
- REPAIR TABLE
- DROP TABLE
- Query su origini dati Spark
- Scritture di origini dati Spark

Note

Le operazioni non elencate sopra continueranno a utilizzare le autorizzazioni IAM per accedere ai dati delle tabelle.

Considerazioni

- Se una tabella Hive viene creata utilizzando un lavoro per cui non è abilitato l'accesso completo alla tabella e non viene inserito alcun record, le letture o scritture successive da un lavoro con accesso completo alla tabella avranno esito negativo. Questo perché EMR Spark senza accesso completo alla tabella aggiunge il `$folder$` suffisso al nome della cartella della tabella. Per risolvere questo problema, è possibile procedere in questi modi:
 - Inserire almeno una riga nella tabella da un processo che non ha FTA abilitato.
 - Configura il lavoro che non ha FTA abilitato in modo che non utilizzi il `$folder$` suffisso nel nome della cartella in S3. Ciò si può ottenere impostando la configurazione `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` di Spark.
 - Crea una cartella S3 nella posizione della tabella `s3://path/to/table/table_name` utilizzando la console AWS S3 o la CLI AWS S3.
- L'accesso completo alla tabella è supportato con il file system EMR (EMRFS) a partire dalla versione 7.8.0 di Amazon EMR e con il file system S3A a partire dalla versione 7.10.0 di Amazon EMR.
- L'accesso completo alle tabelle è supportato per le tabelle Hive, Iceberg, Delta e Hudi.
- Considerazioni su Hudi FTA Write Support:
 - Le scritture Hudi FTA richiedono l'utilizzo `HoodieCredentialedHadoopStorage` per la vendita di credenziali durante l'esecuzione del lavoro. Imposta la seguente configurazione durante l'esecuzione dei job Hudi:
`hoodie.storage.class=org.apache.spark.sql.hudi.storage.HoodieCredentialedHadoopStorage`
 - Il supporto di scrittura Full Table Access (FTA) per Hudi è disponibile a partire dalla release 7.12 di Amazon EMR.
 - Il supporto di scrittura Hudi FTA attualmente funziona solo con le configurazioni Hudi predefinite. Le impostazioni Hudi personalizzate o non predefinite potrebbero non essere completamente supportate e potrebbero causare un comportamento imprevisto.
 - Il clustering per le tabelle Hudi Merge-On-Read (MOR) non è supportato a questo punto in modalità di scrittura FTA.

- I lavori che fanno riferimento alle tabelle con le regole Lake Formation Fine-Grained Access Control (FGAC) o Glue Data Catalog Views avranno esito negativo. Per interrogare una tabella con regole FGAC o Glue Data Catalog View, è necessario utilizzare la modalità FGAC. È possibile abilitare la modalità FGAC seguendo i passaggi descritti nella AWS documentazione: Utilizzo di [EMR Serverless with Lake AWS Formation](#) per un controllo granulare degli accessi.
- L'accesso completo alla tabella non supporta Spark Streaming.
- Quando si scrive Spark DataFrame su una tabella Lake Formation, è supportata solo la modalità APPEND per le tabelle Hive e Iceberg:
`df.write.mode("append").saveAsTable(table_name)`
- La creazione di tabelle esterne richiede le autorizzazioni IAM.
- Poiché Lake Formation memorizza temporaneamente nella cache le credenziali all'interno di un lavoro Spark, un processo batch Spark o una sessione interattiva attualmente in esecuzione potrebbero non riflettere le modifiche alle autorizzazioni.
- È necessario utilizzare un ruolo definito dall'utente e non un ruolo collegato al servizio: [requisiti per i ruoli di Lake Formation](#).

Hudi FTA Write Support - Operazioni supportate

La tabella seguente mostra le operazioni di scrittura supportate per le tabelle Hudi Copy-On-Write (COW) e Merge-On-Read (MOR) in modalità Full Table Access:

Operazioni di scrittura supportate da Hudi FTA

Tipo tabella	Operation	Comando di scrittura SQL	Stato
MUCCA	INSERT	INSERT INTO TABLE	Supportata
MUCCA	INSERT	INSERISCI NELLA TABELLA - PARTIZIONE (statica, dinamica)	Supportata

Tipo tabella	Operation	Comando di scrittura SQL	Stato
MUCCA	INSERT	INSERT OVERWRITE	Supportata
MUCCA	INSERT	INSERT OVERWRITE - PARTIZION E (statica, dinamica)	Supportata
UPDATE	UPDATE	UPDATE TABLE	Supportata
MUCCA	UPDATE	TABELLA DI AGGIORNAM ENTO - Cambia partizione	Non supportato
DELETE	DELETE	DELETE FROM TABLE	Supportata
ALTER	ALTER	MODIFICA TABELLA: RINOMINA IN	Non supportato
MUCCA	ALTER	MODIFICA TABELLA - IMPOSTA TBLPROPER TIES	Supportata
MUCCA	ALTER	ALTER TABLE - ANNULLA TBLPROPER TIES	Supportata

Tipo tabella	Operation	Comando di scrittura SQL	Stato
MUCCA	ALTER	ALTERA TABELLA - MODIFICA COLONNA	Supportata
MUCCA	ALTER	ALTERA TABELLA - AGGIUNGI COLONNE	Supportata
MUCCA	ALTER	ALTER TABLE - AGGIUNGI UNA PARTIZIONE	Supportata
MUCCA	ALTER	ALTERA TABELLA - ELIMINA LA PARTIZIONE	Supportata
MUCCA	ALTER	ALTER TABLE - RECUPERA LE PARTIZIONI	Supportata
MUCCA	ALTER	RIPARA LE PARTIZIONI DI SINCRONIZZAZIONE DELLE TABELLE	Supportata
DROP	DROP	DROP TABLE	Supportata
MUCCA	DROP	DROP TABLE - PURGE	Supportata

Tipo tabella	Operation	Comando di scrittura SQL	Stato
CREATE	CREATE	CREATE TABLE - Gestito	Supportata
MUCCA	CREATE	CREA TABELLA - PARTIZIONA PER	Supportata
MUCCA	CREATE	CREA UNA TABELLA SE NON ESISTE	Supportata
MUCCA	CREATE	CREATE TABLE LIKE	Supportata
MUCCA	CREATE	CREATE TABLE AS SELECT	Supportata
CREATE	CREATE	CREA TABELLA con LOCATION - Tabella esterna	Non supportato
DATAFRAME (INSERIRE)	DATAFRAME (INSERIRE)	saveAsTab le.Sovrascrivi	Supportata
MUCCA	DATAFRAME (INSERISCI)	saveAsTab le.Aggiungi	Non supportato
MUCCA	DATAFRAME (INSERISCI)	saveAsTab le.Ignora	Supportata
MUCCA	DATAFRAME (INSERISCI)	saveAsTab le.ErrorIfExists	Supportata
MUCCA	DATAFRAME (INSERISCI)	saveAsTable - Tabella esterna (Path)	Non supportato

Tipo tabella	Operation	Comando di scrittura SQL	Stato
MUCCA	DATAFRAME (INSERISCI)	salva (percorso) - DF v1	Non supportato
ALTRO	INSERT	INSERT INTO TABLE	Supportata
MOR	INSERT	INSERISCI NELLA TABELLA - PARTIZION E (statica, dinamica)	Supportata
ALTRO	INSERT	INSERT OVERWRITE	Supportata
MOR	INSERT	INSERT OVERWRITE - PARTIZION E (statica, dinamica)	Supportata
UPDATE	UPDATE	UPDATE TABLE	Supportata
ALTRO	UPDATE	TABELLA DI AGGIORNAM ENTO - Cambia partizione	Non supportato
DELETE	DELETE	DELETE FROM TABLE	Supportata
ALTER	ALTER	MODIFICA TABELLA: RINOMINA IN	Non supportato

Tipo tabella	Operation	Comando di scrittura SQL	Stato
- ALTRO	ALTER	MODIFICA TABELLA - IMPOSTA TBLPROPER TIES	Supportata
ALTRO	ALTER	MODIFICA TABELLA - ANNULLA TBLPROPER TIES	Supportata
ALTRO	ALTER	MODIFICA TABELLA - MODIFICA COLONNA	Supportata
ALTRO	ALTER	ALTERA TABELLA - AGGIUNGI COLONNE	Supportata
- ALTRO	ALTER	ALTER TABLE - AGGIUNGI UNA PARTIZIONE	Supportata
- ALTRO	ALTER	ALTERA TABELLA - ELIMINA LA PARTIZIONE	Supportata
- ALTRO	ALTER	ALTER TABLE - RECUPERA LE PARTIZIONI	Supportata

Tipo tabella	Operation	Comando di scrittura SQL	Stato
- ALTRO	ALTER	RIPARA LE PARTIZIONI DI SINCRONIZZAZIONE DELLE TABELLE	Supportata
DROP	DROP	DROP TABLE	Supportata
ALTRO	DROP	DROP TABLE - PURGE	Supportata
CREATE	CREATE	CREATE TABLE - Gestito	Supportata
ALTRO	CREATE	CREA TABELLA - PARTIZIONA PER	Supportata
- ALTRO	CREATE	CREA UNA TABELLA SE NON ESISTE	Supportata
ALTRO	CREATE	CREATE TABLE LIKE	Supportata
MOR	CREATE	CREATE TABLE AS SELECT	Supportata
CREATE	CREATE	CREA TABELLA con LOCATION - Tabella esterna	Non supportato
DATAFRAME (UPSERT)	DATAFRAME (UPSERT)	saveAsTable.Sovrascrivi	Supportata

Tipo tabella	Operation	Comando di scrittura SQL	Stato
ALTRO	DATAFRAME (SCONVOLTO)	saveAsTable.Aggiungi	Non supportato
ALTRO	DATAFRAME (SCONVOLTO)	saveAsTable.Ignore	Supportata
DI PIÙ	DATAFRAME (SCONVOLTO)	saveAsTable.ErrorIfExists	Supportata
PIÙ	DATAFRAME (SCONVOLTO)	saveAsTable - Tabella esterna (Path)	Non supportato
- ALTRO	DATAFRAME (SCONVOLTO)	salva (percorso) - DF v1	Non supportato
DATAFRAME (ELIMINA)	DATAFRAME (ELIMINA)	saveAsTable.Aggiungi	Non supportato
ALTRO	DATAFRAME (ELIMINA)	saveAsTable - Tabella esterna (Path)	Non supportato
- ALTRO	DATAFRAME (ELIMINA)	salva (percorso) - DF v1	Non supportato
DATAFRAME (BULK_INSERT)	DATAFRAME (BULK_INSERT)	saveAsTable.Sovrascrivi	Supportata
ALTRO	DATAFRAME (BULK_INSERT)	saveAsTable.Aggiungi	Non supportato
ALTRO	DATAFRAME (BULK_INSERT)	saveAsTable.Ignore	Supportata

Tipo tabella	Operation	Comando di scrittura SQL	Stato
DI PIÙ	DATAFRAME (BULK_INSERT)	saveAsTable. ErrorIfExists	Supportata
ALTRO	DATAFRAME (BULK_INSERT)	saveAsTable - Tabella esterna (Path)	Non supportato
- ALTRO	DATAFRAME (BULK_INSERT)	salva (percorso) - DF v1	Non supportato

Utilizzo di EMR Serverless per un controllo granulare degli AWS Lake Formation accessi

Panoramica di

Con le versioni 7.2.0 e successive di Amazon EMR, sfrutta AWS Lake Formation per applicare controlli di accesso granulari 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 le query all'interno dei job Amazon EMR Serverless Spark. Per configurare un controllo granulare degli accessi per i processi batch e le sessioni interattive di Apache Spark, usa EMR Studio. Consulta le seguenti sezioni per saperne di più su Lake Formation e su come utilizzarlo con EMR Serverless.

L'utilizzo di Amazon EMR Serverless AWS Lake Formation comporta costi aggiuntivi. Per ulteriori informazioni, consulta i prezzi di [Amazon EMR](#).

Come funziona EMR Serverless con AWS Lake Formation

L'utilizzo di EMR Serverless con Lake Formation ti consente di applicare un livello di autorizzazioni su ogni lavoro Spark per applicare il controllo delle autorizzazioni di Lake Formation quando EMR Serverless esegue i lavori. EMR Serverless 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 policy di Lake Formation. Per ulteriori informazioni, consulta [Cos'è AWS Lake Formation](#) e [Considerazioni](#) e limitazioni.

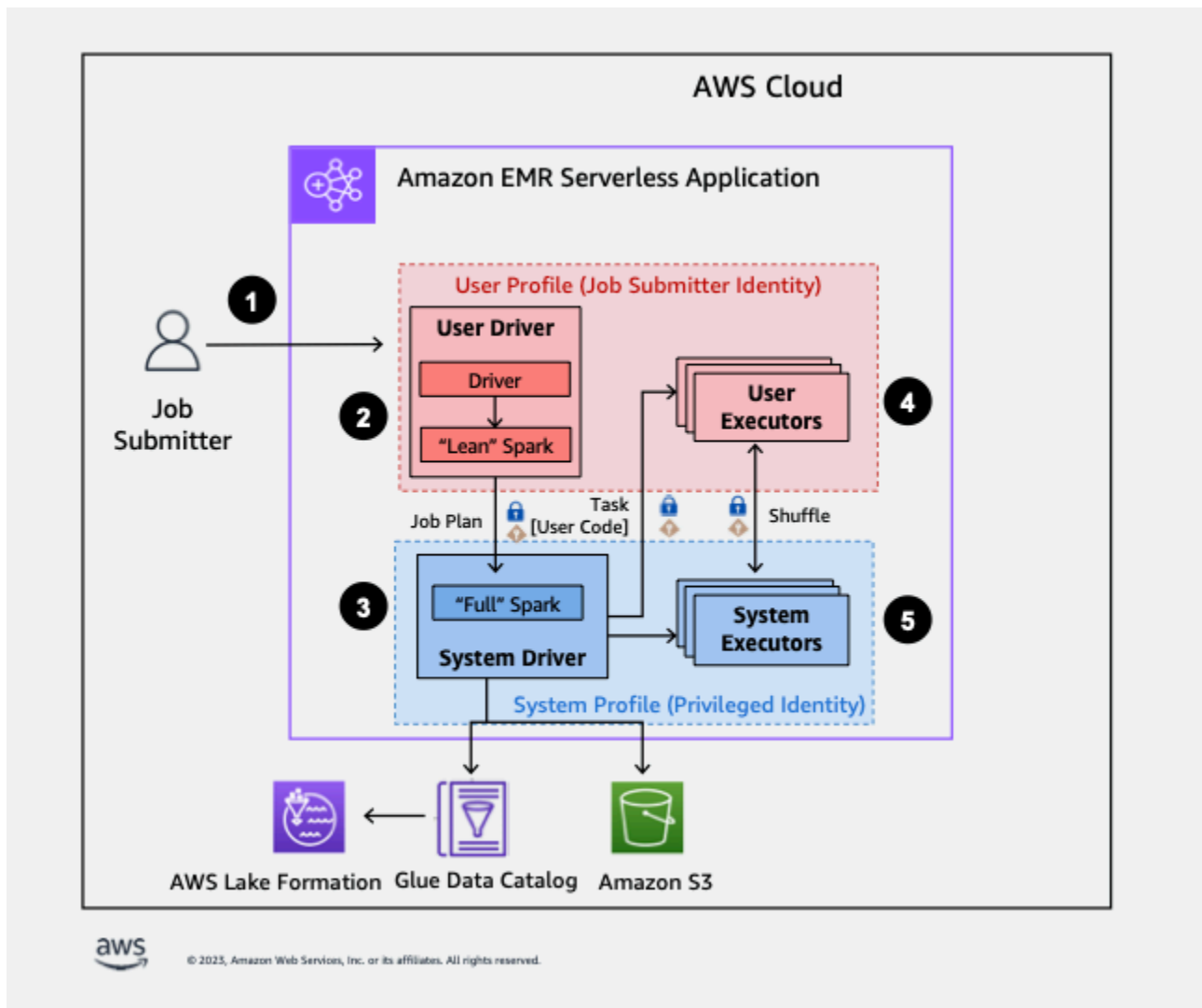
Quando utilizzi la capacità preinizializzata con Lake Formation, ti suggeriamo di avere almeno due driver Spark. Ogni job abilitato per Lake Formation utilizza due driver Spark, uno per il profilo utente e uno per il profilo di sistema. Per prestazioni ottimali, utilizza il doppio del numero di driver per i lavori abilitati a Lake Formation rispetto a chi non utilizza Lake Formation.

Quando esegui i job Spark su EMR Serverless, considera anche 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 agli esecutori utente e 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 sovrascrivere questa configurazione quando si specifica `spark.dynamicAllocation.maxExecutorsRatio` un valore compreso tra 0 e 1. Inoltre, configurate anche 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 ottiene l'accesso ai dati protetti dalle politiche di sicurezza di Lake Formation.



1. Un utente invia un job Spark a un'applicazione EMR AWS Lake Formation Serverless abilitata.
2. EMR Serverless invia il lavoro a un driver utente ed esegue il lavoro 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 catalogo di Glue. Costruisce un piano per il processo.
3. EMR Serverless imposta un secondo driver chiamato driver di sistema e lo esegue nel profilo di sistema (con un'identità privilegiata). EMR Serverless configura un canale TLS crittografato tra i due driver per la comunicazione. Il driver utente utilizza il canale per inviare i piani per il processo al driver di sistema. Il driver di sistema non esegue il codice inviato dall'utente. Esegue Spark completo e comunica con S3 e il catalogo dati per l'accesso ai dati. Richiede esecutori e compila il piano per il processo in una sequenza di fasi di esecuzione.
4. EMR Serverless 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 executor dei profili utente.

5. Le fasi che leggono i dati dalle tabelle del Data Catalog protette da AWS Lake Formation o che applicano filtri di sicurezza vengono delegate agli esecutori di sistema.

Abilitazione di Lake Formation in Amazon EMR

Per abilitare Lake Formation, imposta `spark.emr-serverless.lakeformation.enabled` a `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-7.13.0 \  
  --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
```

Autorizzazioni IAM per il ruolo di runtime del processo

Le autorizzazioni di Lake Formation controllano l'accesso alle risorse di AWS Glue Data Catalog, alle sedi Amazon S3 e ai dati sottostanti in tali sedi. Le autorizzazioni IAM controllano l'accesso a Lake Formation and AWS Glue APIs e alle risorse. Anche se disponi dell'autorizzazione di Lake Formation per accedere a una tabella nel Data Catalog (SELECT), l'operazione fallisce se non disponi dell'autorizzazione IAM per il funzionamento dell'API `glue:Get*`.

Di seguito è riportato un esempio di policy su come fornire le autorizzazioni IAM per accedere a uno script in S3, caricare i log su S3, le autorizzazioni dell'API AWS Glue e l'autorizzazione per accedere a Lake Formation.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3::*.amzn-s3-demo-bucket/scripts",
        "arn:aws:s3::*.amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

```
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Configurazione delle autorizzazioni di Lake Formation per il ruolo di runtime del processo

Innanzitutto, registrare la posizione della tabella Hive con Lake Formation. Poi, creare le autorizzazioni per il ruolo di runtime del processo nella tabella desiderata. Per maggiori dettagli su Lake Formation, consulta [What is AWS Lake Formation?](#) nella Guida per gli AWS Lake Formation sviluppatori.

Dopo aver configurato le autorizzazioni di Lake Formation, invia i lavori Spark su Amazon EMR Serverless. [Per ulteriori informazioni sui job Spark, consulta gli esempi di Spark.](#)

Invio di un'esecuzione di processo

Dopo aver completato la configurazione delle sovvenzioni Lake Formation, puoi [inviare lavori Spark su EMR](#) Serverless. La sezione che segue mostra esempi di come configurare e inviare le proprietà del job run.

Requisiti per l'autorizzazione

Table non registrate in AWS Lake Formation

Per le tabelle non registrate con AWS Lake Formation, il ruolo di job runtime accede sia al AWS Glue Data Catalog che ai dati della tabella sottostante in Amazon S3. Ciò richiede che il ruolo di job runtime disponga delle autorizzazioni IAM appropriate per le operazioni di AWS Glue e Amazon S3.

Tabelle registrate in AWS Lake Formation

Per le tabelle registrate con AWS Lake Formation, il ruolo di job runtime accede ai metadati di AWS Glue Data Catalog, mentre le credenziali temporanee fornite da Lake Formation accedono ai dati della tabella sottostante in Amazon S3. Le autorizzazioni Lake Formation necessarie per eseguire un'operazione dipendono dalle chiamate API di AWS Glue Data Catalog e Amazon S3 avviate dal job Spark e possono essere riassunte come segue:

- L'autorizzazione DESCRIBE consente al ruolo di runtime di leggere i metadati di tabelle o database nel Data Catalog
- L'autorizzazione ALTER consente al ruolo di runtime di modificare i metadati di tabelle o database nel Data Catalog
- L'autorizzazione DROP consente al ruolo di runtime di eliminare i metadati di tabelle o database dal Data Catalog
- L'autorizzazione SELECT consente al ruolo di runtime di leggere i dati della tabella da Amazon S3
- L'autorizzazione INSERT consente al ruolo di runtime di scrivere dati di tabella su Amazon S3
- L'autorizzazione DELETE consente al ruolo di runtime di eliminare i dati della tabella da Amazon S3

Note

Lake Formation valuta le autorizzazioni senza fretta quando un job Spark chiama AWS Glue per recuperare i metadati della tabella e Amazon S3 per recuperare i dati della tabella. I lavori che utilizzano un ruolo di runtime con autorizzazioni insufficienti non falliranno finché Spark non effettuerà una chiamata AWS Glue o Amazon S3 che richiede l'autorizzazione mancante.

Note

Nella seguente matrice di tabelle supportata:

- Le operazioni contrassegnate come Supported utilizzano esclusivamente le credenziali di Lake Formation per accedere ai dati delle tabelle per le tabelle registrate con Lake Formation. Se le autorizzazioni di Lake Formation sono insufficienti, l'operazione non

ricorrerà alle credenziali del ruolo di runtime. Per le tabelle non registrate con Lake Formation, le credenziali del ruolo di job runtime accedono ai dati della tabella.

- Le operazioni contrassegnate come Supportate con autorizzazioni IAM sulla posizione Amazon S3 non utilizzano le credenziali di Lake Formation per accedere ai dati delle tabelle sottostanti in Amazon S3. Per eseguire queste operazioni, il ruolo di job runtime deve disporre delle autorizzazioni IAM di Amazon S3 necessarie per accedere ai dati della tabella, indipendentemente dal fatto che la tabella sia registrata con Lake Formation.

Hive

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
SELECT	SELECT	Supportata
CREATE TABLE	CREATE_TABLE	Supportata
CREATE TABLE LIKE	CREA_TABELLA	Supportato con autorizzazioni IAM sulla posizione Amazon S3
CREATE TABLE AS SELECT	CREATE_TABLE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
DESCRIBE TABLE	DESCRIBE	Supportata
SHOW TBLPROPERTIES	DESCRIBE	Supportata
SHOW COLUMNS	DESCRIBE	Supportata
SHOW PARTITIONS	DESCRIBE	Supportata
SHOW CREATE TABLE	DESCRIBE	Supportata
MODIFICA TABELLA tablename	SELEZIONA e ALTER	Supportata

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
MODIFICA LA POSIZIONE DEL tablename SET DI TABELLE	-	Non supportata
ALTER TABLE tablename AGGIUNGI PARTIZIONE	SELEZIONA, INSERISCI e MODIFICA	Supportata
REPAIR TABLE	SELEZIONA e ALTER	Supportata
CARICARE DATI		Non supportata
INSERT	INSERT e ALTER	Supportata
INSERT OVERWRITE	SELEZIONA, INSERISCI , ELIMINA e ALTER	Supportata
DROP TABLE	SELEZIONA, RILASCIA, ELIMINA e MODIFICA	Supportata
TRUNCATE TABLE	SELEZIONA, INSERISCI , ELIMINA e MODIFICA	Supportata
Dataframe Writer V1	Uguale all'operazione SQL corrispondente	Supportata per l'aggiunta di dati a una tabella esistente. Per ulteriori informazioni, consulta le considerazioni e le limitazioni
Dataframe Writer V2	Uguale all'operazione SQL corrispondente	Supportata per l'aggiunta di dati a una tabella esistente. Per ulteriori informazioni, consulta le considerazioni e le limitazioni

Iceberg

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
SELECT	SELECT	Supportata
CREATE TABLE	CREATE_TABLE	Supportata
CREATE TABLE LIKE	CREA_TABELLA	Supportato con autorizzazioni IAM sulla posizione Amazon S3
CREATE TABLE AS SELECT	CREATE_TABLE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
SOSTITUISCI LA TABELLA CON SELECT	SELEZIONA, INSERISCI e ALTER	Supportata
DESCRIBE TABLE	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
SHOW TBLPROPERTIES	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
SHOW CREATE TABLE	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
ALTER TABLE	SELEZIONA, INSERISCI e MODIFICA	Supportata
ALTER TABLE SET LOCATION	SELEZIONA, INSERISCI e ALTER	Supportato con autorizzazioni IAM sulla posizione Amazon S3
MODIFICA LA SCRITTURA DELLA TABELLA ORDINATA PER	SELEZIONA, INSERISCI e ALTER	Supportato con autorizzazioni IAM sulla posizione Amazon S3
ALTER TABLE WRITE DISTRIBUTED BY	SELECT, INSERT e ALTER	Supportato con autorizzazioni IAM sulla posizione Amazon S3

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
MODIFICARE LA TABELLA, RINOMINAR E LA TABELLA	CREATE_TABLE e DROP	Supportata
INSERT INTO	SELECT, INSERT e ALTER	Supportata
INSERT OVERWRITE	SELEZIONA, INSERISCI e ALTER	Supportata
DELETE	SELEZIONA, INSERISCI e ALTER	Supportata
UPDATE	SELEZIONA, INSERISCI e ALTER	Supportata
MERGE INTO	SELEZIONA, INSERISCI e ALTER	Supportata
DROP TABLE	SELEZIONA, ELIMINA e RILASCIA	Supportata
DataFrame Writer V1	-	Non supportata
DataFrame Scrittore V2	Uguale all'operazione SQL corrispondente	Supportata per l'aggiunta di dati a una tabella esistente. Per ulteriori informazioni, consulta le considerazioni e le limitazioni .
Tabelle dei metadati	SELECT	Supportato. Alcune tabelle sono nascoste. Per ulteriori informazioni, consulta le considerazioni e le limitazioni .

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
Stored procedure	-	<p>Supportato per le tabelle che soddisfano le seguenti condizioni:</p> <ul style="list-style-type: none"> • Tabelle non registrate in AWS Lake Formation • Tabelle che non utilizzano <code>register_table</code> e <code>migrate</code> <p>Per ulteriori informazioni, fare riferimento a considerazioni e limitazioni.</p>

Configurazione Spark per Iceberg: L'esempio seguente mostra come configurare Spark con Iceberg. Per eseguire i job Iceberg, fornisci 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
```

Hudi

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
SELECT	SELECT	Supportata
CREATE TABLE	CREATE_TABLE	Supportato con autorizzazioni IAM sulla posizione Amazon S3

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
CREATE TABLE LIKE	CREATE_TABLE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
CREATE TABLE AS SELECT	-	Non supportata
DESCRIBE TABLE	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
SHOW TBLPROPERTIES	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
SHOW COLUMNS	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
SHOW CREATE TABLE	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
ALTER TABLE	SELECT	Supportato con autorizzazioni IAM sulla posizione Amazon S3
INSERT INTO	SELEZIONA e MODIFICA	Supportato con autorizzazioni IAM sulla posizione Amazon S3
INSERT OVERWRITE	SELEZIONA e MODIFICA	Supportato con autorizzazioni IAM sulla posizione Amazon S3
DELETE	-	Non supportata

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
UPDATE	-	Non supportata
MERGE INTO	-	Non supportata
DROP TABLE	SELEZIONA e RILASCIA	Supportato con autorizzazioni IAM sulla posizione Amazon S3
DataFrame Writer V1	-	Non supportata
DataFrame Scrittore V2	Uguale all'operazione SQL corrispondente	Supportato con autorizzazioni IAM sulla posizione Amazon S3
Tabelle dei metadati	-	Non supportata
Manutenzione delle tabelle e funzionalità di utilità	-	Non supportata

I seguenti esempi configurano Spark con Hudi, specificando le posizioni dei file e altre proprietà necessarie per l'uso.

Configurazione Spark per Hudi: questo frammento, se usato in un notebook, specifica il percorso del file JAR del pacchetto Hudi Spark, che abilita la funzionalità Hudi in Spark. Inoltre configura Spark per utilizzare il AWS Glue Data Catalog come metastore.

```
%%configure -f
{
  "conf": {
    "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
    "spark.hadoop.hive.metastore.client.factory.class":
"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
    "spark.serializer": "org.apache.spark.serializer.JavaSerializer",
    "spark.sql.catalog.spark_catalog":
"org.apache.spark.sql.hudi.catalog.HoodieCatalog",
    "spark.sql.extensions":
"org.apache.spark.sql.hudi.HoodieSparkSessionExtension"
```

```
}
}
```

Configurazione Spark per Hudi with AWS Glue: questo frammento, se utilizzato in un notebook, abilita Hudi come formato data-lake supportato e garantisce la disponibilità delle librerie e delle dipendenze Hudi.

```
%%configure
{
  "--conf": "spark.serializer=org.apache.spark.serializer.JavaSerializer --conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog --
conf
spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
  "--datalake-formats": "hudi",
  "--enable-glue-datacatalog": True,
  "--enable-lakeformation-fine-grained-access": "true"
}
```

Delta Lake

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
SELECT	SELECT	Supportata
CREATE TABLE	CREATE_TABLE	Supportata
CREATE TABLE LIKE	-	Non supportata
CREATE TABLE AS SELECT	CREA_TABELLA	Supportata
SOSTITUISCI LA TABELLA COME SELECT	SELEZIONA, INSERISCI e ALTER	Supportata
DESCRIBE TABLE	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
SHOW TBLPROPERTIES	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
SHOW COLUMNS	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
SHOW CREATE TABLE	DESCRIBE	Supportato con autorizzazioni IAM sulla posizione Amazon S3
ALTER TABLE	SELEZIONA e INSERISCI	Supportata
ALTER TABLE SET LOCATION	SELEZIONA e INSERISCI	Supportato con autorizzazioni IAM sulla posizione Amazon S3
MODIFICA TABLE CLUSTER tablename DI	SELEZIONA e INSERISCI	Supportato con autorizzazioni IAM sulla posizione Amazon S3
ALTER TABLE AGGIUNGI VINCOLO tablename	SELEZIONA e INSERISCI	Supportato con autorizzazioni IAM sulla posizione Amazon S3
MODIFICA IL VINCOLO tablename TABLE DROP	SELEZIONA e INSERISCI	Supportato con autorizzazioni IAM sulla posizione Amazon S3
INSERT INTO	SELEZIONA e INSERISCI	Supportata
INSERT OVERWRITE	SELEZIONA e INSERISCI	Supportata
DELETE	SELEZIONA e INSERISCI	Supportata

Operation	AWS Lake Formation autorizzazioni	Stato del supporto
UPDATE	SELEZIONA e INSERISCI	Supportata
MERGE INTO	SELEZIONA e INSERISCI	Supportata
DROP TABLE	SELEZIONA, ELIMINA e RILASCIA	Supportata
DataFrame Writer V1	-	Non supportata
DataFrame Scrittore V2	Uguale all'operazione SQL corrispondente	Supportata
Manutenzione delle tabelle e funzionalità di utilità	-	Non supportata

EMR Serverless con Delta Lake: per utilizzare Delta Lake with Lake Formation su EMR Serverless, esegui il seguente comando:

```
spark-sql \  
  --conf  
  spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension,com.amazonaws.emr.recordserverless  
 \  
  --conf  
  spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog \  
 \  
  --conf  
  spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension,com.amazonaws.emr.recordserverless
```

Debug dei processi

Note

Con questa funzionalità, accedi ai log stdout e stderr per i worker dei profili di sistema che possono contenere informazioni sensibili e non filtrate. La seguente autorizzazione deve essere utilizzata solo per accedere ai dati non di produzione. Per le applicazioni create per essere utilizzate con lavori di produzione, consigliamo vivamente di aggiungere queste autorizzazioni solo agli amministratori o agli utenti con accesso elevato ai dati.

Con EMR-7.3.0 e versioni successive, EMR Serverless abilita la funzionalità di autodebug per i processi batch abilitati per Lake Formation. A tale scopo, utilizzate il nuovo parametro Logs nell'API. accessSystemProfile [GetDashboardForJobRun](#). Se accessSystemProfileLogs è impostato su true, è possibile accedere ai log stdout e stderr per i worker del profilo di sistema, che possono essere utilizzati per il debug di un processo batch EMR Serverless abilitato per Lake Formation.

```
aws emr-serverless get-dashboard-for-job-run \
  --application-id application-id
  --job-run-id job-run-id
  --access-system-profile-logs
```

Autorizzazioni richieste

Il principale che desidera eseguire il debug dei processi batch abilitati per Lake Formation GetDashboardForJobRun deve disporre delle seguenti autorizzazioni aggiuntive:

```
{
  "Sid": "AccessSystemProfileLogs",
  "Effect": "Allow",
  "Action": [
    "emr-serverless:GetDashboardForJobRun",
    "emr-serverless:AccessSystemProfileLogs",
    "glue:GetDatabases",
    "glue:SearchTables"
  ],
  "Resource": [
    "arn:aws:emr-serverless:region:account-id:/applications/applicationId/jobruns/jobid",
    "arn:aws:glue:region:account-id:catalog",
    "arn:aws:glue:region:account-id:database/*",
    "arn:aws:glue:region:account-id:table/*/*"
  ]
}
```

Considerazioni

I log dei profili di sistema per il debug sono visibili per i lavori che accedono a database o tabelle in Lake Formation all'interno dello stesso account del lavoro. Non sono visibili nei seguenti scenari:

- Se il catalogo dati gestito utilizzando le autorizzazioni di Lake Formation ha database e tabelle tra account

- Se il catalogo dati gestito utilizzando le autorizzazioni di Lake Formation contiene collegamenti alle risorse

Utilizzo delle viste del Glue Data Catalog

È possibile creare e gestire viste nel AWS Glue Data Catalog da utilizzare con EMR Serverless. Queste sono note comunemente come viste del AWS Glue Data Catalog. Queste viste sono utili perché supportano più motori di query SQL, quindi puoi accedere alla stessa vista su AWS servizi diversi, come EMR Serverless e Amazon Amazon Athena Redshift.

Creando una vista nel Data Catalog, utilizza le concessioni di risorse e i controlli di accesso basati su tag per concedere l'accesso AWS Lake Formation ad essa. Utilizzando questo metodo di controllo degli accessi, non è necessario configurare un accesso aggiuntivo alle tabelle a cui hai fatto riferimento durante la creazione della vista. Questo metodo di concessione delle autorizzazioni è chiamato *semantica*, mentre queste viste sono chiamate *viste del definitore*. Per ulteriori informazioni sul controllo degli accessi in Lake Formation, consulta [Concessione e revoca delle autorizzazioni sulle risorse del Data Catalog](#) nella AWS Lake Formation Developer Guide.

Le viste del Catalogo dati sono utili per i seguenti casi d'uso:

- **Controllo granulare degli accessi:** è possibile creare una vista che limiti l'accesso ai dati in base alle autorizzazioni necessarie per l'utente. Ad esempio, puoi utilizzare le viste nel Catalogo dati per impedire ai dipendenti che non lavorano nel reparto delle risorse umane di visualizzare le informazioni di identificazione personale (PII).
- **Definizione completa della vista:** applicando filtri alla visualizzazione nel Data Catalog, ti assicuri che i record di dati disponibili in una visualizzazione del Data Catalog siano sempre completi.
- **Sicurezza avanzata:** la definizione della query utilizzata per creare la vista deve essere completa. Questo vantaggio significa che le visualizzazioni del Data Catalog sono meno suscettibili ai comandi SQL di attori malintenzionati.
- **Condivisione semplice dei dati:** condividi i dati con altri AWS account senza spostare i dati. Per ulteriori informazioni, consulta [Condivisione dei dati tra account in Lake Formation](#).

Creazione di una vista di Catalogo Dati

Esistono diversi modi per creare una visualizzazione del catalogo dati. Questi includono l'utilizzo di AWS CLI o Spark SQL. Seguono alcuni esempi.

Using SQL

Di seguito viene illustrata la sintassi per la creazione di una vista del catalogo dati. Nota il tipo di `MULTI DIALECT` visualizzazione. Ciò distingue la vista del catalogo dati dalle altre viste. Il `SECURITY` predicato è specificato come `DEFINER`. Ciò indica una vista del catalogo dati con `DEFINER` semantica.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW [IF NOT EXISTS] view_name
[(column_name [COMMENT column_comment], ...) ]
[ COMMENT view_comment ]
[TBLPROPERTIES (property_name = property_value, ... )]
SECURITY DEFINER
AS query;
```

Di seguito è riportato un esempio di `CREATE` dichiarazione, che segue la sintassi:

```
CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY order_date
```

È inoltre possibile creare una vista in modalità `dry-run`, utilizzando `SQL`, per testare la creazione della vista, senza creare effettivamente la risorsa. L'utilizzo di questa opzione comporta una «esecuzione a secco» che convalida l'input e, se la convalida ha esito positivo, restituisce il JSON dell'oggetto della tabella AWS Glue che rappresenterà la vista. In questo caso, la visualizzazione effettiva non viene creata.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
SECURITY DEFINER
[ SHOW VIEW JSON ]
AS view-sql
```

Using the AWS CLI

Note

Quando si utilizza il comando CLI, l'SQL utilizzato per creare la vista non viene analizzato. Ciò può causare un caso in cui la vista viene creata, ma le query non hanno esito positivo. Assicuratevi di testare la sintassi SQL prima di creare la vista.

Utilizzate il seguente comando CLI per creare una vista:

```
aws glue create-table --cli-input-json '{
  "DatabaseName": "database",
  "TableInput": {
    "Name": "view",
    "StorageDescriptor": {
      "Columns": [
        {
          "Name": "col1",
          "Type": "data-type"
        },
        ...
        {
          "Name": "col_n",
          "Type": "data-type"
        }
      ],
      "SerdeInfo": {}
    },
    "ViewDefinition": {
      "SubObjects": [
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-table1",
        ...
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-tableN",
      ],
      "IsProtected": true,
      "Representations": [
        {
          "Dialect": "SPARK",
          "DialectVersion": "1.0",
          "ViewOriginalText": "Spark-SQL",
          "ViewExpandedText": "Spark-SQL"
        }
      ]
    }
  }
}
```

```

    }
  ]
}
}'

```

Operazioni supportate per le viste

I seguenti frammenti di comandi mostrano vari modi di lavorare con le viste nel Catalogo dati:

- CREA VISTA

Crea una vista data-catalog. Di seguito è riportato un esempio che mostra la creazione di una vista da una tabella esistente:

```

CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER AS SELECT * FROM my_catalog.my_database.source_table

```

- ALTER VIEW

Sintassi disponibile:

- ALTER VIEW view_name [FORCE] ADD DIALECT AS query
- ALTER VIEW view_name [FORCE] UPDATE DIALECT AS query
- ALTER VIEW view_name DROP DIALECT

È possibile utilizzare l'opzione FORCE ADD DIALECT per forzare l'aggiornamento dello schema e degli oggetti secondari secondo il nuovo dialetto del motore. Tieni presente che questa operazione può causare errori di query se non utilizzi anche FORCE per aggiornare altri dialetti del motore. Di seguito viene illustrato un esempio:

```

ALTER VIEW catalog_view FORCE ADD DIALECT
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY orderdate;

```

Di seguito viene illustrato come modificare una vista per aggiornare il dialetto:

```

ALTER VIEW catalog_view UPDATE DIALECT AS

```

```
SELECT count(*) FROM my_catalog.my_database.source_table;
```

- DESCRIVI LA VISTA

Sintassi disponibile per descrivere una vista:

- `SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]`— Se l'utente dispone delle autorizzazioni AWS Glue and Lake Formation necessarie per descrivere la vista, può elencare le colonne. Di seguito vengono illustrati un paio di comandi di esempio per la visualizzazione delle colonne:

```
SHOW COLUMNS FROM my_database.source_table;
SHOW COLUMNS IN my_database.source_table;
```

- `DESCRIBE view_name`— Se l'utente dispone delle autorizzazioni AWS Glue and Lake Formation necessarie per descrivere la vista, può elencare le colonne della vista insieme ai relativi metadati.

- RILASCIA LA VISTA

Sintassi disponibile:

- `DROP VIEW [IF EXISTS] view_name`

L'esempio seguente mostra un'istruzione `DRDP` che verifica l'esistenza di una vista prima di eliminarla:

```
DROP VIEW IF EXISTS catalog_view;
```

- MOSTRA CREA VISUALIZZAZIONE

- `SHOW CREATE VIEW view_name`: mostra l'istruzione SQL che crea la vista specificata. Di seguito è riportato un esempio che mostra la creazione di una vista data-catalog:

```
SHOW CREATE TABLE my_database.catalog_view;
CREATE PROTECTED MULTI DIALECT VIEW my_catalog.my_database.catalog_view (
  net_profit,
  customer_id,
  item_id,
  sold_date)
TBLPROPERTIES (
  'transient_lastDdlTime' = '1736267222')
SECURITY DEFINER AS SELECT * FROM
```

```
my_database.store_sales_partitioned_lf WHERE customer_id IN (SELECT customer_id
from source_table limit 10)
```

- MOSTRA VISUALIZZAZIONI

Elenca tutte le viste del catalogo, ad esempio viste regolari, visualizzazioni multidialettali (MDV) e MDV senza dialetto Spark. La sintassi disponibile è la seguente:

- `SHOW VIEWS [{ FROM | IN } database_name] [LIKE regex_pattern]:`

Di seguito viene illustrato un comando di esempio per mostrare le viste:

```
SHOW VIEWS IN marketing_analytics LIKE 'catalog_view*';
```

Per ulteriori informazioni sulla creazione e la configurazione delle viste del catalogo dati, consulta Building [AWS Glue Data Catalog views](#) nella Developer Guide. AWS Lake Formation

Interrogazione di una vista di Catalogo Dati

Dopo aver creato una vista del catalogo dati, puoi interrogarla utilizzando un job Amazon EMR Serverless Spark con controllo granulare degli accessi abilitato AWS Lake Formation . Il ruolo di job runtime deve disporre dell'autorizzazione Lake Formation per la visualizzazione Data Catalog. Non è necessario concedere l'accesso alle tabelle sottostanti a cui si fa riferimento nella vista.

Dopo aver impostato tutto, è possibile eseguire query sulla vista. Ad esempio, dopo aver creato un'applicazione EMR Serverless in EMR Studio, esegui la seguente query per accedere a una vista.

```
SELECT * from my_database.catalog_view LIMIT 10;
```

Una funzione utile è la `invoker_principal` Restituisce l'identificatore univoco del ruolo EMRS Job Runtime. Questo può essere usato per controllare l'output della vista, in base al principio di invocazione. Puoi usarlo per aggiungere una condizione nella tua vista che perfeziona i risultati della query, in base al ruolo chiamante. Il ruolo di job runtime deve disporre dell'autorizzazione all'azione `LakeFormation:GetDataLakePrincipal` IAM per utilizzare questa funzione.

```
select invoker_principal();
```

È possibile aggiungere questa funzione a una WHERE clausola, ad esempio, per perfezionare i risultati delle query.

Considerazioni e limitazioni

Quando si creano viste del catalogo dati, si applica quanto segue:

- Puoi creare viste del catalogo dati solo con Amazon EMR 7.6 e versioni successive.
- Il definitore della vista di Catalogo dati deve avere l'accesso SELECT alle tabelle di base sottostanti a cui la vista accede. La creazione della vista di Catalogo dati non riesce se una tabella base specifica ha dei filtri Lake Formation imposti sul ruolo del definitore.
- Le tabelle di base non devono avere l'autorizzazione del IAMAllowedPrincipals data lake in Lake Formation. Se presente, si verifica l'errore Multi Dialect views può fare riferimento solo a tabelle senza le autorizzazioni IAMAllowed Principal.
- La posizione Amazon S3 della tabella deve essere registrata come posizione del data lake. Se la tabella non è registrata, si verifica l'errore Le viste multidialettali possono fare riferimento solo alle tabelle gestite da Lake Formation. Per informazioni su come registrare le sedi Amazon S3 in Lake Formation, consulta [Registrazione di una sede Amazon S3](#) nella Developer Guide. AWS Lake Formation
- Puoi creare solo viste PROTECTED in Catalogo dati. Le viste UNPROTECTED non sono supportate.
- Non puoi fare riferimento alle tabelle di un altro AWS account in una definizione di visualizzazione del catalogo dati. Inoltre, non puoi fare riferimento a una tabella nello stesso account che si trova in una Regione separata.
- Per condividere i dati tra un account o un'area geografica, l'intera visualizzazione deve essere condivisa tra account e regioni, utilizzando i link alle risorse di Lake Formation.
- Le funzioni definite dall'utente (UDFs) non sono supportate.
- È possibile utilizzare viste basate sulle tabelle Iceberg. Sono supportati anche i formati a tabella aperta Apache Hudi e Delta Lake.
- Le viste di Catalogo Dati non possono fare riferimento ad altre viste.
- Lo schema di visualizzazione di AWS Glue Data Catalog viene sempre memorizzato in lettere minuscole. Ad esempio, se si utilizza un'istruzione DDL per creare una vista del Glue Data Catalog con una colonna denominataCastle, la colonna creata nel Glue Data Catalog verrà composta in minuscolo, to. castle Se poi specificate il nome della colonna in una query DML come Castle oCASTLE, EMR Spark renderà il nome in minuscolo per consentirvi di eseguire la query. Tuttavia, l'intestazione della colonna viene visualizzata utilizzando il maiuscolo specificato nella query.

Se desiderate che una query abbia esito negativo nel caso in cui il nome di colonna specificato nella query DML non corrisponda al nome della colonna nel Glue Data Catalog, impostate `spark.sql.caseSensitive=true`.

Supporto per il formato a tabella aperta

EMR Serverless supporta le query SELECT su Apache Hive, Apache Iceberg, Delta Lake (7.6.0+) e Apache Hudi (7.6.0+). A partire da EMR 7.12, le operazioni DML e DDL che modificano i dati delle tabelle sono supportate per le tabelle Apache Hive, Apache Iceberg e Delta Lake utilizzando le credenziali vendute di Lake Formation.

Considerazioni e limitazioni

Ambito generale

Esamina le seguenti limitazioni quando usi Lake Formation con 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 [Calcolatore dei prezzi AWS](#)

- [Amazon EMR Serverless with Lake Formation è disponibile in tutte le regioni serverless EMR supportate.](#)
- Le applicazioni abilitate per Lake Formation non supportano l'utilizzo di immagini [EMR Serverless](#) personalizzate.
- Non puoi smettere di lavorare `DynamicResourceAllocation` per Lake Formation.
- Si può usare Lake Formation solo con i processi Spark.
- EMR Serverless with Lake Formation supporta solo una singola sessione Spark durante un job.
- EMR Serverless with Lake Formation supporta solo le query tabellari tra account condivise tramite link alle risorse.
- I seguenti elementi non sono supportati:
 - Resilient Distributed Dataset (RDD)
 - Streaming di Spark

- Controllo degli accessi per colonne annidate
- EMR Serverless blocca le funzionalità che potrebbero compromettere il completo isolamento dei driver di sistema, tra cui:
 - UDTs, Hive e qualsiasi funzione UDFs definita dall'utente che coinvolga classi personalizzate
 - Origini dati personalizzate
 - Fornitura di jar aggiuntivi per l'estensione, il connettore o il metastore di Spark
 - Comando `ANALYZE TABLE`
- [Se la tua applicazione EMR Serverless si trova in una sottorete privata con endpoint VPC per Amazon S3 e alleggi una policy di endpoint per controllare l'accesso, prima che i tuoi job possano inviare i dati di log a AWS Managed Amazon S3, includi le autorizzazioni dettagliate in Managed Storage nella tua policy VPC all'endpoint gateway S3.](#) Per la risoluzione dei problemi relativi alle richieste, contatta l'assistenza. AWS
- A partire da Amazon EMR 7.9.0, Spark FGAC supporta il AFile sistema S3 se utilizzato con lo schema `s3a://`.
- Amazon EMR 7.11 supporta la creazione di tabelle gestite tramite CTAS.
- Amazon EMR 7.12 supporta la creazione di tabelle gestite ed esterne utilizzando CTAS.

Permissions

- Per applicare i controlli di accesso, le operazioni `EXPLAIN PLAN` e `DDL` come `DESCRIBE TABLE` non espongono informazioni riservate.
- Quando registri una posizione in una tabella con Lake Formation, l'accesso ai dati utilizza le credenziali archiviate di Lake Formation anziché le autorizzazioni IAM del ruolo di job runtime EMR Serverless. I processi falliranno se il ruolo registrato per la posizione della tabella non è configurato correttamente, anche se il ruolo di runtime dispone delle autorizzazioni S3 IAM per quella posizione.
- A partire da Amazon EMR 7.12, puoi scrivere su tabelle Hive e Iceberg esistenti utilizzando `DataFrameWriter (V2)` con credenziali Lake Formation in modalità di aggiunta. Per le operazioni di sovrascrittura o durante la creazione di nuove tabelle, EMR utilizza le credenziali del ruolo di runtime per modificare i dati della tabella.
- Le seguenti limitazioni si applicano quando si utilizzano viste o tabelle memorizzate nella cache come dati di origine (queste limitazioni non si applicano alle viste del AWS Glue Data Catalog):
 - Per le operazioni `MERGE`, `DELETE` e `UPDATE`

- Supportato: utilizzo di viste e tabelle memorizzate nella cache come tabelle di origine.
- Non supportato: utilizzo di viste e tabelle memorizzate nella cache nelle clausole di assegnazione e condizione.
- Per le operazioni CREATE OR REPLACE e REPLACE TABLE AS SELECT:
 - Non supportata: utilizzo di viste e tabelle memorizzate nella cache come tabelle di origine.
- Le tabelle Delta Lake con UDFs dati di origine supportano le operazioni MERGE, DELETE e UPDATE solo quando il vettore di eliminazione è abilitato.

Registri e debug

- EMR Serverless limita l'accesso ai registri Spark dei driver di sistema sulle applicazioni abilitate per Lake Formation. Poiché il driver di sistema viene eseguito con autorizzazioni elevate, gli eventi e i registri 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 disabilita l'accesso ai registri dei driver di sistema.
- I log dei profili di sistema vengono sempre conservati nello storage gestito: si tratta di un'impostazione obbligatoria che non può essere disabilitata. Questi registri vengono archiviati in modo sicuro e crittografati utilizzando una chiave KMS gestita dal cliente o una chiave KMS gestita. AWS

Iceberg

Esamina le seguenti considerazioni quando usi 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`, `files`, `manifests`, e `refs`. Amazon EMR nasconde le colonne che potrebbero contenere dati sensibili, ad esempio `partitions`, `path` e `summaries`. Questa limitazione non si applica alle tabelle Iceberg che non sono registrate in Lake Formation.
- Le tabelle non registrate in Lake Formation supportano tutte le stored procedure Iceberg. Le procedure di `register_table` e `migrate` non sono supportate per nessuna tabella.
- Ti suggeriamo di utilizzare Iceberg DataFrameWriter V2 anziché V1.

Risoluzione dei problemi

Consultare le sezioni seguenti per vedere le soluzioni di risoluzione dei problemi.

Registrazione dei log

EMR Serverless utilizza i profili di risorse Spark per suddividere l'esecuzione del lavoro. EMR Serverless 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 per le attività eseguite come profilo utente.

[Per ulteriori informazioni sul debug dei job abilitati per Lake Formation, consulta *Debugging jobs*.](#)

Live UI e Spark History Server

Live UI e Spark History Server contengono tutti gli eventi Spark generati dal profilo utente e gli eventi oscurati generati dal driver di sistema.

È possibile visualizzare tutte le attività dell'utente e dei driver di sistema nella scheda Executors. Tuttavia, i link di registro sono disponibili solo per il profilo utente. Inoltre, alcune informazioni sono oscurate da Live UI, come il numero di record di output.

Il processo ha avuto esito negativo con autorizzazioni Lake Formation insufficienti

Assicurarsi che il ruolo di runtime del processo disponga delle autorizzazioni per eseguire SELECT e DESCRIBE sulla tabella a cui si sta accedendo.

Processo con esecuzione RDD non riuscita

EMR Serverless attualmente non supporta operazioni RDD (Resilient Distributed Dataset) sui job abilitati per Lake Formation.

Impossibile accedere ai file di dati in Amazon S3

Assicurarsi di aver registrato la posizione del data lake in Lake Formation.

Eccezione di convalida della sicurezza

EMR Serverless ha rilevato un errore di convalida della sicurezza. Contatta l'AWS assistenza per ricevere assistenza.

Condivisione del catalogo dati e delle tabelle di AWS Glue tra account

È possibile condividere database e tabelle tra account e continuare a utilizzare Lake Formation. Per ulteriori informazioni, consulta [Condivisione dei dati tra account in Lake Formation](#) e [Come posso condividere AWS Glue Data Catalog e tabelle utilizzando più account?](#) AWS Lake Formation.

API nativa di Spark con elenco consentito per il controllo degli accessi PySpark

Per mantenere i controlli di sicurezza e accesso ai dati, il controllo granulare degli accessi di Spark (FGAC) limita determinate funzioni. PySpark Queste restrizioni vengono applicate tramite:

- Blocco esplicito che impedisce l'esecuzione della funzione
- Incompatibilità dell'architettura che rendono le funzioni non funzionali
- Funzioni che possono generare errori, restituire messaggi di accesso negato o non eseguire alcuna operazione quando vengono chiamate

Le seguenti PySpark funzionalità non sono supportate in Spark FGAC:

- Operazioni RDD (bloccate con Spark Exception) RDDUnsupported
- Spark Connect (non supportato)
- Spark Streaming (non supportato)

Sebbene abbiamo testato le funzioni elencate in un ambiente FGAC Spark nativo e confermato che funzionano come previsto, i nostri test in genere coprono solo l'utilizzo di base di ciascuna API. Le funzioni con più tipi di input o percorsi logici complessi possono presentare scenari non testati.

Per tutte le funzioni non elencate qui e che non rientrano chiaramente nelle categorie non supportate di cui sopra, consigliamo di:

- Testarle prima in un ambiente gamma o in una distribuzione su piccola scala
- Verifica del loro comportamento prima di utilizzarli in produzione

Note

Se vedi elencato un metodo di classe ma non la sua classe base, il metodo dovrebbe comunque funzionare, significa solo che non abbiamo verificato esplicitamente il costruttore della classe base.

L' PySpark API è organizzata in moduli. Il supporto generale per i metodi all'interno di ciascun modulo è dettagliato nella tabella seguente.

Nome del modulo	Stato	Note
pyspark_core	Supportata	Questo modulo contiene le classi RDD principali e queste funzioni per lo più non sono supportate.
pyspark_sql	Supportata	
pyspark_testing	Supportata	
pyspark_resource	Supportata	
pyspark_streaming	Bloccato	L'utilizzo dello streaming è bloccato in Spark FGAC.
pyspark_mllib	sperimentale	Questo modulo contiene operazioni ML basate su RDD e queste funzioni per lo più non sono supportate. Questo modulo non è stato testato a fondo.
pyspark_ml	sperimentale	Questo modulo contiene operazioni di DataFrame machine learning basate e queste funzioni sono per lo più supportate. Questo modulo non è stato testato a fondo.
pyspark_pandas	Supportata	
pyspark_pandas_slow	Supportata	

Nome del modulo	Stato	Note
pyspark_connect	Bloccato	L'utilizzo di Spark Connect è bloccato in Spark FGAC.
pyspark_pandas_connect	Bloccato	L'utilizzo di Spark Connect è bloccato in Spark FGAC.
pyspark_pandas_slow_connect	Bloccato	L'utilizzo di Spark Connect è bloccato in Spark FGAC.
pyspark_errors	sperimentale	Questo modulo non è stato testato a fondo. Le classi di errore personalizzate non possono essere utilizzate.

Elenco delle API consentite

Per un elenco scaricabile e più facile da cercare, un file con i moduli e le classi è disponibile nelle [funzioni Python consentite](#) in Native FGAC.

Crittografia tra lavoratori

Con le versioni 6.15.0 e successive di Amazon EMR, abilita la comunicazione crittografata Mutual-TLS tra i lavoratori che eseguono i job Spark. Se abilitato, EMR Serverless genera e distribuisce automaticamente un certificato univoco per ogni lavoratore assegnato nell'ambito delle esecuzioni dei job. Quando questi lavoratori comunicano per scambiare messaggi di controllo o trasferire dati shuffle, stabiliscono una connessione TLS reciproca e utilizzano i certificati configurati per verificare l'identità reciproca. Se un lavoratore non è in grado di verificare un altro certificato, l'handshake TLS fallisce e EMR Serverless interrompe la connessione tra i due certificati.

Se utilizzi Lake Formation con EMR Serverless, la crittografia Mutual-TLS è abilitata per impostazione predefinita.

Abilitazione della crittografia Mutual-TLS su EMR Serverless

Per abilitare la crittografia TLS reciproca sulla tua applicazione Spark, imposta su `true` durante la `spark.ssl.internode.enabled` creazione dell'applicazione [EMR Serverless](#). Se utilizzi la AWS console per creare un'applicazione EMR Serverless, scegli Usa impostazioni personalizzate, quindi espandi Configurazione dell'applicazione e inserisci le tue `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 crittografia TLS reciproca per le singole esecuzioni di job Spark, imposta su `true` durante `spark.ssl.internode.enabled` l'utilizzo `spark-submit`

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

Crittografia del disco con KMS CMK

EMR Serverless crittografa per impostazione predefinita tutti i dischi collegati ai lavoratori utilizzando chiavi di crittografia di proprietà del servizio. Facoltativamente, è possibile scegliere di crittografare questi dischi utilizzando le proprie chiavi gestite dal cliente (`CMK`). AWS KMS CMKs Ciò offre un maggiore controllo sulle chiavi di crittografia, inclusa la possibilità di stabilire e mantenere le politiche chiave e di controllare l'utilizzo delle chiavi.

È possibile configurare la crittografia del disco durante la creazione di un'applicazione o quando si inviano singoli lavori. Se abilitata a livello di applicazione, tutti i lavori su quell'applicazione ereditano le impostazioni di crittografia. È inoltre possibile sovrascrivere le impostazioni predefinite dell'applicazione specificando una configurazione di crittografia del disco quando si invia un lavoro.

Note

La crittografia dei dischi EMR Serverless supporta solo chiavi KMS simmetriche. Le chiavi KMS asimmetriche non sono supportate. È necessario utilizzare una chiave KMS di

crittografia simmetrica creata in. AWS KMS [Per ulteriori informazioni su AWS KMS, consulta Cos'è? AWS KMS](#)

Utilizzo del contesto di crittografia

Facoltativamente, EMR Serverless utilizza il contesto di crittografia per fornire dati autenticati aggiuntivi per le operazioni di crittografia. Il contesto di crittografia è un insieme di coppie chiave-valore che possono contenere dati autenticati aggiuntivi non segreti. Il contesto di crittografia è associato crittograficamente ai dati crittografati, quindi è necessario lo stesso contesto di crittografia per decrittografare i dati.

In EMR Serverless, è possibile specificare il contesto di crittografia personalizzato durante la configurazione della crittografia del disco. Questo contesto di crittografia è incluso nei AWS CloudTrail log per aiutarti a identificare e comprendere le tue operazioni KMS.

Note

Non archiviate le informazioni sensibili in un contesto di crittografia così come appaiono in testo semplice nei log. AWS CloudTrail

Configurazione della crittografia del disco con chiavi gestite dal cliente

CreateApplication

Per crittografare i dischi con la tua chiave KMS, includi il `diskEncryptionConfiguration` parametro durante la creazione di un'applicazione EMR Serverless.

```
aws emr-serverless create-application \  
  --type TYPE \  
  --name APPLICATION_ID \  
  --release-label RELEASE_LABEL \  
  --region AWS_REGION \  
  --disk-encryption-configuration '{  
    "encryptionKeyArn": "key-arn",  
    "encryptionContext": {  
      "key": "value"  
    }  
  }'  
'
```

UpdateApplication

Per aggiornare il contesto di crittografia and/or ARN della chiave KMS, specificare `diskEncryptionConfiguration` il parametro con i nuovi valori durante l'aggiornamento di un'applicazione.

```
aws emr-serverless update-application \  
  --name APPLICATION_ID \  
  --region AWS_REGION \  
  --disk-encryption-configuration '{  
    "encryptionKeyArn": "key-arn",  
    "encryptionContext": {  
      "key": "value"  
    }  
  }'
```

Note

Per annullare la crittografia del disco configurata su un'applicazione, inserite un valore vuoto `diskEncryptionConfiguration` durante l'aggiornamento dell'applicazione.

StartJobRun

Per crittografare i dischi con la tua chiave KMS, usa la `diskEncryptionConfiguration` configurazione quando invii un job run.

```
--configuration-overrides '{  
  "diskEncryptionConfiguration": {  
    "encryptionKeyArn": "key-arn",  
    "encryptionContext": {  
      "key": "value"  
    }  
  }  
}'
```

Endpoint Livy pubblici

Per crittografare i dischi con la tua chiave KMS durante la creazione di sessioni Spark tramite endpoint Livy pubblici, specifica la configurazione di crittografia nell'oggetto della sessione. `conf`

```

data = {
  "kind": "pyspark",
  "heartbeatTimeoutInSeconds": 60,
  "conf": {
    "emr-serverless.session.executionRoleArn": "role_arn",
    "spark.emr-serverless.disk.encryptedKeyArn": "key-arn",
    "spark.emr-serverless.disk.encryptedContext": "key1:value1,key2:value2" #
Optional
  }
}

# Send request to create a session with the Livy API endpoint
request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
headers=headers)

```

Autorizzazioni richieste per la crittografia del disco

Autorizzazioni delle chiavi di crittografia per EMR Serverless

Quando si crittografano i dischi con la propria chiave di crittografia, è necessario configurare le seguenti autorizzazioni per la chiave KMS per il principale: `emr-serverless.amazonaws.com`

- `kms:GenerateDataKey`: Per generare chiavi dati per crittografare i volumi del disco
- `kms:Decrypt`: Per decrittografare le chiavi dati quando si accede ai contenuti crittografati del disco

```

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

```

```

    "StringEquals": {
      "kms:EncryptionContext:applicationId": "application-id",
      "aws:SourceAccount": "aws-account-id"
    }
  }
}

```

Come procedura consigliata in materia di sicurezza, ti consigliamo di aggiungere una chiave di `aws:SourceArn` condizione alla politica delle chiavi KMS. La chiave di condizione globale IAM `aws:SourceArn` aiuta a garantire che EMR Serverless utilizzi la chiave KMS solo per l'ARN di un'applicazione. Inoltre, l'inclusione della chiave di `aws:SourceAccount` condizione fornisce un altro livello di sicurezza limitando l'uso della chiave KMS alle richieste provenienti dall'ID dell'account specificato nella AWS condizione.

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

```

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

```

Autorizzazioni utente richieste

L'utente che invia il lavoro deve disporre delle autorizzazioni per utilizzare la chiave. Puoi specificare le autorizzazioni nella politica delle chiavi KMS o nella politica IAM per l'utente, il gruppo o il ruolo. Se l'utente che invia il lavoro non dispone delle autorizzazioni della chiave KMS, EMR Serverless rifiuta l'invio dell'esecuzione del lavoro.

Esempi di policy delle chiavi

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

- `kms:DescribeKey`: Per verificare che la chiave KMS gestita dal cliente sia abilitata e SYMMETRIC prima di utilizzarla.

```
{
  "Sid": "Enable DescribeKey",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/user-name"
  },
  "Action": [
    "kms:DescribeKey"
  ],
  "Resource": "*"
},
{
  "Sid": "Enable GDK and Decrypt",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/user-name"
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "emr-serverless.region.amazonaws.com",
      "kms:EncryptionContext:key": "value"
    }
  }
}
}
```

Come procedura consigliata in materia di sicurezza, consigliamo di aggiungere una chiave di `kms:viaService` condizione alla politica delle chiavi KMS. Limita l'uso della chiave KMS alle richieste di convalida provenienti solo da `emr-serverless`.

Policy IAM di esempio

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

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

Monitoraggio dell'utilizzo delle chiavi

È possibile monitorare l'uso delle chiavi gestite dai clienti in EMR Serverless tramite AWS CloudTrail. AWS CloudTrail acquisisce tutte le chiamate API AWS KMS come eventi, incluse le chiamate dalla console EMR Serverless, dall'API AWS Serverless EMR, dalla CLI o dall'SDK. AWS

Le informazioni acquisite includono il contesto di crittografia specificato, che può aiutarti a identificare e controllare le risorse EMR Serverless specifiche che hanno utilizzato la tua chiave KMS. Ad esempio, potresti vedere eventi simili ai seguenti in AWS CloudTrail. Per ulteriori informazioni sull'utilizzo AWS CloudTrail, consulta la [Guida AWS CloudTrail per l'utente](#).

GenerateDataKey

Evento di esempio per GenerateDataKey le operazioni quando EMR Serverless crea volumi di dischi crittografati

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AWSService",
    "principalId": "user",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2025-07-28T21:43:51Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "ipAddress",
  "userAgent": "userAgent",
```

```

    "requestParameters": {
      "encryptionContext": {
        "applicationId": "test"
      },
      "keyId": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample",
      "keySpec": "AES_256"
    },
    "responseElements": null,
    "additionalEventData": {
      "keyMaterialId":
"145c963debe558dfb01848d2a4539da940f3478852f86cfe2f52d5df796a5a02"
    },
    "requestID": "cc9d1c5e-97c4-4a4f-ae7a-e576sample",
    "eventID": "0b0fef09-f28d-4da8-a5a1-17b74sample",
    "readOnly": true,
    "resources": [
      {
        "accountId": "account",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "accountId",
    "eventCategory": "Management"
  }

```

Decrypt

Evento di esempio per le operazioni di decrittografia quando EMR Serverless accede a dati crittografati.

```

{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AWSService",
    "principalId": "user",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2025-07-28T21:43:51Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",

```

```

"awsRegion": "us-west-2",
"sourceIPAddress": "ipAddress",
"userAgent": "userAgent",
"requestParameters": {
  "encryptionContext": {
    "applicationId": "test"
  },
  "keyId": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample",
  "keySpec": "AES_256"
},
"responseElements": null,
"additionalEventData": {
  "keyMaterialId":
"145c963debe558dfb01848d2a4539da940f3478852f86cfe2f52d5df796a5a02"
},
"requestID": "cc9d1c5e-97c4-4a4f-ae7a-e576sample",
"eventID": "0b0fef09-f28d-4da8-a5a1-17b74sample",
"readOnly": true,
"resources": [
  {
    "accountId": "account",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "accountId",
"eventCategory": "Management"
}

```

Ulteriori informazioni

Le seguenti risorse forniscono ulteriori informazioni sulla crittografia dei dati a riposo.

- [Per ulteriori informazioni sui concetti di AWS KMS base, consulta la Guida per gli sviluppatori.AWS KMS](#)
- Per ulteriori informazioni sulle migliori pratiche di sicurezza per AWS KMS, consulta la [Guida per gli AWS KMS sviluppatori](#).

Secrets Manager per la protezione dei dati con EMR Serverless

Gestione dei segreti AWS è un servizio di archiviazione segreto per proteggere le credenziali del database, le chiavi API e altre informazioni segrete. Quindi, nel codice, sostituisci le credenziali hardcoded con una chiamata API a Secrets Manager. Questo aiuta a garantire che il segreto non possa essere compromesso da qualcuno che esamina il tuo codice, perché il segreto non è presente. Per una panoramica, consulta la Guida per l'[Gestione dei segreti AWS utente](#).

Secrets Manager crittografa i segreti utilizzando AWS Key Management Service le chiavi. Per ulteriori informazioni, consulta la sezione [Crittografia e decrittografia segrete nella Guida](#) per l'Gestione dei segreti AWS 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 [Rotate Gestione dei segreti AWS secrets nella Guida](#) per l'Gestione dei segreti AWS utente.

Amazon EMR Serverless si integra Gestione dei segreti AWS 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 trasmettono dati di configurazione sensibili a EMR Serverless in testo semplice per poi esporli a fonti esterne. APIs Se si indica 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 i job in esecuzione.

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 job.

Come creare un segreto

Per creare un segreto, segui i passaggi in [Creare un Gestione dei segreti AWS segreto](#) nella Guida per l'Gestione dei segreti AWS utente. Nel passaggio 3, scegli il campo Testo normale per inserire il tuo valore sensibile.

Fornisci un segreto in una classificazione di configurazione

Gli esempi seguenti 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, fare riferimento a [Configurazione predefinita dell'applicazione per EMR Serverless](#).

Negli esempi, `SecretName` sostituisco con il nome del segreto da recuperare. Per ulteriori informazioni, vedi [Come creare un segreto](#).

In questa sezione

- [Specificate i riferimenti segreti - Spark](#)
- [Specificare i riferimenti segreti - Hive](#)

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

```

    "s3MonitoringConfiguration": {
      "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
    }
  }
}'

```

Example— 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://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.q1",
      "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/
emr-serverless-hive/hive/warehouse
                    --hiveconf javax.jdo.option.ConnectionUserName=username
                    --hiveconf
javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
                    --hiveconf
hive.metastore.client.factory.class=org.apache.hadoop.hive.q1.metadata.SessionHiveMetaStoreCli
                    --hiveconf
javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver

```

```

        --hiveconf javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name"
    }
}' \
--configuration-overrides '{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket"
        }
    }
}'

```

Example— 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.q1.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 chiave KMS gestita dal cliente per consentire a EMR Serverless di leggere il valore segreto. Per ulteriori informazioni, consulta la sezione [Autorizzazioni per la chiave KMS nella Guida per l'utente.Gestione dei segreti AWS](#)

Nella seguente politica, sostituiscila *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 secret in per consentire Gestione dei segreti AWS a EMR Serverless di recuperare i valori segreti. Per garantire

che solo un'applicazione specifica possa recuperare questo segreto, è possibile specificare facoltativamente l'ID dell'applicazione EMR Serverless come condizione nella policy.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSECRETSMANAGERGetsecretvalue",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "emr-serverless.amazonaws.com"
        ]
      },
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/
**
        }
      }
    }
  ]
}
```

Crea il tuo segreto con la seguente politica per la chiave gestita dal cliente AWS Key Management Service ():AWS KMS

Politica per la chiave gestita dal cliente AWS KMS

```
{
  "Sid": "Allow EMR Serverless to use the key for decrypting secrets",
  "Effect": "Allow",
```

```
"Principal": {
  "Service": [
    "emr-serverless.amazonaws.com"
  ]
},
"Action": [
  "kms:Decrypt",
  "kms:DescribeKey"
],
"Resource": "*",
"Condition": {
  "StringEquals": {
    "kms:ViaService": "secretsmanager.Regione AWS.amazonaws.com"
  }
}
}
```

Ruotare il segreto

La rotazione avviene quando si aggiorna periodicamente un segreto. Puoi Gestione dei segreti AWS configurare la rotazione automatica del segreto secondo una pianificazione da te specificata. In questo modo, puoi sostituire i segreti a lungo termine con segreti a breve termine. Questo aiuta a ridurre il rischio di compromessi. EMR Serverless 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 EMR Serverless

Panoramica di S3 Access Grants per EMR Serverless

Con le versioni 6.15.0 e successive di Amazon EMR, Amazon S3 Access Grants fornisce una soluzione di controllo degli accessi scalabile per aumentare l'accesso ai dati Amazon S3 da EMR Serverless. Se disponi di una configurazione di autorizzazioni complessa o di grandi dimensioni

per i tuoi dati S3, usa Access Grants per scalare le autorizzazioni dei 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 ruoli IAM collegati alle identità con accesso alla tua applicazione EMR Serverless.

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 conoscere i passaggi per utilizzare S3 Access Grants con altre implementazioni Amazon EMR, consulta la seguente documentazione:

- [Utilizzo di S3 Access Grants con Amazon EMR](#)
- [Utilizzo di S3 Access Grants con Amazon EMR su 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 EMR Serverless. Includi le autorizzazioni IAM 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 ruoli IAM per l'esecuzione del lavoro che dispongono di autorizzazioni aggiuntive per accedere direttamente a S3, gli utenti possono 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 release Amazon EMR 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://amzn-s3-demo-destination-bucket1/
wordcount_output"],
      "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g
--conf spark.executor.instances=1"
    }
  }' \
  --configuration-overrides '{
  "applicationConfiguration": [{
    "classification": "spark-defaults",
    "properties": {
      "spark.hadoop.fs.s3.s3AccessGrants.enabled": "true",
      "spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM": "false"
    }
  }]
}'
```

S3 Access concede considerazioni con EMR Serverless

Per informazioni importanti su supporto, compatibilità e comportamento quando usi Amazon S3 Access Grants con EMR Serverless, [consulta le considerazioni su S3 Access Grants con Amazon EMR nella Amazon EMR Management Guide](#).

Registrazione delle chiamate API Serverless di Amazon EMR utilizzando AWS CloudTrail

Amazon EMR Serverless è integrato con AWS CloudTrail un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o servizio AWS in EMR Serverless. CloudTrail acquisisce tutte le chiamate API per EMR Serverless come eventi. Le chiamate acquisite includono chiamate dalla console EMR Serverless e chiamate in codice alle operazioni dell'API EMR Serverless. Se crei un trail, abilita la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per EMR Serverless. Se non configuri un percorso, puoi comunque accedere agli 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 dettagli aggiuntivi.

Per ulteriori informazioni, consulta la Guida per l'[AWS CloudTrail utente](#).

Informazioni EMR Serverless in CloudTrail

CloudTrail è abilitato sul tuo Account AWS quando crei l'account. Quando si verifica un'attività in EMR Serverless, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella cronologia degli eventi. Puoi accedere, cercare e scaricare eventi recenti in 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, compresi 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 si crea un percorso nella console, questo sarà valido in tutte le Regioni AWS. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, configura altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, fare riferimento a quanto segue:

- [Panoramica della creazione di un percorso](#)

- [CloudTrail servizi e integrazioni supportati](#)
- [Configurazione delle notifiche Amazon SNS per CloudTrail](#)
- [Ricezione di file di CloudTrail registro da più regioni](#) e [ricezione di file di CloudTrail registro da più account](#)

Tutte le azioni EMR Serverless vengono registrate CloudTrail e documentate nell'[EMR Serverless API Reference](#). Ad esempio, le chiamate a `StartJobRun` e le `CancelJobRun` azioni generano `CreateApplication` voci nei file di registro. CloudTrail

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente root o AWS Identity and Access Management (IAM).
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, fare riferimento all'elemento [CloudTrail userIdentity](#).

Informazioni sulle voci dei file di registro EMR 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 ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'`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 EMR Serverless

La sicurezza e la conformità di EMR Serverless vengono valutate da revisori di terze parti nell'ambito di più programmi di AWS conformità, tra cui:

- System and Organization Controls (SOC)
- Payment Card Industry Data Security Standard (PCI DSS)
- Programma federale di gestione dei rischi e delle autorizzazioni (FedRAMP) Moderato
- Health Insurance Portability and Accountability Act (HIPAA)

AWS fornisce un elenco frequentemente aggiornato di AWS servizi nell'ambito di specifici programmi di conformità nella pagina [AWS Services in Scope by Compliance Program](#).

I report di audit di terze parti possono essere scaricati utilizzando AWS Artifact. Per ulteriori informazioni, consulta [Scaricamento dei report in AWS Artifact](#).

Per ulteriori informazioni sui programmi di AWS conformità, consulta Programmi di [AWS conformità](#).

La responsabilità della 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 HIPAA, PCI o FedRAMP Moderate, fornisce risorse per aiutare a: AWS

- [Guide introduttive su sicurezza e conformità che illustrano le](#) considerazioni sull'architettura e i passaggi per implementare ambienti di base incentrati sulla sicurezza e la conformità su AWS
- [AWS Le guide alla conformità per i clienti](#) possono aiutarti a comprendere il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione di 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 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 aiuta a controllare continuamente AWS l'utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

Resilienza in Amazon EMR Serverless

L'infrastruttura AWS globale è costruita attorno AWS a regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con Availability Zones, progetta e gestisci applicazioni e database che eseguono automaticamente il failover tra le zone senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta [AWS Global Infrastructure](#).

Oltre all'infrastruttura AWS globale, Amazon EMR Serverless offre l'integrazione con Amazon S3 tramite EMRFS per supportare le tue esigenze di resilienza e backup dei dati.

Sicurezza dell'infrastruttura in Amazon EMR Serverless

In quanto servizio gestito, Amazon EMR è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi di AWS sicurezza e su come AWS protegge l'infrastruttura, consulta [AWS Cloud Security](#). Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi [Infrastructure Protection](#) in Security Pillar AWS Well-Architected Framework.

Utilizzi chiamate API AWS pubblicate per accedere ad Amazon EMR attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Analisi della configurazione e delle vulnerabilità in Amazon EMR Serverless

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 EMR Serverless](#)
- [Modello di responsabilità condivisa](#)
- [Amazon Web Services: panoramica dei processi di sicurezza](#)

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 l'URL del punto di ingresso per un servizio Web. AWS Oltre agli AWS endpoint standard, alcuni Servizi AWS offrono endpoint FIPS in regioni selezionate. La tabella seguente elenca gli endpoint di servizio per EMR Serverless. [Per ulteriori informazioni, fare riferimento agli endpoint.Servizio AWS](#)

Endpoint di servizio EMR Serverless

Nome Regione	Regione	Endpoint	Protocollo
Stati Uniti orientali (Ohio)	us-east-2 (limitato alle seguenti zone di disponibilità: use2-az1, use2-az2 e) use2-az3	emr-serve rless.us- east-2.am azonaws.com	HTTPS
Stati Uniti orientali (Virginia settentrionale)	us-east-1 (limitato alle seguenti zone di disponibilità: use1-az1, use1-az2, use1-az4, use1-az5, use1-az6)	emr-serve rless.us- east-1.am azonaws.com emr-serverless- fips.us-east -1.amazon aws.com	HTTPS
Stati Uniti occidentali (California settentrionale)	us-west-1	emr-serve rless.us- west-1.am azonaws.com	HTTPS
Stati Uniti occidentali (Oregon)	us-west-2	emr-serve rless.us- west-2.am azonaws.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 (Melbourne)	ap-southeast-4	emr-serverless.ap-southeast-4.amazonaws.com	HTTPS
Asia Pacifico (Malesia)	ap-southeast-5	emr-serverless.ap-southeast-5.amazonaws.com	HTTPS

Nome Regione	Regione	Endpoint	Protocollo
Asia Pacifico (Mumbai)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
Asia Pacifico (Osaka)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS
Asia Pacifico (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

Nome Regione	Regione	Endpoint	Protocollo
Canada (Centrale)	ca-central-1 (limitato alle seguenti zone di disponibilità: cac1-az1 ecac1-az2)	emr-serverless.ca-central-1.amazonaws.com	HTTPS
Canada occidentale (Calgary)	ca-west-1	emr-serverless.ca-west-1.amazonaws.com	HTTPS
Europa (Francoforte)	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS
Europa (Zurigo)	eu-central-2	emr-serverless.eu-central-2.amazonaws.com	HTTPS
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

Nome Regione	Regione	Endpoint	Protocollo
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
Israele (Tel Aviv)	il-central-1	emr-serverless.il-central-1.amazonaws.com	HTTPS
Medio Oriente (Bahrein)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS
Medio Oriente (Emirati Arabi Uniti)	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

Nome Regione	Regione	Endpoint	Protocollo
Cina (Pechino)	cn-north-1 (limitato alle seguenti zone di disponibilità: cnn1-az1, cnn1-az2)	emr-serverless.cn-north-1.amazonaws.com.cn	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

Service Quotas

Le quote di servizio, note anche come limiti, sono il numero massimo di risorse o operazioni di servizio che è Account AWS possibile utilizzare. EMR Serverless raccoglie i parametri di utilizzo delle quote di servizio ogni minuto e li pubblica nel namespace. `AWS/Usage`

Note

AWS I nuovi account hanno quote iniziali inferiori che possono aumentare nel tempo. Amazon EMR Serverless monitora l'utilizzo degli account all'interno di ciascun account Regione AWS, quindi aumenta automaticamente le quote in base all'utilizzo.

La tabella seguente elenca le quote di servizio per EMR Serverless. [Per ulteriori informazioni, fare riferimento alle quote.Servizio AWS](#)

Nome	Limite predefinito	Modificabile?	Description
Numero massimo di v CPUs simultanei per account	16	Sì	Il numero massimo di v CPUs che può essere eseguito contemporaneamente per l'account nella versione corrente. Regione AWS

Limiti di API

Di seguito vengono descritti i limiti API per regione per la tua Account AWS.

Risorsa	Quota predefinita
ListApplications	10 transazioni al secondo. Burst di 50 transazioni 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 transazioni 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 su EMR Serverless.

- EMR Serverless 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 (Taipei)
 - Asia Pacifico (Giacarta)
 - Asia Pacifico (Mumbai)
 - Asia Pacifico (Hyderabad)
 - Asia Pacifico (Osaka)
 - Asia Pacifico (Seoul)
 - Asia Pacifico (Singapore)
 - Asia Pacifico (Sydney)
 - Asia Pacifico (Malesia)
 - Asia Pacifico (Nuova Zelanda)
 - Asia Pacifico (Thailandia)
 - Asia Pacifico (Tokyo)
 - Canada (Centrale)
 - Europa (Francoforte)
 - Europa (Irlanda)
 - Europa (Londra)
 - Europa (Milano)
 - Europa (Parigi)
 - Europa (Spagna)
 - Europa (Stoccolma)

- Medio Oriente (Bahrein)
- Medio Oriente (Emirati Arabi Uniti)
- Messico (centrale)
- Sud America (San Paolo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Per un elenco degli endpoint associati a queste regioni, fare riferimento a [Endpoint di servizio](#)

- Il timeout predefinito per l'esecuzione di un processo è di 12 ore. Puoi modificare questa impostazione con la `executionTimeoutMinutes` proprietà nell'`startJobRunAPI` o nell' AWS SDK. È possibile `executionTimeoutMinutes` impostare su 0 se si desidera che l'esecuzione del processo non scada mai. Ad esempio, se disponi di un'applicazione di streaming, imposta su 0 `executionTimeoutMinutes` per consentire l'esecuzione continua del processo di streaming.
- La `billedResourceUtilization` proprietà nell'`getJobRunAPI` mostra la vCPU, la memoria e lo storage aggregati fatturati per AWS l'esecuzione del processo. 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 la connettività VPC, un job può accedere ad alcuni Servizio AWS endpoint nello stesso. Regione AWS Questi servizi includono Amazon S3, AWS Glue, AWS Lake Formation, Amazon CloudWatch Logs, AWS KMS AWS Security Token Service, Amazon DynamoDB e. Gestione dei segreti AWS Puoi abilitare la connettività VPC per accedere ad altri Servizi AWS tramite [AWS PrivateLink](#), ma non è necessario farlo. Per accedere a servizi esterni, crea la tua applicazione con un VPC.
- EMR Serverless non supporta HDFS. I dischi locali sui lavoratori sono lo storage temporaneo che EMR Serverless utilizza per riprodurre in ordine casuale ed elaborare i dati durante le operazioni di lavoro.

Versioni di rilascio di Amazon EMR Serverless

Una versione di Amazon EMR è un insieme di applicazioni open source dell'ecosistema dei big data. Ogni versione include applicazioni, componenti e funzionalità per big data selezionati per consentire la distribuzione e la configurazione di Amazon EMR Serverless durante l'esecuzione del lavoro.

Con Amazon EMR 6.6.0 e versioni successive, implementa EMR Serverless. Questa opzione di distribuzione non è disponibile con le versioni precedenti di Amazon EMR. Quando invii il lavoro, specifica una delle seguenti versioni supportate.

Argomenti

- [Runtime AWS per Apache Spark \(emr-spark-8.0.0\)](#)
- [Runtime AWS per Apache Spark \(emr-spark-8.0-preview\)](#)
- [EMR Serverless 7.13.0](#)
- [EMR Serverless 7.12.0](#)
- [EMR Serverless 7.11.0](#)
- [EMR Serverless 7.10.0](#)
- [EMR Serverless 7.9.0](#)
- [EMR Serverless 7,80](#)
- [EMR Serverless 7,7,0](#)
- [EMR Serverless 7,6,0](#)
- [EMR Serverless 7,5,0](#)
- [EMR Serverless 7,40](#)
- [EMR Serverless 7.3.0](#)
- [EMR Serverless 7.2.0](#)
- [EMR Serverless 7.1.0](#)
- [EMR Serverless 7,0,0](#)
- [EMR Serverless 6,15,0](#)
- [EMR Serverless 6.14.0](#)
- [EMR Serverless 6,13,0](#)
- [EMR Serverless 6,12,0](#)

- [EMR Serverless 6.11,0](#)
- [EMR Serverless 6.10.0](#)
- [EMR Serverless 6.9.0](#)
- [EMR Serverless 6.8.0](#)
- [EMR Serverless 6.7.0](#)
- [EMR Serverless 6.6.0](#)

Runtime AWS per Apache Spark (emr-spark-8.0.0)

La tabella seguente elenca le versioni delle applicazioni disponibili con (emr-spark-8.0.0). AWS runtime for Apache Spark

Informazioni sulla versione dell'applicazione

Applicazione	Versione
Spark	4.0.2-amzn-0
Iceberg	1.10.1-amzn-0
Delta	4.0.0-amzn-1-spark
Hudi	1.1.0-amzn-0

Note di rilascio di AWS runtime per Apache Spark (emr-spark-8.0.0)

- **Versione GA:** questa è la versione a disponibilità generale che include Apache Spark 4.0.2. AWS runtime for Apache Spark Questa versione è disponibile su EMR Serverless, EMR su EC2 ed EMR su EKS.
- **Disponibilità regionale:** disponibile in tutte le AWS regioni in cui è disponibile EMR Serverless, ad eccezione delle regioni del Medio Oriente (Bahrein) e del Medio Oriente (Emirati Arabi Uniti).
- **Limitazioni note:** l'endpoint sicuro Spark Connect con supporto FGAC nativo non è disponibile in questa versione.
- **Documentazione aggiuntiva** - Per la documentazione [aggiuntiva su Apache Spark, consultate la documentazione di rilascio di Apache Spark 4.0.2.](#)

Nozioni di base

Per iniziare con Apache Spark 4.0.2, crea un'applicazione EMR Serverless utilizzando la CLI: AWS

```
aws emr-serverless create-application --type SPARK \  
  --release-label emr-spark-8.0.0 \  
  --name spark4-serverless \  
  --region us-east-1
```

Note

- Questa versione sostituisce la versione di anteprima (emr-spark-8.0-preview). L'anteprima era limitata a Spark 4.0.1 e mancava di FGAC, Hudi, connettori dati e Persistent Spark History Server.
- Il `--type` parametro per gli usi (maiuscolo). `create-application SPARK`

Runtime AWS per Apache Spark (emr-spark-8.0-preview)

La tabella seguente elenca le versioni dell'applicazione disponibili con (emr-spark-8.0-preview). AWS runtime for Apache Spark

Informazioni sulla versione dell'applicazione

Applicazione	Versione
Spark	4.0.1-amzn-0

Note sulla **versione di AWS runtime per Apache Spark** (emr-spark-8.0-preview)

- **Versione di anteprima:** questa è una versione di anteprima di Apache Spark 4.0.1. AWS runtime for Apache Spark Questa anteprima è disponibile solo su EMR Serverless.
- **Disponibilità regionale:** questa versione di anteprima è disponibile in tutte le AWS regioni in cui è disponibile EMR Serverless, ad eccezione della Cina e delle regioni AWS GovCloud (Stati Uniti).
- **Informazioni sulla versione dell'applicazione:** questa versione viene fornita con le seguenti versioni dell'applicazione:
 - AWS SDK per Java 2.35.5, 1.12.792
 - Python 3.9, 3.11, 3.12
 - Scala 2.13.16

- AmazonCloudWatchAgent 1.300034.0-amzn-0
- Delta 4.0.0-amzn-0-spark
- Iceberg 1.10.0-amzn-spark-0
- Questa versione viene fornita con Amazon Corretto 17 (basato su OpenJDK) per impostazione predefinita per le applicazioni che supportano Corretto 17 (JDK 17).
- Limitazioni dell'anteprima: le seguenti funzionalità non sono disponibili in questa versione di anteprima:
 - Funzionalità interattive e di integrazione: SageMaker Unified Studio, integrazione con EMR Studio, Spark Connect, Livy e non sono supportate. JupyterEnterpriseGateway
 - Formati di tabelle e controllo degli accessi: Hudi, Delta Universal Format e il controllo degli accessi a grana fine (FGAC) con filtri e operatori a livello di riga o colonna non sono supportati. DDL/DML
 - Connettori dati: i connettori spark-sql-kinesis, emr-dynamodb e spark-redshift non sono disponibili.
 - History Server: Il Persistent Spark History Server non è disponibile in questa versione di anteprima. Gli utenti possono comunque accedere all'interfaccia utente live di Spark per monitorare ed eseguire il debug dei job serverless attivi in tempo reale.
 - Funzionalità specializzate: le viste materializzate non sono disponibili.
- Funzionalità di anteprima: è possibile testare le seguenti funzionalità in questa versione di anteprima. Questa versione di anteprima non è consigliata per i carichi di lavoro di produzione:
 - Caratteristiche SQL: modalità ANSI SQL con gestione dei tipi più rigorosa, sintassi SQL PIPE (|>) per le operazioni di concatenamento, tipo di dati VARIANT per dati JSON semistrutturati, script SQL con istruzioni di flusso di controllo e variabili di sessione e funzioni SQL definite dall'utente.
 - Miglioramenti allo streaming: Arbitrary Stateful Processing API v2 con WithState operatore di trasformazione, State Data Source Reader per lo stato di streaming interrogabile (sperimentale) e archivio di stato avanzato con checkpoint migliorato del changelog RockSDB.
 - Supporto per formati di tabella: Apache Iceberg v3 con supporto del tipo di dati VARIANT, integrazione con AWS S3 Tables e Full Table Access (FTA) AWS Lake Formation per le tabelle Iceberg, Delta Lake e Hive.
- Documentazione aggiuntiva - Per la documentazione aggiuntiva [di Apache Spark, consulta la documentazione di rilascio di Apache Spark 4.0.1.](#)

Nozioni di base

Per iniziare con l'anteprima di Apache Spark 4.0.1, crea un'applicazione EMR Serverless utilizzando la CLI: AWS

```
aws emr-serverless create-application --type spark \  
  --release-label emr-spark-8.0-preview \  
  --region us-east-1 --name spark4-preview
```

EMR Serverless 7.13.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.13.0.

Applicazione	Versione
Apache Spark	3.5.6
Apache Hive	3.1.3
Apache Tez	0,10,2

Note di rilascio di EMR Serverless 7.13.0

- Nuove funzionalità
 - Spark Connect per PySpark sessioni interattive — EMR Serverless ora supporta sessioni PySpark interattive tramite Apache Spark Connect. Con Amazon EMR versione 7.13.0 e successive, puoi connetterti a Spark in esecuzione su EMR Serverless da client PySpark locali, inclusi IDE come VS Code e notebook Jupyter. PyCharm Per ulteriori informazioni, consulta [Esegui sessioni interattive con Amazon EMR Serverless tramite Spark Connect](#).

EMR Serverless 7.12.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.12.0.

Applicazione	Versione
Apache Spark	3.5.6

Applicazione	Versione
Apache Hive	3.1.3
Apache Tez	0,10,2

Note di rilascio di EMR Serverless 7.12.0

- Nuove funzionalità
 - Storage serverless per EMR Serverless: Amazon EMR serverless introduce lo storage serverless, con EMR versione 7.12 e successive, che elimina il provisioning locale del disco per i carichi di lavoro Apache Spark. EMR Serverless gestisce automaticamente operazioni di dati intermedi come lo shuffle senza costi di archiviazione. Lo storage serverless separa lo storage dall'elaborazione, permettendo a Spark di rilasciare immediatamente i dipendenti quando sono inattivi anziché mantenerli attivi per conservare i dati temporanei. [Per saperne di più, consulta Serverless storage.](#)
 - Iceberg Materialized Views - A partire da Amazon EMR 7.12.0, Amazon EMR Spark supporta la creazione e la gestione di Iceberg Materialized Views (MV)
 - Hudi Full Table Access - A partire da Amazon EMR 7.12.0, Amazon EMR supporta ora il controllo Full Table Access (FTA) per Apache Hudi in Apache Spark in base alle politiche definite in Lake Formation. Questa funzionalità consente le operazioni di lettura e scrittura dai job di Amazon EMR Spark sulle tabelle registrate di Lake Formation quando il ruolo lavorativo ha accesso completo alla tabella.
 - Aggiornamento della versione Iceberg: Amazon EMR 7.12.0 supporta la versione 1.10 di Apache Iceberg

EMR Serverless 7.11.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.11.0.

Applicazione	Versione
Apache Spark	3.5.6
Apache Hive	3.1.3

Applicazione	Versione
Apache Tez	0,10,2

Note di rilascio di EMR Serverless 7.11.0

- Tempo massimo di esecuzione del Job: il valore massimo per `executionTimeoutMinutes` in `StartJobRun` azione per i job BATCH è di 7 giorni a partire da questa versione. `executionTimeoutMinutes` non può più essere impostato su nessun timeout, ad `0` esempio su nessun timeout, per le esecuzioni di processi in batch.

EMR Serverless 7.10.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.10.0.

Applicazione	Versione
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0,10,2

Note di rilascio di EMR Serverless 7.10.0

- Metriche per EMR Serverless: le metriche di monitoraggio vengono ristrutturare per concentrarsi sulle dimensioni e. `ApplicationName` `JobName` Le metriche precedenti non verranno più aggiornate d'ora in avanti. Per ulteriori informazioni, fare riferimento a [Monitoraggio delle applicazioni e dei job EMR Serverless](#).

EMR Serverless 7.9.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.9.0.

Applicazione	Versione
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0,10,2

EMR Serverless 7,80

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.8.0.

Applicazione	Versione
Apache Spark	3.5.4
Apache Hive	3.1.3
Apache Tez	0,10,2

EMR Serverless 7,7,0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.7.0.

Applicazione	Versione
Apache Spark	3.5.3
Apache Hive	3.1.3
Apache Tez	0,10,2

EMR Serverless 7,6,0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.6.0.

Applicazione	Versione
Apache Spark	3.5.3
Apache Hive	3.1.3
Apache Tez	0,10,2

EMR Serverless 7,5,0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.5.0.

Applicazione	Versione
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7,40

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.4.0.

Applicazione	Versione
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.3.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless la versione 7.3.0.

Applicazione	Versione
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0,10,2

Note di rilascio di EMR Serverless 7.3.0

- Concorrenza e accodamento dei lavori con EMR Serverless: la concorrenza e l'accodamento dei lavori sono abilitati per impostazione predefinita quando crei una nuova applicazione EMR Serverless su Amazon EMR versione 7.3.0 o successiva. Per ulteriori informazioni, consulta [the section called “Concorrenza e attesa dei lavori”](#), che descrive in dettaglio come iniziare con la concorrenza e l'accodamento e contiene anche un elenco di considerazioni sulle funzionalità.

EMR Serverless 7.2.0

La tabella seguente elenca le versioni delle applicazioni disponibili con la versione 7.2.0. EMR Serverless

Applicazione	Versione
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0,10,2

Note di rilascio di EMR Serverless 7.2.0

- Lake Formation con EMR Serverless: ora puoi utilizzarlo AWS Lake Formation per applicare controlli di accesso granulari alle tabelle del 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 EMR Serverless Spark. Per ulteriori informazioni, consulta [the section called “Lake Formation per FGAC”](#) e [the section called “Considerazioni e limitazioni”](#).

EMR Serverless 7.1.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.1.0.

Applicazione	Versione
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7,0,0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 7.0.0.

Applicazione	Versione
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6,15,0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless 6.15.0.

Applicazione	Versione
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

Note di rilascio di EMR Serverless 6.15.0

- **Supporto TLS:** con le versioni 6.15.0 e successive di Amazon EMR Serverless, abilita la comunicazione crittografata Mutual-TLS tra i lavoratori durante le esecuzioni dei job Spark. Se abilitato, EMR Serverless genera automaticamente un certificato univoco per ogni lavoratore che fornisce nell'ambito di un job run che i lavoratori utilizzano durante l'handshake TLS per autenticarsi a vicenda e stabilire un canale crittografato per elaborare i dati in modo sicuro. [Per ulteriori informazioni sulla crittografia Mutual-TLS, fare riferimento alla crittografia. Inter-worker](#)

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	0.10.2

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	0.10.2

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	0.10.2

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	0.10.2

Note di rilascio di EMR Serverless 6.11.0

- [Accedi alle risorse S3 da altri account](#) - Con le versioni 6.11.0 e successive, puoi configurare più ruoli IAM da assumere quando accedi ai bucket Amazon S3 in account diversi da EMR Serverless. AWS

EMR Serverless 6.10.0

La tabella seguente elenca le versioni delle applicazioni disponibili con la versione 6.10.0. EMR Serverless

Applicazione	Versione
Apache Spark	3.3.1
Apache Hive	3.1.3

Applicazione	Versione
Apache Tez	0.10.2

Note di rilascio di EMR Serverless 6.10.0

- Per le applicazioni EMR Serverless con release 6.10.0 o superiore, il valore predefinito per la proprietà è `spark.dynamicAllocation.maxExecutors infinity`. Le versioni precedenti hanno come impostazione predefinita `100`. Per ulteriori informazioni, vedi [Proprietà del lavoro Spark](#).

EMR Serverless 6.9.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless la versione 6.9.0.

Applicazione	Versione
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

Note di rilascio di EMR Serverless 6.9.0

- L'integrazione di Amazon Redshift per Apache Spark è inclusa in Amazon EMR rilascio 6.9.0 e successivi. In precedenza uno strumento open source, l'integrazione nativa è un connettore Spark che è possibile utilizzare per creare applicazioni Apache Spark in grado di leggere e scrivere dati in Amazon Redshift e Amazon Redshift Serverless. Per ulteriori informazioni, consulta [Utilizzo dell'integrazione di Amazon Redshift per Apache Spark su Amazon EMR Serverless](#).
- La release 6.9.0 di EMR Serverless aggiunge il supporto per l'architettura AWS Graviton2 (arm64). È possibile utilizzare il `architecture` parametro per le API `and` per scegliere l'architettura `arm64` `create-application`. `update-application`. Per ulteriori informazioni, vedi [Opzioni di architettura Serverless Amazon EMR](#).

- Ora puoi esportare, importare, interrogare e unire tabelle Amazon DynamoDB direttamente dalle tue applicazioni EMR Serverless Spark e Hive. Per ulteriori informazioni, vedi [Connessione a DynamoDB con Amazon EMR Serverless](#).

Problemi noti

- Se utilizzi l'integrazione Amazon Redshift per Apache Spark e disponi di un orario, `timetz`, `timestamp` o `timestamptz` con precisione al microsecondo in formato Parquet, il connettore arrotonda i valori temporali al valore in millisecondi più vicino. Come soluzione alternativa, utilizza il parametro `unload_s3_format` del formato di scaricamento del testo.

EMR Serverless 6.8.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless la versione 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 la versione 6.7.0.

Applicazione	Versione
Apache Spark	3.2.1
Apache Hive	3.1.3
Apache Tez	0.9.2

Engine-specific modifiche, miglioramenti e problemi risolti

La tabella seguente elenca una nuova funzionalità specifica del motore.

Modifica	Description
Funzionalità	Tez scheduler ora supporta la priorità dell'attività Tez anziché la priorità del contenitore

EMR Serverless 6.6.0

La tabella seguente elenca le versioni delle applicazioni disponibili con EMR Serverless la versione 6.6.0.

Applicazione	Versione
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

Note di rilascio iniziali di EMR Serverless

- EMR Serverless supporta la classificazione della configurazione Spark. `spark-defaults` Questa classificazione modifica i valori nel file XML di Spark. `spark-defaults.conf` Le classificazioni di configurazione consentono di personalizzare le applicazioni. Per maggiori informazioni, consulta [Configurare le applicazioni](#).
- EMR Serverless supporta le classificazioni `hive-site` di configurazione Hive,, e. `tez-site` `emrfs-site` `core-site` Questa classificazione può modificare rispettivamente i valori nel file di Hive, nel `hive-site.xml` file di Tez, nelle impostazioni EMRFS di Amazon EMR o nel `tez-site.xml` file di Hadoop. `core-site.xml` Le classificazioni di configurazione consentono di personalizzare le applicazioni. [Per ulteriori informazioni, consulta Configurare le applicazioni](#).

Engine-specific modifiche, miglioramenti e problemi risolti

- La tabella seguente elenca i backport di Hive e Tez.

Modifiche a Hive e Tez

Modifica	Description
Backport	TEZ-4430 : è stato risolto il problema relativo alla proprietà <code>tez.task.launch.cmd-opts</code>
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 abbonarti a un feed RSS.

Modifica	Descrizione	Data
Nuova versione GA	Runtime AWS per Apache Spark (emr-spark-8.0.0)	21 maggio 2026
Nuova funzionalità	Esegui sessioni interattive con Spark Connect	23 aprile 2026
Nuova versione di anteprima	Runtime AWS per Apache Spark (emr-spark-8.0-preview)	21 novembre 2025
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.	Aggiunto il nuovo Sid CloudWatchPolicyStatement e EC2PolicyStatement alla AmazonEMRServerlessServiceRolePolicy politica .	25 gennaio 2024
Nuova versione	EMR Serverless 7,0,0	29 dicembre 2023
Nuova versione	EMR Serverless 6,15,0	17 novembre 2023
Nuova funzionalità	Configura più ruoli IAM da assumere quando accedi ai bucket Amazon S3 in account	18 ottobre 2023

	diversi da EMR Serverless (6.11 e versioni successive)	
Nuova versione	EMR Serverless 6.14.0	17 ottobre 2023
Nuova funzionalità	Configurazione predefinita dell'applicazione per EMR Serverless	25 settembre 2023
Aggiornamento alle proprietà Hive predefinite	Sono stati aggiornati i valori predefiniti per <code>hive.driver.disk</code> , <code>hive.tez.disk.size</code> , <code>hive.tez.auto.reducer.parallelism</code> , e le proprietà del lavoro <code>tez.grouping.min-size</code> Hive .	12 settembre 2023
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 ruolo AmazonEMRServerlessServiceRolePolicy <code>_SLR</code> 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.