Guida per l'utente

AWS Glue

AWS Glue: Guida per l'utente

Table of Contents

Che cos'è AWS Glue?

AWS Glue è un servizio di integrazione dati serverless che semplifica agli utenti analitici il rilevamento, la preparazione, lo spostamento e l'integrazione di dati da più origini. Puoi usarlo per analisi, machine learning e sviluppo di applicazioni. Include anche strumenti aggiuntivi di produttività e gestione dei dati per la creazione, l'esecuzione di processi e l'implementazione di flussi di lavoro aziendali.

Con AWS Glue puoi rilevare e collegarti a oltre 70 diverse origini di dati e gestire i tuoi dati in un catalogo dati centralizzato. Puoi creare, eseguire e monitorare visivamente pipeline di estrazione, trasformazione e caricamento (ETL) per caricare dati nei data lake. Inoltre, puoi eseguire ricerche e query immediatamente nei dati catalogati utilizzando Amazon Athena, Amazon EMR e Amazon Redshift Spectrum.

AWS Glue consolida le principali funzionalità di integrazione dei dati in un singolo servizio. Tali funzionalità includono rilevamento dati, ETL moderno, pulizia, trasformazione e catalogazione a livello centralizzato. È anche serverless, per cui non esiste alcuna infrastruttura da gestire. Con un supporto flessibile per tutti i carichi di lavoro come ETL, ELT e streaming in un unico servizio, AWS Glue supporta gli utenti tra vari carichi di lavoro e tipi di utenti.

AWS Glue, inoltre, semplifica l'integrazione dei dati nell'architettura. Si integra con i servizi AWS di analisi e i data lake Amazon S3. AWS Gluedispone di interfacce di integrazione e strumenti per la creazione di lavori facili da usare per tutti gli utenti, dagli sviluppatori agli utenti aziendali, con soluzioni su misura per diverse competenze tecniche.

Grazie alla scalabilità on demand, AWS Glue è utile per concentrarsi su attività di elevato valore che massimizzano il valore dei dati. È scalabile per qualunque dimensione di dati e supporta tutti i tipi di dati e varianti di schemi. Per aumentare l'agilità e ottimizzare i costi, AWS Glue offre disponibilità e fatturazione integrate elevate. pay-as-you-go

Per informazioni sui prezzi, consulta Prezzi di AWS Glue.

AWS Glue Studio

AWS Glue Studio è un'interfaccia grafica che facilita la creazione, l'esecuzione e il monitoraggio di processi di integrazione dati in AWS Glue. Puoi comporre visivamente flussi di lavoro di trasformazione dei dati ed eseguirli con facilità sul motore ETL serverless basato su Apache Spark di AWS Glue.

Con AWS Glue Studio, puoi creare e gestire processi di raccolta, trasformazione e pulizia di dati. Puoi utilizzare AWS Glue Studio anche per risolvere problemi e modificare script di processi.

Argomenti

Caratteristiche di AWS Glue

Le funzioni di AWS Glue si dividono in tre categorie principali:

- Rilevamento e organizzazione dei dati
- Trasformazione, preparazione e pulizia dei dati per l'analisi
- Creazione e monitoraggio di pipeline di dati

Rilevamento e organizzazione dei dati

- Unifica e cerca in più archivi di dati: archivia, indicizza e cerca su più fonti di dati e sink catalogando tutti i tuoi dati. AWS
- Rilevamento automatico dei dati: utilizzo dei crawler AWS Glue per la deduzione automatica delle informazioni degli schemi e l'integrazione di tali informazioni in AWS Glue Data Catalog.
- Gestione di schemi e autorizzazioni: convalida e controllo dell'accesso a database e tabelle.
- Connettiti a un'ampia varietà di fonti di dati: accedi a più fonti di dati, sia in locale che in locale AWS, utilizzando AWS Glue le connessioni per creare il tuo data lake.

Trasformazione, preparazione e pulizia dei dati per l'analisi

- Trasforma visivamente i dati con un' drag-and-dropinterfaccia: definisci il processo ETL nel drag-and-drop Job Editor e genera automaticamente il codice per estrarre, trasformare e caricare i dati.
- Creazione di complesse pipeline ETL con una semplice pianificazione del processo: richiamo di processi AWS Glue in base a un programma, on demand o in base a un evento.

- Pulizia e trasformazione dei dati in streaming in transito: possibilità di consumo dati continuo e pulizia e trasformazione dei dati in transito. In tal modo, i dati sono disponibili per l'analisi in pochi secondi nell'archivio dei dati di destinazione.
- Deduplicazione e pulizia dei dati con machine learning integrato: pulizia e preparazione dei dati per l'analisi senza diventare esperti di machine learning, utilizzando la funzione `FindMatches`. Questa funzione deduplica e trova registri non perfettamente corrispondenti tra loro.
- Notebook di processo integrati: i notebook di processo AWS Glue forniscono notebook serverless con una configurazione minima in AWS Glue, per poter cominciare rapidamente.
- Modifica, debug e verifica del codice ETL: con le sessioni interattive di AWS Glue, puoi esplorare e preparare i dati in modo interattivo. Puoi esplorare, sperimentare ed elaborare i dati in modo interattivo utilizzando l'IDE o il notebook di tua scelta.
- Definizione, rilevamento e correzione di dati sensibili: il rilevamento dei dati sensibili di AWS Glue consente di definire, identificare ed elaborare dati sensibili nella pipeline di dati e nel data lake.

Creazione e monitoraggio di pipeline di dati

- Scalabilità automatica in base al carico di lavoro: aumento o riduzione delle risorse in modo dinamico in base al carico di lavoro. In tal modo, i processi vengono assegnati agli operatori solo quando necessario.
- Automatizzazione di processi con trigger basati su eventi: avvio di crawler o processi AWS Glue con trigger basati su eventi e progettazione di una catena di percorsi e crawler dipendenti.
- Esegui e monitora i processi: esegui i processi AWS Glue con il motore che preferisci, Spark o Ray. Monitorali con strumenti di monitoraggio automatizzati, approfondimenti sull'esecuzione dei processi AWS Glue e AWS CloudTrail. Migliora il monitoraggio dei processi supportati da Spark con l'interfaccia utente di Apache Spark.
- Definizione di flussi di lavoro per attività ETL e di integrazione: definizione di flussi di lavoro per ETL e attività di integrazione per più crawler, processi e trigger.

Scopri di più sulle innovazioni in AWS Glue

Scopri le ultime innovazioni AWS Glue e scopri in che modo i clienti utilizzano AWS Glue per consentire la preparazione dei dati in modalità self-service in tutta l'organizzazione.

Scopri come i clienti AWS Glue vanno oltre la configurazione tradizionale e come si configurano AWS Glue per il monitoraggio del lavoro e delle prestazioni.

Nozioni di base su AWS Glue

Ti consigliamo di iniziare con le sezioni seguenti:

- Panoramica sull'utilizzo di AWS Glue
- Concetti di AWS Glue
- Configurazione di autorizzazioni IAM per AWS Glue
- Nozioni di base su AWS Glue Data Catalog
- Creazione di processi in AWS Glue
- Nozioni di base sulle sessioni interattive di AWS Glue
- Orchestrazione in AWS Glue

Accesso a AWS Glue

Puoi creare, visualizzare e gestire i processi AWS Glue utilizzando una qualunque delle interfacce seguenti:

- Console AWS Glue: fornisce un'interfaccia web per la creazione, la visualizzazione e la gestione di processi AWS Glue. Per accedere alla console, consulta AWS Glue.
- AWS Glue Studio: fornisce un'interfaccia grafica per la creazione e la modifica dei processi AWS Glue in modo visivo. Per ulteriori informazioni, consulta Cos'è AWS Glue Studio.
- AWS Gluesezione della Guida di AWS CLI riferimento: fornisce AWS CLI comandi utilizzabili conAWS Glue. Per ulteriori informazioni, consulta la AWS CLI Documentazione di riferimento per AWS Glue.
- AWS GlueAPI: fornisce una documentazione di riferimento dell'API completa per gli sviluppatori. Per ulteriori informazioni, consulta API AWS Glue.

Servizi correlati

Gli utenti di AWS Glue utilizzano anche:

- AWS Lake Formation : un servizio costituito da un livello di autorizzazione che fornisce un controllo granulare fine dell'accesso alle risorse in AWS Glue Data Catalog.
- AWS GlueAWS Glue DataBrew— Uno strumento visivo di preparazione dei dati che è possibile utilizzare per pulire e normalizzare i dati senza scrivere alcun codice.

AWS Glue: Come funziona

AWS Glue utilizza altri servizi AWS per orchestrare i processi di estrazione, trasformazione e caricamento (ETL) per creare data warehouse e data lake e generare flussi di output. AWS Glue richiama le operazioni API per trasformare i dati, creare log di runtime, archiviare la logica dei processi e creare notifiche che aiutano a monitorare l'esecuzione dei processi. La console AWS Glue si connette a questi servizi in un'applicazione gestita, in modo che tu possa concentrarti sulla creazione e sul monitoraggio del lavoro ETL. La console esegue le operazioni amministrative e di sviluppo del processo per tuo conto. Devi fornire le credenziali e altre proprietà a AWS Glue per accedere alle origini dati e scrivere nelle destinazioni dati.

AWS Glue si occupa di effettuare il provisioning delle risorse necessarie per l'esecuzione del carico di lavoro e di gestire tali risorse. Non devi creare l'infrastruttura per uno strumento ETL, perché l'operazione viene eseguita da AWS Glue. Quando sono necessarie risorse, per ridurre i tempi di avvio, AWS Glue usa un'istanza del pool di istanze attivo per eseguire il carico di lavoro.

Con AWS Glue crei i processi usando le definizioni di tabella nel catalogo dati. I processi consistono in script che contengono la logica di programmazione che esegue la trasformazione. Per avviare i processi, in base a una pianificazione o come risultato di un evento specificato, potrai utilizzare i trigger. Puoi decidere dove conservare i dati dell'obiettivo e quale origine dati popola l'obiettivo. Con il tuo input, AWS Glue genera il codice necessario per trasformare i dati dall'origine alla destinazione. Puoi anche fornire script nella console AWS Glue o usare l'API per elaborare i dati.

Origini dati e destinazioni

AWS Glue per Spark consente di leggere e scrivere dati da più sistemi e database, tra cui:

- Amazon S3
- Amazon DynamoDB
- Amazon Redshift
- Amazon Relational Database Service (Amazon RDS)
- Database accessibili da JDBC di terze parti
- MongoDB e Amazon DocumentDB (compatibile con MongoDB)
- Altri connettori del marketplace e plug-in Apache Spark

Flussi di dati

AWS Glue per Spark può trasmettere dati dai seguenti sistemi:

- Flusso di dati Amazon Kinesis
- Apache Kafka

AWS Glue è disponibile in diverse regioni AWS. Per ulteriori informazioni, consulta la sezione relativa a regioni ed endpoint AWS nella Riferimenti generali di Amazon Web Services.

Argomenti

Processi ETL serverless eseguiti in isolamento

AWS Glue esegue i processi ETL in un ambiente serverless con un motore a scelta tra Spark e Ray. AWS Glue esegue questi processi su risorse virtuali di cui effettua il provisioning e che gestisce nel proprio account di servizio.

AWS Glue ha gli scopi seguenti:

- Isolare i dati dei clienti.
- Proteggere i dati dei clienti in transito e quelli memorizzati.
- Accedere ai dati dei clienti solo in risposta alle richieste dei clienti, utilizzando le credenziali contestuali e temporanee o con il consenso del cliente ai ruoli IAM nel suo account.

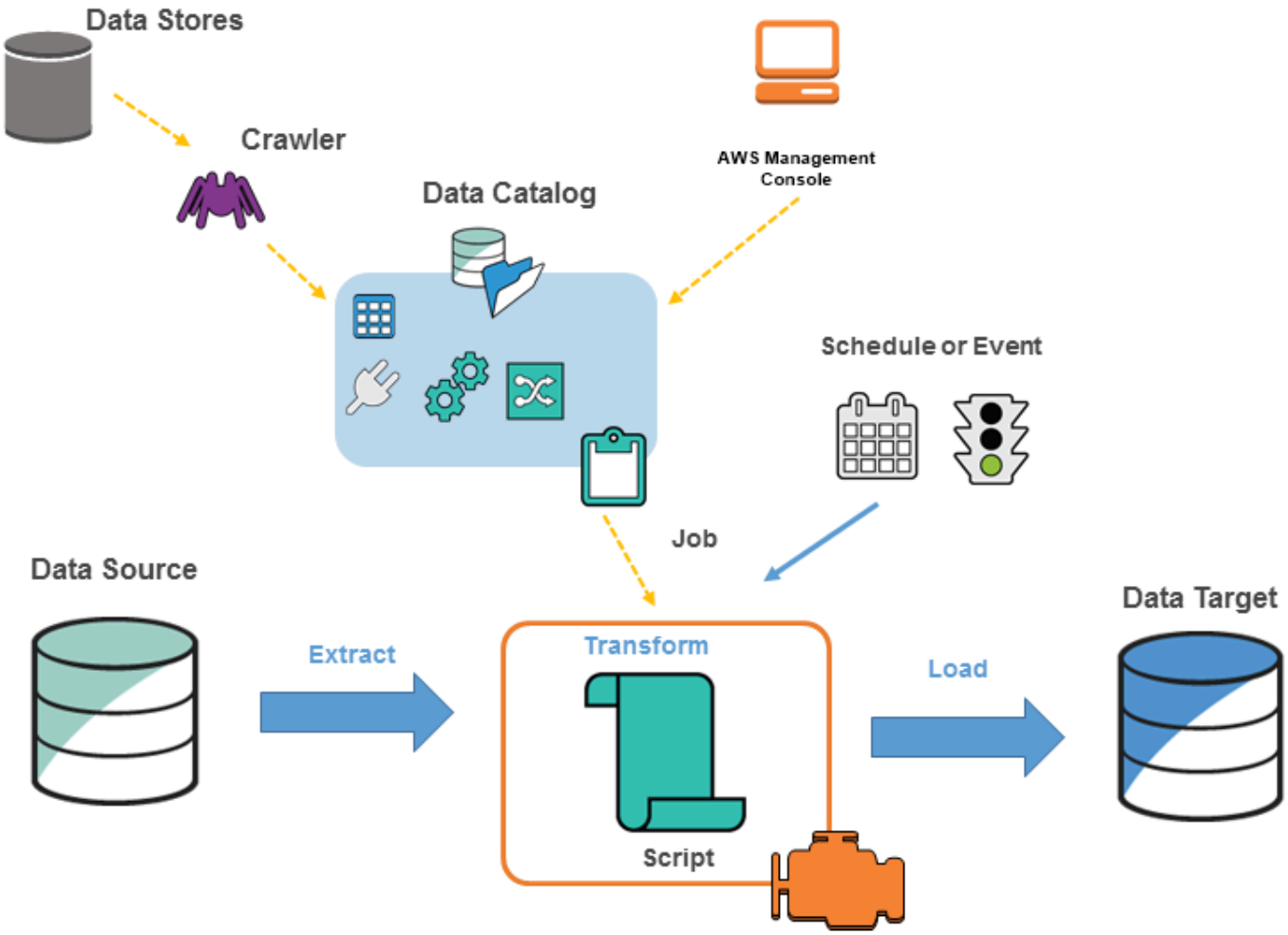
Durante il provisioning di un processo ETL, fornisci origini dati di input e destinazioni dati di output nel Virtual Private Cloud (VPC). Inoltre, puoi fornire il ruolo IAM, l'ID VPC, l'ID sottorete e il gruppo di sicurezza che sono necessari per accedere alle origini dati e alle destinazioni. Per ogni tupla (ID account del cliente, ruolo IAM, ID di sottorete e gruppo di sicurezza), AWS Glue crea un nuovo ambiente isolato a livello di rete e di gestione da tutti gli altri ambienti all'interno dell'account di servizio AWS Glue.

AWS Glue crea interfacce di rete elastiche nella sottorete usando indirizzi IP privati. I processi utilizzano queste interfacce di rete elastiche per accedere alle origini dati e alle destinazioni dati. Il traffico in uscita e all'interno dell'ambiente di esecuzione del processo è regolato dal VPC e dalle policy di rete con un'eccezione: le chiamate effettuate alle librerie AWS Glue possono indirizzare il traffico verso operazioni API AWS Glue tramite il VPC AWS Glue. Tutte le chiamate API AWS Glue vengono registrate, pertanto i proprietari dei dati possono controllare l'accesso API abilitando AWS CloudTrail, che fornisce i log di controllo all'account.

Gli ambienti gestiti da AWS Glue che eseguono i processi ETL sono protetti con le stesse prassi di sicurezza seguite da altri servizi AWS. Per una panoramica delle procedure e delle responsabilità di sicurezza condivise, consulta il whitepaper sull'introduzione ai processi di sicurezza di AWS.

Concetti AWS Glue

Il diagramma seguente illustra l'architettura di un ambiente AWS Glue.



È possibile definire processi in AWS Glue per svolgere il lavoro necessario per estrarre, trasformare e caricare (ETL) i dati da un'origine dati a una destinazione. In genere, si svolgono le azioni seguenti:

- Per le origini dei datastore, si definisce un crawler per popolare il AWS Glue Data Catalog con definizioni di tabelle di metadati. Si punta il crawler a un datastore e il crawler crea le definizioni di tabelle nel catalogo dati. Per le origini di streaming, è possibile definire manualmente le tabelle del catalogo dati e specificare le proprietà del flusso dei dati.

  Oltre alle definizioni di tabelle, il AWS Glue Data Catalog contiene altri metadati necessari per definire i processi ETL. Si utilizzano questi metadati quando si definisce un processo per trasformare i dati.
- AWS Glue è in grado di generare uno script per trasformare i dati. In alternativa, è possibile fornire lo script nella console o nell'API AWS Glue.

- È possibile eseguire il processo on demand oppure configurarlo affinché si avvii al verificarsi di un trigger specifico. Il trigger può essere una pianificazione basata sul tempo o un evento.

  Durante l'esecuzione del processo, uno script estrae i dati dall'origine dati, li trasforma e li carica sulla destinazioni dati. Lo script viene eseguito in un ambiente Apache Spark in AWS Glue.

> **Important**
>
> Le tabelle e i database in AWS Glue sono oggetti nel AWS Glue Data Catalog. Essi contengono metadati, non dati provenienti da un datastore.

I dati basati su testo, come i CSV, devono essere codificati in **`UTF-8`** affinché AWS Glue li possa elaborare correttamente. Per ulteriori informazioni, consulta UTF-8 in Wikipedia.

Terminologia di AWS Glue

AWS Glue si basa sull'interazione di diversi componenti per creare e gestire il flusso di lavoro di estrazione, trasformazione e caricamento (ETL).

AWS Glue Data Catalog

Archivio di metadati persistente in AWS Glue. Il catalogo contiene definizioni di tabelle, definizioni di processi e altre informazioni di controllo per la gestione dell'ambiente AWS Glue. Ogni account AWS dispone di un AWS Glue Data Catalog per regione.

Classificatore

Determina lo schema dei dati. AWS Glue fornisce classificatori per i tipi di file più comuni, ad esempio CSV, JSON, XML, AVRO e molti altri. Fornisce inoltre classificatori per i più comuni sistemi di gestione di database relazionali utilizzando una connessione JDBC. Puoi scrivere un classificatore personalizzato usando un pattern grok o specificando un tag di riga in un documento XML.

Connessione

Un oggetto del catalogo dati che contiene le proprietà necessarie per connettersi a un particolare archivio dati.

Crawler

Programma che si connette a un datastore (di origine o di destinazione), avanza attraverso un elenco di classificatori ordinato per priorità per determinare lo schema dei dati e quindi crea tabelle di metadati nel AWS Glue Data Catalog.

Database

Set di definizioni di tabelle del catalogo dati associate organizzate in un gruppo logico.

Datastore, origine dati, target dati

Un datastore è un repository per archiviare i dati in modo permanente. Alcuni esempi includono bucket Amazon S3 e database relazionali. Un'origine dati è un datastore utilizzato come input per un processo o una trasformazione. Un target dati è un datastore su cui scrive un processo o una trasformazione.

Endpoint di sviluppo

Ambiente che è possibile usare per sviluppare e testare gli script ETL AWS Glue.

Frame dinamico

Una tabella distribuita che supporta dati nidificati come strutture e array. Ogni record è auto descrittivo, progettato per la flessibilità dello schema con dati semi-strutturati. Ogni record contiene sia i dati, sia lo schema che li descrive. Negli script ETL è possibile utilizzare sia frame dinamici che DataFrame Apache Spark ed eseguire conversioni tra di essi. I frame dinamici forniscono una serie di trasformazioni avanzate per la pulizia dei dati e l'ETL.

Processo

Logica di business necessaria per eseguire il lavoro ETL. È composto da uno script di trasformazione, da origini dati e da destinazioni dati. Le esecuzioni dei processi sono avviate tramite trigger pianificabili o attivabili tramite eventi.

Pannello di controllo per le prestazioni del processo

AWS Glue fornisce un pannello di controllo completo per l'esecuzione dei processi ETL. Nel pannello di controllo vengono visualizzate informazioni sulle esecuzioni dei processi in un intervallo di tempo specifico.

Interfaccia notebook

Un'esperienza notebook migliorata con una configurazione in un solo clic per semplificare la creazione di processi e l'esplorazione dei dati. Il notebook e le connessioni vengono configurati automaticamente per te. È possibile utilizzare l'interfaccia notebook basata sul notebook Jupyter per sviluppare, eseguire il debug e implementare in modo interattivo script e flussi di lavoro utilizzando l'infrastruttura serverless Apache Spark ETL AWS Glue. Nell'ambiente notebook, è possibile anche eseguire query ad hoc, analisi dei dati e visualizzazione (ad esempio tabelle e grafici).

Script

Codice che estrae i dati dalle origini, li trasforma e li carica nelle destinazioni. AWS Glue genera script PySpark o Scala.

Tabella

Definizione di metadati che rappresenta i tuoi dati. Sia che i dati siano in un file Amazon Simple Storage Service (Amazon S3), in una tabella Amazon Relational Database Service (Amazon RDS) o in un altro set di dati, una tabella definisce lo schema dei dati. Una tabella nel AWS Glue Data Catalog contiene i nomi delle colonne, le definizioni dei tipi di dati, le informazioni sulle partizioni e altri metadati su un set di dati di base. Lo schema dei dati è rappresentato nella definizione di tabella AWS Glue. I dati effettivi rimangono nel datastore originale, sia che si trovino in un file o in una tabella di un database relazionale. AWS Glue cataloga i file e le tabelle di database relazionali nel AWS Glue Data Catalog. Questi fungono da origini e destinazioni quando si crea un processo ETL.

Trasformazione

Logica di codice utilizzata per modificare i dati in un formato diverso.

Trigger

Avvia un processo ETL. È possibile definire i trigger sulla base di un orario programmato o di un evento.

Editor visivo dei processi

L'editor visivo del processo è un'interfaccia grafica che consente di creare, eseguire e monitorare in modo semplice processi di estrazione, trasformazione e caricamento (ETL) in AWS Glue. È possibile comporre visivamente flussi di lavoro per la trasformazione dei dati ed eseguirli senza problemi sul motore ETL serverless basato su Apache Spark di AWS Glue, con la possibilità di ispezionare lo schema e i risultati dei dati in ogni fase del processo.

Worker

Con AWS Glue, paghi solo per il tempo di esecuzione del processo ETL. Non ci sono risorse da gestire, quindi non ti vengono addebitati costi anticipati e tariffe per il tempo di avvio o di arresto. Viene addebitata una tariffa oraria calcolata in base al numero di Unità di elaborazione dati (o DPU, Data Processing Units) utilizzate per eseguire il processo ETL. Una singola unità di elaborazione dati (DPU) è anche definita come un dipendente. AWS Glue viene fornito con tre tipi di dipendente, per aiutarti a selezionare la configurazione che soddisfa i requisiti di latenza del processo e quelli di costo. I worker sono disponibili nelle configurazioni Standard, G.1X, G.2X, e G.025X.

AWS Glue componenti

AWS Glue fornisce una console e operazioni API per configurare e gestire il carico di lavoro di estrazione, trasformazione e caricamento (ETL). Puoi usare le operazioni API tramite vari SDK specifici dei linguaggi e nell'AWS Command Line Interface (AWS CLI). Per informazioni sull'utilizzo di AWS CLI, consulta Riferimento ai comandi AWS CLI.

AWS Glue usa il AWS Glue Data Catalog per archiviare i metadati su origini dati, trasformazioni e destinazioni. Il catalogo dati sostituisce il metastore Apache Hive. Il AWS Glue Jobs system fornisce un'infrastruttura gestita per la definizione, la pianificazione e l'esecuzione di operazioni ETL sui dati. Per ulteriori informazioni sull'API AWS Glue, consulta API AWS Glue.

Console AWS Glue

Puoi usare la console AWS Glue per definire e orchestrare il flusso di lavoro ETL. La console chiama diverse operazioni API nel AWS Glue Data Catalog e nel AWS Glue Jobs system per eseguire le attività seguenti:

- Definire gli oggetti AWS Glue, come processi, tabelle, crawler e connessioni.
- Pianificare l'esecuzione dei crawler.
- Definire eventi o programmi per i trigger di processo.
- Cercare e filtrare elenchi di oggetti AWS Glue.
- Modificare gli script di trasformazione.

AWS Glue Data Catalog

La AWS Glue Data Catalog è il tuo archivio di metadati tecnico persistente in AWS Cloud.

Ogni account AWS dispone di un AWS Glue Data Catalog per regione AWS. Ogni catalogo dati è una raccolta altamente scalabile di tabelle organizzate in database. Una tabella è rappresentazione dei metadati di una raccolta di dati strutturati o semi-strutturati archiviati in origini come Amazon RDS, Apache Hadoop Distributed File System, Amazon OpenSearch Service e altre. La AWS Glue Data Catalog fornisce un repository uniforme in cui sistemi diversi possono archiviare e trovare metadati per tenere traccia dei dati in silo di dati. È quindi possibile utilizzare i metadati per eseguire query e trasformare i dati in modo coerente su un'ampia varietà di applicazioni.

Utilizzi il catalogo dati insieme alle policy AWS Identity and Access Management e Lake Formation per controllare l'accesso alle tabelle e ai database. In questo modo, consenti a diversi gruppi nella tua azienda di pubblicare in modo sicuro i dati per la più ampia organizzazione proteggendo allo stesso tempo le informazioni sensibili in modo altamente granulare.

Il catalogo dati, insieme a CloudTrail e Lake Formation, fornisce inoltre funzionalità di verifica e governance complete, con rilevamento delle modifiche dello schema e controlli dell'accesso ai dati. Questo contribuisce a garantire che i dati non vengono modificati impropriamente o condivisi inavvertitamente.

Per informazioni su come proteggere e controllare ila AWS Glue Data Catalog, consulta:

- AWS Lake Formation – Per ulteriori informazioni, consulta Cos'è AWS Lake Formation? nella Guida per gli sviluppatori di AWS Lake Formation.
- CloudTrail – Per ulteriori informazioni, consulta Che cos'è CloudTrail? nella Guida per l'utente di AWS CloudTrail.

Di seguito sono riportati altri servizi AWS e progetti open source che utilizzano AWS Glue Data Catalog:

- Amazon Athena – Per ulteriori informazioni, consulta Comprensione di tabelle, database e catalogo dati nella Guida per l'utente di Amazon Athena.
- Amazon Redshift Spectrum – Per ulteriori informazioni, consulta Utilizzo di Amazon Redshift Spectrum per eseguire query su dati esterni nella Guida per gli sviluppatori di Amazon Redshift.
- Amazon EMR – Per ulteriori informazioni, consulta Utilizzo di policy basate su risorse per l'accesso Amazon EMR ad AWS Glue Data Catalog nella Guida alla gestione di Amazon EMR.
- Client di AWS Glue Data Catalog per Apache Hive Metastore – Per ulteriori informazioni su questo progetto GitHub, consulta l'argomento relativo al client di AWS Glue Data Catalog per Metastore Apache Hive.

Crawler e classificatori di AWS Glue

AWS Glue permette inoltre di configurare i crawler che possono effettuare la scansione dei dati in tutti i tipi di repository, classificarli, estrarne informazioni sullo schema e archiviare i metadati automaticamente nel AWS Glue Data Catalog. AWS Glue Data Catalog può essere utilizzato per guidare le operazioni ETL.

Per informazioni su come configurare i crawler e i classificatori, consulta l'articolo Definizione di crawler in AWS Glue. Per informazioni su come programmare i crawler e i classificatori usando l'API AWS Glue, consulta API crawler e classificatori.

Operazioni ETL di AWS Glue

Usando i metadati nel catalogo dati, AWS Glue è in grado di generare automaticamente gli script Scala o PySpark (API Python per Apache Spark) con estensioni AWS Glue che puoi usare e modificare per eseguire diverse operazioni ETL. Ad esempio, puoi estrarre, pulire e trasformare dati grezzi, quindi memorizzare il risultato in un diverso archivio, dove può essere interrogato e analizzato. Tale script potrebbe convertire un file CSV in una struttura dati relazionale e salvarlo in Amazon Redshift.

Per ulteriori informazioni su come usare le funzionalità ETL di AWS Glue, consulta Script di programmazione Spark.

Streaming ETL in AWS Glue

AWS Glue consente di eseguire operazioni ETL sui dati di streaming utilizzando processi in esecuzione continua. AWS Glue Streaming ETL è basato sul motore Apache Spark Structured Streaming e può importare flussi da Amazon Kinesis Data Streams, Apache Kafka e Amazon Managed Streaming for Apache Kafka (Amazon MSK). Streaming ETL può pulire e trasformare i dati di streaming e caricarli in Amazon S3 o in archivi dati JDBC. Usa Streaming ETL in AWS Glue per elaborare i dati degli eventi come flussi IoT, clickstream e registri di rete.

Se si conosce lo schema dell'origine dati di streaming, è possibile specificarlo in una tabella del catalogo dati. In caso contrario, è possibile abilitare il rilevamento dello schema nel processo ETL di streaming. Il processo determina automaticamente lo schema dai dati in entrata.

Il processo ETL può utilizzare le trasformazioni integrate di AWS Glue e le trasformazioni native di Apache Spark Structured Streaming. Per ulteriori informazioni, consulta Operazioni sullo streaming di DataFrames/DataSet sul sito web di Apache Spark.

Per ulteriori informazioni, consulta the section called “Aggiunta di processi di streaming ETL”.

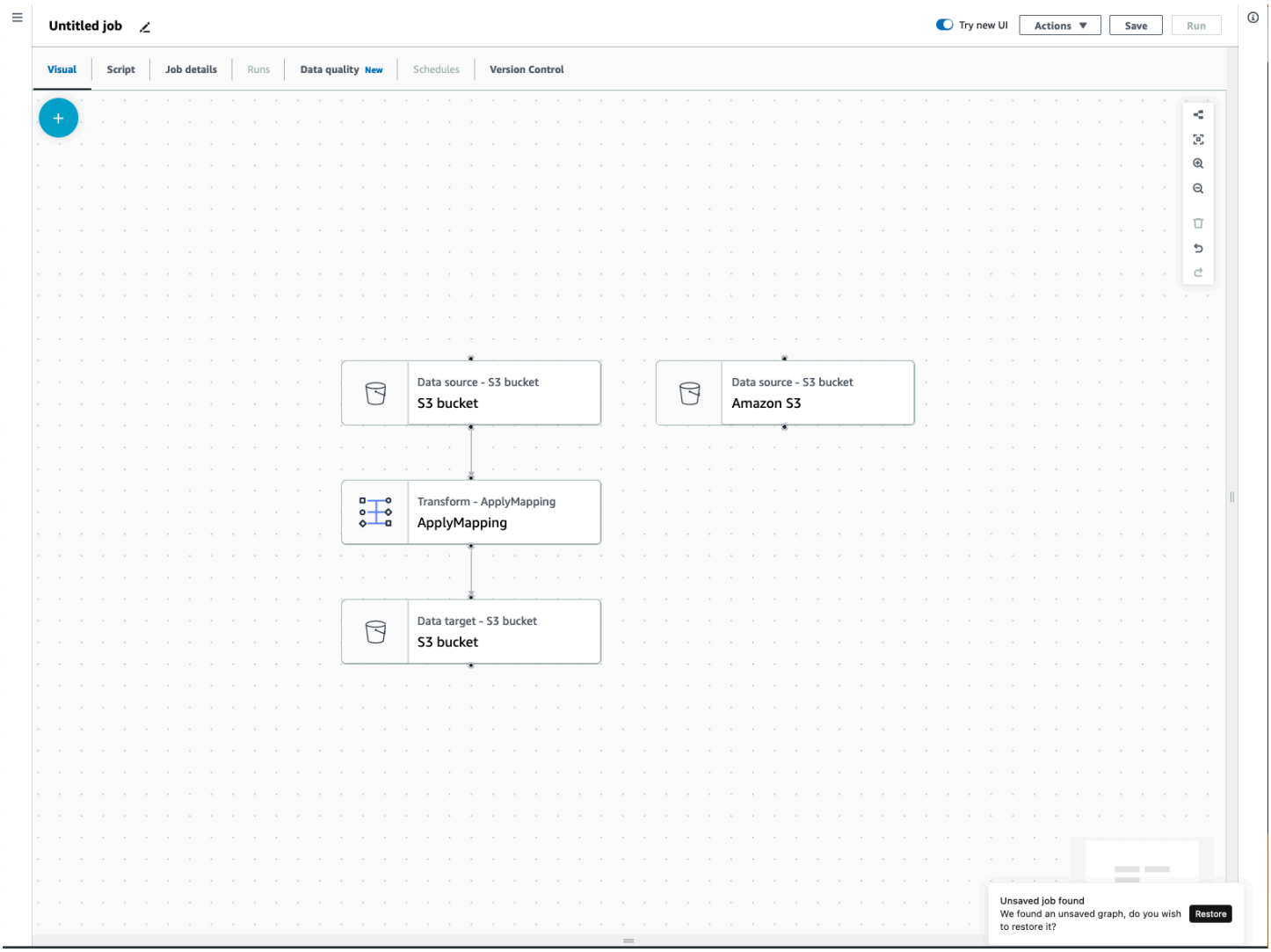
Il sistema di processi AWS Glue

Il AWS Glue Jobs system fornisce un'infrastruttura gestita per orchestrare il flusso di lavoro ETL. Puoi creare processi in AWS Glue che automatizzano gli script usati per estrarre, trasformare e trasferire dati in posizioni diverse. I processi possono essere programmati e concatenati oppure possono essere attivati da eventi quali l'arrivo di nuovi dati.

Per ulteriori informazioni sull'uso di AWS Glue Jobs system, consulta Monitoraggio di AWS Glue. Per informazioni sulla programmazione tramite l'API AWS Glue Jobs system, consulta API dei processi.

Componenti ETL visivi

AWS Glue consente di creare processi ETL attraverso un canvas visivo che è possibile manipolare.



Menu dei processi ETL

Le opzioni di menu nella parte superiore del canvas consentono di accedere alle varie visualizzazioni e ai dettagli di configurazione relativi al processo.

- Visivo: il canvas dell'editor di processo visivo. Da qui è possibile aggiungere nodi per creare un processo.
- Script: la rappresentazione in script del tuo processo ETL. AWS Glue genera lo script in base alla rappresentazione visiva del processo. È inoltre possibile modificare lo script o scaricarlo.

> **Note**
>
> Se scegli di modificare lo script, l'esperienza di creazione del processo viene convertita in modo permanente in modalità di solo script. Successivamente, non è più possibile utilizzare l'editor visivo per modificare il processo. È necessario aggiungere tutte le origini, le trasformazioni e le destinazioni del processo e apportare tutte le modifiche necessarie con l'editor visivo prima di scegliere di modificare lo script.

- Dettagli del processo: la scheda Dettagli del processo consente di configurare il processo impostandone le proprietà. Sono disponibili proprietà di base, come nome e descrizione del processo, ruolo IAM, tipo di processo, versione di AWS Glue, lingua, tipo di worker, numero di worker, segnalibro del processo, esecuzione flessibile, numero di ritirati e timeout del processo, così come alcune proprietà avanzate, come connessioni, librerie, parametri di processo e tag.
- Esecuzioni: dopo l'esecuzione del processo, è possibile accedere a questa scheda per visualizzare i processi eseguiti in passato.
- Qualità dei dati: la qualità dei dati consente di valutare e monitorare la qualità delle risorse di dati. Puoi saperne di più su come utilizzare la qualità dei dati in questa scheda e aggiungere una trasformazione della qualità dei dati al tuo processo.
- Pianificazioni: i processi che hai pianificato vengono visualizzati in questa scheda. Se non esistono pianificazioni collegate a questo processo, questa scheda non è accessibile.
- Controllo della versione: puoi utilizzare Git con il tuo processo configurandolo in un repository Git.

Pannelli ETL visivi

Quando lavori nel canvas, sono disponibili diversi pannelli che ti aiutano a configurare i nodi o a visualizzare l'anteprima dei dati e visualizzare lo schema di output.

- Proprietà: il pannello Proprietà viene visualizzato quando si sceglie un nodo nel canvas.
- Anteprima dei dati: il pannello di anteprima dei dati fornisce un'anteprima dell'output dei dati in modo da poter prendere decisioni prima di eseguire il processo ed esaminare l'output.
- Schema di output: la scheda Schema di output consente di visualizzare e modificare lo schema dei nodi di trasformazione.

Ridimensionamento dei pannelli

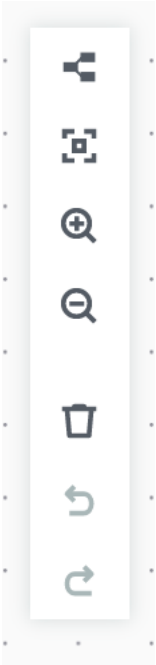
È possibile ridimensionare il pannello Proprietà sul lato destro dello schermo e il pannello inferiore che contiene le schede Anteprima dati e Schema di output facendo clic sul bordo del pannello e trascinandolo a sinistra e a destra o su e giù.

- Pannello delle proprietà: ridimensiona il pannello delle proprietà facendo clic sul bordo del canvas sul lato destro dello schermo e trascinandolo verso sinistra per aumentarne la larghezza. Per impostazione predefinita, il pannello è compresso, mentre quando viene selezionato un nodo il pannello delle proprietà si apre alla dimensione predefinita.
- Pannello Anteprima dei dati e Schema di output: ridimensiona il pannello inferiore facendo clic sul bordo inferiore del canvas nella parte inferiore dello schermo e trascinandolo verso l'alto per aumentarne l'altezza. Per impostazione predefinita, il pannello è compresso, mentre quando viene selezionato un nodo il pannello inferiore si apre alla dimensione predefinita.

Canvas del processo

È possibile aggiungere, rimuovere e spostare/riordinare i nodi direttamente sul canvas visivo ETL. Puoi immaginarlo come uno spazio di lavoro per creare un processo ETL completamente funzionale, a partire da un'origine dati fino alla destinazione dati.

Quando lavori con i nodi sul canvas, hai a disposizione una barra degli strumenti che può aiutarti a ingrandire e ridurre le dimensioni, rimuovere nodi, creare o modificare connessioni tra i nodi, cambiare l'orientamento del flusso di processo e annullare o ripetere un'operazione.



La barra degli strumenti mobile è ancorata al bordo in alto a destra del canvas e contiene diverse immagini che eseguono altrettante operazioni:

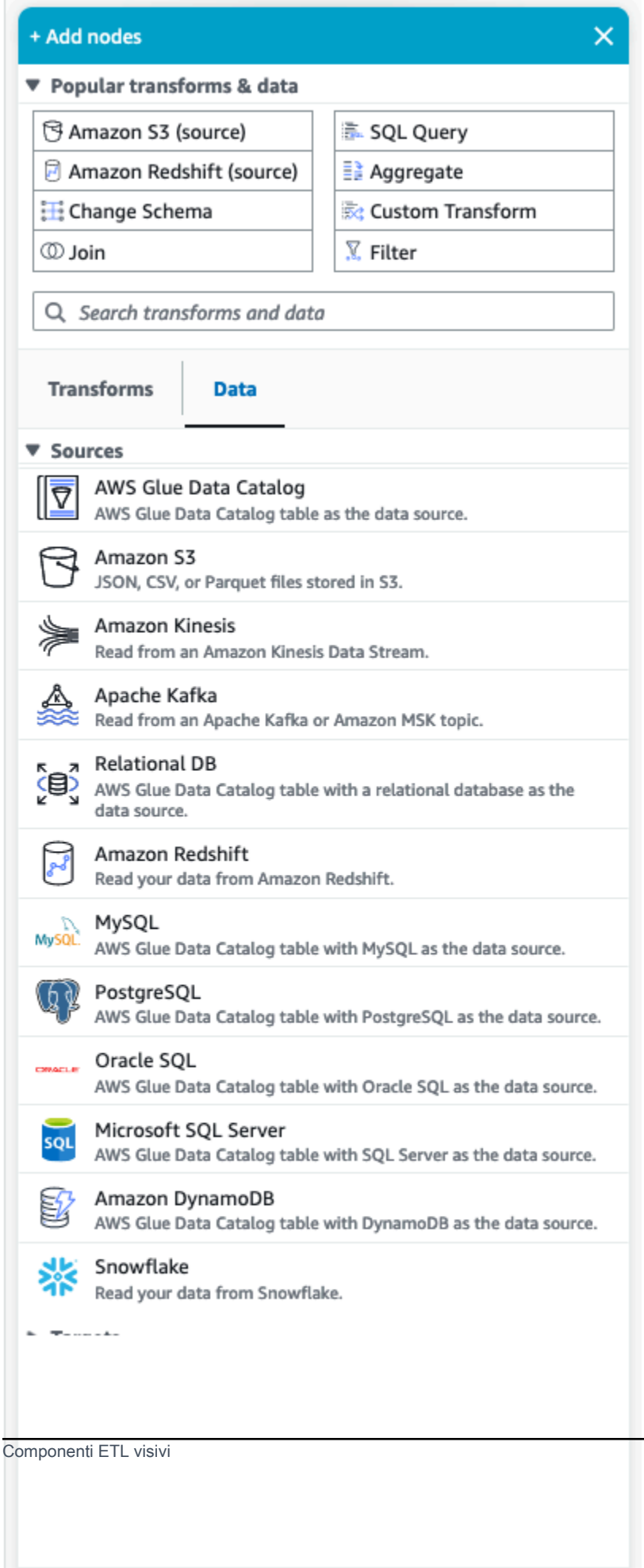
- Icona del layout: la prima icona nella barra degli strumenti è l'icona del layout. Per impostazione predefinita, la direzione dei processi visivi è dall'alto verso il basso. Riorganizza la direzione del processo visivo disponendo i nodi orizzontalmente da sinistra a destra. Facendo nuovamente clic sull'icona del layout, la direzione torna dall'alto verso il basso.
- Icona Ricentra: questa icona consente di modificare la visualizzazione del canvas centrandola. È possibile utilizzarla con processi di grandi dimensioni per tornare alla posizione centrale.
- Icona Ingrandisci: questa icona consente di aumentare la dimensione dei nodi sul canvas.
- Icona Riduci: questa icona consente di ridurre la dimensione dei nodi sul canvas.
- Icona del cestino: l'icona del cestino rimuove un nodo dal processo visivo. Prima è necessario selezionare un nodo.
- Icona Annulla: questa icona consente di annullare l'ultima operazione eseguita sul processo visivo.
- Icona Ripeti: questa icona consente di ripetere l'ultima operazione eseguita sul processo visivo.

Utilizzo della minimappa

Pannello delle risorse

Il pannello delle risorse contiene tutte le origini dati, le operazioni di trasformazione e le connessioni disponibili. Apri il pannello delle risorse sul canvas facendo clic sull'icona "+". Si aprirà il pannello delle risorse.

Per chiudere il pannello delle risorse, fai clic sulla X nell'angolo in alto a destra del pannello delle risorse. In questo modo il pannello rimarrà nascosto fino a quando non lo riaprirai.


+ Add nodes
Popular transforms & data
Amazon S3 (source)
Amazon Redshift (source)
Change Schema
Join
SQL Query
Aggregate
Custom Transform
Filter
Search transforms and data
Transforms
Data
Sources
AWS Glue Data Catalog
AWS Glue Data Catalog table as the data source.
Amazon S3
JSON, CSV, or Parquet files stored in S3.
Amazon Kinesis
Read from an Amazon Kinesis Data Stream.
Apache Kafka
Read from an Apache Kafka or Amazon MSK topic.
Relational DB
AWS Glue Data Catalog table with a relational database as the data source.
Amazon Redshift
Read your data from Amazon Redshift.
MySQL
AWS Glue Data Catalog table with MySQL as the data source.
PostgreSQL
AWS Glue Data Catalog table with PostgreSQL as the data source.
Oracle SQL
AWS Glue Data Catalog table with Oracle SQL as the data source.
Microsoft SQL Server
AWS Glue Data Catalog table with SQL Server as the data source.
Amazon DynamoDB
AWS Glue Data Catalog table with DynamoDB as the data source.
Snowflake
Read your data from Snowflake.

Trasformazioni e dati comuni

Nella parte superiore del pannello è presente una raccolta di Trasformazioni e dati comuni. Questi nodi vengono comunemente utilizzati in AWS Glue. Scegline uno per aggiungerlo al canvas. Puoi anche nascondere Trasformazioni e dati comuni facendo clic sul triangolo accanto all'intestazione Trasformazioni e dati comuni.

Nella sezione Trasformazioni e dati comuni, puoi cercare trasformazioni e nodi di origini dati. I risultati vengono visualizzati durante la digitazione. Più lettere aggiungi alla tua query di ricerca, più l'elenco dei risultati si ridurrà. I risultati della ricerca vengono compilati in base al nome e/o alla descrizione del nodo. Scegli il nodo per aggiungerlo al canvas.

Trasformazioni e dati

Esistono due schede che organizzano i nodi in Trasformazioni e Dati.

Trasformazioni: quando si sceglie la scheda Trasformazioni, è possibile selezionare tutte le trasformazioni disponibili. Scegli una trasformazione per aggiungerla al canvas. Puoi anche scegliere Aggiungi trasformazione nella parte inferiore dell'elenco Trasformazioni; questa operazione aprirà una nuova pagina alla documentazione per la creazione di Trasformazioni visive personalizzate. Seguendo i passaggi potrai creare trasformazioni personalizzate. Le trasformazioni verranno quindi visualizzate nell'elenco delle trasformazioni disponibili.

Dati: la scheda dati contiene tutti i nodi per Origini e Destinazioni. È possibile nascondere le origini e le destinazioni facendo clic sul triangolo accanto all'intestazione Origini o Destinazioni. È possibile visualizzare le origini e le destinazioni facendo nuovamente clic sul triangolo. Scegli un nodo di origine o di destinazione per aggiungerlo al canvas. È inoltre possibile scegliere Gestisci connessioni per aggiungere una nuova connessione. Si aprirà la pagina Connettori nella console.

AWS Glue per Spark e AWS Glue per Ray

In AWS Glue su Apache Spark (ETL AWS Glue), è possibile usare PySpark per scrivere codice Python per gestire i dati su larga scala. Spark è una soluzione comune per questo problema, ma i data engineer con background incentrati su Python possono trovare la transizione poco intuitiva. Il modello Spark DataFrame non è perfettamente adatto a Python, il che riflette il linguaggio Scala e il runtime Java su cui è basato.

In AWS Glue, è possibile utilizzare i processi della shell (interprete di comandi) di Python per eseguire integrazioni di dati Python native. Questi processi vengono eseguiti su una singola istanza

Amazon EC2 e sono limitati dalla capacità di tale istanza. Ciò limita la velocità di trasmissione effettiva dei dati che è possibile elaborare e diventa costoso da mantenere quando si tratta di Big Data.

AWS Glue per Ray consente di aumentare i carichi di lavoro di Python senza investimenti sostanziali nell'apprendimento di Spark. È possibile sfruttare alcuni scenari in cui Ray si comporta meglio. Offrendoti una scelta, puoi utilizzare i punti di forza di Spark e Ray in base ai casi.

AWS Glue ETL e AWS Glue per Ray sono diversi alla base, quindi supportano funzionalità diverse. Controlla le documentazione per determinare le funzionalità supportate.

Cos'è AWS Glue per Ray?

Ray è un framework di calcolo distribuito open source che può essere utilizzato per scalare i carichi di lavoro, con particolare attenzione a Python. Per ulteriori informazioni su Ray, consulta il sito Web di Ray. AWS Glue I processi Ray e le sessioni interattive consentono di utilizzare Ray all'interno di AWS Glue.

AWS Glue per Ray può essere utilizzato per scrivere script Python per calcoli che verranno eseguiti in parallelo su più macchine. Nei processi e nelle sessioni interattive di Ray, è possibile utilizzare le librerie Python comuni come pandas per facilitare la scrittura e l'esecuzione dei flussi di lavoro. Per ulteriori informazioni sui set di dati di Ray, consulta Set di dati di Ray nella documentazione di Ray. Per ulteriori informazioni su Pandas, consulta il sito Web di Pandas.

Quando si utilizza AWS Glue per Ray, è possibile eseguire i flussi di lavoro di pandas sui big data su scala aziendale con poche righe di codice. È possibile creare un processo Ray dalla console AWS Glue o con l'SDK AWS. Per eseguire il codice su un ambiente Ray serverless, è possibile anche aprire una sessione interattiva AWS Glue. I processi visivi in AWS Glue Studio non sono ancora supportati.

I processi AWS Glue per Ray consentono di eseguire uno script in base a una pianificazione o in risposta a un evento di Amazon EventBridge. I processi archiviano le informazioni dei log e le statistiche di monitoraggio in CloudWatch che ti consentono di comprendere l'integrità e l'affidabilità del tuo script. Per ulteriori informazioni sull'utilizzo del sistema dei processi AWS Glue, consulta the section called “Utilizzo dei processi Ray”.

Le sessioni interattive di AWS Glue per Ray (anteprima) consentono di eseguire frammenti di codice uno dopo l'altro sulle stesse risorse in provisioning. Questa funzionalità può essere utilizzata per creare prototipi e sviluppare script in modo efficiente o creare le proprie applicazioni interattive. È possibile utilizzare le sessioni interattive AWS Glue dai notebook AWS Glue Studio nella AWS

Management Console. Per ulteriori informazioni, consulta Utilizzare Notebooks with AWS Glue Studio e AWS Glue Puoi anche usarle tramite un kernel Jupyter, che consente di eseguire le sessioni interattive da strumenti di modifica del codice esistenti che supportano i notebook Jupyter, come VSCode. Per ulteriori informazioni, consulta the section called “AWS Glue per le sessioni interattive di Ray (anteprima)”.

Ray automatizza il lavoro di dimensionamento del codice Python distribuendo l'elaborazione su un cluster di macchine che riconfigura in tempo reale, in base al carico. Ciò può portare a un miglioramento delle prestazioni per dollaro di determinati carichi di lavoro. Con i processi Ray, abbiamo integrato la scalabilità automatica in modo nativo nel modello del processo AWS Glue in modo da poter sfruttare appieno questa funzionalità. I processi Ray vengono eseguiti su AWS Graviton, con conseguente aumento del rapporto prezzo/prestazioni complessivo.

Oltre ai risparmi sui costi, è possibile utilizzare la scalabilità automatica nativa per eseguire i carichi di lavoro Ray senza investire tempo in operazioni di manutenzione, ottimizzazione e amministrazione del cluster. È possibile utilizzare le comuni librerie open source pronte all'uso, come pandas e l'SDK AWS per Pandas. Questi migliorano la velocità di iterazione durante lo sviluppo su AWS Glue per Ray. Quando si utilizza AWS Glue per Ray, è possibile sviluppare ed eseguire rapidamente carichi di lavoro di integrazione dei dati a costi ridotti.

Conversione da schemi semistrutturati a schemi relazionali con AWS Glue

La conversione dei dati semistrutturati in tabelle relazionali è piuttosto comune. Concettualmente, lo schema gerarchico viene appiattito per ottenere uno schema relazionale. AWS Glue è in grado di eseguire questa conversione in tempo reale.

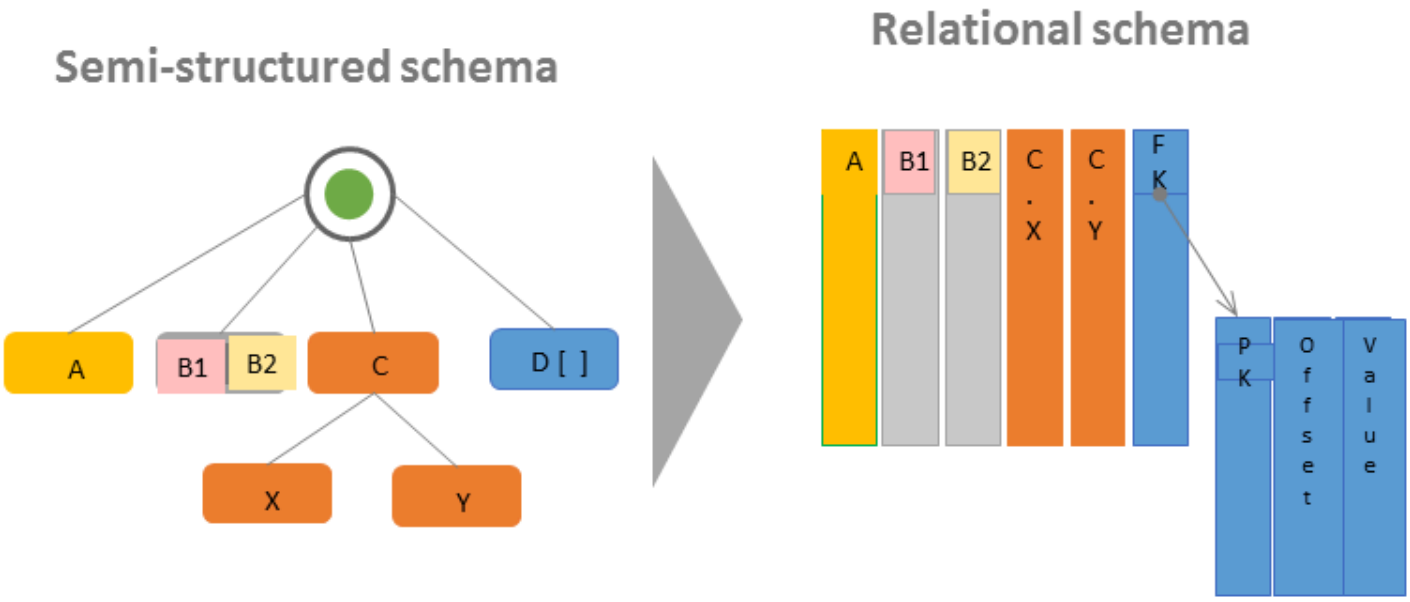
I dati semistrutturati in genere contengono mark-up per identificare le entità all'interno dei dati. Si possono avere strutture di dati annidate senza schema fisso. Per ulteriori informazioni sui dati semistrutturati, consulta Dati semistrutturati in Wikipedia.

I dati relazionali sono rappresentati da tabelle che contengono righe e colonne. Le relazioni tra tabelle possono essere rappresentate da una relazione chiave primaria (PK) su chiave esterna (FK). Per ulteriori informazioni, consulta Database relazionale in Wikipedia.

AWS Glue usa i crawler per dedurre gli schemi per i dati semistrutturati. Quindi trasforma i dati in uno schema relazionale utilizzando un processo ETL (estrarre, trasformare e caricare). Ad esempio, è possibile analizzare i dati JSON da file di origine Amazon Simple Storage Service (Amazon S3)

a tabelle Amazon Relational Database Service (Amazon RDS). Comprendere come AWS Glue gestisce le differenze tra gli schemi può aiutare a capire il processo di trasformazione.

Questo diagramma mostra come AWS Glue trasforma uno schema semistrutturato in uno schema relazionale.



Il diagramma illustra quanto segue:

- Il singolo valore `A` converte direttamente in una colonna relazionale.
- La coppia di valori `B1` e `B2` converte in due colonne relazionali.
- Struttura `C`, con figli `X` e `Y`, converte in due colonne relazionali.
- L'array `D[]` converte in una colonna relazionale con una chiave esterna (FK) che punta a un'altra tabella relazionale. Oltre a una chiave primaria (PK), la seconda tabella relazionale dispone di colonne che contengono l'offset e il valore degli oggetti nell'array.

Sistemi dei tipi di AWS Glue

AWS Glue utilizza vari sistemi dei tipi per fornire un'interfaccia versatile su sistemi di dati che archiviano i dati in modi molto diversi. Questo documento chiarisce i sistemi dei tipi e gli standard di dati di AWS Glue.

Tipi di catalogo dati di AWS Glue

Il catalogo dati è un registro di tabelle e campi archiviati in vari sistemi di dati, un metastore. Quando i componenti di AWS Glue, come i crawler AWS Glue e i processi AWS Glue con Spark, scrivono nel catalogo dati, lo fanno con un sistema dei tipi interno per tracciare i tipi di campi. Questi valori sono mostrati nella colonna Tipo di dati dello schema della tabella nella console AWS Glue. Questo sistema dei tipi è basato sul sistema dei tipi di Apache Hive. Per ulteriori informazioni sul sistema dei tipi di Apache Hive, consulta la sezione Tipi nella wiki di Apache Hive. Per ulteriori informazioni sul supporto e su tipi specifici, sono disponibili esempi nella console AWS Glue, come parte del generatore di schemi.

Convalida, compatibilità e altri usi

Il catalogo dati non convalida i tipi scritti nei campi del tipo. Quando i componenti di AWS Glue leggono e scrivono nel catalogo dati, saranno compatibili tra loro. AWS I componenti di Glue mirano anche a preservare un elevato grado di compatibilità con i tipi di Hive. Tuttavia, i componenti di AWS Glue non garantiscono la compatibilità con tutti i tipi di Hive. Ciò consente l'interoperabilità con strumenti come Athena DDL quando si lavora con le tabelle nel catalogo dati.

Poiché il catalogo dati non convalida i tipi, altri servizi possono utilizzare il catalogo dati per tenere traccia dei tipi utilizzando sistemi strettamente conformi al sistema dei tipi di Hive o a qualsiasi altro sistema.

Tipi negli script in AWS Glue con Spark

Quando uno script di AWS Glue con Spark interpreta o trasforma un set di dati, forniamo un `DynamicFrame`, una rappresentazione in memoria del set di dati così come viene utilizzato nello script. L'obiettivo di un `DynamicFrame` è simile a quello del `DataFrame` di Spark: modella il set di dati in modo che Spark possa pianificare ed eseguire trasformazioni sui dati. Garantiamo che la rappresentazione del tipo di `DynamicFrame` sia intercompatibile con il `DataFrame` fornendo i metodi `toDF` e `fromDF`.

Se le informazioni sul tipo possono essere inferite o fornite a un `DataFrame`, possono essere inferite o fornite a un `DynamicFrame`, se non diversamente documentato. Quando forniamo lettori o scrittori ottimizzati per formati di dati specifici, se Spark è in grado di leggere o scrivere i tuoi dati, i nostri lettori e scrittori forniti saranno in grado di farlo, ad esclusione delle limitazioni documentate. Per ulteriori informazioni su lettori e scrittori, consulta the section called “Opzioni del formato dei dati”.

Il tipo di scelta

Il `DynamicFrames` fornisce un meccanismo per modellare i campi in un set di dati il cui valore può avere tipi incoerenti su disco tra le righe. Ad esempio, un campo può contenere un numero memorizzato come stringa in alcune righe e un numero intero in altre. Questo meccanismo è un tipo in memoria denominato `Choice`. Forniamo trasformazioni come il metodo `ResolveChoice`, per risolvere le colonne Choice in un tipo concreto. AWS Glue ETL non scriverà il tipo Choice nel catalogo dati nel normale corso dell'operazione; i tipi Choice esistono solo nel contesto dei modelli di memoria DynamicFrame dei set di dati. Per un esempio di utilizzo del tipo Choice, consulta the section called “Esempio di preparazione dei dati”.

Tipi di Crawler di AWS Glue

I crawler mirano a produrre uno schema coerente e utilizzabile per il tuo set di dati, quindi lo archiviano in catalogo dati per utilizzarlo in altri componenti di AWS Glue e in Athena. I crawler gestiscono i tipi come descritto nella sezione precedente sul catalogo dati, the section called “Tipi di catalogo dati di AWS Glue”. Per produrre un tipo utilizzabile negli scenari di tipo "Choice", in cui una colonna contiene valori di due o più tipi, i crawler creeranno un tipo `struct` che modella i tipi potenziali.

Nozioni di base su AWS Glue

Nelle sezioni seguenti vengono fornite informazioni sulla configurazione di AWS Glue. Non tutte le sezioni per la configurazione sono necessarie per iniziare a utilizzare AWS Glue. È possibile utilizzare le istruzioni in base alle necessità per configurare le autorizzazioni IAM, la crittografia, il DNS se si utilizza un VPC, l'ambiente per accedere agli archivi dati o se si utilizzano le sessioni interattive.

Argomenti

Panoramica dell'utilizzo di AWS Glue

Con AWS Glue, puoi archiviare i metadati nel AWS Glue Data Catalog. Puoi utilizzare questi metadati per orchestrare i processi ETL che trasformano le origini dati e caricano il data warehouse o il data lake. Le fasi seguenti descrivono il flusso di lavoro generale e alcune delle scelte che effettui quando usi AWS Glue.

> **Note**
>
> È possibile seguire i passaggi riportati di seguito oppure creare un flusso di lavoro che esegua automaticamente i passaggi da 1 a 3. Per ulteriori informazioni, consulta the section called "Esecuzione di attività ETL complesse utilizzando gli schemi e i flussi di lavoro".

1. Popola il AWS Glue Data Catalog con le definizioni di tabella.

   Nella console, per gli archivi dati persistenti, è possibile aggiungere un crawler per popolare AWS Glue Data Catalog. Puoi avviare la procedura guidata Add crawler (Aggiungi crawler) dall'elenco delle tabelle o dall'elenco dei crawler. Puoi scegliere uno o più datastore a cui accede il tuo crawler. Puoi anche creare una pianificazione per determinare la frequenza di esecuzione del

crawler. Per i flussi di dati, è possibile creare manualmente la definizione della tabella e definire le proprietà del flusso.

Facoltativamente, puoi fornire un classificatore personalizzato che ricava lo schema dei dati. Puoi creare classificatori personalizzati usando un pattern grok. AWS Glue fornisce tuttavia classificatori integrati che vengono usati automaticamente dai crawler se un classificatore personalizzato non riconosce i dati. Quando definisci un crawler, non devi necessariamente selezionare un classificatore. Per ulteriori informazioni sui classificatori in AWS Glue, consulta Aggiunta di classificatori a un crawler in AWS Glue.

Il crawling di alcuni tipi di datastore richiede una connessione che fornisce l'autenticazione e le informazioni sull'ubicazione. Se necessario, puoi creare una connessione che fornisce queste informazioni obbligatorie nella console AWS Glue.

Il crawler legge il datastore e crea definizioni dei dati e tabelle denominate nel AWS Glue Data Catalog. Queste tabelle sono organizzate in un database di tua scelta. Puoi inoltre popolare il catalogo dati con tabelle create manualmente. Con questo metodo, fornisci lo schema e altri metadati per creare le definizioni di tabelle nel catalogo dati. Poiché questo metodo può essere un po' noioso e suscettibile di errori, spesso è meglio avere un crawler che crea le definizioni di tabella.

Per ulteriori informazioni sul popolamento del AWS Glue Data Catalog con le definizioni di tabelle, consulta Tabelle AWS Glue.

2. Definisci un processo che descrive la trasformazione dei dati dall'origine alla destinazione.

   Di solito, per creare un processo, devi effettuare le seguenti scelte:

   - Seleziona una tabella da AWS Glue Data Catalog da usare come origine del processo. Il processo utilizza questa definizione di tabella per accedere alla tua origine dati e interpretare il formato dei dati.
   - Seleziona una tabella o una posizione da AWS Glue Data Catalog da usare come destinazione del processo. Il processo utilizza queste informazioni per accedere al tuo datastore.
   - Indica a AWS Glue di generare uno script per trasformare l'origine nella destinazione. AWS Glue genera il codice per chiamare le trasformazioni integrate e convertire i dati dal formato dello schema di origine a quello dello schema di destinazione. Queste trasformazioni eseguono le operazioni quali copiare i dati, rinominare le colonne e filtrare i dati per trasformare i dati in base alle esigenze. Puoi modificare lo script nella console AWS Glue.

Per ulteriori informazioni sulla definizione dei processi in AWS Glue, consulta Creazione di processi ETL visivi con AWS Glue Studio.

3. Esegui il tuo processo per trasformare i dati.

   Puoi eseguire il processo on demand oppure avviarlo in base a uno dei seguenti tipi di trigger:

   - Trigger basato su una pianificazione cron.
   - Trigger basato su un evento: ad esempio il completamento di un altro processo può avviare un processo AWS Glue.
   - Trigger che avvia un processo on demand.

   Per ulteriori informazioni sui trigger in AWS Glue, consulta Avvio di lavori e crawler utilizzando i trigger.

4. Monitora i crawler pianificati e i processi attivati.

   Usa la console AWS Glue per visualizzare gli elementi seguenti:

   - Dettagli ed errori dell'esecuzione del processo.
   - Dettagli ed errori dell'esecuzione del crawler.
   - Notifiche sulle attività AWS Glue

   Per ulteriori informazioni sul monitoraggio di crawler e processi in AWS Glue, consulta Monitoraggio di AWS Glue.

Impostazione delle autorizzazioni IAM per AWS Glue

Le istruzioni in questo argomento consentono di configurare rapidamente le autorizzazioni AWS Identity and Access Management (IAM) per AWS Glue. Completa le seguenti operazioni:

- Concedi alle tue identità IAM l'accesso alle risorse AWS Glue.
- Crea un ruolo di servizio per eseguire processi, accedere ai dati ed eseguire attività di Qualità dei dati di AWS Glue.

Per istruzioni dettagliate su come personalizzare le autorizzazioni IAM per AWS Glue, consulta la pagina Configurazione delle autorizzazioni IAM per AWS Glue.

Configurazione delle autorizzazioni IAM per AWS Glue nella AWS Management Console

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Selezionare Getting started (Nozioni di base).
3. In Prepara il tuo account per AWS Glue, scegli Configura le autorizzazioni IAM.
4. Scegli le identità IAM (ruoli o utenti) a cui vuoi concedere le autorizzazioni AWS Glue. AWS Gluecollega la policy gestita `AWSGlueConsoleFullAccess` a queste identità. Puoi saltare questo passaggio se desideri impostare queste autorizzazioni manualmente o impostare solo un ruolo di servizio predefinito.
5. Seleziona Successivo.
6. Scegli il livello di accesso ad Amazon S3 di cui hanno bisogno i tuoi ruoli e i tuoi utenti. Le opzioni scelte in questo passaggio vengono applicate a tutte le identità selezionate.

   a. In Scegli le posizioni S3, scegli le posizioni Amazon S3 a cui desideri concedere l'accesso.

   b. Quindi, scegli se le tue identità devono avere accesso di Sola lettura (consigliato) o di Lettura e scrittura alle posizioni che hai selezionato in precedenza. AWS Glue aggiunge policy di autorizzazione alle tue identità in base alla combinazione di posizioni e autorizzazioni di lettura o scrittura selezionate.

   La tabella seguente mostra le autorizzazioni che AWS Glue collega per l'accesso ad Amazon S3.

Se scegli...	AWS Glue collega...
Nessuna modifica	Nessuna autorizzazione. AWS Glue non apporterà alcuna modifica alle autorizza zioni della tua identità.
Concedi l'accesso a posizioni Amazon S3 specifiche (solo lettura)	Una policy inline incorporata nelle identità IAM selezionate. Per ulteriori informazi oni, consulta la sezione Inline policies nella Guida per l'utente di IAM. AWS Glue nomina la policy utilizzan do la seguente convenzione: `AWSGlueConsole <Role/Use`

Se scegli...	AWS Glue collega...
	r> `InlinePolicy-read-specific-access-` *<UUID>*. Ad esempio: `AWSGlueConsoleRoleInlinePolicy-read-specific-access-123456780123` . Di seguito è riportato un esempio di policy inline che AWS Glue collega per concedere l'accesso in sola lettura a una posizione Amazon S3 specificata. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "s3:Get*", "s3:List*" ], "Resource": [ "arn:aws:s3::: DOC-EXAMPLE-BUCKET /*" ] } ] }`

Se scegli...	AWS Glue collega...
Concedi l'accesso a posizioni Amazon S3 specifiche (lettura e scrittura)	Una policy inline incorporata nelle identità IAM selezionate. Per ulteriori informazioni, consulta la sezione Inline policies nella Guida per l'utente di IAM. AWS Glue nomina la policy utilizzando la seguente convenzione: `AWSGlueConsole <Role/User> InlinePolicy-read -and-write-specific-access- <UUID>`. Ad esempio: `AWSGlueConsoleRoleInlinePolicy-read-and-write-specific-access-123456780123 .` Di seguito è riportato un esempio di policy inline che AWS Glue collega per concedere l'accesso in lettura e scrittura a una posizione Amazon S3 specificata. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "s3:Get*", "s3:List*", "s3:*Object*" ], "Resource": [ "arn:aws: s3::: DOC-EXAMPLE-BUCKET1 /*", "arn:aws: s3::: DOC-EXAMPLE-BUCKET2 /*" ] } ]`

Se scegli...	AWS Glue collega...
	`}`
Concedi l'accesso completo ad Amazon S3 (solo lettura)	La policy IAM gestita `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta la sezione AWS managed policy: AmazonS3ReadOnlyAccess.
Concedi l'accesso completo ad Amazon S3 (lettura e scrittura)	La policy IAM gestita `AmazonS3FullAccess`. Per ulteriori informazioni, consulta la sezione AWS managed policy: AmazonS3FullAccess.

7. Seleziona Successivo.
8. Scegli un ruolo di servizio AWS Glue predefinito per il tuo account. Un ruolo di servizio è un ruolo IAM che AWS Glue utilizza per accedere alle risorse di altri servizi AWS per tuo conto. Per ulteriori informazioni, consulta Ruoli di servizio per AWS Glue.

   - Quando scegli il ruolo di servizio AWS Glue standard, AWS Glue crea un nuovo ruolo IAM nel tuo Account AWS denominato `AWSGlueServiceRole` con le seguenti policy gestite collegate. Se il tuo account ha già un ruolo IAM denominato `AWSGlueServiceRole`, AWS Glue collega queste policy al ruolo esistente.
     - AWSGlueServiceRole
     - AmazonS3FullAccess
   - Quando scegli un ruolo IAM esistente, AWS Glue imposta il ruolo come predefinito, ma non aggiunge alcuna autorizzazione. Assicurati di aver configurato il ruolo da utilizzare come ruolo di servizio per AWS Glue. Per ulteriori informazioni, consulta Fase 1: creare una policy IAM per il servizio AWS Glue e Fase 2: creare un ruolo IAM per AWS Glue.

9. Seleziona Successivo.
10. Infine, rivedi le autorizzazioni che hai selezionato, quindi scegli Applica le modifiche. Quando applichi le modifiche, AWS Glue aggiunge le autorizzazioni IAM alle identità che hai selezionato. Puoi visualizzare o modificare le nuove autorizzazioni nella console IAM all'indirizzo https://console.aws.amazon.com/iam/.

Ora hai completato la configurazione minima delle autorizzazioni IAM per AWS Glue. In un ambiente di produzione, ti consigliamo di acquisire familiarità con Sicurezza in AWS Glue e Gestione delle identità e degli accessi per AWS Glue per proteggere le risorse AWS per il tuo caso d'uso.

Fasi successive

Ora che hai configurato le autorizzazioni IAM, puoi esplorare i seguenti argomenti per iniziare a utilizzare AWS Glue:

- Getting Started with AWS Glue in AWS Skill Builder
- Nozioni di base su AWS Glue Data Catalog

Configurazione di AWS Glue Studio

Completa i processi in questa sezione quando utilizzi AWS Glue per l'ETL visivo la prima volta:

Argomenti

- Esaminare le autorizzazioni IAM necessarie per l'utente AWS Glue Studio
- Esaminare le autorizzazioni IAM necessarie per i processi ETL
- Impostazione delle autorizzazioni IAM per AWS Glue Studio
- Configurazione di un VPC per il tuo processo ETL

Esaminare le autorizzazioni IAM necessarie per l'utente AWS Glue Studio

Per utilizzare AWS Glue Studio, l'utente deve avere accesso a varie risorse AWS. L'utente deve essere in grado di visualizzare e selezionare i bucket Amazon S3, le policy e i ruoli IAM e gli oggetti AWS Glue Data Catalog.

Autorizzazioni di servizio AWS Glue

AWS Glue Studio utilizza le operazioni e le risorse del servizio AWS Glue. Per utilizzare in modo efficace AWS Glue Studio, l'utente ha bisogno delle autorizzazioni per tali operazioni e risorse. È possibile concedere all'utente di AWS Glue Studio la policy gestita da `AWSGlueConsoleFullAccess` oppure creare una policy personalizzata con un set di autorizzazioni più piccolo.

> **Important**
>
> In base alle best practice di sicurezza, si consiglia di limitare l'accesso rafforzando le policy per limitare ulteriormente l'accesso al bucket Amazon S3 e ai gruppi di log Amazon CloudWatch. Per un esempio di policy Amazon S3, consulta la pagina relativa alla scrittura di policy IAM per concedere l'accesso a un bucket Amazon S3.

Creazione di criteri IAM personalizzati per AWS Glue Studio

È possibile creare una policy personalizzata con un set di autorizzazioni più piccolo per AWS Glue Studio. La policy può concedere autorizzazioni per un sottoinsieme di oggetti o operazioni. Durante la creazione di una policy personalizzata, utilizza le seguenti informazioni.

Per utilizzare AWS Glue Studio API, includi `glue:UseGlueStudio` nella policy di operazione nelle autorizzazioni IAM. L'utilizzo di `glue:UseGlueStudio` ti permetterà di accedere a tutte le operazioni di AWS Glue Studio anche quando più operazioni vengono aggiunte all'API nel tempo.

Operazioni del grafo aciclico orientato (DAG)

- CreateDag
- UpdateDag
- GetDag
- DeleteDag

Operazioni di processo

- SaveJob
- GetJob
- CreateJob
- DeleteJob
- GetJob
- UpdateJob

Opzione di esecuzione del processo

- StartJobRun
- GetJobRuns
- BatchStopJobRun
- GetJobRuns
- QueryJobRuns
- QueryJobs
- QueryJobRunsAggregated

Operazioni dello schema

- GetSchema
- GetInferredSchema

Operazioni del database

- GetDatabases

Operazioni del piano

- GetPlan

Operazioni della tabella

- SearchTables
- GetTables
- GetTables

Operazioni di connessione

- CreateConnection
- DeleteConnection
- UpdateConnection
- GetConnections
- GetConnection

Operazioni di mappatura

- GetMapping

Operazioni proxy S3

- ListBuckets
- ListObjectsV2
- GetBucketLocation

Operazioni di configurazione di sicurezza

- GetSecurityConfigurations

Operazioni di script

- CreateScript (diverso dall'API con lo stesso nome in AWS Glue)

Accedendo alle API AWS Glue Studio

Per accedere a AWS Glue Studio, aggiungi `glue:UseGlueStudio` nell'elenco delle policy delle operazioni nelle autorizzazioni IAM.

Nell'esempio riportato di seguito `glue:UseGlueStudio` è incluso nella policy delle operazioni, ma le API AWS Glue Studio non sono identificate singolarmente. Questo perché quando includi `glue:UseGlueStudio`, viene automaticamente consentito l'accesso alle API interne senza dover specificare l'individuo AWS Glue Studio API nelle autorizzazioni IAM.

Nell'esempio, le policy delle operazioni aggiuntive elencate (ad esempio `glue:SearchTables`) non sono AWS Glue Studio API, quindi dovranno essere incluse nelle autorizzazioni IAM come richiesto. Potresti inoltre includere operazioni Amazon S3 Proxy per specificare il livello di accesso Amazon S3 da concedere. La policy di esempio riportata di seguito fornisce l'accesso per aprire AWS Glue Studio, creare un processo visivo e salvarlo/eseguirlo se il ruolo IAM selezionato dispone dell'accesso sufficiente.

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

```
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "glue:UseGlueStudio",
                "iam:ListRoles",
                "iam:ListUsers",
                "iam:ListGroups",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "glue:SearchTables",
                "glue:GetConnections",
                "glue:GetJobs",
                "glue:GetTables",
                "glue:BatchStopJobRun",
                "glue:GetSecurityConfigurations",
                "glue:DeleteJob",
                "glue:GetDatabases",
                "glue:CreateConnection",
                "glue:GetSchema",
                "glue:GetTable",
                "glue:GetMapping",
                "glue:CreateJob",
                "glue:DeleteConnection",
                "glue:CreateScript",
                "glue:UpdateConnection",
                "glue:GetConnection",
                "glue:StartJobRun",
                "glue:GetJobRun",
                "glue:UpdateJob",
                "glue:GetPlan",
                "glue:GetJobRuns",
                "glue:GetTags",
                "glue:GetJob",
                "glue:QueryJobRuns",
                "glue:QueryJobs",
                "glue:QueryJobRunsAggregated"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "iam:PassRole"
```

```
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Autorizzazioni per notebook e anteprima dati

Le anteprime dei dati e i notebook consentono di visualizzare un campione dei dati in qualsiasi fase del processo (lettura, trasformazione, scrittura), senza doverlo eseguire. Specifica un ruolo AWS Identity and Access Management (IAM) per AWS Glue Studio da utilizzare quando si accede ai dati. I ruoli IAM sono destinati ad essere prevedibili e non hanno credenziali standard a lungo termine come una password o chiavi d'accesso associate ad esso. Invece, quando AWS Glue Studio assume il ruolo, IAM fornisce credenziali di sicurezza temporanee.

Per garantire che le anteprime dei dati e i comandi del notebook funzionino correttamente, usa un ruolo con un nome che inizia con la stringa `AWSGlueServiceRole`. Se decidi di usare un altro nome per il ruolo, dovrai aggiungere l'autorizzazione `iam:passrole` e configurare una policy per il ruolo in IAM. Per ulteriori informazioni, consulta Crea una policy IAM per ruoli non denominati "AWSGlueServiceRole*".

> **Warning**
>
> Se un ruolo concede l'autorizzazione `iam:passrole` per un notebook e tu implementi il concatenamento dei ruoli, un utente potrebbe ottenere involontariamente l'accesso al notebook. Al momento non è implementato alcun controllo che permetta di monitorare a quali utenti sia stato concesso l'accesso al notebook.

Se desideri negare a un'identità IAM la possibilità di creare sessioni di anteprima dei dati, consulta l'esempio di the section called “Negare a un'identità la possibilità di creare sessioni di anteprima dei dati” seguente.

Autorizzazioni di Amazon CloudWatch

Puoi monitorare i processi AWS Glue Studio usando Amazon CloudWatch, che raccoglie i dati non elaborati da AWS Glue e li elabora trasformandoli in parametri leggibili quasi in tempo reale. Per impostazione predefinita, i dati dei parametri AWS Glue vengono inviati automaticamente a CloudWatch. Per maggiori informazioni, consulta Che cos'è Amazon CloudWatch? nella Guida per l'utente di Amazon CloudWatch e Parametri AWS Glue nella Guida per gli sviluppatori di AWS Glue.

Per accedere ai pannelli di controllo di CloudWatch, l'utente che accede ad AWS Glue Studio ha bisogno di uno dei seguenti elementi:

- La policy `AdministratorAccess`
- La policy `CloudWatchFullAccess`
- Una policy personalizzata che includa una o più di queste autorizzazioni specifiche:
  - `cloudwatch:GetDashboard` e `cloudwatch:ListDashboards` per visualizzare i pannelli di controllo
  - `cloudwatch:PutDashboard` per creare o modificare i pannelli di controllo
  - `cloudwatch:DeleteDashboards` per eliminare i pannelli di controllo

Per ulteriori informazioni sulla modifica delle autorizzazioni per un utente IAM utilizzando le policy, consulta Modifica delle autorizzazioni per un utente IAM nella Guida per l'utente IAM.

Esaminare le autorizzazioni IAM necessarie per i processi ETL

Quando si crea un processo utilizzando AWS Glue Studio, il processo assume le autorizzazioni del ruolo IAM specificate al momento della creazione. Questo ruolo IAM deve avere le autorizzazioni per estrarre dati dall'origine dati, scriverli nella destinazione e accedere alle risorse AWS Glue.

Il nome del ruolo creato per il processo deve iniziare con la stringa `AWSGlueServiceRole` per poter essere usato correttamente da AWS Glue Studio. È ad esempio possibile denominare il proprio ruolo `AWSGlueServiceRole-FlightDataJob`.

Autorizzazioni origine dati e destinazione dati

Un processo AWS Glue Studio deve avere accesso ad Amazon S3 per le origini, le destinazioni, gli script e le directory temporanee che usi nel processo. Puoi creare una policy per fornire un accesso granulare a risorse Amazon S3 specifiche.

- Le origini dati richiedono le autorizzazioni `s3:ListBucket` e `s3:GetObject`.
- Le destinazioni dati richiedono le autorizzazioni `s3:ListBucket`, `s3:PutObject` e `s3:DeleteObject`.

Se si sceglie Amazon Redshift come origine dati, è possibile fornire un ruolo per le autorizzazioni del cluster. Processi che vengono eseguiti su un cluster Amazon Redshift inviano comandi che accedono ad Amazon S3 per l'archiviazione temporanea utilizzando credenziali temporanee. Se il processo viene eseguito per più di un'ora, queste credenziali scadranno causando l'esito negativo del processo. Per evitare questo problema, puoi assegnare un ruolo al cluster Amazon Redshift stesso che concede le autorizzazioni necessarie ai processi utilizzando le credenziali temporanee. Per ulteriori informazioni, consulta Spostamento di dati da e verso Amazon Redshift nella Guida per gli sviluppatori di AWS Glue.

Se il processo utilizza origini dati o destinazioni diverse da Amazon S3, devi allegare le autorizzazioni necessarie al ruolo IAM utilizzato dal processo per accedere a tali origini dati e destinazioni. Per ulteriori informazioni, consulta Impostazione dell'ambiente per accedere a archivio dati nella Guida per gli sviluppatori di AWS Glue.

Se si utilizzano connettori e connessioni per l'archivio dati, è necessario disporre di autorizzazioni aggiuntive, come descritto in the section called “Autorizzazioni richieste per l'utilizzo dei connettori”.

Autorizzazioni necessarie per l'eliminazione dei processi

In AWS Glue Studio, è possibile selezionare nella console più processi da eliminare. Per eseguire questa azione, è necessario disporre delle autorizzazioni `glue:BatchDeleteJob`. Diversamente, la console AWS Glue richiede l'autorizzazione `glue:DeleteJob` per l'eliminazione dei processi.

Autorizzazioni AWS Key Management Service

Se intendi accedere a origini e destinazioni Amazon S3 che utilizzano la crittografia lato server con AWS Key Management Service (AWS KMS), allega una policy al ruolo AWS Glue Studio utilizzato dal processo, che consente al processo di decrittare i dati. Il ruolo del processo necessita delle autorizzazioni `kms:ReEncrypt`, `kms:GenerateDataKey` e `kms:DescribeKey`. Richiede inoltre

l'autorizzazione `kms:Decrypt` per caricare o scaricare un oggetto Amazon S3 crittografato con una chiave master del cliente AWS KMS (CMK)..

Sono previsti costi aggiuntivi per l'utilizzo delle chiavi CMK AWS KMS. Per ulteriori informazioni, consulta Concetti di AWS Key Management Service - Chiavi master cliente (CMK) in Prezzi di AWS Key Management Service e nella Guida per gli sviluppatori di AWS Key Management Service.

Autorizzazioni richieste per l'utilizzo dei connettori

Se usi una connessione e un connettore personalizzato AWS Glue per accedere a un archivio dati, il ruolo utilizzato per eseguire il processo ETL AWS Glue necessita di autorizzazioni aggiuntive allegate:

- La policy gestita da AWS `AmazonEC2ContainerRegistryReadOnly` per accedere ai connettori acquistati da Marketplace AWS.
- Le autorizzazioni `glue:GetJob` e `glue:GetJobs`.
- Le autorizzazioni AWS Secrets Manager per accedere ai segreti utilizzati con le connessioni. Per le policy IAM di esempio, consulta Esempio: Autorizzazione per recuperare valori segreti.

Se il processo ETL AWS Glue viene eseguito all'interno di un VPC che esegue Amazon VPC, il VPC deve essere configurato come descritto in the section called “Configurazione di un VPC per il tuo processo ETL”.

Impostazione delle autorizzazioni IAM per AWS Glue Studio

È possibile creare i ruoli e assegnare policy agli utenti e ai ruoli del processo utilizzando l'utente amministratore AWS.

È possibile utilizzare la policy gestita AWS AWSGlueConsoleFullAccess per fornire le autorizzazioni necessarie per l'utilizzo della console AWS Glue Studio.

Per creare una policy personalizzata, segui la procedura descritta in Creare una policy IAM per il servizio AWS Glue nella Guida per gli sviluppatori di AWS Glue. Includi le autorizzazioni IAM descritte in precedenza in Esaminare le autorizzazioni IAM necessarie per l'utente AWS Glue Studio.

Argomenti

- Allega le policy all'utente AWS Glue Studio
- Crea una policy IAM per ruoli non denominati "AWSGlueServiceRole*"

Allega le policy all'utente AWS Glue Studio

Qualsiasi utente AWS che accede alla console AWS Glue Studio deve avere le autorizzazioni per accedere a risorse specifiche. Puoi fornire queste autorizzazioni usando l'assegnazione delle policy IAM all'utente.

Per collegare la policy gestita AWSGlueConsoleFullAccess a un utente

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Policies (Policy).
3. Nell'elenco delle policy, seleziona la casella di controllo accanto a AWSGlueConsoleFullAccess. Puoi utilizzare il menu Filtro e la casella di ricerca per filtrare l'elenco di policy.
4. Scegli Operazioni di policy, quindi Collega.
5. Sceglie l'utente a cui collegare la policy. Puoi usare il menu Filtro e la casella di ricerca per filtrare l'elenco delle entità principali. Dopo aver scelto l'utente a cui collegare la policy, scegli Attach policy (Collega policy).
6. Ripeti i passaggi precedenti per allegare ulteriori policy all'utente, in base alle esigenze.

Crea una policy IAM per ruoli non denominati "AWSGlueServiceRole*"

Come configurare una policy IAM per i ruoli utilizzati da AWS Glue Studio

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Aggiunta di una nuova policy IAM. È possibile aggiungere a una policy esistente o creare una nuova policy IAM inline. Per creare una policy IAM
    1. Seleziona Policy, quindi scegli Create Policy (Crea policy). Se viene visualizzato il pulsante Get Started (Inizia), sceglilo, quindi scegli Create Policy (Crea policy)
    2. Accanto a Create Your Own Policy (Crea la tua policy) scegli Select (Seleziona).
    3. In Policy Name (Nome policy) digita un nome facile da ricordare. Facoltativamente, digita un testo descrittivo in Description (Descrizione).
    4. In Policy Document (Documento policy) digita un'istruzione di policy nel formato seguente, quindi scegli Create Policy (Crea policy):

3. Copia e incolla i seguenti blocchi nella policy sotto l'array "Statement", sostituendo *my-interactive-session-role-prefix* con il prefisso che tutti i ruoli comuni possono utilizzare per associarsi alle autorizzazioni per AWS Glue.

```
{
    "Action": [
        "iam:PassRole"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/my-interactive-session-role-prefix*",
    "Condition": {
        "StringLike": {
            "iam:PassedToService": [
                "glue.amazonaws.com "
            ]
        }
    }
}
```

Ecco l'esempio completo degli array Version (versione) e Statement (istruzione) inclusi nella policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Action": [
            "iam:PassRole"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iam::*:role/my-interactive-session-role-prefix*",
        "Condition": {
            "StringLike": {
                "iam:PassedToService": [
                    "glue.amazonaws.com "
                ]
            }
        }
    }
  ]
}
```

4. Per abilitare la policy per un utente, scegli Users (Utenti).
5. Seleziona l'utente a cui desideri collegare la policy.

Configurazione di un VPC per il tuo processo ETL

Amazon Virtual Private Cloud (Amazon VPC) consente di definire una rete virtuale nella propria area isolata logicamente nell'Cloud AWS, nota come cloud privato virtuale (VPC). È possibile avviare le risorse AWS, ad esempio le istanze, nel VPC. Il VPC è molto simile a una rete tradizionale gestibile nel data center locale, ma con i vantaggi legati all'utilizzo dell'infrastruttura scalabile di AWS. È·possibile configurare il VPC, selezionare l'intervallo di indirizzi IP, creare sottoreti e configurare tabelle di routing, gateway di rete e impostazioni di sicurezza. È possibile connettere le istanze nel VPC a Internet. Per rendere Cloud AWS un'estensione del data center, puoi connettere il VPC al data center aziendale. Per proteggere le risorse in ciascuna sottorete, è possibile usare diversi livelli di sicurezza, compresi gruppi di sicurezza e liste di controllo accessi alla rete. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon VPC.

È possibile configurare i processi ETL AWS Glue da eseguire all'interno di un VPC quando si utilizzano connettori. È necessario configurare il VPC per quanto segue, in base alle esigenze:

- Accesso alla rete pubblica per archivi dati non in AWS. Tutti gli archivi dati JDBC ai quali il processo accede devono essere disponibili dalla sottorete VPC.
- Se il processo deve accedere sia alle risorse VPC che alla rete Internet pubblica, il VPC deve disporre di un gateway NAT (Network Address Translation) al suo interno.

  Per ulteriori informazioni, consulta Impostazione dell'ambiente per accedere agli archivi dati nella Guida per gli sviluppatori di AWS Glue.

Nozioni di base sui notebook in AWS Glue Studio

All'avvio di un notebook tramite AWS Glue Studio, tutti i passaggi di configurazione vengono eseguiti, in modo che tu possa esplorare i dati e iniziare a sviluppare lo script del processo dopo pochi secondi.

Nelle seguenti sezioni viene descritto come creare un ruolo e concedere le autorizzazioni appropriate per utilizzare i notebook in AWS Glue Studio per i processi ETL.

Argomenti

- Concessione di autorizzazioni per il ruolo IAM

Concessione di autorizzazioni per il ruolo IAM

Configurare AWS Glue Studio è un prerequisito per l'utilizzo di notebook.

Per utilizzare i notebook in AWS Glue, il tuo ruolo richiede quanto segue:

- Una relazione di fiducia con AWS Glue per l'operazione `sts:AssumeRole` e, se vuoi, taggare `sts:TagSession`.
- Una policy IAM contenente tutte le operazioni API per notebook, AWS Glue e sessioni interattive.
- Una policy IAM per un ruolo pass poiché il ruolo deve essere in grado di passare da notebook alle sessioni interattive.

Ad esempio, quando si crea un nuovo ruolo, è possibile aggiungere al ruolo una policy gestita da AWS standard come `AWSGlueConsoleFullAccessRole`, quindi aggiungere una nuova policy per le operazioni del notebook e un'altra per la policy IAM PassRole.

Azioni necessarie per una relazione di fiducia con AWS Glue

Quando si avvia una sessione di notebook, è necessario aggiungere `sts:AssumeRole` alla relazione di fiducia del ruolo che viene trasmesso a notebook. Se la tua sessione include tag, devi anche passare l'operazione `sts:TagSession`. Senza queste azioni, la sessione di notebook non può essere avviata.

Ad esempio:

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

Policy contenenti le operazioni API per notebook

Nella seguente policy di esempio sono descritte le Autorizzazioni IAM AWS per notebook. Se stai creando un nuovo ruolo, crea una policy che contenga quanto segue:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartNotebook",
                "glue:TerminateNotebook",
                "glue:GlueNotebookRefreshCredentials",
                "glue:DeregisterDataPreview",
                "glue:GetNotebookInstanceStatus",
                "glue:GlueNotebookAuthorize"
            ],
            "Resource": "*"
        }
    ]
}
```

È possibile utilizzare le seguenti policy IAM per consentire l'accesso a risorse specifiche:

- AwsGlueSessionUserRestrictedNotebookServiceRole: fornisce accesso totale a tutte le risorse AWS Glue tranne che per le sessioni. Permette agli utenti di creare e utilizzare solo le sessioni notebook associate all'utente. Questa policy include anche altre autorizzazioni necessarie ad AWS Glue per gestire le risorse AWS Glue in altri servizi AWS.
- AwsGlueSessionUserRestrictedNotebookPolicy: fornisce autorizzazioni che consentono agli utenti di creare e utilizzare solo le sessioni di notebook associate all'utente. Questa policy include anche le autorizzazioni per consentire esplicitamente agli utenti di passare un ruolo di sessione AWS Glue limitato.

Policy IAM per trasmettere un ruolo

Quando crei un notebook con un ruolo, tale ruolo viene passato alle sessioni interattive in modo che lo stesso ruolo possa essere utilizzato in entrambe le posizioni. Come tale, il permesso `iam:PassRole` deve essere parte della policy del ruolo.

Crea una nuova policy per il tuo ruolo utilizzando l'esempio seguente. Sostituisci il numero di account con il tuo e il nome del ruolo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::090000000210:role/<role_name>"
        }
    ]
}
```

Nozioni di base su AWS Glue Data Catalog

Il AWS Glue Data Catalog è un archivio di metadati persistente. Si tratta di un servizio gestito che puoi utilizzare per archiviare, annotare e condividere i metadati in AWS Cloud. Per ulteriori informazioni, consulta AWS Glue Data Catalog.

La AWS Glue console e alcune interfacce utente sono state aggiornate di recente.

Panoramica

Puoi utilizzare questo tutorial per creare il tuo primo catalogo dati AWS Glue, che utilizza un bucket Amazon S3 come origine dati.

In questo tutorial, verranno eseguite le operazioni seguenti tramite la console AWS Glue:

1. Creare un database
2. Creare una tabella
3. Utilizza un bucket Amazon S3 come origine dei dati

Dopo aver completato questi passaggi, avrai utilizzato correttamente un bucket Amazon S3 come origine dati per popolare il catalogo dati AWS Glue.

Fase 1: crea un database

Per iniziare, accedi AWS Management Console e apri la AWS Glueconsole.

Per creare un database utilizzando la console AWS Glue:

1. Nella console AWS Glue, scegli Databases (Database) dal menu a sinistra Data catalog (Catalogo dati).
2. Scegli Aggiungi database.
3. Nella pagina Crea database, immetti un nome per il database. Nella sezione Posizione - facoltativa, imposta la posizione dell'URI che i client di Catalogo dati devono utilizzare. Se la ignori, puoi continuare con la creazione del database.
4. (Facoltativo). Inserisci una descrizione per il database.
5. Scegliere Crea database.

Congratulazioni, hai appena configurato il tuo primo database usando la console AWS Glue. Il nuovo database verrà visualizzato nell'elenco dei database disponibili. È possibile modificare il database scegliendo il nome del database dal pannello di controllo Databases (Database).

Fasi successive

Altri modi per creare un database:

Hai appena creato un database utilizzando la console AWS Glue, ma esistono altri modi per creare un database:

- È possibile utilizzare i crawler per creare automaticamente un database e delle tabelle. Per configurare un database utilizzando i crawler, consulta Uso di crawler nella console AWS Glue.
- È possibile utilizzare modelli AWS CloudFormation. Consulta Creazione di risorse AWS Glue utilizzando i modelli AWS Glue Data Catalog.
- Puoi creare un database anche utilizzando il plugin Operazioni API del database AWS Glue.

  Per creare un database utilizzando il plugin dell'operazione `create`, strutturare la richiesta includendo i parametri `DatabaseInput` (obbligatori).

  Ad esempio:

Di seguito sono riportati esempi di come è possibile utilizzare la CLI, Boto3 o DDL per definire una tabella basata sullo stesso file flights_data.csv dal bucket S3 utilizzato nel tutorial.

CLI

```
aws glue create-database --database-input "{\"Name\":\"clidb\"}"
```

Boto3

```
glueClient = boto3.client('glue')

response = glueClient.create_database(
    DatabaseInput={
        'Name': 'boto3db'
    }
)
```

Per ulteriori informazioni sui tipi di dati, sulla struttura e sulle operazioni API del database, consulta API database.

Fasi successive

Nella sezione successiva, creerai una tabella e la aggiungerai al database.

Puoi anche esplorare le impostazioni e le autorizzazioni per il catalogo dati. Consulta Uso delle impostazioni dei cataloghi dati nella console AWS Glue.

Fase 2. Creare una tabella

In questa fase, utilizzerai la console AWS Glue per creare una tabella.

1. Nella console AWS Glue, scegli Tables (Tabelle) dal menu a sinistra.
2. Scegli Aggiungi tabella.
3. Imposta le proprietà della tabella inserendo un nome per la tabella in Table details (Dettagli della tabella).
4. Nella sezione Databases (Database), scegli il database creato nella fase 1 dal menu a discesa.

5. Nella sezione Add a data store (Aggiungi un datastore), per impostazione predefinita il tipo di origine sarà S3.
6. Per Data is located in (I dati si trovano in), scegli Specified path in another account (Percorso specificato in un altro account).
7. Copia e incolla il percorso per il campo di input Include path (Percorso di inclusione):

   ```
   s3://crawler-public-us-west-2/flight/2016/csv/
   ```
8. Nella sezione Data format (Formato dei dati), per Classification (Classificazione), scegli CSV e per Delimiter (Delimitatore), scegli comma (,) (virgola [,]). Seleziona Avanti.
9. Ti viene chiesto di definire uno schema. Uno schema definisce la struttura e il formato di un registro di dati. Scegli Add column (Aggiungi colonna). (Per ulteriori informazioni, consulta Registri degli schemi).
10. Specifica le proprietà della colonna:

    a. Inserisci un nome per la colonna.
    b. Per Column type (Tipo di colonna), 'string' è già selezionata per impostazione predefinita.
    c. Per Column number (Numero di colonna), '1' è già selezionato per impostazione predefinita.
    d. Scegli Aggiungi.
11. Ti viene richiesto di aggiungere indici di partizione. Si tratta di un'opzione facoltativa. Per saltare questo passaggio, scegli Next (Successivo).
12. Viene visualizzato un riepilogo delle proprietà della tabella. Se tutto appare come previsto, scegli Crea. In caso contrario, scegli Back (Indietro) e modifica in base alle necessità.

Congratulazioni, hai creato manualmente una tabella in modo corretto e l'hai associata a un database. La tabella appena creata apparirà nel pannello di controllo Tables (Tabelle). Dal pannello di controllo, puoi modificare e gestire le tabelle.

Per ulteriori informazioni, consulta Utilizzo di tabelle nella console AWS Glue.

Passaggi successivi

Fasi successive

Ora che il catalogo dati è popolato, è possibile iniziare a creare i processi in AWS Glue. Vedi Creazione di lavori ETL visivi con AWS Glue Studio.

Oltre a utilizzare la console, esistono altri modi per definire le tabelle nel catalogo dati, tra cui:

- Creare ed eseguire un crawler
- Aggiungere classificatori a un crawler in AWS Glue
- Usare la Tabella API AWS Glue
- Usare il modello AWS Glue Data Catalog
- Eseguire la migrazione di un metastore Apache Hive
- Utilizzo della AWS CLI, Boto3 o Data Definition Language (DDL)

Di seguito sono riportati esempi di come è possibile utilizzare la CLI, Boto3 o DDL per definire una tabella basata sullo stesso file flights_data.csv dal bucket S3 utilizzato nel tutorial.

Consulta la documentazione su come strutturare un comando AWS CLI. L'esempio della CLI contiene la sintassi JSON per il valore "aws glue create-table --table-input".

CLI

```
{
        "Name": "flights_data_cli",
        "StorageDescriptor": {
            "Columns": [
                {
                    "Name": "year",
                    "Type": "bigint"
                },
                {
                    "Name": "quarter",
                    "Type": "bigint"
                }
            ],
            "Location": "s3://crawler-public-us-west-2/flight/2016/csv",
            "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
            "OutputFormat":
 "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
            "Compressed": false,
            "NumberOfBuckets": -1,
            "SerdeInfo": {
                "SerializationLibrary":
 "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
                "Parameters": {
                    "field.delim": ",",
                    "serialization.format": ","
                }
```

```
            }
        },
        "PartitionKeys": [
            {
                "Name": "mon",
                "Type": "string"
            }
        ],
        "TableType": "EXTERNAL_TABLE",
        "Parameters": {
            "EXTERNAL": "TRUE",
            "classification": "csv",
            "columnsOrdered": "true",
            "compressionType": "none",
            "delimiter": ",",
            "skip.header.line.count": "1",
            "typeOfData": "file"
        }
    }
```

Boto3

```
import boto3

glue_client = boto3.client("glue")

response = glue_client.create_table(
    DatabaseName='sampledb',
    TableInput={
        'Name': 'flights_data_manual',
    'StorageDescriptor': {
      'Columns': [{
        'Name': 'year',
        'Type': 'bigint'
      }, {
        'Name': 'quarter',
        'Type': 'bigint'
      }],
      'Location': 's3://crawler-public-us-west-2/flight/2016/csv',
      'InputFormat': 'org.apache.hadoop.mapred.TextInputFormat',
      'OutputFormat':
 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat',
```

```
      'Compressed': False,
      'NumberOfBuckets': -1,
      'SerdeInfo': {
        'SerializationLibrary':
 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe',
       'Parameters': {
         'field.delim': ',',
         'serialization.format': ','
       }
      },
    },
    'PartitionKeys': [{
      'Name': 'mon',
      'Type': 'string'
    }],
    'TableType': 'EXTERNAL_TABLE',
    'Parameters': {
      'EXTERNAL': 'TRUE',
      'classification': 'csv',
      'columnsOrdered': 'true',
      'compressionType': 'none',
      'delimiter': ',',
      'skip.header.line.count': '1',
      'typeOfData': 'file'
    }
    }
)
```

DDL

```
CREATE EXTERNAL TABLE `sampledb`.`flights_data` (
  `year` bigint,
  `quarter` bigint)
PARTITIONED BY (
  `mon` string)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
```

```
  's3://crawler-public-us-west-2/flight/2016/csv/'
TBLPROPERTIES (
  'classification'='csv',
  'columnsOrdered'='true',
  'compressionType'='none',
  'delimiter'=',',
  'skip.header.line.count'='1',
  'typeOfData'='file')
```

Impostazione dell'accesso di rete agli archivi di dati

Per eseguire i processi ETL (Extract, Transform and Load, estrazione, trasformazione e caricamento), AWS Glue deve poter accedere ai datastore. Se un processo non deve essere necessariamente eseguito nella tua sottorete Virtual Private Cloud (VPC) (es. trasformazione di dati da Amazon S3 ad Amazon S3) non servono ulteriori configurazioni.

Se un processo deve essere eseguito nella tua sottorete VPC, (es. trasformazione di dati da un datastore JDBC a una sottorete privata), AWS Glue imposta le interfacce di rete elastiche che consentono di connettere i processi dell'utente ad altre risorse all'interno del VPC in modo sicuro. A ogni interfaccia di rete elastica è assegnato un indirizzo IP privato preso dall'intervallo di indirizzi IP nella sottorete che hai specificato. Nessun indirizzo IP pubblico assegnato. I gruppi di sicurezza specificati nella connessione AWS Glue vengono applicati a ciascuna delle interfacce di rete elastiche. Per ulteriori informazioni, consulta Configurazione di Amazon VPC per connessioni JDBC agli archivi dati Amazon RDS da AWS Glue.

Tutti i datastore JDBC ai quali il processo accede devono essere disponibili dalla sottorete VPC. Per accedere ad Amazon S3 dal VPC, serve un endpoint VPC. Se il processo deve accedere sia alle risorse VPC che alla rete Internet pubblica, il VPC deve disporre di un gateway NAT (Network Address Translation) al suo interno.

Un processo o endpoint di sviluppo può accedere a un solo VPC (e sottorete) alla volta. Se devi accedere a datastore in VPC diversi hai a disposizione le seguenti opzioni:

- Utilizza VPC in peering per accedere ai datastore. Per ulteriori informazioni su VPC in peering, consulta Nozioni di base sul VPC in peering
- Usa un bucket Amazon S3 come posizione di storage intermedia. Dividi il lavoro in due processi, con l'output Amazon S3 del processo 1 come input per il processo 2.

Per dettagli su come connettersi a un datastore Amazon Redshift utilizzando Amazon VPC, consulta la pagina the section called “Configurazione di Redshift”.

Per dettagli su come connettersi a un datastore Amazon RDS utilizzando Amazon VPC, consulta la pagina the section called “Configurazione di Amazon VPC per la connessione agli archivi dati Amazon RDS”.

Una volta impostate le regole necessarie in Amazon VPC, puoi creare una connessione in AWS Glue con le proprietà necessarie per connetterti ai datastore. Per ulteriori informazioni sulla connessione, consulta Connessione ai dati.

> **Note**
>
> Assicurati di configurare l'ambiente DNS per AWS Glue. Per ulteriori informazioni, consulta Configurazione di DNS nel VPC.

Argomenti

- Configurazione di un VPC per la connessione a PyPI per AWS Glue
- Configurazione di DNS nel VPC

Configurazione di un VPC per la connessione a PyPI per AWS Glue

Il Python Package Index (PyPI) è un repository di software per il linguaggio di programmazione Python. Questo argomento affronta i dettagli necessari per supportare l'utilizzo dei pacchetti pip installati (come specificato dal creatore della sessione utilizzando il flag `--additional-python-modules`).

L'utilizzo di sessioni interattive AWS Glue con un connettore comporta l'uso della rete VPC tramite la sottorete specificata per il connettore. Di conseguenza, i servizi AWS e le altre destinazioni di rete non sono disponibili a meno che non si utilizzi una configurazione speciale.

Le soluzioni a questo problema includono:

- Utilizzo di un gateway Internet raggiungibile dalla sessione.
- Configurazione e utilizzo di un bucket S3 con un repository PyPI/Simple contenente la chiusura transitiva delle dipendenze di un set di pacchetti.
- Utilizzo di un repository CodeArtifact che esegue il mirroring di PyPI ed è collegato al VPC.

Impostazione di un gateway Internet

Gli aspetti tecnici sono descritti in dettaglio nei casi d'uso del gateway NAT, ma tieni presente questi requisiti per l'utilizzo di `--additional-python-modules`. In particolare, `--additional-python-modules` richiede l'accesso a pypi.org, che è determinato dalla configurazione del tuo VPC. Si notino i requisiti seguenti:

1. Il requisito di installare moduli python aggiuntivi tramite pip install per la sessione di un utente. Se la sessione utilizza un connettore, la configurazione potrebbe risentirne.
2. Quando viene utilizzato un connettore con `--additional-python-modules`, all'avvio della sessione la sottorete associata al connettore `PhysicalConnectionRequirements` deve fornire un percorso di rete per raggiungere pypi.org.
3. È necessario determinare se la configurazione è corretta o meno.

Configurazione di un bucket Amazon S3 per ospitare un repository PyPI/Simple mirato

Questo esempio configura un mirror PyPI in Amazon S3 per un set di pacchetti e le relative dipendenze.

Per configurare il mirror PyPI per un set di pacchetti:

```
# pip download all the dependencies
pip download -d s3pypi --only-binary :all: plotly gglplot
pip download -d s3pypi --platform manylinux_2_17_x86_64 --only-binary :all: psycopg2-
binary
# create and upload the pypi/simple index and wheel files to the s3 bucket
s3pypi -b test-domain-name --put-root-index -v s3pypi/*
```

Se disponi già di un repository di artefatti esistente, esso avrà un URL di indice per l'utilizzo di pip che puoi fornire al posto dell'URL di esempio per il bucket Amazon S3 di cui sopra.

Per utilizzare l'index-url personalizzato, con alcuni pacchetti di esempio:

```
%%configure
{
    "--additional-python-modules": "psycopg2_binary==2.9.5",
    "python-modules-installer-option": "--no-cache-dir --verbose --index-url https://
test-domain-name.s3.amazonaws.com/ --trusted-host test-domain-name.s3.amazonaws.com"
}
```

Configurazione di un mirror CodeArtifact di pypi collegato al VPC

Per configurare un mirror:

1. Crea un repository nella stessa regione della sottorete usata dal connettore.

   Seleziona `Public upstream repositories` e scegli `pypi-store`.
2. Fornisci l'accesso al repository dal VPC per la sottorete.
3. Specifica il valore `--index-url` corretto utilizzando l'`python-modules-installer-option`.

```
%%configure
{
    "--additional-python-modules": "psycopg2_binary==2.9.5",
    "python-modules-installer-option": "--no-cache-dir --verbose --index-url https://
test-domain-name.s3.amazonaws.com/ --trusted-host test-domain-name.s3.amazonaws.com"
}
```

Per ulteriori informazioni, consulta la pagina Use CodeArtifact from a VPC.

Configurazione di DNS nel VPC

Domain Name System (DNS) è uno standard che consente di risolvere i nomi utilizzati su Internet nei corrispondenti indirizzi IP. Un nome host DNS assegna un nome a un computer in modo univoco ed è costituito da un nome host e un nome di dominio. I server DNS risolvono i nomi host DNS nei corrispondenti indirizzi IP.

Per configurare il DNS nel VPC, accertarsi che i nomi host DNS e la risoluzione DNS siano abilitati nel VPC. Gli attributi di rete VPC `enableDnsHostnames` e `enableDnsSupport` devono essere impostati su `true`. Per visualizzare e modificare questi attributi, accedi alla console VPC all'indirizzo https://console.aws.amazon.com/vpc/.

Per ulteriori informazioni, consulta Utilizzo del DNS con il tuo VPC. Inoltre, è possibile utilizzare il comando AWS CLI e chiamare il comando modify-vpc-attribute per configurare gli attributi della rete VPC.

> **Note**
>
> Se usi Route 53, verifica che la configurazione non sostituisca gli attributi di rete DNS.

Configurazione della crittografia in AWS Glue

L'esempio di flusso di lavoro seguente evidenzia le opzioni da configurare quando si usa la crittografia con AWS Glue. L'esempio illustra l'uso di chiavi AWS Key Management Service (AWS KMS) specifiche, ma puoi scegliere altre impostazioni in base alle esigenze specifiche. Questo flusso di lavoro mette in evidenza solo le opzioni della configurazione di AWS Glue relative alla crittografia.

1. Se l'utente della console AWS Glue non usa una policy di autorizzazioni che permette tutte le operazioni API AWS Glue (ad esempio, `"glue:*"`), verificaì che siano permesse le operazioni seguenti:
   - `"glue:GetDataCatalogEncryptionSettings"`
   - `"glue:PutDataCatalogEncryptionSettings"`
   - `"glue:CreateSecurityConfiguration"`
   - `"glue:GetSecurityConfiguration"`
   - `"glue:GetSecurityConfigurations"`
   - `"glue:DeleteSecurityConfiguration"`
2. Qualsiasi client che accede o scrive in un catalogo crittografato, ovvero qualsiasi utente della console, crawler, processo o endpoint di sviluppo, necessita delle autorizzazioni seguenti.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "<key-arns-used-for-data-catalog>"
   }
}
```

3. Qualsiasi utente o ruolo che accede a una password di connessione crittografata necessita delle autorizzazioni seguenti.

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

```
      "Effect": "Allow",
      "Action": [
            "kms:Decrypt"
           ],
      "Resource": "<key-arns-used-for-password-encryption>"
           }
}
```

4. Il ruolo di qualsiasi processo di estrazione, trasformazione e caricamento (ETL) che scrive dati crittografati in Amazon S3 necessita delle autorizzazioni seguenti.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:Decrypt",
           "kms:Encrypt",
           "kms:GenerateDataKey"
       ],
     "Resource": "<key-arns-used-for-s3>"
   }
}
```

5. Qualsiasi crawler o processo ETL che scrive voci di File di log Amazon CloudWatch crittografate, necessita delle autorizzazioni seguenti nella policy delle chiavi e nelle policy IAM.

   Nella policy della chiave (non nella policy IAM):

```
{
  "Effect": "Allow",
  "Principal": {
   "Service": "logs.region.amazonaws.com"
  },
  "Action": [
   "kms:Encrypt*",
   "kms:Decrypt*",
   "kms:ReEncrypt*",
   "kms:GenerateDataKey*",
   "kms:Describe*"
  ],
  "Resource": "<arn of key used for ETL/crawler cloudwatch encryption>"
```

```
 }
```

Per ulteriori informazioni sulle policy delle chiavi , consulta Utilizzo delle policy delle chiavi in AWS KMS nella Guida per gli sviluppatori di AWS Key Management Service.

Nella policy IAM collega l'autorizzazione `logs:AssociateKmsKey`:

```
{
  "Effect": "Allow",
  "Principal": {
   "Service": "logs.region.amazonaws.com"
  },
  "Action": [
   "logs:AssociateKmsKey"
  ],
  "Resource": "<arn of key used for ETL/crawler cloudwatch encryption>"
 }
```

6. Un processo ETL che usa un segnalibro di processo crittografato necessita delle autorizzazioni seguenti.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "<key-arns-used-for-job-bookmark-encryption>"
   }
}
```

7. Nella console AWS Glue scegliere Settings (Impostazioni) nel riquadro di navigazione.
   a. Nella pagina Data catalog settings (Impostazioni del catalogo dati) crittografare il catalogo dati selezionando la casella di controllo Metadata encryption (Crittografia dei metadati). Questa opzione permette di crittografare tutti gli oggetti nel catalogo dati con la chiave AWS KMS scelta.
   b. Per la chiave AWS KMS, scegliere aws/glue. Puoi anche scegliere una chiave AWS KMS creata da te.

> **Important**
>
> AWS Glue supporta solo chiavi master del cliente simmetriche (CMK). L'elenco di chiavi AWS KMS mostra solo le chiavi simmetriche. Tuttavia, selezionando Choose a AWS KMS key ARN (Scegli un ARN chiave), la console consente di inserire un ARN per qualsiasi tipo di chiave. Inserisci solo ARN per le chiavi simmetriche.

Quando la crittografia è abilitata, il client che accede al catalogo dati deve disporre delle autorizzazioni per AWS KMS.

8. Nel riquadro di navigazione scegliere Security configurations (Configurazioni di sicurezza). Una configurazione della sicurezza è un set di proprietà di sicurezza che è possibile usare per configurare i processi di AWS Glue. Scegliere quindi Add security configuration (Aggiungi configurazione di sicurezza). Nella configurazione scegliere tra le opzioni seguenti:

   a. Selezionare la crittografia S3. Per Encryption mode (Modalità crittografia), scegliere SSE-KMS. Per AWS KMS key (Chiave AWS KMS), scegliere aws/s3 (verificare che l'utente disponga dell'autorizzazione per usare questa chiave). In questo modo i dati scritti dal processo in Amazon S3 possono usare la chiave AWS Glue AWS KMS gestita da AWS.

   b. Selezionare Crittografia dei log di CloudWatch e scegliere una CMK (Assicurarsi che l'utente disponga dell'autorizzazione per utilizzare questa chiave). Per ulteriori informazioni, consulta Crittografia dei dati di log in CloudWatch Logs utilizzando AWS KMS nella Guida per l'utente di AWS Key Management Service.

      > **Important**
      >
      > AWS Glue supporta solo chiavi master del cliente simmetriche (CMK). L'elenco di chiavi AWS KMS mostra solo le chiavi simmetriche. Tuttavia, selezionando Choose a AWS KMS key ARN (Scegli un ARN chiave), la console consente di inserire un ARN per qualsiasi tipo di chiave. Inserisci solo ARN per le chiavi simmetriche.

   c. Scegliere Advanced properties (Proprietà avanzate) e selezionare la casella di controllo Job bookmark encryption (Crittografia segnalibro del processo). Per AWS KMS key (Chiave AWS KMS), scegliere aws/glue (verificare che l'utente disponga dell'autorizzazione per usare questa chiave). Ciò permette la crittografia dei segnalibri dei processi scritti in Amazon S3 con la chiave AWS Glue AWS KMS.

9. Nel riquadro di navigazione, scegli Connections (Connessioni).
   a. Scegliere Add connection (Aggiungi connessione) per creare una connessione al datastore JDBC (Java Database Connectivity) di destinazione del processo ETL.
   b. Per applicare la crittografia SSL (Secure Sockets Layer), selezionare la casella di controllo Require SSL connection (Richiedi connessione SSL) e testare la connessione.
10. Nel riquadro di navigazione scegliere Jobs (Processi).
    a. Scegliere Add job (Aggiungi processo) per creare un processo che trasforma i dati.
    b. Nella definizione del processo scegliere la configurazione della sicurezza creata.
11. Nella console AWS Glue eseguire il processo on demand. Verificare che i dati Amazon S3 scritti dal processo, le voci di CloudWatch Logs scritte dal processo e i segnalibri del processo siano tutti crittografati.

Creazione di reti per lo sviluppo per AWS Glue

Per eseguire gli script di estrazione, trasformazione e caricamento (ETL) con AWS Glue, puoi sviluppare e testare gli script usando un endpoint di sviluppo. Gli endpoint di sviluppo non sono supportati per l'utilizzo con i processi AWS Glue versione 2.0. Per le versioni 2.0 e successive, il metodo di sviluppo preferito è l'utilizzo di Jupyter Notebook con uno dei kernel AWS Glue. Per ulteriori informazioni, consulta the section called “Nozioni di base sulle sessioni interattive AWS Glue”.

Impostazione della rete per un endpoint di sviluppo

Quando imposti un endpoint di sviluppo, specifichi un Virtual Private Cloud (VPC), una sottorete e i gruppi di sicurezza.

> **Note**
>
> Assicurati di configurare l'ambiente DNS per AWS Glue. Per ulteriori informazioni, consulta Configurazione di DNS nel VPC.

Per permettere a AWS Glue di accedere alle risorse necessarie, aggiungi una riga nella tabella di routing della sottorete per associare un elenco di prefissi per Amazon S3 all'endpoint VPC. È necessario un ID elenco prefisso per la creazione di una regola del gruppo di sicurezza in uscita

che consenta al traffico da un VPC di accedere a un servizio AWS tramite un endpoint VPC. Per semplificare la connessione a un server notebook associato a questo endpoint di sviluppo, dal computer locale, aggiungi una riga alla tabella di routing per aggiungere un ID Internet Gateway. Per ulteriori informazioni, consulta Endpoint VPC. Aggiorna la tabella di routing della sottorete in modo simile alla tabella seguente:

Destinazione	Target		
10.0.0.0/16	locale		
pl-id per Amazon S3	vpce-id		
0.0.0.0/0	igw-xxxx		

Per permettere la comunicazione dei componenti di AWS Glue, specifica un gruppo di sicurezza con una regola per il traffico in entrata autoreferenziale per tutte le porte TCP. Creando una regola autoreferenziale, puoi limitare l'origine allo stesso gruppo di sicurezza del VPC senza essere aperta a tutte le reti. Il gruppo di sicurezza predefinito per il tuo VPC potrebbe già avere una regola autoreferenziata in entrata per ALL Traffic.

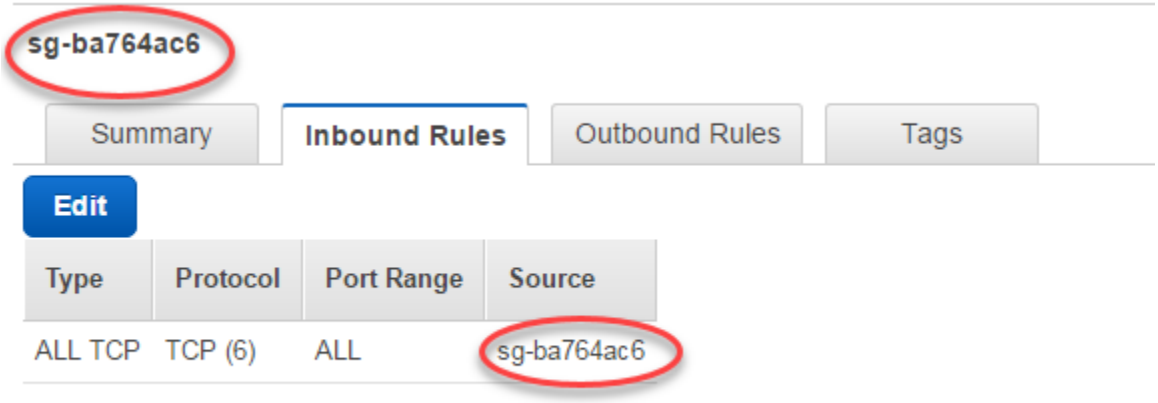
Per configurare un gruppo di sicurezza

1. Accedi a AWS Management Console e apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel riquadro di navigazione a sinistra, scegli Security Groups (Gruppi di sicurezza).
3. Scegli un gruppo di sicurezza esistente dall'elenco o Create Security Group (Crea gruppo di sicurezza) da usare con l'endpoint di sviluppo.
4. Nel riquadro del gruppo di sicurezza, passa alla scheda Inbound (In entrata).
5. Aggiungi una regola autoreferenziale per permettere ai componenti di AWS Glue di comunicare tra loro. In particolare, aggiungi o verifica che sia presente una regola con Type (Tipo) `All TCP`, Protocol (Protocollo) `TCP`, Port Range (Intervallo porte) che include tutte le porte e Source (Origine) corrispondente al nome del gruppo di sicurezza indicato da Group ID (ID gruppo).

   La regola in entrata è simile alla seguente:

Type (Tipo)	Protocollo	Intervallo porte	Origine
Tutte le regole TCP	TCP	0–65535	*security-group*

Il seguente è un esempio di regola in entrata autoreferenziale:



6. Aggiungi una regola anche per il traffico in uscita. Apri il traffico in uscita a tutte le porte o crea una regola autoreferenziale di Type (Tipo) `All TCP`, con Protocol (Protocollo) `TCP` e Port Range (Intervallo porte) che includa tutte le porte, la cui Source (Origine) abbia lo stesso nome del gruppo di sicurezza di Group ID (ID gruppo).

   La regola in uscita è simile a una delle seguenti regole:

Type (Tipo)	Protocollo	Intervallo porte	Destinazione
Tutte le regole TCP	TCP	0–65535	*security-group*
All Traffic	ALL	ALL	0.0.0.0/0

Impostazione di Amazon EC2 per un server notebook

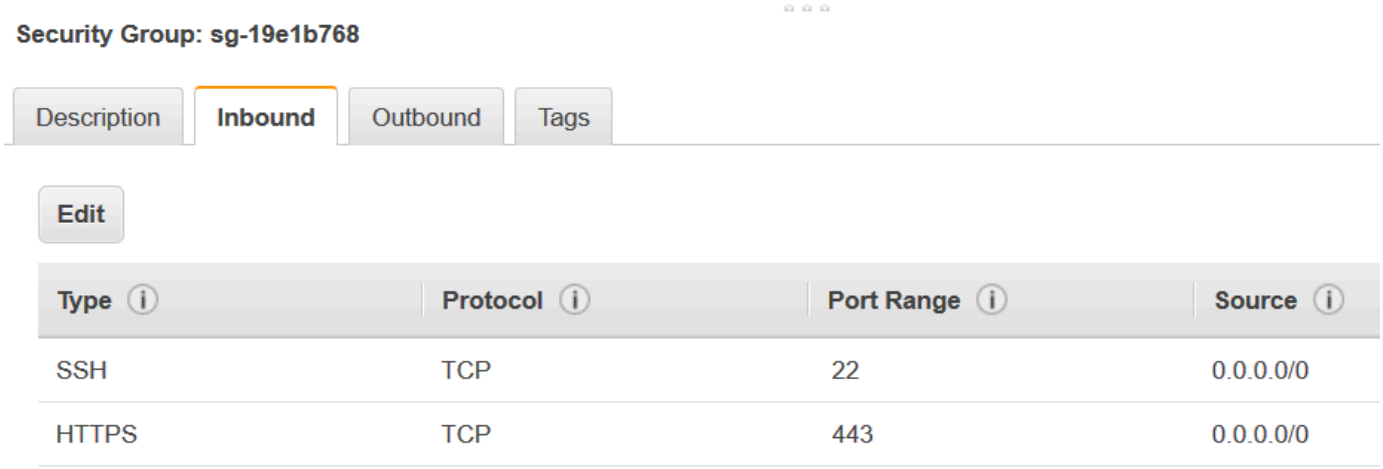
Con un endpoint di sviluppo, puoi creare un server notebook per testare gli script ETL con i notebook Jupyter. Per abilitare la comunicazione con il notebook, specifica un gruppo di sicurezza con regole in entrata per HTTPS (porta 443) e SSH (porta 22). Verifica che l'origine della regola sia 0.0.0.0/0 o l'indirizzo IP del computer che si collega al notebook.

Per configurare un gruppo di sicurezza

1. Accedi a AWS Management Console e apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel riquadro di navigazione a sinistra, scegli Security Groups (Gruppi di sicurezza).
3. Scegli un gruppo di sicurezza esistente dall'elenco o Create Security Group (Crea gruppo di sicurezza) da usare con il server notebook. Il gruppo di sicurezza associato al tuo endpoint di sviluppo viene utilizzato anche per creare il server notebook.
4. Nel riquadro del gruppo di sicurezza, passa alla scheda Inbound (In entrata).
5. Aggiungi le regole in entrata simili alla seguente:

Type (Tipo)	Protocollo	Intervallo porte	Origine
SSH	TCP	22	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

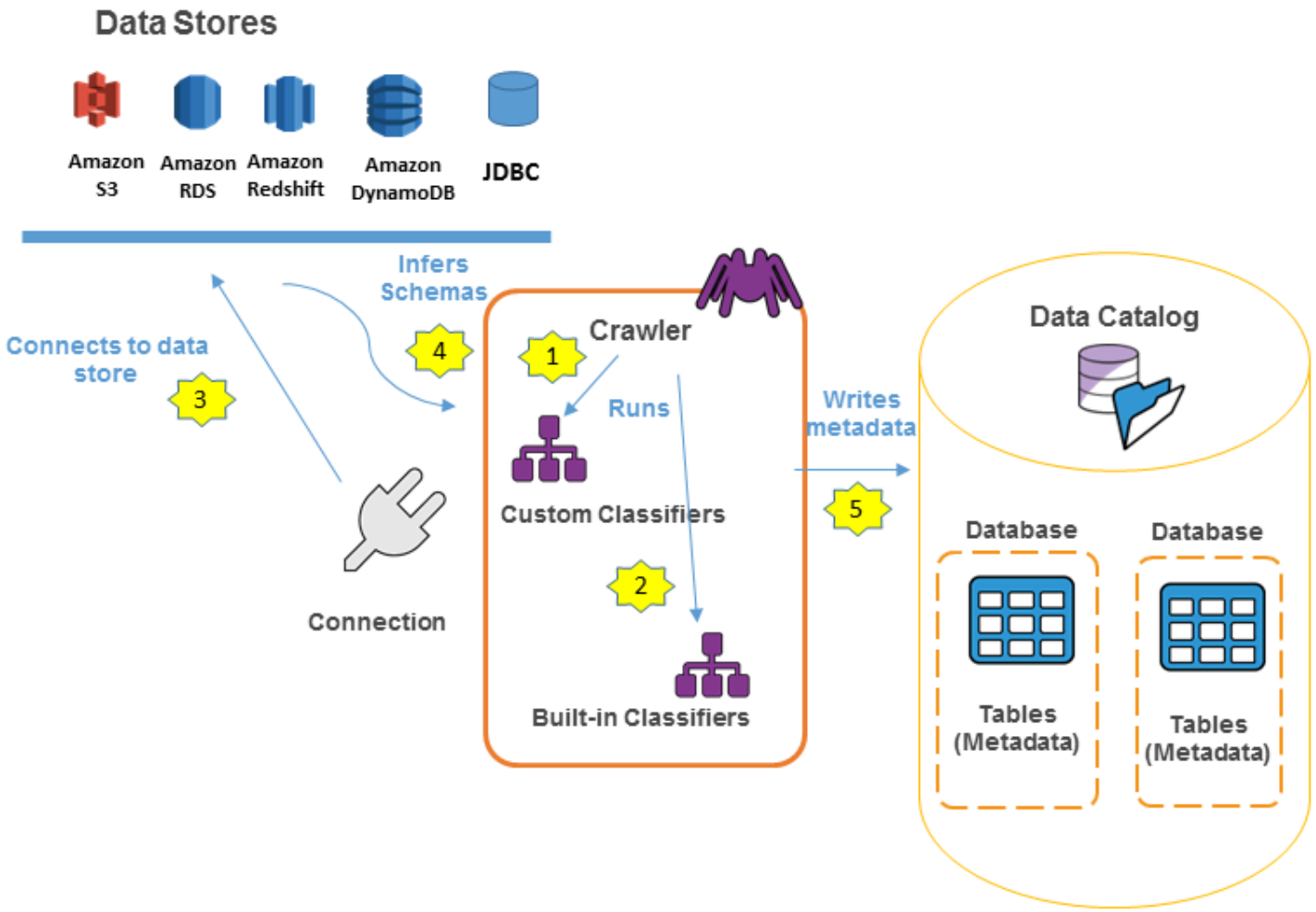
Di seguito è riportato un esempio di regole in entrata per il gruppo di sicurezza:



Catalogo dati e crawler in AWS Glue

Il AWS Glue Data Catalog contiene riferimenti ai dati usati come origini e destinazioni dei processi di estrazione, trasformazione e caricamento (ETL, Extract, Transform and Load) in AWS Glue. Per creare il data warehouse o il data lake, è necessario catalogare questi dati. Il AWS Glue Data Catalog è un indice per i parametri di posizione, schema e runtime dei dati. È possibile utilizzare le informazioni presenti nel catalogo dati per creare e monitorare i processi ETL. Le informazioni nel catalogo vengono memorizzate come tabelle di metadati, dove ogni tabella specifica un singolo datastore. In genere, per richiedere l'inventario dei dati contenuti nel datastore, si esegue un crawler. Tuttavia, è possibile aggiungere tabelle di metadati nel catalogo dati in altri modi. Per ulteriori informazioni, consulta Tabelle AWS Glue.

Il diagramma del flusso di lavoro seguente mostra in che modo i crawler AWS Glue interagiscono con i datastore e altri elementi per popolare il catalogo dati.



Di seguito è riportato il flusso di lavoro generale che descrive il modo in cui il crawler popola il AWS Glue Data Catalog:

1. Un crawler esegue qualsiasi classificatore personalizzato da te selezionato per dedurre il formato e lo schema dei dati. É possibile fornire il codice per i classificatori personalizzati, i quali vengono eseguiti nell'ordine specificato.

   Il primo classificatore personalizzato che riconosce in modo corretto la struttura dei dati viene utilizzato per creare uno schema. I classificatori personalizzati in basso nell'elenco vengono ignorati.
2. Se nessun classificatore personalizzato è in grado di abbinar lo schema dei dati, i classificatori integrati ne tenteranno il riconoscimento. Un esempio di un classificatore incorporato è quello che riconosce JSON.

3. Il crawler si collega al datastore. Alcuni datastore richiedono proprietà di connessione per l'accesso al crawler.
4. Lo schema dedotto dei dati viene creato.
5. Il crawler scrive metadati nel catalogo dati. Una definizione di tabella contiene i metadati sui dati presenti nel datastore. La tabella viene scritta in un database, che è un container di tabelle nel catalogo dati. Gli attributi di una tabella includono la classificazione, ovvero un'etichetta creata dal classificatore che ha dedotto lo schema della tabella.

Argomenti

Database AWS Glue

I database vengono utilizzati per organizzare le tabelle dei metadati nella AWS Glue. Quando definisci una tabella nel AWS Glue Data Catalog, la aggiungi a un database. Una tabella può essere in un solo database.

Il tuo database può contenere tabelle che definiscono dati provenienti da datastore diversi. Questi dati possono includere oggetti in Amazon Simple Storage Service (Amazon S3) e tabelle relazionali in Amazon Relational Database Service.

> **Note**
>
> Quando elimini un database da AWS Glue Data Catalog, tutte le tabelle in esso contenute vengono eliminate.

Per ulteriori informazioni sulla definizione di un database tramite la console AWS Glue, consulta Uso di database nella console AWS Glue.

Collegamenti di risorsa al database

> La console AWS Glue è stata recentemente aggiornata. La versione corrente della console non supporta i collegamenti di risorsa al database.

Il catalogo dati può anche contenere collegamenti di risorsa ai database. Un collegamento di risorsa al database è un collegamento a un database locale o condiviso. Al momento, puoi creare collegamenti di risorsa solo in AWS Lake Formation. Dopo aver creato un collegamento di risorsa a un database, è possibile utilizzare il nome del collegamento di risorsa ovunque desideri utilizzare il nome del database. Insieme ai database di cui si è proprietari o che sono condivisi con l'utente, i collegamenti di risorsa al database vengono restituiti da `glue:GetDatabases()` e vengono visualizzati come voci nella pagina Database della console AWS Glue.

Il catalogo dati può anche contenere collegamenti di risorsa alla tabella.

Per ulteriori informazioni sui collegamenti di risorsa, consulta Creazione di collegamenti di risorsa nella Guida per gli sviluppatori di AWS Lake Formation.

Uso di database nella console AWS Glue

Un database nel AWS Glue Data Catalog è un container che contiene tabelle. Puoi utilizzare i database per organizzare le tabelle in categorie separate. I database vengono creati quando esegui un crawler o aggiungi una tabella manualmente. L'elenco dei database nella console AWS Glue visualizza le descrizioni di tutti i database.

Per visualizzare l'elenco dei database, accedi alla AWS Management Console e apri la console AWS Glue su https://console.aws.amazon.com/glue/. Scegli Databases (Database) e quindi un nome di database nell'elenco per visualizzarne i dettagli.

Nella scheda Databases (Database) nella console AWS Glue puoi aggiungere, modificare ed eliminare database:

- Per creare un nuovo database, scegli Add database (Aggiungi database) e fornisci un nome e una descrizione. Per compatibilità con altri store di metadati, ad esempio Apache Hive, il nome è in caratteri minuscoli.

> **Note**
>
> Se prevedi di accedere al database da Amazon Athena, fornisci un nome con solo caratteri alfanumerici e di sottolineatura. Per ulteriori informazioni, consulta Nomi di Athena.

- Per modificare la descrizione di un database, seleziona la casella di controllo accanto al nome del database e scegli Edit (Modifica).
- Per eliminare un database, seleziona la casella di controllo accanto al nome del database e scegli Remove (Rimuovi).
- Per visualizzare l'elenco delle tabelle contenute nel database, scegli il nome del database e le proprietà del database mostreranno tutte le tabelle.

Per modificare il database in un crawler scrive devi modificare la definizione del crawler. Per ulteriori informazioni, consulta Definizione di crawler in AWS Glue.

Tabelle AWS Glue

Puoi aggiungere definizioni di tabella nel catalogo dati in questi modi:

- Eseguendo un crawler che si connette a uno o più datastore, determina le strutture dei dati e scrive tabelle nel catalogo dati. Il crawler utilizza classificatori incorporati o personalizzati per riconoscere la struttura dei dati. È possibile eseguire il crawler su pianificazione. Per ulteriori informazioni, consulta Definizione di crawler in AWS Glue.
- Usando la console AWS Glue per creare manualmente una tabella nel AWS Glue Data Catalog. Per ulteriori informazioni, consulta Utilizzo di tabelle nella console AWS Glue.
- Usando l'operazione `CreateTable` nell'API AWS Glue per creare una tabella nel AWS Glue Data Catalog Per ulteriori informazioni, consulta Operazione CreateTable (Python: create_table).
- Usando i modelli AWS CloudFormation. Per ulteriori informazioni, consulta AWS CloudFormation per AWS Glue.
- Eseguendo la migrazione di un metastore Apache Hive. Per ulteriori informazioni, consulta Migrazione tra Hive Metastore e così via. AWS Glue Data Catalog GitHub

Quando definisci manualmente una tabella utilizzando la console o un'API, specifichi lo schema della tabella e il valore di un campo di classificazione che indica il tipo e il formato dei dati nell'origine dati.

Se un crawler crea la tabella, lo schema e il formato dei dati sono determinati da un classificatore incorporato o da un classificatore personalizzato. Per ulteriori informazioni sulla creazione di una tabella tramite la console AWS Glue, consulta Utilizzo di tabelle nella console AWS Glue.

Argomenti

Partizioni tabella

Un a definizione di tabella AWS Glue di una cartella Amazon Simple Storage Service (Amazon S3) può descrivere una tabella partizionata. Ad esempio, per migliorare le prestazioni delle query, una tabella partizionata potrebbe separare i dati mensili in diversi file utilizzando il nome del mese come chiave. Le definizioni di tabella in AWS Glue includono la chiave di partizionamento di una tabella. Quando AWS Glue valuta i dati nelle cartelle Amazon S3 per catalogare una tabella, determina se viene aggiunta una singola tabella o una tabella partizionata.

È possibile creare indici delle partizioni su una tabella per recuperare un sottoinsieme delle partizioni invece di caricare tutte le partizioni nella tabella. Per ulteriori informazioni sull'utilizzo degli indici delle partizioni, consulta Utilizzo degli indici delle partizioni in AWS Glue.

Perché AWS Glue possa creare una tabella partizionata per una cartella Amazon S3, devono essere vere tutte le condizioni seguenti:

- Gli schemi dei file sono simili, come stabilito da AWS Glue.
- Il formato dati dei file è lo stesso.
- Il formato di compressione dei file è lo stesso.

Ad esempio, puoi avere un tuo bucket Amazon S3 denominato `my-app-bucket`, in cui vengono memorizzati i dati di vendita delle app iOS e Android. I dati sono partizionati in base ad anno, mese e giorno. I file di dati per le vendite iOS e Android hanno lo stesso schema, formato dei dati e formato di

compressione. Nel AWS Glue Data Catalog il crawler AWS Glue crea una definizione di tabella con chiavi di partizionamento per anno, mese e giorno.

Il seguente elenco Amazon S3 di `my-app-bucket` mostra alcune delle partizioni. Il simbolo = viene utilizzato per assegnare i valori di chiave di partizione.

```
my-app-bucket/Sales/year=2010/month=feb/day=1/iOS.csv
my-app-bucket/Sales/year=2010/month=feb/day=1/Android.csv
my-app-bucket/Sales/year=2010/month=feb/day=2/iOS.csv
my-app-bucket/Sales/year=2010/month=feb/day=2/Android.csv
...
my-app-bucket/Sales/year=2017/month=feb/day=4/iOS.csv
my-app-bucket/Sales/year=2017/month=feb/day=4/Android.csv
```

Collegamenti di risorsa della tabella

La console AWS Glue è stata recentemente aggiornata. La versione corrente della console non supporta i collegamenti di risorsa alla tabella.

Il catalogo dati può anche contenere collegamenti di risorsa della tabella. Un collegamento di risorsa della tabella è un collegamento a un database locale o condiviso. Al momento, puoi creare collegamenti di risorsa solo in AWS Lake Formation. Dopo aver creato un collegamento di risorsa a una tabella, è possibile utilizzare il nome del collegamento di risorsa ovunque desideri utilizzare il nome della tabella. Insieme alle tabelle di cui si è proprietari o che sono condivisi con l'utente, i collegamenti di risorsa alla tabella vengono restituiti da `glue:GetTables()` e vengono visualizzati come voci nella pagina Tables (Tabelle) della console AWS Glue.

Il catalogo dati può anche contenere collegamenti di risorsa ai database.

Per ulteriori informazioni sui collegamenti di risorsa, consulta Creazione di collegamenti di risorsa nella Guida per gli sviluppatori di AWS Lake Formation.

Aggiornamento delle tabelle del catalogo dati create manualmente usando i crawler

Puoi creare le tabelle del AWS Glue Data Catalog manualmente e quindi tenerle aggiornati con i crawler AWS Glue. I crawler in esecuzione su una pianificazione possono aggiungere nuove

partizioni e aggiornare le tabelle con qualsiasi modifica dello schema. Questo vale anche per le tabelle migrate da un metastore Apache Hive.

Per fare ciò, quando definisci un crawler, invece di specificare uno o più datastoe come origine di un crawling, puoi specificare una o più tabelle del catalogo dati esistenti. Il crawler esegue quindi il crawling dei datastore specificati dalle tabelle del catalogo. In questo caso, non vengono create nuove tabelle mentre le tabelle create manualmente vengono aggiornate.

Di seguito sono riportati altri motivi per cui puoi creare manualmente le tabelle di catalogo e specificarle come origini del crawler:

- Desideri scegliere il nome della tabella del catalogo e non fare affidamento sull'algoritmo di denominazione della tabella del catalogo.
- Desideri impedire la creazione di nuove tabelle nel caso in cui i file con un formato che potrebbe interrompere il rilevamento della partizione vengano erroneamente salvati nel percorso dell'origine dati.

Per ulteriori informazioni, consulta Tipo di origine del crawler.

Proprietà della tabella del catalogo dati

Le proprietà delle tabelle, o parametri, così note nella CLI di AWS, sono stringhe di chiavi e valori non convalidate. È possibile impostare le proprie proprietà sulla tabella per supportare gli usi del catalogo dati all'esterno di AWS Glue. Anche altri servizi che utilizzano il catalogo dati possono farlo. AWS Glue imposta alcune proprietà delle tabelle durante l'esecuzione di processi o crawler. Salvo diversa indicazione, queste proprietà sono per uso interno, non supportiamo il fatto che continuino a esistere nella loro forma attuale o che supportino il comportamento del prodotto se queste proprietà vengono modificate manualmente.

Per ulteriori informazioni sulle proprietà delle tabelle impostate dai crawler AWS Glue, consulta the section called “Parametri impostati sulle tabelle del catalogo dati dal crawler”.

Utilizzo di tabelle nella console AWS Glue

Una tabella nel AWS Glue Data Catalog è la definizione dei metadati che rappresenta i dati in un datastore. Puoi creare tabelle quando esegui un crawler oppure puoi creare una tabella manualmente nella console AWS Glue. L'elenco Tables (Tabelle) nella console AWS Glue mostra i valori dei metadati della tabella. Usa le definizioni di tabella per specificare origini e destinazioni al momento della creazione di processi ETL (estrazione, trasformazione e caricamento).

> **Note**
>
> Alla luce delle recenti modifiche alla console di gestione AWS, potrebbe essere necessario modificare i ruoli IAM esistenti per ottenere l'autorizzazione `SearchTables`. Per la creazione di nuovi ruoli, l'autorizzazione dell'API `SearchTables` è già stata aggiunta come impostazione predefinita.

Per iniziare, accedi alla AWS Management Console e apri la console AWS Glue su https://console.aws.amazon.com/glue/. Scegli la scheda Tables (Tabelle) e usa il pulsante Add tables (Aggiungi tabelle) per creare tabelle con un crawler o digitando manualmente gli attributi.

Aggiunta di tabelle nella console

Per usare un crawler per aggiungere tabelle, scegli Add tables (Aggiungi tabelle), Add tables using a crawler (Aggiungi tabelle utilizzando un crawler). Quindi segui le istruzioni nella procedura guidata Add crawler (Aggiungi crawler). Quando il crawler viene eseguito, vengono aggiunte tabelle al AWS Glue Data Catalog. Per ulteriori informazioni, consulta Definizione di crawler in AWS Glue.

Se conosci gli attributi necessari per creare una definizione di tabella Amazon Simple Storage Service (Amazon S3) nel catalogo dati, puoi crearla con la procedura guidata di creazione di tabelle. Scegli Add tables (Aggiungi tabelle), Add table manually (Aggiungi tabella manualmente) e segui le istruzioni della procedura guidata Add tables (Aggiungi tabella).

Quando aggiungi una tabella manualmente attraverso la console, considera quanto segue:

- Se prevedi di accedere alla tabella da Amazon Athena, fornisci un nome con solo caratteri alfanumerici e di sottolineatura. Per ulteriori informazioni, consulta Nomi di Athena.
- L'ubicazione dei dati di origine deve essere un percorso Amazon S3.
- Il formato dei dati deve corrispondere a uno dei formati elencati nella procedura guidata. La classificazione corrispondente e SerDe le altre proprietà della tabella vengono compilate automaticamente in base al formato scelto. Puoi definire tabelle con i seguenti formati:

  Avro

  Formato binario Apache Avro JSON.

CSV

Character separated values. Puoi anche specificare il delimitatore di virgola, barra verticale, punto e virgola, tab o Ctrl-A.

JSON

JavaScript Notazione degli oggetti.

XML

Formato Extensible Markup Language. Specifica il tag XML che definisce una riga nei dati. Le colonne sono definite all'interno di tag di riga.

Parquet

Storage a colonne Apache Parquet.

ORC

Formato di file Optimized Row Columnar (ORC). Un formato progettato per archiviare in modo efficiente i dati Hive.

- Puoi definire una chiave di partizione per la tabella.
- Al momento, le tabelle partizionate che crei con la console non possono essere utilizzate in processi ETL.

Attributi della tabella

Di seguito sono elencati alcuni importanti attributi della tua tabella:

Nome

Il nome viene stabilito in fase di creazione della tabella e non può essere modificato. Dovrai fare riferimento a un nome di tabella in molte operazioni AWS Glue.

Database

L'oggetto container in cui la tabella risiede. Questo oggetto contiene un'organizzazione delle tabelle inclusa all'interno del AWS Glue Data Catalog e può differire dall'organizzazione nel datastore. Quando elimini un database, anche tutte le tabelle in esso contenute vengono eliminate dal catalogo dati.

Descrizione

La descrizione della tabella. Puoi scrivere una descrizione per aiutarti a comprendere i contenuti della tabella.

Formato della tabella

Specifica la creazione di una tabella AWS Glue standard o di una tabella in formato Apache Iceberg.

Abilita la compattazione

Scegli Abilita compattazione per compattare piccoli oggetti Amazon S3 nella tabella in oggetti più grandi.

Ruolo IAM

Per eseguire la compattazione, il servizio assume un ruolo IAM per tuo conto. Puoi scegliere un ruolo IAM utilizzando il menu a discesa. Assicurati che il ruolo disponga delle autorizzazioni necessarie per abilitare la compattazione.

Consulta Prerequisiti per l'ottimizzazione delle tabelle per ulteriori informazioni sulle autorizzazioni necessarie per il ruolo IAM.

Ubicazione

Il puntatore all'ubicazione dei dati in un datastore rappresentato da questa definizione di tabella.

Classificazione

Un valore di categorizzazione fornito al momento della creazione della tabella. Solitamente viene scritto quando un crawler viene eseguito e specifica il formato dei dati di origine.

Ultimo aggiornamento

Data e ora (UTC) in cui questa tabella è stata aggiornata nel catalogo dati.

Data aggiunta

Data e ora (UTC) in cui questa tabella è stata aggiunta al catalogo dati.

Deprecated

Se AWS Glue rileva che una tabella nel catalogo dati non è più presente nel datastore originale, la contrassegna come obsoleta nel catalogo dati. Se esegui un processo che fa riferimento a una tabella obsoleta, il processo potrebbe fallire. Modifica processi che fanno riferimento a tabelle obsolete per rimuoverle come origini e destinazioni. Consigliamo di eliminare le tabelle obsolete quando non sono più necessarie.

Connessione

Se AWS Glue richiede una connessione al datastore, il nome della connessione è associato alla tabella.

Visualizzazione e modifica dei dettagli della tabella

Per vedere i dettagli di una tabella esistente, scegli il nome tabella nell'elenco quindi scegli Action, View details (Operazione, Mostra dettagli).

I dettagli tabella includono le proprietà della tabella e del relativo schema. Questa vista mostra lo schema della tabella, inclusi i nomi colonna nell'ordine definito per la tabella, i tipi di dati e le colonne chiave per le partizioni. Se una colonna è di tipo complesso, puoi scegliere View properties (Visualizza proprietà) per visualizzare i dettagli della struttura di tale campo, come mostrato nell'esempio seguente:

```
{
"StorageDescriptor":
    {
      "cols": {
         "FieldSchema": [
           {
             "name": "primary-1",
             "type": "CHAR",
             "comment": ""
           },
           {
             "name": "second ",
             "type": "STRING",
             "comment": ""
           }
         ]
      },
      "location": "s3://aws-logs-111122223333-us-east-1",
      "inputFormat": "",
      "outputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
      "compressed": "false",
      "numBuckets": "0",
      "SerDeInfo": {
           "name": "",
           "serializationLib": "org.apache.hadoop.hive.serde2.OpenCSVSerde",
           "parameters": {
```

```
                "separatorChar": "|"
            }
        },
        "bucketCols": [],
        "sortCols": [],
        "parameters": {},
        "SkewedInfo": {},
        "storedAsSubDirectories": "false"
    },
    "parameters": {
       "classification": "csv"
    }
}
```

Per ulteriori informazioni sulle proprietà di una tabella, come `StorageDescriptor`, consulta Struttura StorageDescriptor.

Per modificare lo schema di una tabella, scegli Edit schema (Modifica schema) per aggiungere o rimuovere colonne, modificarne i nomi e modificare i tipi di dati.

Per confrontare diverse versioni di una tabella, incluso lo schema, scegli Confronta versioni per visualizzare un side-by-side confronto tra due versioni dello schema per una tabella. Per ulteriori informazioni, consulta Confronta le versioni dello schema della tabella .

Per visualizzare i file che costituiscono una partizione Amazon S3, scegli View partition (Visualizza partizione). Per tabelle Amazon S3, la colonna Key (Chiave) visualizza le chiavi di partizione usate per partizionare la tabella nel datastore di origine. Il partizionamento è un modo per dividere una tabella in parti correlate in base ai valori di una colonna chiave, ad esempio data, ubicazione o reparto. Per ulteriori informazioni sulle partizioni, cercare "hive partitioning." su Internet.

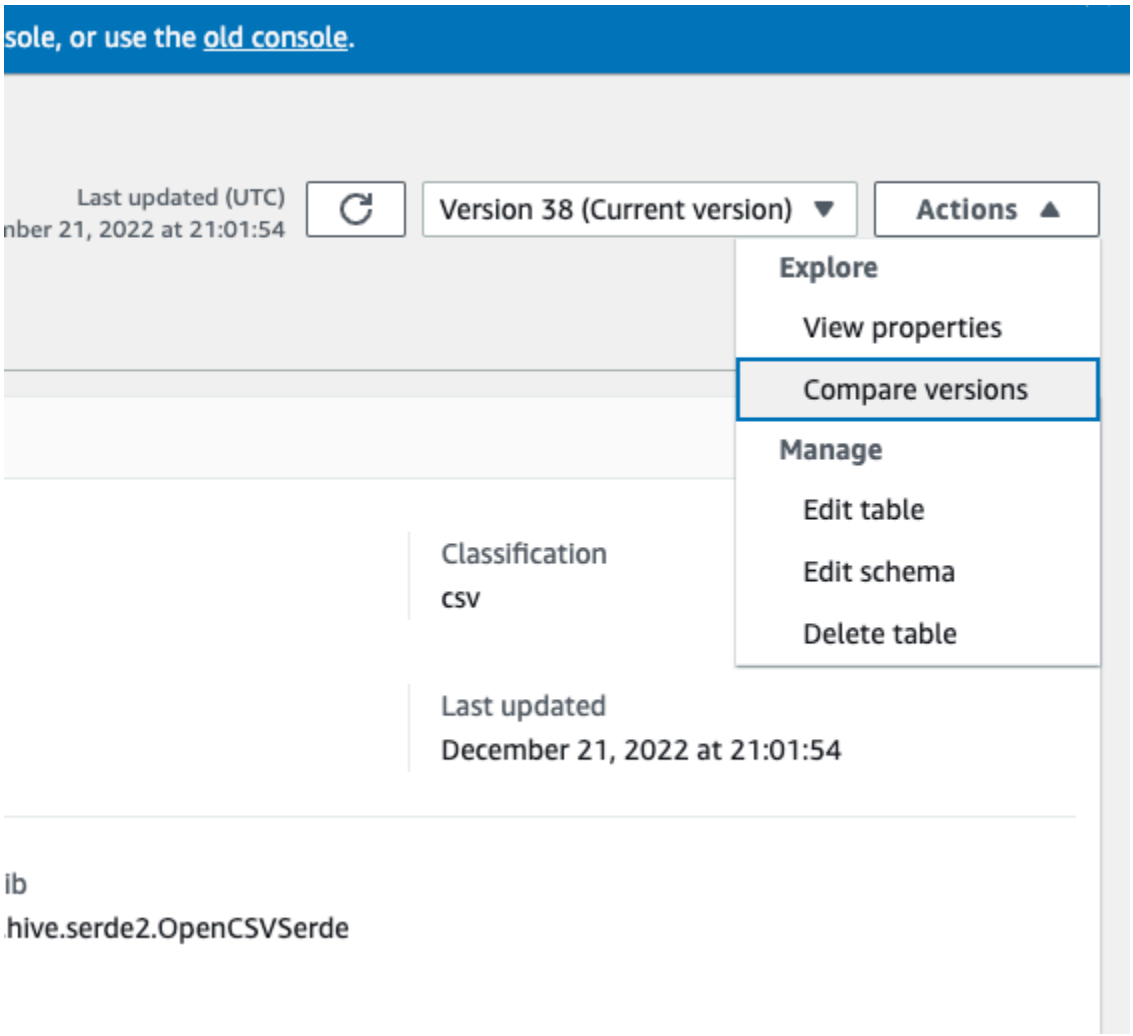
> **Note**
>
> Per ottenere step-by-step indicazioni sulla visualizzazione dei dettagli di una tabella, consulta il tutorial Esplora la tabella nella console.

Confronta le versioni dello schema della tabella

Quando si confrontano due versioni di schemi di tabelle, è possibile confrontare le modifiche delle righe nidificate espandendo e comprimendo le righe nidificate, confrontare gli schemi di due versioni e visualizzare le side-by-side proprietà delle tabelle. side-by-side

Come confrontare le versioni

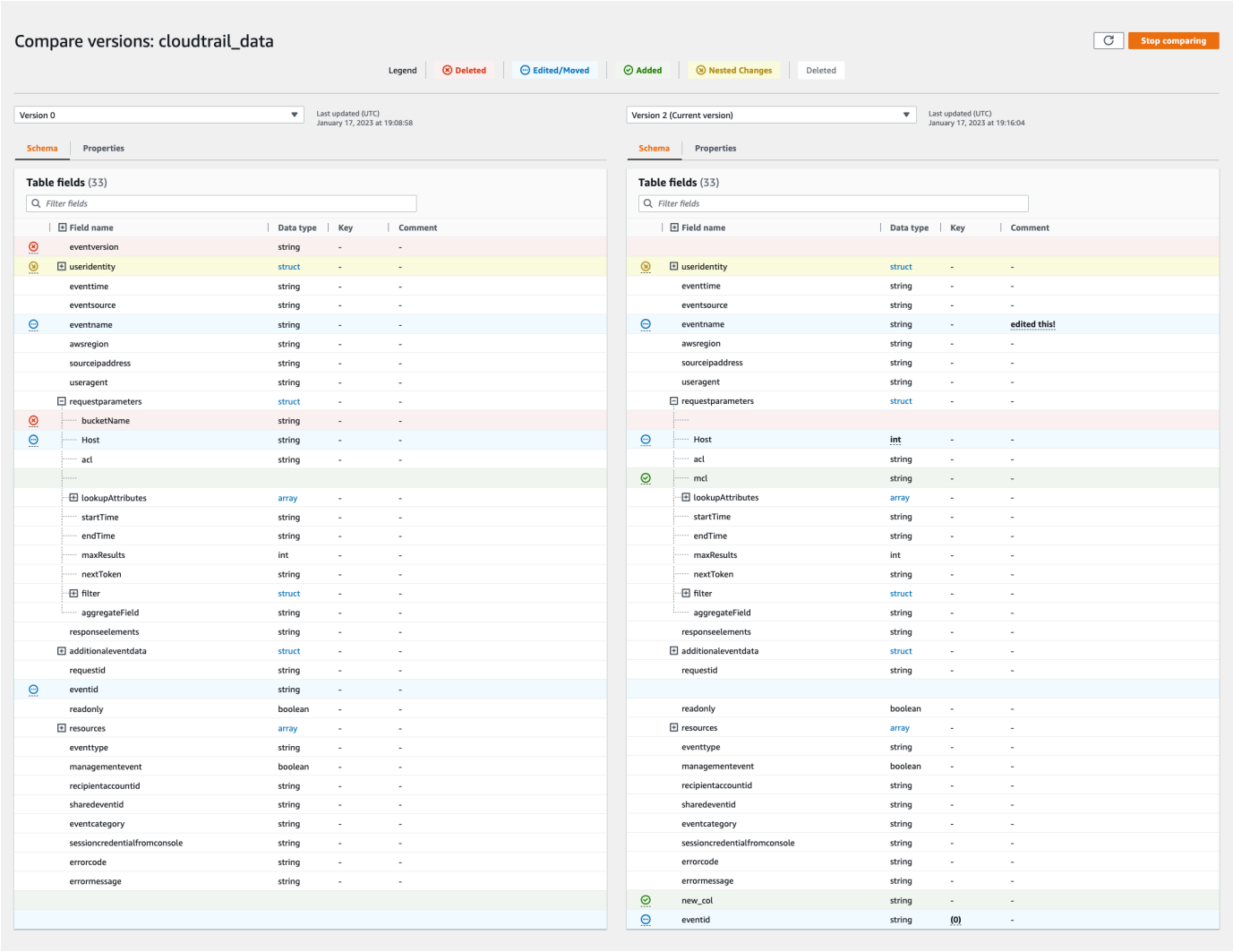
1. Dalla console AWS Glue, scegli Tabelle, quindi Azioni e scegli Confronta versioni.



2. Scegli una versione da confrontare scegliendo il menu a discesa delle versioni. Quando si confrontano gli schemi, la scheda Schema è evidenziata in arancione.
3. Quando si confrontano le tabelle tra due versioni, gli schemi delle tabelle vengono visualizzati sul lato sinistro e destro dello schermo. Ciò consente di determinare visivamente le modifiche confrontando il nome della colonna, il tipo di dati, la chiave e i campi di commento. side-by-side Quando viene apportata una modifica, un'icona colorata mostra il tipo di modifica apportata.

   - Eliminata: contrassegnata da un'icona rossa, indica dove la colonna è stata rimossa da una versione precedente dello schema della tabella.
   - Modificata o spostata: contrassegnata da un'icona blu, indica dove la colonna è stata modificata o spostata in una versione più recente dello schema della tabella.

- Aggiunta: contrassegnata da un'icona verde, indica dove la colonna è stata aggiunta a una versione più recente dello schema della tabella.
- Modifiche annidate: contrassegnata da un'icona gialla, indica le modifiche nella colonna annidata. Scegli la colonna da espandere e visualizza le colonne che sono state eliminate, modificate, spostate o aggiunte.



4. Utilizza la barra di ricerca dei campi di filtro per visualizzare i campi in base ai caratteri che inserisci qui. Se immetti un nome di colonna in una delle versioni della tabella, i campi filtrati vengono visualizzati in entrambe le versioni della tabella per mostrare dove sono state apportate le modifiche.
5. Per confrontare le proprietà, scegli la scheda Proprietà.
6. Per interrompere il confronto tra le versioni, scegli Interrompi confronto per tornare all'elenco delle tabelle.

Ottimizzazione delle tabelle Iceberg

I data lake Amazon S3 che utilizzano formati di tabelle aperte, come Apache Iceberg, archiviano i dati come oggetti Amazon S3. La presenza di migliaia di piccoli oggetti Amazon S3 in una tabella di data lake aumenta il sovraccarico dei metadati sulle tabelle Iceberg e influisce sulle prestazioni di lettura. Per migliorare le prestazioni di lettura tramite servizi di analisi AWS come Amazon Athena e Amazon EMR e processi AWS Glue ETL, AWS Glue Data Catalog offre la compattazione gestita (un processo che compatta piccoli oggetti Amazon S3 in oggetti più grandi) per le tabelle Iceberg in Catalogo dati. Puoi utilizzare la consoleAWS Glue, la console Lake Formation, AWS CLI o l'API AWS per abilitare o disabilitare la compattazione per le singole tabelle Iceberg presenti nel Catalogo dati.

L'ottimizzatore delle tabelle monitora costantemente le partizioni delle tabelle e avvia il processo di compattazione quando viene superata la soglia per il numero di file e le dimensioni dei file. Nel Catalogo dati, il valore di soglia predefinito per avviare la compattazione è impostato su 384 MB, mentre nella libreria Iceberg la soglia per la compattazione è di circa 75% rispetto alla dimensione del file di destinazione. Catalogo dati esegue la compattazione senza interferire con le query simultanee. Catalogo dati supporta la compattazione dei dati solo per le tabelle in formato Parquet.

Argomenti

Prerequisiti per l'ottimizzazione delle tabelle

L'ottimizzatore di tabelle assume le autorizzazioni del ruolo (IAM) AWS Identity and Access Management specificate quando si abilita la compattazione per una tabella. Il ruolo IAM deve disporre delle autorizzazioni per leggere i dati e aggiornare i metadati nel Catalogo dati. Puoi creare un ruolo IAM e collegare le seguenti policy in linea:

- Aggiungi la seguente policy in linea che concede le autorizzazioni di lettura/scrittura di Amazon S3 sulla posizione per i dati non registrati con Lake Formation. Questa politica include anche le autorizzazioni per aggiornare la tabella nel Catalogo dati e consentire a AWS Glue di aggiungere

log nei log Amazon CloudWatch e la pubblicazione di parametri. Per i dati di origine in Amazon S3 che non sono registrati con Lake Formation, l'accesso è determinato dalle policy di autorizzazione IAM per Amazon S3 e dalle operazioni AWS Glue.

Nelle seguenti policy in linea, sostituisci `bucket-name` con il nome del bucket Amazon S3, `aws-account-id` e `region` con un numero di account AWS valido e una regione del Catalogo dati, `database_name` con il nome del database e `table_name` con il nome della tabella.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:UpdateTable",
                "glue:GetTable"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<aws-account-id>:table/<database-name>/<table-
name>",
                "arn:aws:glue:<region>:<aws-account-id>:database/<database-name>",
                "arn:aws:glue:<region>:<aws-account-id>:catalog"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/
iceberg-compaction/logs:*"
        }
    ]
}
```

- Utilizza la seguente policy per abilitare la compattazione dei dati registrati con Lake Formation.

  Per ulteriori informazioni su come registrare un bucket Amazon S3 con Lake Formation, consulta Requisiti per i ruoli utilizzati per registrare i percorsi.

```
{
 "Version": "2012-10-17",
 "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
        "glue:UpdateTable",
        "glue:GetTable"
        ],

        "Resource": [
            "arn:aws:glue:<region>:<aws-account-
id>:table/<databaseName>/<tableName>",
            "arn:aws:glue:<region>:<aws-account-id>:database/<database-name>",
            "arn:aws:glue:<region>:<aws-account-id>:catalog"
        ]
    },
```

```
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
        ],
        "Resource": "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/
iceberg-compaction/logs:*"
        }
    ]
 }
```

Se al ruolo di compattazione non sono concesse le autorizzazioni di `IAM_ALLOWED_PRINCIPALS` gruppo sulla tabella, il ruolo richiede le autorizzazioni ALTER, DESCRIBE, INSERT e DELETE di Lake Formation sulla tabella.

- (Facoltativo) Per compattare le tabelle Iceberg con i dati nei bucket Amazon S3 crittografati utilizzando la Crittografia lato server, il ruolo di compattazione richiede le autorizzazioni per decrittografare gli oggetti Amazon S3 e per generare una nuova chiave dati per scrivere oggetti nei bucket crittografati. Aggiungere la policy seguente alla chiave AWS KMS desiderata. Supportiamo solo la crittografia a livello di bucket.

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::<aws-account-id>:role/<compaction-role-name>"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

- (Facoltativo) Per la posizione dei dati registrati con Lake Formation, il ruolo utilizzato per registrare la posizione richiede le autorizzazioni per decrittografare gli oggetti Amazon S3 e generare una nuova chiave dati per scrivere oggetti nei bucket crittografati. Per ulteriori informazioni, consulta la pagina Registrazione di una posizione crittografata Amazon S3.

- (Facoltativo) Se la chiave AWS KMS è memorizzata in un altro account AWS, è necessario includere le seguenti autorizzazioni per il ruolo di compattazione.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": ["arn:aws:kms:<REGION>:<KEY_OWNER_ACCOUNT_ID>:key/<KEY_ID>"]
        }
    ]
}
```

- Il ruolo utilizzato per eseguire la compattazione deve disporre dell'autorizzazione `iam:PassRole` relativa al ruolo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<account-id>:role/<compaction-role-name>"
            ]
        }
    ]
}
```

- Aggiungi la seguente policy di attendibilità al ruolo per il servizio AWS Glue per assumere il ruolo IAM ed eseguire il processo di compattazione.

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

```
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Abilitazione della compattazione

Puoi utilizzare la consoleAWS Glue, la console Lake Formation, AWS CLI o l'API AWS per abilitare la compattazione per le tabelle Apache Iceberg presenti nel Catalogo dati. Per le nuove tabelle, puoi scegliere Apache Iceberg come formato di tabella e abilitare la compattazione quando crei la tabella. La compattazione è disabilitata per impostazione predefinita per le nuove tabelle.

Console

Per abilitare la compattazione

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/ e accedi come amministratore del data lake, creatore della tabella o utente a cui sono state concesse le autorizzazioni `lakeformation:GetDataAccess` e `glue:UpdateTable` sulla tabella.
2. Nel pannello di navigazione, in Catalogo dati, seleziona Tabelle.
3. Nella pagina Tabelle, scegli una tabella in formato tabella aperta per la quale desideri abilitare la compattazione, quindi nel menu Azioni, scegli Abilita compattazione.
4. Puoi anche abilitare la compattazione selezionando la tabella e aprendo la pagina dei Dettagli della tabella. Scegli la scheda Ottimizzazione della tabella nella sezione inferiore della pagina e scegli Abilita compattazione.
5. Successivamente, seleziona un ruolo IAM esistente dal menu a discesa con le autorizzazioni mostrate nella sezione Prerequisiti per l'ottimizzazione delle tabelle .

   Quando scegli l'opzione Crea un nuovo ruolo IAM, il servizio crea un ruolo personalizzato con le autorizzazioni necessarie per eseguire la compattazione.

   Segui la procedura riportata di seguito per aggiornare un ruolo IAM esistente:

a. Per aggiornare la politica di autorizzazione per il ruolo IAM, nella console IAM, vai al ruolo IAM utilizzato per eseguire la compattazione.
b. Nella sezione Autorizzazioni, scegli Aggiungi policy bucket. Nella finestra del browser appena aperta, crea una nuova policy da utilizzare con il tuo ruolo.
c. Nella pagina Crea policy, scegli la scheda JSON. Copia il codice `JSON` mostrato nella sezione the section called “Prerequisiti” nel campo dell'editor delle policy.

AWS CLI

L'esempio seguente mostra come abilitare la compattazione. Sostituisci l'ID dell'account con un ID dell'account AWS valido. Sostituisci il nome del database e della tabella con quello effettivo della tabella Iceberg e del database. Sostituisci `roleArn` con il nome della risorsa (ARN) AWS del ruolo IAM e il nome del ruolo IAM che dispone delle autorizzazioni necessarie per eseguire la compattazione.

```
aws glue create-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::<123456789012>:role/<compaction_role>",
 "enabled":'true'}' \
  --type compaction
```

AWS API

Chiama l'operazione `CreateTableOptimizer` per abilitare la compattazione di una tabella.

Dopo aver abilitato la compattazione, la scheda di Ottimizzazione della tabella mostra i seguenti dettagli di compattazione, dopo circa 15-20 minuti:

- Ora di inizio: l'ora in cui è iniziato il processo di compattazione all'interno di Lake Formation. Il valore è un timestamp in formato UTC.
- Ora di fine: l'ora in cui è finito il processo di compattazione all'interno di Lake Formation. Il valore è un timestamp in formato UTC.
- Stato: lo stato dell'esecuzione di compattazione. I valori sono esito positivo o negativo.
- File compattati: numero totale di file compattati.

- Byte compattati: numero totale di byte compattati.

Disabilitazione della compattazione

È possibile disabilitare la compattazione automatica per una particolare tabella Apache Iceberg utilizzando la console AWS Glue oppure AWS CLI.

Console

1. Scegli Catalogo dati e poi Tabelle. Dall'elenco delle tabelle, scegli la tabella in formato tabella aperta di cui desideri disabilitare la compattazione.
2. Puoi scegliere una tabella Iceberg e scegliere Disabilita compattazione in Azioni.

   Puoi anche disabilitare la compattazione per la tabella scegliendo Disabilita compattazione nella sezione inferiore della pagina dei Dettagli delle tabelle.
3. Scegli Disabilita la compattazione nel messaggio di conferma. È possibile abilitare nuovamente la compattazione in un secondo momento.

   Dopo la conferma, la compattazione viene disabilitata e il relativo stato torna a `Off`.

AWS CLI

Nell'esempio seguente, sostituisci l'ID account con un ID account AWS valido. Sostituisci il nome del database e della tabella con quello effettivo della tabella Iceberg e del database. Sostituisci `roleArn` con il nome della risorsa (ARN) AWS del ruolo IAM e il nome effettivo del ruolo IAM che dispone delle autorizzazioni necessarie per eseguire la compattazione.

```
aws glue update-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/compaction_role", "enabled":'false'}'\
  --type compaction
```

AWS API

Chiama l'operazione `UpdateTableOptimizer` per disabilitare la compattazione per una tabella specifica.

Visualizzazione dei dettagli della compattazione

È possibile visualizzare lo stato di compattazione per Apache Iceberg utilizzando la console AWS Glue, AWS CLI o le operazioni API AWS.

Console

Per visualizzare lo stato di compattazione delle tabelle Iceberg

- È possibile visualizzare lo stato di compattazione delle tabelle Iceberg nella console AWS Glue selezionando Tabelle in Catalogo dati. Il campo Stato di compattazione mostra lo stato dell'esecuzione di compattazione. È possibile visualizzare il formato della tabella e lo stato di compattazione utilizzando le preferenze della tabella.
- Per visualizzare la cronologia delle esecuzioni di compattazione per una tabella specifica, scegli Tabelle in AWS Glue Data Catalog e scegli una tabella per visualizzarne i dettagli. La scheda Ottimizzazione della tabella ne mostra la cronologia di compattazione.

AWS CLI

È possibile visualizzare i dettagli della compattazione utilizzando AWS CLI.

Negli esempi seguenti, sostituite l'ID account con un ID account AWS valido, il nome del database e della tabella con il nome effettivo della tabella Iceberg.

- Per ottenere i dettagli dell'ultima esecuzione di compattazione per una tabella

```
aws get-table-optimizer \
--catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

- Utilizza l'esempio seguente per recuperare la cronologia di un ottimizzatore per una tabella specifica.

```
aws list-table-optimizer-runs \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
```

```
  --type compaction
```

- L'esempio seguente mostra come recuperare l'esecuzione di compattazione e i dettagli di configurazione per più ottimizzatori. Puoi specificare un massimo di 20 ottimizzatori.

```
aws glue batch-get-table-optimizer \
--entries '[{"catalogId":"123456789012", "databaseName":"iceberg_db",
 "tableName":"iceberg_table", "type":"compaction"}]'
```

AWS API

- Usa l'operazione `GetTableOptimizer` per recuperare i dettagli dell'ultima esecuzione di un ottimizzatore.
- Usa l'operazione `ListTableOptimizerRuns` per recuperare la cronologia di un determinato ottimizzatore su una tabella specifica. È possibile specificare 20 ottimizzatori in una singola chiamata API.
- Usa l'operazione `BatchGetTableOptimizer` per recuperare i dettagli di configurazione per più ottimizzatori nel tuo account. Questa operazione non supporta le chiamate multi-account.

Visualizzazione dei parametri Amazon CloudWatch

Dopo aver eseguito correttamente la compattazione, il servizio crea parametri Amazon CloudWatch sulle prestazioni del processo di compattazione. Puoi andare alla CloudWatchconsole e scegliere Metriche, Tutte le metriche. Puoi filtrare i parametri in base allo spazio dei nomi specifico (ad esempio AWS Glue), al nome della tabella o al nome del database.

Per ulteriori informazioni, consulta Visualizzazione di parametri disponibili nella Guida per l'utente di Amazon CloudWatch.

- Numero di byte compattati
- Numero di file compattati
- Numero di DPU assegnate ai processi
- Durata del processo (ore)

Eliminazione di un ottimizzatore

È possibile eliminare un ottimizzatore e i metadati associati per la tabella utilizzando AWS CLI o un'operazione API AWS.

Esegui il comando AWS CLI seguente per eliminare la cronologia di compattazione per una tabella.

```
aws glue delete-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

Usa l'operazione `DeleteTableOptimizer` per eliminare un ottimizzatore per una tabella.

Considerazioni e limitazioni

La compattazione dei dati supporta:

- Tipi di dati: Boolean, Integer, Lungo, Float, Doppio, Stringa, Decimale, Data, Ora, Timestamp, Stringa, UUID, Binary
- Compressione: zstd, gzip, snappy, non compresso
- Crittografia: la compattazione dei dati supporta solo la crittografia Amazon S3 (SSE-S3) e la crittografia KMS lato server (SSE-KMS).
- Compattazione BinPack
- Evoluzione dello schema
- Tabelle con dimensione del file di destinazione (scrittura). target-file-size-bytes proprietà (in configurazione iceberg) nell'intervallo compreso tra 128 MB e 512 MB.
- Regioni
  - Asia Pacifico (Tokyo)
  - Asia Pacifico (Seul)
  - Asia Pacifico (Mumbai)
  - Europa (Irlanda)
  - Europa (Francoforte)
  - Stati Uniti orientali (Virginia settentrionale)

  - Stati Uniti orientali (Ohio)
  - Stati Uniti occidentali (California settentrionale)
- Puoi eseguire la compattazione dall'account in cui risiede il Catalogo dati quando il bucket Amazon S3 che archivia i dati sottostanti si trova in un altro account. Per eseguire questa azione, il ruolo di compattazione richiede l'accesso al bucket Amazon S3.

La compattazione dei dati attualmente non supporta:

- Tipi di dati: fissi
- Compressione: brotli, lz4
- Compattazione dei file mentre le specifiche della partizione si evolvono.
- Ordinamento regolare o con ordine z
- Unisci o elimina file: il processo di compattazione non considera i file di dati a cui sono associati file eliminati.
- Compattazione su tabelle con più account: non è possibile eseguire la compattazione su tabelle con più account.
- Compattazione su tabelle con più regioni: non è possibile eseguire la compattazione su tabelle con più regioni.
- Abilitazione della compattazione sui link alle risorse
- Endpoint VPC per bucket Amazon S3

Utilizzo degli indici delle partizioni in AWS Glue

Nel corso del tempo, centinaia di migliaia di partizioni vengono aggiunte a una tabella. L'GetPartitions API viene utilizzata per recuperare le partizioni nella tabella. L'API restituisce partizioni che corrispondono all'espressione fornita nella richiesta.

Prendiamo come esempio una tabella sales_data che è partizionata dalle chiavi Country, Category, Year, Month e CreationDate. Se desideri ottenere i dati di vendita per tutti gli articoli venduti per la categoria Libri nell'anno 2020 dopo il 15/08/2020, devi effettuare una `GetPartitions` richiesta con l'espressione «Category = 'Books' and CreationDate > '2020-08-15'» al Data Catalog.

Se nella tabella non sono presenti indici delle partizioni, AWS Glue carica tutte le partizioni della tabella e quindi filtra le partizioni caricate utilizzando l'espressione di query fornita dall'utente nella richiesta `GetPartitions`. L'esecuzione della query richiede più tempo man mano che il numero di

partizioni aumenta in una tabella senza indici. Con un indice, la query `GetPartitions` cercherà di recuperare un sottoinsieme delle partizioni invece di caricare tutte le partizioni nella tabella.

Argomenti

Informazioni sugli indici delle partizioni

Quando crei un indice di partizione, specifichi un elenco di chiavi di partizione già esistenti in una determinata tabella. L'indice delle partizioni è un sottoelenco di chiavi di partizione definite nella tabella. Un indice di partizione può essere creato su qualsiasi permutazione delle chiavi di partizione definite nella tabella. Per la tabella sales_data precedente, gli indici possibili sono (country, category, CreationDate), (country, category, year), (country, category), (country), (category, country, year, month) e così via.

Il catalogo dati concatenerà i valori delle partizioni nell'ordine fornito al momento della creazione dell'indice. L'indice viene creato in modo coerente man mano che le partizioni vengono aggiunte alla tabella. Gli indici possono essere creati per i tipi di colonna String (string, char e varchar), Numeric (int, bigint, long, tinyint e smallint) e Date (yyyy-MM-DD).

Tipi di dati supportati

- Data: una data in formato ISO, ad esempio`YYYY-MM-DD`. Ad esempio, data`2020-08-15`. Il formato utilizza i trattini (-) per separare l'anno, il mese e il giorno. L'intervallo consentito per le date per l'indicizzazione va da a. `0000-01-01` `9999-12-31`
- String: una stringa letterale racchiusa tra virgolette singole o doppie.
- Char: dati di caratteri a lunghezza fissa, con una lunghezza specificata compresa tra 1 e 255, come char (10).
- Varchar — Dati di caratteri a lunghezza variabile, con una lunghezza specificata compresa tra 1 e 65535, come varchar (10).

- Numerico: int, bigint, long, tinyint e smallint

Gli indici sui tipi di dati Numeric, String e Date supportano =, >, >=, <, <= e tra gli operatori. La soluzione di indicizzazione attualmente supporta solo l'operatore logico AND. Le sottoespressioni con gli operatori "LIKE", "IN", "OR" e "NOT" vengono ignorate nell'espressione il filtraggio con indice. Il filtraggio per la sottoespressione ignorata viene eseguito sulle partizioni recuperate dopo aver applicato il filtraggio dell'indice.

Per ogni partizione aggiunta a una tabella, viene creato un elemento indice corrispondente. Per una tabella con partizioni "n", l'indice di partizione 1 risulterà come elementi indice di partizione "n". L'indice di partizione "m" sulla stessa tabella risulterà come elementi di indice di partizione "m*n". Ogni elemento dell'indice della partizione verrà addebitato in base alla policy dei prezzi AWS Glue corrente per l'archiviazione del catalogo dati. Per informazioni dettagliate sui prezzi degli oggetti di archiviazione, consulta Prezzi di AWS Glue.

Creazione di una tabella con indici delle partizioni

È possibile creare un indice di partizione durante la creazione di una tabella. La richiesta `CreateTable` utilizza un elenco di oggetti `PartitionIndex` come input. È possibile creare un massimo di 3 indici delle partizioni su una determinata tabella. Ogni indice di partizione richiede un nome e un elenco di `partitionKeys` definite per la tabella. Gli indici creati su una tabella possono essere recuperati utilizzando l'API `GetPartitionIndexes`

Aggiunta di un indice di partizione a una tabella esistente

Per aggiungere un indice di partizione a una tabella esistente, utilizza l'operazione `CreatePartitionIndex`. Puoi creare un solo `PartitionIndex` per ogni operazione `CreatePartitionIndex`. L'aggiunta di un indice non influisce sulla disponibilità di una tabella, poiché durante la creazione degli indici la tabella continua a essere disponibile.

Lo stato dell'indice per una partizione aggiunta è impostato su CREATING (CREAZIONE IN CORSO) e la creazione dei dati di indice viene avviata. Se il processo per la creazione degli indici ha esito positivo, IndexStatus viene aggiornato in ACTIVE (ATTIVO) e, per un processo non riuscito, lo stato dell'indice viene aggiornato a FAILED (NON RIUSCITO). La creazione dell'indice può avere esito negativo per diversi motivi ed è possibile utilizzare l'operaizione `GetPartitionIndexes` per recuperare i dettagli dell'errore. I possibili errori sono:

- ENCRYPTED_PARTITION_ERROR: la creazione di indici su una tabella con partizioni crittografate non è supportata.

- INVALID_PARTITION_TYPE_DATA_ERROR: riscontrato quando il parametro `partitionKey` non è un valore valido per il tipo di dati della `partitionKey` corrispondente. Ad esempio: una `partitionKey` con il tipo di dati 'int' ha un valore 'foo'.
- MISSING_PARTITION_VALUE_ERROR: riscontrato quando il `partitionValue` per un `indexedKey` non è presente. Ciò può accadere quando una tabella non è partizionata in modo coerente.
- UNSUPPORTED_PARTITION_CHARACTER_ERROR: riscontrato quando il valore di una chiave di partizione indicizzata contiene i caratteri \u0000, \u0001 o \u0002
- INTERNAL_ERROR: si è verificato un errore interno durante la creazione degli indici.

Descrizione degli indici delle partizioni su una tabella

Per recuperare gli indici delle partizioni creati su una tabella, utilizza l'operazione `GetPartitionIndexes`. La parte risposta mostra tutti gli indici della tabella insieme allo stato corrente di ciascun indice (`IndexStatus`).

L'`IndexStatus` di un indice può essere uno dei seguenti:

- `CREATING`: l'indice è in fase di creazione e non è ancora disponibile per l'uso.
- `ACTIVE`: l'indice è pronto per l'uso. Le richieste possono utilizzare l'indice per eseguire una query ottimizzata.
- `DELETING`: l'indice è attualmente in fase di eliminazione e non può essere più utilizzato. Un indice nello stato attivo può essere eliminato utilizzando la richiesta `DeletePartitionIndex`, che sposta lo stato da ACTIVE (ATTIVO) a DELETING (ELIMINAZIONE IN CORSO).
- `FAILED`: la creazione dell'indice su una tabella esistente non è riuscita. Ogni tabella memorizza gli ultimi 10 indici non riusciti.

Le possibili transizioni di stato per gli indici creati su una tabella esistente sono le seguenti:

- CREATING → ACTIVE → DELETING (CREAZIONE → ATTIVO → ELIMINAZIONE IN CORSO)
- CREATING → FAILED (IN FASE DI CREAZIONE → NON RIUSCITO)

Limitazioni all'utilizzo degli indici delle partizioni

Dopo aver creato un indice delle partizioni, prendi nota delle seguenti modifiche alla funzionalità di tabella e partizione:

Creazione di nuove partizioni (dopo l'aggiunta dell'indice)

Dopo aver creato un indice delle partizioni su una tabella, tutte le nuove partizioni aggiunte alla tabella verranno convalidate per i controlli del tipo di dati per le chiavi indicizzate. Il valore di partizione delle chiavi indicizzate verrà convalidato per il formato del tipo di dati. Se il controllo del tipo di dati non riesce, l'operazione di creazione della partizione avrà esito negativo. Per la tabella sales_data, se viene creato un indice per le chiavi (category [categoria], year [anno]) in cui la categoria è di tipo `string` e l'anno di tipo `int`, la creazione della nuova partizione con un valore YEAR come "foo" non riuscirà.

Dopo che gli indici sono abilitati, l'aggiunta di partizioni con valori chiave indicizzati aventi i caratteri U +0000, U+00001 e U+0002 inizierà ad avere esito negativo.

Aggiornamenti delle tabelle

Una volta creato un indice delle partizioni in una tabella, non è possibile modificare i nomi delle chiavi di partizione esistenti né modificare il tipo o l'ordine delle chiavi registrate con l'indice.

Utilizzo degli indici per una chiamata ottimizzata GetPartitions

Quando chiami `GetPartitions` su una tabella con un indice, puoi includere un'espressione e, se possibile, il catalogo dati utilizzerà un indice. La prima chiave dell'indice deve essere trasmessa nell'espressione per utilizzare gli indici nel filtro. L'ottimizzazione dell'indice nel filtraggio viene applicata come miglior tentativo. Il catalogo dati tenta di utilizzare il più possibile l'ottimizzazione dell'indice, ma in caso di indice mancante o di operatore non supportato, ritorna all'implementazione esistente di caricamento di tutte le partizioni.

Per la tabella sales_data di cui sopra, aggiungiamo l'indice [Country, Category, Year]. Se "Country" non viene trasmesso nell'espressione, l'indice registrato non sarà in grado di filtrare le partizioni utilizzando gli indici. È possibile aggiungere fino a 3 indici per supportare vari modelli di query.

Prendiamo alcune espressioni di esempio e vediamo come funzionano gli indici:

Espressioni	Come verrà usato l'indice
Country = 'US'	L'indice verrà utilizzato per filtrare le partizioni.
Country = 'US' and Category = 'Shoes'	L'indice verrà utilizzato per filtrare le partizioni.
Category = 'Shoes'	Gli indici non utilizzati come "country" non verranno specificati nell'espressione. Tutte le

Espressioni	Come verrà usato l'indice
	partizioni verranno caricate per restituire una risposta.
Country = 'US' and Category = 'Shoes' and Year > '2018'	L'indice verrà utilizzato per filtrare le partizioni.
Country = 'US' and Category = 'Shoes' and Year > '2018' and month = 2	L'indice verrà utilizzato per recuperare tutte le partizioni con country = "US" e category = "shoes" e year > 2018. Quindi, verrà eseguito il filtraggio sull'espressione del mese.
Country = 'US' AND Category = 'Shoes' OR Year > '2018'	Gli indici non verranno utilizzati poiché nell'espressione è presente l'operatore `OR`.
Country = 'US' AND Category = 'Shoes' AND (Year = 2017 OR Year = '2018')	L'indice verrà utilizzato per recuperare tutte le partizioni con country = "US" e category = "shoes" e quindi verrà eseguito il filtraggio sull'espressione del mese.
Country in ('US', 'UK') AND Category = 'Shoes'	Gli indici non verranno utilizzati per il filtraggio poiché l'operatore `IN` al momento non è supportato.
Country = 'US' AND Category in ('Shoes', 'Books')	L'indice verrà utilizzato per recuperare tutte le partizioni con country = "US", quindi verrà eseguito il filtro sull'espressione Category.
Paese = «Stati Uniti» E categoria in («Scarpe», «Libri») AND (CreationDate > '2023-9-01'	L'indice verrà utilizzato per recuperare tutte le partizioni con country = «US», con CreationDate > '2023-9-01', quindi verrà eseguito il filtraggio sull'espressione Category.

Integrazione con i motori

Redshift Spectrum, Amazon EMR ed AWS Glue ETL Spark DataFrames sono in grado di utilizzare gli indici per recuperare le partizioni dopo che gli indici sono in uno stato ATTIVO in. AWS Glue Athena e

i frame dinamici AWS Glue ETL richiedono di eseguire ulteriori passaggi per utilizzare gli indici per il miglioramento delle query.

Abilita il filtraggio delle partizioni

Per abilitare il filtraggio delle partizioni in Athena, è necessario aggiornare le proprietà della tabella come segue:

1. Nella AWS Glue console, in Data Catalog, scegli Tabelle.
2. Scegliere una tabella .
3. In Azioni, scegli Modifica tabella.
4. In Proprietà della tabella, aggiungi quanto segue:
   - Chiave — `partition_filtering.enabled`
   - Valore — `true`
5. Scegli Applica.

In alternativa, è possibile impostare questo parametro eseguendo una query ALTER TABLE SET PROPERTIES in Athena.

```
ALTER TABLE partition_index.table_with_index
SET TBLPROPERTIES ('partition_filtering.enabled' = 'true')
```

Utilizzo delle statistiche delle colonne

Puoi calcolare statistiche a livello di colonna per AWS Glue Data Catalog tabelle in formati di dati come Parquet, ORC, JSON, ION, CSV e XML senza configurare pipeline di dati aggiuntive. Le statistiche delle colonne consentono di comprendere i profili di dati ottenendo informazioni dettagliate sui valori all'interno di una colonna. Catalogo dati supporta la generazione di statistiche per valori di colonna come valore minimo, valore massimo, valori nulli totali, valori distinti totali, lunghezza media dei valori e occorrenze totali di valori reali.

AWS servizi di analisi come Amazon Redshift Amazon Athena possono utilizzare queste statistiche a colonne per generare piani di esecuzione delle query e scegliere il piano ottimale che migliori le prestazioni delle query.

Puoi configurare l'esecuzione di attività di generazione di statistiche sulle colonne utilizzando la AWS Glue console o AWS CLI. Quando avvii il processo, AWS Glue avvia un job

Spark in background e aggiorna i metadati della AWS Glue tabella nel Data Catalog. Puoi visualizzare le statistiche delle colonne utilizzando la AWS Glue console AWS CLI o chiamando l'GetColumnStatisticsForTableoperazione API.

> **Note**
>
> Se utilizzi le autorizzazioni di Lake Formation per controllare l'accesso alla tabella, il ruolo assunto dall'attività di statistica delle colonne richiede l'accesso completo alla tabella per generare statistiche.

Argomenti

Prerequisiti per la generazione delle statistiche delle colonne

Per generare o aggiornare le statistiche delle colonne, l'attività di generazione delle statistiche assume un ruolo (IAM) AWS Identity and Access Management . In base alle autorizzazioni concesse al ruolo, l'attività di generazione delle statistiche delle colonne può leggere i dati dal datastore di Amazon S3.

> **Note**
>
> Per generare statistiche per le tabelle gestite da Lake Formation, il ruolo IAM utilizzato per generare le statistiche richiede l'accesso completo alla tabella.

Per utilizzare il controllo degli accessi basato su ruoli, è necessario creare un ruolo IAM con le autorizzazioni elencate nella policy riportata di seguito e aggiungere il ruolo all'attività di generazione delle statistiche delle colonne.

Per creare un ruolo IAM per la generazione delle statistiche delle colonne

1. Per creare un ruolo IAM, consulta l'argomento relativo alla creazione di ruoli IAM per AWS Glue.
2. Per aggiornare un ruolo esistente, nella console IAM, vai al ruolo IAM utilizzato dal processo di generazione delle statistiche delle colonne.
3. Nella sezione Autorizzazioni, scegli Collega policy. Nella finestra del browser appena aperta, scegli policy `AWSGlueServiceRole` AWS gestita.
4. È necessario includere anche le autorizzazioni di lettura dei dati dalla posizione dei dati Amazon S3.

   Nella sezione Autorizzazioni, scegli Aggiungi policy bucket. Nella finestra del browser appena aperta, crea una nuova policy da utilizzare con il tuo ruolo.
5. Nella pagina Crea policy seleziona la scheda JSON. Copia il codice seguente `JSON` nel campo dell'editor di policy.

> **Note**
>
> Nelle seguenti politiche, sostituisci l'ID dell'account con un nome valido Account AWS, la regione della tabella e `bucket-name` il nome del bucket Amazon S3. `region`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>/*",
                "arn:aws:s3:::<bucket-name>"
            ]
```

```
        }
     ]
}
```

6. (Facoltativo) Se utilizzi le autorizzazioni di Lake Formation per fornire l'accesso ai tuoi dati, il ruolo IAM richiede le autorizzazioni `lakeformation:GetDataAccess`.

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "LakeFormationDataAccess",
            "Effect": "Allow",
            "Action": "lakeformation:GetDataAccess",
            "Resource": [
                "*"
            ]
        }
    ]
}
```

   Se la posizione dei dati di Amazon S3 è registrata con Lake Formation e il ruolo IAM assunto dall'attività di generazione delle statistiche delle colonne non dispone delle autorizzazioni di gruppo `IAM_ALLOWED_PRINCIPALS` concesse sulla tabella, il ruolo richiede le autorizzazioni `ALTER` e `DESCRIBE` di Lake Formation sulla tabella. Il ruolo utilizzato per la registrazione del bucket Amazon S3 richiede le autorizzazioni `INSERT` e `DELETE` di Lake Formation sulla tabella.

   Se la posizione dei dati di Amazon S3 non è registrata con Lake Formation e il ruolo IAM non dispone delle autorizzazioni di gruppo `IAM_ALLOWED_PRINCIPALS` concesse sulla tabella, il ruolo richiede le autorizzazioni `ALTER`, `DESCRIBE`, `INSERT` e `DELETE` di Lake Formation sulla tabella.

7. (Facoltativo) L'attività di generazione delle statistiche delle colonne che scrive Amazon CloudWatch Logs crittografati necessita delle autorizzazioni seguenti nella policy della chiave.

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

```
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:AssociateKmsKey"
        ],
        "Resource": [
            "arn:aws:logs:<region>:111122223333:log-group:/aws-glue:*"
        ]
    },
    {
        "Sid": "KmsPermissions",
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt",
            "kms:Encrypt"
        ],
        "Resource": [
        "arn:aws:kms:<region>:111122223333:key/"arn of key used for ETL cloudwatch
 encryption"
        ],
        "Condition": {
            "StringEquals": {
                "kms:ViaService": ["glue.<region>.amazonaws.com"]
            }
        }
    }
    ]
}
```

8. Il ruolo che usi per eseguire le statistiche sulle colonne deve avere l'`iam:PassRole`autorizzazione per il ruolo.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": [
            "arn:aws:iam::111122223333:role/<columnstats-role-name>"
```

```
        ]
    }]
}
```

9. Quando crei un ruolo IAM per la generazione delle statistiche delle colonne, tale ruolo deve disporre anche della policy di attendibilità seguente che consente al servizio di assumere il ruolo.

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

Generazione delle statistiche delle colonne

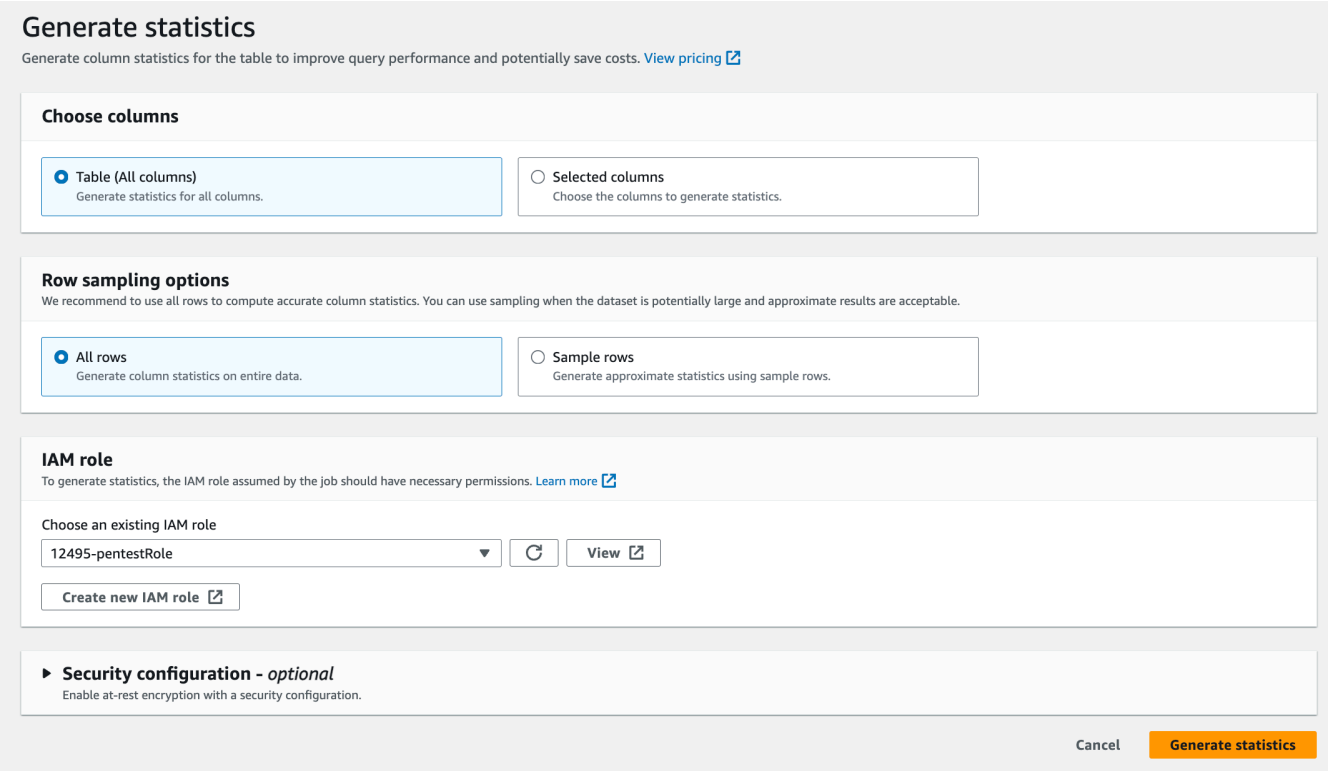
Segui questi passaggi per gestire la generazione delle statistiche nel Catalogo dati utilizzando la console AWS Glue o la AWS CLI.

Console

Per generare statistiche delle colonne utilizzando la console

1. Accedi alla console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/
2. Seleziona le tabelle del Catalogo dati.
3. Scegliere una tabella dall'elenco.
4. Scegli Genera statistiche nel menu Azioni.

   Puoi anche scegliere il pulsante Genera statistiche nella scheda Statistiche di colonna nella sezione inferiore della pagina Tabelle.
5. Nella pagina Genera statistiche, specifica le seguenti opzioni:



- Tabella (tutte le colonne): scegli questa opzione per generare statistiche per tutte le colonne della tabella.
- Colonne selezionate: scegli questa opzione per generare statistiche per colonne specifiche. È possibile selezionare le colonne dall'elenco a discesa.
- Tutte le righe: scegli tutte le righe dalla tabella per generare statistiche accurate.
- Righe di esempio: scegli solo una percentuale specifica di righe dalla tabella per generare statistiche. Il valore predefinito è Tutte le righe. Utilizzate le frecce su e giù per aumentare o diminuire il valore percentuale.

> **Note**
>
> Includi tutte le righe nella tabella per calcolare statistiche accurate. Utilizza righe di esempio per generare statistiche delle colonne solo quando i valori approssimativi sono accettabili.

6. (Facoltativo) Scegli quindi una configurazione di sicurezza per abilitare la crittografia dei dati inattivi per i log.
7. Scegli Genera statistiche per eseguire il processo.

AWS CLI

Nell'esempio seguente, sostituisci i valori per `DatabaseName`, `TableName` e `ColumnNameList` con i nomi effettivi di database, tabelle e colonne. Sostituisci l'ID dell'account con un valido Account AWS e il nome del ruolo con il nome del ruolo IAM che stai utilizzando per generare statistiche.

```
aws glue start-column-statistics-task-run --input-cli-json file://input.json
{
    "DatabaseName": "<test-db>",
    "TableName": "<test-table>",
    "ColumnNameList": [
        "<column1>",
        "<column2>",
    ],
    "Role": "arn:aws:iam::<123456789012>:role/<Stats-Role>",
    "SampleSize": 10.0
}
```

Puoi generare statistiche sulle colonne anche chiamando l'operazione StartColumnStatisticsTaskRun.

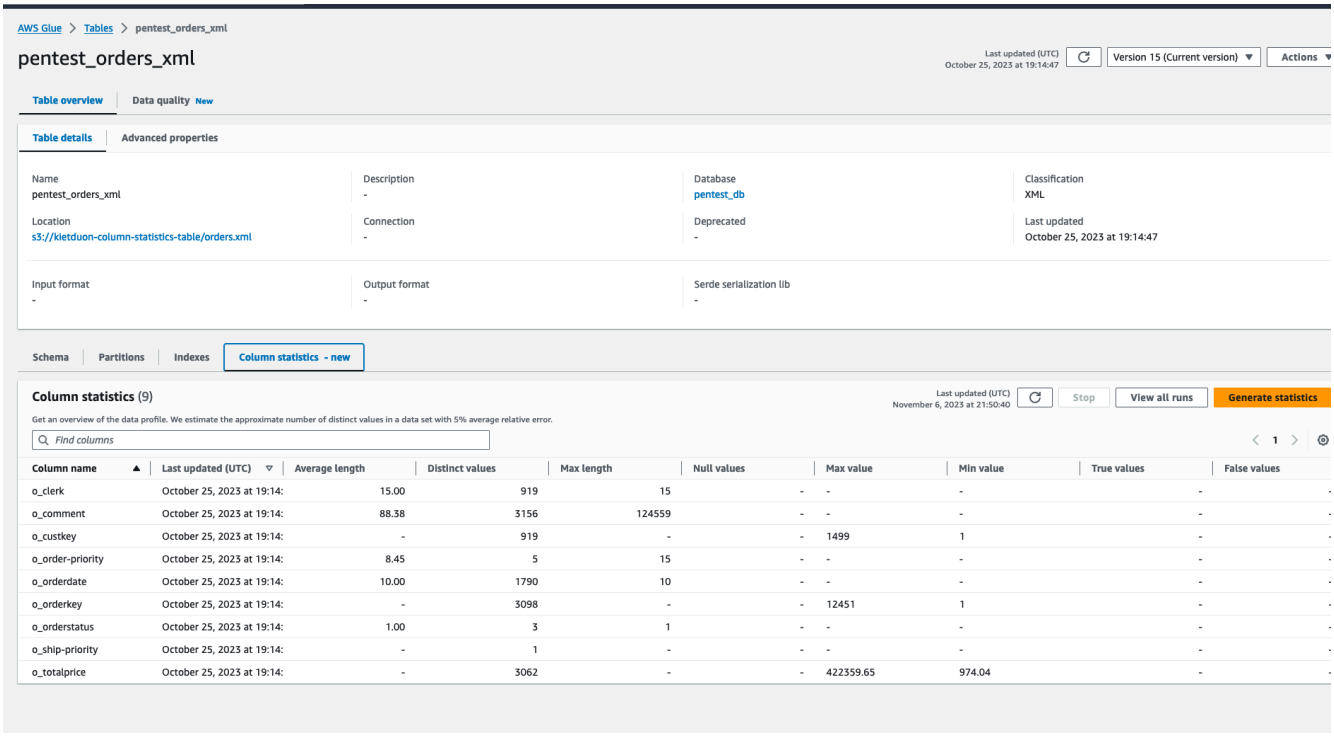
Visualizzazione delle statistiche delle colonne

Dopo aver generato correttamente le statistiche, il Catalogo dati memorizza queste informazioni per gli ottimizzatori basati sui costi in Amazon Athena e in Amazon Redshift per effettuare scelte ottimali durante l'esecuzione delle query. Le statistiche variano in base al tipo di colonna.

AWS Management Console

Per visualizzare le statistiche delle colonne per una tabella

- Dopo l'esecuzione dell'attività di statistica delle colonne, la scheda Statistiche delle colonne della pagina dei Dettagli della tabella mostra le statistiche relative alla tabella.



Sono disponibili le seguenti statistiche:

- Nome colonna: nome della colonna utilizzato per generare statistiche
- Ultimo aggiornamento: data e ora in cui sono state generate le statistiche
- Lunghezza media: lunghezza media dei valori nella colonna
- Valori distinti: il numero totale di valori distinti nella colonna. Eseguiamo una stima del numero di valori distinti in una colonna con un errore relativo del 5%.
- Valore massimo: il valore più alto nella colonna.
- Valore minimo: il valore più basso nella colonna.
- Lunghezza massima: la lunghezza del valore più alto nella colonna.
- Valori null: il numero di valori null nella colonna.
- Valori true: il numero di valori true nella colonna.
- Valori false: il numero di valori false nella colonna.

AWS CLI

L'esempio seguente mostra come recuperare le statistiche delle colonne utilizzando AWS CLI.

```
aws glue get-column-statistics-for-table \
```

```
    --database-name <test_db> \
    --table-name <test_tble> \
    --column-names <col1>
```

Puoi anche visualizzare le statistiche delle colonne utilizzando l'operazione API GetColumnStatisticsForTable.

Aggiornamento delle statistiche delle colonne

Mantenere aggiornate le statistiche migliora le prestazioni delle query perché consente al pianificatore di query di scegliere i piani ottimali. È necessario eseguire esplicitamente l'attività Genera statistiche dalla console AWS Glue per aggiornare le statistiche delle colonne. Il Catalogo dati non aggiorna automaticamente le statistiche.

Se non utilizzi la funzionalità di generazione delle statistiche di AWS Glue nella console, puoi aggiornare manualmente le statistiche delle colonne utilizzando l'operazione API UpdateColumnStatisticsForTable oppure AWS CLI. L'esempio seguente mostra come aggiornare le statistiche delle colonne utilizzando AWS CLI.

```
aws glue update-column-statistics-for-table --cli-input-json:

{
    "CatalogId": "111122223333",
    "DatabaseName": "test_db",
    "TableName": "test_table",
    "ColumnStatisticsList": [
        {
            "ColumnName": "col1",
            "ColumnType": "Boolean",
            "AnalyzedTime": "1970-01-01T00:00:00",
            "StatisticsData": {
                "Type": "BOOLEAN",
                "BooleanColumnStatisticsData": {
                    "NumberOfTrues": 5,
                    "NumberOfFalses": 5,
                    "NumberOfNulls": 0
                }
            }
        }
    ]
}
```

Eliminazione delle statistiche delle colonne

È possibile eliminare le statistiche delle colonne utilizzando l'operazione API DeleteColumnStatisticsForTable oppure AWS CLI. L'esempio seguente mostra come eliminare le statistiche delle colonne utilizzando AWS Command Line Interface (AWS CLI).

```
aws glue delete-column-statistics-for-table \
    --database-name test_db \
    --table-name test_table \
    --column-name col1
```

Visualizzazione dell'attività relativa alle statistiche delle colonne

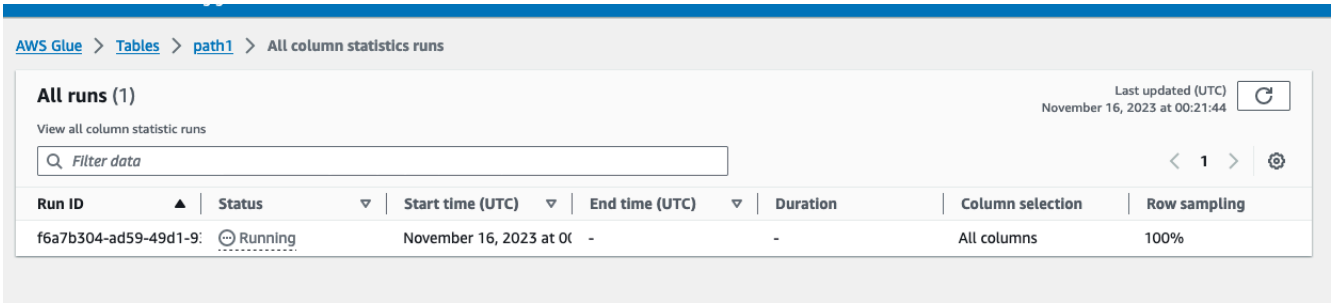
Dopo aver eseguito un'attività relativa alle statistiche delle colonne, è possibile esplorarne i dettagli per una tabella utilizzando la console AWS Glue, AWS CLI o utilizzando l'operazione GetColumnStatisticsTaskRuns.

Console

Per visualizzare i dettagli dell'attività relativa alle statistiche sulle colonne

1. Sulla console AWS Glue, scegli Tabelle nel Catalogo dati.
2. Seleziona una tabella con le statistiche delle colonne.
3. Nella pagina dei Dettagli della tabella, scegli Statistiche delle colonne.
4. Scegli Visualizza esecuzioni.

   Puoi visualizzare le informazioni su tutte le esecuzioni associate alla tabella specificata.



AWS CLI

Nell'esempio seguente, sostituisci i valori per `DatabaseName` e `TableName` con i nomi effettivi di database e tabelle.

```
aws glue get-column-statistics-task-runs --input-cli-json file://input.json
{
    "DatabaseName": "<test-db>",
    "TableName": "<test-table>"
}
```

Interruzione dell'esecuzione relativa alle statistiche delle colonne

È possibile interrompere l'esecuzione di un'attività di statistica delle colonne per una tabella utilizzando la console AWS Glue, AWS CLI o l'operazione StopColumnStatisticsTaskRun.

Console

Per interrompere un'attività di statistica delle colonne, esegui:

1. Sulla console AWS Glue, scegli Tabelle nel Catalogo dati.
2. Seleziona la tabella con la colonna "Attività statistiche". L'operazione è in corso.
3. Nella pagina dei Dettagli della tabella, scegli Statistiche delle colonne.
4. Scegli Stop (Arresta).

   Se interrompi l'attività prima del completamento dell'esecuzione, le statistiche delle colonne non verranno generate per la tabella.

AWS CLI

Nell'esempio seguente, sostituisci i valori per `DatabaseName` e `TableName` con i nomi effettivi di database e tabelle.

```
aws glue stop-column-statistics-task-run --input-cli-json file://input.json
{
    "DatabaseName": "<test-db>",
    "TableName": "<test-table>"
}
```

Considerazioni e limitazioni

Le seguenti considerazioni e limitazioni si applicano alla generazione di statistiche delle colonne.

Considerazioni

- L'utilizzo del campionamento per generare statistiche riduce il tempo di esecuzione ma può generare statistiche imprecise.
- Ogni esecuzione delle statistiche delle colonne richiede l'elaborazione dell'intero set di dati.
- Il Catalogo dati non memorizza versioni diverse delle statistiche.
- È possibile eseguire solo un'attività alla volta per la generazione di statistiche per tabella.
- Se una tabella è crittografata utilizzando la chiave cliente AWS KMS registrata con il Catalogo dati, AWS Glue utilizza la stessa chiave per crittografare le statistiche.

L'attività di Statistiche delle colonne supporta la generazione di statistiche:

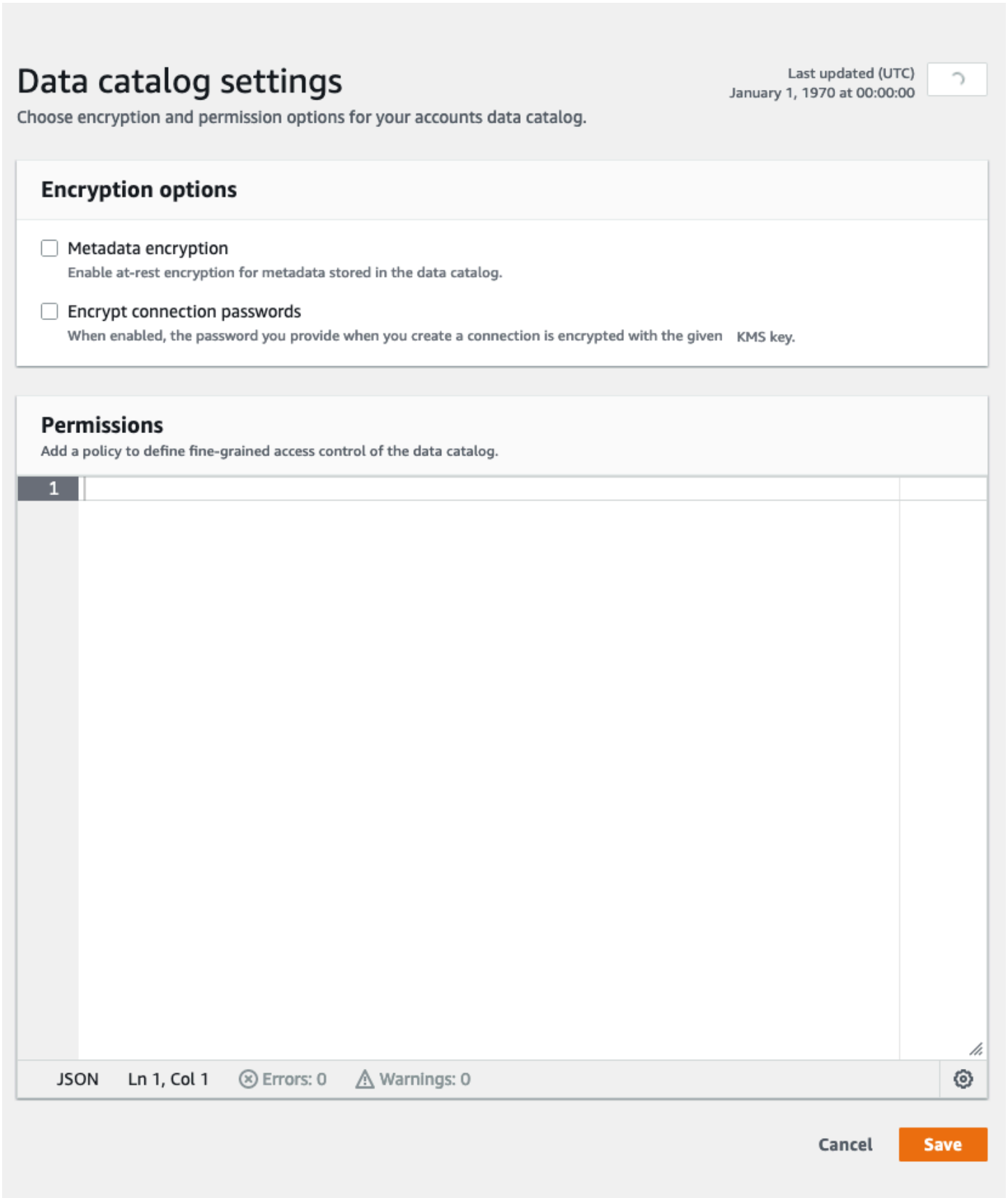
- Quando il ruolo IAM dispone delle autorizzazioni complete per la tabella (IAM o Lake Formation).
- Quando il ruolo IAM dispone di autorizzazioni sulla tabella utilizzando la modalità di accesso ibrida di Lake Formation.

L'attività di Statistiche delle colonne non supporta la generazione di statistiche per:

- Tabelle con controllo degli accessi basato su celle di Lake Formation.
- Data lake transazionali: Linux Foundation Delta Lake, Apache Iceberg, Apache Hudi.
- Tabelle in database federati - Hive metastore, unità di condivisione dati Amazon Redshift
- Colonne, matrici e tipi di dati di struttura nidificati.
- Tabella condivisa con te da un altro account.

Uso delle impostazioni dei cataloghi dati nella console AWS Glue

La pagina delle impostazioni del catalogo dati contiene opzioni per impostare le relative proprietà nell'account.


Data catalog settings
Choose encryption and permission options for your accounts data catalog.
Last updated (UTC)
January 1, 1970 at 00:00:00
Encryption options
Metadata encryption
Enable at-rest encryption for metadata stored in the data catalog.
Encrypt connection passwords
When enabled, the password you provide when you create a connection is encrypted with the given KMS key.
Permissions
Add a policy to define fine-grained access control of the data catalog.
1
JSON
Ln 1, Col 1
Errors: 0
Warnings: 0
Cancel
Save

Per modificare il controllo granulare degli accessi del catalogo dati

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Scegli un'opzione di crittografia.

   - Crittografia dei metadati – Seleziona questa casella di controllo per crittografare i metadati nel catalogo dati. I metadati vengono crittografati quando sono inattivi usando la chiave AWS Key Management Service (AWS KMS) da te specificata. Per ulteriori informazioni, consulta Crittografia del catalogo dati.
   - Crittografa le password di connessione – Seleziona questa casella di controllo per crittografare le password nell'oggetto di connessione AWS Glue quando la connessione viene creata o aggiornata. Le password vengono crittografate utilizzando la chiave AWS KMS specificata. Quando le password vengono restituite, sono crittografate. Questa opzione è un'impostazione globale per tutte le connessioni AWS Glue nel catalogo dati. Se si deseleziona questa casella di controllo, le password precedentemente crittografate rimangono crittografate utilizzando la chiave usata quando sono state create o aggiornate. Per ulteriori informazioni sulle connessioni AWS Glue, consulta Connessione ai dati.

     Quando abiliti questa opzione, seleziona una chiave AWS KMS oppure scegli Enter a key ARN (Immetti un ARN chiave) e specifica l'Amazon Resource Name (ARN) per la chiave. Immetti l'ARN usando questo formato: `arn:aws:kms:`*`region`*`:`*`account-id`*`:key/`*`key-id`* . Puoi specificare l'ARN anche sotto forma di alias di chiavi, ad esempio `arn:aws:kms:`*`region`*`:`*`account-id`*`:alias/`*`alias-name`* .

     > **Important**
     >
     > Se si seleziona questa opzione, qualsiasi utente o ruolo che crea o aggiorna una connessione deve avere l'autorizzazione `kms:Encrypt` sulla chiave KMS specificata.

     Per ulteriori informazioni, consulta Crittografia delle password di connessione.
3. Scegli Settings (Impostazioni), quindi nell'editor Permissions (Autorizzazioni) aggiungi l'istruzione di policy per modificare il controllo granulare degli accessi del catalogo dati per il tuo account. Al catalogo dati è possibile collegare una sola policy per volta. È possibile incollare una policy di risorsa JSON in questo controllo. Per ulteriori informazioni, consulta Policy basate su risorse all'interno di AWS Glue.

4. Scegliere Save (Salva) per aggiornare il catalogo dati con le modifiche apportate.

È inoltre possibile utilizzare le operazioni API AWS Glue per archiviare, recuperare ed eliminare le policy delle risorse. Per ulteriori informazioni, consulta API di sicurezza in AWS Glue.

Creazione di tabelle, aggiornamento dello schema e aggiunta di nuove partizioni nel catalogo dati da processi ETL AWS Glue

Il processo di estrazione, trasformazione e caricamento (ETL) potrebbe creare nuove partizioni di tabella nell'archivio dati di destinazione. Lo schema del set di dati può evolversi e divergere dallo schema del catalogo dati AWS Glue nel corso del tempo. AWS Glue I processi ETL ora forniscono diverse funzionalità che puoi utilizzare all'interno dello script ETL per aggiornare lo schema e le partizioni nel catalogo dati. Queste caratteristiche ti consentono di vedere i risultati del processo ETL nel catalogo dati, senza dover eseguire nuovamente il crawler.

Nuove partizioni

Se si desidera visualizzare le nuove partizioni nel AWS Glue Data Catalog, è possibile effettuare una delle seguenti operazioni:

- Al termine del processo, esegui nuovamente il crawler e visualizza le nuove partizioni sulla console al termine del crawler.
- Al termine del processo, visualizza immediatamente le nuove partizioni sulla console, senza dover eseguire nuovamente il crawler. Puoi abilitare questa caratteristica aggiungendo alcune righe di codice allo script ETL, come mostrato negli esempi seguenti. Il codice utilizza l'argomento `enableUpdateCatalog` per indicare che il catalogo dati deve essere aggiornato durante l'esecuzione del processo quando vengono create nuove partizioni.

Metodo 1

Passare `enableUpdateCatalog` e `partitionKeys` in un argomento opzioni.

Python

```
additionalOptions = {"enableUpdateCatalog": True}
additionalOptions["partitionKeys"] = ["region", "year", "month", "day"]
```

```
sink = glueContext.write_dynamic_frame_from_catalog(frame=last_transform,
 database=<target_db_name>,

 table_name=<target_table_name>, transformation_ctx="write_sink",

 additional_options=additionalOptions)
```

Scala

```
val options = JsonOptions(Map(
    "path" -> <S3_output_path>,
    "partitionKeys" -> Seq("region", "year", "month", "day"),
    "enableUpdateCatalog" -> true))
val sink = glueContext.getCatalogSink(
    database = <target_db_name>,
    tableName = <target_table_name>,
    additionalOptions = options)sink.writeDynamicFrame(df)
```

Metodo 2

Passare `enableUpdateCatalog` e `partitionKeys` in `getSink( )` e chiamare `setCatalogInfo( )` sull'oggetto `DataSink`.

Python

```
sink = glueContext.getSink(
    connection_type="s3",
    path="<S3_output_path>",
    enableUpdateCatalog=True,
    partitionKeys=["region", "year", "month", "day"])
sink.setFormat("json")
sink.setCatalogInfo(catalogDatabase=<target_db_name>,
 catalogTableName=<target_table_name>)
sink.writeFrame(last_transform)
```

Scala

```
val options = JsonOptions(
   Map("path" -> <S3_output_path>,
       "partitionKeys" -> Seq("region", "year", "month", "day"),
       "enableUpdateCatalog" -> true))
val sink = glueContext.getSink("s3", options).withFormat("json")
sink.setCatalogInfo(<target_db_name>, <target_table_name>)
```

```
sink.writeDynamicFrame(df)
```

Ora, puoi creare nuove tabelle di catalogo, aggiornare le tabelle esistenti con schema modificato e aggiungere nuove partizioni di tabella nel catalogo dati utilizzando direttamente un processo ETL di AWS Glue, senza la necessità di eseguire nuovamente i crawler.

Aggiornamento dello schema della tabella

Se desideri sovrascrivere lo schema della tabella del catalogo dati, puoi eseguire una delle seguenti operazioni:

- Al termine del processo, esegui nuovamente il crawler e assicurati che il crawler sia configurato per aggiornare anche la definizione della tabella. Visualizza le nuove partizioni sulla console insieme agli eventuali aggiornamenti dello schema, al termine del crawler. Per maggiori informazioni, consulta Configurazione di un crawler utilizzando l'API.
- Al termine del processo, visualizza immediatamente lo schema modificato sulla console, senza dover eseguire nuovamente il crawler. Puoi abilitare questa caratteristica aggiungendo alcune righe di codice allo script ETL, come mostrato negli esempi seguenti. Il codice utilizza `enableUpdateCatalog` impostato su true, e anche `updateBehavior` impostato su `UPDATE_IN_DATABASE`, il che indica di sovrascrivere lo schema e aggiungere nuove partizioni nel catalogo dati durante l'esecuzione del processo.

Python

```
additionalOptions = {
    "enableUpdateCatalog": True,
    "updateBehavior": "UPDATE_IN_DATABASE"}
additionalOptions["partitionKeys"] = ["partition_key0", "partition_key1"]

sink = glueContext.write_dynamic_frame_from_catalog(frame=last_transform,
 database=<dst_db_name>,
    table_name=<dst_tbl_name>, transformation_ctx="write_sink",
    additional_options=additionalOptions)
job.commit()
```

Scala

```
val options = JsonOptions(Map(
```

```
    "path" -> outputPath,
    "partitionKeys" -> Seq("partition_0", "partition_1"),
    "enableUpdateCatalog" -> true))
val sink = glueContext.getCatalogSink(database = nameSpace, tableName = tableName,
 additionalOptions = options)
sink.writeDynamicFrame(df)
```

Puoi inoltre impostare il valore `updateBehavior` su `LOG` se desideri impedire che lo schema di tabella venga sovrascritto, ma se desidera comunque aggiungere le nuove partizioni. Il valore predefinito di `updateBehavior` è `UPDATE_IN_DATABASE`, quindi se non lo definisci esplicitamente, lo schema della tabella verrà sovrascritto.

Se `enableUpdateCatalog` non è impostato su true, indipendentemente da qualsiasi opzione selezionata per `updateBehavior`, il processo ETL non aggiornerà la tabella nel catalogo dati.

Creazione di nuove tabelle

Puoi inoltre utilizzare le stesse opzioni per creare una nuova tabella nel catalogo dati. Puoi specificare il database e il nome della nuova tabella utilizzando `setCatalogInfo`.

Python

```
sink = glueContext.getSink(connection_type="s3", path="s3://path/to/data",
    enableUpdateCatalog=True, updateBehavior="UPDATE_IN_DATABASE",
    partitionKeys=["partition_key0", "partition_key1"])
sink.setFormat("<format>")
sink.setCatalogInfo(catalogDatabase=<dst_db_name>, catalogTableName=<dst_tbl_name>)
sink.writeFrame(last_transform)
```

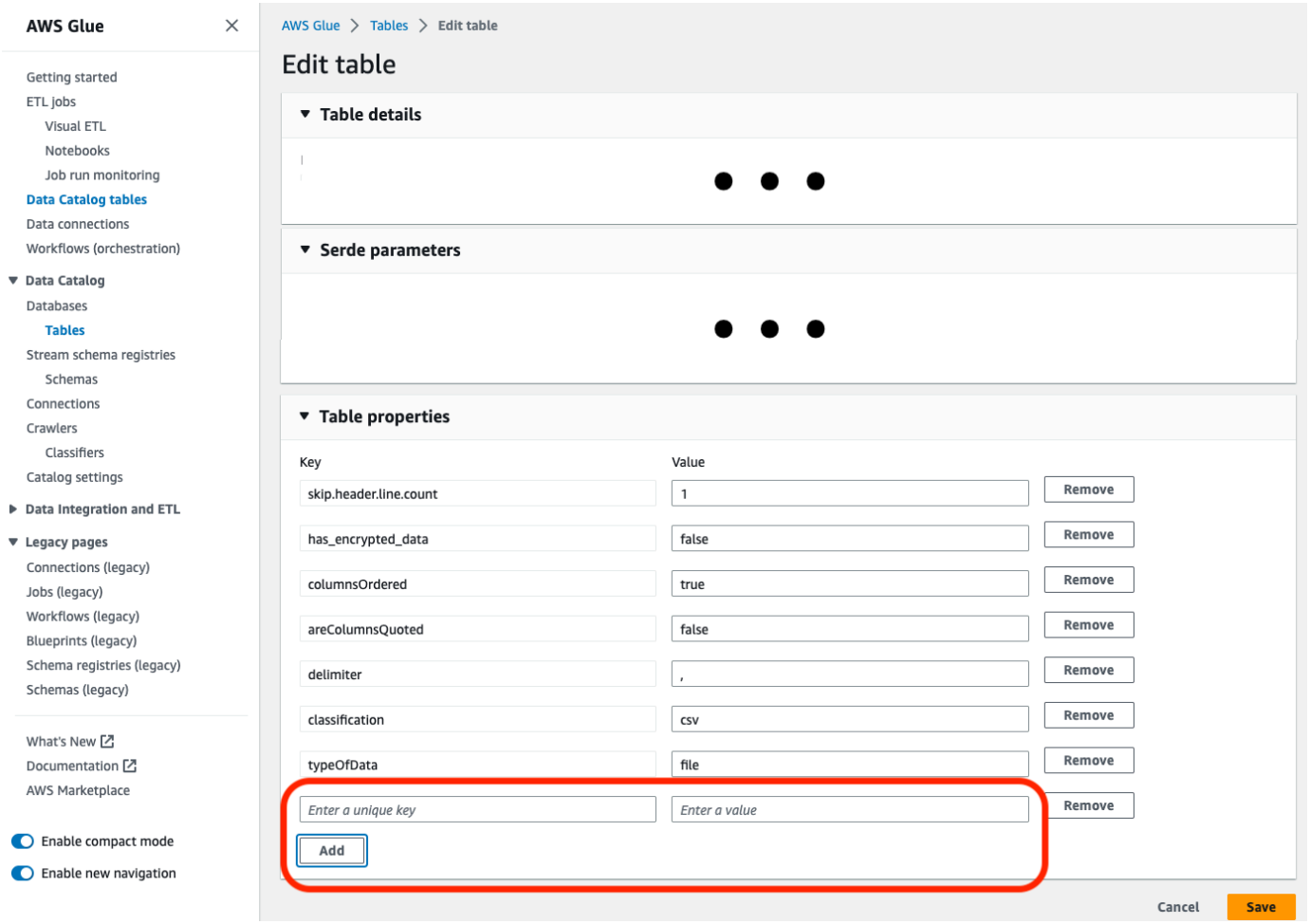
Scala

```
val options = JsonOptions(Map(
    "path" -> outputPath,
    "partitionKeys" -> Seq("<partition_1>", "<partition_2>"),
    "enableUpdateCatalog" -> true,
    "updateBehavior" -> "UPDATE_IN_DATABASE"))
val sink = glueContext.getSink(connectionType = "s3", connectionOptions =
 options).withFormat("<format>")
sink.setCatalogInfo(catalogDatabase = “<dst_db_name>”, catalogTableName =
 “<dst_tbl_name>”)
sink.writeDynamicFrame(df)
```

Restrizioni

Prestare attenzione alle seguenti restrizioni:

- Sono supportate solo le destinazioni di Amazon Simple Storage Service (Amazon S3).
- La funzionalità `enableUpdateCatalog` non è supportata per le tabelle governate.
- Sono supportati solo i seguenti formati: `json`, `csv`, `avro`, e `parquet`.
- Per creare o aggiornare le tabelle con la classificazione `parquet`, è necessario utilizzare lo scrittore di parquet AWS Glue ottimizzato per DynamicFrames. È possibile farlo in uno dei modi seguenti:
  - Se stai aggiornando una tabella esistente nel catalogo con la classificazione `parquet`, la proprietà della tabella `"useGlueParquetWriter"` deve essere impostata su `true` prima di aggiornarla. È possibile impostare questa proprietà tramite le API/l'SDK AWS Glue, la console o un'istruzione Athena DDL.



Una volta impostata la proprietà della tabella del catalogo, puoi utilizzare il seguente frammento di codice per aggiornare la tabella del catalogo con i nuovi dati:

```
glueContext.write_dynamic_frame.from_catalog(
    frame=frameToWrite,
    database="dbName",
    table_name="tableName",
    additional_options={
        "enableUpdateCatalog": True,
        "updateBehavior": "UPDATE_IN_DATABASE"
    }
)
```

- Se la tabella non esiste ancora nel catalogo, puoi utilizzare il metodo `getSink()` nello script con `connection_type="s3"` per aggiungere la tabella e le sue partizioni al catalogo, oltre a scrivere i dati su Amazon S3. Fornisci i valori appropriati di `partitionKeys` e `compression` per il tuo flusso di lavoro.

```
s3sink = glueContext.getSink(
    path="s3://bucket/folder/",
    connection_type="s3",
    updateBehavior="UPDATE_IN_DATABASE",
    partitionKeys=[],
    compression="snappy",
    enableUpdateCatalog=True
)

s3sink.setCatalogInfo(
    catalogDatabase="dbName", catalogTableName="tableName"
)

s3sink.setFormat("parquet", useGlueParquetWriter=true)
s3sink.writeFrame(frameToWrite)
```

- Il valore del formato `glueparquet` è un metodo obsoleto per l'abilitazione del writer Parquet AWS Glue.

- Quando `updateBehavior` è impostato su `LOG`, nuove partizioni verranno aggiunte solo se lo schema `DynamicFrame` è equivalente o contiene un sottoinsieme delle colonne definite nello schema della tabella del catalogo dati.

- Gli aggiornamenti dello schema non sono supportati per le tabelle non partizionate (che non utilizzano l'opzione “partitionKeys”).
- Le PartitionKeys devono essere equivalenti, e nello stesso ordine, tra il parametro passato nello script ETL e le PartitionKeys nello schema della tabella del catalogo dati.
- Al momento questa funzionalità non supporta ancora l'aggiornamento/creazione di tabelle in cui gli schemi di aggiornamento sono nidificati (ad esempio, array all'interno di strutture).

Per ulteriori informazioni, consulta the section called “AWS Glue per Spark”.

Lavorare con le connessioni MongoDB nei processi ETL

Puoi creare una connessione per MongoDB e usare quella connessione nel tuo processo AWS Glue. Per ulteriori informazioni, consulta the section called “Connessioni MongoDB” nella guida di programmazione di AWS Glue. La connessione`url`, `username` e `password` sono archiviati nella connessione MongoDB. Altre opzioni possono essere specificate nello script del processo ETL utilizzando il parametro `additionalOptions` di `glueContext.getCatalogSource`. Le altre opzioni possono includere:

- `database`: (Obbligatorio) Il database MongoDB da cui leggere.
- `collection`: (Obbligatorio) La raccolta MongoDB da cui leggere.

Posizionando le informazioni di `database` e `collection` all'interno dello script del processo ETL, puoi utilizzare la stessa connessione in più processi.

1. Crea una connessione AWS Glue Data Catalog per l'origine dati MongoDB. Consulta "connectionType": "mongodb" per una descrizione dei parametri di connessione. Puoi creare la connessione utilizzando la console, l'API o la CLI.
2. Crea un database in AWS Glue Data Catalog per memorizzare le definizioni di tabella per i dati MongoDB. Per ulteriori informazioni, consulta Database AWS Glue.
3. Crea un crawler che esegue il crawling dati in MongoDB utilizzando le informazioni nella connessione per connettersi a MongoDB. Il crawler crea le tabelle in AWS Glue Data Catalog, che descrivono le tabelle nel database MongoDB che usi nel tuo processo. Per ulteriori informazioni, consulta Definizione di crawler in AWS Glue.
4. Crea un processo con uno script personalizzato. Puoi creare il processo utilizzando la console, l'API o la CLI. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.

5. Scegli le destinazioni dati per il tuo processo. Le tabelle che rappresentano la destinazione dei dati possono essere definite nel catalogo dati oppure il processo può creare le tabelle di destinazione quando viene eseguito. Puoi scegliere una posizione di destinazione al momento della creazione del processo. Se la destinazione richiede una connessione, anche la connessione ha un riferimento nel tuo processo. Se il processo richiede più destinazioni dati, in seguito potrai aggiungerle modificando lo script.
6. Personalizza l'ambiente di elaborazione del processo fornendo gli argomenti per il tuo processo e lo script generato.

   Di seguito è illustrato un esempio di creazione di un `DynamicFrame` dal database MongoDB in base alla struttura della tabella definita nel catalogo dati. Il codice utilizza `additionalOptions` per fornire le informazioni aggiuntive sull'origine dati:

   Scala

   ```
   val resultFrame: DynamicFrame = glueContext.getCatalogSource(
           database = catalogDB,
           tableName = catalogTable,
           additionalOptions = JsonOptions(Map("database" -> DATABASE_NAME,
                   "collection" -> COLLECTION_NAME))
         ).getDynamicFrame()
   ```

   Python

   ```
   glue_context.create_dynamic_frame_from_catalog(
           database = catalogDB,
           table_name = catalogTable,
           additional_options = {"database":"database_name",
               "collection":"collection_name"})
   ```
7. Esegui il processo, on demand o tramite un trigger.

Definizione di crawler in AWS Glue

Puoi usare un crawler per popolare il AWS Glue Data Catalog con tabelle. Questo è il metodo principale usato dalla maggior parte degli utenti di AWS Glue. Un crawler è in grado di eseguire il crawling di archivi dati con una sola esecuzione. Al termine dell'operazione, il crawler crea o aggiorna una o più tabelle nel catalogo dati. I processi di estrazione, trasformazione e caricamento (ETL) definiti in AWS Glue usano queste tabelle del catalogo dati come origini e destinazioni. Il

processo ETL legge e scrive negli archivi dati specificati nelle tabelle del catalogo dati di origine e di destinazione.

Per ulteriori informazioni sull'uso della console AWS Glue per aggiungere un crawler, consulta Uso di crawler nella console AWS Glue.

Argomenti

Quali datastore posso sottoporre a crawling?

Un crawler può eseguire il crawling dei seguentgli archivi dati basati su file e su tabelle.

Tipo di accesso utilizzato dal crawler	Archivi dati
Client nativo	• Amazon Simple Storage Service (Amazon S3) • Amazon DynamoDB • Delta Lake 2.0.x • Apache Iceberg 1.5 • Apache Hudi 0.14
JDBC	Amazon Redshift Snowflake

Tipo di accesso utilizzato dal crawler	Archivi dati
	All'interno di Amazon Relational Database Service (Amazon RDS) o esterno ad Amazon RDS: • Amazon Aurora • MariaDB • Microsoft SQL Server • MySQL • Oracle • PostgreSQL
Client MongoDB	• MongoDB • MongoDB Atlas • Amazon DocumentDB (compatibile con MongoDB)

> **Note**
>
> Attualmente AWS Glue non supporta i crawler per i flussi di dati.

Per gli archivi dati JDBC, MongoDB e Amazon DocumentDB (compatibile con MongoDB), è necessario specificare una connessione AWS Glue che il crawler può utilizzare per connettersi all'archivio dati. Per Amazon S3, puoi facoltativamente specificare una connessione di tipo Rete. Una connessione è un oggetto del catalogo dati che memorizza informazioni di connessione, ad esempio credenziali, URL, informazioni su Amazon Virtual Private Cloud e altro ancora. Per ulteriori informazioni, consulta Connessione ai dati.

Di seguito sono elencate le versioni dei driver supportate dal crawler:

Product	Driver supportato da Crawler
PostgreSQL	42.2.1
Amazon Aurora	Uguali ai driver crawler nativi

Product	Driver supportato da Crawler
MariaDB	8.0.13
Microsoft SQL Server	6.1.0
MySQL	8.0.13
Oracle	11.2.2
Amazon Redshift	4.1
Snowflake	3,13,20
MongoDB	4,7,2
MongoDB Atlas	4,7,2

Di seguito sono elencate le note sui vargli archivi dati.

Amazon S3

Puoi scegliere di eseguire il crawling di un percorso nel tuo account o in un altro account. Se tutti i file Amazon S3 in una cartella hanno lo stesso schema, il crawler crea una tabella. Inoltre, se l'oggetto Amazon S3 è partizionato, viene creata solo una tabella di metadati e le informazioni di partizionamento vengono aggiunte al catalogo dati per tale tabella.

Amazon S3 e Amazon DynamoDB

I crawler utilizzano un ruolo AWS Identity and Access Management (IAM) per l'autorizzazione ad accedere ai tuoi archivi di dati. Il ruolo passato al crawler deve avere l'autorizzazione per accedere ai percorsi Amazon S3 e alle tabelle Amazon DynamoDB di cui viene eseguito il crawling.

Amazon DynamoDB

Quando definisci un crawler usando la console AWS Glue, devi specificare una tabella DynamoDB. Se usi l'API di AWS Glue, puoi specificare un elenco di tabelle. È possibile scegliere di eseguire il crawling solo di un piccolo campione di dati per ridurre i tempi di esecuzione del crawler.

Delta Lake

Per ogni archivio dati Delta Lake, specifichi la modalità di creazione delle tabelle Delta:

- Creazione di tabelle native: consente l'integrazione con i motori di query che supportano l'interrogazione diretta del log delle transazioni Delta. Per ulteriori informazioni, consulta Interrogare le tabelle Delta Lake.
- Creazione di tabelle Symlink: crea una cartella `_symlink_manifest` con file manifesti partizionati con chiavi di partizioni in base ai parametri di configurazione specificati.

Iceberg

Per ogni datastore Iceberg, specifichi un percorso Amazon S3 che contiene i metadati per le tabelle Iceberg. Se il crawler rileva i metadati della tabella Iceberg, li registra in Catalogo dati. È possibile impostare una pianificazione per il crawler per mantenere aggiornate le tabelle.

È possibile definire questi parametri per il datastore:

- Esclusioni: consente di ignorare determinate cartelle.
- Profondità di attraversamento massima: imposta il limite di profondità che il crawler può esplorare nel bucket Amazon S3. La profondità di attraversamento massima predefinita è 10, mentre la profondità massima che puoi impostare è 20.

Hudi

Per ogni datastore Iceberg, specifichi un percorso Amazon S3 che contiene i metadati per le tabelle Hudi. Se il crawler rileva i metadati della tabella Hudi, li registra in Catalogo dati. È possibile impostare una pianificazione per il crawler per mantenere aggiornate le tabelle.

È possibile definire questi parametri per il datastore:

- Esclusioni: consente di ignorare determinate cartelle.
- Profondità di attraversamento massima: imposta il limite di profondità che il crawler può esplorare nel bucket Amazon S3. La profondità di attraversamento massima predefinita è 10, mentre la profondità massima che puoi impostare è 20.

> **Note**
>
> Le colonne di timestamp con tipi logici `millis` verranno interpretate come `bigint` a causa di un'incompatibilità con Hudi 0.13.1 e i tipi di timestamp. Una risoluzione potrebbe essere fornita nella prossima versione di Hudi.

Le tabelle Hudi sono classificate come segue, con implicazioni specifiche per ciascuna di esse:

- Copia in scrittura (CoW): i dati vengono archiviati in un formato colonnare (Parquet) e ogni aggiornamento crea una nuova versione dei file durante una scrittura.
- Unisci in lettura (MoW): i dati vengono archiviati utilizzando una combinazione di formati colonnare (Parquet) e basati su righe (Avro). Gli aggiornamenti vengono registrati nei file delta basati su righe e vengono compattati come necessario per creare nuove versioni dei file colonnari.

Con i set di dati CoW, ogni volta che c'è un aggiornamento a un record, il file che contiene il record viene riscritto con i valori aggiornati. Quando si lavora con un set di dati MoR, ogniqualvolta è disponibile un aggiornamento Hudi scrive solo la riga per il registro modificato. MoR è più adatto per carichi di lavoro pesanti in scrittura o modifiche con meno letture. CoW è più adatto per carichi di lavoro pesanti di lettura su dati che cambiano meno frequentemente.

Hudi fornisce tre tipi di query per accedere ai dati:

- Query snapshot: query che visualizzano l'ultimo snapshot della tabella a partire da una determinata operazione di commit o compattazione. Per le tabelle MoR, le query snapshot espongono lo stato più recente della tabella unendo i file di base e delta della parte di file più recente al momento della query.
- Query incrementali: le query vedono solo i nuovi dati scritti nella tabella dal momento di un determinato commit/compattazione. Questo fornisce in modo efficace flussi di modifica per abilitare pipeline di dati incrementali.
- Query ottimizzate per la lettura (ReadOptimized): per le tabelle MoR, le query vedono i dati compattati più recenti. Per le tabelle CoW, le query vedono i dati più recenti impegnati.

Per le tabelle Copy-On-Write, i crawler creano una singola tabella nel Data Catalog con il serde. ReadOptimized `org.apache.hudi.hadoop.HoodieParquetInputFormat`

Per le tabelle Unisci in lettura, il crawler crea due tabelle in Catalogo dati per la stessa posizione della tabella:

- Una tabella con suffisso che utilizza il serde. `_ro` ReadOptimized `org.apache.hudi.hadoop.HoodieParquetInputFormat`
- Una tabella con suffisso `_rt` che utilizza RealTime Serde che consente di eseguire query Snapshot:. `org.apache.hudi.hadoop.realtime.HoodieParquetRealtimeInputFormat`

MongoDB e Amazon DocumentDB (compatibile con MongoDB)

Sono supportate le versioni 3.2 e successive di MongoDB. È possibile scegliere di eseguire il crawling solo di un piccolo campione di dati per ridurre i tempi di esecuzione del crawler.

Database relazionale

L'autenticazione avviene con un nome utente e una password del database. A seconda del tipo di motore di database, è possibile scegliere quali oggetti sono sottoposti a crawling, ad esempio database, schemi e tabelle.

Snowflake

Il crawler JDBC Snowflake supporta il crawling della tabella, della tabella esterna, della vista e della vista materializzata. La definizione della vista materializzata non verrà compilata.

Per le tabelle esterne Snowflake, il crawler eseguirà il crawling se punta a una posizione Amazon S3. Oltre allo schema della tabella, il crawler eseguirà il crawling anche della posizione di Amazon S3, del formato del file e dell'output come parametri di tabella nella tabella del catalogo di dati. Tenere presente che le informazioni sulla partizione della tabella esterna con partizioni non sono compilate.

ETL non è attualmente supportato per le tabelle del catalogo dati create utilizzando il crawler Snowflake.

Come funzionano i crawler

Quando un crawler viene eseguito, è necessario effettuare le seguenti operazioni per interrogare un archivio dati:

- Classifica dei dati per determinare il formato, lo schema e le relative proprietà dei dati grezzi : è possibile configurare i risultati di classificazione creando un classificatore personalizzato.
- Raggruppamento di dati in tabelle o partizioni : i dati vengono raggruppati in base a processi euristici del crawler.
- Scrittura dei metadati nel catalogo dati: è possibile configurare il modo in cui il crawler aggiunge, aggiorna ed elimina tabelle e partizioni.

Quando si definisce un crawler, si scelgono uno o più classificatori che valutano il formato dei dati per dedurre uno schema. Quando il crawler viene eseguito, il primo classificatore nella lista

per riconoscere con successo gli archivi dati è utilizzato per creare uno schema per la tabella. È possibile utilizzare classificatori incorporati o definirne uno proprio. Definisci i classificatori predefiniti in un'operazione separata prima di definire i crawler. AWS Glue fornisce classificatori predefiniti per dedurre gli schemi da file comuni con formati che includono JSON, CSV e Apache Avro. Per l'elenco aggiornato dei classificatori integrati nel AWS Glue, consulta Classificatori predefiniti in AWS Glue.

Le tabelle di metadati create dal crawler sono contenute in un database nel momento in cui si definisce il crawler. Se il crawler non specifica un database, le tabelle sono posizionate in un database di default. Inoltre, ogni tabella ha una colonna di classificazione che viene compilata dal classificatore che per primo ha riconosciuto con successo l'archivio dati.

Se il file sottoposto a crawling è compresso, il crawler deve scaricarlo per elaborarlo. Quando viene eseguito, un crawler interroga i file per determinarne il formato e il tipo di compressione e scrive queste proprietà nel catalogo dati. Alcuni formati di file, ad esempio Apache Parquet, permettono di comprimere parti del file non appena vengono scritte. Per questi file, i dati compressi sono un componente interno del file e AWS Glue non popola la proprietà `compressionType` quando scrive tabelle nel catalogo dati. Al contrario, se un intero file viene compresso tramite un algoritmo di compressione, ad esempio gzip, la proprietà `compressionType` viene popolata quando vengono scritte tabelle nel catalogo dati.

Il crawler genera i nomi per le tabelle che crea. I nomi delle tabelle archiviate nel AWS Glue Data Catalog seguono queste regole:

- Sono ammessi solo caratteri alfanumerici e caratteri di sottolineatura (_).
- Qualsiasi prefisso personalizzato non può essere più lungo di 64 caratteri.
- La lunghezza massima del nome non può essere superiore a 128 caratteri. Il crawler tronca i nomi generati per rientrare nel limite.
- Se vengono rilevati nomi di tabelle duplicati, il crawler aggiunge al nome un suffisso stringa hash.

Se il crawler viene eseguito più di una volta, su pianificazione, cerca file nuovi o modificati o tabelle nel tuo archivio dati. L'output del crawler include nuove tabelle e partizioni trovate nel corso di un'esecuzione precedente.

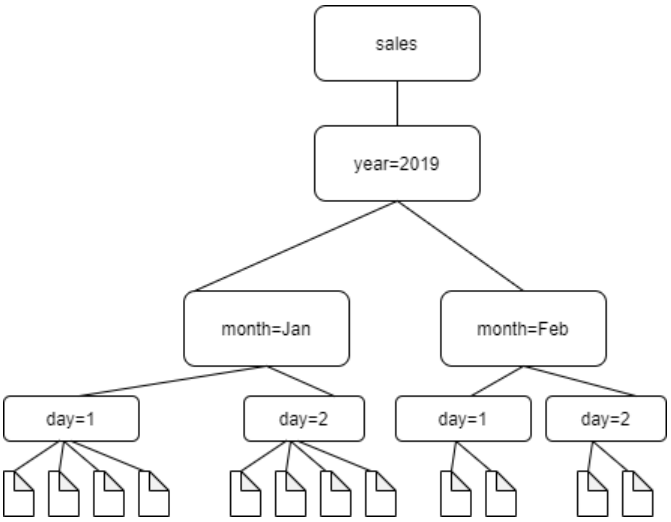
Argomenti

- In che modo un crawler determina quando creare le partizioni?
- Crawl incrementali per l'aggiunta di nuove partizioni in AWS Glue

In che modo un crawler determina quando creare le partizioni?

Quando un crawler AWS Glue analizza Amazon S3 e rileva più cartelle in un bucket, determina la root di una tabella nella struttura di cartelle e quali cartelle sono partizioni di una tabella. Il nome della tabella si basa sul prefisso Amazon S3 o sul nome della cartella. Si deve specificare un Include path (Percorso di inclusione) che indichi il livello della cartella su cui eseguire il crawling. Quando la maggior parte degli schemi a un livello della cartella sono simili, il crawler crea partizioni di una tabella invece di tabelle separate. Per fare in modo che il crawler crei tabelle separate, aggiungere ogni cartella radice della tabella come archivio dati separato quando si definisce il crawler.

Considera, ad esempio, la seguente struttura di cartelle Amazon S3.



I percorsi alle quattro cartelle più in basso sono i seguenti:

```
S3://sales/year=2019/month=Jan/day=1
S3://sales/year=2019/month=Jan/day=2
S3://sales/year=2019/month=Feb/day=1
S3://sales/year=2019/month=Feb/day=2
```

Supponiamo che la destinazione del crawler sia impostata su `Sales` e che tutti i file nelle cartelle `day=`*n* siano nello stesso formato (ad esempio, JSON, non crittografato) e abbiano schemi uguali o molto simili. Il crawler creerà una singola tabella con quattro partizioni, con chiavi di partizione `year`, `month`, e `day`.

Nel prossimo esempio consideriamo la seguente struttura di cartelle Amazon S3:

```
s3://bucket01/folder1/table1/partition1/file.txt
s3://bucket01/folder1/table1/partition2/file.txt
s3://bucket01/folder1/table1/partition3/file.txt
s3://bucket01/folder1/table2/partition4/file.txt
s3://bucket01/folder1/table2/partition5/file.txt
```

Se gli schemi per i file sotto `table1` e `table2` sono simili e nel crawler viene definito un unico archivio dati con Include path (Percorso di inclusione) `s3://bucket01/folder1/`, il crawler crea un'unica tabella con due colonne delle chiavi di partizione. La prima colonna delle chiavi di partizione contiene `table1` e `table2` e la seconda colonna delle chiavi di partizione contiene da `partition1` a `partition3` per la partizione `table1` e `partition4` e `partition5` per la partizione `table2`. Per creare due tabelle separate, definire il crawler con due archivi dati. In questo esempio, definire il primo Include path (Percorso di inclusione) come `s3://bucket01/folder1/table1/` e il secondo come `s3://bucket01/folder1/table2`.

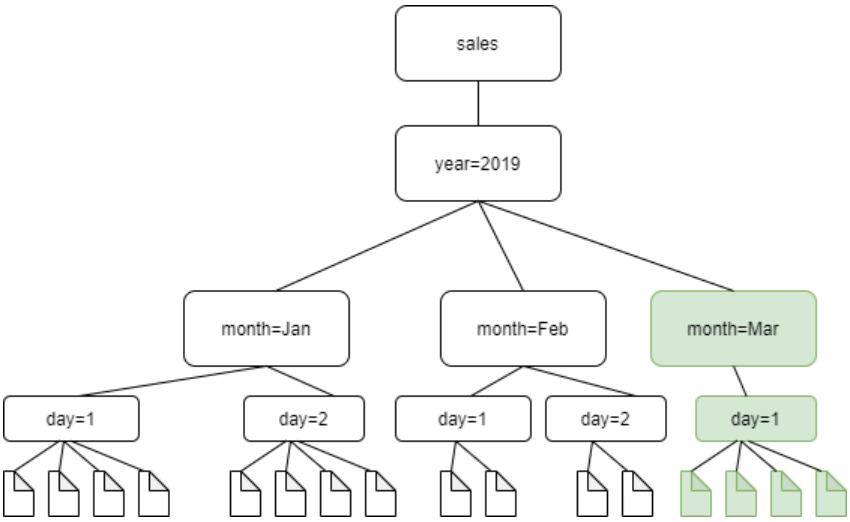
> **Note**
>
> In Amazon Athena, ogni tabella corrisponde a un prefisso Amazon S3 con tutti gli oggetti contenuti. Se gli oggetti hanno schemi diversi, Athena non riconosce gli oggetti diversi all'interno dello stesso prefisso come tabelle separate. Questo può verificarsi se un crawler crea più tabelle dallo stesso prefisso Amazon S3. In questo caso, alcune query in Athena possono restituire zero risultati. Perché Athena riconosca correttamente le tabelle ed esegua query sulle tabelle, crea il crawler con un Include path (Percorso di inclusione) separato per ogni schema di tabella diverso nella struttura di cartelle Amazon S3. Per ulteriori informazioni, consulta Best practice durante l'utilizzo di Athena con AWS Glue e questo articolo del Portale del sapere AWS.

Crawl incrementali per l'aggiunta di nuove partizioni in AWS Glue

Il crawler offre un'opzione per aggiungere nuove partizioni con conseguente crawling più rapidi per set di dati incrementali con uno schema di tabella stabile. Il caso d'uso tipico è per i crawler pianificati, dove durante ogni crawling vengono aggiunte nuove partizioni. Quando questa opzione è attivata, eseguirà innanzitutto un crawling completo sul set di dati di destinazione per consentire al crawler di registrare lo schema iniziale e la struttura della partizione. Durante un nuovo processo di crawling, verranno aggiunte nuove partizioni solo alle tabelle esistenti se i loro schemi sono compatibili. Non vengono apportate modifiche allo schema e non verranno aggiunte nuove tabelle al Data Catalog dopo la prima esecuzione per indicizzazione.

Puoi utilizzare questa opzione per configurare un'origine dati Amazon S3. Puoi impostare la `RecrawlPolicy` con `RecrawlBehavior` come "Crawl_New_Folders" nell'API `CreateCrawler` oppure le esecuzioni di crawler successivo come Effettua il crawling solo nelle nuove sottocartelle nella console.

Continuando con l'esempio in the section called “In che modo un crawler determina quando creare le partizioni?”, il diagramma seguente mostra che sono stati aggiunti file per il mese di marzo.



Se imposti `RecrawlBehavior` con l'opzione "Crawl_New_Folders", solo la nuova cartella `month=Mar` viene sottoposta a crawling.

Note e restrizioni

Quando questa opzione è attivata, non è possibile modificare gli archivi dati di destinazione Amazon S3 quando si modifica il crawler. Questa opzione influisce su alcune impostazioni di configurazione del crawler. Quando è attivata, impone il comportamento di aggiornamento e di eliminazione del crawler a `LOG`. Ciò significa che:

- Se rileva oggetti con schemi non compatibili, il crawler non aggiungerà gli oggetti nel Data Catalog e aggiungerà questi dettagli come log in Logs. CloudWatch
- Non aggiornerà gli oggetti eliminati nel catalogo dati.

Per ulteriori informazioni, consulta the section called “Impostazione delle opzioni di configurazione del crawler”.

Prerequisiti del crawler

Il crawler dispone delle autorizzazioni del ruolo AWS Identity and Access Management (IAM) specificate al momento della sua definizione. Questo ruolo IAM deve avere le autorizzazioni necessarie per estrarre dati dall'archivio dati e scriverli nel catalogo dati. La console AWS Glue elenca solo i ruoli IAM cui è collegata una policy di trust per il servizio dell'entità principale AWS Glue. Dalla console puoi anche creare un ruolo IAM con una policy IAM per accedere ad archivi dati Amazon S3 cui accede il crawler. Per ulteriori informazioni su come specificare ruoli per AWS Glue, consulta Policy basate su identità per AWS Glue.

> **Note**
>
> Durante il crawling di un datastore Delta Lake, è necessario disporre delle autorizzazioni di lettura/scrittura per la posizione Amazon S3.

Per il crawler, è possibile creare un ruolo e allegare le seguenti policy:

- La policy `AWSGlueServiceRole` gestita da AWS, che concede le autorizzazioni necessarie per il catalogo dati
- Una policy inline che concede le autorizzazioni per l'origine dati.

Un approccio più rapido consiste nel lasciare che la procedura guidata del crawler della console AWS Glue crei un ruolo per te. Il ruolo che crea è specifico per il crawler e include la policy gestita `AWSGlueServiceRole` AWS oltre alla policy inline richiesta per l'origine dati specificata.

Se si specifica un ruolo esistente per un crawler, bisogna assicurarsi che includa la policy `AWSGlueServiceRole` o equivalente (o una versione ridotta di questa policy), oltre alle policy inline richieste. Ad esempio, per un archivio dati Amazon S3, la policy inline sarebbe almeno la seguente:

```
{
   "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
              "s3:GetObject",
              "s3:PutObject"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::bucket/object*"
            ]
        }
    ]
}
```

Per un archivio dati Amazon DynamoDB, la policy sarebbe almeno la seguente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:Scan"
      ],
      "Resource": [
        "arn:aws:dynamodb:region:account-id:table/table-name*"
      ]
    }
  ]
}
```

Se il crawler legge i dati crittografati Amazon S3 AWS Key Management Service (AWS KMS), il ruolo IAM deve avere l'autorizzazione alla decrittografia sulla chiave AWS KMS. Per ulteriori informazioni, consulta Fase 2: creare un ruolo IAM per AWS Glue.

Proprietà del crawler

Quando si definisce un crawler utilizzando la console AWS Glue o l'AWS Glue API, è necessario specificare le informazioni riportate di seguito.

Fase 1: Configurazione delle proprietà del crawler

Nome

Il nome può contenere un massimo di 255 caratteri, tra cui lettere (A-Z), numeri (0-9), trattini (-) o caratteri di sottolineatura (_).

Descrizione

La descrizione può contenere un massimo di 2048 caratteri.

Tag

Utilizza i tag per organizzare e identificare le risorse. Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS tag in AWS Glue

Fase 2: Scelta delle origini dei dati e dei classificatori

Configurazione dell'origine dati

Seleziona l'opzione appropriata per I tuoi dati sono già mappati alle tabelle AWS Glue?

Il crawler è in grado di accedere agli archivi dati direttamente come origine del crawling oppure utilizza le tabelle di catalogo esistenti come origine. Se il crawler utilizza le tabelle di catalogo esistenti, esegue il crawling degli archivi dati specificati dalle tabelle di catalogo. Per ulteriori informazioni, consulta Tipo di origine del crawler.

- Non ancora: seleziona una o più origini dati da sottoporre a crawling. Un crawler è in grado di eseguire il crawling di più archivi dati di diversi tipi (Amazon S3, JDBC e così via).

  È possibile configurare un solo archivio dati alla volta. Dopo aver fornito le informazioni di connessione e incluso percorsi ed escluso i modelli, è possibile aggiungere un altro archivio dati.

  Per ulteriori informazioni, consulta Tipo di origine del crawler.
- Sì: seleziona le tabelle esistenti dal tuo catalogo dati AWS Glue. Le tabelle del catalogo specificano gli archivi dati da sottoporre a crawling. In una singola esecuzione, il crawler è in grado di eseguire il crawling solo di tabelle di catalogo; non può combinare altri tipi di origine.

Origini dati

Seleziona o aggiungi l'elenco di origini dati che devono essere scansionate dal crawler.

Percorso di inclusione

Per un archivio dati Amazon S3

Scegli se specificare un percorso in questo account o in uno differente, quindi seleziona un percorso Amazon S3.

Per un archivio dati Delta Lake

Specifica uno o più percorsi Amazon S3 per le tabelle Delta, come s3://*bucket*/*prefix*/*object*.

Per un datastore Iceberg o Hudi

Specifica uno o più percorsi Amazon S3 che contengono cartelle con i metadati delle tabelle Iceberg o Hudi come s3://*bucket*/*prefix*.

Per un datastore Hudi, la cartella Hudi può trovarsi in una cartella figlia della principale. Il crawler scansionerà tutte le cartelle al di sotto del percorso di una cartella Hudi.

Per un archivio dati JDBC

Inserisci *<database>*/*<schema>*/*<table>* o *<database>*/*<table>*, a seconda del prodotto di database. Oracle Database e MySQL non supportano lo schema nel percorso. È possibile sostituire il carattere di percentuale (%) per *<schema>* o *<table>*. Ad esempio, per un database Oracle con un identificatore di sistema (SID) di `orcl`, immettere `orcl/%` per importare tutte le tabelle a cui può accedere l'utente denominato nella connessione.

> **Important**
>
> Questo campo fa distinzione tra minuscole e maiuscole.

Per un archivio dati MongoDB, MongoDB Atlas o Amazon DocumentDB

Inserire *database*/*collection*.

Per ulteriori informazioni, consulta Modelli di inclusione e di esclusione.

Profondità trasversale massima (solo per i datastore Iceberg o Hudi)

Definisce la profondità massima del percorso Amazon S3 che il crawler può percorrere per scoprire la cartella di metadati Iceberg o Hudi nel percorso Amazon S3. Lo scopo di questo parametro è limitare il tempo di esecuzione del crawler. Il valore predefinito è 10, il valore massimo è 20.

Modelli di esclusione

Consentono di escludere alcuni file o tabelle dal crawling. Per ulteriori informazioni, consulta Modelli di inclusione e di esclusione.

Ulteriori parametri per l'origine del crawler

Ogni tipo di origine richiede un diverso set di parametri aggiuntivi. Di seguito è riportato un elenco incompleto:

Connessione

Selezionare o aggiungere una connessione AWS Glue. Per informazioni sulle connessioni, consulta Connessione ai dati.

Metadati aggiuntivi: facoltativi (per gli archivi di dati JDBC)

Seleziona proprietà di metadati aggiuntive per il crawler da sottoporre a crawling.

- Commenti: esegui il crawling dei commenti associati a livello di tabella e colonna.
- Tipi non elaborati: mantengono i tipi di dati non elaborati delle colonne della tabella in metadati aggiuntivi. Come comportamento predefinito, il crawler traduce i tipi di dati non elaborati in tipi compatibili con Hive.

Nome della classe del driver JDBC: facoltativo (per i datastore JDBC)

Digita un nome di classe del driver JDBC personalizzato per consentire al crawler di connettersi all'origine dati:

- Postgres: org.postgresql.Driver
- MySQL: com.mysql.jdbc.Driver, com.mysql.cj.jdbc.Driver
- Redshift: com.amazon.redshift.jdbc.Driver, com.amazon.redshift.jdbc42.Driver
- Oracle: oracle.jdbc.driver. OracleDriver
- SQL Server: com.microsoft.sqlserver.jdbc.SQL ServerDriver

Percorso S3 del driver JDBC: facoltativo (per i datastore JDBC)

Scegli un percorso Amazon S3 esistente verso un file `.jar`. Questa è la posizione in cui verrà archiviato il file `.jar` quando si utilizza un driver JDBC personalizzato per la connessione del crawler all'origine dati.

Abilitazione del campionamento dei dati (solo per gli archivi dati Amazon DynamoDB, MongoDB, MongoDB Atlas e Amazon DocumentDB)

Selezionare se eseguire il crawling solo di un campione di dati. Se non è selezionata, l'intera tabella viene sottoposta a crawling. La scansione di tutti i registri può richiedere molto tempo quando la tabella non è una tabella di throughput elevato.

Creazione di tabelle per l'esecuzione di query (solo per gli archivi dati Delta Lake)

Seleziona come desideri creare le tabelle Delta Lake:

- Creazione di tabelle native: consente l'integrazione con i motori di query che supportano l'interrogazione diretta del log delle transazioni Delta.
- Creazione di tabelle Symlink: crea una cartella dei manifesti Symlink con file manifesti partizionati con chiavi di partizioni in base ai parametri di configurazione specificati.

Velocità di scansione (facoltativo, solo per i datastore DynamoDB)

Specifica la percentuale delle unità di capacità di lettura della tabella DynamoDB che deve essere utilizzata dal crawler. L'unità di capacità di lettura è un termine definito da DynamoDB ed è un valore numerico che funge da limitatore di velocità per il numero di letture che possono essere eseguite su tale tabella al secondo. Inserire un valore tra 0.1 e 1.5. Se non specificato, il valore predefinito è 0,5% per le tabelle con provisioning e 1/4 della capacità massima configurata per le tabelle on demand. Con i crawler AWS Glue è necessario utilizzare solo la modalità di capacità assegnata.

> **Note**
>
> Per gli archivi dati DynamoDB, impostare la modalità di capacità assegnata per l'elaborazione delle letture e delle scritture sulle tabelle. Non utilizzare il crawler AWS Glue con la modalità di capacità on demand.

Connessione di rete: facoltativo (solo per gli archivi dati Amazon S3)

Facoltativamente, includi una connessione di rete da utilizzare con questa destinazione Amazon S3. Tieni presente che ogni crawler è limitato a una connessione di rete, quindi anche qualsiasi altra destinazione Amazon S3 utilizzerà la stessa connessione (o nessuna, se lasciata vuota).

Per informazioni sulle connessioni, consulta Connessione ai dati.

Campionamento di un solo sottoinsieme di file e dimensioni del campione (solo per gli archivi dati Amazon S3)

Specificare il numero di file in ogni cartella foglia da sottoporre al crawling durante il crawling di file di esempio in un set di dati. Quando questa caratteristica è attivata, invece di eseguire il crawling di tutti i file in questo set di dati, il crawler seleziona in modo casuale alcuni file in ogni cartella foglia da sottoporre a crawling.

Il crawler di campionamento è adatto ai clienti che hanno conoscono i loro formati di dati e sanno che gli schemi nelle loro cartelle non cambiano. L'attivazione di questa funzione ridurrà significativamente il tempo di esecuzione del crawler.

Un valore valido è un numero intero compreso tra 1 e 249. Se non è specificato, tutti i file vengono sottoposti al crawling.

Esecuzioni successive del crawler

Questo campo è un campo globale che riguarda tutte le origini dati Amazon S3.

- Esecuzione del crawling di tutte le sottocartelle: esegui nuovamente il crawling di tutte le cartelle ad ogni crawling successivo.
- Esecuzione del crawling solo delle nuove sottocartelle: verrà eseguito il crawling solo delle cartelle Amazon S3 che sono state aggiunte dall'ultimo crawling. Se gli schemi sono compatibili, verranno aggiunte nuove partizioni alle tabelle esistenti. Per ulteriori informazioni, consulta the section called “Crawl incrementali per l'aggiunta di nuove partizioni in AWS Glue”.
- Crawling in base agli eventi: basati sugli eventi di Amazon S3 per controllare quali cartelle sottoporre a crawling. Per ulteriori informazioni, consulta the section called “Accelerazione del crawling con le notifiche eventi Amazon S3”.

Classificatori personalizzati: facoltativi

Definisci i classificatori personalizzati prima di definire i crawler. Un classificatore verifica se un determinato file è in un formato che può essere gestito dal crawler. In questo caso il classificatore crea uno schema nel formato di un oggetto `StructType` che corrisponde a quel formato di dati.

Per ulteriori informazioni, consulta Aggiunta di classificatori a un crawler in AWS Glue.

Fase 3: configurare le impostazioni di sicurezza

Ruolo IAM

Il crawler assume questo ruolo. Deve avere autorizzazioni alla policy gestita AWS `AWSGlueServiceRole`. Per le origini Amazon S3 e DynamoDB, deve disporre anche delle autorizzazioni per accedere all'archivio dati. Se il crawler legge i dati crittografati Amazon S3 AWS Key Management Service (AWS KMS), il ruolo deve avere le autorizzazioni alla decrittografia sulla chiave AWS KMS.

Per un archivio dati Amazon S3, autorizzazioni aggiuntive collegate al ruolo saranno simili alle seguenti:

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

Per un archivio dati Amazon DynamoDB, autorizzazioni aggiuntive collegate al ruolo saranno simili alle seguenti:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:Scan"
      ],
      "Resource": [
        "arn:aws:dynamodb:region:account-id:table/table-name*"
      ]
    }
  ]
}
```

Per ulteriori informazioni, consultare Fase 2: creare un ruolo IAM per AWS Glue e Gestione delle identità e degli accessi per AWS Glue.

Configurazione Lake Formation: facoltativa

Consenti al crawler di utilizzare le credenziali di Lake Formation per il crawling dell'origine dati.

Selezionando Use Lake Formation credentials for crawling S3 data source (Usa le credenziali di Lake Formation per il crawling dell'origine dati S3), il crawler potrà utilizzare le credenziali di Lake Formation per il crawling dell'origine dati. Se l'origine appartiene a un altro account, è necessario fornire l'ID dell'account registrato. Altrimenti, il crawler eseguirà il crawling solo delle origini dati associate all'account. Applicabile solo a origini dati Amazon S3 e del catalogo dati.

Configurazione di sicurezza: facoltativa

Le impostazioni includono le configurazioni di sicurezza. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Crittografia dei dati scritti da AWS Glue

> **Note**
>
> Una volta impostata una configurazione di sicurezza su un crawler, puoi modificarla, ma non puoi rimuoverla. Per ridurre il livello di sicurezza su un crawler, imposta esplicitamente la funzionalità di sicurezza all'`DISABLED`interno della tua configurazione o crea un nuovo crawler.

Fase 4: Configurazione dell'output e della pianificazione

Configurazione dell'output

Le opzioni includono in che modo il crawler deve gestire le modifiche dello schema rilevate, gli oggetti eliminati nell'archivio dati e altro. Per ulteriori informazioni, consultare Impostazione delle opzioni di configurazione del crawler

Pianificazione del crawler

Puoi definire un crawler on demand oppure definire una pianificazione temporale per i crawler e i processi in AWS Glue. La definizione di queste pianificazioni usa la sintassi cron di tipo Unix. Per ulteriori informazioni, consulta Pianificazione di un crawler in AWS Glue.

Passaggio 5: revisione e creazione

Controlla le impostazioni del crawler configurate e crea il crawler.

Tipo di origine del crawler

Un crawler è in grado di accedere agli archivi dati direttamente come origine del crawling oppure utilizza le tabelle di catalogo esistenti come origine. Se il crawler utilizza le tabelle di catalogo esistenti, esegue il crawling degli archivi dati specificati dalle tabelle di catalogo.

Un motivo comune per specificare una tabella di catalogo come origine è quando la tabella viene creata manualmente (poiché si conosce già la struttura dell'archivio dati) e desideri che un crawler mantenga aggiornata la tabella, inclusa l'aggiunta di nuove partizioni. Per una discussione degli altri motivi, consulta Aggiornamento delle tabelle del catalogo dati create manualmente usando i crawler.

Quando specifichi le tabelle esistenti come tipo di origine crawler, si applicano le seguenti condizioni:

- Il nome del database è facoltativo.
- Solo le tabelle di catalogo che specificano archivi dati Amazon S3 o Amazon DynamoDB sono consentite.
- Nessuna nuova tabella di catalogo viene creata quando viene eseguito il crawler. Le tabelle esistenti vengono aggiornate in base alle esigenze, inclusa l'aggiunta di nuove partizioni.
- Gli oggetti eliminati trovati negli archivi dati vengono ignorati e non vengono eliminate le tabelle di catalogo. Al contrario, il crawler scrive un messaggio di log. (`SchemaChangePolicy.DeleteBehavior=LOG`)
- L'opzione di configurazione del crawler per creare un singolo schema per ogni percorso Amazon S3 è abilitata per impostazione predefinita e non può essere disattivata. (`TableGroupingPolicy=CombineCompatibleSchemas`) Per ulteriori informazioni, consulta Come creare un singolo schema per ogni percorso di inclusione Amazon S3
- Non è possibile combinare le tabelle di catalogo come origine con altri tipi di origine (ad esempio Amazon S3 o Amazon DynamoDB).

Modelli di inclusione e di esclusione

Per stabilire cosa includere o escludere dal crawling, il crawler inizia la valutazione del percorso di inclusione richiesto. Per gli archivi dati Amazon S3, MongoDB, MongoDB Atlas, Amazon DocumentDB (compatibile con MongoDB) e relazionali, è necessario specificare un percorso di inclusione.

Per gli archivi dati Amazon S3, la sintassi del percorso di inclusione è `bucket-name/folder-name/file-name.ext`. Per eseguire il crawling di tutti gli oggetti in un bucket, devi specificare solo

il nome del bucket nel percorso di inclusione. Il modello di esclusione è relativo rispetto al percorso di inclusione

Per MongoDB, MongoDB Atlas e Amazon DocumentDB (compatibile con MongoDB), la sintassi è `database/collection`.

Per gli archivi dati JDBC, la sintassi è `database-name/schema-name/table-name` oppure `database-name/table-name`. La sintassi dipende dal fatto che il motore di database supporti schemi all'interno di un database. Ad esempio, per i motori di database come MySQL o Oracle, non è necessario specificare un `schema-name` nel percorso di inclusione. È possibile sostituire il segno di percentuale (`%`) per uno schema o una tabella nel percorso di inclusione per rappresentare tutti gli schemi o tutte le tabelle all'interno di un database. Non è possibile sostituire il segno di percentuale (`%`) per il database nel percorso di inclusione. Il percorso di esclusione è relativo rispetto al percorso di inclusione. Ad esempio, per escludere una tabella nel tuo archivio dati JDBC, digita il nome della tabella nel percorso di esclusione.

Un crawler si connette a un archivio dati JDBC usando una connessione AWS Glue che contiene una stringa di connessione URI JDBC. Il crawler ha accesso a oggetti nel motore di database solo usando il nome utente e la password JDBC nella connessione AWS Glue. Il crawler può creare solo le tabelle cui può accedere tramite la connessione JDBC. Dopo che il crawler accede al motore del database con l'URI JDBC, viene usato il percorso di inclusione per determinare quali tabelle del motore del database vengono create nel catalogo dati. Ad esempio, se con MySQL specifichi il percorso di inclusione `MyDatabase/%`, nel catalogo dati verranno create tutte le tabelle presenti in `MyDatabase`. Se quando accedi ad Amazon Redshift specifichi il percorso di inclusione di `MyDatabase/%`, nel catalogo dati vengono create tutte le tabelle in tutti gli schemi per il database `MyDatabase`. Se si specifica un percorso di inclusione di `MyDatabase/MySchema/%`, vengono create tutte le tabelle nel database `MyDatabase` e lo schema `MySchema`.

Dopo aver specificato un percorso di inclusione, puoi quindi escludere dal crawling oggetti che il percorso di inclusione altrimenti includerebbe, specificando uno o più modelli di esclusione `glob` in stile Unix. Questi modelli vengono applicati al tuo percorso di inclusione per determinare quali oggetti sono esclusi. Questi modelli vengono memorizzati anche come proprietà di tabelle create dal crawler. AWS Glue PySpark estensioni, ad esempio`create_dynamic_frame.from_catalog`, leggono le proprietà della tabella ed escludono gli oggetti definiti dal modello di esclusione.

AWS Glue supporta i tipi di modelli `glob` seguenti nel modello di esclusione.

Modello di esclusione	Descrizione
`*.csv`	Individua un percorso Amazon S3 che rappresenta un nome di oggetto nella cartella corrente che termina con `.csv`
`*.*`	Individua tutti i nomi degli oggetti che contengono un punto
`*.{csv,avro}`	Individua i nomi di oggetti che terminano con `.csv` o `.avro`
`foo.?`	Individua i nomi degli oggetti che iniziano con `foo.` che sono seguiti da un'estensione di un singolo carattere.
`myfolder/*`	Individua gli oggetti in un livello di sottocart ella di `myfolder`, ad esempio `/myfolder/ mysource`
`myfolder/*/*`	Individua gli oggetti in due livelli di sottocart ella di `myfolder`, ad esempio `/myfolder/ mysource/data`
`myfolder/**`	Individua gli oggetti in tutte le sottocartelle di `myfolder`, ad esempio `/myfolder/ mysource/mydata` e `/myfolder/ mysource/data`
`myfolder**`	Individua la sottocartella `myfolder`, insieme ai file all'interno di `myfolder`, ad esempio `/ myfolder` e `/myfolder/mydata.txt`
`Market*`	Individua le tabelle in un database JDBC con i nomi che iniziano con `Market`, quali `Market_us` e `Market_fr`

AWS Glue interpreta i modelli di esclusione `glob` in questo modo:

- La barra (`/`) è il delimitatore per separare chiavi Amazon S3 in una gerarchia di cartelle.
- L'asterisco (`*`) individua zero o più caratteri di un componente del nome entro i limiti della cartella.
- Un doppio asterisco (`**`) individua zero o più caratteri al di fuori della cartella o dello schema.
- Il punto interrogativo (`?`) individua esattamente un carattere di un componente del nome.
- La barra rovesciata (`\`) è utilizzata per ignorare i caratteri che potrebbero essere interpretati come caratteri speciali. L'espressione `\\` individua una singola barra rovesciata, mentre `\{` a una parentesi aperta.
- Le parentesi `[ ]` creano un'espressione tra parentesi che individua un singolo carattere di un componente nome fuori da un set di caratteri. Ad esempio, `[abc]` individua `a`, `b` o `c`. Il trattino (`-`) può essere utilizzato per specificare un intervallo, quindi `[a-z]` specifica un intervallo `a` – `z` (estremi inclusi). Questi moduli possono essere mescolati, perciò [`abce-g`] individua `a`, `b`, `c`, `e`, `f` o `g`. Se il carattere dopo la parentesi (`[`) è un punto esclamativo (`!`), l'espressione all'interno della parentesi viene ignorata. Ad esempio, `[!a-c]` corrisponde a tutti i caratteri, tranne `a`, `b` o `c`.

  All'interno di un'espressione tra parentesi, i caratteri `*`, `?` e `\` corrispondono. Il trattino (`-`) corrisponde a se stesso se è il primo carattere all'interno della parentesi oppure se è il primo carattere dopo il `!` quando l'espressione viene ignorata.
- Le parentesi graffe (`{ }`) racchiudono un gruppo di sotto-modelli, dove il gruppo individua un eventuale sotto-modello corrispondente all'interno del gruppo. Una virgola (`,`) è utilizzata per separare i sotto-modelli. I gruppi non possono essere annidati.
- Il punto iniziale o i punti nei nomi di file vengono trattati come caratteri normali nelle operazioni di confronto. Ad esempio, il modello di esclusione `*` corrisponde al nome di file `.hidden`.

Example Modelli di esclusione Amazon S3

Ogni modello di esclusione è valutato rispetto al percorso di inclusione. Ad esempio, supponiamo di avere la seguente struttura di directory Amazon S3:

```
/mybucket/myfolder/
   departments/
      finance.json
      market-us.json
      market-emea.json
      market-ap.json
   employees/
      hr.json
      john.csv
```

```
    jane.csv
    juan.txt
```

Dato il percorso di inclusione `s3://mybucket/myfolder/`, i seguenti sono risultati di esempio per i modelli di esclusione:

Modello di esclusione	Risultati
`departments/**`	Esclude tutti i file e le cartelle sotto `departments` e include la cartella `employees` e i relativi file
`departments/market*`	Esclude `market-us.json` , `market-emea.json` e `market-ap.json`
`**.csv`	Esclude tutti gli oggetti sotto `myfolder` che hanno un nome che termina con `.csv`
`employees/*.csv`	Esclude tutti i file `.csv` nella cartella `employees`

Example Esclusione di un sottoinsieme di partizioni Amazon S3

Supponiamo che i dati vengano partizionati in base al giorno, in modo che ogni giorno di un anno si trovi in una partizione Amazon S3 separata. Per gennaio 2015, ci sono 31 partizioni. Ora, per eseguire il crawling solo per i dati della prima settimana di gennaio, devi escludere tutte le partizioni eccetto i giorni da 1 a 7:

```
2015/01/{[!0],0[8-9]}**, 2015/0[2-9]/**, 2015/1[0-2]/**
```

Diamo uno sguardo alle parti di questo modello glob. La prima parte, `2015/01/{[!0],0[8-9]}**`, esclude tutti i giorni che non iniziano con "0" oltre ai giorni 08 e 09 dal mese 01 nel 2015. Tieni presente che "**" viene utilizzato come suffisso per il modello del numero del giorno e si estende alle cartelle di livello inferiore. Se viene utilizzato "*", i livelli di cartella inferiori non sono esclusi.

La seconda parte, `2015/0[2-9]/**`, esclude i giorni nei mesi da 02 a 09, nel 2015.

La terza parte, `2015/1[0-2]/**`, esclude i giorni nei mesi 10, 11 e 12, nel 2015.

Example Modelli di esclusione JDBC

Supponiamo di eseguire il crawling di un database JDBC con la seguente struttura di schema:

```
MyDatabase/MySchema/
   HR_us
   HR_fr
   Employees_Table
   Finance
   Market_US_Table
   Market_EMEA_Table
   Market_AP_Table
```

Dato il percorso di inclusione `MyDatabase/MySchema/%`, i seguenti sono risultati di esempio per i modelli di esclusione:

Modello di esclusione	Risultati
`HR*`	Esclude le tabelle con nomi che iniziano con `HR`
`Market_*`	Esclude le tabelle con nomi che iniziano con `Market_`
`**_Table`	Esclude tutte le tabelle con nomi che terminano con `_Table`

Impostazione delle opzioni di configurazione del crawler

Durante l'esecuzione, un crawler può rilevare modifiche nel datastore che fanno sì che uno schema o una partizione siano diversi rispetto a un'operazione di crawling precedente. Puoi usare la AWS Management Console o l'API di AWS Glue per configurare il modo in cui il crawler elabora determinati tipi di modifiche.

Argomenti

- Impostazione delle opzioni di configurazione del crawler degli indici di partizione
- Come impedire al crawler di modificare uno schema esistente
- Come creare un singolo schema per ogni percorso di inclusione Amazon S3

- Come specificare la posizione della tabella e il livello di partizionamento
- Come specificare il numero massimo di tabelle che il crawler può creare
- Come specificare le opzioni di configurazione per un archivio dati Delta Lake
- Come configurare un crawler per utilizzare le credenziali di Lake Formation

Console

Quando definisci un crawler usando la console AWS Glue, puoi scegliere tra diverse opzioni per configurare il comportamento del crawler. Per ulteriori informazioni sull'uso della console AWS Glue per aggiungere un crawler, consulta Uso di crawler nella console AWS Glue.

Quando un crawler viene eseguito su un datastore precedentemente sottoposto a crawling, può rilevare che uno schema è stato modificato o che alcuni oggetti nel datastore sono stati eliminati. Il crawler registra le modifiche in uno schema. A seconda del tipo di origine per il crawler, è possibile che vengano create nuove tabelle e partizioni indipendentemente dalla policy di modifica dello schema.

Per specificare il comportamento del crawler quando trova modifiche nello schema, puoi scegliere una delle operazioni seguenti nella console:

- Update the table definition in the Data Catalog (Aggiorna definizione di tabella nel catalogo dati): puoi aggiungere nuove colonne, rimuovere colonne mancanti e modificare le definizioni di colonne esistenti in AWS Glue Data Catalog. Rimuove inoltre tutti i metadati non impostati dal crawler. Si tratta dell'impostazione di default.
- Add new columns only (Aggiungi solo nuove colonne): per le tabelle mappate a un datastore Amazon S3, puoi aggiungere nuove colonne man mano che vengono rilevate, senza rimuovere o modificare il tipo delle colonne esistenti nel catalogo dati. Scegli questa opzione quando le colonne correnti nel catalogo dati sono corrette e non vuoi che il crawler rimuova o modifichi il tipo delle colonne esistenti. Se un attributo essenziale di una tabella Amazon S3 cambia, ad esempio una classificazione, un tipo di compressione o un delimitatore per file CSV, contrassegna la tabella come obsoleta. Puoi mantenere il formato di input e di output esistenti nel catalogo dati. Aggiorna SerDe i parametri solo se il parametro è impostato dal crawler. Per tutti gli altri datastore, puoi modificare le definizioni delle colonne esistenti.
- Ignore the change and don't update the table in the Data Catalog (Ignora la modifica e non aggiornare la tabella nel catalogo dati): vengono create solo nuove tabelle e partizioni.

  Questa è l'impostazione predefinita per i crawling incrementali.

Un crawler potrebbe anche rilevare partizioni nuove o modificate. Per impostazione predefinita, le nuove partizioni vengono aggiunte e le partizioni esistenti vengono aggiornate se sono state modificate. Inoltre, puoi impostare l'opzione di configurazione del crawler Update all new and existing partitions with metadata from the table (Aggiorna tutte le partizioni nuove ed esistenti con metadati della tabella) nella console AWS Glue. Quando questa opzione è impostata, le partizioni ereditano le proprietà dei metadati, come la classificazione, il formato di input, il formato di output, le informazioni e lo schema, dalla tabella principale. SerDe Tutte le modifiche apportate a queste proprietà in una tabella vengono propagate alle partizioni. Quando questa opzione di configurazione è impostata su un crawler esistente, le partizioni esistenti vengono aggiornate in modo che corrispondano alle proprietà della tabella padre alla prossima esecuzione del crawler.

Per specificare il comportamento del crawler quando trova un oggetto eliminato nel datastore, scegli una delle operazioni seguenti:

- Elimina tabelle e partizioni dal catalogo dati
- Ignora la modifica e non aggiornare la tabella nel catalogo dati

  Questa è l'impostazione predefinita per i crawling incrementali.
- Mark the table as deprecated in the Data Catalog (Contrassegna la tabella come obsoleta nel catalogo dati: questa è l'impostazione predefinita.

AWS CLI

```
aws glue create-crawler \
--name "your-crawler-name" \
--role "your-iam-role-arn" \
--database-name "your-database-name" \
--targets 'S3Targets=[{Path="s3://your-bucket-name/path-to-data"}]' \
--configuration '{"Version": 1.0, "CrawlerOutput": {"Partitions":
 {"AddOrUpdateBehavior": "InheritFromTable"}, "Tables": {"AddOrUpdateBehavior":
 "MergeNewColumns"}}}'
```

API

Quando definisci un crawler usando l'API di AWS Glue, puoi scegliere tra diversi campi per configurare il crawler. Il campo `SchemaChangePolicy` nell'API del crawler determina il comportamento del crawler quando rileva uno schema modificato o un oggetto eliminato. Durante l'esecuzione, il crawler registra le modifiche dello schema.

Esempio di codice Python che mostra le opzioni di configurazione del crawler

```
import boto3
import json

# Initialize a boto3 client for AWS Glue
glue_client = boto3.client('glue', region_name='us-east-1')  # Replace 'us-east-1'
 with your desired AWS region

# Define the crawler configuration
crawler_configuration = {
    "Version": 1.0,
    "CrawlerOutput": {
        "Partitions": {
            "AddOrUpdateBehavior": "InheritFromTable"
        },
        "Tables": {
            "AddOrUpdateBehavior": "MergeNewColumns"
        }
    }
}

configuration_json = json.dumps(crawler_configuration)
# Create the crawler with the specified configuration
response = glue_client.create_crawler(
    Name='your-crawler-name',  # Replace with your desired crawler name
    Role='crawler-test-role',  # Replace with the ARN of your IAM role for Glue
    DatabaseName='default',  # Replace with your target Glue database name
    Targets={
        'S3Targets': [
            {
                'Path': "s3://your-bucket-name/path/",  # Replace with your S3 path
 to the data
            },
        ],
        # Include other target types like 'JdbcTargets' if needed
    },
    Configuration=configuration_json,
    # Include other parameters like Schedule, Classifiers, TablePrefix,
 SchemaChangePolicy, etc., as needed
)

print(response)a
```

Quando viene eseguito un crawler, vengono create sempre nuove tabelle e partizioni indipendentemente dalle policy di modifica dello schema. Puoi scegliere una delle operazioni seguenti nel campo `UpdateBehavior` nella struttura `SchemaChangePolicy` per determinare il comportamento del crawler quando trova uno schema di tabella modificato:

- `UPDATE_IN_DATABASE`: aggiorna la tabella nel AWS Glue Data Catalog. Aggiunge nuove colonne, rimuove le colonne mancanti e modifica le definizioni delle colonne esistenti. Rimuove inoltre tutti i metadati non impostati dal crawler.
- `LOG`: ignora le modifiche e non aggiorna la tabella nel catalogo dati.

  Questa è l'impostazione predefinita per i crawling incrementali.

Puoi anche sostituire la struttura `SchemaChangePolicy` usando un oggetto JSON fornito nel campo `Configuration` dell'API del crawler. Questo oggetto JSON può contenere una coppia chiave/valore per impostare la policy in modo da non aggiornare colonne esistenti e aggiungere solo nuove colonne. Ad esempio, puoi specificare l'oggetto JSON seguente come stringa:

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Tables": { "AddOrUpdateBehavior": "MergeNewColumns" }
   }
}
```

Questa opzione corrisponde all'opzione Add new columns only (Aggiungi solo nuove colonne) nella console AWS Glue. L'opzione sostituisce la struttura `SchemaChangePolicy` solo per tabelle derivanti dal crawling di datastore Amazon S3. Scegli questa opzione se vuoi mantenere i metadati esistenti nel catalogo dati ("source of truth"). Le nuove colonne vengono aggiunte man mano che vengono rilevate, inclusi i tipi di dati annidati. Tuttavia, le colonne esistenti non vengono rimosse e i rispettivi tipi non vengono modificati. Se un attributo di una tabella Amazon S3 cambia in misura significativa, puoi contrassegnare la tabella come obsoleta e registrare un avviso riguardo alla necessità di risolvere un attributo non compatibile. Questa opzione non è applicabile al crawler incrementale.

Quando un crawler viene eseguito su un datastore sottoposto precedentemente a crawling, potrebbe scoprire partizioni nuove o modificate. Per impostazione predefinita, le nuove partizioni

vengono aggiunte e le partizioni esistenti vengono aggiornate se sono state modificate. Inoltre, puoi impostare l'opzione di configurazione del crawler `InheritFromTable`, corrispondente all'opzione Update all new and existing partitions with metadata from the table (Aggiorna tutte le partizioni nuove ed esistenti con i metadati della tabella) nella console AWS Glue. Quando questa opzione è impostata, le partizioni ereditano le proprietà dei metadati dalla tabella principale, come la classificazione, il formato di input, il formato di output, le informazioni e lo schema. SerDe Tutte le modifiche di proprietà rilevate nella tabella madre vengono propagate alle partizioni.

Quando questa opzione di configurazione è impostata su un crawler esistente, le partizioni esistenti vengono aggiornate in modo che corrispondano alle proprietà della tabella padre alla prossima esecuzione del crawler. Questo comportamento viene impostato nel campo `Configuration` dell'API del crawler. Ad esempio, puoi specificare l'oggetto JSON seguente come stringa:

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" }
   }
}
```

Il campo `Configuration` dell'API del crawler può impostare più opzioni di configurazione. Ad esempio, per configurare l'output del crawler per partizioni e tabelle, puoi specificare una rappresentazione di stringa dell'oggetto JSON seguente:

```
{
    "Version": 1.0,
    "CrawlerOutput": {
       "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
       "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
    }
}
```

Puoi scegliere una delle operazioni seguenti per determinare il comportamento del crawler quando trova un oggetto eliminato nel datastore. Il campo `DeleteBehavior` nella struttura `SchemaChangePolicy` nell'API del crawler imposta il comportamento del crawler quando rileva un oggetto eliminato.

- `DELETE_FROM_DATABASE`: elimina tabelle e partizioni dal catalogo dati.

- `LOG`: ignora la modifica. Non aggiornare il catalogo dati. Scrivi un messaggio di log.
- `DEPRECATE_IN_DATABASE`: contrassegna la tabella come obsoleta nel catalogo dati. Si tratta dell'impostazione di default.

Impostazione delle opzioni di configurazione del crawler degli indici di partizione

Catalogo dati supporta gli indici di partizione per fornire una ricerca efficiente di partizioni specifiche. Per ulteriori informazioni, consulta la pagina Working with partition indexes in AWS Glue. Per impostazione predefinita, il AWS Glue crawler crea indici di partizione per le destinazioni Amazon S3 e Delta Lake.

Quando definisci un crawler, l'opzione per la creazione automatica degli indici di partizione è abilitata per impostazione predefinita in Opzioni avanzate nella pagina Imposta output e pianificazione.

Per disabilitare questa opzione, puoi deselezionare la casella di controllo Crea automaticamente gli indici delle partizioni nella console. Puoi anche disabilitare questa opzione utilizzando l'API del crawler, impostata in. `CreatePartitionIndex  Configuration` Il valore di default è true.

Note di utilizzo sugli indici di partizione

- Le tabelle create dal crawler non hanno la variabile `partition_filtering.enabled` per impostazione predefinita. Per ulteriori informazioni, consulta la pagina AWS Glue partition indexing and filtering.
- La creazione di indici di partizione per partizioni crittografate non è supportata.

Come impedire al crawler di modificare uno schema esistente

Se non vuoi che un crawler sovrascriva gli aggiornamenti apportati a campi esistenti in una definizione di tabella Amazon S3, scegli l'opzione Add new columns only (Aggiungi solo nuove colonne) nella console oppure imposta l'opzione di configurazione `MergeNewColumns`. Questa opzione si applica a tabelle e partizioni, a meno che il campo `Partitions.AddOrUpdateBehavior` non venga sostituito da `InheritFromTable`.

Se non vuoi che uno schema di tabella venga modificato in alcun modo durante l'esecuzione di un crawler, imposta la policy di modifica dello schema su `LOG`. Puoi anche specificare un'opzione di configurazione che imposta gli schemi delle partizioni in modo da ereditare dalla tabella.

Se stai configurando il crawler nella console, puoi scegliere tra le operazioni seguenti:

- Ignora la modifica e non aggiornare la tabella nel catalogo dati
- Update all new and existing partitions with metadata from the table (Aggiorna tutte le partizioni nuove ed esistenti con metadati della tabella)

Quando configuri il crawler tramite l'API, imposta i parametri seguenti:

- Imposta il campo `UpdateBehavior` nella struttura `SchemaChangePolicy` su `LOG`.
- Imposta il campo `Configuration` con una rappresentazione di stringa dell'oggetto JSON seguente nell'API del crawler, ad esempio:

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" }
   }
}
```

Come creare un singolo schema per ogni percorso di inclusione Amazon S3

Per impostazione predefinita, quando un crawler definisce tabelle per i dati archiviati in Amazon S3, considera sia la compatibilità dei dati sia la somiglianza dello schema. I fattori di compatibilità dei dati presi in considerazione includono il fatto che i dati abbiano o meno lo stesso formato (ad esempio JSON), lo stesso tipo di compressione (ad esempio GZIP), la stessa struttura del percorso Amazon S3 e altri attributi di dati. La somiglianza dello schema misura il livello di somiglianza degli schemi di oggetti Amazon S3 separati.

È possibile configurare un crawler in modo che esegua l'operazione `CombineCompatibleSchemas` per combinare schemi compatibili in una definizione di tabella comune quando possibile. Con questa opzione, il crawler continua a considerare la compatibilità dei dati, ma ignora la somiglianza degli schemi specifici durante la valutazione di oggetti Amazon S3 nel percorso di inclusione specificato.

Se stai configurando il crawler nella console, per combinare gli schemi seleziona l'opzione del crawler Create a single schema for each S3 path (Crea un singolo schema per ogni percorso S3).

Quando configuri il crawler tramite l'API, imposta l'opzione di configurazione seguente:

- Imposta il campo `Configuration` con una rappresentazione di stringa dell'oggetto JSON seguente nell'API del crawler, ad esempio:

```
{
   "Version": 1.0,
   "Grouping": {
      "TableGroupingPolicy": "CombineCompatibleSchemas" }
}
```

Per descrivere meglio questa opzione, supponiamo di definire un crawler con un percorso di inclusione `s3://bucket/table1/`. Quando il crawler viene eseguito, individua due file JSON con le caratteristiche seguenti:

- File 1: `S3://bucket/table1/year=2017/data1.json`
- Contenuto del file: `{“A”: 1, “B”: 2}`
- Schema: `A:int, B:int`

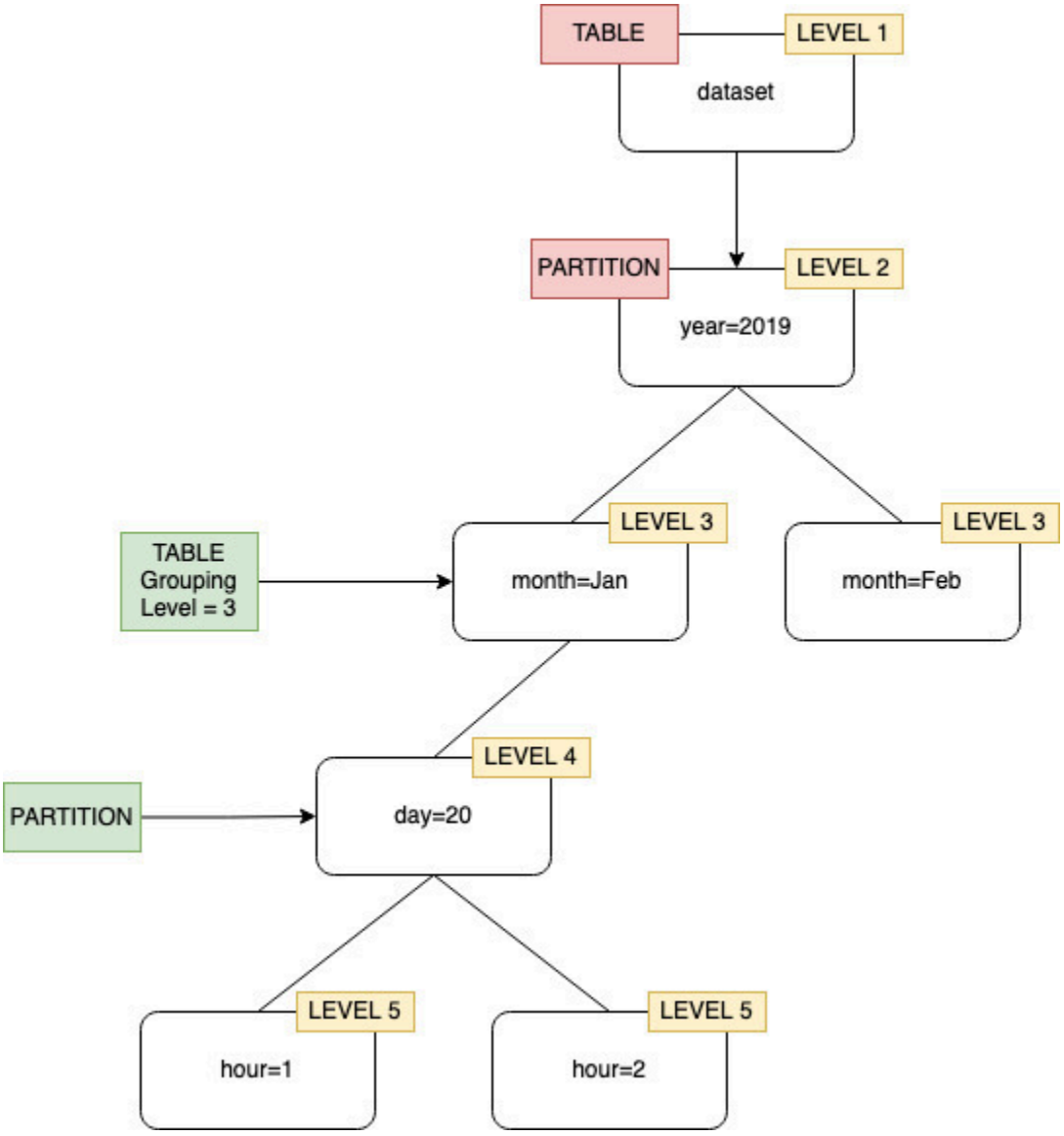
- File 2: `S3://bucket/table1/year=2018/data2.json`
- Contenuto del file – `{“C”: 3, “D”: 4}`
- Schema – `C: int, D: int`

Per impostazione predefinita, il crawler crea due tabelle, denominate `year_2017` e `year_2018`, perché gli schemi non sono abbastanza simili. Tuttavia, se l'opzione Create a single schema for each S3 path (Crea un singolo schema per ogni percorso S3) è selezionata e se i dati sono compatibili, il crawler crea una tabella. La tabella contiene lo schema `A:int,B:int,C:int,D:int` e `partitionKey year:string`.

Come specificare la posizione della tabella e il livello di partizionamento

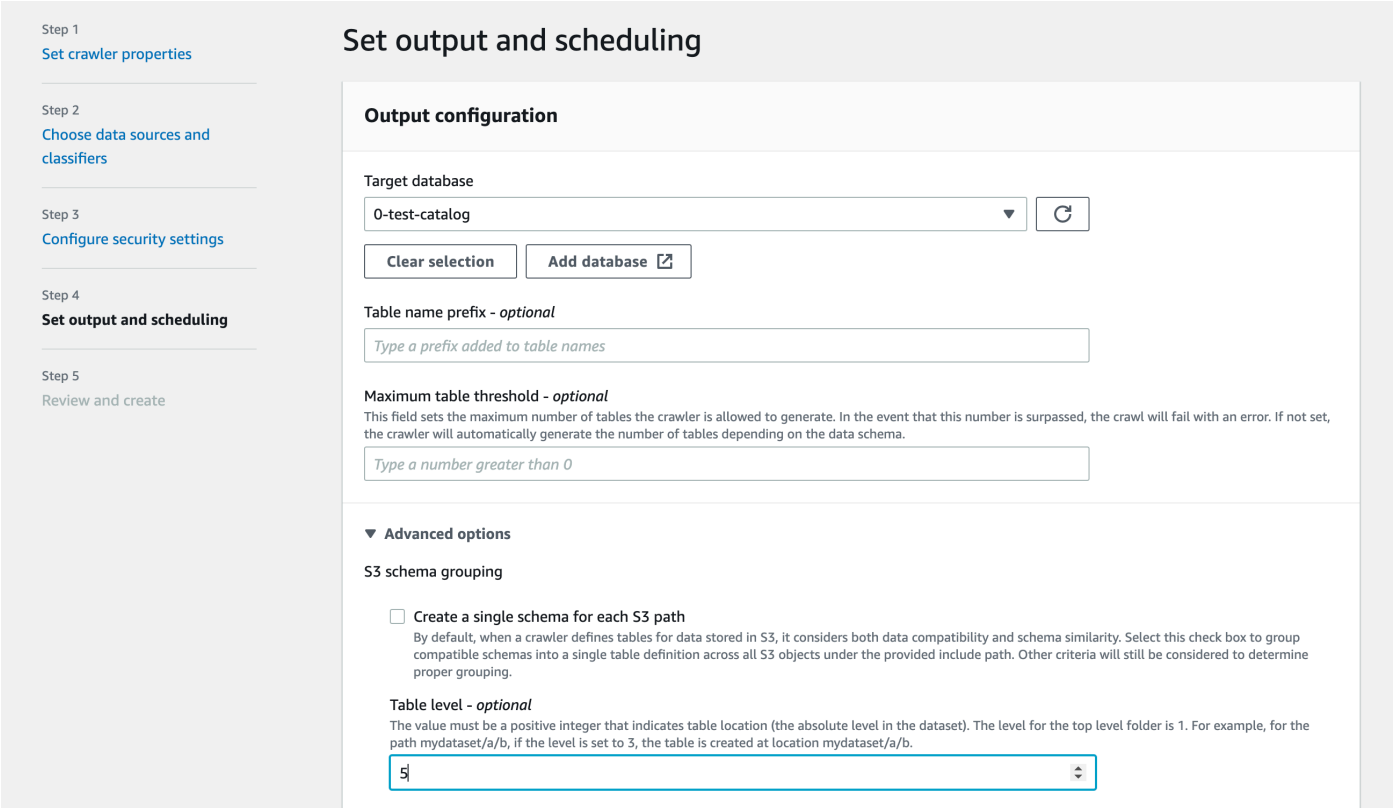
Per impostazione predefinita, quando un crawler definisce tabelle per i dati archiviati in Amazon S3, il crawler tenta di unire gli schemi e creare tabelle di primo livello (`year=2019`). In alcuni casi, è possibile che, invece di creare una tabella per la cartella `month=Jan` come previsto, il crawler crei una partizione poiché una cartella di pari livello (`month=Mar`) è stata unita alla stessa tabella.

L'opzione crawler a livello di tabella offre la flessibilità necessaria per indicare al crawler dove si trovano le tabelle e come si desidera creare le partizioni. Quando si specifica un Table level (Livello della tabella), la tabella viene creata a quel livello assoluto dal bucket Amazon S3.



Quando si configura il crawler nella console, è possibile specificare un valore per l'opzione crawler Table level (Livello della tabella). Il valore deve essere un numero intero positivo che indica la posizione della tabella (il livello assoluto nel set di dati). Il livello per la cartella di livello superiore è 1. Ad esempio, per il percorso `mydataset/year/month/day/hour`, se il livello è impostato su 3, la tabella viene creata nella posizione `mydataset/year/month`.

Console



API

Quando configuri il crawler usando l'API, imposta il campo `Configuration` con una rappresentazione stringa del seguente oggetto JSON; per esempio:

```
configuration = jsonencode(

{
   "Version": 1.0,
   "Grouping": {
            TableLevelConfiguration = 2
        }
})
```

CloudFormation

In questo esempio, imposti l'opzione Table level disponibile nella console all'interno del modello: CloudFormation

```
"Configuration": "{
```

```
    \"Version\":1.0,
    \"Grouping\":{\"TableLevelConfiguration\":2}
}"
```

Come specificare il numero massimo di tabelle che il crawler può creare

Facoltativamente, puoi stabilire il numero massimo di tabelle che il crawler può creare specificando un parametro `TableThreshold` tramite la console AWS Glue o CLI. Se il numero di tabelle rilevate dal crawler durante il crawling è superiore a questo valore di input, il crawling ha esito negativo e non vengono scritti dati nel Catalogo dati.

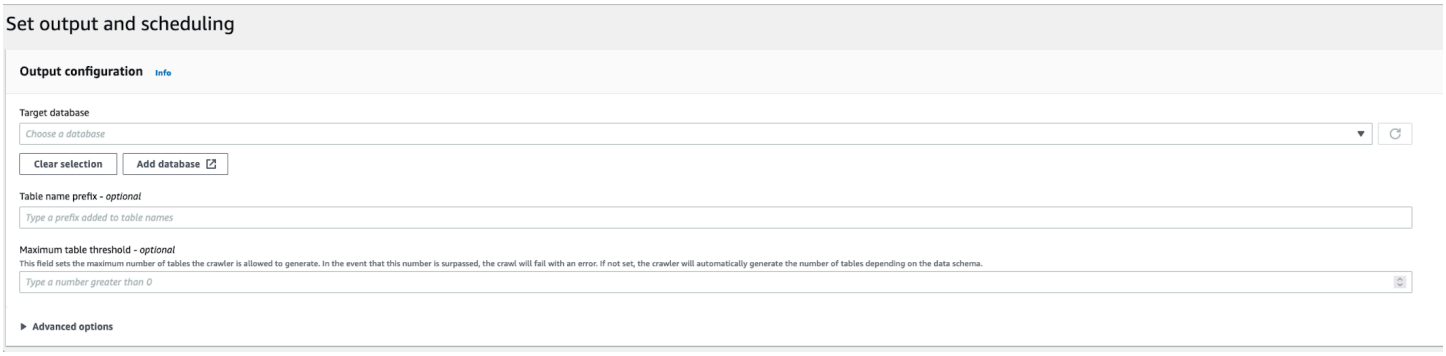
Questo parametro è utile quando le tabelle che verrebbero rilevate e create dal crawler sono molto più grandi del previsto. I motivi possono essere molteplici, come ad esempio:

- L'utilizzo di un processo AWS Glue per popolare le posizioni Amazon S3 potrebbe restituire file vuoti allo stesso livello di una cartella. In questo caso, quando esegui un crawler su questa posizione Amazon S3, il crawler crea più tabelle a causa di file e cartelle presenti allo stesso livello.
- La mancata configurazione di `"TableGroupingPolicy": "CombineCompatibleSchemas"` potrebbe restituire un numero maggiore di tabelle rispetto al previsto.

Specifica il parametro `TableThreshold` come un valore intero maggiore di 0. Questo valore è configurato in base al crawler, quindi viene preso in considerazione per ogni crawling. Si supponga ad esempio di avere un crawler con il valore `TableThreshold` impostato su 5. In ogni crawling, AWS Glue confronta il numero di tabelle rilevate con questo valore di soglia della tabella (5). Se il numero di tabelle rilevate è inferiore a 5, AWS Glue scrive le tabelle nel catalogo dati, in caso contrario il crawling ha esito negativo e non comporta alcuna scrittura nel catalogo dati.

Console

Per impostare `TableThreshold` mediante la console AWS:



CLI

Per impostare `TableThreshold` mediante la CLI AWS:

```
"{"Version":1.0,
"CrawlerOutput":
{"Tables":{"AddOrUpdateBehavior":"MergeNewColumns",
"TableThreshold":5}}}";
```

I messaggi di errore vengono registrati per facilitare l'identificazione dei percorsi delle tabelle e la pulizia dei dati. Di seguito è riportato un esempio di accesso all'account in caso di esito negativo del crawler a causa del numero di tabelle superiore al valore di soglia:

```
Table Threshold value = 28, Tables detected - 29
```

In CloudWatch, registriamo tutte le posizioni delle tabelle rilevate come messaggio INFO. Le informazioni relative all'errore vengono registrate in un messaggio.

```
ERROR com.amazonaws.services.glue.customerLogs.CustomerLogService - CustomerLogService
 received CustomerFacingException with message
The number of tables detected by crawler: 29 is greater than the table threshold value
 provided: 28. Failing crawler without writing to Data Catalog.
com.amazonaws.services.glue.exceptions.CustomerFacingInternalException: The number of
 tables detected by crawler: 29 is greater than the table threshold value provided:
 28.
Failing crawler without writing to Data Catalog.
```

Come specificare le opzioni di configurazione per un archivio dati Delta Lake

Quando configuri un crawler per un archivio dati Delta Lake, specifica i seguenti parametri di configurazione:

Connessione

Facoltativamente, seleziona o aggiungi una connessione di rete da utilizzare con questa destinazione Amazon S3. Per informazioni sulle connessioni, consulta Connessione ai dati.

Creazione di tabelle per l'interrogazione

Seleziona come desideri creare le tabelle Delta Lake:

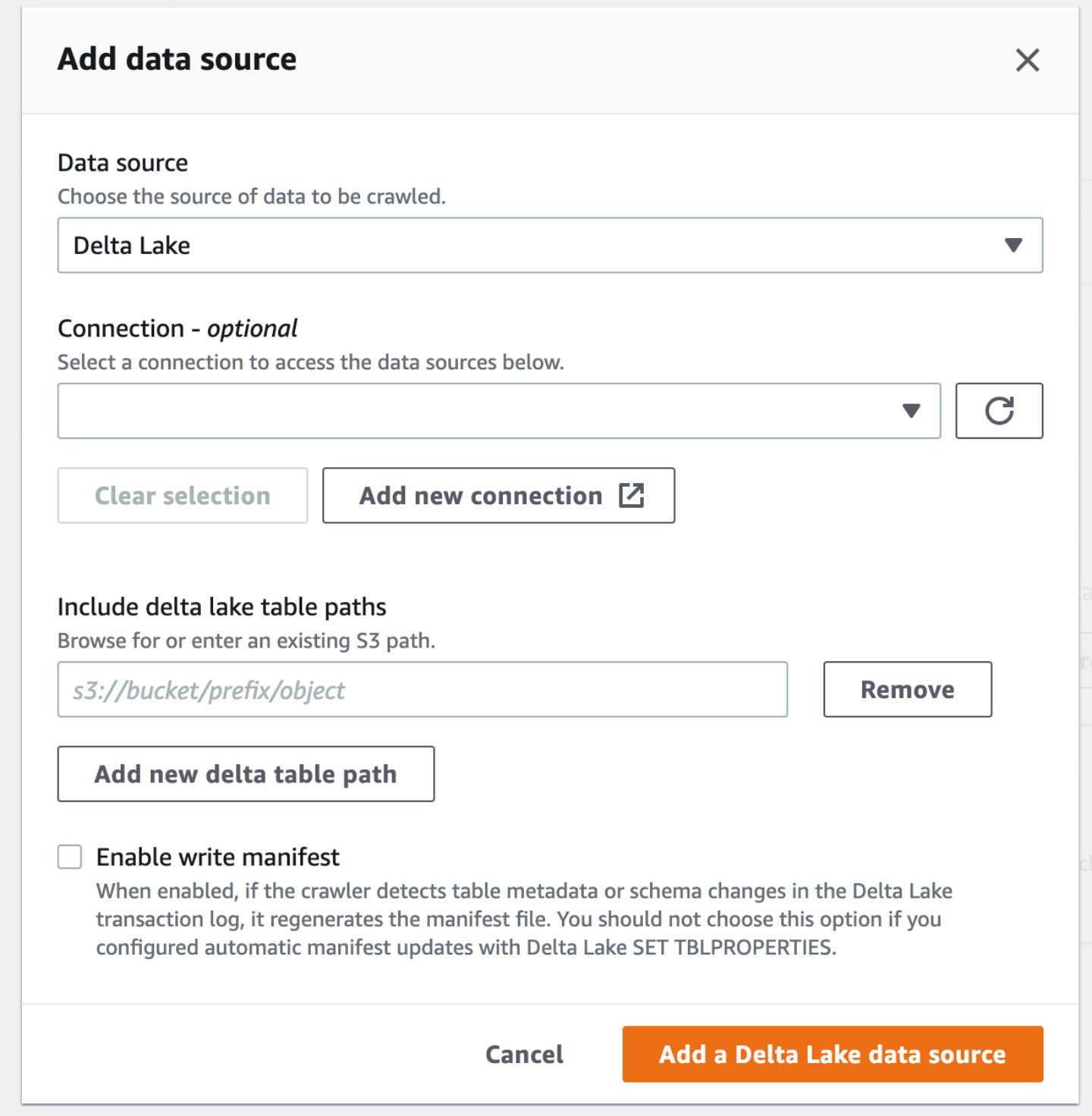
- Creazione di tabelle native: consente l'integrazione con i motori di query che supportano l'interrogazione diretta del log delle transazioni Delta.
- Creazione di tabelle Symlink: crea una cartella dei manifesti Symlink con file manifesti partizionati con chiavi di partizioni in base ai parametri di configurazione specificati.

Abilitazione del manifesto di scrittura (configurabile solo se hai selezionato Crea tabelle Symlink per una sorgente Delta Lake).

Scegli se rilevare i metadati della tabella o le modifiche dello schema nel log delle transazioni Delta Lake. Questa operazione rigenera il file manifesto. Non scegliere questa opzione se è stato configurato un aggiornamento automatico del manifesto con `SET TBLPROPERTIES` di Delta Lake.

Includere i percorsi delle tabelle Delta Lake

Specifica uno o più percorsi Amazon S3 per le tabelle Delta, come s3://*bucket*/*prefix*/*object*.



Come configurare un crawler per utilizzare le credenziali di Lake Formation

È possibile configurare un crawler in modo che utilizzi le credenziali AWS Lake Formation per accedere a un data store Amazon S3 o a una tabella del catalogo dati con una posizione Amazon S3 sottostante all'interno dello stesso Account AWS o di un altro Account AWS. È possibile configurare

una tabella del catalogo dati esistente come destinazione del crawler se entrambi si trovano nello stesso account. Attualmente, è consentita solo una singola destinazione del catalogo per una singola tabella quando si utilizza una tabella del catalogo dati come destinazione del crawler.

> **Note**
>
> Quando si definisce una tabella del catalogo dati come destinazione del crawler, assicurarsi che la posizione sottostante della tabella sia una posizione Amazon S3. I crawler che utilizzano le credenziali Lake Formation supportano solo le destinazioni del catalogo con le posizioni Amazon S3 sottostanti.

Configurazione richiesta quando il crawler e la posizione registrata di Amazon S3 o la tabella del catalogo dati si trovano nello stesso account (crawling all'interno dell'account)

Per consentire al crawler di accedere a un datastore o a una tabella del catalogo dati utilizzando le credenziali di Lake Formation, è necessario registrare la posizione dei dati con Lake Formation. Inoltre, il ruolo IAM del crawler deve disporre delle autorizzazioni necessarie per leggere i dati dalla destinazione in cui è registrato il bucket Amazon S3.

Puoi completare i passaggi di configurazione seguenti utilizzando la AWS Management Console o AWS Command Line Interface (AWS CLI).

AWS Management Console

1. Prima di configurare un crawler per accedere alla sua origine, registra la posizione dei dati del datastore o del catalogo dati con Lake Formation. Nella console di Lake Formation (https://console.aws.amazon.com/lakeformation/), registra una posizione Amazon S3 come posizione principale del data lake nell'Account AWS in cui è definito il crawler. Per ulteriori informazioni, consulta la pagina Registrazione di una posizione Amazon S3.
2. Concedi le autorizzazioni relative alla posizione dei dati al ruolo IAM utilizzato per l'esecuzione del crawler in modo che il crawler possa leggere i dati dalla destinazione in Lake Formation. Per ulteriori informazioni, consulta la pagina Concessione delle autorizzazioni per la posizione dei dati (stesso account).
3. Concessione al ruolo crawler delle autorizzazioni di accesso (`Create`) al database, che è specificato come database di output. Per ulteriori informazioni, consulta la pagina Concessione delle autorizzazioni al database tramite la console di Lake Formation e il metodo delle risorse denominate.

4. Nella console IAM (https://console.aws.amazon.com/iam/), crea un ruolo IAM per il crawler. Aggiungi la policy `lakeformation:GetDataAccess` al ruolo.
5. Nella console AWS Glue (https://console.aws.amazon.com/glue/), durante la configurazione del crawler, seleziona l'opzione Use Lake Formation credentials for crawling Amazon S3 data source (Usa le credenziali di Lake Formation per eseguire la ricerca per indicizzazione dell'origine dei dati Amazon S3).

> **Note**
>
> Il campo accountId è facoltativo per il crawling all'interno dell'account.

AWS CLI

```
aws glue --profile demo create-crawler --debug --cli-input-json '{
    "Name": "prod-test-crawler",
    "Role": "arn:aws:iam::111122223333:role/service-role/AWSGlueServiceRole-prod-
test-run-role",
    "DatabaseName": "prod-run-db",
    "Description": "",
    "Targets": {
    "S3Targets":[
                {
                 "Path": "s3://crawl-testbucket"
                }
                ]
                },
   "SchemaChangePolicy": {
      "UpdateBehavior": "LOG",
      "DeleteBehavior": "LOG"
  },
  "RecrawlPolicy": {
    "RecrawlBehavior": "CRAWL_EVERYTHING"
  },
  "LineageConfiguration": {
    "CrawlerLineageSettings": "DISABLE"
  },
  "LakeFormationConfiguration": {
    "UseLakeFormationCredentials": true,
    "AccountId": "111122223333"
  },
```

```
  "Configuration": {
           "Version": 1.0,
           "CrawlerOutput": {
             "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
             "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
           },
           "Grouping": { "TableGroupingPolicy": "CombineCompatibleSchemas" }
         },
  "CrawlerSecurityConfiguration": "",
  "Tags": {
    "KeyName": ""
  }
}'
```

Configurazione richiesta quando il crawler e la posizione registrata di Amazon S3 si trovano in account diversi (crawling tra più account)

Per consentire al crawler di accedere a un datastore in un account diverso utilizzando le credenziali di Lake Formation, è necessario prima registrare la posizione dei dati di Amazon S3 con Lake Formation. Quindi, concedi le autorizzazioni per la posizione dei dati all'account del crawler eseguendo la procedura seguente.

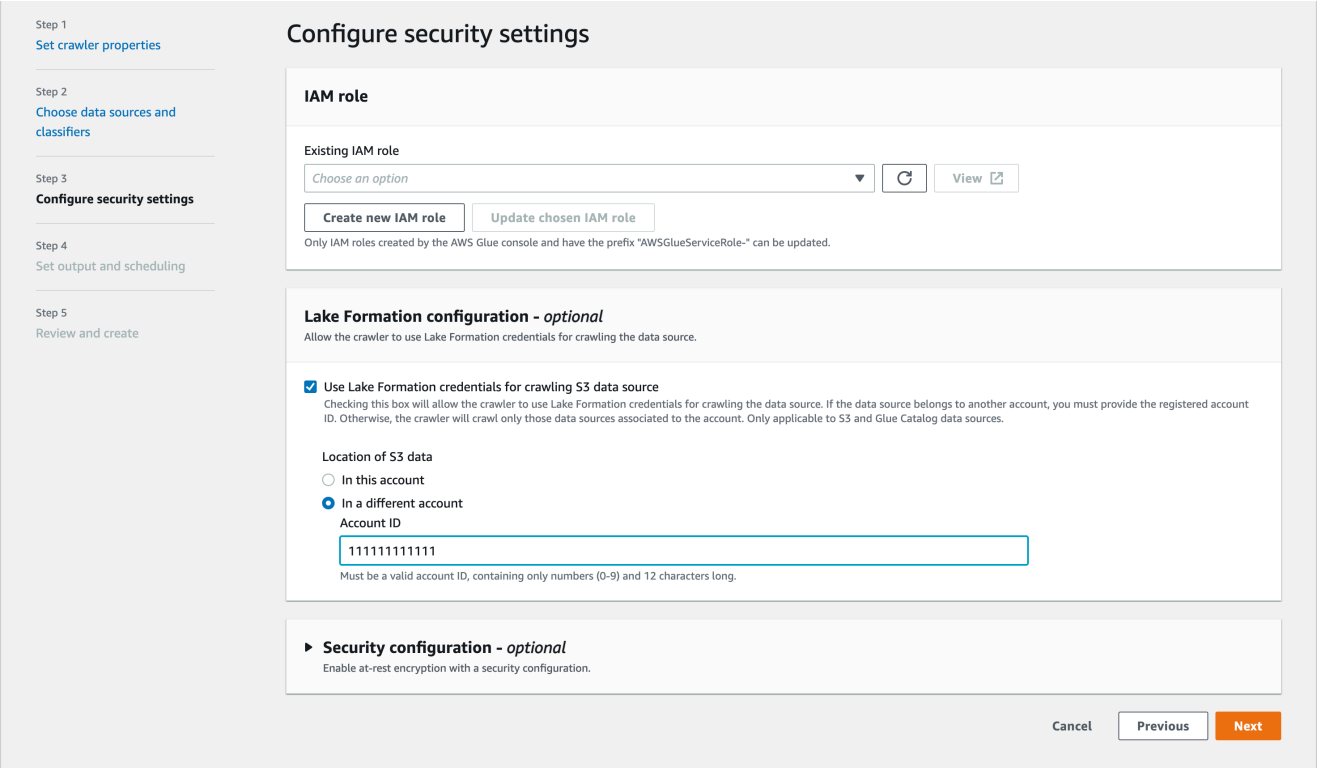
Completa i passaggi seguenti utilizzando la AWS Management Console o AWS CLI.

AWS Management Console

1. Nell'account in cui è registrata la posizione Amazon S3 (account B):
   a. Registra un percorso Amazon S3 con Lake Formation. Per ulteriori informazioni, consulta la pagina Registrazione della posizione Amazon S3.
   b. Concedi le autorizzazioni per la posizione dei dati all'account (A) in cui verrà eseguito il crawler. Per ulteriori informazioni, consulta la pagina Concessione delle autorizzazioni per la posizione dei dati.
   c. Crea un database vuoto in Lake Formation con la posizione sottostante come posizione Amazon S3 di destinazione. Per ulteriori informazioni, consulta la pagina Creazione di un database.
   d. Concedi l'accesso al database all'account A (l'account in cui verrà eseguito il crawler) creato nel passaggio precedente. Per ulteriori informazioni, consulta la pagina Concessione delle autorizzazioni al database.

2. Nell'account in cui è stato creato e verrà eseguito il crawler (account A):
   a. Tramite la console AWS RAM, accetta il database condiviso dall'account esterno (account B). Per ulteriori informazioni, consulta la pagina Accettare un invito alla condivisione di risorse da AWS Resource Access Manager.
   b. Crea un ruolo IAM per il crawler. Aggiungi la policy `lakeformation:GetDataAccess` al ruolo.
   c. Nella console di Lake Formation (https://console.aws.amazon.com/lakeformation/), concedi le autorizzazioni di Data Location (Posizione dei dati) nella posizione Amazon S3 di destinazione al ruolo IAM utilizzato per l'esecuzione del crawler, in modo che quest'ultimo possa leggere i dati dalla destinazione in Lake Formation. Per ulteriori informazioni, consulta la pagina Concessione delle autorizzazioni per la posizione dei dati.
   d. Crea un collegamento alla risorsa nel database condiviso. Per ulteriori informazioni, consulta la pagina Creare un collegamento alla risorsa.
   e. Concessione al ruolo crawler delle autorizzazioni di accesso (`Create`) sul database condiviso e (`Describe`) sul collegamento alla risorsa. Il collegamento alla risorsa è specificato nell'output del crawler.
   f. Nella console AWS Glue (https://console.aws.amazon.com/glue/), durante la configurazione del crawler, seleziona l'opzione Use Lake Formation credentials for crawling Amazon S3 data source (Usa le credenziali di Lake Formation per eseguire la ricerca per indicizzazione dell'origine dei dati Amazon S3).

      Per il crawling tra più account, specifica l'ID Account AWS nel quale la posizione Amazon S3 di destinazione è registrata con Lake Formation. Per il crawling all'interno dell'account, il campo accountId è facoltativo.



AWS CLI

```
aws glue --profile demo create-crawler --debug --cli-input-json '{
    "Name": "prod-test-crawler",
    "Role": "arn:aws:iam::111122223333:role/service-role/AWSGlueServiceRole-prod-
test-run-role",
    "DatabaseName": "prod-run-db",
    "Description": "",
    "Targets": {
    "S3Targets":[
                {
                 "Path": "s3://crawl-testbucket"
                }
                ]
                },
   "SchemaChangePolicy": {
      "UpdateBehavior": "LOG",
      "DeleteBehavior": "LOG"
  },
  "RecrawlPolicy": {
    "RecrawlBehavior": "CRAWL_EVERYTHING"
  },
```

```
  "LineageConfiguration": {
    "CrawlerLineageSettings": "DISABLE"
  },
  "LakeFormationConfiguration": {
    "UseLakeFormationCredentials": true,
    "AccountId": "111111111111"
  },
  "Configuration": {
           "Version": 1.0,
           "CrawlerOutput": {
             "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
             "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
           },
           "Grouping": { "TableGroupingPolicy": "CombineCompatibleSchemas" }
         },
  "CrawlerSecurityConfiguration": "",
  "Tags": {
    "KeyName": ""
  }
}'
```

Note

- Un crawler che utilizza le credenziali Lake Formation è supportato solo per le destinazioni Amazon S3 e Catalogo dati.
- Per le destinazioni che utilizzano la distribuzione delle credenziali Lake Formation, le posizioni Amazon S3 sottostanti devono appartenere allo stesso bucket. Ad esempio, gli utenti possono utilizzare più destinazioni (s3://bucket1/folder1, s3://bucket1/folder2) purché tutte le posizioni di destinazione si trovino nello stesso bucket (bucket1). Non è consentito specificare bucket diversi (s3://bucket1/folder1, s3://bucket2/folder2).
- Per i crawler di destinazione del catalogo dati è attualmente consentita solo una singola destinazione del catalogo per una singola tabella.

Pianificazione di un crawler in AWS Glue

Puoi eseguire un crawler AWS Glue on demand o in base a una pianificazione regolare. Le pianificazioni del crawler possono essere espresse in formato cron. Per ulteriori informazioni, consulta cron in Wikipedia.

Quando crei un crawler basato su una pianificazione, puoi specificare alcuni vincoli, come la frequenza, i giorni della settimana e l'orario di esecuzione del crawler. Questi vincoli si basano sul comando cron. Quando si configura la pianificazione di un crawler, devi tener conto delle caratteristiche e delle limitazioni del cron. Ad esempio, se vuoi eseguire il crawler il giorno 31 di ogni mese, devi ricordare che alcuni mesi non sono di 31 giorni.

I crawler per ogni crawler sono validi solo per un massimo di 12 mesi

Per ulteriori informazioni sull'utilizzo di cron per pianificare processi e crawler, consulta Pianificazioni basate sul tempo per processi e crawler.

Uso di crawler nella console AWS Glue

Un crawler accede al datastore, estrae i metadati e crea definizioni di tabella nel AWS Glue Data Catalog. Il riquadro Crawlers (Crawler) nella console AWS Glue elenca tutti i crawler che hai creato. L'elenco mostra stato e parametri dall'ultima esecuzione del crawler.

> **Note**
>
> Se scegli di importare le tue versioni dei driver JDBC, i crawler AWS Glue consumeranno risorse nei processi AWS Glue e nei bucket Amazon S3 per garantire che i driver forniti vengano eseguiti nel tuo ambiente. L'utilizzo aggiuntivo delle risorse si rifletterà nel tuo account. Inoltre, è importante sottolineare che anche se si fornisce il proprio driver JDBC, ciò non implica automaticamente che il crawler possa sfruttare tutte le funzionalità offerte da tale driver. I driver sono limitati alle proprietà descritte nella sezione Adding an AWS Glue connection.

Per aggiungere un crawler utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/. Nel riquadro di navigazione scegli Crawlers (Crawler).

2. Scegli Crea crawler e segui le istruzioni della procedura guidata Aggiungi crawler. La procedura guidata ti consentirà di completare i seguenti passaggi.

   1. Imposta le proprietà del crawler. Inserisci un nome per il crawler e una descrizione (facoltativa).

      Facoltativamente puoi applicare il tag al crawler tramite una chiave tag e un valore tag opzionale. Una volta create, le chiavi tag sono di sola lettura. Usa i tag su alcune risorse per facilitarne l'organizzazione e l'individuazione. Per ulteriori informazioni, consulta AWS tag in AWS Glue.

   2. Scegli le origini dati e i classificatori. In Configurazione dell'origine dati, scegli "Non ancora" o "Sì" per rispondere alla domanda "I tuoi dati sono mappati su tabelle AWS Glue?". Per impostazione predefinita, è selezionata la risposta "Non ancora".

      Se i tuoi dati sono già mappati su tabelle AWS Glue, scegli Aggiungi un'origine dati. Per ulteriori informazioni, consulta Aggiunta di una connessione AWS Glue.

      Nella finestra Aggiungi origine dati, scegli la tua origine dati e scegli le opzioni appropriate per l'origine dati.

      (Facoltativo) Se scegli JDBC come origine dati, puoi utilizzare i tuoi driver JDBC per specificare l'accesso alla connessione in cui sono archiviate le informazioni sul driver.

   3. Configurare le impostazioni di sicurezza. Scegli un ruolo IAM esistente o creane uno nuovo.

      > **Note**
      >
      > Per aggiungere il proprio driver JDBC, è necessario aggiungere ulteriori autorizzazioni. Per ulteriori informazioni, consulta la pagina
      >
      > - Concedi le autorizzazioni per le seguenti operazioni di processo: `CreateJob`, `DeleteJob`, `GetJob`, `GetJobRun`, `StartJobRun`.
      > - Concedi le autorizzazioni per le operazioni di Amazon S3: `s3:DeleteObjects`, `s3:GetObject`, `s3:ListBucket`, `s3:PutObject`.
      >
      > > **Note**
      > >
      > > Il valore `s3:ListBucket` non è necessario se la policy del bucket Amazon S3 è disabilitata.

- Concedi al principale del servizio l'accesso al bucket/cartella nella policy di Amazon S3.

Esempio di policy Amazon S3:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name/driver-parent-folder/
driver.jar",
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

AWS Glue crea le seguenti cartelle (`_crawler` e `_glue_job_crawler` allo stesso livello del driver JDBC) nel tuo bucket Amazon S3. Ad esempio, se il percorso del driver è `<s3-path/driver_folder/driver.jar>`, verranno create le seguenti cartelle, se non esistono ancora:

- <s3-path/driver_folder/_crawler>
- <s3-path/driver_folder/_glue_job_crawler>

Puoi facoltativamente aggiungere una configurazione di sicurezza a un crawler per specificare opzioni di crittografia dei dati inattivi.

4. Configura l'output e la pianificazione. È possibile scegliere il database di destinazione, aggiungere un prefisso da aggiungere ai nomi delle tabelle e impostare una soglia massima per la tabella (facoltativo).

   Quando selezioni una pianificazione del crawler, scegli la frequenza.

5. Revisione e creazione. Scegli Modifica per apportare modifiche a uno qualsiasi dei passaggi della procedura guidata. Al termine, scegli Crea crawler.

Quando esegui il crawling di tabelle DynamoDB, puoi scegliere un nome di tabella nell'elenco delle tabelle DynamoDB nel tuo account.

> **Tip**
>
> Per maggiori informazioni sulla configurazione dei crawler, consulta the section called "Proprietà del crawler".

Visualizzazione dei risultati e dei dettagli del crawler

Dopo l'esecuzione corretta del crawler, questo crea le definizioni di tabella nel catalogo dati. Scegli Tables (Tabelle) nel pannello di navigazione per visualizzare le tabelle create dal crawler nel database specificato.

È possibile visualizzare le informazioni relative al crawler stesso nel modo seguente:

- La pagina Crawler nella console AWS Glue mostra le seguenti proprietà per un crawler:

Proprietà	Description
Nome	Quando crei un crawler, devi assegnargli un nome univoco.
Stato	Un crawler può essere pronto, in fase avvio, in fase di arresto, pianificato o con pianifica zione in pausa. Un crawler in esecuzione avanza dall'avvio all'arresto. Puoi riprendere o sospendere una pianificazione collegata a un crawler.

Proprietà	Description
Pianificazione	Puoi scegliere di eseguire il tuo crawler on demand oppure scegliere una frequenza mediante una pianificazione. Per ulteriori informazioni sulla pianificazione di un crawler, consulta Pianificazione di un crawler.
Ultima esecuzione	Data e ora dell'ultima esecuzione del crawler.
Log	Link ai log disponibili dall'ultima esecuzione del crawler.
Modifiche alle tabelle rispetto all'ultima esecuzione	Numero di tabelle nel AWS Glue Data Catalog che sono state aggiornate dall'ultima esecuzione del crawler.

- Per visualizzare la cronologia di un crawler, scegli Crawlers (Crawler) nel pannello di navigazione per visualizzare i crawler creati. Scegli un crawler dall'elenco dei crawler disponibili. Puoi visualizzare le proprietà e la cronologia del crawler nella scheda Crawler runs (Esecuzioni del crawler).

  La scheda Crawler runs (Esecuzioni del crawler) mostra le informazioni relative a ogni esecuzione del crawler, tra cui Start time (UTC) (Ora di inizio [UTC]), End time (UTC) (Ora di fine [UTC]), Duration (Durata), Status (Stato), DPU hours (Ore DPU) e Table changes (Modifiche alla tabella).

  La scheda Esecuzioni del crawler riporta solo i crawling che si sono verificati dalla data di avvio della funzionalità di cronologia del crawler e conserva solo fino a 12 mesi di crawling. I crawling più vecchi non verranno restituiti.
- Per visualizzare le informazioni aggiuntive, scegli una scheda nella pagina dei dettagli del crawler. Ogni scheda mostrerà le informazioni relative al crawler.
  - Schedule (Pianificazione): tutte le pianificazioni create per il crawler saranno visibili qui.
  - Data sources (Origini dei dati): tutte le origini dei dati scansionate dal crawler saranno visibili qui.
  - Classifiers (Classificatori): tutti i classificatori assegnati al crawler saranno visibili qui.
  - Tag: ogni tag creato e assegnato a una risorsa AWS sarà visibile qui.

Accelerazione del crawling con le notifiche eventi Amazon S3

Invece di elencare gli oggetti da una destinazione Amazon S3 o catalogo dati, puoi configurare il crawler in modo che utilizzi gli eventi Amazon S3 per trovare eventuali modifiche. Questa caratteristica migliora il tempo di recupero utilizzando gli eventi Amazon S3 per identificare le modifiche tra due ricerche per indicizzazione elencando tutti i file della sottocartella che ha attivato l'evento invece che elencare l'intera destinazione Amazon S3 o catalogo dati.

Il primo crawling elenca tutti gli oggetti Amazon S3 dalla destinazione. Dopo il primo crawling riuscito, è possibile scegliere di effettuare una ricerca manualmente o in base a una pianificazione prestabilita. Il crawler elencherà solo gli oggetti di tali eventi invece di elencare tutti gli oggetti.

I vantaggi di passare a un crawler basato su eventi Amazon S3 sono:

- Non è necessario un nuovo crawling più rapido, poiché non è necessario l'elenco di tutti gli oggetti della destinazione, invece l'elenco di cartelle specifiche viene eseguito dove gli oggetti vengono aggiunti o eliminati.
- Si ha una riduzione del costo complessivo del crawling man mano che vengono elencate le cartelle specifiche nelle quali gli oggetti vengono aggiunti o eliminati.

Il crawling degli eventi Amazon S3 viene eseguito consumando gli eventi Amazon S3 dalla coda SQS in base alla pianificazione del crawler. Non ci saranno costi se non ci sono eventi nella coda. Gli eventi Amazon S3 possono essere configurati in modo che passino direttamente alla coda SQS o, nei casi in cui più utenti hanno bisogno dello stesso evento, verso una combinazione di SNS e SQS. Per ulteriori informazioni, consulta the section called “Impostazione dell’account per le notifiche eventi Amazon S3”.

Dopo aver creato e configurato il crawler in modalità evento, il primo crawling viene eseguito in modalità elenco eseguendo un elenco completo della destinazione Amazon S3 o catalogo dati. Il seguente log conferma il funzionamento del crawling consumando gli eventi Amazon S3 dopo la prima scansione riuscita: "il crawling è in esecuzione consumando eventi Amazon S3".

Dopo aver creato la ricerca per indicizzazione degli eventi Amazon S3 e aver aggiornato le proprietà del crawler che potrebbero influire sul crawling, quest'ultima funziona in modalità elenco e viene aggiunto il seguente log: "Il crawling non è in esecuzione in modalità evento S3".

Destinazione catalogo

Quando la destinazione è il catalogo dati, il crawler aggiorna le tabelle esistenti al suo interno con le modifiche (ad esempio, partizioni aggiuntive in una tabella).

Argomenti

- Impostazione dell'account per le notifiche eventi Amazon S3

Impostazione dell'account per le notifiche eventi Amazon S3

Questa sezione descrive come configurare il tuo account per le notifiche degli eventi Amazon S3 e fornisce istruzioni per farlo utilizzando uno script o la console AWS Glue.

Prerequisiti

Completa i seguenti processi di configurazione. Nota che i valori tra parentesi fanno riferimento alle impostazioni configurabili dello script.

1. Crea un bucket Amazon S3 (`s3_bucket_name`).
2. Identifica un crawler di destinazione (`folder_name`, come "test1") che sia un percorso nel bucket identificato.
3. Prepara un nome per il crawler (`crawler_name`)
4. Prepara un nome dell'argomento SNS (`sns_topic_name`), che potrebbe essere lo stesso del nome del crawler.
5. Prepara la Regione AWS in cui deve funzionare il crawler e in cui esista il bucket S3 (`region`).
6. Facoltativamente, se utilizzi l'e-mail per ricevere gli eventi Amazon S3 (`subscribing_email`), prepara un indirizzo e-mail.

Puoi anche utilizzare lo stack CloudFormation per creare le tue risorse. Completa questa procedura:

1. Avvia lo stack CloudFormation negli Stati Uniti orientali (Virginia settentrionale):
2. In Parametri, inserisci un nome per il bucket Amazon S3 (includendo il numero di account).
3. Seleziona `I acknowledge that AWS CloudFormation might create IAM resources with custom names`.
4. Scegli `Create stack`.

Restrizioni:

- Il crawler di destinazione ne supporta una sola, sia per quanto riguarda le destinazioni Amazon S3 che per le destinazioni Amazon S3.
- L'SQS su VPC privato non è supportato.
- Il campionamento Amazon S3 non è supportato.
- La destinazione del crawler deve essere una cartella per una destinazione Amazon S3 o una o più tabelle di catalogo dati di AWS Glue per una destinazione catalogo dati.
- Il carattere jolly del percorso “tutto” non è supportato: s3: //%
- Per una destinazione catalogo dati, tutte le tabelle del catalogo devono puntare allo stesso bucket Amazon S3 per la modalità evento di Amazon S3.
- Per una destinazione catalogo dati, una tabella di catalogo non deve indicare una posizione Amazon S3 nel formato Delta Lake (contenente cartelle _symlink o controllando le tabelle del catalogo `InputFormat`).

Per utilizzare il crawler basato su eventi Amazon S3, è necessario abilitare la notifica degli eventi sul bucket S3 con eventi filtrati dal prefisso uguale alla destinazione S3 e archiviarlo in SQS. È possibile configurare SQS e la notifica degli eventi tramite la console seguendo la procedura descritta in Spiegazione passo per passo: come configurare un bucket per le notifiche o usando il the section called “Script per generare SQS e configurare gli eventi Amazon S3 dalla destinazione”.

Policy SQS

Aggiungi la seguente policy SQS che è necessario allegare al ruolo utilizzato dal crawler.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sqs:DeleteMessage",
                "sqs:GetQueueUrl",
                "sqs:ListDeadLetterSourceQueues",
                "sqs:ReceiveMessage",
                "sqs:GetQueueAttributes",
                "sqs:ListQueueTags",
                "sqs:SetQueueAttributes",
```

```
                "sqs:PurgeQueue"
            ],
            "Resource": "arn:aws:sqs:{region}:{accountID}:cfn-sqs-queue"
        }
    ]
}
```

Script per generare SQS e configurare gli eventi Amazon S3 dalla destinazione

Dopo aver verificato che i prerequisiti siano soddisfatti, è possibile eseguire il seguente script Python per creare l'SQS. Sostituisci le impostazioni configurabili con i nomi preparati dai prerequisiti.

> **Note**
>
> Dopo aver eseguito lo script, accedi alla console SQS per trovare l'ARN dell'SQS creato.

Amazon SQS imposta un timeout visibilità, cioè un periodo di tempo durante il quale Amazon SQS impedisce ad altri consumatori di ricevere ed elaborare il messaggio. Imposta il timeout visibilità con valore approssimativamente uguale al tempo di esecuzione del crawling.

```
#!venv/bin/python
import boto3
import botocore

#---------Start : READ ME FIRST ----------------------#
# 1. Purpose of this script is to create the SQS, SNS and enable S3 bucket
 notification.
#     The following are the operations performed by the scripts:
#      a. Enable S3 bucket notification to trigger 's3:ObjectCreated:' and
 's3:ObjectRemoved:' events.
#      b. Create SNS topic for fan out.
#      c. Create SQS queue for saving events which will be consumed by the crawler.
#          SQS Event Queue ARN will be used to create the crawler after running the
 script.
# 2. This script does not create the crawler.
# 3. SNS topic is created to support FAN out of S3 events. If S3 event is also used by
 another
#    purpose, SNS topic created by the script can be used.
# 1. Creation of bucket is an optional step.
#    To create a bucket set create_bucket variable to true.
# 2. The purpose of crawler_name is to easily locate the SQS/SNS.
```

```
#     crawler_name is used to create SQS and SNS with the same name as crawler.
# 3. 'folder_name' is the target of crawl inside the specified bucket 's3_bucket_name'
#
#---------End : READ ME FIRST ------------------------#


#--------------------------------#
# Start : Configurable settings  #
#--------------------------------#

#Create
region = 'us-west-2'
s3_bucket_name = 's3eventtestuswest2'
folder_name = "test"
crawler_name = "test33S3Event"
sns_topic_name = crawler_name
sqs_queue_name = sns_topic_name
create_bucket = False

#--------------------------------#
# End : Configurable settings   #
#--------------------------------#

# Define aws clients
dev = boto3.session.Session(profile_name='myprofile')
boto3.setup_default_session(profile_name='myprofile')
s3 = boto3.resource('s3', region_name=region)
sns = boto3.client('sns', region_name=region)
sqs = boto3.client('sqs', region_name=region)
client = boto3.client("sts")
account_id = client.get_caller_identity()["Account"]
queue_arn = ""


def print_error(e):
    print(e.message + ' RequestId: ' + e.response['ResponseMetadata']['RequestId'])

def create_s3_bucket(bucket_name, client):
    bucket = client.Bucket(bucket_name)
    try:
        if not create_bucket:
            return True
        response = bucket.create(
            ACL='private',
```

```
            CreateBucketConfiguration={
                'LocationConstraint': region
            },
        )
        return True
    except botocore.exceptions.ClientError as e:
        print_error(e)
        if 'BucketAlreadyOwnedByYou' in e.message:  # we own this bucket so continue
            print('We own the bucket already. Lets continue...')
            return True
    return False

def create_s3_bucket_folder(bucket_name, client, directory_name):
    s3.put_object(Bucket=bucket_name, Key=(directory_name + '/'))

def set_s3_notification_sns(bucket_name, client, topic_arn):
    bucket_notification = client.BucketNotification(bucket_name)
    try:

        response = bucket_notification.put(
            NotificationConfiguration={
                'TopicConfigurations': [
                    {
                        'Id' : crawler_name,
                        'TopicArn': topic_arn,
                        'Events': [
                            's3:ObjectCreated:*',
                            's3:ObjectRemoved:*',

                        ],
                        'Filter' :  {'Key': {'FilterRules': [{'Name': 'prefix',
 'Value': folder_name}]}}
                    },
                ]
            }
        )
        return True
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return False


def create_sns_topic(topic_name, client):
    try:
```

```
        response = client.create_topic(
            Name=topic_name
        )
        return response['TopicArn']
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return None

def set_sns_topic_policy(topic_arn, client, bucket_name):
    try:
        response = client.set_topic_attributes(
            TopicArn=topic_arn,
            AttributeName='Policy',
            AttributeValue='''{
              "Version": "2008-10-17",
              "Id": "s3-publish-to-sns",
              "Statement": [{
                  "Effect": "Allow",
                  "Principal": { "AWS" : "*" },
                  "Action": [ "SNS:Publish" ],
                  "Resource": "%s",
                  "Condition": {
                    "StringEquals": {
                        "AWS:SourceAccount": "%s"
                      },
                      "ArnLike": {
                          "aws:SourceArn": "arn:aws:s3:*:*:%s"
                      }
                  }
              }]
  }''' % (topic_arn, account_id, bucket_name)
        )
        return True
    except botocore.exceptions.ClientError as e:
        print_error(e)

    return False


def subscribe_to_sns_topic(topic_arn, client, protocol, endpoint):
    try:
        response = client.subscribe(
            TopicArn=topic_arn,
            Protocol=protocol,
```

```
            Endpoint=endpoint
        )
        return response['SubscriptionArn']
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return None


def create_sqs_queue(queue_name, client):
    try:
        response = client.create_queue(
            QueueName=queue_name,
        )
        return response['QueueUrl']
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return None


def get_sqs_queue_arn(queue_url, client):
    try:
        response = client.get_queue_attributes(
            QueueUrl=queue_url,
            AttributeNames=[
                'QueueArn',
            ]
        )
        return response['Attributes']['QueueArn']
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return None

def set_sqs_policy(queue_url, queue_arn, client, topic_arn):
    try:
        response = client.set_queue_attributes(
            QueueUrl=queue_url,
            Attributes={
                'Policy': '''{
                  "Version": "2012-10-17",
                  "Id": "AllowSNSPublish",
                  "Statement": [
                    {
                      "Sid": "AllowSNSPublish01",
                      "Effect": "Allow",
```

```
                      "Principal": "*",
                      "Action": "SQS:SendMessage",
                      "Resource": "%s",
                      "Condition": {
                        "ArnEquals": {
                          "aws:SourceArn": "%s"
                        }
                      }
                    }
                  ]
}''' % (queue_arn, topic_arn)
            }
        )
        return True
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return False


if __name__ == "__main__":
    print('Creating S3 bucket %s.' % s3_bucket_name)
    if create_s3_bucket(s3_bucket_name, s3):
        print('\nCreating SNS topic %s.' % sns_topic_name)
        topic_arn = create_sns_topic(sns_topic_name, sns)
        if topic_arn:
            print('SNS topic created successfully: %s' % topic_arn)

            print('Creating SQS queue %s' % sqs_queue_name)
            queue_url = create_sqs_queue(sqs_queue_name, sqs)
            if queue_url is not None:
                print('Subscribing sqs queue with sns.')
                queue_arn = get_sqs_queue_arn(queue_url, sqs)
                if queue_arn is not None:
                    if set_sqs_policy(queue_url, queue_arn, sqs, topic_arn):
                        print('Successfully configured queue policy.')
                        subscription_arn = subscribe_to_sns_topic(topic_arn, sns,
 'sqs', queue_arn)
                        if subscription_arn is not None:
                            if 'pending confirmation' in subscription_arn:
                                print('Please confirm SNS subscription by visiting the
 subscribe URL.')
                            else:
                                print('Successfully subscribed SQS queue: ' +
 queue_arn)
```

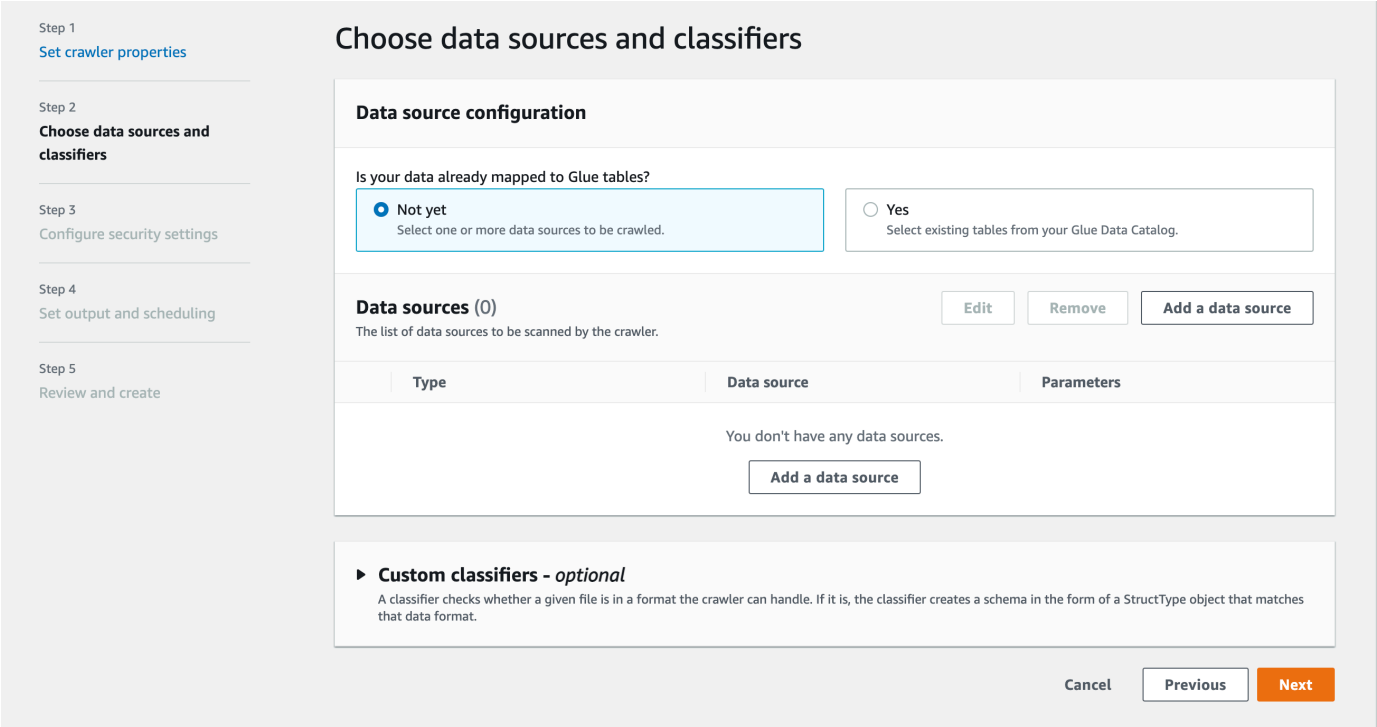
```
                    else:
                        print('Failed to subscribe SNS')
                else:
                    print('Failed to set queue policy.')
            else:
                print("Failed to get queue arn for %s" % queue_url)
        # ------------ End subscriptions to SNS topic -----------------

        print('\nSetting topic policy to allow s3 bucket %s to publish.' %
s3_bucket_name)
        if set_sns_topic_policy(topic_arn, sns, s3_bucket_name):
            print('SNS topic policy added successfully.')
            if set_s3_notification_sns(s3_bucket_name, s3, topic_arn):
                print('Successfully configured event for S3 bucket %s' %
s3_bucket_name)
                print('Create S3 Event Crawler using SQS ARN %s' % queue_arn)
            else:
                print('Failed to configure S3 bucket notification.')
        else:
            print('Failed to add SNS topic policy.')
    else:
        print('Failed to create SNS topic.')
```

Configurazione di un crawler per le notifiche degli eventi Amazon S3 tramite la console (destinazione Amazon S3)

Configurare un crawler per le notifiche eventi Amazon S3 utilizzando la console AWS Glue per una destinazione Amazon S3:

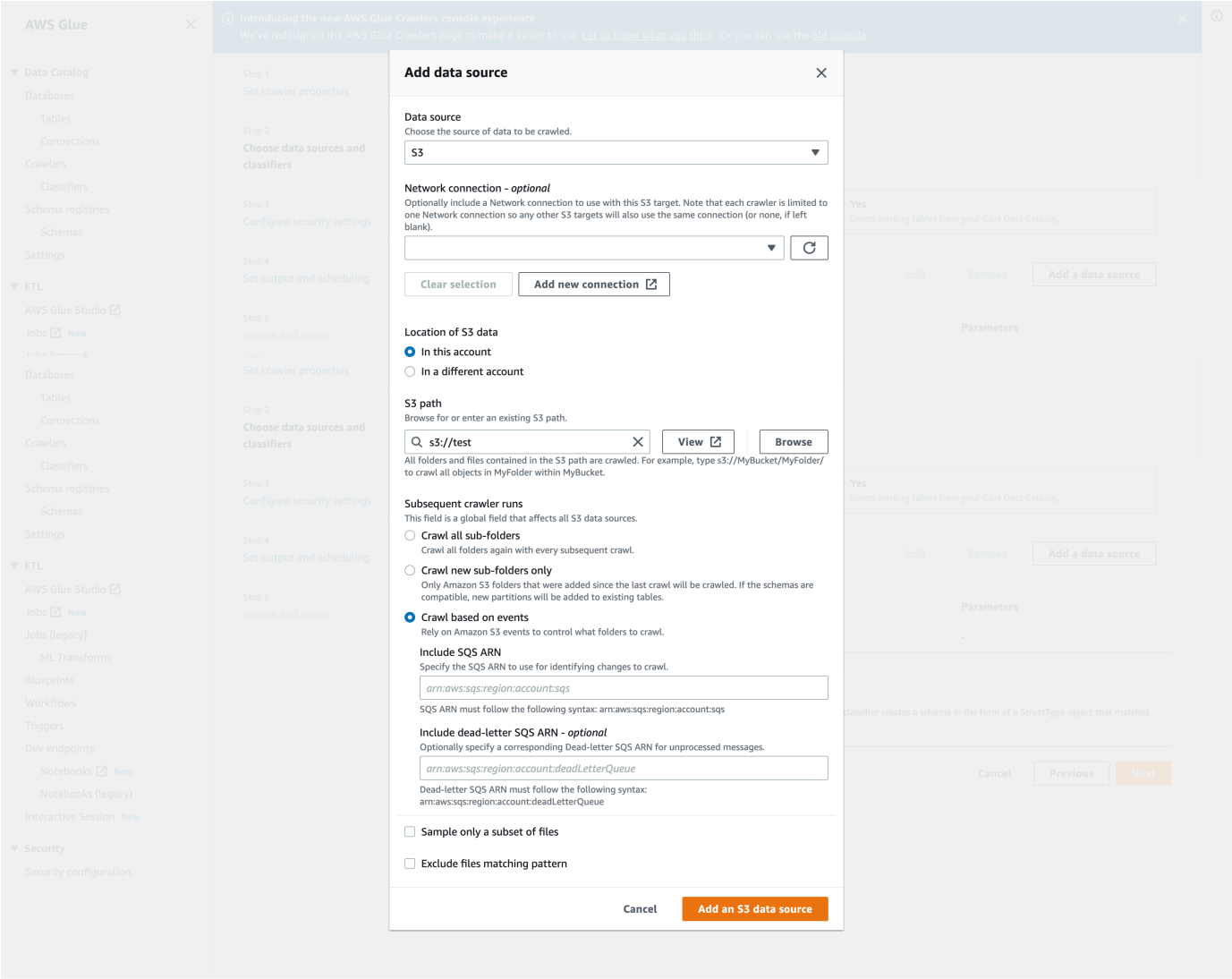
1. Imposta le proprietà del crawler. Per ulteriori informazioni, consulta la pagina Impostazione delle opzioni di configurazione del crawler nella console AWS Glue.
2. Nella sezione Data source configuration (Configurazione origine dei dati) viene chiesto Is your data already mapped to AWS Glue tables? .

   Per impostazione predefinita, la risposta Not yet (Non ancora) è già selezionata. Lascia questa impostazione come predefinita poiché stai utilizzando un'origine dei dati Amazon S3 e i dati non sono ancora mappati su tabelle AWS Glue.
3. Nella sezione Data sources (Origini dei dati), scegli Add a data source (Aggiungi un'origine dei dati).



4. Nella modalità Add data source (Aggiungi origine dei dati), configura l'origine dati di Amazon S3:

   - Data source (Origine dei dati): per impostazione predefinita, è selezionato Amazon S3.
   - Network connection (Connessione di rete) (Facoltativo): seleziona Add new connection (Aggiungi una nuova connessione).
   - Location of Amazon S3 data (Posizione dei dati Amazon S3): per impostazione predefinita, è selezionata l'opzione In this account (In questo account).
   - Amazon S3 path (Percorso Amazon S3): specifica il percorso Amazon S3 in cui effettuare il crawling in cartelle e file.
   - Subsequent crawler runs (Esecuzione successiva del crawler): seleziona Crawl based on events (Crawling in base agli eventi) per utilizzare le notifiche degli eventi di Amazon S3 per il crawler.
   - Include SQS ARN (Includi ARN SQS): specifica i parametri del datastore, incluso un ARN SQS valido. Ad esempio, `arn:aws:sqs:region:account:sqs`.
   - Include dead-letter SQS ARN (Includi ARN SQS non recapitabili): specifica un ARN SQS non recapitabile di Amazon valido. Ad esempio, `arn:aws:sqs:region:account:deadLetterQueue`.
   - Scegli Add an Amazon S3 data source (Aggiungi un'origine dei dati Amazon S3).



Configurazione di un crawler per le notifiche degli eventi Amazon S3 tramite la AWS CLI

Di seguito è riportato un esempio di chiamata della AWS CLI Amazon S3 per creare code SQS e configurare notifiche di eventi sul bucket di destinazione Amazon S3.

```
S3 Event AWS CLI
aws sqs create-queue --queue-name MyQueue --attributes file://create-queue.json
create-queue.json
```
```
{
    "Policy": {
```

```
        "Version": "2012-10-17",
        "Id": "example-ID",
        "Statement": [
            {
                "Sid": "example-statement-ID",
                "Effect": "Allow",
                "Principal": {
                    "Service": "s3.amazonaws.com"
                },
                "Action": [
                    "SQS:SendMessage"
                ],
                "Resource": "SQS-queue-ARN",
                "Condition": {
                    "ArnLike": {
                        "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                    },
                    "StringEquals": {
                        "aws:SourceAccount": "bucket-owner-account-id"
                    }
                }
            }
        ]
    }
}
```

```
aws s3api put-bucket-notification-configuration --bucket customer-data-pdx --
notification-configuration file://s3-event-config.json
s3-event-config.json
```

```
{
    "QueueConfigurations": [
        {
          "Id": "s3event-sqs-queue",
          "QueueArn": "arn:aws:sqs:{region}:{account}:queuename",
          "Events": [
                "s3:ObjectCreated:*",
                "s3:ObjectRemoved:*"
          ],
          "Filter": {
              "Key": {
                 "FilterRules": [
                     {
                         "Name": "Prefix",
```

```
                        "Value": "/json"
                    }
                ]
            }
        }
    }
]
}
```

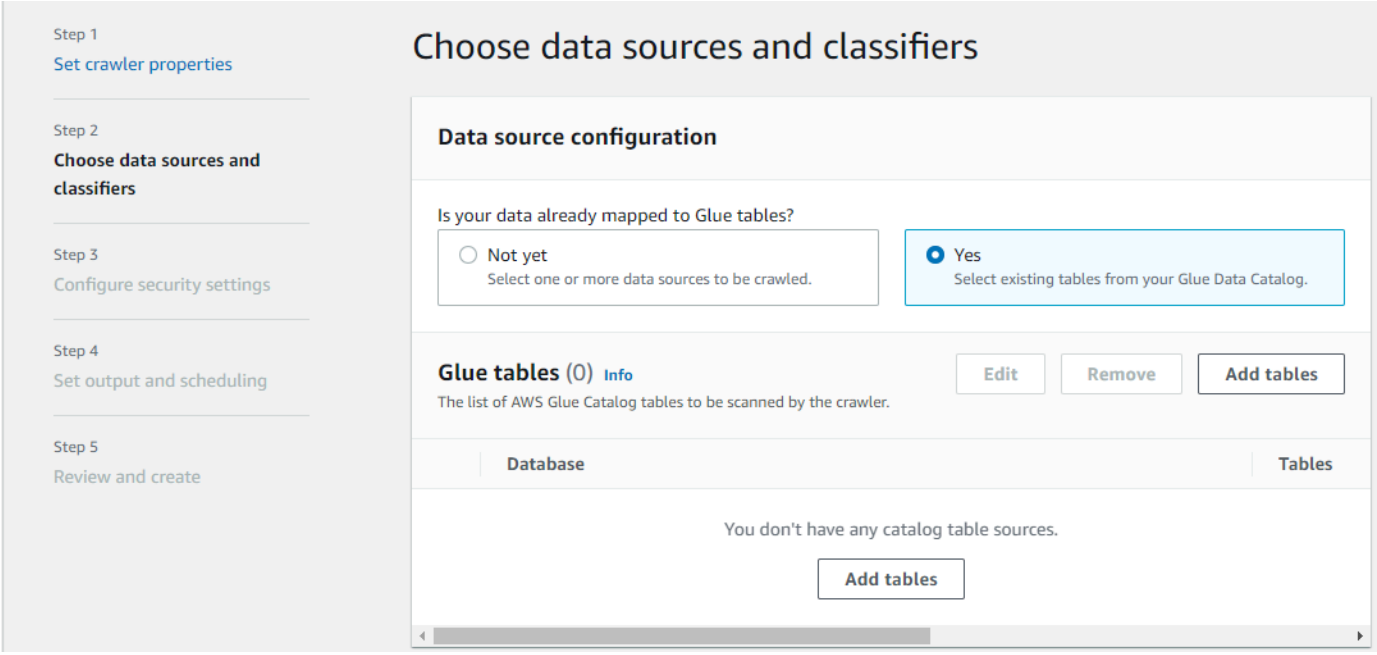
Create Crawler:

Configurazione di un crawler per le notifiche degli eventi Amazon S3 tramite la console (destinazione catalogo dati)

In presenza di una destinazione catalogo, configurare un crawler per le notifiche eventi Amazon S3 utilizzando la console AWS Glue:

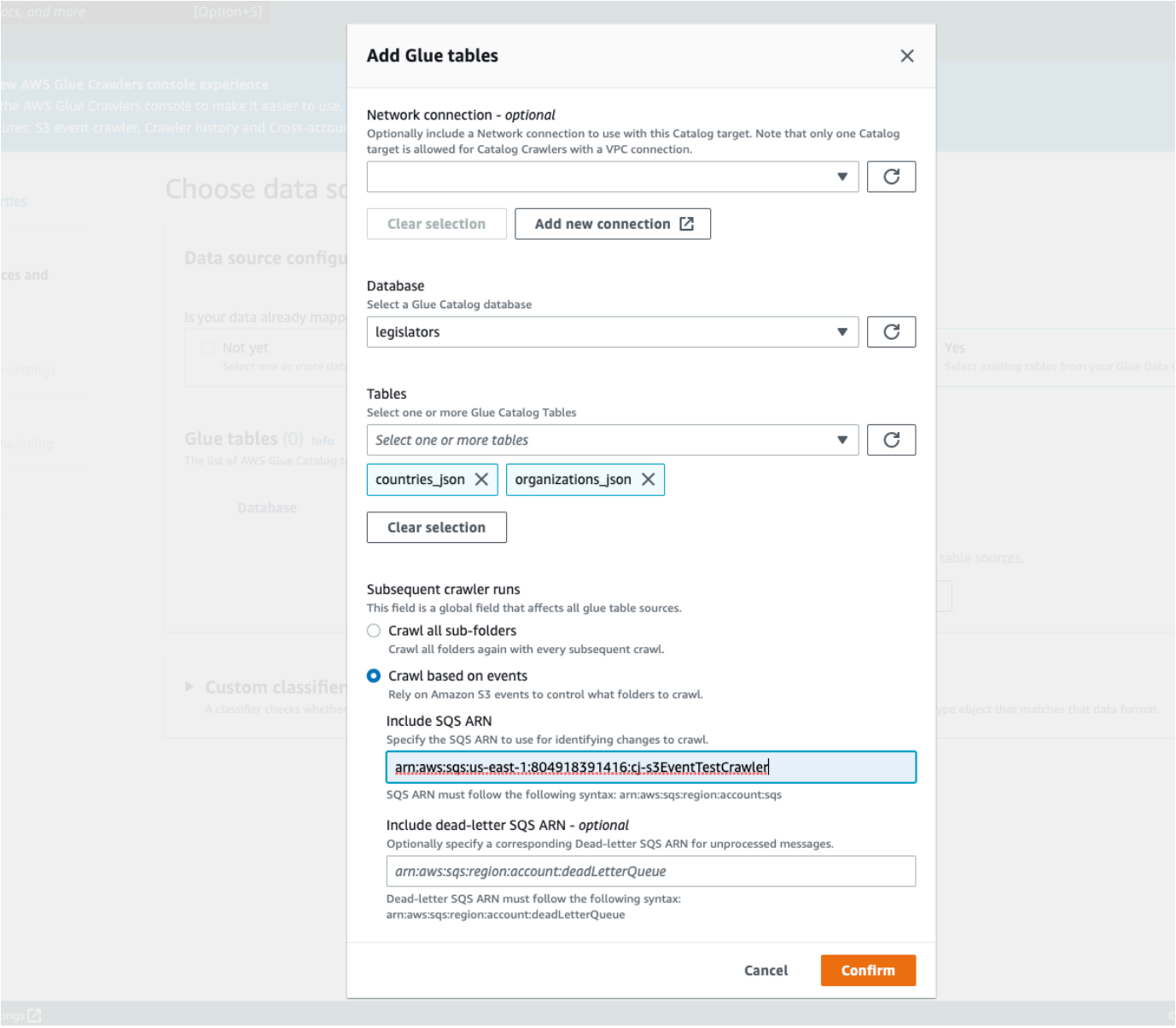
1. Imposta le proprietà del crawler. Per ulteriori informazioni, consulta la pagina Impostazione delle opzioni di configurazione del crawler nella console AWS Glue.
2. Nella sezione Data source configuration (Configurazione origine dei dati) viene chiesto Is your data already mapped to AWS Glue tables? .

   Seleziona Yes (Sì) per selezionare le tabelle esistenti dal catalogo dati come origine dati.
3. Nella sezione Glue tables, (tabelle Glue) scegli Add tables (aggiungi tabelle).



4. Nella modalità Add table (aggiungi tabella), configura il database e le tabelle:

   - Network connection (Connessione di rete) (Facoltativo): seleziona Add new connection (Aggiungi una nuova connessione).
   - Database: selezionare un database nel catalogo dati.
   - Tabelle: seleziona una o più tabelle da quel database nel catalogo dati.
   - Subsequent crawler runs (Esecuzione successiva del crawler): seleziona Crawl based on events (Crawling in base agli eventi) per utilizzare le notifiche degli eventi di Amazon S3 per il crawler.
   - Include SQS ARN (Includi ARN SQS): specifica i parametri del datastore, incluso un ARN SQS valido. Ad esempio, `arn:aws:sqs:region:account:sqs`.
   - Include dead-letter SQS ARN (Includi ARN SQS non recapitabili): specifica un ARN SQS non recapitabile di Amazon valido. Ad esempio, `arn:aws:sqs:region:account:deadLetterQueue`.
   - Scegli Conferma.



Utilizzo della crittografia con il crawler di eventi di Amazon S3

Questa sezione descrive l'utilizzo della crittografia solo su SQS o sia su SQS che su Amazon S3.

Argomenti

- Abilitazione della crittografia solo su SQS
- Abilitazione della crittografia sia su SQS che su Amazon S3
- Domande frequenti

Abilitazione della crittografia solo su SQS

Amazon SQS fornisce la crittografia in transito per impostazione predefinita. Per aggiungere la crittografia lato server (SSE) opzionale alla coda è possibile allegare una chiave principale del cliente (CMK) nel pannello di modifica. Ciò significa che SQS crittografa tutti i dati a riposo dei clienti sui server SQS.

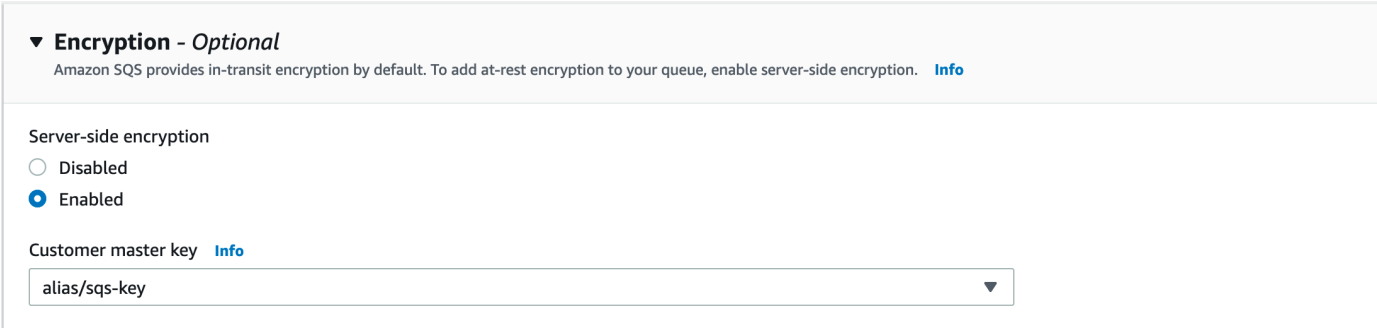
Creazione di una chiave principale del cliente (CMK)

1. Scegli Key Management Service (KMS) (Servizio di gestione delle chiavi (KMS)) > Customer Managed Keys (Chiavi gestite dal cliente) > Create key (Crea chiave).
2. Segui la procedura per aggiungere il tuo alias e la tua descrizione.
3. Aggiungi i rispettivi ruoli IAM che desideri utilizzino questa chiave.
4. Nella policy chiave, aggiungi un'altra dichiarazione all'elenco "Dichiarazione" in modo che la tua policy delle chiavi personalizzate fornisca ad Amazon SNS sufficienti autorizzazioni per l'utilizzo delle chiavi.

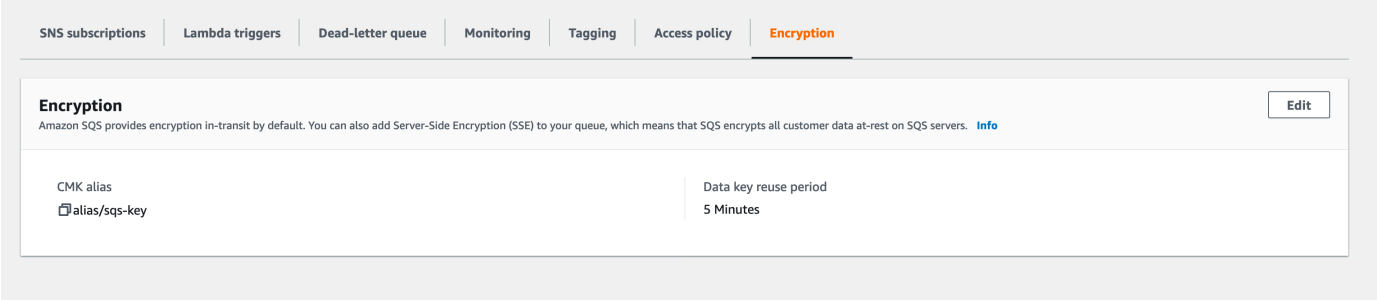
```
"Statement": [
       {
           "Effect": "Allow",
           "Principal": {
               "Service": "sns.amazonaws.com"
           },
           "Action": [
               "kms:GenerateDataKey",
               "kms:Decrypt"
           ],
           "Resource": "*"
       }
]
```

Abilitazione della crittografia lato server (SSE) nella coda

1. Scegli Amazon SQS > Queues (Code) > sqs_queue_name > scheda Encryption (Crittografia).
2. Scegli Edit (Modifica) e scorri verso il basso fino al menu a tendina Encryption (Crittografia).
3. Seleziona Enabled (Abilitato) per aggiungere SSE.
4. Seleziona la CMK creata in precedenza e non la chiave di default con il nome `alias/aws/sqs`.



Dopo averla aggiunta, la scheda Encryption (Crittografia) viene aggiornata con la nuova chiave.



> **Note**
>
> Amazon SQS elimina automaticamente i messaggi che sono stati in una coda per un periodo superiore quello massimo di conservazione. Il periodo predefinito per la conservazione dei messaggi è 4 giorni. Per evitare eventi mancanti, modifica l'SQS MessageRetentionPeriod fino al massimo di 14 giorni.

Abilitazione della crittografia sia su SQS che su Amazon S3

Abilitazione della crittografia lato server (SSE) su SQS

1. Seguire la procedura riportata in the section called “Abilitazione della crittografia solo su SQS”.
2. Nell'ultimo passaggio della configurazione della CMK, concedi ad Amazon S3 sufficienti autorizzazioni per l'utilizzo delle chiavi.

   Incolla quanto segue nell'elenco "Dichiarazione":

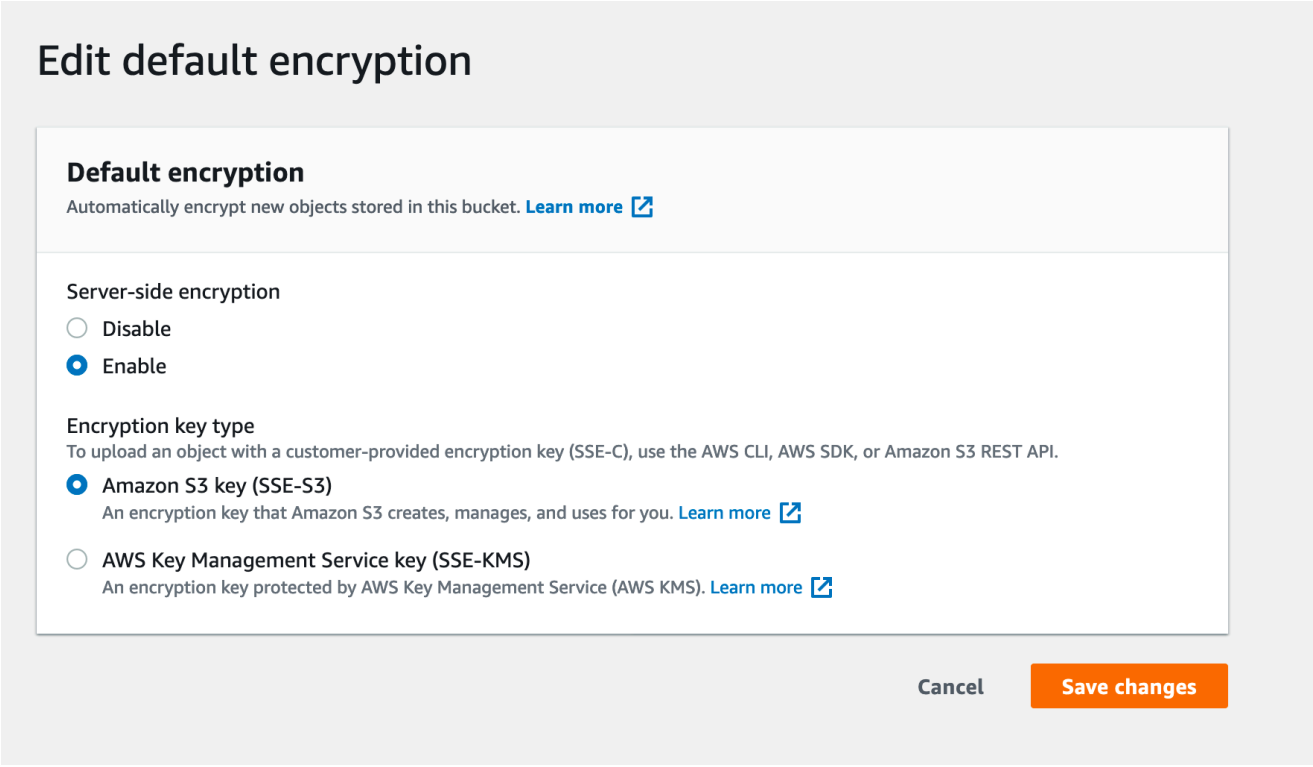
```
"Statement": [
      {
          "Effect": "Allow",
```

```
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
]
```

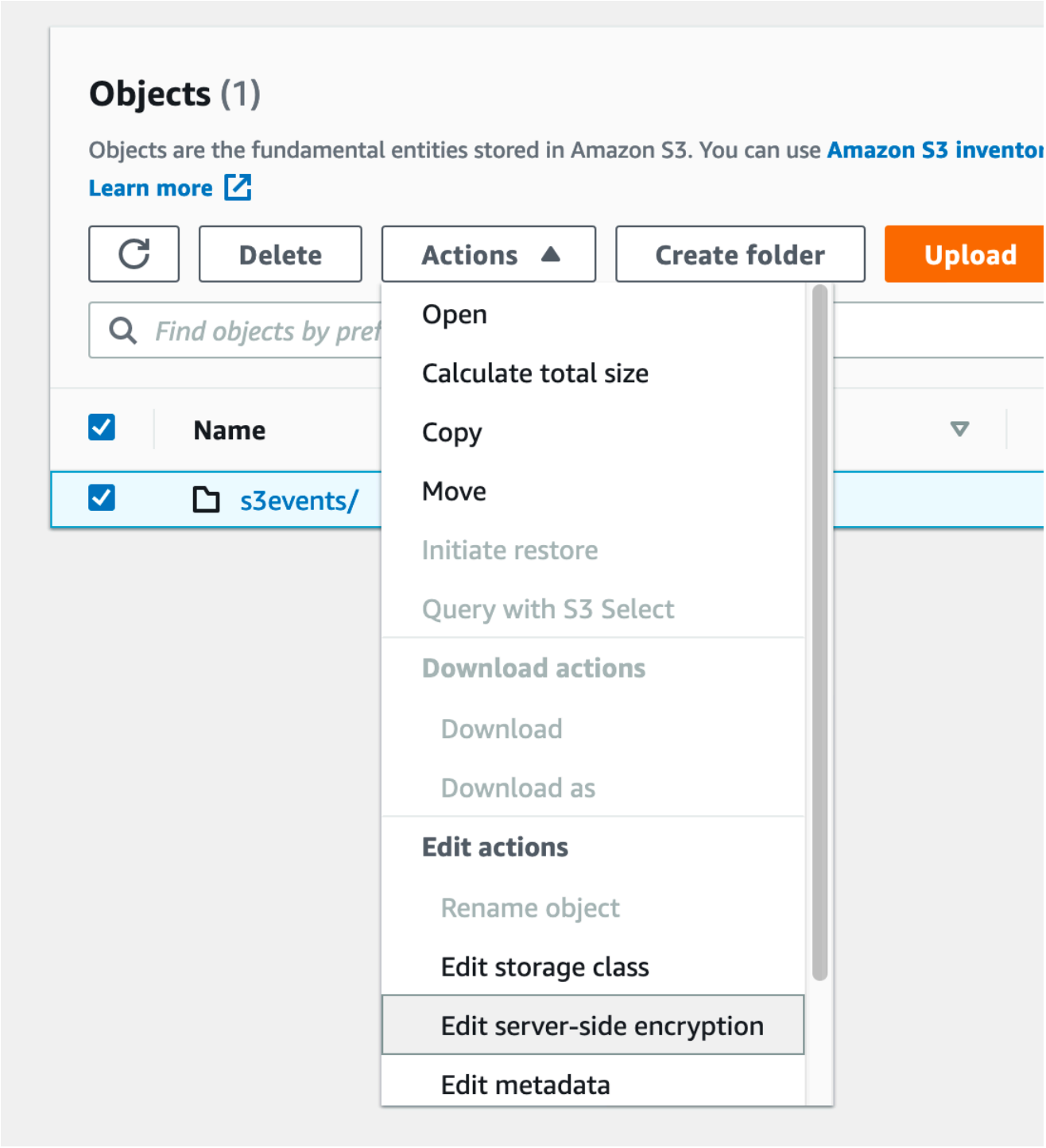
Abilitazione della crittografia lato server (SSE) sul bucket Amazon S3

1. Seguire la procedura riportata in the section called “Abilitazione della crittografia solo su SQS”.
2. Completa una delle seguenti operazioni:
   - Per abilitare SSE per l'intero bucket S3, accedi alla scheda Properties (Proprietà) nel bucket di destinazione.

     Qui puoi abilitare SSE e scegliere il tipo di crittografia che desideri utilizzare. Amazon S3 fornisce una chiave di crittografia che Amazon S3 crea, gestisce e utilizza per te, in alternativa puoi scegliere una chiave da KMS.



- Per abilitare SSE su una cartella specifica, fai clic sulla casella di controllo accanto alla cartella di destinazione e scegli Edit server-side encryption (Modifica della crittografia lato server) sotto il menu a tendina Actions (Operazioni).



Domande frequenti

Perché i messaggi pubblicati sul mio argomento Amazon SNS non vengono consegnati alla coda Amazon SQS sottoscritta con crittografia lato server (SSE) abilitata?

Verifica che la tua coda Amazon SQS stia utilizzando:

1. Una chiave master del cliente (CMK) che sia gestita dal cliente. Non quella di default fornita da SQS.
2. Il tuo CMK da (1) include una policy delle chiavi personalizzate che fornisce ad Amazon SNS sufficienti autorizzazioni per l'utilizzo delle chiavi.

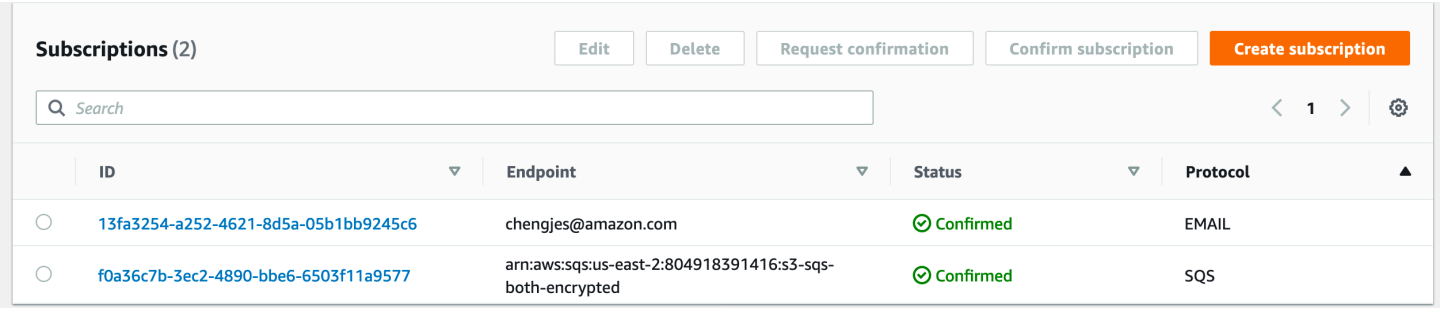
Per ulteriori informazioni, consulta il seguente articolo nel Portale del sapere.

Mi sono iscritto alle notifiche via e-mail, ma non ricevo aggiornamenti via e-mail quando modifico il mio bucket Amazon S3.

Assicurati di aver confermato il tuo indirizzo e-mail facendo clic sul link "Conferma abbonamento" nella tua email. È possibile verificare lo stato della conferma controllando la tabella Subscriptions (Abbonamenti) sotto il tuo argomento SNS.

Scegli Amazon SNS > Topics (Argomenti) > `sns_topic_name` > Subscriptions table (Tabella degli abbonamenti).

Se hai seguito il nostro script prerequisito, scoprirai che `sns_topic_name` è uguale al tuo `sqs_queue_name`. La schermata visualizzata dovrebbe risultare simile a quella nell'immagine seguente:



Solo alcune delle cartelle che ho aggiunto vengono visualizzate nella tabella dopo aver abilitato la crittografia lato server nella coda SQS. Perché mi mancano alcuni parquet?

Se le modifiche del bucket Amazon S3 sono state apportate prima di abilitare SSE nella coda SQS, potrebbero non essere rilevate dal crawler. Per assicurarti di aver eseguito il crawling di tutti gli aggiornamenti del bucket S3, esegui di nuovo il crawler in modalità elenco ("Crawl All Folders", Ricerca per indicizzazione di tutte le cartelle). Un'altra opzione è quella di iniziare di nuovo creando un nuovo crawler con eventi S3 abilitati.

Parametri impostati sulle tabelle del catalogo dati dal crawler

Queste proprietà delle tabelle sono impostate dai crawler AWS Glue. Ci aspettiamo che gli utenti utilizzino `compressionType` e le proprietà `classification`. Altre proprietà, comprese le stime delle dimensioni delle tabelle, vengono utilizzate per i calcoli interni e non garantiamo la loro accuratezza o applicabilità ai casi d'uso dei clienti. La modifica di questi parametri può alterare il comportamento del crawler, non supportiamo questo flusso di lavoro.

Chiave di proprietà	Valore proprietà
`UPDATED_BY_CRAWLER`	Nome del crawler che esegue l'aggiornamento.
`connectionName`	Il nome della connessione nel catalogo di dati per il crawler utilizzato per connettersi all'archivio dati.
`recordCount`	Stima del numero di registri nella tabella, in base alle dimensioni e alle intestazioni dei file.
`skip.header.line.count`	Righe saltate per saltare l'intestazione. Impostato su tabelle classificate come CSV.
`CrawlerSchemaSerializerVersion`	Per uso interno
`classification`	Formato dei dati, dedotto dal crawler. Per ulteriori informazioni sui formati supportati dai crawler AWS Glue, consulta the section called “Classificatori predefiniti in AWS Glue”.
`CrawlerSchemaDeserializerVersion`	Per uso interno
`sizeKey`	Dimensione combinata dei file nella tabella sottoposti a scansione.
`averageRecordSize`	La dimensione media della riga nella tabella in byte.

Chiave di proprietà	Valore proprietà
`compressionType`	Tipo di compressione utilizzato sui dati della tabella. Per ulteriori informazioni sui formati supportati dai crawler AWS Glue, consulta the section called “Classificatori predefiniti in AWS Glue”.
`typeOfData`	`file`, `table`, oppure `view`.
`objectCount`	Numero di oggetti nel percorso Amazon S3 per la tabella.

Queste proprietà aggiuntive della tabella vengono impostate dai crawler AWS Glue degli archivi dati Snowflake.

Chiave di proprietà	Valore proprietà
`aws:RawTableLastAltered`	Registra l'ultimo timestamp modificato della tabella Snowflake.
`ViewOriginalText`	Visualizza l'istruzione SQL.
`ViewExpandedText`	Visualizza l'istruzione SQL codificata nel formato Base64.
`ExternalTable:S3Location`	La posizione di Amazon S3 della tabella esterna Snowflake.
`ExternalTable:FileFormat`	Formato di file Amazon S3 della tabella esterna Snowflake.

Queste proprietà aggiuntive delle tabelle vengono impostate dai crawler AWS Glue per archivi di dati di tipo JDBC come Amazon Redshift, Microsoft SQL Server, MySQL, PostgreSQL e Oracle.

Chiave di proprietà	Valore proprietà
`aws:RawType`	Quando un crawler archivia i dati nel catalogo dati, traduce i tipi di dati in tipi compatibili con Hive, il che spesso causa la perdita delle informazioni sul tipo di dati nativo. Il crawler emette il parametro `aws:RawType` per fornire il tipo di dati a livello nativo.
`aws:RawColumnComment`	Se un commento è associato a una colonna del database, il crawler emette il commento corrispondente nella tabella del catalogo. La stringa di commento viene troncata a 255 byte. I commenti non sono supportati per Microsoft SQL Server.
`aws:RawTableComment`	Se un commento è associato a una tabella nel database, il crawler emette il commento corrispondente nella tabella del catalogo. La stringa di commento viene troncata a 255 byte. I commenti non sono supportati per Microsoft SQL Server.

Aggiunta di classificatori a un crawler in AWS Glue

Un classificatore legge i dati in un archivio dati. Se riconosce il formato dei dati, genera uno schema. Il classificatore inoltre restituisce un numero di certezza per indicare quanto è stato certo il riconoscimento del formato.

AWS Glue fornisce un set di classificatori predefiniti, ma puoi anche crearne di personalizzati. AWS Glue richiama per primi i classificatori personalizzati, in base all'ordine che hai specificato nella definizione del crawler. A seconda dei risultati restituiti dai classificatori personalizzati, AWS Glue può anche richiamare i classificatori predefiniti. Se un classificatore restituisce `certainty=1.0` durante l'elaborazione, significa che la certezza di creare uno schema corretto è del 100%. AWS Glue usa quindi l'oputput di questo classificatore.

Se nessun classificatore restituisce `certainty=1.0`, AWS Glue usa l'output del classificatore che indica la certezza maggiore. Se nessun classificatore restituisce una certezza superiore a `0.0`, AWS Glue restituisce la stringa di classificazione predefinita `UNKNOWN`.

Quando si utilizza un classificatore?

Puoi usare classificatori quando esegui il crawling di un datastore per definire tabelle di metadati nel AWS Glue Data Catalog. È possibile configurare il crawler con un set ordinato di classificatori. Quando il crawler richiama un classificatore, il classificatore determina se i dati vengono riconosciuti. Se il classificatore non è in grado di riconoscere i dati o non è certo al 100 percento, il crawler richiama il prossimo classificatore nell'elenco per determinare se è in grado di riconoscere i dati.

Per ulteriori informazioni sulla creazione di un classificatore tramite la console AWS Glue, consulta Uso di classificatori nella console AWS Glue.

Classificatori personalizzati

L'output di un classificatore include una stringa che indica la classificazione del file o il formato (ad esempio, `json`) e lo schema del file. Per classificatori personalizzati, è possibile definire la logica per la creazione di uno schema in base al tipo di classificatore. I tipi di classificatore includono definizioni di schemi in base a pattern grok, tag XML e percorsi JSON.

Se si modifica una definizione di classificatore, qualsiasi tipo di dati precedentemente sottoposto a crawling utilizzando il classificatore non è riclassificato. Un crawler tiene traccia dei dati precedentemente sottoposti a crawling. I nuovi dati sono classificati con il classificatore aggiornato, che potrebbe dare origine a uno schema aggiornato. Se lo schema dei dati si è evoluto, aggiornare il classificatore per verificare qualsiasi modifica dello schema quando viene eseguito il crawler. Per riclassificare i dati al fine di correggere un classificatore errato, creare un nuovo crawler con il classificatore aggiornato.

Per ulteriori informazioni sulla creazione di classificatori personalizzati in AWS Glue, consulta Scrittura di classificatori personalizzati.

> **Note**
>
> Se il formato di dati è riconosciuto da uno dei classificatori integrati, non è necessario creare un classificatore personalizzato.

Classificatori predefiniti in AWS Glue

AWS Glue fornisce classificatori predefiniti per diversi formati, tra cui JSON, CSV, log Web e molti sistemi di database.

Se AWS Glue non trova un classificatore personalizzato appropriato al formato di dati di input con una certezza del 100%, richiama i classificatori predefiniti in base all'ordine mostrato nella tabella seguente. Il classificatore integrato restituisce un risultato per indicare se il formato corrisponde (`certainty=1.0`) o non, corrisponde (`certainty=0.0`). Il primo classificatore con `certainty=1.0` fornisce la stringa di classificazione e lo schema per una tabella di metadati nel catalogo dati.

Tipo di classificatore	Stringa di classificazione	Note
Apache Avro	`avro`	Legge lo schema nella parte iniziale del file per determinare il formato.
Apache ORC	`orc`	Legge i metadati dei file per determinare il formato.
Apache Parquet	`parquet`	Legge lo schema nella parte finale del file per determinare il formato.
JSON	`json`	Legge la parte iniziale del file per determinare il formato.
Binary JSON	`bson`	Legge la parte iniziale del file per determinare il formato.
XML	`xml`	Legge la parte iniziale del file per determinare il formato. AWS Glue determina lo schema della tabella in base ai tag XML nel documento. Per ulteriori informazioni su come creare un classificatore XML personalizzato per specificare le righe nel documento, consulta Scrittura di classificatori personalizzati XML.
Amazon Ion	`ion`	Legge la parte iniziale del file per determinare il formato.
Combined Apache log	`combined_apache`	Determina i formati dei log attraverso un pattern grok.

Tipo di classificatore	Stringa di classificazione	Note
Apache log	`apache`	Determina i formati dei log attraverso un pattern grok.
Linux kernel log	`linux_kernel`	Determina i formati dei log attraverso un pattern grok.
Microsoft log	`microsoft_log`	Determina i formati dei log attraverso un pattern grok.
Ruby log	`ruby_logger`	Legge la parte iniziale del file per determinare il formato.
Squid 3.x log	`squid`	Legge la parte iniziale del file per determinare il formato.
Redis monitor log	`redismonlog`	Legge la parte iniziale del file per determinare il formato.
Redis log	`redislog`	Legge la parte iniziale del file per determinare il formato.
CSV	`csv`	Verifica dei seguenti delimitatori: virgola (,), barra verticale (|), tabulazione (\ t), punto e virgola (;) e Ctrl-A (\u0001). Ctrl-A è il carattere di controllo Unicode per `Start Of Heading`.
Amazon Redshift	`redshift`	Utilizza la connessione JDBC per importare i metadati.
MySQL	`mysql`	Utilizza la connessione JDBC per importare i metadati.
PostgreSQL	`postgresql`	Utilizza la connessione JDBC per importare i metadati.

Tipo di classificatore	Stringa di classificazione	Note
Oracle database	`oracle`	Utilizza la connessione JDBC per importare i metadati.
Microsoft SQL Server	`sqlserver`	Utilizza la connessione JDBC per importare i metadati.
Amazon DynamoDB	`dynamodb`	Legge i dati dalla tabella DynamoDB.

I file nei seguenti formati compressi possono essere classificati:

- ZIP (supportato per gli archivi che contengono un solo file). Il formato ZIP non è supportato in modo adeguato in altri servizi (a causa dell'archivio).
- BZIP
- GZIP
- LZ4
- Snappy (supportato sia per i formati Snappy standard che nativi Hadoop)

Classificatore CSV integrato

Il classificatore CSV predefinito analizza il contenuto dei file CSV per determinare lo schema per una tabella AWS Glue. Questo classificatore controlla i seguenti delimitatori:

- Virgola (,)
- Barra verticale (|)
- Tabulazione (\t)
- Punto e virgola (;)
- Ctrl-A (\u0001)

  Ctrl-A è il carattere di controllo Unicode per `Start Of Heading`.

Per essere classificato come CSV, lo schema della tabella deve avere almeno due colonne e due righe di dati. Il classificatore CSV utilizza una serie di procedimenti euristici per determinare se un'intestazione è presente in un determinato file. Se il classificatore non è in grado di determinare l'intestazione della prima riga di dati, le intestazioni delle colonne vengono visualizzate come `col1`, `col2`, `col3` e così via. Il classificatore CSV integrato stabilisce se dedurre un'intestazione valutando le seguenti caratteristiche del file:

- Ogni colonna in una potenziale intestazione compare come tipo di dati STRING.
- Ad eccezione dell'ultima colonna, ogni colonna in una potenziale intestazione contiene meno di 150 caratteri. Per consentire un delimitatore finale, l'ultima colonna può essere vuota in tutto il file.
- Ogni colonna in una potenziale intestazione deve soddisfare i requisiti di AWS Glue in `regex`.
- La riga di intestazione deve essere sufficientemente diversa dalle righe di dati. Per determinarla, una o più righe devono comparire come diverse dal tipo STRING. Se tutte le colonne sono di tipo STRING, la prima riga di dati non è abbastanza diversa dalle righe successive per essere utilizzata come intestazione.

> **Note**
>
> Se il classificatore CSV predefinito non crea la tabella AWS Glue come desideri, puoi usare una delle alternative seguenti:
>
> - Modifica i nomi delle colonne nel catalogo dati, imposta `SchemaChangePolicy` su LOG e imposta la configurazione di output della partizione su `InheritFromTable` per le future esecuzioni del crawler.
> - Creare un classificatore grok personalizzato per analizzare i dati e assegnare le colonne desiderate.
> - Il classificatore CSV integrato crea tabelle referenziando `LazySimpleSerDe` come libreria di serializzazione, che è una valida scelta per l'inferenza del tipo. Tuttavia, se i dati CSV contengono stringhe racchiuse tra virgolette, modificare la definizione della tabella e modificare la libreria SerDe in `OpenCSVSerDe`. Modificare tutti i tipi dedotti in STRING, impostare `SchemaChangePolicy` su LOG e impostare la configurazione di output delle partizioni su `InheritFromTable` per le future esecuzioni del crawler. Per ulteriori informazioni sulle librerie SerDe, consulta la documentazione di riferimento su SerDe nella guida per l'utente di Amazon Athena.

Scrittura di classificatori personalizzati

Puoi fornire un classificatore personalizzato per classificare i dati in AWS Glue. Un classificatore personalizzato si può creare utilizzando un pattern grok, un tag XML, JavaScript Object Notation (JSON) o CSV (valori separati da virgola). Un crawler AWS Glue richiama un classificatore personalizzato. Se il classificatore riconosce i dati, esso restituisce la classificazione e lo schema dei dati al crawler. Potrebbe essere necessario definire un classificatore personalizzato nel caso in cui i dati non corrispondano ad alcun classificatore integrato, oppure se si intende personalizzare le tabelle create dal crawler.

Per ulteriori informazioni sulla creazione di un classificatore tramite la console AWS Glue, consulta Uso di classificatori nella console AWS Glue.

AWS Glue esegue i classificatori personalizzati prima dei classificatori predefiniti, in base all'ordine da te specificato. Quando un crawler individua un classificatore corrispondente ai dati, lo schema e la stringa di classificazione vengono usati nella definizione delle tabelle che vengono scritte nel AWS Glue Data Catalog.

Argomenti

Scrittura di classificatori personalizzati grok

Grok è uno strumento che consente di analizzare dati testuali dato un pattern corrispondente. Un pattern grok è un set denominato di espressioni regolari (regex) che vengono usate per confrontare i dati di una riga per volta. AWS Glue usa pattern grok dedurre lo schema dei dati. Quando un pattern grok individua la corrispondenza dei dati, AWS Glue usa il pattern per determinare la struttura dei dati e mapparla in campi.

AWS Glue offre numerosi pattern predefiniti, ma puoi definirne di personalizzati. È possibile creare un pattern grok tramite pattern integrati o pattern personalizzati nella definizione di classificatori personalizzati. È possibile adattare un pattern grok per classificare i formati di file di testo personalizzati.

> **Note**
>
> I classificatori personalizzati grok AWS Glue usano la libreria di serializzazione `GrokSerDe` per le tabelle create nel AWS Glue Data Catalog. Se usi AWS Glue Data Catalog con Amazon Athena, Amazon EMR o Redshift Spectrum, consulta la documentazione di questi servizi per informazioni sul supporto di `GrokSerDe`. Al momento, possono verificarsi problemi di esecuzione di query su tabelle create con `GrokSerDe` da Amazon EMR e Redshift Spectrum.

Di seguito è riportata la sintassi di base per i componenti di un pattern grok:

```
%{PATTERN:field-name}
```

I dati corrispondenti al `PATTERN` denominato vengono mappati alla colonna `field-name` nello schema, con un tipo di `string` di dati predefinito. Facoltativamente, è possibile eseguire il cast del tipo di dati per il campo in `byte`, `boolean`, `double`, `short`, `int`, `long` o `float` nello schema risultante.

```
%{PATTERN:field-name:data-type}
```

Ad esempio, per trasmettere un campo `num` a un tipo di dati `int`, si può utilizzare questo pattern:

```
%{NUMBER:num:int}
```

I pattern possono essere costituiti da altri pattern. Ad esempio, può esserci un pattern per un timestamp `SYSLOG` definito da pattern per mese, giorno del mese e ora (ad esempio, `Feb 1 06:25:43`). Per questi dati, è possibile definire il pattern seguente:

```
SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
```

> **Note**
>
> I pattern grok sono in grado di elaborare una sola riga alla volta. I pattern a righe multiple non sono supportati. In aggiunta, le interruzioni di riga all'interno dei pattern non sono supportate.

Valori dei classificatori personalizzati in AWS Glue

Quando definisci un classificatore grok, devi specificare i valori seguenti in AWS Glue per creare il classificatore personalizzato.

Nome

Nome del classificatore.

Classificazione

La stringa di testo scritta per descrivere il formato dei dati da classificare, ad esempio `special-logs`.

Pattern grok

Il set di pattern applicati al datastore per determinare l'esistenza di corrispondenze. Questi pattern provengono dai AWS Gluepattern predefiniti di e da eventuali pattern personalizzati da te definiti.

Di seguito è riportato un esempio di pattern grok:

```
%{TIMESTAMP_ISO8601:timestamp} \[%{MESSAGEPREFIX:message_prefix}\]
 %{CRAWLERLOGLEVEL:loglevel} : %{GREEDYDATA:message}
```

Quando i dati corrispondono a `TIMESTAMP_ISO8601`, viene creata una colonna di schema `timestamp`. Il funzionamento è analogo per i restanti pattern denominati nell'esempio.

Pattern personalizzati

Pattern personalizzati facoltativi da te definiti. Il pattern grok che classifica i dati fa riferimento a questi pattern. È possibile fare riferimento a questi pattern personalizzati nel pattern grok applicato ai dati. Ciascun pattern personalizzato che compone il pattern grok deve trovarsi su righe separate. Per definire il pattern, si utilizza la sintassi Espressione regolare (regex).

Di seguito è riportato un esempio di utilizzo di pattern personalizzati:

```
CRAWLERLOGLEVEL (BENCHMARK|ERROR|WARN|INFO|TRACE)
MESSAGEPREFIX .*-.*-.*-.*-.*
```

Il primo pattern personalizzato denominato, `CRAWLERLOGLEVEL`, costituisce una corrispondenza quando i dati corrispondono a una delle stringhe enumerate. Il secondo pattern personalizzato, `MESSAGEPREFIX`, tenta di abbinare una stringa di messaggio di prefisso.

AWS Glue tiene traccia della data e dell'ora di creazione, della data e dell'ora dell'ultimo aggiornamento e della versione del classificatore.

Pattern predefiniti in AWS Glue

AWS Glue offre molti pattern comuni che puoi usare per creare un classificatore personalizzato. É possibile aggiungere un pattern denominato al `grok pattern` all'interno di una definizione di classificatore.

L'elenco seguente comprende una riga per ciascun pattern. In ciascuna riga, il nome del pattern è seguito dalla sua definizione. Per definire il pattern, viene usata la sintassi delle espressioni regolari (regex).

```
#<noloc>&GLU;</noloc> Built-in patterns
 USERNAME [a-zA-Z0-9._-]+
 USER %{USERNAME:UNWANTED}
 INT (?:[+-]?(?:[0-9]+))
 BASE10NUM (?<![0-9.+-])(?>[+-]?(?:(?:[0-9]+(?:\.[0-9]+)?)|(?:\.[0-9]+)))
 NUMBER (?:%{BASE10NUM:UNWANTED})
 BASE16NUM (?<![0-9A-Fa-f])(?:[+-]?(?:0x)?(?:[0-9A-Fa-f]+))
 BASE16FLOAT \b(?<![0-9A-Fa-f.])(?:[+-]?(?:0x)?(?:(?:[0-9A-Fa-f]+(?:\.[0-9A-Fa-f]*)?)|
(?:\.[0-9A-Fa-f]+)))\b
 BOOLEAN (?i)(true|false)

 POSINT \b(?:[1-9][0-9]*)\b
 NONNEGINT \b(?:[0-9]+)\b
 WORD \b\w+\b
 NOTSPACE \S+
 SPACE \s*
 DATA .*?
 GREEDYDATA .*
 #QUOTEDSTRING (?:(?<!\\)(?:"(?:\\.|[^\\"])*"|(?:'(?:\\.|[^\\'])*')|(?:`(?:\\.|[^\
\`])*`)))
 QUOTEDSTRING (?>(?<!\\)(?>"(?>\\.|[^\\"]+)+"|""|(?>'(?>\\.|[^\\']+)+')|''|(?>`(?>\\.|
[^\\`]+)+`)|``))
 UUID [A-Fa-f0-9]{8}-(?:[A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}

 # Networking
 MAC (?:%{CISCOMAC:UNWANTED}|%{WINDOWSMAC:UNWANTED}|%{COMMONMAC:UNWANTED})
 CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2}[A-Fa-f0-9]{4})
 WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-){5}[A-Fa-f0-9]{2})
 COMMONMAC (?:(?:[A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2})
```

```
 IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-
Fa-f]{1,4}|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))
{3})|:))|(([0-9A-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2})|:((25[0-5]|2[0-4]\d|1\d
\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){4}(((:
[0-9A-Fa-f]{1,4}){1,3})|((:[0-9A-Fa-f]{1,4})?:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.
(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-Fa-f]
{1,4}){1,4})|((:[0-9A-Fa-f]{1,4}){0,2}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|
2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){2}(((:[0-9A-Fa-f]{1,4}){1,5})|
((:[0-9A-Fa-f]{1,4}){0,3}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d
\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){1}(((:[0-9A-Fa-f]{1,4}){1,6})|((:[0-9A-Fa-
f]{1,4}){0,4}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))
{3}))|:))|(:(((:[0-9A-Fa-f]{1,4}){1,7})|((:[0-9A-Fa-f]{1,4}){0,5}:((25[0-5]|2[0-4]\d|
1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:)))(%.+)?
 IPV4 (?<![0-9])(?:(?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|
[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|
[0-1]?[0-9]{1,2}))(?![0-9])
 IP (?:%{IPV6:UNWANTED}|%{IPV4:UNWANTED})
 HOSTNAME \b(?:[0-9A-Za-z][0-9A-Za-z-_]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-_]
{0,62}))*(\.?|\b)
 HOST %{HOSTNAME:UNWANTED}
 IPORHOST (?:%{HOSTNAME:UNWANTED}|%{IP:UNWANTED})
 HOSTPORT (?:%{IPORHOST}:%{POSINT:PORT})

 # paths
 PATH (?:%{UNIXPATH}|%{WINPATH})
 UNIXPATH (?>/(?>[\w_%!$@:.,~-]+|\\.)*)+
 #UNIXPATH (?<![\w\/])(?:/[^\/\s?*]*)+
 TTY (?:/dev/(pts|tty([pq])?)(\w+)?/?(?:[0-9]+))
 WINPATH (?>[A-Za-z]+:|\\)(?:\\[^\\?*]*)+
 URIPROTO [A-Za-z]+(\+[A-Za-z+]+)?
 URIHOST %{IPORHOST}(?::%{POSINT:port})?
 # uripath comes loosely from RFC1738, but mostly from what Firefox
 # doesn't turn into %XX
 URIPATH (?:/[A-Za-z0-9$.+!*'(){},~:;=@#%_\-]*)+
 #URIPARAM \?(?:[A-Za-z0-9]+(?:=(?:[^&]*))?(?:&(?:[A-Za-z0-9]+(?:=(?:[^&]*))?)?)*)?
 URIPARAM \?[A-Za-z0-9$.+!*'|(){},~@#%&/=:;_?\-\[\]]*
 URIPATHPARAM %{URIPATH}(?:%{URIPARAM})?
 URI %{URIPROTO}://(?:%{USER}(?::[^@]*)?@)?(?:%{URIHOST})?(?:%{URIPATHPARAM})?

 # Months: January, Feb, 3, 03, 12, December
 MONTH \b(?:Jan(?:uary)?|Feb(?:ruary)?|Mar(?:ch)?|Apr(?:il)?|May|Jun(?:e)?|Jul(?:y)?|
Aug(?:ust)?|Sep(?:tember)?|Oct(?:ober)?|Nov(?:ember)?|Dec(?:ember)?)\b
 MONTHNUM (?:0?[1-9]|1[0-2])
 MONTHNUM2 (?:0[1-9]|1[0-2])
```

```
 MONTHDAY (?:(?:0[1-9])|(?:[12][0-9])|(?:3[01])|[1-9])

 # Days: Monday, Tue, Thu, etc...
 DAY (?:Mon(?:day)?|Tue(?:sday)?|Wed(?:nesday)?|Thu(?:rsday)?|Fri(?:day)?|
Sat(?:urday)?|Sun(?:day)?)

 # Years?
 YEAR (?>\d\d){1,2}
 # Time: HH:MM:SS
 #TIME \d{2}:\d{2}(?::\d{2}(?:\.\d+)?)?
 # TIME %{POSINT<24}:%{POSINT<60}(?::%{POSINT<60}(?:\.%{POSINT})?)?
 HOUR (?:2[0123]|[01]?[0-9])
 MINUTE (?:[0-5][0-9])
 # '60' is a leap second in most time standards and thus is valid.
 SECOND (?:(?:[0-5]?[0-9]|60)(?:[:.,][0-9]+)?)
 TIME (?!<[0-9])%{HOUR}:%{MINUTE}(?::%{SECOND})(?![0-9])
 # datestamp is YYYY/MM/DD-HH:MM:SS.UUUU (or something like it)
 DATE_US %{MONTHNUM}[/-]%{MONTHDAY}[/-]%{YEAR}
 DATE_EU %{MONTHDAY}[./-]%{MONTHNUM}[./-]%{YEAR}
 DATESTAMP_US %{DATE_US}[- ]%{TIME}
 DATESTAMP_EU %{DATE_EU}[- ]%{TIME}
 ISO8601_TIMEZONE (?:Z|[+-]%{HOUR}(?::?%{MINUTE}))
 ISO8601_SECOND (?:%{SECOND}|60)
 TIMESTAMP_ISO8601 %{YEAR}-%{MONTHNUM}-%{MONTHDAY}[T ]%{HOUR}:?%{MINUTE}(?::?
%{SECOND})?%{ISO8601_TIMEZONE}?
 TZ (?:[PMCE][SD]T|UTC)
 DATESTAMP_RFC822 %{DAY} %{MONTH} %{MONTHDAY} %{YEAR} %{TIME} %{TZ}
 DATESTAMP_RFC2822 %{DAY}, %{MONTHDAY} %{MONTH} %{YEAR} %{TIME} %{ISO8601_TIMEZONE}
 DATESTAMP_OTHER %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{TZ} %{YEAR}
 DATESTAMP_EVENTLOG %{YEAR}%{MONTHNUM2}%{MONTHDAY}%{HOUR}%{MINUTE}%{SECOND}
 CISCOTIMESTAMP %{MONTH} %{MONTHDAY} %{TIME}

 # Syslog Dates: Month Day HH:MM:SS
 SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
 PROG (?:[\w._/%-]+)
 SYSLOGPROG %{PROG:program}(?:\[%{POSINT:pid}\])?
 SYSLOGHOST %{IPORHOST}
 SYSLOGFACILITY <%{NONNEGINT:facility}.%{NONNEGINT:priority}>
 HTTPDATE %{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME} %{INT}

 # Shortcuts
 QS %{QUOTEDSTRING:UNWANTED}

 # Log formats
```

```
 SYSLOGBASE %{SYSLOGTIMESTAMP:timestamp} (?:%{SYSLOGFACILITY} )?%{SYSLOGHOST:logsource}
 %{SYSLOGPROG}:

 MESSAGESLOG %{SYSLOGBASE} %{DATA}

 COMMONAPACHELOG %{IPORHOST:clientip} %{USER:ident} %{USER:auth}
 \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/
%{NUMBER:httpversion})?|%{DATA:rawrequest})" %{NUMBER:response} (?:%{Bytes:bytes=
%{NUMBER}|-})
 COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}
 COMMONAPACHELOG_DATATYPED %{IPORHOST:clientip} %{USER:ident;boolean} %{USER:auth}
 \[%{HTTPDATE:timestamp;date;dd/MMM/yyyy:HH:mm:ss Z}\] "(?:%{WORD:verb;string}
 %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion;float})?|%{DATA:rawrequest})"
 %{NUMBER:response;int} (?:%{NUMBER:bytes;long}|-)


 # Log Levels
 LOGLEVEL ([A|a]lert|ALERT|[T|t]race|TRACE|[D|d]ebug|DEBUG|[N|n]otice|NOTICE|[I|i]nfo|
INFO|[W|w]arn?(?:ing)?|WARN?(?:ING)?|[E|e]rr?(?:or)?|ERR?(?:OR)?|[C|c]rit?(?:ical)?|
CRIT?(?:ICAL)?|[F|f]atal|FATAL|[S|s]evere|SEVERE|EMERG(?:ENCY)?|[Ee]merg(?:ency)?)
```

Scrittura di classificatori personalizzati XML

XML definisce la struttura di un documento tramite l'uso di tag nel file. Con un classificatore personalizzato XML, è possibile specificare il nome tag utilizzato per definire una riga.

Valori dei classificatori personalizzati in AWS Glue

Quando definisci un classificatore XML, devi specificare i valori seguenti in AWS Glue per creare il classificatore. Il campo classificazione di questo classificatore è impostato su `xml`.

Nome

Nome del classificatore.

Tag di riga

Il nome tag XML che definisce una riga di tabella nel documento XML, senza parentesi angolate < >. Il nome deve rispettare le regole XML relative ai tag.

> **Note**
>
> L'elemento contenente i dati di riga non può essere un elemento vuoto che si auto-chiude. Questo elemento vuoto, ad esempio, non viene analizzato da AWS Glue:

```
<row att1=”xx” att2=”yy” />
```

È possibile scrivere gli elementi vuoti come segue:

```
<row att1=”xx” att2=”yy”> </row>
```

AWS Glue tiene traccia della data e dell'ora di creazione, della data e dell'ora dell'ultimo aggiornamento e della versione del classificatore.

Supponi, ad esempio, di avere il file XML seguente. Per creare una tabella AWS Glue che contiene solo le colonne per autore e titolo, crea un classificatore nella console AWS Glue con Row tag (Tag di riga) impostato su `AnyCompany`. Aggiungi ed esegui quindi un crawler che usa questo classificatore personalizzato.

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
     <AnyCompany>
       <author>Rivera, Martha</author>
       <title>AnyCompany Developer Guide</title>
     </AnyCompany>
   </book>
   <book id="bk102">
     <AnyCompany>
       <author>Stiles, John</author>
       <title>Style Guide for AnyCompany</title>
     </AnyCompany>
   </book>
</catalog>
```

Scrittura di classificatori personalizzati JSON

JSON è un formato per lo scambio di dati. Definisce le strutture di dati con coppie nome-valore o con un elenco ordinato di valori. Con un classificatore personalizzato JSON, puoi specificare il percorso JSON di una struttura di dati usata per definire lo schema per la tabella.

Valori dei classificatori personalizzati in AWS Glue

Quando definisci un classificatore JSON, devi specificare i valori seguenti in AWS Glue per creare il classificatore. Il campo classificazione di questo classificatore è impostato su `json`.

Nome

Nome del classificatore.

Percorso JSON

Percorso JSON che fa riferimento a un oggetto usato per definire uno schema di tabella. È possibile scrivere il percorso JSON in notazione punto o in notazione parentesi. Sono supportati i seguenti operatori:

Descrizione
Elemento radice di un oggetto JSON. Avvia tutte le espressioni del percorso
Carattere jolly. Disponibile ovunque siano necessari un nome o un elemento numerico nel percorso JSON.
Figlio con notazione dot. Specifica un campo figlio in un oggetto JSON.
Figlio con notazione parentesi. Specifica campi figli in un oggetto JSON. Solo un singolo campo figlio può essere specificato.
Indice di matrice. Specifica il valore di una matrice in base all'indice.

AWS Glue tiene traccia della data e dell'ora di creazione, della data e dell'ora dell'ultimo aggiornamento e della versione del classificatore.

Example Utilizzo di un classificatore JSON per estrarre record da una matrice

Supponi che i dati JSON siano costituiti da una matrice di record. Ad esempio, le prime righe del file potrebbero apparire come segue:

```
[
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ak",
    "name": "Alaska"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:1",
    "name": "Alabama's 1st congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:2",
    "name": "Alabama's 2nd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:3",
    "name": "Alabama's 3rd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:4",
    "name": "Alabama's 4th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:5",
    "name": "Alabama's 5th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:6",
    "name": "Alabama's 6th congressional district"
  },
  {
    "type": "constituency",
```

```
    "id": "ocd-division\/country:us\/state:al\/cd:7",
    "name": "Alabama's 7th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:1",
    "name": "Arkansas's 1st congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:2",
    "name": "Arkansas's 2nd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:3",
    "name": "Arkansas's 3rd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:4",
    "name": "Arkansas's 4th congressional district"
  }
]
```

Quando esegui un crawler usando il classificatore JSON predefinito, l'intero file viene utilizzato per definire lo schema. Poiché non specifichi un percorso JSON, il crawler tratta i dati come un unico oggetto, ovvero, come una matrice. Ad esempio, lo schema potrebbe apparire come segue:

```
root
|-- record: array
```

Tuttavia, per creare uno schema basato su ciascun record presente nella matrice JSON, crea un classificatore JSON personalizzato e specifica il percorso JSON come `$[*]`. Quando si specifica questo percorso JSON, il classificatore interroga tutti i 12 record presenti nella matrice per determinare lo schema. Lo schema risultante contiene campi separati per ciascun oggetto, analogamente all'esempio seguente:

```
root
|-- type: string
```

```
|-- id: string
|-- name: string
```

Example Utilizzo di un classificatore JSON per esaminare solo alcune parti di un file

Supponi che i dati JSON seguano il pattern del file di esempio JSON `s3://awsglue-datasets/examples/us-legislators/all/areas.json` estratto dal sito http://everypolitician.org/. Gli oggetti di esempio nel file JSON hanno l'aspetto seguente:

```
{
  "type": "constituency",
  "id": "ocd-division\/country:us\/state:ak",
  "name": "Alaska"
}
{
  "type": "constituency",
  "identifiers": [
    {
      "scheme": "dmoz",
      "identifier": "Regional\/North_America\/United_States\/Alaska\/"
    },
    {
      "scheme": "freebase",
      "identifier": "\/m\/0hjy"
    },
    {
      "scheme": "fips",
      "identifier": "US02"
    },
    {
      "scheme": "quora",
      "identifier": "Alaska-state"
    },
    {
      "scheme": "britannica",
      "identifier": "place\/Alaska"
    },
    {
      "scheme": "wikidata",
      "identifier": "Q797"
    }
  ],
```

```
  "other_names": [
    {
      "lang": "en",
      "note": "multilingual",
      "name": "Alaska"
    },
    {
      "lang": "fr",
      "note": "multilingual",
      "name": "Alaska"
    },
    {
      "lang": "nov",
      "note": "multilingual",
      "name": "Alaska"
    }
  ],
  "id": "ocd-division\/country:us\/state:ak",
  "name": "Alaska"
}
```

Quando esegui un crawler usando il classificatore JSON predefinito, l'intero file viene utilizzato per creare lo schema. È possibile ritrovarsi con uno schema di questo tipo:

```
root
|-- type: string
|-- id: string
|-- name: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
```

Tuttavia, per creare uno schema usando solo l'oggetto "`id`", crea un classificatore JSON personalizzato e specifica il percorso JSON come `$.id`. Lo schema sarà così basato solo sul campo "`id`":

```
root
|-- record: string
```

Le prime righe di dati estratte con questo schema hanno l'aspetto seguente:

```
{"record": "ocd-division/country:us/state:ak"}
{"record": "ocd-division/country:us/state:al/cd:1"}
{"record": "ocd-division/country:us/state:al/cd:2"}
{"record": "ocd-division/country:us/state:al/cd:3"}
{"record": "ocd-division/country:us/state:al/cd:4"}
{"record": "ocd-division/country:us/state:al/cd:5"}
{"record": "ocd-division/country:us/state:al/cd:6"}
{"record": "ocd-division/country:us/state:al/cd:7"}
{"record": "ocd-division/country:us/state:ar/cd:1"}
{"record": "ocd-division/country:us/state:ar/cd:2"}
{"record": "ocd-division/country:us/state:ar/cd:3"}
{"record": "ocd-division/country:us/state:ar/cd:4"}
{"record": "ocd-division/country:us/state:as"}
{"record": "ocd-division/country:us/state:az/cd:1"}
{"record": "ocd-division/country:us/state:az/cd:2"}
{"record": "ocd-division/country:us/state:az/cd:3"}
{"record": "ocd-division/country:us/state:az/cd:4"}
{"record": "ocd-division/country:us/state:az/cd:5"}
{"record": "ocd-division/country:us/state:az/cd:6"}
{"record": "ocd-division/country:us/state:az/cd:7"}
```

Per creare uno schema basato su un oggetto con nidificazione profonda, come "`identifier`", nel file JSON puoi creare un classificatore JSON personalizzato e specificare il percorso JSON come `$.identifiers[*].identifier`. Sebbene lo schema sia simile a quello dell'esempio precedente, si basa su un oggetto diverso nel file JSON.

Lo schema ha il seguente aspetto:

```
root
```

```
|-- record: string
```

Elencando le prime righe di dati della tabella è possibile vedere che lo schema si basa sui dati nell'oggetto "`identifier`":

```
{"record": "Regional/North_America/United_States/Alaska/"}
{"record": "/m/0hjy"}
{"record": "US02"}
{"record": "5879092"}
{"record": "4001016-8"}
{"record": "destination/alaska"}
{"record": "1116270"}
{"record": "139487266"}
{"record": "n79018447"}
{"record": "01490999-8dec-4129-8254-eef6e80fadc3"}
{"record": "Alaska-state"}
{"record": "place/Alaska"}
{"record": "Q797"}
{"record": "Regional/North_America/United_States/Alabama/"}
{"record": "/m/0gyh"}
{"record": "US01"}
{"record": "4829764"}
{"record": "4084839-5"}
{"record": "161950"}
{"record": "131885589"}
```

Per creare una tabella basata su un altro oggetto con nidificazione profonda, come il campo "`name`" nella matrice "`other_names`" nel file JSON, puoi creare un classificatore JSON personalizzato e specificare il percorso JSON come `$.other_names[*].name`. Sebbene lo schema sia simile a quello dell'esempio precedente, si basa su un oggetto diverso nel file JSON. Lo schema ha il seguente aspetto:

```
root
|-- record: string
```

Elencando le prime righe di dati della tabella è possibile vedere che lo schema si basa sui dati nell'oggetto "`name`" nella matrice "`other_names`":

```
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Аляска"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "######"}
{"record": "######"}
{"record": "######"}
{"record": "Alaska"}
{"record": "Alyaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Штат Аляска"}
{"record": "Аляска"}
{"record": "Alaska"}
{"record": "#######"}
```

Scrittura di classificatori personalizzati CSV

I classificatori CSV personalizzati consentono di specificare i tipi di dati per ogni colonna nel campo del classificatore CSV personalizzato. È possibile specificare il tipo di dati di ogni colonna separato da una virgola. Specificando i tipi di dati, è possibile sovrascrivere i tipi di dati dedotti dai crawler e garantire che i dati vengano classificati in modo appropriato.

È possibile impostare il SerDe per l'elaborazione del CSV nel classificatore, che verrà applicato in Catalogo dati.

Quando crei un classificatore personalizzato, puoi anche riutilizzare il classificatore per diversi crawler.

- Per i file csv con solo intestazioni (nessun dato), questi file verranno classificati come SCONOSCIUTI poiché non vengono fornite informazioni sufficienti. Se specifichi che il CSV “contiene titoli” nell'opzione Column headings (Intestazioni delle colonne) e fornisci i tipi di dati, possiamo classificare correttamente questi file.

Puoi usare un classificatore CSV personalizzato per dedurre lo schema di vari tipi di dati CSV. Gli attributi personalizzati che puoi fornire per il classificatore includono i delimitatori, un'opzione SerDe CSV, le opzioni relative all'intestazione e l'indicazione se eseguire determinate convalide sui dati.

Valori dei classificatori personalizzati in AWS Glue

Quando definisci un classificatore CSV, devi fornire i valori seguenti in AWS Glue per creare il classificatore. Il campo classificazione di questo classificatore è impostato su `csv`.

Nome del classificatore

Nome del classificatore.

SerDe CSV

Imposta il SerDe per l'elaborazione del CSV nel classificatore, che verrà applicato in Catalogo dati. Le opzioni sono Open CSV SerDe, Lazy Simple SerDe e None. È possibile specificare il valore None quando si desidera che il crawler esegua il rilevamento.

Delimitatore di colonna

Un simbolo personalizzato per indicare il separatore di ogni voce di colonna nella riga. Fornisci un carattere unicode. Se non riesci a digitare il delimitatore, puoi copiarlo e incollarlo. Questo vale per i caratteri stampabili, compresi quelli non supportati dal sistema (in genere indicati come □).

Simbolo di virgolette

Un simbolo personalizzato per indicare la combinazione dei contenuti in un singolo valore di colonna. Deve essere diverso dal delimitatore di colonna. Fornisci un carattere unicode. Se non riesci a digitare il delimitatore, puoi copiarlo e incollarlo. Questo vale per i caratteri stampabili, compresi quelli non supportati dal sistema (in genere indicati come □).

Intestazioni di colonna

Indica il comportamento per il modo in cui le intestazioni di colonna devono essere rilevate nel file CSV. Se il file CSV personalizzato include le intestazioni di colonna, inserisci un elenco di intestazioni di colonna delimitate da virgole.

Opzioni di elaborazione: consenti i file con una singola colonna

Abilita l'elaborazione dei file che contengono una sola colonna.

Opzioni di elaborazione: taglia lo spazio vuoto prima dell'identificazione dei valori di colonna

Specifica se tagliare i valori prima di individuare il tipo dei valori di colonna.

Tipi di dati personalizzati: facoltativo

Inserisci il tipo di dati personalizzato separato da una virgola. Specifica i tipi di dati personalizzati nel file CSV. Il tipo di dati personalizzato deve essere un tipo di dati supportato. I tipi di dati supportati sono: "BINARY", "BOOLEAN", "DATE", "DECIMAL", "DOUBLE", "FLOAT", "INT", "LONG", "SHORT", "STRING", "TIMESTAMP". I tipi di dati non supportati mostreranno un errore.

Uso di classificatori nella console AWS Glue

Un classificatore determina lo schema dei dati. Puoi scrivere un classificatore personalizzato e puntarvi da AWS Glue.

Visualizzazione dei classificatori

Per visualizzare un elenco di tutti i classificatori creati, apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/ e scegli la scheda Classifiers (Classificatori).

Nell'elenco sono riportate le seguenti proprietà per ogni classificatore:

- Classifier (Classificatore) – Il nome del classificatore. Quando crei un classificatore, devi specificarne il nome.
- Classification (Classificazione) – Il tipo di classificazione delle tabelle dedotte dal classificatore.
- Last updated (Ultimo aggiornamento) – L'ultima volta in cui è stato aggiornato il classificatore.

Gestione dei classificatori

Nell'elenco Classifiers (Classificatori) nella console AWS Glue puoi aggiungere, modificare ed eliminare classificatori. Per visualizzare ulteriori dettagli per un classificatore, scegli il nome nell'elenco. I dettagli sono le informazioni cha hai definito al momento della creazione del classificatore.

Creazione dei classificatori

Per aggiungere un classificatore nella console AWS Glue, scegli Add classifier (Aggiungi classificatore). Quando definisci un classificatore, specifichi i valori per le seguenti opzioni:

- Classifier name (Nome del classificatore) – Fornisci un nome univoco per il tuo classificatore.
- Classifier type (Tipo di classificazione) – Il tipo di classificazione delle tabelle dedotte dal classificatore.

- Last updated (Ultimo aggiornamento) – L'ultima volta in cui è stato aggiornato il classificatore.

Nome del classificatore

Fornisci un nome univoco per il tuo classificatore.

Tipo di classificatore

Scegli il tipo di classificatore da creare.

A seconda del tipo di classificatore scelto, configurare le seguenti proprietà per il classificatore:

Grok

- Classificazione

  Descrivi il formato o il tipo di dati classificati o fornisci un'etichetta personalizzata.

- Pattern grok

  Viene utilizzato per analizzare i dati in uno schema strutturato. Il pattern grok è composto da modelli denominati che descrivono il formato del datastore. Puoi scrivere questo pattern grok usando il modello predefinito denominato fornito da AWS Glue e i modelli personalizzati che scrivi e includi nel campo Custom patterns (Modelli personalizzati). Anche se i risultati dei debugger grok potrebbero non corrispondere esattamente ai risultati di AWS Glue, ti consigliamo di provare il modello usando alcuni dati di esempio con un debugger grok. Puoi trovare i debugger grok sul Web. I modelli predefiniti denominati forniti da AWS Glue sono generalmente compatibili con i modelli grok disponibili nel Web.

  Crea il tuo pattern grok aggiungendo iterativamente i modelli denominati e controlla i risultati in un debugger. Questa attività garantisce che i dati vengano analizzati quando il crawler AWS Glue esegue il pattern grok.

- Pattern personalizzati

  Per i classificatori grok, questi sono elementi costitutivi facoltativi per il Grok pattern (Pattern grok) che scrivi. Quando i modelli integrati non sono in grado di analizzare i dati, potrebbe essere necessario scrivere un modello personalizzato. Questi modelli personalizzati sono definiti in questo campo e referenziati nel campo Grok pattern (Pattern grok). Ciascun modello personalizzato è definito su una riga separata. Proprio come i modelli integrati, è costituito da una definizione di modello denominato che utilizza la sintassi di espressione regolare (regex).

Ad esempio, di seguito è riportato il nome `MESSAGEPREFIX` seguito da una definizione di espressione regolare da applicare ai dati per determinare se segue il modello.

```
MESSAGEPREFIX .*-.*-.*-.*-.*
```

XML

- Tag di riga

  Per i classificatori XML, questo è il nome del tag XML che definisce una riga di tabella nel documento XML. Digita il nome senza parentesi angolari < >. Il nome deve rispettare le regole XML relative ai tag.

  Per ulteriori informazioni, consulta Scrittura di classificatori personalizzati XML.

JSON

- Percorso JSON

  Per i classificatori JSON, questo è il percorso JSON dell'oggetto, della matrice o del valore che definisce una riga della tabella creata. Digita il nome nella sintassi JSON con punti o parentesi usando gli operatori supportati in AWS Glue.

  Per ulteriori informazioni, vedi l'elenco degli operatori in Scrittura di classificatori personalizzati JSON.

CSV

- Delimitatore di colonna

  Un singolo carattere o simbolo personalizzato per indicare il separatore di ogni voce di colonna nella riga. Scegli il delimitatore dall'elenco o scegli `Other` per immettere un delimitatore personalizzato.

- Simbolo di virgolette

Un singolo carattere o simbolo personalizzato per indicare la combinazione dei contenuti in un singolo valore di colonna. Deve essere diverso dal delimitatore di colonna. Scegli il simbolo di virgolette dall'elenco o scegli `Other` per immettere delle virgolette personalizzate.

- Intestazioni di colonna

  Indica il comportamento per il modo in cui le intestazioni di colonna devono essere rilevate nel file CSV. È possibile scegliere `Has headings`, `No headings`, oppure `Detect headings`. Se il file CSV personalizzato include le intestazioni di colonna, inserisci un elenco di intestazioni di colonna delimitate da virgole.

- Consenti i file con una singola colonna

  Per essere classificato come CSV, i dati devono avere almeno due colonne e due righe di dati. Utilizza questa opzione per consentire l'elaborazione dei file che contengono una sola colonna.

- Taglia lo spazio vuoto prima dell'identificazione dei valori di colonna

  Questa opzione specifica se tagliare i valori prima di individuare il tipo dei valori di colonna.

- Tipo di dati personalizzato

  (Facoltativo) - Inserisci tipi di dati personalizzati in un elenco delimitato da virgole. I tipi di dati supportati sono: "BINARY", "BOOLEAN", "DATE", "DECIMAL", "DOUBLE", "FLOAT", "INT", "LONG", "SHORT", "STRING", "TIMESTAMP".

- SerDe CSV

  (Facoltativo) - Un SerDe per l'elaborazione del file CSV nel classificatore che verrà applicato in Catalogo dati. Scegli tra `Open CSV SerDe`, `Lazy Simple SerDe` o `None`. È possibile specificare il valore `None` quando si desidera che il crawler esegua il rilevamento.

Per ulteriori informazioni, consulta Scrittura di classificatori personalizzati.

Registro dello schema di AWS Glue

> **Note**
>
> Il registro dello schema di AWS Glue non è supportato nelle seguenti Regioni della console AWS Glue : Asia Pacifico (Giacarta) e Medio Oriente (Emirati Arabi Uniti).

Il registro dello schema di AWS Glue è una nuova funzionalità per l'individuazione, il controllo e l'evoluzione in modo centralizzato degli schemi dei flussi dei dati. Uno schema definisce la struttura e il formato di un registro di dati. Con AWS Glue Schema Registry, puoi gestire e applicare gli schemi sulle tue applicazioni di streaming di dati utilizzando comode integrazioni con Apache Kafka, Amazon Kinesis Amazon Managed Streaming for Apache KafkaData Streams, Amazon Managed Service for Apache Flink e. AWS Lambda

Il registro degli schemi di AWS Glue supporta il formato dati AVRO (v1.10.2), il formato dati JSON con formato di schemi JSON per lo schema (specifiche Draft-04, Draft-06 e Draft-07) con la convalida dello schema JSON utilizzando la libreria Everit, le versioni proto2 e proto3 di Protocol Buffers (Protobuf) senza supporto per `extensions` o `groups` e il supporto Java, con altri linguaggi e formati di dati disponibili in futuro. Le caratteristiche supportate includono compatibilità, acquisizione dello schema tramite metadati, registrazione automatica degli schemi, compatibilità IAM e compressione ZLIB facoltativa per ridurre l'archiviazione e il trasferimento dei dati. Il registro degli schemi . AWS Glue Schema Registry è serverless e gratuito.

L'utilizzo di uno schema come contratto di formato dati tra produttori e consumer comporta una migliore governance dei dati, dati di qualità superiore e consente ai consumer di dati di essere resilienti alle modifiche upstream compatibili.

Il registro degli schemi consente a sistemi diversi di condividere uno schema per la serializzazione e la deserializzazione. Ad esempio, si supponga di avere un produttore e un consumer di dati. Il produttore conosce lo schema quando pubblica i dati. Il registro degli schemi fornisce un serializzatore e un deserializzatore per alcuni sistemi come Amazon MSK o Apache Kafka.

Per ulteriori informazioni, consulta Funzionamento del registro degli schemi.

Argomenti

- Migrazione da un registro degli schemi di terze parti al registro degli schemi di AWS Glue

Schemi

Uno schema definisce la struttura e il formato di un registro di dati. Uno schema è una specifica con versioni per la pubblicazione, il consumo o l'archiviazione dei dati in modo affidabile.

In questo schema di esempio per Avro, il formato e la struttura sono definiti dal layout e dai nomi dei campi e il formato dei nomi dei campi è definito dai tipi di dati (ad esempio, `string`, `int`).

```
{
    "type": "record",
    "namespace": "ABC_Organization",
    "name": "Employee",
    "fields": [
        {
            "name": "Name",
            "type": "string"
        },
        {
            "name": "Age",
            "type": "int"
        },
        {
            "name": "address",
            "type": {
                "type": "record",
                "name": "addressRecord",
                "fields": [
                    {
                        "name": "street",
                        "type": "string"
                    },
                    {
                        "name": "zipcode",
                        "type": "int"
                    }
                ]
            }
        }
    ]
}
```

In questo schema JSON Draft-07 per JSON di esempio, il formato è definito dall'organizzazione dello schema JSON.

```
{
 "$id": "https://example.com/person.schema.json",
 "$schema": "http://json-schema.org/draft-07/schema#",
 "title": "Person",
 "type": "object",
 "properties": {
  "firstName": {
   "type": "string",
   "description": "The person's first name."
  },
  "lastName": {
   "type": "string",
   "description": "The person's last name."
  },
  "age": {
   "description": "Age in years which must be equal to or greater than zero.",
   "type": "integer",
   "minimum": 0
  }
 }
}
```

In questo esempio per Protobuf, il formato è definito dalla versione 2 del linguaggio Protocol Buffers (proto2).

```
syntax = "proto2";

package tutorial;

option java_multiple_files = true;
option java_package = "com.example.tutorial.protos";
option java_outer_classname = "AddressBookProtos";

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
```

```
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    optional string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

Registri

Un registro è un container logico di schemi. I registri consentono di organizzare gli schemi e gestire il controllo degli accessi per le applicazioni. Un registro dispone di un Amazon Resource Name (ARN) che consente di organizzare e impostare diverse autorizzazioni di accesso per le operazioni dello schema all'interno del registro.

È possibile utilizzare il registro di default o creare tutti i nuovi registri necessari.

Gerarchia del registro degli schemi di AWS Glue

- RegistryName: [stringa]
  - RegistryArn: [AWS ARN]
  - CreatedTime: [timestamp]
  - UpdatedTime: [timestamp]
- SchemaName: [stringa]
  - SchemaArn: [AWS ARN]
  - DataFormat: [Avro, Json o Protobuf]
  - Compatibility: (ad es. BACKWARD, BACKWARD_ALL, FORWARD, FORWARD_ALL, FULL, FULL_ALL, NONE, DISABLED]
  - Status: [ad es. PENDING, AVAILABLE, DELETING]

  - SchemaCheckpoint: [numero intero]
  - CreatedTime: [timestamp]
  - UpdatedTime: [timestamp]
- SchemaVersion: [stringa]
  - SchemaVersionNumber: [numero intero]
  - Status: [ad es. PENDING, AVAILABLE, DELETING, FAILURE]
  - SchemaDefinition: [stringa, valore: JSON]
  - CreatedTime: [timestamp]
- SchemaVersionMetadata: [elenco]
  - MetadataKey: [stringa]
  - MetadataInfo
  - MetadataValue: [stringa]
  - CreatedTime: [timestamp]

Controllo delle versioni e compatibilità degli schemi

Ogni schema può avere più versioni. Il controllo delle versioni è regolato da una regola di compatibilità applicata a uno schema. Le richieste di registrazione di nuove versioni dello schema vengono verificate in base a questa regola dal registro degli schemi prima che possano avere esito positivo.

Una versione dello schema contrassegnata come checkpoint viene utilizzata per determinare la compatibilità della registrazione di nuove versioni di uno schema. Quando uno schema viene creato per la prima volta, il checkpoint di default sarà la prima versione. Man mano che lo schema evolve con più versioni, è possibile utilizzare la CLI/l'SDK per modificare il checkpoint a una versione di uno schema utilizzando l'API `UpdateSchema` che aderisce a un insieme di vincoli. Nella console, la modifica della definizione dello schema o della modalità di compatibilità modificherà il checkpoint alla versione più recente per impostazione predefinita.

Le modalità di compatibilità consentono di controllare come gli schemi possono o non possono evolvere nel tempo. Queste modalità costituiscono il contratto tra le applicazioni che producono e consumano dati. Quando una nuova versione di uno schema viene inviata al registro, la regola di compatibilità applicata al nome dello schema viene utilizzata per determinare se la nuova versione

può essere accettata. Esistono 8 modalità di compatibilità: NONE, DISABLED, BACKWARD, BACKWARD_ALL, FORWARD, FORWARD_ALL, FULL, FULL_ALL.

Nel formato dati Avro, i campi possono essere facoltativi o obbligatori. Un campo facoltativo è un campo in cui il `Type` include null. I campi obbligatori non contengono il valore null come `Type`.

Nel formato dati Protobuf, i campi possono essere facoltativi (anche ripetuti) o obbligatori nella sintassi proto2, mentre tutti i campi sono facoltativi (anche ripetuti) nella sintassi proto3. Tutte le regole di compatibilità sono determinate in base alla comprensione delle specifiche di Protocol Buffers e dalle linee guida della Documentazione Google Protocol Buffers.

- NONE: non si applica alcuna modalità di compatibilità. È possibile utilizzare questa opzione negli scenari di sviluppo o se non si conoscono le modalità di compatibilità da applicare agli schemi. Qualsiasi nuova versione aggiunta sarà accettata senza essere sottoposta a un controllo di compatibilità.
- DISABLED: questa scelta di compatibilità impedisce il controllo delle versioni per uno schema specifico. Non è possibile aggiungere nuove versioni.
- BACKWARD: questa scelta di compatibilità è consigliata in quanto consente ai consumer di leggere sia la versione attuale che quella precedente dello schema. È possibile utilizzare questa opzione per verificare la compatibilità con la versione precedente dello schema quando si eliminano campi o si aggiungono campi facoltativi. Un tipico caso d'uso per BACKWARD è quando l'applicazione è stata creata per lo schema più recente.

  AVRO

  Ad esempio, si supponga di avere uno schema definito da nome (obbligatorio), cognome (obbligatorio), e-mail (obbligatorio) e numero di telefono (facoltativo).

  Se la versione successiva dello schema rimuove il campo e-mail obbligatorio, questa operazione verrebbe registrata correttamente. La compatibilità BACKWARD richiede ai consumer di essere in grado di leggere la versione corrente e precedente dello schema. I consumer potranno leggere il nuovo schema in quanto il campo dell'e-mail in più viene ignorato dai vecchi messaggi.

  Se si dispone di una nuova versione dello schema proposta che aggiunge un campo obbligatorio, ad esempio il codice postale, con la compatibilità BACKWARD questo non verrebbe registrato correttamente. I consumer della nuova versione non sarebbero in grado di leggere i messaggi precedenti alla modifica dello schema, poiché mancherebbe il campo obbligatorio del codice postale. Tuttavia, se il campo del codice postale è impostato come facoltativo nel nuovo schema,

la versione proposta viene registrata correttamente poiché i consumer sono in grado di leggere il vecchio schema senza il campo facoltativo del codice postale.

JSON

Ad esempio, si supponga di avere una versione dello schema definita da nome (facoltativo), cognome (facoltativo), e-mail (facoltativa) e numero di telefono (facoltativo).

Se nella versione successiva dello schema viene aggiunta la proprietà facoltativa del numero di telefono, la registrazione viene eseguita correttamente, a condizione che la versione originale dello schema non consenta proprietà aggiuntive impostando il campo `additionalProperties` su false. La compatibilità BACKWARD richiede ai consumer di essere in grado di leggere la versione corrente e precedente dello schema. I consumer saranno in grado di leggere i dati prodotti con lo schema originale in cui la proprietà numero di telefono non esiste.

Se si dispone di una nuova versione dello schema proposta che aggiunge la proprietà facoltativa del numero di telefono, questa operazione non viene registrata correttamente con la compatibilità BACKWARD se la versione originale dello schema imposta il campo `additionalProperties` su true, vale a dire consentendo qualsiasi proprietà aggiuntiva. I consumer della nuova versione non sarebbero in grado di leggere i messaggi precedenti alla modifica dello schema, in quanto non possono leggere i dati con la proprietà numero di telefono in un tipo differente, ad esempio come stringa anziché numero.

PROTOBUF

Ad esempio, supponiamo di avere una versione di uno schema definito da un messaggio `Person` con i campi `first name` (obbligatorio), `last name` (obbligatorio), `email` (obbligatorio), e `phone number` (facoltativo) sotto la sintassi proto2.

Come negli scenari AVRO, se la versione successiva dello schema rimuove il campo `email` obbligatorio, questa operazione verrà registrata correttamente. La compatibilità BACKWARD richiede ai consumer di essere in grado di leggere la versione corrente e precedente dello schema. I consumer potranno leggere il nuovo schema in quanto il campo `email` in più viene ignorato dai vecchi messaggi.

Se si dispone di una nuova versione dello schema proposta che aggiunge un campo obbligatorio, ad esempio `zip code`, con la compatibilità BACKWARD questo non verrebbe registrato correttamente. I consumer della nuova versione non sarebbero in grado di leggere i messaggi precedenti alla modifica dello schema, poiché mancherebbe il campo obbligatorio `zip code`.

Tuttavia, se il campo `zip code` è impostato come facoltativo nel nuovo schema, la versione proposta viene registrata correttamente poiché i consumer sono in grado di leggere il vecchio schema senza il campo facoltativo `zip code`.

In caso di utilizzo di gRPC, l'aggiunta di un nuovo servizio RPC o metodo RPC è una modifica compatibile con le versioni precedenti. Ad esempio, supponiamo di avere una versione di uno schema definito da un servizio RPC `MyService` con due metodi RPC `Foo` e `Bar`.

Se la prossima versione dello schema aggiunge un nuovo metodo RPC chiamato `Baz`, questo si verrà registrato correttamente. I consumer saranno in grado di leggere i dati prodotti con lo schema originale in base alla compatibilità BACKWARD dal nuovo metodo RPC `Baz` è facoltativo.

Se si dispone di una nuova versione dello schema proposta che rimuove un campo obbligatorio, ad esempio il metodo RPC `Foo` esistente, con la compatibilità BACKWARD questo non verrebbe registrato correttamente. I consumer della nuova versione non sarebbero in grado di leggere i messaggi precedenti alla modifica dello schema, in quanto non possono comprendere e leggere i dati con il metodo RPC inesistente `Foo` in un'applicazione gRPC.

- BACKWARD_ALL:: questa scelta di compatibilità consente ai consumer di leggere sia la versione attuale che tutte quelle precedenti dello schema. È possibile utilizzare questa opzione per verificare la compatibilità con tutte le versioni precedenti dello schema quando si eliminano campi o si aggiungono campi facoltativi.
- FORWARD: questa scelta di compatibilità consente ai consumer di leggere sia la versione attuale che le versioni successive dello schema, ma non necessariamente le versioni più recenti. È possibile utilizzare questa opzione per verificare la compatibilità con l'ultima versione dello schema quando si aggiungono campi o si eliminano campi facoltativi. Un tipico caso d'uso per FORWARD è quando l'applicazione è stata creata per uno schema precedente e deve poter elaborare uno schema più recente.

  AVRO

  Ad esempio, si supponga di avere una versione di uno schema definito da nome (obbligatorio), cognome (obbligatorio), e-mail (facoltativo).

  Se si dispone di una nuova versione dello schema che aggiunge un campo obbligatorio, ad esempio il numero di telefono, la registrazione viene eseguita correttamente. La compatibilità FORWARD richiede ai consumatori di essere in grado di leggere i dati prodotti con il nuovo schema utilizzando la versione precedente.

Se si dispone di una versione dello schema proposta che elimina il campo obbligatorio del nome, con la compatibilità BACKWARD questo non verrebbe registrato correttamente. I consumatori della versione precedente non sarebbero in grado di leggere gli schemi proposti in quanto mancherebbe il campo obbligatorio del nome. Tuttavia, se il campo del nome era originariamente facoltativo, il nuovo schema proposto verrebbe registrato correttamente poiché i consumatori potrebbero leggere i dati in base al nuovo schema che non dispone del campo facoltativo del nome.

JSON

Ad esempio, si supponga di avere una versione dello schema definita da nome (facoltativo), cognome (facoltativo), e-mail (facoltativa) e numero di telefono (facoltativo).

Se si dispone di una nuova versione dello schema in cui viene rimossa la proprietà facoltativa del numero di telefono, la registrazione viene eseguita correttamente, a condizione che la nuova versione dello schema non consenta proprietà aggiuntive impostando il campo `additionalProperties` su false. La compatibilità FORWARD richiede ai consumatori di essere in grado di leggere i dati prodotti con il nuovo schema utilizzando la versione precedente.

Se si dispone di una versione dello schema proposta che elimina la proprietà facoltativa del numero di telefono, questa operazione non viene registrata correttamente con la compatibilità FORWARD se la nuova versione dello schema imposta il campo `additionalProperties` su true, vale a dire consentendo qualsiasi proprietà aggiuntiva. I consumer della versione precedente non sarebbero in grado di leggere lo schema proposto, in quanto potrebbero disporre della proprietà numero di telefono in un tipo differente, ad esempio come stringa anziché numero.

PROTOBUF

Ad esempio, supponiamo di avere una versione di uno schema definito da un messaggio `Person` con i campi `first name` (obbligatorio), `last name` (obbligatorio), `email` (facoltativo) sotto la sintassi proto2.

Se si dispone di una nuova versione dello schema che aggiunge un campo obbligatorio, ad esempio `phone number`, la registrazione viene eseguita correttamente. La compatibilità FORWARD richiede ai consumatori di essere in grado di leggere i dati prodotti con il nuovo schema utilizzando la versione precedente.

Se si dispone di una versione dello schema proposta che elimina il campo obbligatorio `first name`, con la compatibilità BACKWARD questo non verrebbe registrato correttamente. I

consumatori della versione precedente non sarebbero in grado di leggere gli schemi proposti in quanto mancherebbe il campo obbligatorio `first name`. Tuttavia, se il campo `first name` era originariamente facoltativo, il nuovo schema proposto verrebbe registrato correttamente poiché i consumer potrebbero leggere i dati in base al nuovo schema che non dispone del campo facoltativo `first name`.

In caso di utilizzo di gRPC, la rimozione di un nuovo servizio RPC o metodo RPC è una modifica compatibile con le versioni future. Ad esempio, supponiamo di avere una versione di uno schema definito da un servizio RPC `MyService` con due metodi RPC `Foo` e `Bar`.

Se la versione successiva dello schema elimina il metodo RPC esistente denominato `Foo`, si registrerà correttamente in base alla compatibilità FORWARD in quanto i consumer possono leggere i dati prodotti con il nuovo schema utilizzando la versione precedente. Se si dispone di una versione dello schema proposta che elimina il campo obbligatorio `Baz`, con la compatibilità FORWARD questo non verrà registrato correttamente. I consumer della versione precedente non sarebbero in grado di leggere gli schemi proposti in quanto mancherebbe il metodo RPC `Baz`.

- FORWARD_ALL: questa scelta di compatibilità consente ai consumer di leggere dati scritti dai produttori di qualsiasi nuovo schema registrato. È possibile utilizzare questa opzione quando è necessario aggiungere campi o eliminare campi facoltativi e verificare la compatibilità con tutte le versioni precedenti dello schema.
- FULL: questa scelta di compatibilità consente consumer di leggere i dati scritti dai produttori utilizzando la versione precedente o successiva dello schema, ma non versioni precedenti o successive. È possibile utilizzare questa opzione per verificare la compatibilità con l'ultima versione dello schema quando si aggiungono o si eliminano campi facoltativi.
- FULL_ALL: questa scelta di compatibilità consente ai consumer di leggere i dati scritti dai produttori utilizzando tutte le versioni precedenti dello schema. È possibile utilizzare questa opzione per verificare la compatibilità con tutte le versioni precedenti dello schema quando si aggiungono o si eliminano campi facoltativi.

Librerie Serde open source

AWS fornisce librerie Serde open source come framework per la serializzazione e la deserializzazione dei dati. La progettazione open source di queste librerie permette alle applicazioni e ai framework open source comuni di supportare queste librerie nei loro progetti.

Per ulteriori dettagli sul funzionamento delle librerie Serde, consulta Funzionamento del registro degli schemi.

Quote del registro degli schemi

Le quote, note anche come limiti in, sono i valori massimi per le risorse AWS, le azioni e gli elementi presenti nell'account. AWS Di seguito sono riportati i limiti flessibili per il registro degli schemi di AWS Glue.

Coppia chiave-valore dei metadati della versione dello schema

Puoi avere fino a 10 coppie chiave-valore per regione. SchemaVersion AWS

È possibile visualizzare o impostare le coppie di metadati chiave-valore utilizzando le API QuerySchemaVersionMetadata azione (Python: query_schema_version_metadata) o PutSchemaVersionMetadata azione (Python: put_schema_version_metadata).

Di seguito sono riportati i limiti rigidi per il registro degli schemi di AWS Glue.

Registri

Puoi avere fino a 100 registri per AWS regione per questo account.

SchemaVersion

Puoi avere fino a 10000 versioni dello schema per AWS regione per questo account.

Ogni nuovo schema crea una nuova versione dello schema, quindi in teoria puoi avere fino a 10000 schemi per account per regione, se ogni schema ha una sola versione.

Payload dello schema

Esiste un limite di dimensioni pari a 170 KB per i payload dello schema.

Funzionamento del registro degli schemi

Questa sezione descrive il funzionamento dei processi di serializzazione e deserializzazione nel registro degli schemi.

1. Registrare uno schema: se lo schema non esiste già nel registro, può essere registrato con un nome uguale al nome della destinazione (ad esempio, test_topic, test_stream, prod_firehose) oppure il produttore può fornire un nome personalizzato per lo schema. I produttori possono anche aggiungere coppie chiave-valore allo schema come metadati, ad esempio fonte: msk_kafka_topic_A, o applicare tag AWS agli schemi al momento della creazione. Una volta

registrato uno schema, il registro degli schemi restituisce l'ID della versione dello schema al serializzatore. Se lo schema esiste ma il serializzatore utilizza una nuova versione non esistente, il registro degli schemi controllerà il riferimento allo schema con una regola di compatibilità per garantire che la nuova versione sia compatibile prima di registrarla.

Esistono due metodi per registrare uno schema: registrazione manuale e registrazione automatica. È possibile registrare uno schema manualmente tramite la console AWS Glue o la CLI/l'SDK.

Quando la registrazione automatica è attivata nelle impostazioni del serializzatore, lo schema verrà registrato automaticamente. Se `REGISTRY_NAME` non viene fornito nelle configurazioni del produttore, la registrazione automatica registrerà la nuova versione dello schema nel registro predefinito (default-registry). Consulta Installazione delle SerDe librerie per informazioni su come specificare la proprietà di registrazione automatica.

2. Il serializzatore convalida i record di dati rispetto allo schema: quando l'applicazione che produce i dati ha registrato il proprio schema, il serializzatore del registro degli schemi convalida il record prodotto dall'applicazione, strutturato con i campi e i tipi di dati corrispondenti a uno schema registrato. Se lo schema del record non corrisponde a uno schema registrato, il serializzatore restituirà un'eccezione e l'applicazione non riuscirà a consegnare il record alla destinazione.

   Se non esiste uno schema e se il nome dello schema non viene fornito tramite le configurazioni del produttore, lo schema viene creato con lo stesso nome dell'argomento (se Apache Kafka o Amazon MSK) o del flusso (se Kinesis Data Streams).

   Ogni record comprende dati e una definizione dello schema. Viene eseguita una query sulla definizione dello schema in base agli schemi e alle versioni esistenti nel registro degli schemi.

   Per impostazione predefinita, i produttori memorizzano nella cache le definizioni dello schema e gli ID della versione dello schema degli schemi registrati. Se la definizione della versione dello schema di un record non corrisponde a ciò che è disponibile nella cache, il produttore tenterà di convalidare lo schema con il registro degli schemi. Se la versione dello schema è valida, l'ID della versione e la definizione verranno memorizzati nella cache locale del produttore.

   È possibile regolare il periodo di cache di default (24 ore) all'interno delle proprietà del produttore facoltative nel passaggio 3 di Installazione delle SerDe librerie.

3. Serializza e distribuisci i record: se il record è conforme allo schema, il serializzatore aggiunge a ogni record l'ID della versione dello schema, serializza il record in base al formato di dati selezionato (AVRO, JSON o Protobuf e altri formati disponibili a breve), comprime il record (configurazione opzionale del produttore) e lo consegna alla destinazione.

4. I consumer deserializzano i dati: i consumer che leggono questi dati utilizzano la libreria del deserializzatore del registro degli schemi che analizza l'ID della versione dello schema dal payload del record.
5. Il deserializzatore può richiedere lo schema dal registro degli schemi: se il deserializzatore vede i record con un particolare ID della versione dello schema per la prima volta, richiederà lo schema dal registro degli schemi utilizzando questo ID e memorizzerà lo schema nella cache locale del consumer. Se il registro degli schemi non è in grado di deserializzare il record, il consumer può registrare i dati dal record e passare oltre o arrestare l'applicazione.
6. Il deserializzatore utilizza lo schema per deserializzare il record: quando il deserializzatore recupera l'ID della versione dello schema dal registro degli schemi, decomprime il record (se il record inviato dal produttore è compresso) e utilizza lo schema per deserializzarlo. L'applicazione elabora il record.

> **Note**
>
> Crittografia: i client comunicano con il registro degli schemi tramite chiamate API che crittografano i dati in transito utilizzando la crittografia TLS su HTTPS. Gli schemi archiviati nel registro degli schemi vengono sempre crittografati in modo inattivo utilizzando una chiave AWS Key Management Service (AWS KMS).

> **Note**
>
> Autorizzazione utente: il registro degli schemi supporta le policy IAM basati sull'identità.

Guida introduttiva al registro degli schemi

Le sezioni seguenti offrono una panoramica e illustrano la configurazione e l'uso del registro degli schemi. Per informazioni su concetti e componenti del registro degli schemi, consulta Registro dello schema di AWS Glue.

Argomenti

- Installazione delle SerDe librerie
- Utilizzo di AWS CLI per le api del registro degli schemi di AWS Glue
- Creazione di un registro

- Utilizzo di un record specifico (JAVA POJO) per JSON
- Creazione di uno schema
- Aggiornamento di uno schema o di un registro
- Eliminazione di uno schema o di un registro
- Esempi di IAM per i serializzatori
- Esempi di IAM per i deserializzatori
- Connettività privata utilizzando AWS PrivateLink
- Accesso ai CloudWatch parametri di Amazon
- Esempio di modello AWS CloudFormation per il registro degli schemi

Installazione delle SerDe librerie

> **Note**
>
> Prerequisiti: prima di completare i passaggi riportati di seguito, dovrai disporre di un cluster in esecuzione Amazon Managed Streaming for Apache Kafka (Amazon MSK) o Apache Kafka. I produttori e i consumer devono essere in esecuzione su Java 8 o versione successiva.

Le SerDe librerie forniscono un framework per la serializzazione e la deserializzazione dei dati.

Installerai il serializzatore open source per le applicazioni che producono dati (collettivamente i "serializzatori"). Il serializzatore gestisce la serializzazione, la compressione e l'interazione con il registro degli schemi. Il serializzatore estrae automaticamente lo schema da un record in fase di scrittura in una destinazione compatibile con il registro degli schemi, ad esempio Amazon MSK. Allo stesso modo, installerai il deserializzatore open source sulle applicazioni che consumano dati.

Per installare le librerie su produttori e consumer:

1. All'interno dei file pom.xml dei produttori e dei consumer, aggiungi questa dipendenza tramite il codice qui sotto:

```
<dependency>
    <groupId>software.amazon.glue</groupId>
    <artifactId>schema-registry-serde</artifactId>
    <version>1.1.5</version>
</dependency>
```

In alternativa, è possibile clonare il repository Github del registro degli schemi di AWS Glue.

2. Imposta i tuoi produttori con le seguenti proprietà obbligatorie:

```
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
 StringSerializer.class.getName()); // Can replace StringSerializer.class.getName())
 with any other key serializer that you may use
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
 GlueSchemaRegistryKafkaSerializer.class.getName());
props.put(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2");
properties.put(AWSSchemaRegistryConstants.DATA_FORMAT, "JSON"); // OR "AVRO"
```

Se non esistono schemi, è necessario attivare la registrazione automatica (passaggio successivo). Se si dispone di uno schema da applicare, sostituire "my-schema" con il nome dello schema. Se la registrazione automatica dello schema è disattivata deve essere fornito anche "registry-name". Se lo schema viene creato sotto "default-registry", il nome del registro può essere omesso.

3. (Facoltativo) Impostare una di queste proprietà facoltative del produttore. Per descrizioni dettagliate delle proprietà, consultate il file. ReadMe

```
props.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, "true"); // If
 not passed, uses "false"
props.put(AWSSchemaRegistryConstants.SCHEMA_NAME, "my-schema"); // If not passed,
 uses transport name (topic name in case of Kafka, or stream name in case of Kinesis
 Data Streams)
props.put(AWSSchemaRegistryConstants.REGISTRY_NAME, "my-registry"); // If not passed,
 uses "default-registry"
props.put(AWSSchemaRegistryConstants.CACHE_TIME_TO_LIVE_MILLIS, "86400000"); // If
 not passed, uses 86400000 (24 Hours)
props.put(AWSSchemaRegistryConstants.CACHE_SIZE, "10"); // default value is 200
props.put(AWSSchemaRegistryConstants.COMPATIBILITY_SETTING, Compatibility.FULL); //
 Pass a compatibility mode. If not passed, uses Compatibility.BACKWARD
props.put(AWSSchemaRegistryConstants.DESCRIPTION, "This registry is used for several
 purposes."); // If not passed, constructs a description
props.put(AWSSchemaRegistryConstants.COMPRESSION_TYPE,
 AWSSchemaRegistryConstants.COMPRESSION.ZLIB); // If not passed, records are sent
 uncompressed
```

La registrazione automatica registra la versione dello schema nel registro di default ("default-registry"). Se nel passaggio precedente non è stato specificato un `SCHEMA_NAME`, il nome dell'argomento viene dedotto come `SCHEMA_NAME`.

Per ulteriori informazioni sulle modalità di compatibilità, consulta Controllo delle versioni e compatibilità degli schemi.

4. Imposta i tuoi consumer con le seguenti proprietà obbligatorie:

```
props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 StringDeserializer.class.getName());
props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 GlueSchemaRegistryKafkaDeserializer.class.getName());
props.put(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2"); // Pass an Regione AWS
props.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName()); // Only required for AVRO data format
```

5. (Facoltativo) Imposta queste proprietà facoltative del consumer. Per descrizioni dettagliate delle proprietà, consultate il ReadMe file.

```
properties.put(AWSSchemaRegistryConstants.CACHE_TIME_TO_LIVE_MILLIS, "86400000"); //
 If not passed, uses 86400000
props.put(AWSSchemaRegistryConstants.CACHE_SIZE, "10"); // default value is 200
props.put(AWSSchemaRegistryConstants.SECONDARY_DESERIALIZER,
 "com.amazonaws.services.schemaregistry.deserializers.external.ThirdPartyDeserializer"); //
 For migration fall back scenario
```

Utilizzo di AWS CLI per le api del registro degli schemi di AWS Glue

Per utilizzare AWS CLI per le API del registro degli schemi di AWS Glue, assicurati di aggiornare AWS CLI alla versione più recente.

Creazione di un registro

È possibile utilizzare il registro di default o creare tutti i nuovi registri necessari utilizzando le API AWS Glue o la console AWS Glue.

API AWS Glue

È possibile utilizzare questi passaggi per eseguire questa attività utilizzando le API AWS Glue.

Per aggiungere un nuovo registro, utilizza l'API CreateRegistry azione (Python: create_registry). Specifica `RegistryName` come nome del registro da creare, con una lunghezza massima di 255 caratteri e può contenere solo lettere, numeri, trattini, trattini bassi, simboli del dollaro o cancelletti.

Specificate a `Description` come stringa di lunghezza non superiore a 2048 byte, corrispondente allo schema di stringa multilinea dell'indirizzo URI.

Facoltativamente, puoi specificare uno o più `Tags` per il registro, come una matrice di mappe di coppie chiave-valore.

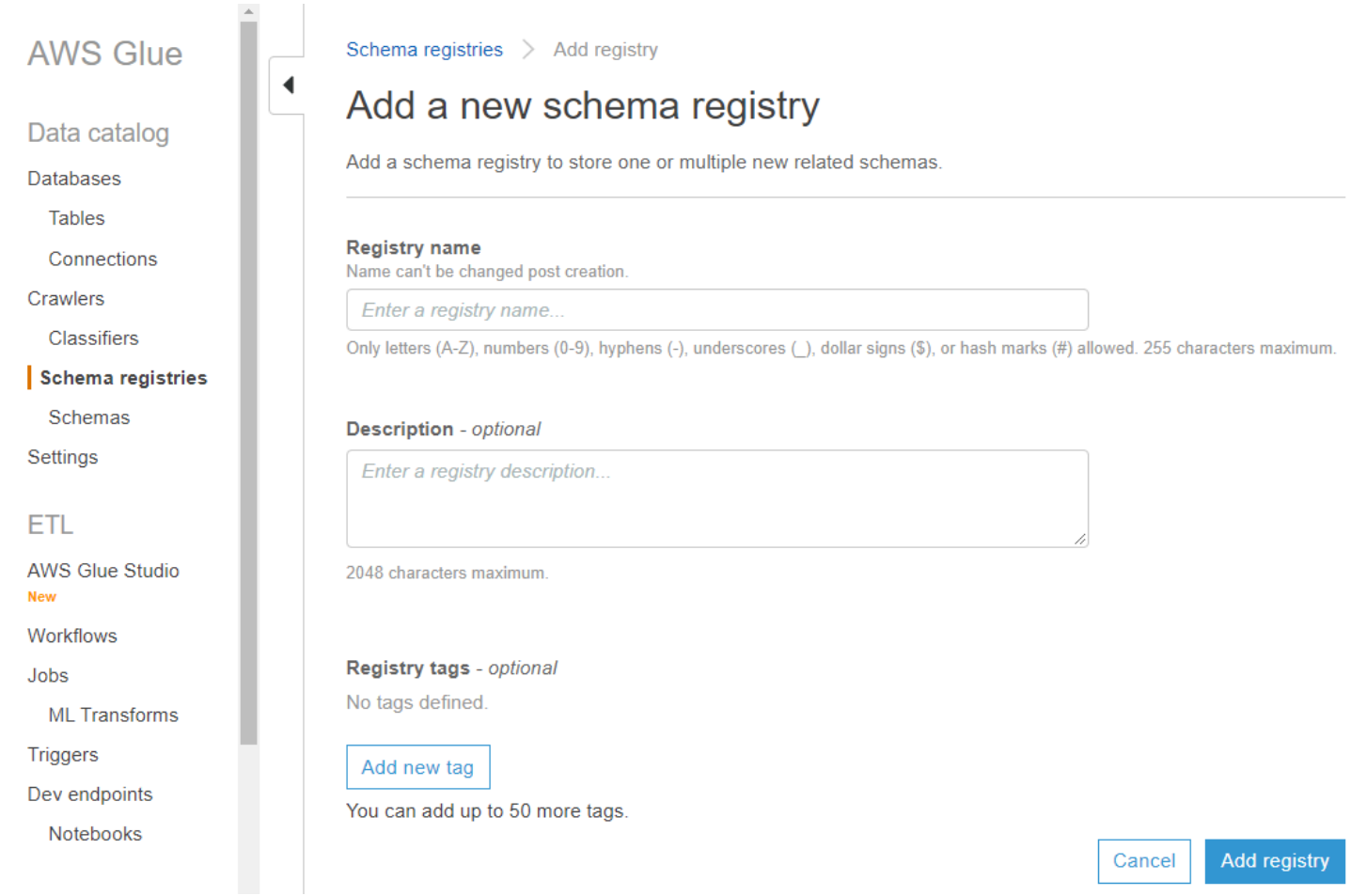
```
aws glue create-registry --registry-name registryName1 --description description
```

Al momento della creazione del registro, gli viene assegnato un Amazon Resource Name (ARN) che puoi visualizzare nel `RegistryArn` della risposta API. Dopo aver creato un registro, crea uno o più schemi per questo registro.

Console AWS Glue

Per aggiungere un nuovo registro nella console AWS Glue:

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Schema registries (Registri degli schemi).
3. Scegli Add registry (Aggiungi registro).
4. Inserisci un Registry name (Nome registro) per il registro, composto da lettere, numeri, trattini o trattini bassi. Questo nome non può essere modificato.
5. Inserisci una Description (Descrizione) (facoltativo) per il registro.
6. Facoltativamente, applica uno o più tag al registro. Seleziona Add new tag (Aggiungi nuovo tag) e immetti una Tag key (Chiave di tag) e facoltativamente un Tag value (Valore di tag).
7. Scegli Add registry (Aggiungi registro).



Al momento della creazione del registro, gli viene assegnato un Amazon Resource Name (ARN) che puoi visualizzare selezionando il registro dall'elenco in Schema registries (Registri degli schemi). Dopo aver creato un registro, crea uno o più schemi per questo registro.

Utilizzo di un record specifico (JAVA POJO) per JSON

È possibile utilizzare un POJO (Plain Old Java Object) e passare l'oggetto come record. È simile alla nozione di un record specifico in AVRO. mbknor-jackson-jsonschemaPuò generare uno schema JSON per il POJO passato. Questa libreria può anche inserire informazioni aggiuntive nello schema JSON.

La libreria del registro degli schemi di AWS Glue utilizza il campo "className" inserito nello schema per fornire un nome di classe completamente classificato. Il campo "className" viene utilizzato dal deserializzatore per eseguire la deserializzazione in un oggetto di quella classe.

```
 Example class :

@JsonSchemaDescription("This is a car")
```

```
@JsonSchemaTitle("Simple Car Schema")
@Builder
@AllArgsConstructor
@EqualsAndHashCode
// Fully qualified class name to be added to an additionally injected property
// called className for deserializer to determine which class to deserialize
// the bytes into
@JsonSchemaInject(
        strings = {@JsonSchemaString(path = "className",
                value =
 "com.amazonaws.services.schemaregistry.integrationtests.generators.Car")}
)
// List of annotations to help infer JSON Schema are defined by https://github.com/
mbknor/mbknor-jackson-jsonSchema
public class Car {
    @JsonProperty(required = true)
    private String make;

    @JsonProperty(required = true)
    private String model;

    @JsonSchemaDefault("true")
    @JsonProperty
    public boolean used;

    @JsonSchemaInject(ints = {@JsonSchemaInt(path = "multipleOf", value = 1000)})
    @Max(200000)
    @JsonProperty
    private int miles;

    @Min(2000)
    @JsonProperty
    private int year;

    @JsonProperty
    private Date purchaseDate;

    @JsonProperty
    @JsonFormat(shape = JsonFormat.Shape.NUMBER)
    private Date listedDate;

    @JsonProperty
    private String[] owners;
```

```
    @JsonProperty
    private Collection<Float> serviceChecks;

    // Empty constructor is required by Jackson to deserialize bytes
    // into an Object of this class
    public Car() {}
}
```

Creazione di uno schema

Puoi creare uno schema utilizzando le API AWS Glue o la console AWS Glue.

API AWS Glue

È possibile utilizzare questi passaggi per eseguire questa attività utilizzando le API AWS Glue.

Per aggiungere un nuovo schema, utilizza l'API CreateSchema azione (Python: create_schema).

Specifica una struttura `RegistryId` per indicare un registro per lo schema. Oppure, ometti il `RegistryId` per utilizzare il registro di default.

Specifica un `SchemaName` composto da lettere, numeri, trattini o trattini bassi e `DataFormat` come **`AVRO`** o **`JSON`**. Una volta impostato su uno schema, `DataFormat` non è modificabile.

Specifica una modalità di `Compatibility`:

- Backward (consigliato): il consumer può leggere sia la versione attuale che quella precedente.
- Backward all: il consumer può leggere la versione attuale e tutte quelle precedenti.
- Forward: il consumer può leggere sia la versione attuale che quella successiva.
- Forward all: il consumer può leggere sia la versione attuale che tutte quelle successive.
- Full: combinazione di Backward e Forward.
- Full all: combinazione di Backward all e Forward all.
- None: non vengono eseguiti controlli di compatibilità.
- Disabled: impedisce il controllo delle versioni per questo schema.

Facoltativamente, specifica i `Tags` per lo schema.

Specifica un valore di `SchemaDefinition` per definire lo schema in formato dati Avro, JSON o Protobuf. Consulta questi esempi.

Per il formato dati Avro:

```
aws glue create-schema --registry-id RegistryName="registryName1" --schema-name
 testschema --compatibility NONE --data-format AVRO --schema-definition "{\"type\":
 \"record\", \"name\": \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"},
 {\"name\": \"f2\", \"type\": \"string\"} ]}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName1" --schema-name testschema --compatibility
 NONE --data-format AVRO  --schema-definition "{\"type\": \"record\", \"name\": \"r1\",
 \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type\":
 \"string\"} ]}"
```

Per il formato dati JSON:

```
aws glue create-schema --registry-id RegistryName="registryName" --schema-name
 testSchemaJson --compatibility NONE --data-format JSON --schema-definition "{\"$schema
\": \"http://json-schema.org/draft-07/schema#\",\"type\":\"object\",\"properties\":
{\"f1\":{\"type\":\"string\"}}}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName" --schema-name testSchemaJson --compatibility
 NONE --data-format JSON --schema-definition "{\"$schema\": \"http://json-schema.org/
draft-07/schema#\",\"type\":\"object\",\"properties\":{\"f1\":{\"type\":\"string\"}}}"
```

Per il formato dati Protobuf:

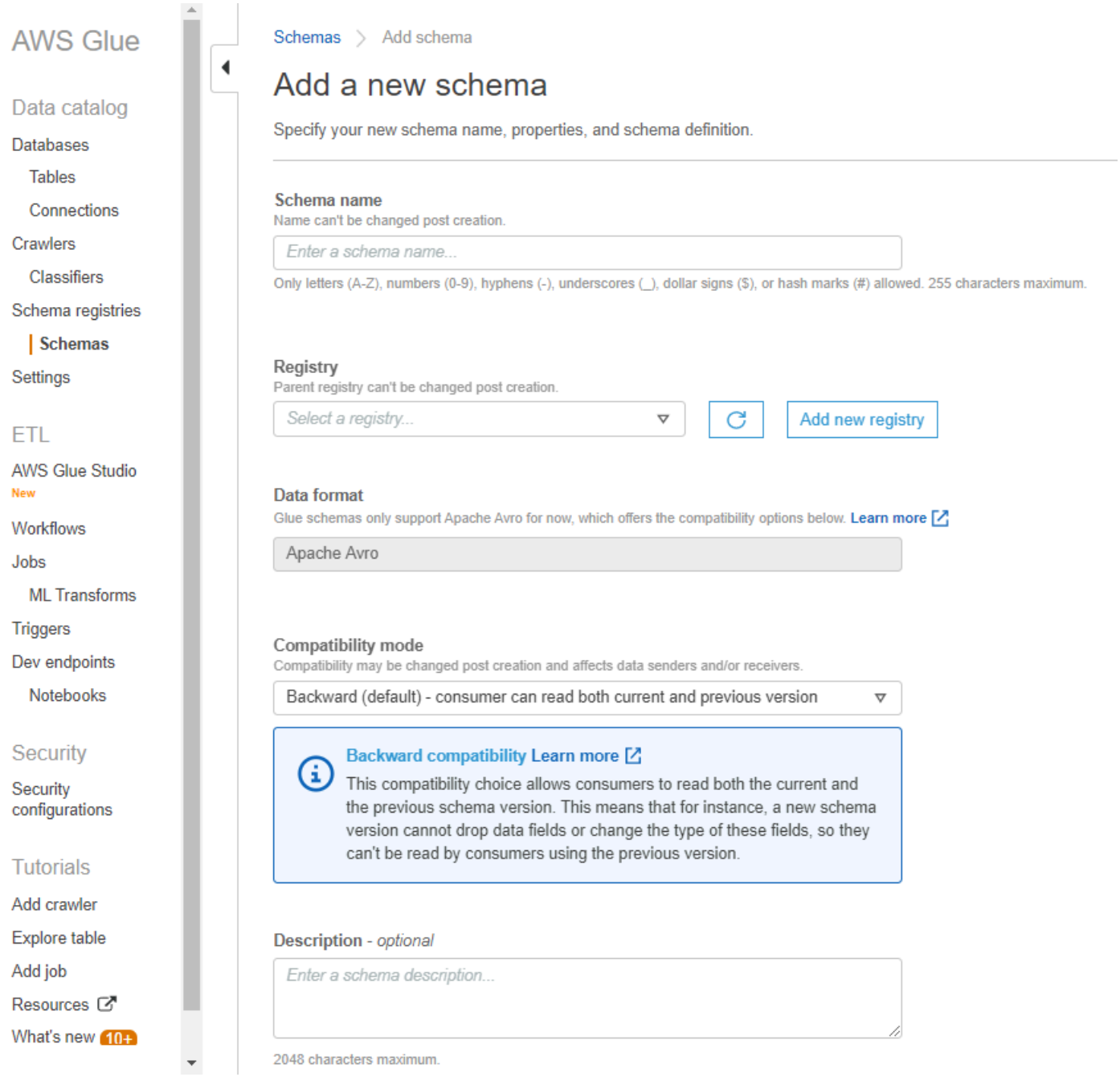
```
aws glue create-schema --registry-id RegistryName="registryName" --schema-name
 testSchemaProtobuf --compatibility NONE --data-format PROTOBUF --schema-definition
 "syntax = \"proto2\";package org.test;message Basic { optional int32 basic = 1;}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName" --schema-name testSchemaProtobuf
 --compatibility NONE --data-format PROTOBUF --schema-definition "syntax =
 \"proto2\";package org.test;message Basic { optional int32 basic = 1;}"
```

Console AWS Glue

Per aggiungere un nuovo schema utilizzando la console AWS Glue:

1. Accedi alla Console di gestione AWS e apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Schema (Schemi).
3. Seleziona Aggiungi schema (Aggiungi schema).
4. Inserisci uno Schema name (Nome schema) composto da lettere, numeri, trattini, trattini bassi, simboli di dollaro o cancelletti. Questo nome non può essere modificato.
5. Seleziona il registro in cui lo schema verrà archiviato dal menu a discesa. Il registro padre non può essere modificato dopo la creazione.
6. Lascia Data format (Formato dei dati) come Apache Avro o JSON. Questo formato si applica a tutte le versioni di questo schema.
7. Scegli una Compatibility mode (Modalità Compatibilità).
   - Backward (consigliato): il ricevitore può leggere sia la versione attuale che quella precedente.
   - Backward all: il ricevitore può leggere la versione attuale e tutte quelle precedenti.
   - Forward: il mittente può scrivere sia la versione attuale che quelle precedenti.
   - Forward All: il mittente può scrivere sia la versione attuale che tutte quelle precedenti.
   - Full: combinazione di Backward e Forward.
   - Full All: combinazione di Backward All e Forward All.
   - None: non vengono eseguiti controlli di compatibilità.
   - Disabled: impedisce il controllo delle versioni per questo schema.
8. Immetti un a Description (Descrizione) facoltativa per il registro con un massimo di 250 caratteri.

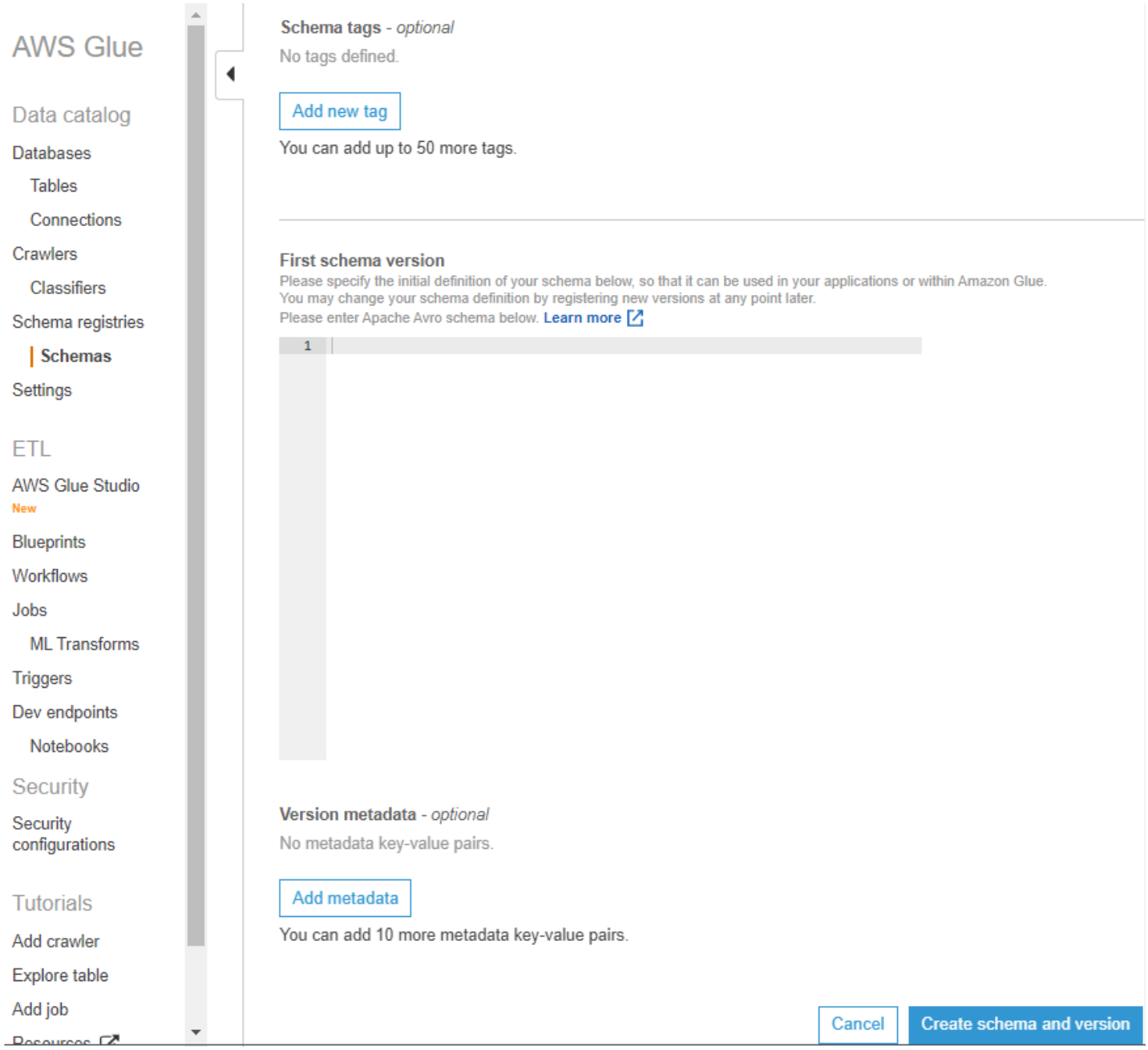


9. Facoltativamente, applica uno o più tag allo schema. Seleziona Add new tag (Aggiungi nuovo tag) e immetti una Tag key (Chiave di tag) e facoltativamente un Tag value (Valore di tag).
10. Immetti o incolla lo schema iniziale nella casella First schema version (Prima versione dello schema).

    Per il formato Avro, consulta Utilizzare il formato di dati Avro

    Per il formato JSON, consulta Utilizzare il formato di dati JSON

11. Facoltativamente, scegli Add metadata (Aggiungi metadata) per aggiungere metadati di versione per annotare o classificare la versione dello schema.
12. Scegli Create schema and version (Crea schema e versione).



Lo schema viene creato e viene visualizzato nell'elenco sotto Schemas (Schemi).

Utilizzare il formato di dati Avro

Avro fornisce servizi di serializzazione dei dati e scambio di dati. Avro memorizza la definizione dei dati in formato JSON semplificando la lettura e l'interpretazione. I dati stessi sono memorizzati in formato binario.

Per informazioni sulla definizione di uno schema Apache Avro, consulta la specifica di Apache Avro.

Utilizzare il formato di dati JSON

I dati possono essere serializzati con il formato JSON. Il formato di schemi JSON definisce lo standard per il formato di schemi JSON.

Aggiornamento di uno schema o di un registro

Una volta creati, è possibile modificare gli schemi, le versioni degli schemi o il registro.

Aggiornamento di un registro

È possibile aggiornare un registro utilizzando l'API AWS Glue o la console AWS Glue. Il nome di un registro esistente non può essere modificato. È possibile modificare la descrizione di un registro.

API AWS Glue

Per aggiornare un registro esistente, utilizza l'API UpdateRegistry azione (Python: update_registry).

Specifica una struttura `RegistryId` per indicare il registro da aggiornare. Passa una `Description` per modificare la descrizione di un registro.

```
aws glue update-registry --description updatedDescription --registry-id
 RegistryArn="arn:aws:glue:us-east-2:901234567890:registry/registryName1"
```

Console AWS Glue

Per aggiornare un registro utilizzando la console AWS Glue:

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Schema registries (Registri degli schemi).
3. Scegli un registro dall'elenco dei registri selezionando la relativa casella.
4. Dal menu Action (Operazioni), seleziona Edit registry (Modifica registro).

Aggiornamento di uno schema

È possibile aggiornare la descrizione o l'impostazione di compatibilità per uno schema.

Per aggiornare uno schema esistente, utilizza l'API UpdateSchema azione (Python: update_schema).

Specifica una struttura `SchemaId` per indicare lo schema da aggiornare. Deve essere fornito `VersionNumber` o `Compatibility`.

Esempio di codice 11:

```
aws glue update-schema --description testDescription --schema-id
 SchemaName="testSchema1",RegistryName="registryName1" --schema-version-number
 LatestVersion=true --compatibility NONE
```

```
aws glue update-schema --description testDescription --schema-id
 SchemaArn="arn:aws:glue:us-east-2:901234567890:schema/registryName1/testSchema1" --
schema-version-number LatestVersion=true --compatibility NONE
```

Aggiunta di una versione dello schema

Quando si aggiunge una versione dello schema, è necessario confrontare le versioni per assicurarsi che il nuovo schema venga accettato.

Per aggiungere una nuova versione a uno schema esistente, utilizza l'API RegisterSchemaVersion azione (Python: register_schema_version).

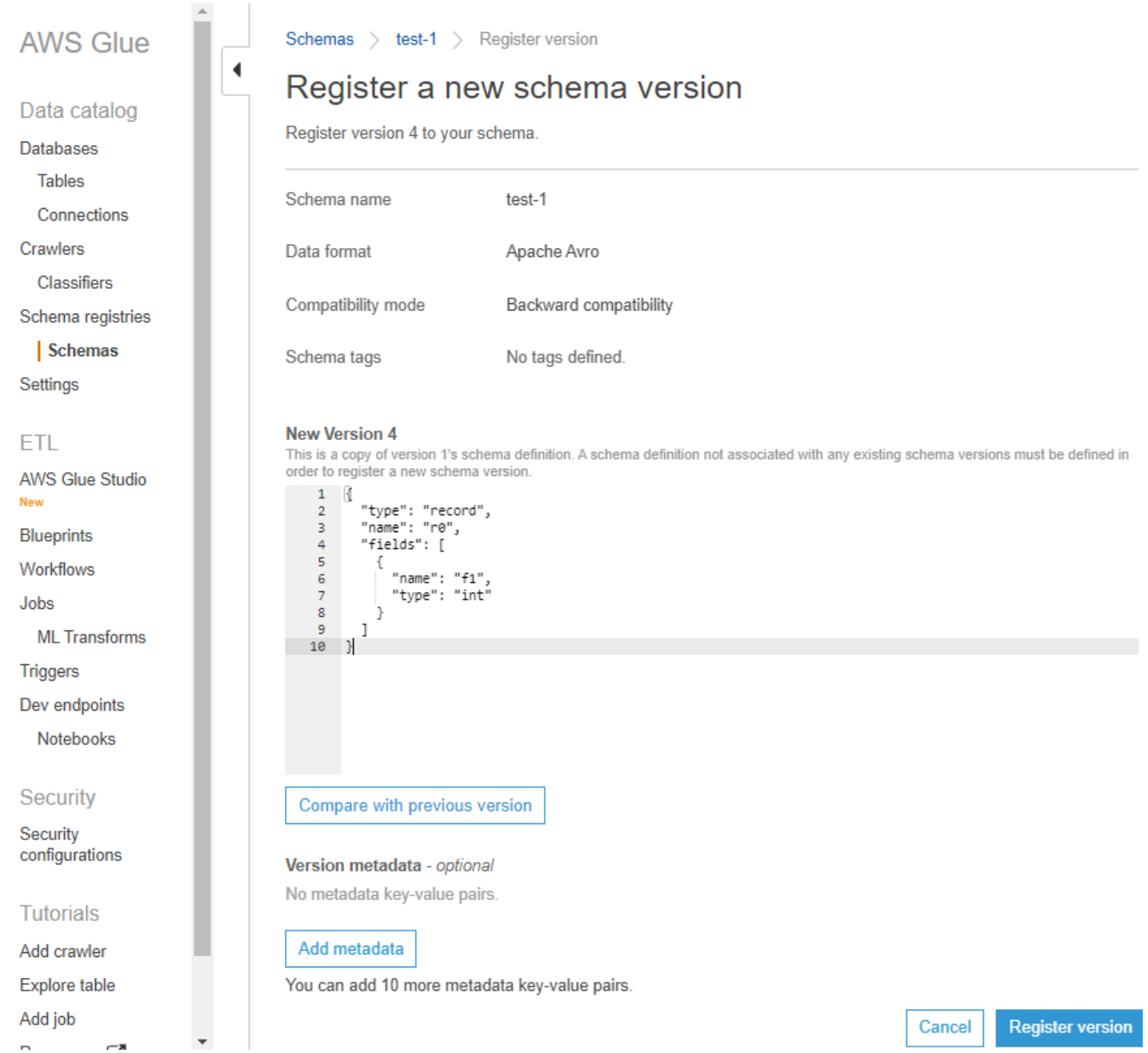
Specifica una struttura `SchemaId` per indicare lo schema per il quale si desidera aggiungere una versione e un valore di `SchemaDefinition` per definire lo schema.

Esempio di codice 12:

```
aws glue register-schema-version --schema-definition "{\"type\": \"record\", \"name\":
 \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type
\": \"string\"} ]}" --schema-id SchemaArn="arn:aws:glue:us-east-1:901234567890:schema/
registryName/testschema"
```

```
aws glue register-schema-version --schema-definition "{\"type\": \"record\", \"name\":
 \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type
\": \"string\"} ]}" --schema-id SchemaName="testschema",RegistryName="testregistry"
```

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Schema (Schemi).
3. Scegli lo schema dall'elenco degli schemi selezionando la relativa casella.
4. Seleziona uno o più schemi dall'elenco selezionando le caselle.
5. Nel menu Action (Operazioni), seleziona Register new version (Registra nuova versione).
6. Nella casella New version (Nuova versione), immetti o incolla il nuovo schema.
7. Seleziona Compare with previous version (Confronta con la versione precedente) per visualizzare le differenze con la versione precedente dello schema.
8. Facoltativamente, scegli Add metadata (Aggiungi metadata) per aggiungere metadati di versione per annotare o classificare la versione dello schema. Inserisci Key (Chiave) e facoltativamente Value (Valore).
9. Scegli Register version (Registra versione).



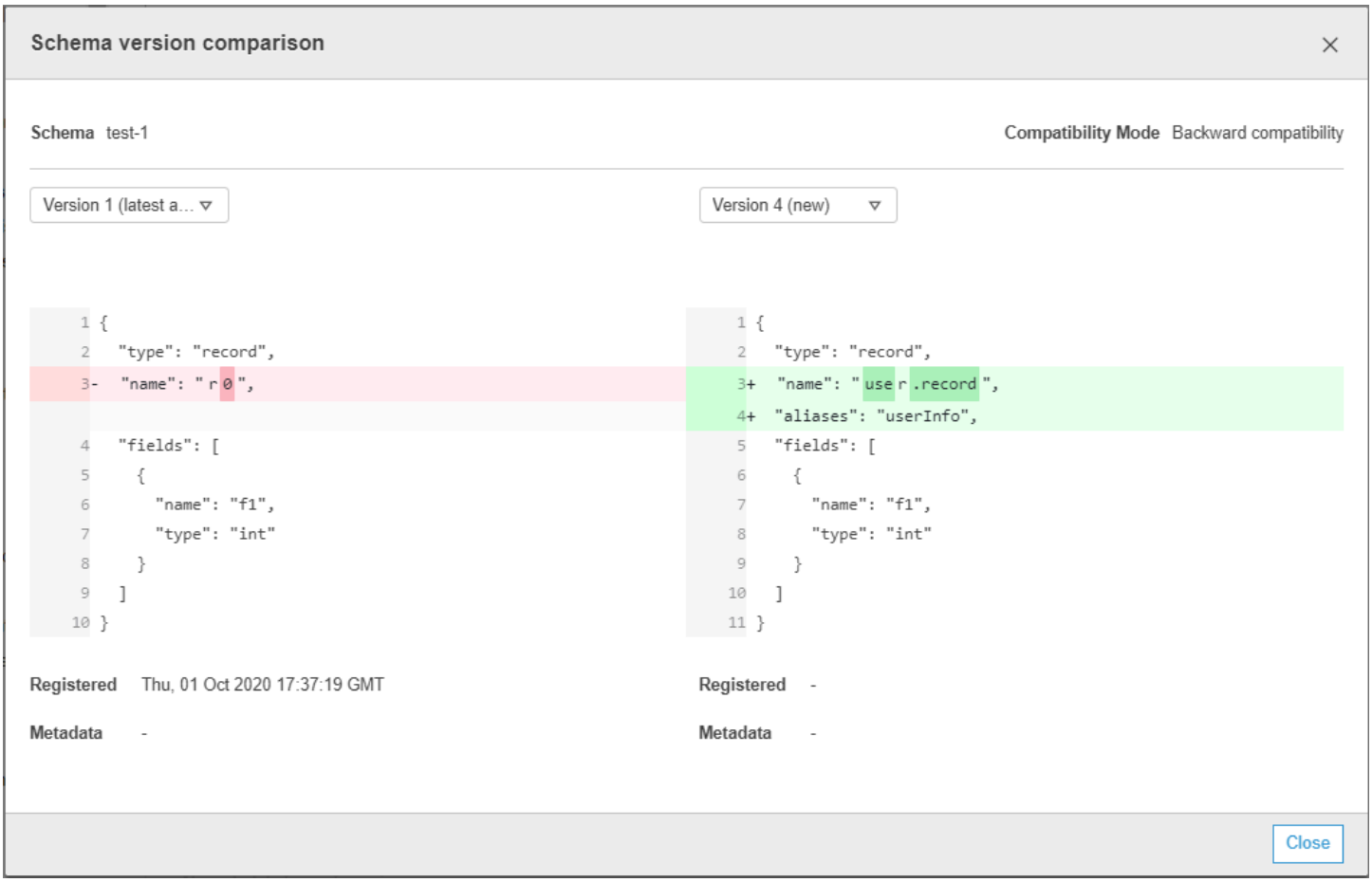
La versione degli schemi viene visualizzata nell'elenco delle versioni. Se la versione ha modificato la modalità di compatibilità, la versione verrà contrassegnata come checkpoint.

Esempio di confronto tra le versioni di uno schema

Selezionando Compare with previous version (Confronta con la versione precedente), le versioni precedenti e quelle nuove verranno mostrate insieme. Le informazioni modificate saranno evidenziate come segue:

- Giallo: indica le informazioni modificate.
- Verde: indica il contenuto aggiunto nella versione più recente.
- Rosso: indica il contenuto rimosso nella versione più recente.

È possibile eseguire il confronto anche con le versioni precedenti.



Eliminazione di uno schema o di un registro

L'eliminazione di uno schema, di una versione dello schema o di un registro è un'azione permanente che non può essere annullata.

Eliminazione di uno schema

Puoi eliminare uno schema quando non questo verrà più utilizzato all'interno di un registro, utilizzando la AWS Management Console o l'API DeleteSchema azione (Python: delete_schema).

L'eliminazione di uno o più schemi è un'azione permanente che non può essere annullata. Accertati che lo schema o gli schemi non siano più necessari.

Per eliminare uno schema dal registro, chiama l'API DeleteSchema azione (Python: delete_schema), specificando la struttura `SchemaId` per identificare lo schema.

Ad esempio:

```
aws glue delete-schema --schema-id SchemaArn="arn:aws:glue:us-
east-2:901234567890:schema/registryName1/schemaname"
```

```
aws glue delete-schema --schema-id SchemaName="TestSchema6-
deleteschemabyname",RegistryName="default-registry"
```

Console AWS Glue

Per eliminare uno schema dalla console AWS Glue:

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Schema registries (Registri degli schemi).
3. Scegli il registro che contiene lo schema dall'elenco dei registri.
4. Seleziona uno o più schemi dall'elenco selezionando le caselle.
5. Dal menu Action (Operazioni), scegli Delete schema (Elimina schema).
6. Inserisci il testo **`Delete`** nel campo per confermare l'eliminazione.
7. Seleziona Delete (Elimina).

Gli schemi specificati vengono eliminati dal registro.

Eliminazione di una versione dello schema

Man mano che gli schemi si accumulano nel registro, puoi eliminare le versioni indesiderate utilizzando la AWS Management Console o l'API DeleteSchemaVersions azione (Python: delete_schema_versions). L'eliminazione di una o più versioni degli schemi è un'azione permanente che non può essere annullata. Accertati che le versioni degli schemi non siano più necessarie.

Durante l'eliminazione delle versioni degli schemi, tieni presente i seguenti vincoli:

- Non è possibile eliminare una versione con segno di spunta.

- L'intervallo di versioni contigue non può essere superiore a 25.
- La versione più recente non deve trovarsi nello stato in sospeso.

Specifica la struttura `SchemaId` per identificare lo schema e specifica `Versions` come intervallo di versioni da eliminare. Per ulteriori informazioni sulla specifica di una versione o un intervallo di versioni, consulta DeleteRegistry azione (Python: delete_registry). Le versioni degli schemi specificate vengono eliminate dal registro.

Chiamare l'API ListSchemaVersions azione (Python: list_schema_versions) dopo questa chiamata elencherà lo stato delle versioni eliminate.

Ad esempio:

```
aws glue delete-schema-versions --schema-id
 SchemaName="TestSchema6",RegistryName="default-registry" --versions "1-1"
```

```
aws glue delete-schema-versions --schema-id SchemaArn="arn:aws:glue:us-
east-2:901234567890:schema/default-registry/TestSchema6-NON-Existent" --versions "1-1"
```

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Schema registries (Registri degli schemi).
3. Scegli il registro che contiene lo schema dall'elenco dei registri.
4. Seleziona uno o più schemi dall'elenco selezionando le caselle.
5. Dal menu Action (Operazioni), scegli Delete schema (Elimina schema).
6. Inserisci il testo **`Delete`** nel campo per confermare l'eliminazione.
7. Seleziona Delete (Elimina).

Le versioni degli schemi specificate vengono eliminate dal registro.

Eliminazione di un registro

È possibile eliminare un registro quando gli schemi in esso contenuti non devono più essere organizzati in tale registro. Sarà necessario riassegnare tali schemi a un altro registro.

L'eliminazione di uno o più registri è un'azione permanente che non può essere annullata. Accertati che il registro o i registri non siano più necessari.

Il registro predefinito può essere eliminato utilizzando AWS CLI.

API AWS Glue

Per eliminare l'intero registro, inclusi gli schemi e tutte le relative versioni, chiama l'API DeleteRegistry azione (Python: delete_registry). Specifica una struttura `RegistryId` per identificare lo schema.

Ad esempio:

```
aws glue delete-registry --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName1"
```

```
aws glue delete-registry --registry-id RegistryName="TestRegistry-deletebyname"
```

Per ottenere lo stato dell'operazione di eliminazione, è possibile chiamare l'API `GetRegistry` dopo la chiamata asincrona.

Console AWS Glue

Per eliminare un registro dalla console AWS Glue:

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Schema registries (Registri degli schemi).
3. Scegli un registro dall'elenco selezionando una casella.
4. Dal menu Action (Operazioni), seleziona Delete registry (Elimina registro).
5. Inserisci il testo **`Delete`** nel campo per confermare l'eliminazione.
6. Seleziona Delete (Elimina).

I registri selezionati vengono eliminati da AWS Glue.

Esempi di IAM per i serializzatori

> **Note**
>
> Le policy gestite da AWS concedono le autorizzazioni necessarie per casi di utilizzo comuni. Per informazioni sull'utilizzo delle policy gestite per gestire il registro degli schemi, consulta Policy gestite da AWS (predefinite) per AWS Glue.

Per i serializzatori, è necessario creare una policy minima simile a quella riportata di seguito per avere la possibilità di trovare lo `schemaVersionId` per una determinata definizione dello schema. Nota, per leggere gli schemi nel registro è necessario disporre delle autorizzazioni di lettura. È possibile limitare i registri che possono essere letti utilizzando la clausola `Resource`.

Esempio di codice 13:

```
{
    "Sid" : "GetSchemaByDefinition",
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaByDefinition"
    ],
        "Resource" : ["arn:aws:glue:us-east-2:012345678:registry/registryname-1",
                      "arn:aws:glue:us-east-2:012345678:schema/registryname-1/
schemaname-1",
                      "arn:aws:glue:us-east-2:012345678:schema/registryname-1/
schemaname-2"
                     ]
}
```

Inoltre, è possibile consentire ai produttori di creare nuovi schemi e versioni includendo i seguenti metodi aggiuntivi. Nota, devi essere in grado di ispezionare il registro per aggiungere/rimuovere/modificare gli schemi al suo interno. È possibile limitare i registri che possono essere ispezionati utilizzando la clausola `Resource`.

Esempio di codice 14:

```
{
    "Sid" : "RegisterSchemaWithMetadata",
```

```
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaByDefinition",
        "glue:CreateSchema",
        "glue:RegisterSchemaVersion",
        "glue:PutSchemaVersionMetadata",
    ],
    "Resource" : ["arn:aws:glue:aws-region:123456789012:registry/registryname-1",
                  "arn:aws:glue:aws-region:123456789012:schema/registryname-1/
schemaname-1",
                  "arn:aws:glue:aws-region:123456789012:schema/registryname-1/
schemaname-2"
                ]
}
```

Esempi di IAM per i deserializzatori

Per i deserializzatori (lato consumer), è necessario creare una policy simile a quella riportata di seguito per consentire al deserializzatore di recuperare lo schema dal registro degli schemi per la deserializzazione. Nota, devi essere in grado di ispezionare il registro per recuperare gli schemi al suo interno.

Esempio di codice 15:

```
{
    "Sid" : "GetSchemaVersion",
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaVersion"
    ],
    "Resource" : ["*"]
}
```

Connettività privata utilizzando AWS PrivateLink

È possibile utilizzare AWS PrivateLink per collegare il VPC del produttore di dati ad AWS Gluedefinendo di un endpoint VPC dell'interfaccia per AWS Glue. Quando utilizzi un endpoint VPC di interfaccia, la comunicazione tra il VPC e AWS Glue avviene completamente all'interno della rete AWS. Per ulteriori informazioni, consulta la pagina relativa all'utilizzo di AWS Glue con endpoint VPC.

Accesso ai CloudWatch parametri di Amazon

CloudWatch I parametri di Amazon sono disponibili come parte CloudWatch del piano gratuito. Puoi accedere a queste metriche nella Console. CloudWatch Le metriche a livello di API includono CreateSchema (Successo e latenza), (Successo e latenza) GetSchemaByDefinition, (Successo e latenza), GetSchemaVersion (Successo e latenza), RegisterSchemaVersion (Successo e latenza). PutSchemaVersionMetadata Le metriche a livello di risorsa includono Registry. ThrottledByLimit, SchemaVersion. ThrottledByLimit, SchemaVersion .Taglia.

Esempio di modello AWS CloudFormation per il registro degli schemi

Di seguito è riportato un modello di esempio per la creazione di risorse del registro degli schemi in AWS CloudFormation. Per creare questo stack nell'account, copia il modello sopra in un file `SampleTemplate.yaml` ed esegui il comando seguente:

```
aws cloudformation create-stack --stack-name ABCSchemaRegistryStack --template-body
 "'cat SampleTemplate.yaml'"
```

Questo esempio usa `AWS::Glue::Registry` per creare un registro, `AWS::Glue::Schema` per creare uno schema, `AWS::Glue::SchemaVersion`per creare una versione dello schema e `AWS::Glue::SchemaVersionMetadata` per popolare i metadati della versione dello schema.

```
Description: "A sample CloudFormation template for creating Schema Registry resources."
Resources:
  ABCRegistry:
    Type: "AWS::Glue::Registry"
    Properties:
      Name: "ABCSchemaRegistry"
      Description: "ABC Corp. Schema Registry"
      Tags:
        - Key: "Project"
          Value: "Foo"
  ABCSchema:
    Type: "AWS::Glue::Schema"
    Properties:
      Registry:
        Arn: !Ref ABCRegistry
      Name: "TestSchema"
      Compatibility: "NONE"
      DataFormat: "AVRO"
      SchemaDefinition: >
```

```
        {"namespace":"foo.avro","type":"record","name":"user","fields":
[{"name":"name","type":"string"},{"name":"favorite_number","type":"int"}]}
      Tags:
        - Key: "Project"
          Value: "Foo"
  SecondSchemaVersion:
    Type: "AWS::Glue::SchemaVersion"
    Properties:
      Schema:
        SchemaArn: !Ref ABCSchema
      SchemaDefinition: >
        {"namespace":"foo.avro","type":"record","name":"user","fields":
[{"name":"status","type":"string", "default":"ON"}, {"name":"name","type":"string"},
{"name":"favorite_number","type":"int"}]}
  FirstSchemaVersionMetadata:
    Type: "AWS::Glue::SchemaVersionMetadata"
    Properties:
      SchemaVersionId: !GetAtt ABCSchema.InitialSchemaVersionId
      Key: "Application"
      Value: "Kinesis"
  SecondSchemaVersionMetadata:
    Type: "AWS::Glue::SchemaVersionMetadata"
    Properties:
      SchemaVersionId: !Ref SecondSchemaVersion
      Key: "Application"
      Value: "Kinesis"
```

Integrazione con il registro degli schemi di AWS Glue

Queste sezioni descrivono le integrazioni con il registro degli schemi di AWS Glue. Gli esempi riportati in questa sezione mostrano uno schema con formato di dati AVRO. Per altri esempi, inclusi schemi con formato dati JSON, consulta i test di integrazione e ReadMe le informazioni nel repository open source di AWS Glue Schema Registry.

Argomenti

- Caso d'uso: connessione del registro degli schemi ad Amazon MSK o Apache Kafka
- Caso d'uso: integrazione del flusso di dati Amazon Kinesis con il registro degli schemi di AWS Glue
- Caso d'uso: Amazon Managed Service per Apache Flink
- Caso d'uso: integrazione con AWS Lambda

- Caso d'uso: AWS Glue Data Catalog
- Caso d'uso: streaming AWS Glue
- Caso d'uso: flussi Apache Kafka
- Caso d'uso: Apache Kafka Connect

Caso d'uso: connessione del registro degli schemi ad Amazon MSK o Apache Kafka

Supponiamo che tu stia scrivendo dati su un argomento Apache Kafka e che possa seguire questi passaggi per iniziare.

1. Crea unAmazon Managed Streaming for Apache Kafka (Amazon MSK) o cluster Apache Kafka con almeno un argomento. Se si crea un cluster Amazon MSK, è possibile utilizzare il AWS Management Console. Segui queste istruzioni: Nozioni di base per l'uso di Amazon MSK nella Guida per gli sviluppatori di Amazon Managed Streaming for Apache Kafka.
2. Segui il passaggio Installazione delle SerDe librerie sopra.
3. Per creare registri dello schema, schemi o versioni, segui le istruzioni riportate nella sezione Guida introduttiva al registro degli schemi di questo documento.
4. Avvia i produttori e i consumer all'utilizzo del registro degli schemi per scrivere e leggere i record da/per l'argomento Amazon MSK o Apache Kafka. Alcuni esempi di codice per produttori e consumatori sono disponibili nel ReadMe file delle librerie Serde. La libreria del registro degli schemi del produttore serializzerà automaticamente il record e aggiungerà un ID della versione dello schema al record.
5. Se lo schema di questo record è stato inserito o se la registrazione automatica è attivata, lo schema risulterà registrato nel registro degli schemi.
6. Il consumer che legge dall'argomento Amazon MSK o Apache Kafka, utilizzando la libreria del registro degli schemi di AWS Glue, cercherà automaticamente lo schema dal registro degli schemi.

Caso d'uso: integrazione del flusso di dati Amazon Kinesis con il registro degli schemi di AWS Glue

Per questa integrazione è necessario disporre di un flusso dei dati Amazon Kinesis. Per ulteriori informazioni, consulta Nozioni di base su Amazon Kinesis Data Streams nella Guida per gli sviluppatori di Amazon Kinesis Data Streams.

Esistono due modi per interagire con i dati in un flusso dei dati Kinesis.

- Tramite le librerie Kinesis Producer Library (KPL) e Kinesis Client Library (KCL) in Java. Il supporto multilingue non viene fornito.
- Tramite Le API `PutRecords`, `PutRecord`, e `GetRecords` di Kinesis Data Streams disponibili in AWS SDK for Java.

Se al momento si utilizzano le librerie KPL/KCL, si consiglia di continuare a utilizzare tale metodo. Come mostrato negli esempi, esistono versioni KCL e KPL aggiornate con il registro degli schemi integrato. In alternativa, se si utilizzano direttamente le API KDS, è possibile utilizzare il codice di esempio per sfruttare il registro degli schemi di AWS Glue.

L'integrazione del registro degli schemi è disponibile solo con KPL v0.14.2 o versioni successive e con KCL v2.3 o versioni successive. L'integrazione del registro degli schemi con il formato di dati JSON è disponibile solo con KPL v0.14.8 o versioni successive e con KCL v2.3.6 o versioni successive.

Interazione con i dati utilizzando Kinesis SDK V2

Questa sezione descrive l'interazione con Kinesis utilizzando Kinesis SDK V2

```
// Example JSON Record, you can construct a AVRO record also
private static final JsonDataWithSchema record =
 JsonDataWithSchema.builder(schemaString, payloadString);
private static final DataFormat dataFormat = DataFormat.JSON;

//Configurations for Schema Registry
GlueSchemaRegistryConfiguration gsrConfig = new GlueSchemaRegistryConfiguration("us-
east-1");

GlueSchemaRegistrySerializer glueSchemaRegistrySerializer =
        new GlueSchemaRegistrySerializerImpl(awsCredentialsProvider, gsrConfig);
GlueSchemaRegistryDataFormatSerializer dataFormatSerializer =
        new GlueSchemaRegistrySerializerFactory().getInstance(dataFormat, gsrConfig);

Schema gsrSchema =
        new Schema(dataFormatSerializer.getSchemaDefinition(record), dataFormat.name(),
 "MySchema");

byte[] serializedBytes = dataFormatSerializer.serialize(record);

byte[] gsrEncodedBytes = glueSchemaRegistrySerializer.encode(streamName, gsrSchema,
 serializedBytes);
```

```
PutRecordRequest putRecordRequest = PutRecordRequest.builder()
        .streamName(streamName)
        .partitionKey("partitionKey")
        .data(SdkBytes.fromByteArray(gsrEncodedBytes))
        .build();
shardId = kinesisClient.putRecord(putRecordRequest)
        .get()
        .shardId();

GlueSchemaRegistryDeserializer glueSchemaRegistryDeserializer = new
 GlueSchemaRegistryDeserializerImpl(awsCredentialsProvider, gsrConfig);

GlueSchemaRegistryDataFormatDeserializer gsrDataFormatDeserializer =
        glueSchemaRegistryDeserializerFactory.getInstance(dataFormat, gsrConfig);

GetShardIteratorRequest getShardIteratorRequest = GetShardIteratorRequest.builder()
        .streamName(streamName)
        .shardId(shardId)
        .shardIteratorType(ShardIteratorType.TRIM_HORIZON)
        .build();

String shardIterator = kinesisClient.getShardIterator(getShardIteratorRequest)
        .get()
        .shardIterator();

GetRecordsRequest getRecordRequest = GetRecordsRequest.builder()
        .shardIterator(shardIterator)
        .build();
GetRecordsResponse recordsResponse = kinesisClient.getRecords(getRecordRequest)
        .get();

List<Object> consumerRecords = new ArrayList<>();
List<Record> recordsFromKinesis = recordsResponse.records();

for (int i = 0; i < recordsFromKinesis.size(); i++) {
    byte[] consumedBytes = recordsFromKinesis.get(i)
            .data()
            .asByteArray();

    Schema gsrSchema = glueSchemaRegistryDeserializer.getSchema(consumedBytes);
    Object decodedRecord =
 gsrDataFormatDeserializer.deserialize(ByteBuffer.wrap(consumedBytes),
```

```
 gsrSchema.getSchemaDefinition());
    consumerRecords.add(decodedRecord);
}
```

Interazione con i dati utilizzando le librerie KPL/KCL

Questa sezione descrive l'integrazione di Kinesis Data Streams con il registro degli schemi utilizzando le librerie KPL/KCL. Per ulteriori informazioni sull'utilizzo di KPL/KCL, consulta Sviluppo di produttori utilizzando Amazon Kinesis Producer Library nella Guida per gli sviluppatori di Amazon Kinesis Data Streams.

Impostazione del registro degli schemi in KPL

1. Definisci la definizione dello schema per i dati, il formato dei dati e il nome dello schema creati nel registro degli schemi di AWS Glue.
2. Facoltativamente, puoi configurare l'oggetto `GlueSchemaRegistryConfiguration`.
3. Trasferisci l'oggetto dello schema a `addUserRecord API`.

```
private static final String SCHEMA_DEFINITION = "{"namespace": "example.avro",\n"
+ " "type": "record",\n"
+ " "name": "User",\n"
+ " "fields": [\n"
+ " {"name": "name", "type": "string"},\n"
+ " {"name": "favorite_number", "type": ["int", "null"]},\n"
+ " {"name": "favorite_color", "type": ["string", "null"]}\n"
+ " ]\n"
+ "}";

KinesisProducerConfiguration config = new KinesisProducerConfiguration();
config.setRegion("us-west-1")

//[Optional] configuration for Schema Registry.

GlueSchemaRegistryConfiguration schemaRegistryConfig =
new GlueSchemaRegistryConfiguration("us-west-1");

schemaRegistryConfig.setCompression(true);

config.setGlueSchemaRegistryConfiguration(schemaRegistryConfig);

///Optional configuration ends.
```

```
final KinesisProducer producer =
      new KinesisProducer(config);

final ByteBuffer data = getDataToSend();

com.amazonaws.services.schemaregistry.common.Schema gsrSchema =
    new Schema(SCHEMA_DEFINITION, DataFormat.AVRO.toString(), "demoSchema");

ListenableFuture<UserRecordResult> f = producer.addUserRecord(
config.getStreamName(), TIMESTAMP, Utils.randomExplicitHashKey(), data, gsrSchema);

private static ByteBuffer getDataToSend() {
      org.apache.avro.Schema avroSchema =
        new org.apache.avro.Schema.Parser().parse(SCHEMA_DEFINITION);

      GenericRecord user = new GenericData.Record(avroSchema);
      user.put("name", "Emily");
      user.put("favorite_number", 32);
      user.put("favorite_color", "green");

      ByteArrayOutputStream outBytes = new ByteArrayOutputStream();
      Encoder encoder = EncoderFactory.get().directBinaryEncoder(outBytes, null);
      new GenericDatumWriter<>(avroSchema).write(user, encoder);
      encoder.flush();
      return ByteBuffer.wrap(outBytes.toByteArray());
 }
```

Impostazione della libreria client di Kinesis

Svilupperai un consumer Kinesis Client Library in Java. Per ulteriori informazioni su, consulta Sviluppo di app Consumer Kinesis Client Library in Java nella Guida per gli sviluppatori di Amazon Kinesis Data Streams.

1. Crea un'istanza di `GlueSchemaRegistryDeserializer` passando un oggetto `GlueSchemaRegistryConfiguration`.
2. Passa `GlueSchemaRegistryDeserializer` a `retrievalConfig.glueSchemaRegistryDeserializer`.
3. Accedi allo schema dei messaggi in arrivo chiamando `kinesisClientRecord.getSchema()`.

```
GlueSchemaRegistryConfiguration schemaRegistryConfig =
```

```
   new GlueSchemaRegistryConfiguration(this.region.toString());

GlueSchemaRegistryDeserializer glueSchemaRegistryDeserializer =
   new
GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(),
schemaRegistryConfig);

RetrievalConfig retrievalConfig =
configsBuilder.retrievalConfig().retrievalSpecificConfig(new
PollingConfig(streamName, kinesisClient));
retrievalConfig.glueSchemaRegistryDeserializer(glueSchemaRegistryDeserializer);

 Scheduler scheduler = new Scheduler(
           configsBuilder.checkpointConfig(),
           configsBuilder.coordinatorConfig(),
           configsBuilder.leaseManagementConfig(),
           configsBuilder.lifecycleConfig(),
           configsBuilder.metricsConfig(),
           configsBuilder.processorConfig(),
           retrievalConfig
       );

public void processRecords(ProcessRecordsInput processRecordsInput) {
           MDC.put(SHARD_ID_MDC_KEY, shardId);
           try {
               log.info("Processing {} record(s)",
               processRecordsInput.records().size());
               processRecordsInput.records()
               .forEach(
                   r ->
                       log.info("Processed record pk: {} -- Seq: {} : data {} with
schema: {}",
                       r.partitionKey(),
r.sequenceNumber(), recordToAvroObj(r).toString(), r.getSchema()));
           } catch (Throwable t) {
               log.error("Caught throwable while processing records. Aborting.");
               Runtime.getRuntime().halt(1);
           } finally {
               MDC.remove(SHARD_ID_MDC_KEY);
           }
}

private GenericRecord recordToAvroObj(KinesisClientRecord r) {
   byte[] data = new byte[r.data().remaining()];
```

```
    r.data().get(data, 0, data.length);
    org.apache.avro.Schema schema = new
 org.apache.avro.Schema.Parser().parse(r.schema().getSchemaDefinition());
    DatumReader datumReader = new GenericDatumReader<>(schema);

    BinaryDecoder binaryDecoder = DecoderFactory.get().binaryDecoder(data, 0,
 data.length, null);
    return (GenericRecord) datumReader.read(null, binaryDecoder);
 }
```

Interazione con i dati utilizzando le API del flusso di dati Kinesis

Questa sezione descrive l'integrazione di Kinesis Data Streams con il registro degli schemi utilizzando le API di Kinesis Data Streams.

1. Aggiorna queste dipendenze di Maven:

```
<dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>com.amazonaws</groupId>
                <artifactId>aws-java-sdk-bom</artifactId>
                <version>1.11.884</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-kinesis</artifactId>
        </dependency>

        <dependency>
            <groupId>software.amazon.glue</groupId>
            <artifactId>schema-registry-serde</artifactId>
            <version>1.1.5</version>
        </dependency>

        <dependency>
```

```
            <groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-cbor</artifactId>
            <version>2.11.3</version>
        </dependency>
    </dependencies>
```

2. Nel produttore, aggiungi le informazioni sull'intestazione dello schema utilizzando il l'API `PutRecords` o `PutRecord` in Kinesis Data Streams.

```
//The following lines add a Schema Header to the record
        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
            new com.amazonaws.services.schemaregistry.common.Schema(schemaDefinition,
 DataFormat.AVRO.name(),
                schemaName);
        GlueSchemaRegistrySerializerImpl glueSchemaRegistrySerializer =
            new
 GlueSchemaRegistrySerializerImpl(DefaultCredentialsProvider.builder().build(), new
 GlueSchemaRegistryConfiguration(getConfigs()));
        byte[] recordWithSchemaHeader =
            glueSchemaRegistrySerializer.encode(streamName, awsSchema,
 recordAsBytes);
```

3. Nel produttore, utilizza l'API `PutRecords` o `PutRecord` per inserire il record nel flusso dei dati.
4. Nel consumer, rimuovi il record dello schema dall'intestazione e serializza un record dello schema Avro.

```
//The following lines remove Schema Header from record
        GlueSchemaRegistryDeserializerImpl glueSchemaRegistryDeserializer =
            new
 GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(),
 getConfigs());
        byte[] recordWithSchemaHeaderBytes = new
 byte[recordWithSchemaHeader.remaining()];
        recordWithSchemaHeader.get(recordWithSchemaHeaderBytes, 0,
 recordWithSchemaHeaderBytes.length);
        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
            glueSchemaRegistryDeserializer.getSchema(recordWithSchemaHeaderBytes);
        byte[] record =
 glueSchemaRegistryDeserializer.getData(recordWithSchemaHeaderBytes);

        //The following lines serialize an AVRO schema record
        if (DataFormat.AVRO.name().equals(awsSchema.getDataFormat())) {
```

```
            Schema avroSchema = new
  org.apache.avro.Schema.Parser().parse(awsSchema.getSchemaDefinition());
            Object genericRecord = convertBytesToRecord(avroSchema, record);
            System.out.println(genericRecord);
        }
```

Interazione con i dati utilizzando le API del flusso di dati Kinesis

Di seguito è riportato un codice di esempio per l'utilizzo delle API `PutRecords` e `GetRecords`.

```
//Full sample code
import
 com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializerImpl;
import
 com.amazonaws.services.schemaregistry.serializers.GlueSchemaRegistrySerializerImpl;
import com.amazonaws.services.schemaregistry.utils.AVROUtils;
import com.amazonaws.services.schemaregistry.utils.AWSSchemaRegistryConstants;
import org.apache.avro.Schema;
import org.apache.avro.generic.GenericData;
import org.apache.avro.generic.GenericDatumReader;
import org.apache.avro.generic.GenericDatumWriter;
import org.apache.avro.generic.GenericRecord;
import org.apache.avro.io.Decoder;
import org.apache.avro.io.DecoderFactory;
import org.apache.avro.io.Encoder;
import org.apache.avro.io.EncoderFactory;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.services.glue.model.DataFormat;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;


public class PutAndGetExampleWithEncodedData {
    static final String regionName = "us-east-2";
    static final String streamName = "testStream1";
    static final String schemaName = "User-Topic";
    static final String AVRO_USER_SCHEMA_FILE = "src/main/resources/user.avsc";
```

```
    KinesisApi kinesisApi = new KinesisApi();

    void runSampleForPutRecord() throws IOException {
        Object testRecord = getTestRecord();
        byte[] recordAsBytes = convertRecordToBytes(testRecord);
        String schemaDefinition =
AVROUtils.getInstance().getSchemaDefinition(testRecord);

        //The following lines add a Schema Header to a record
        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
            new com.amazonaws.services.schemaregistry.common.Schema(schemaDefinition,
DataFormat.AVRO.name(),
                schemaName);
        GlueSchemaRegistrySerializerImpl glueSchemaRegistrySerializer =
            new
GlueSchemaRegistrySerializerImpl(DefaultCredentialsProvider.builder().build(), new
GlueSchemaRegistryConfiguration(regionName));
        byte[] recordWithSchemaHeader =
            glueSchemaRegistrySerializer.encode(streamName, awsSchema, recordAsBytes);

        //Use PutRecords api to pass a list of records
        kinesisApi.putRecords(Collections.singletonList(recordWithSchemaHeader),
streamName, regionName);

        //OR
        //Use PutRecord api to pass single record
        //kinesisApi.putRecord(recordWithSchemaHeader, streamName, regionName);
    }

    byte[] runSampleForGetRecord() throws IOException {
        ByteBuffer recordWithSchemaHeader = kinesisApi.getRecords(streamName,
regionName);

        //The following lines remove the schema registry header
        GlueSchemaRegistryDeserializerImpl glueSchemaRegistryDeserializer =
            new
GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(), new
GlueSchemaRegistryConfiguration(regionName));
        byte[] recordWithSchemaHeaderBytes = new
byte[recordWithSchemaHeader.remaining()];
        recordWithSchemaHeader.get(recordWithSchemaHeaderBytes, 0,
recordWithSchemaHeaderBytes.length);

        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
```

```
            glueSchemaRegistryDeserializer.getSchema(recordWithSchemaHeaderBytes);

        byte[] record =
glueSchemaRegistryDeserializer.getData(recordWithSchemaHeaderBytes);

        //The following lines serialize an AVRO schema record
        if (DataFormat.AVRO.name().equals(awsSchema.getDataFormat())) {
            Schema avroSchema = new
org.apache.avro.Schema.Parser().parse(awsSchema.getSchemaDefinition());
            Object genericRecord = convertBytesToRecord(avroSchema, record);
            System.out.println(genericRecord);
        }

        return record;
    }

    private byte[] convertRecordToBytes(final Object record) throws IOException {
        ByteArrayOutputStream recordAsBytes = new ByteArrayOutputStream();
        Encoder encoder = EncoderFactory.get().directBinaryEncoder(recordAsBytes,
null);
        GenericDatumWriter datumWriter = new
GenericDatumWriter<>(AVROUtils.getInstance().getSchema(record));
        datumWriter.write(record, encoder);
        encoder.flush();
        return recordAsBytes.toByteArray();
    }

    private GenericRecord convertBytesToRecord(Schema avroSchema, byte[] record) throws
IOException {
        final GenericDatumReader<GenericRecord> datumReader = new
GenericDatumReader<>(avroSchema);
        Decoder decoder = DecoderFactory.get().binaryDecoder(record, null);
        GenericRecord genericRecord = datumReader.read(null, decoder);
        return genericRecord;
    }

    private Map<String, String> getMetadata() {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("event-source-1", "topic1");
        metadata.put("event-source-2", "topic2");
        metadata.put("event-source-3", "topic3");
        metadata.put("event-source-4", "topic4");
        metadata.put("event-source-5", "topic5");
        return metadata;
```

```
    }

    private GlueSchemaRegistryConfiguration getConfigs() {
        GlueSchemaRegistryConfiguration configs = new
 GlueSchemaRegistryConfiguration(regionName);
        configs.setSchemaName(schemaName);
        configs.setAutoRegistration(true);
        configs.setMetadata(getMetadata());
        return configs;
    }

    private Object getTestRecord() throws IOException {
        GenericRecord genericRecord;
        Schema.Parser parser = new Schema.Parser();
        Schema avroSchema = parser.parse(new File(AVRO_USER_SCHEMA_FILE));

        genericRecord = new GenericData.Record(avroSchema);
        genericRecord.put("name", "testName");
        genericRecord.put("favorite_number", 99);
        genericRecord.put("favorite_color", "red");

        return genericRecord;
    }
}
```

Caso d'uso: Amazon Managed Service per Apache Flink

Apache Flink è un diffuso framework open source e motore di elaborazione distribuito per calcoli con stato su flussi dei dati illimitati e limitati. Amazon Managed Service per Apache Flink è un AWS servizio completamente gestito che consente di creare e gestire applicazioni Apache Flink per elaborare dati di streaming.

Apache Flink open source fornisce una serie di origini e sink. Ad esempio, le origini dati predefinite includono la lettura da file, directory e socket e l'inserimento di dati da raccolte e iteratori. I DataStream connettori Apache Flink forniscono codice per Apache Flink per interfacciarsi con vari sistemi di terze parti, come Apache Kafka o Kinesis, come sorgenti e/o sink.

Per ulteriori informazioni, consulta la Guida per sviluppatori di Amazon Kinesis Data Analytics.

Connettore Apache Flink Kafka

Apache Flink fornisce un connettore per flusso dei dati Apache Kafka per la lettura e la scrittura di dati su argomenti Kafka con garanzie exactly-once. Il consumer Kafka di Flink,

`FlinkKafkaConsumer`, fornisce l'accesso alla lettura da uno o più argomenti di Kafka. Il produttore Kafka di Apache Flink, `FlinkKafkaProducer`, consente di scrivere un flusso di record su uno o più argomenti Kafka. Per ulteriori informazioni, consulta Apache Kafka Connector.

Connettore di flussi Apache Flink Kinesis

Il connettore del flusso dei dati Kinesis consente di accedere ai Amazon Kinesis Data Streams. `FlinkKinesisConsumer` è un'origine dati in streaming parallela exactly-once che esegue la sottoscrizione a più flussi Kinesis all'interno della stessa regione del servizio AWS e può gestire in modo trasparente la ripartizione dei flussi mentre il processo è in esecuzione. Ogni sottoattività del consumer è responsabile del recupero dei record di dati da più partizioni Kinesis. Il numero di partizioni recuperate da ogni sottoattività cambierà man mano che le partizioni vengono chiuse e create da Kinesis. `FlinkKinesisProducer` utilizza Kinesis Producer Library (KPL) per inserire i dati da un flusso Apache Flink in un flusso Kinesis. Per ulteriori informazioni, consulta Amazon Kinesis Streams Connector.

Per ulteriori informazioni, consulta il AWS Gluerepository Github di schema .

Integrazione con Apache Flink

La SerDes libreria fornita con Schema Registry si integra con Apache Flink. Per utilizzare Apache Flink, sarà necessario implementare le interfacce `SerializationSchema` e `DeserializationSchema` denominate `GlueSchemaRegistryAvroSerializationSchema` e `GlueSchemaRegistryAvroDeserializationSchema`, che possono essere collegate ai connettori Apache Flink.

Aggiunta di una dipendenza del registro degli schemi di AWS Glue nell'applicazione Apache Flink

Per impostare le dipendenze di integrazione sul registro degli schemi di AWS Glue nell'applicazione Apache Flink:

1. Aggiungi la dipendenza al file `pom.xml`.

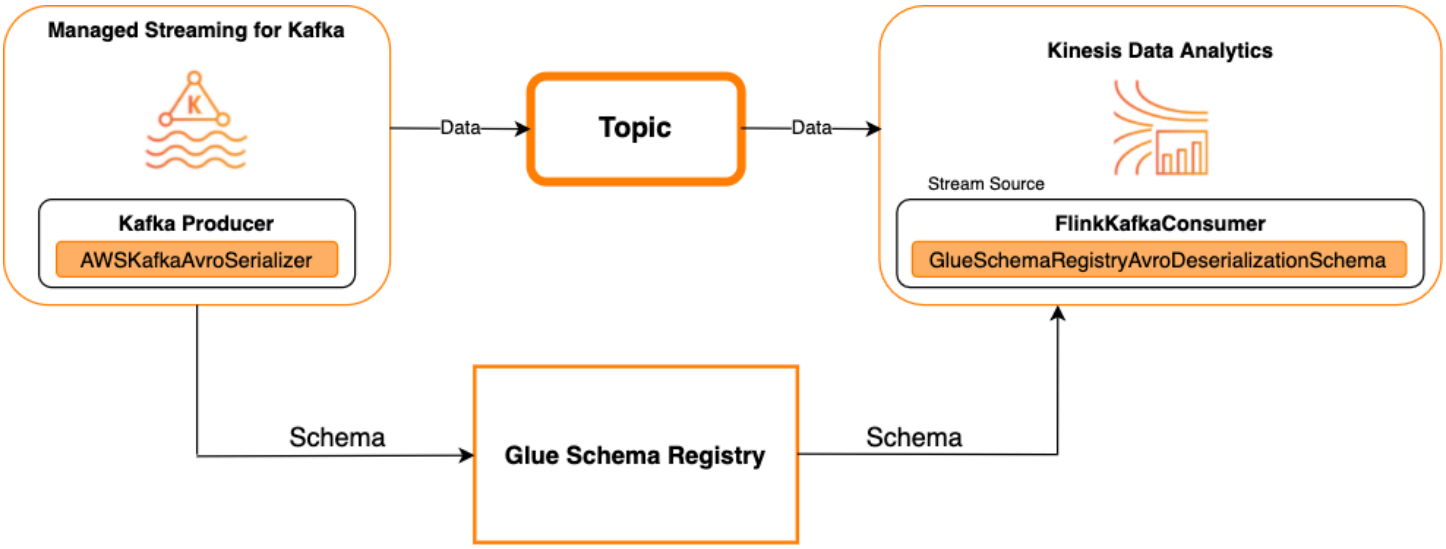
```
<dependency>
    <groupId>software.amazon.glue</groupId>
    <artifactId>schema-registry-flink-serde</artifactId>
    <version>1.0.0</version>
</dependency>
```

Integrazione di Kafka o Amazon MSK con Apache Flink

È possibile usare Servizio gestito da Amazon per Apache Flink per Apache Flink con Kafka come origine o Kafka come sink.

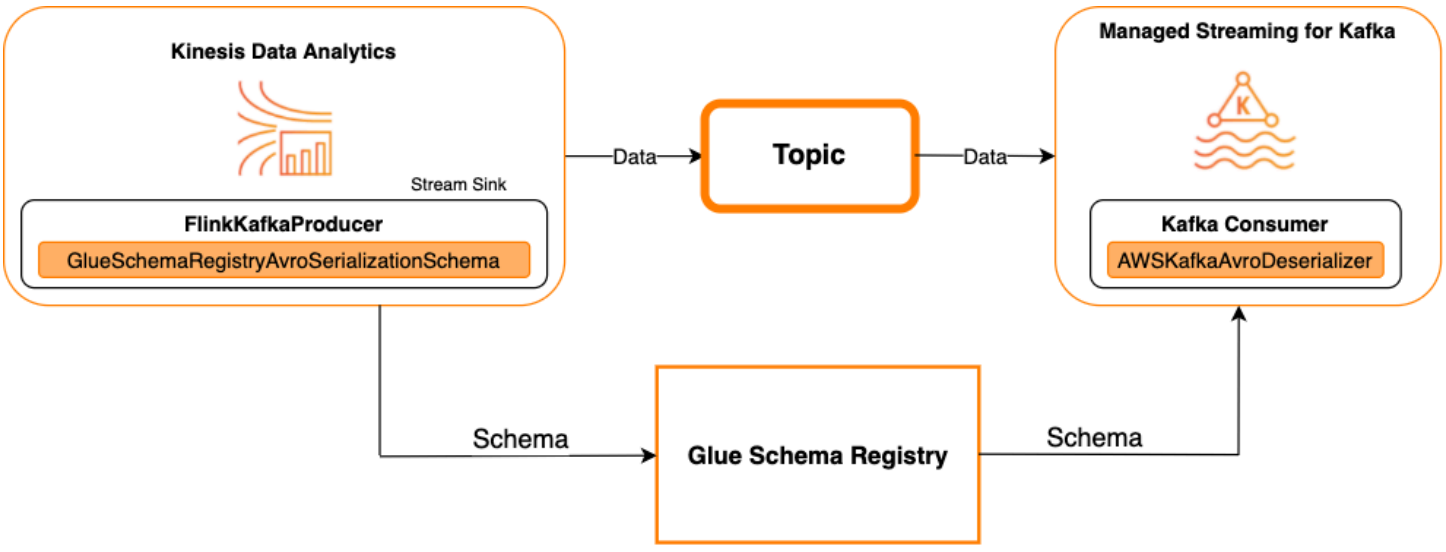
Kafka come fonte

Il diagramma seguente mostra l'integrazione di Flusso di dati Kinesis con Servizio gestito da Amazon per Apache Flink per Apache Flink, con Kafka come origine.



Kafka come sink

Il diagramma seguente mostra l'integrazione di Flusso di dati Kinesis con Servizio gestito da Amazon per Apache Flink per Apache Flink, con Kafka come sink.



Per integrare Kafka (o Amazon MSK) con Servizio gestito da Amazon per Apache Flink per Apache Flink, con Kafka come origine o Kafka come sink, apporta le modifiche al codice riportate di seguito. Aggiungi i blocchi di codice in grassetto al codice corrispondente nelle sezioni analoghe.

Se Kafka è la fonte, utilizza il codice deserializzatore (blocco 2). Se Kafka è il sink, utilizza il codice serializzatore (blocco 3).

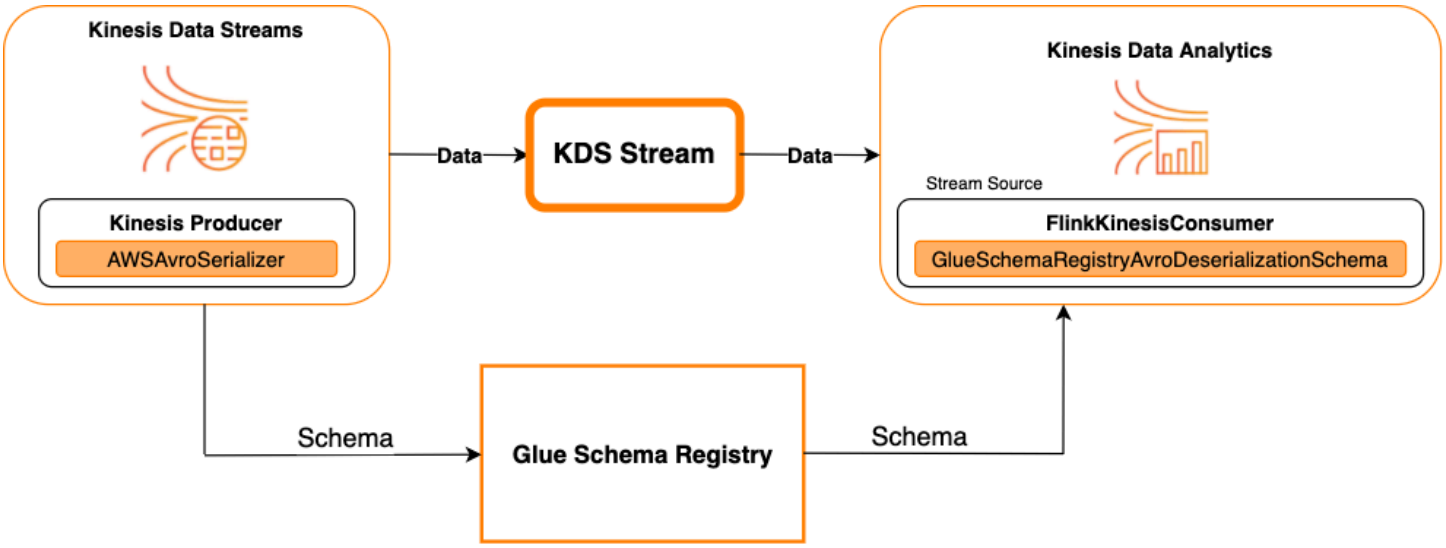
```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

String topic = "topic";
Properties properties = new Properties();
properties.setProperty("bootstrap.servers", "localhost:9092");
properties.setProperty("group.id", "test");

// block 1
Map<String, Object> configs = new HashMap<>();
configs.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
configs.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
configs.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());

FlinkKafkaConsumer<GenericRecord> consumer = new FlinkKafkaConsumer<>(
    topic,
    // block 2
    GlueSchemaRegistryAvroDeserializationSchema.forGeneric(schema, configs),
    properties);

FlinkKafkaProducer<GenericRecord> producer = new FlinkKafkaProducer<>(
    topic,
    // block 3
    GlueSchemaRegistryAvroSerializationSchema.forGeneric(schema, topic, configs),
    properties);

DataStream<GenericRecord> stream = env.addSource(consumer);
stream.addSink(producer);
env.execute();
```

Integrazione di Kinesis Data Streams con Apache Flink

È possibile utilizzare Servizio gestito da Amazon per Apache Flink per Apache Flink con Flusso di dati Kinesis come origine o sink.

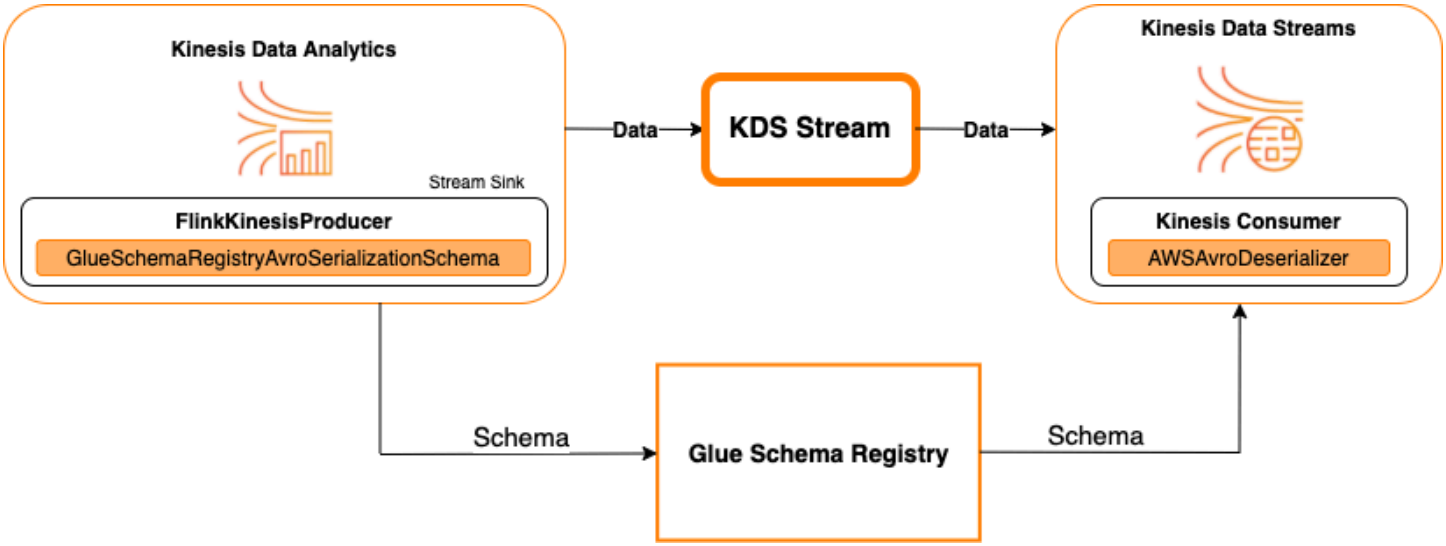
Kinesis Data Streams come fonte

Il diagramma seguente mostra l'integrazione di Flusso di dati Kinesis con Servizio gestito per Apache Flink per Apache Flink, con Flusso di dati Kinesis come origine.



Kinesis Data Streams come sink

Il diagramma seguente mostra l'integrazione di Flusso di dati Kinesis con Servizio gestito per Apache Flink per Apache Flink, con Flusso di dati Kinesis come sink.



Per integrare Flusso di dati Kinesis con Servizio gestito per Apache Flink per Apache Flink, con Flusso di dati Kinesis come origine o Flusso di dati Kinesis come sink, apporta le modifiche al codice riportate di seguito. Aggiungi i blocchi di codice in grassetto al codice corrispondente nelle sezioni analoghe.

Se Kinesis Data Streams è l'origine, utilizza il codice deserializzatore (blocco 2). Se Kinesis Data Streams il sink, utilizza il codice serializzatore (blocco 3).

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

String streamName = "stream";
Properties consumerConfig = new Properties();
consumerConfig.put(AWSConfigConstants.AWS_REGION, "aws-region");
consumerConfig.put(AWSConfigConstants.AWS_ACCESS_KEY_ID, "aws_access_key_id");
consumerConfig.put(AWSConfigConstants.AWS_SECRET_ACCESS_KEY, "aws_secret_access_key");
consumerConfig.put(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "LATEST");

// block 1
Map<String, Object> configs = new HashMap<>();
configs.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
configs.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
configs.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());

FlinkKinesisConsumer<GenericRecord> consumer = new FlinkKinesisConsumer<>(
    streamName,
    // block 2
    GlueSchemaRegistryAvroDeserializationSchema.forGeneric(schema, configs),
    properties);

FlinkKinesisProducer<GenericRecord> producer = new FlinkKinesisProducer<>(
    // block 3
    GlueSchemaRegistryAvroSerializationSchema.forGeneric(schema, topic, configs),
    properties);
producer.setDefaultStream(streamName);
producer.setDefaultPartition("0");

DataStream<GenericRecord> stream = env.addSource(consumer);
stream.addSink(producer);
env.execute();
```

Caso d'uso: integrazione con AWS Lambda

Per utilizzare una funzione AWS Lambda consumer Apache Kafka/Amazon MSK e deserializzare i messaggi con codifica Avro utilizzando il registro degli schemi di AWS Glue, visita la pagina sui laboratori MSK.

Caso d'uso: AWS Glue Data Catalog

Le tabelle AWS Glue supportano gli schemi che è possibile specificare manualmente o facendo riferimento al registro degli schemi di AWS Glue. Il registro degli schemi si integra con il catalogo dati per consentire l'utilizzo facoltativo degli schemi memorizzati nel registro degli schemi durante la creazione o l'aggiornamento di tabelle o partizioni AWS Glue nel catalogo dati. Per identificare una definizione dello schema nel registro degli schemi, è necessario conoscere almeno l'ARN dello schema di cui fa parte. Una versione di uno schema, che contiene una definizione dello schema, può essere referenziata dal relativo UUID o numero di versione. C'è sempre una versione dello schema, la versione "più recente", che può essere cercata senza conoscere il suo numero di versione o UUID.

Chiamando le operazioni `CreateTable` o `UpdateTable`, passerai una struttura `TableInput` che contiene un `StorageDescriptor`, che può avere un `SchemaReference` a uno schema esistente nel registro degli schemi. Allo stesso modo, quando si chiamano le API `GetTable` o `GetPartition`, la risposta può contenere lo schema e il `SchemaReference`. Quando una tabella o una partizione è stata creata utilizzando riferimenti allo schema, il catalogo dati tenterà di recuperare lo schema per questo riferimento. Nel caso in cui non sia in grado di trovare lo schema nel registro degli schemi, restituisce uno schema vuoto nella risposta `GetTable`; in caso contrario, la risposta conterrà sia lo schema che il riferimento allo schema.

Puoi eseguire le seguenti operazioni dalla console AWS Glue.

Per eseguire queste operazioni e creare, aggiornare o visualizzare le informazioni sullo schema, è necessario assegnare un ruolo IAM all'utente chiamante che fornisce le autorizzazioni per l'API `GetSchemaVersion`.

Aggiunta di una tabella o aggiornamento dello schema di una tabella

L'aggiunta di una nuova tabella da uno schema esistente associa la tabella a una versione specifica dello schema. Una volta registrate le nuove versioni dello schema, è possibile aggiornare la definizione della tabella dalla pagina View table (Visualizza tabella) nella console AWS Glue o utilizzando l'API Operazione UpdateTable (Python: update_table).

Aggiunta di una tabella da uno schema esistente

Puoi creare una tabella AWS Glue da una versione dello schema nel registro utilizzando la console AWS Glue o l'API `CreateTable`.

API AWS Glue

Chiamando l'API `CreateTable`, passerai un `TableInput` che contiene una `StorageDescriptor`, che può avere un `SchemaReference` a uno schema esistente nel registro degli schemi.

Console AWS Glue

Per creare una tabella dalla consoleAWS Glue:

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Tables (Tabelle).
3. Nel menu Add Tables (Aggiungi tabelle), scegli Add table from existing schema (Aggiungi tabella da schema esistente).
4. Configura le proprietà della tabella e il datastore come indicato nella Guida per gli sviluppatori di AWS Glue.
5. Nella pagina Choose a Glue schema (Scegli uno schema di Glue), seleziona il Registry (Registro) in cui si trova lo schema.
6. Scegli Schema name (Nome schema) e seleziona la Version (Versione) dello schema da applicare.
7. Esamina l'anteprima dello schema e scegli Next (Successivo).
8. Rivedi e crea la tabella.

Lo schema e la versione applicati alla tabella vengono visualizzati nella colonna Glue schema (Schema di Glue) nell'elenco delle tabelle. È possibile visualizzare la tabella per vedere ulteriori dettagli.

Aggiornamento dello schema per una tabella

Quando una nuova versione dello schema diventa disponibile, si consiglia di aggiornare lo schema di una tabella utilizzando l'API Operazione UpdateTable (Python: update_table) o la console AWS Glue.

> **Important**
>
> Aggiornando lo schema per una tabella esistente con uno schema AWS Glue specificato manualmente, il nuovo schema a cui si fa riferimento nel registro degli schemi potrebbe essere incompatibile. Questo potrebbe comportare la non riuscita dei processi.

API AWS Glue

Chiamando l'API `UpdateTable`, passerai un `TableInput` che contiene una `StorageDescriptor`, che può avere un `SchemaReference` a uno schema esistente nel registro degli schemi.

Console AWS Glue

Per aggiornare lo schema di una tabella dalla console AWS Glue:

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in Data catalog (Catalogo dati), seleziona Tables (Tabelle).
3. Visualizza la tabella nell'elenco delle tabelle.
4. Fai clic su Update schema (Aggiorna schema) nella casella che ti informa di una nuova versione.
5. Esamina le differenze tra lo schema attuale e quello nuovo.
6. Scegli Show all schema differences (Mostra tutte le differenze di schema) per ulteriori dettagli.
7. Scegli Save table (Salva tabella) per accettare la nuova versione.

Caso d'uso: streaming AWS Glue

Lo streaming AWS Glue consuma dati provenienti da fonti di streaming ed esegue operazioni ETL prima di scrivere su un sink di output. L'origine di streaming di input può essere specificata utilizzando una tabella dati o direttamente specificando la configurazione di origine.

Lo streaming AWS Glue supporta una tabella del catalogo dati per la sorgente di streaming creata con lo schema presente nel Registro degli schemi di AWS Glue. È possibile creare uno schema nel Registro degli schemi di AWS Glue e creare una tabella AWS Glue con una sorgente di streaming utilizzando questo schema. Questa tabella AWS Glue può essere utilizzata come input per un processo di streaming AWS Glue per la deserializzazione dei dati nel flusso di input.

Un punto da notare qui è quando lo schema nel Registro degli schemi AWS Glue cambia, è necessario riavviare il processo di streaming AWS Glue in modo da riflettere i cambiamenti nello schema.

Caso d'uso: flussi Apache Kafka

L'API di Apache Kafka Streams è una libreria client per l'elaborazione e l'analisi dei dati memorizzati in Apache Kafka. Questa sezione descrive l'integrazione di Apache Kafka Streams con il registro

degli schemi di AWS Glue, che consente di gestire e applicare gli schemi sulle applicazioni di streaming di dati. Per ulteriori informazioni su Apache Kafka Streams, consulta Apache Kafka Streams.

Integrazione con le librerie SerDes

Esiste una classe di `GlueSchemaRegistryKafkaStreamsSerde` con cui è possibile configurare un'applicazione Streams.

Codice di esempio di applicazione di flussi Kafka

Per utilizzare il registro degli schemi di AWS Glue all'interno di un'applicazione Apache Kafka Streams:

1. Configura l'applicazione Kafka Streams.

```
final Properties props = new Properties();
    props.put(StreamsConfig.APPLICATION_ID_CONFIG, "avro-streams");
    props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
    props.put(StreamsConfig.CACHE_MAX_BYTES_BUFFERING_CONFIG, 0);
    props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG,
 Serdes.String().getClass().getName());
    props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG,
 AWSKafkaAvroSerDe.class.getName());
    props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");

    props.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
    props.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
    props.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());
 props.put(AWSSchemaRegistryConstants.DATA_FORMAT, DataFormat.AVRO.name());
```

2. Crea un flusso dall'argomento avro-input.

```
StreamsBuilder builder = new StreamsBuilder();
final KStream<String, GenericRecord> source = builder.stream("avro-input");
```

3. Elabora i record di dati (nell'esempio vengono filtrati i record il cui valore favorite_color è rosa o in cui il valore amount è 15).

```
final KStream<String, GenericRecord> result = source
```

```
    .filter((key, value) -
> !"pink".equals(String.valueOf(value.get("favorite_color"))));
    .filter((key, value) -> !"15.0".equals(String.valueOf(value.get("amount"))));
```

4. Scrivi i risultati nell'argomento avro-output.

```
result.to("avro-output");
```

5. Avvia l'applicazione Apache Kafka Streams.

```
KafkaStreams streams = new KafkaStreams(builder.build(), props);
streams.start();
```

Risultati dell'implementazione

Questi risultati mostrano il processo di filtraggio dei record che sono stati filtrati nel passaggio 3 in base a favorite_color come "rosa" o valore come "15.0".

Record prima del filtraggio:

```
{"name": "Sansa", "favorite_number": 99, "favorite_color": "white"}
{"name": "Harry", "favorite_number": 10, "favorite_color": "black"}
{"name": "Hermione", "favorite_number": 1, "favorite_color": "red"}
{"name": "Ron", "favorite_number": 0, "favorite_color": "pink"}
{"name": "Jay", "favorite_number": 0, "favorite_color": "pink"}

{"id": "commute_1","amount": 3.5}
{"id": "grocery_1","amount": 25.5}
{"id": "entertainment_1","amount": 19.2}
{"id": "entertainment_2","amount": 105}
 {"id": "commute_1","amount": 15}
```

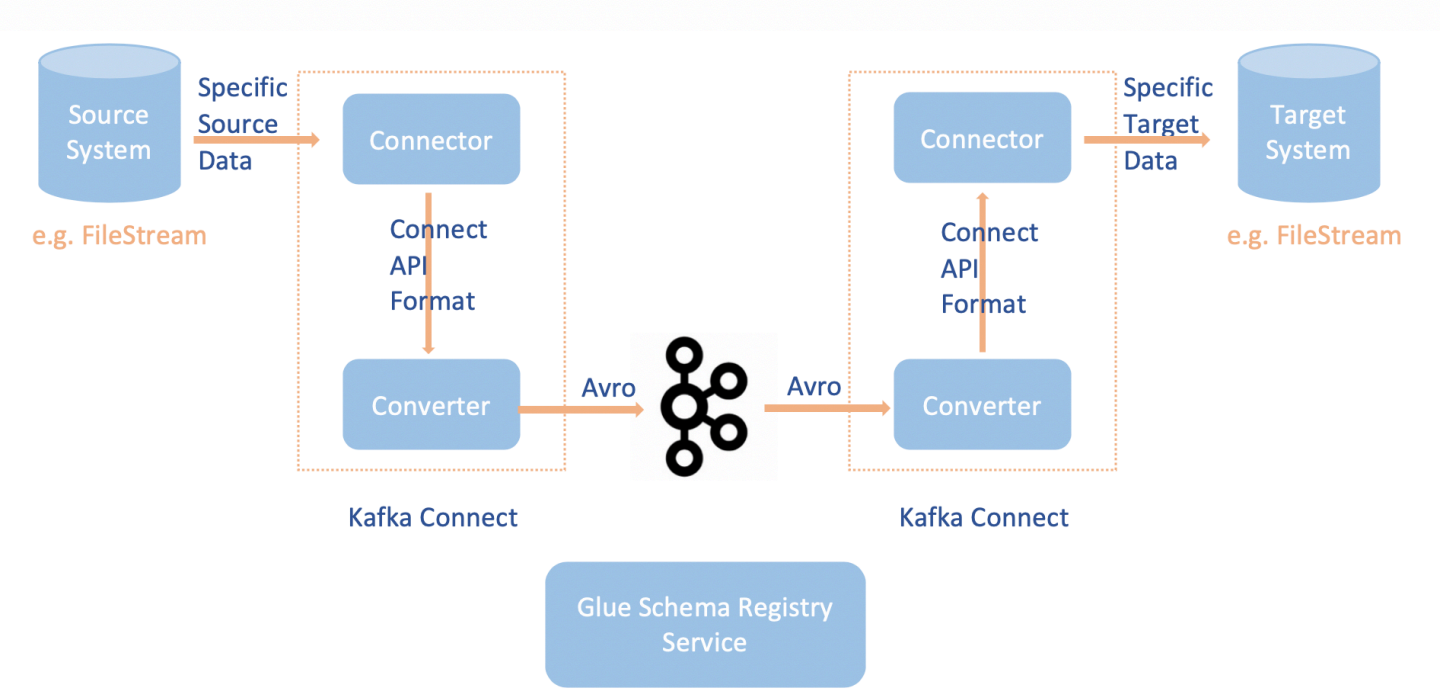
Record dopo il filtraggio:

```
{"name": "Sansa", "favorite_number": 99, "favorite_color": "white"}
{"name": "Harry", "favorite_number": 10, "favorite_color": "black"}
{"name": "Hermione", "favorite_number": 1, "favorite_color": "red"}
{"name": "Ron", "favorite_number": 0, "favorite_color": "pink"}
```

```
{"id": "commute_1","amount": 3.5}
{"id": "grocery_1","amount": 25.5}
{"id": "entertainment_1","amount": 19.2}
{"id": "entertainment_2","amount": 105}
```

Caso d'uso: Apache Kafka Connect

L'integrazione di Apache Kafka Connect con il registro degli schemi di AWS Glue consente di ottenere informazioni sullo schema dai connettori. I convertitori Apache Kafka specificano il formato dei dati all'interno di Apache Kafka e come tradurli in dati Apache Kafka Connect. Ogni utente di Apache Kafka Connect dovrà configurare questi convertitori in base al formato desiderato dei dati quando vengono caricati o memorizzati in Apache Kafka. In questo modo, è possibile definire i propri convertitori per tradurre i dati di Apache Kafka Connect nel tipo utilizzato nel registro degli schemi di AWS Glue (ad esempio: Avro) e utilizzare il nostro serializzatore per registrare il suo schema ed eseguire la serializzazione. I convertitori sono anche in grado di utilizzare il nostro deserializzatore per deserializzare i dati ricevuti da Apache Kafka e convertirli nuovamente in dati Apache Kafka Connect. Di seguito è riportato un diagramma di flusso di lavoro di esempio.



1. Installare del progetto `aws-glue-schema-registry` con la clonazione del repository Github per il registro degli schemi di AWS Glue.

```
git clone git@github.com:awslabs/aws-glue-schema-registry.git
cd aws-glue-schema-registry
mvn clean install
mvn dependency:copy-dependencies
```

2. Se hai intenzione di utilizzare Apache Kafka Connect in modalità standalone, aggiorna connect-standalone.properties seguendo le istruzioni di seguito per questo passaggio. Se prevedi di utilizzare Apache Kafka Connect in modalità distribuita, connect-avro-distributedaggiorna .properties seguendo le stesse istruzioni.

   a. Aggiungi queste proprietà anche al file delle proprietà di Apache Kafka connect:

   ```
   key.converter.region=aws-region
   value.converter.region=aws-region
   key.converter.schemaAutoRegistrationEnabled=true
   value.converter.schemaAutoRegistrationEnabled=true
   key.converter.avroRecordType=GENERIC_RECORD
   value.converter.avroRecordType=GENERIC_RECORD
   ```

   b. Aggiungi il comando seguente alla sezione Launch mode in .sh: kafka-run-class

   ```
   -cp $CLASSPATH:"<your AWS GlueSchema Registry base directory>/target/dependency/*"
   ```

3. Aggiungi il comando seguente alla sezione Launch mode in .sh kafka-run-class

   ```
   -cp $CLASSPATH:"<your AWS GlueSchema Registry base directory>/target/dependency/*"
   ```

   L'URL dovrebbe essere simile a questo:

   ```
   # Launch mode
   if [ "x$DAEMON_MODE" = "xtrue" ]; then
     nohup "$JAVA" $KAFKA_HEAP_OPTS $KAFKA_JVM_PERFORMANCE_OPTS $KAFKA_GC_LOG_OPTS
    $KAFKA_JMX_OPTS $KAFKA_LOG4J_OPTS -cp $CLASSPATH:"/Users/johndoe/aws-glue-schema-
   registry/target/dependency/*" $KAFKA_OPTS "$@" > "$CONSOLE_OUTPUT_FILE" 2>&1 < /dev/
   null &
   else
     exec "$JAVA" $KAFKA_HEAP_OPTS $KAFKA_JVM_PERFORMANCE_OPTS $KAFKA_GC_LOG_OPTS
    $KAFKA_JMX_OPTS $KAFKA_LOG4J_OPTS -cp $CLASSPATH:"/Users/johndoe/aws-glue-schema-
   registry/target/dependency/*" $KAFKA_OPTS "$@"
   fi
   ```

4. Se usi bash, esegui i seguenti comandi per configurare il tuo CLASSPATH nel bash_profile. Per qualsiasi altra shell, aggiorna l'ambiente di conseguenza.

```
echo 'export GSR_LIB_BASE_DIR=<>' >>~/.bash_profile
echo 'export GSR_LIB_VERSION=1.0.0' >>~/.bash_profile
echo 'export KAFKA_HOME=<your Apache Kafka installation directory>' >>~/.bash_profile
echo 'export CLASSPATH=$CLASSPATH:$GSR_LIB_BASE_DIR/avro-kafkaconnect-converter/
target/schema-registry-kafkaconnect-converter-$GSR_LIB_VERSION.jar:$GSR_LIB_BASE_DIR/
common/target/schema-registry-common-$GSR_LIB_VERSION.jar:$GSR_LIB_BASE_DIR/
avro-serializer-deserializer/target/schema-registry-serde-$GSR_LIB_VERSION.jar'
 >>~/.bash_profile
source ~/.bash_profile
```

5. (Facoltativo) Se vuoi eseguire il test con un'origine file semplice, clona il connettore dell'origine file.

```
git clone https://github.com/mmolimar/kafka-connect-fs.git
cd kafka-connect-fs/
```

   a. Sotto la configurazione del connettore di origine, modifica il formato dei dati in Avro, il lettore di file in `AvroFileReader` e aggiorna un oggetto Avro di esempio dal percorso del file da cui stai leggendo. Ad esempio:

```
vim config/kafka-connect-fs.properties
```

```
fs.uris=<path to a sample avro object>
policy.regexp=^.*\.avro$
file_reader.class=com.github.mmolimar.kafka.connect.fs.file.reader.AvroFileReader
```

   b. Installa il connettore di origine.

```
mvn clean package
echo "export CLASSPATH=\$CLASSPATH:\"\$(find target/ -type f -name '*.jar'| grep
 '\-package' | tr '\n' ':')\"" >>~/.bash_profile
source ~/.bash_profile
```

   c. Aggiorna le proprietà del sink in *<your Apache Kafka installation directory>*`/config/connect-file-sink.properties`, aggiorna il nome dell'argomento e il nome del file in uscita.

```
file=<output file full path>
```

```
topics=<my topic>
```

6. Avvia il connettore di origine (in questo esempio si tratta di un connettore dell'origine file).

```
$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-
standalone.properties config/kafka-connect-fs.properties
```

7. Esegui il connettore sink (in questo esempio si tratta di un connettore sink di file).

```
$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-
standalone.properties $KAFKA_HOME/config/connect-file-sink.properties
```

Per un esempio di utilizzo di Kafka Connect, guarda lo run-local-tests script.sh nella cartella integration-tests nel repository Github per lo Schema Registry. AWS Glue

Migrazione da un registro degli schemi di terze parti al registro degli schemi di AWS Glue

La migrazione da un registro degli schemi di terze parti al registro degli schemi di AWS Glue ha una dipendenza sull'attuale registro degli schemi di terze parti esistente. Se in un argomento Apache Kafka sono presenti record che sono stati inviati utilizzando un registro degli schemi di terze parti, i consumer hanno bisogno di questo registro per deserializzare tali record. `AWSKafkaAvroDeserializer` offre la possibilità di specificare una classe di deserializzatore secondario che punta al deserializzatore di terze parti e viene utilizzato per deserializzare questi record.

Esistono due criteri per il ritiro di uno schema di terze parti. Innanzitutto, il ritiro può avvenire solo dopo che i record negli argomenti Apache Kafka che utilizzano il registro degli schemi di terze parti non sono più richiesti da e per tutti i consumer. In secondo luogo, il ritiro può avvenire quando gli argomenti di Apache Kafka diventano datati, a seconda del periodo di conservazione specificato per tali argomenti. Tieni presente che se disponi di argomenti con tempo di conservazione infinito, è comunque possibile eseguire la migrazione al registro degli schemi di AWS Glue, ma non sarà possibile ritirare il registro degli schemi di terze parti. Come soluzione alternativa, è possibile utilizzare un'applicazione o Mirror Maker 2 per leggere l'argomento corrente e generarlo in un nuovo argomento nel registro degli schemi di AWS Glue.

Per eseguire la migrazione da un registro degli schemi di terze parti al registro degli schemi di AWS Glue:

1. Crea un registro nel registro degli schemi di AWS Glue o utilizza il registro di default.
2. Arresta il consumer. Modificalo per includere il registro degli schemi di AWS Glue come deserializzatore primario e il registro degli schemi di terze parti come secondario.
   - Imposta le proprietà del consumer. In questo esempio, il secondary_deserializer è impostato su un deserializzatore diverso. Il comportamento è il seguente: il consumer recupera i record da Amazon MSK e tenta di utilizzare prima `AWSKafkaAvroDeserializer`. Se non è in grado di leggere il magic byte che contiene l'ID dello schema Avro per il registro degli schemi di AWS Glue, `AWSKafkaAvroDeserializer` tenta quindi di utilizzare la classe di deserializzatore fornita nel secondary_deserializer. Le proprietà specifiche del deserializzatore secondario devono anche essere fornite nelle proprietà del consumer, come schema_registry_url_config e specific_avro_reader_config, come illustrato di seguito.

```
consumerProps.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 StringDeserializer.class.getName());
consumerProps.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 AWSKafkaAvroDeserializer.class.getName());
consumerProps.setProperty(AWSSchemaRegistryConstants.AWS_REGION,
 KafkaClickstreamConsumer.gsrRegion);
consumerProps.setProperty(AWSSchemaRegistryConstants.SECONDARY_DESERIALIZER,
 KafkaAvroDeserializer.class.getName());
consumerProps.setProperty(KafkaAvroDeserializerConfig.SCHEMA_REGISTRY_URL_CONFIG,
 "URL for third-party schema registry");
consumerProps.setProperty(KafkaAvroDeserializerConfig.SPECIFIC_AVRO_READER_CONFIG,
 "true");
```

3. Riavvia il consumer.
4. Arresta il produttore e puntalo al registro degli schemi di AWS Glue.
   a. Imposta le proprietà del produttore. In questo esempio, il produttore utilizzerà il registro di default ed eseguirà la registrazione automatica delle versioni degli schemi.

```
producerProps.setProperty(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
 StringSerializer.class.getName());
producerProps.setProperty(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
 AWSKafkaAvroSerializer.class.getName());
producerProps.setProperty(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2");
producerProps.setProperty(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.SPECIFIC_RECORD.getName());
```

```
producerProps.setProperty(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING,
 "true");
```

5. (Facoltativo) Sposta manualmente gli schemi e le versioni degli schemi esistenti dall'attuale registro degli schemi di terze parti al registro degli schemi di AWS Glue, nel registro di default nel registro degli schemi di AWS Glue o in un registro specifico non predefinito nel registro degli schemi di AWS Glue. Questa operazione può essere eseguita esportando gli schemi dai registri degli schemi di terze parti in formato JSON e creando nuovi schemi nel registro degli schemi di AWS Glue utilizzando la AWS Management Console o AWS CLI.

   Questo passaggio è importante se è necessario abilitare i controlli di compatibilità con le versioni precedenti degli schemi per le versioni appena create utilizzando AWS CLI e la AWS Management Console o quando i produttori inviano messaggi con un nuovo schema con l'opzione per la registrazione automatica delle versioni dello schema attivata.
6. Avvia il produttore.

Tutorial: aggiunta di un crawler AWS Glue

Per questo AWS Glue, ti viene chiesto di analizzare i dati degli arrivi dei principali vettori aerei per calcolare la popolarità degli aeroporti di partenza mese dopo mese. Hai i dati dei voli per l'anno 2016 in formato CSV memorizzati in Amazon S3. Prima di trasformare e analizzare i dati, è necessario catalogarne i metadati in AWS Glue Data Catalog.

In questo tutorial, aggiungiamo un crawler che deduce i metadati da questi registri di volo in Amazon S3 e creiamo una tabella nel catalogo dati.

Argomenti

- Prerequisiti
- Fase 1: aggiunta di un crawler
- Fase 2: esecuzione del crawler
- Fase 3: visualizzazione degli oggetti AWS Glue Data Catalog

Prerequisiti

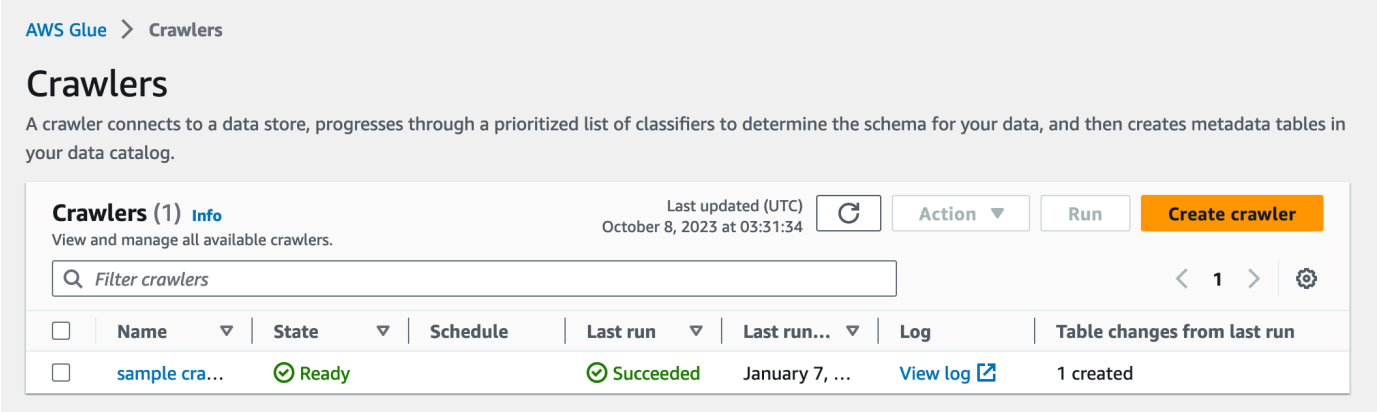
Questo tutorial presuppone che si abbia un account AWS e l'accesso ad AWS Glue.

Fase 1: aggiunta di un crawler

Segui questa procedura per configurare ed eseguire un crawler che estrae i metadati da un file CSV archiviato in Amazon S3.

Per creare un crawler in grado di leggere i file archiviati su Amazon S3

1. Sulla console di servizio AWS Glue, nel menu a sinistra, scegli Crawlers (Crawler).
2. Nella pagina Crawler, scegli Crea crawler. In questo modo viene avviata una serie di pagine che richiedono di specificare i dettagli del crawler.



3. Rinomina il crawler Crawler name (Nome crawler), inserisci **`Flights Data Crawler`**, quindi scegli Next (Avanti).

   I crawler invocano classificatori per dedurre lo schema dei dati. Questo tutorial utilizza il classificatore incorporato per CSV per impostazione predefinita.
4. Per il tipo di origine crawler, scegli Data stores (Datastore) e scegli Next (Avanti).
5. Ora puntiamo il crawler ai dati. Nella pagina Add a data store (Aggiungi datasore), scegli il datastore Amazon S3. Questa esercitazione non usa una connessione, quindi lascia il campo Connection (Connessione) vuoto se è visibile.

   Per l'opzione Crawl data in (Crawling dati), scegli Specified path in another account (Percorso specificato in un altro account). Quindi, nel campo Include path (Percorso di inclusione), inserisci il percorso in cui il crawler può trovare i dati dei voli, che è **`s3://crawler-public-us-east-1/flight/2016/csv`**. Dopo aver inserito il percorso, il titolo di questo campo cambia in Include path (Percorso di inclusione). Seleziona Next (Successivo).
6. È possibile eseguire il crawling di più datasore con un crawler singolo. Tuttavia, in questa esercitazione, stiamo utilizzando un solo datastore, quindi scegli No e poi Next (Successivo).

7. Il crawler ha bisogno delle autorizzazioni per accedere al Data Store e creare oggetti in AWS Glue Data Catalog. Per configurare queste autorizzazioni, scegli Create an IAM role (Crea un ruolo IAM). Il nome del ruolo IAM inizia con `AWSGlueServiceRole-` e, nel campo, inserisci l'ultima parte del nome del ruolo. Inserisci **`CrawlerTutorial`**, quindi seleziona Save (Salva).

   > **Note**
   >
   > Per creare un ruolo IAM, il tuo utente AWS deve avere le autorizzazioni `CreateRole`, `CreatePolicy` e `AttachRolePolicy`.

   La procedura guidata crea un ruolo IAM denominato `AWSGlueServiceRole-CrawlerTutorial`, allega la policy gestita da AWS `AWSGlueServiceRole` al ruolo e aggiunge una policy inline che consente l'accesso in lettura alla posizione di Amazon S3 `s3://crawler-public-us-east-1/flight/2016/csv`.

8. Crea una pianificazione per il crawler. Per Frequency (Frequenza), scegli Run on demand (Esegui on demand) e scegli Next (Successivo).

9. I crawler creano le tabelle nel catalogo dati. Un database nel catalogo dati contiene le tabelle. Per prima cosa, scegli Add database (Aggiungi database) per creare un database. Nella finestra popup, inserisci **`test-flights-db`** per il nome del database, quindi scegli Create (crea).

   Quindi, inserisci **`flights`** per Prefix added to tables (Prefisso aggiunto alle tabelle). Utilizza i valori predefiniti per il resto delle opzioni e scegli Next (Successivo).

10. Controlla le selezioni eseguite nella procedura guidata Add crawler (Aggiungi crawler). Se vedi errori, puoi scegliere Back (Indietro) per tornare alle pagine precedenti e apportare modifiche.

    Dopo aver esaminato le informazioni, scegli Finish (Termina) per creare il crawler.

Fase 2: esecuzione del crawler

Dopo aver creato un crawler, la procedura guidata ti reindirizza alla pagina di visualizzazione del crawler. Poiché crei il crawler con una pianificazione on demand, ti viene data la possibilità di eseguirlo.

Per eseguire il crawler

1. Il banner nella parte superiore di questa pagina ti permette di sapere che il crawler è stato creato e chiede se si desidera eseguirlo ora. Seleziona Run it now? (Eseguirlo adesso?) per eseguire il crawler.

   Il banner cambia e mostra i messaggi "Attempting to run" (Tentativo di esecuzione) e "Running" (In esecuzione) per il crawler. Dopo l'avvio del crawler, il banner scompare e la visualizzazione del crawler viene aggiornata per mostrare lo stato avvio del crawler. Dopo un minuto, puoi fare clic sull'icona Refresh (Aggiorna) per aggiornare lo stato del crawler visualizzato nella tabella.
2. Al completamento del crawler, viene visualizzato un nuovo banner che descrive le modifiche apportate dal crawler. Puoi scegliere la voce test-flights-db per visualizzare gli oggetti del catalogo dati.

Fase 3: visualizzazione degli oggetti AWS Glue Data Catalog

Il crawler legge i dati nella posizione di origine e crea tabelle nel catalogo dati. Una tabella è la definizione di metadati che rappresentano i tuoi dati, incluso il relativo schema. Le tabelle del catalogo dati non contengono dati. Vengono invece utilizzate come origine o destinazione in una definizione di processo.

Per visualizzare gli oggetti del catalogo dati creati dal crawler

1. Nel pannello di navigazione a sinistra, sotto Data catalog (Catalogo dati), scegli Database. Qui è possibile visualizzare database `flights-db` creato dal crawler.
2. Nel pannello di navigazione a sinistra, sotto Data catalog (Catalogo dati) e sotto Databases (Database), scegli Tables (Tabelle). Qui è possibile visualizzare la tabella `flightscsv` creata dal crawler. Scegliendo il nome della tabella, è possibile visualizzare le impostazioni, i parametri e le proprietà della tabella. Scorrendo verso il basso nella visualizzazione, puoi visualizzare lo schema, ovvero informazioni sulle colonne e sui tipi di dati della tabella.
3. Se scegli View partitions (Visualizza le partizioni) nella pagina di visualizzazione della tabella, puoi vedere le partizioni create per i dati. La prima colonna è la chiave di partizione.

Connessione ai dati

Una connessione di AWS Glue è un oggetto del Catalogo dati che memorizza le credenziali di accesso, le stringhe URI, le informazioni relative al cloud privato virtuale (VPC) e altri dati per uno specifico archivio dati. I crawler, i processi e gli endpoint di sviluppo di AWS Glue usano le connessioni per accedere a determinati tipi di archivi dati. È possibile utilizzare le connessioni sia per le origini che per le destinazioni e riutilizzare la stessa connessione su più crawler o più processi di estrazione, trasformazione e caricamento (ETL).

AWS Glue supporta i tipi di connessione seguenti:

- Amazon DocumentDB
- Amazon OpenSearch Service, da utilizzare con AWS Glue Spark.
- Amazon Redshift
- Kafka
- Azure Cosmos, per l'uso di Azure Cosmos DB per NoSQL con processi AWS Glue ETL
- Azure SQL, da usare con AWS Glue per Spark.
- Google BigQuery, da utilizzare con AWS Glue Spark.
- JDBC
- MongoDB
- MongoDB Atlas
- SAP HANA, da utilizzare con AWS Glue per Spark.
- Snowflake, da utilizzare con AWS Glue per Spark.
- Teradata Vantage, se viene utilizzato AWS Glue per Spark.
- Vertica, da utilizzare con AWS Glue per Spark.
- Varie offerte Amazon Relational Database Service (Amazon RDS).
- Rete (designa una connessione a un'origine dei dati all'interno di un Amazon Virtual Private Cloud [Amazon VPC]).
- Aurora (supportato se si utilizza il driver JDBC nativo; non tutte le funzionalità del driver sono utilizzabili)

Con AWS Glue Studio è anche possibile creare una connessione per un connettore, Un connettore è un pacchetto di codice opzionale che facilita l'accesso ai datastore in AWS Glue Studio. Per ulteriori informazioni, consulta Utilizzo di connettori e connessioni con AWS Glue Studio

Per informazioni su come connettersi ai database locali, consulta How to access and analyze on-premises data stores using AWS Glue nel blog AWS Big Data.

Questa sezione include gli argomenti seguenti per informazioni su come utilizzare le connessioni di AWS Glue:

- Proprietà della connessione AWS Glue
- Archiviazione delle credenziali di connessione in AWS Secrets Manager
- Aggiunta di una connessione AWS Glue
- Test di una connessione AWS Glue
- Configurazione delle chiamate AWS affinché passino attraverso il tuo VPC
- La connessione a un archivio dati JDBC in un VPC
- Utilizzo di una connessione MongoDB o MongoDB Atlas
- Crawling di un archivio di dati Amazon S3 utilizzando un endpoint VPC
- Risoluzione dei problemi di connessione in AWS Glue
- Tutorial: utilizzo del connettore AWS Glue per Elasticsearch

Proprietà della connessione AWS Glue

Questo argomento include informazioni sulle proprietà delle AWS Glue connessioni.

Argomenti

- Proprietà di connessione richieste
- Proprietà della connessione JDBC AWS Glue
- Proprietà di connessione AWS Glue MongoDB e MongoDB Atlas
- Connessione Snowflake
- Connessione Vertica
- Connessione SAP HANA
- Connessione Azure SQL
- Connessione Teradata Vantage

- OpenSearch Connessione al servizio
- Connessione Azure Cosmos
- Proprietà della connessione SSL AWS Glue
- Proprietà della connessione Apache Kafka per l'autenticazione client
- BigQuery Connessione a Google
- Connessione Vertica

Proprietà di connessione richieste

Quando definisci una connessione sulla console AWS Glue, devi specificare i valori per le proprietà seguenti:

Nome della connessione

Inserisci un nome univoco per la connessione.

Tipo di connessione

Scegliere JDBC o uno dei tipi di connessione specifici.

Per informazioni dettagliate sul tipo di connessione JDBC, consulta the section called “Proprietà della connessione JDBC”

Scegli Network (Rete) per la connessione a un'origine dati all'interno di un ambiente Amazon Virtual Private Cloud [Amazon VPC]).

A seconda del tipo scelto, la console AWS Glue visualizza altri campi obbligatori. Ad esempio, se selezioni Amazon RDS, devi scegliere il motore di database.

Require SSL connection (Connessione SSL necessaria)

Quando selezioni questa opzione, AWS Glue deve verificare che la connessione all'archivio dati avvenga tramite una connessione Secure Sockets Layer (SSL).

Per ulteriori informazioni, incluse le opzioni aggiuntive disponibili quando selezioni questa opzione, consulta the section called “Proprietà della connessione SSL”.

Select MSK cluster (Amazon managed streaming for Apache Kafka (MSK) only) (Seleziona cluster MSK [solo Amazon Managed Streaming for Apache Kafka])

Speciifica un cluster MSK da un altro AWS account.

Kafka bootstrap server URLs (Kafka only) (URL del server bootstrap Kafka [solo Kafka])

Specifica un elenco separato da virgole di URL del server bootstrap. Includi il numero di porta. Ad esempio: b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094, b-2.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094, b-3.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094

Proprietà della connessione JDBC AWS Glue

AWS Glue può connettersi ai seguentgli archivi dati tramite una connessione JDBC:

- Amazon Redshift
- Amazon Aurora
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Snowflake, quando si usano i crawler. AWS Glue
- Aurora (supportato se si utilizza il driver JDBC nativo; non tutte le funzionalità del driver sono utilizzabili)
- Amazon RDS for MariaDB

Important

Al momento, un processo ETL può utilizzare solo una connessione sottorete. Se disponi di più archivi dati in un processo, devono essere nella stessa sottorete o essere accessibili dalla sottorete.
Se scegli di importare le tue versioni dei driver JDBC per i crawler AWS Glue, i crawler consumeranno risorse nei processi AWS Glue e in Amazon S3 per garantire che i driver forniti vengano eseguiti nel tuo ambiente. L'utilizzo aggiuntivo delle risorse si rifletterà nel tuo account. Inoltre, anche se fornisci il tuo driver JDBC, non significa che il crawler sarà in grado di sfruttare tutte le funzionalità del driver. I driver sono limitati alle proprietà descritte nella sezione Defining connections in the Data Catalog.

Di seguito sono riportate le proprietà aggiuntive per il tipo di connessione JDBC.

URL JDBC

Inserisci l'URL per l'archivio dati JDBC. Per la maggior parte dei motori di database, questo campo appare nel seguente formato. In questo formato, sostituisci *protocol*, *host*, *port*, e *db_name* con le tue informazioni.

```
jdbc:protocol://host:port/db_name
```

A seconda del motore di database, potrebbe essere necessario un altro formato di URL JDBC. Questo formato può avere un utilizzo leggermente diverso dei due punti (:) e della barra (/) o delle diverse parole chiave per specificare i database.

Affinché JDBC si connetta all'archivio dati, è necessario fornire un db_name nell'archivio dati. Il db_name viene utilizzato per stabilire una connessione di rete con lo username e la password forniti. Una volta effettuata la connessione, AWS Glue può accedere ad altri database nell'archivio dati per eseguire un crawler o un processo ETL.

I seguenti esempi di URL JDBC mostrano la sintassi per diversi motori di database.

- Per la connessione a un archivio dati cluster Amazon Redshift con un database dev:

  ```
  jdbc:redshift://xxx.us-east-1.redshift.amazonaws.com:8192/dev
  ```

- Per la connessione a un archivio dati Amazon RDS for MySQL con un database employee:

  ```
  jdbc:mysql://xxx-cluster.cluster-xxx.us-east-1.rds.amazonaws.com:3306/
  employee
  ```

- Per la connessione a un archivio dati Amazon RDS for PostgreSQL con un database employee:

  ```
  jdbc:postgresql://xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:5432/employee
  ```

- Per connettersi a un archivio dati Amazon RDS for Oracle con un nome del servizio employee:

  ```
  jdbc:oracle:thin://@xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:1521/employee
  ```

  La sintassi per Amazon RDS for Oracle può seguire i seguenti modelli. In questi modelli, sostituisci *host*, *port*, *service_name*, e *SID* con le tue informazioni.

  - `jdbc:oracle:thin://@host:port/service_name`
  - `jdbc:oracle:thin://@host:port:SID`
- Per connettersi a un archivio dati Amazon RDS for Microsoft SQL Server con un database `employee`:

  ```
  jdbc:sqlserver://xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:1433;databaseName=employee
  ```

  La sintassi per Amazon RDS for SQL Server può seguire i seguenti modelli. In questi modelli, sostituisci *server_name*, *port* e *db_name* con le tue informazioni.

  - `jdbc:sqlserver://server_name:port;database=db_name`
  - `jdbc:sqlserver://server_name:port;databaseName=db_name`
- Per connetterti a un' Amazon Aurora PostgreSQL istanza del `employee` database, specifica l'endpoint per l'istanza del database, la porta e il nome del database:

  ```
  jdbc:postgresql://employee_instance_1.xxxxxxxxxxxx.us-
  east-2.rds.amazonaws.com:5432/employee
  ```
- Per connetterti a un Amazon RDS for MariaDB data store con un `employee` database, specifica l'endpoint per l'istanza del database, la porta e il nome del database:

  ```
  jdbc:mysql://xxx-cluster.cluster-xxx.aws-
  region.rds.amazonaws.com:3306/employee
  ```
-

  > **Warning**
  >
  > Le connessioni JDBC Snowflake sono supportate solo dai crawler. AWS Glue Quando si utilizza il connettore Snowflake nei job, utilizzare il tipo di connessione Snowflake. AWS Glue

  Per la connessione a un'istanza Snowflake del database `sample`, specifica l'endpoint per l'istanza Snowflake, l'utente, il nome del database e il nome del ruolo. Inoltre, puoi aggiungere il parametro `warehouse`.

  ```
  jdbc:snowflake://account_name.snowflakecomputing.com/?
  user=user_name&db=sample&role=role_name&warehouse=warehouse_name
  ```

> **Important**
>
> Per le connessioni Snowflake tramite JDBC, viene applicato l'ordine dei parametri nell'URL, che deve seguire l'ordine `user`, `db`, `role_name` e `warehouse`.

- Per connetterti a un'istanza Snowflake del `sample` database con link AWS privato, specifica l'URL JDBC snowflake come segue:

  ```
  jdbc:snowflake://account_name.region.privatelink.snowflakecomputing.com/?
  user=user_name&db=sample&role=role_name&warehouse=warehouse_name
  ```

Username

> **Note**
>
> Ti consigliamo di utilizzare un AWS segreto per memorizzare le credenziali di connessione invece di fornire direttamente il nome utente e la password. Per ulteriori informazioni, consulta Archiviazione delle credenziali di connessione in AWS Secrets Manager.

Fornisci un nome utente che dispone dell'autorizzazione per accedere all'archivio dati JDBC.

Password

Inserisci la password per il nome utente che dispone dell'autorizzazione per accedere all'archivio dati JDBC.

Porta

Inserisci la porta usata nell'URL JDBC per la connessione a un'istanza Amazon RDS Oracle. Questo campo viene visualizzato solo quando l'opzione Require SSL connection (Richiedi connessione SSL) è selezionata per un'istanza Amazon RDS Oracle.

VPC

Scegli il nome del Virtual Private Cloud (VPC) che contiene l'archivio dati. La console AWS Glue elenca tutti i VPC per la regione corrente.

> **Important**
>
> Quando si utilizza una connessione JDBC ospitata all'esterno AWS, ad esempio con dati provenienti da Snowflake, il VPC deve disporre di un gateway NAT che suddivide il traffico in sottoreti pubbliche e private. La sottorete pubblica viene utilizzata per la connessione alla fonte esterna e la sottorete interna viene utilizzata per l'elaborazione da. AWS Glue Per informazioni sulla configurazione di Amazon VPC per le connessioni esterne, consulta le pagine Connect to the internet or other networks using NAT devices e Configurazione di Amazon VPC per connessioni JDBC agli archivi dati Amazon RDS da AWS Glue.

Sottorete

Scegli la sottorete all'interno della VPC che contiene l'archivio dati. La console AWS Glue elenca tutte le sottoreti per l'archivio dati nel VPC.

Gruppi di sicurezza

Scegli i gruppi di sicurezza associati agli archivi dati. AWS Glue richiede uno o più gruppi di sicurezza con una regola di origine per il traffico in entrata che permette a AWS Glue di connettersi. La console AWS Glue elenca tutti i gruppi di sicurezza cui è concesso accesso in entrata al VPC. AWS Glue associa questi gruppi di sicurezza all'interfaccia di rete elastica collegata alla sottorete VPC.

Nome della classe del driver JDBC: facoltativo

Fornisci il nome personalizzato della classe del driver JDBC:

- Postgres: org.postgresql.Driver
- MySQL: com.mysql.jdbc.Driver, com.mysql.cj.jdbc.Driver
- Redshift: com.amazon.redshift.jdbc.Driver, com.amazon.redshift.jdbc42.Driver
- Oracle — oracle.jdbc.driver. OracleDriver
- SQL Server — com.microsoft.sqlserver.jdbc.SQL ServerDriver

Percorso S3 del driver JDBC: facoltativo

Fornisci la posizione Amazon S3 del driver JDBC personalizzato. Si tratta di un percorso assoluto verso un file .jar. Se desideri fornire dei driver JDBC per connetterti alle tue origini dati per i tuoi database supportati dai crawler, puoi specificare valori per i parametri `customJdbcDriverS3Path` e `customJdbcDriverClassName`.
L'utilizzo di un driver JDBC fornito da un cliente è limitato alle Proprietà di connessione richieste necessarie.

Proprietà di connessione AWS Glue MongoDB e MongoDB Atlas

Di seguito sono riportate le proprietà aggiuntive per il tipo di connessione MongoDB o MongoDB Atlas.

URL MongoDB

Inserisci l'URL del tuo archivio dati MongoDB o MongoDB Atlas:

- Per MongoDB: mongodb://host:port/database. L'host può essere un nome host, un indirizzo IP o un socket di dominio UNIX. Se la stringa di connessione non specifica una porta, utilizza la porta MongoDB predefinita, 27017.
- Per MongoDB Atlas: mongodb+srv://server.example.com/database. L'host può essere un nome host che corrisponde a un record DNS SRV. Il formato SRV non richiede una porta e utilizzerà la porta MongoDB predefinita, 27017.

Username

> **Note**
>
> Si consiglia di utilizzare un AWS segreto per archiviare le credenziali di connessione anziché fornire direttamente il nome utente e la password. Per ulteriori informazioni, consulta Archiviazione delle credenziali di connessione in AWS Secrets Manager.

Fornisci un nome utente che dispone dell'autorizzazione per accedere all'archivio dati JDBC.

Password

Inserisci la password per il nome utente che dispone dell'autorizzazione per accedere all'archivio dati MongoDB o MongoDB Atlas.

Connessione Snowflake

Le seguenti proprietà vengono utilizzate per configurare una connessione Snowflake utilizzata nei job ETL. AWS Glue Quando esegui il crawling di Snowflake, utilizza una connessione JDBC.

URL di Snowflake

L'URL dell'endpoint Snowflake. Per ulteriori informazioni sugli URL degli endpoint Snowflake, consulta la pagina Connecting to Your Accounts nella documentazione di Snowflake.

AWS Segreto

Il nome segreto di un segreto in AWS Secrets Manager. AWS Glue si connetterà a Snowflake utilizzando le `sfPassword` chiavi `sfUser` e del tuo segreto.

Ruolo Snowflake (facoltativo)

Durante la connessione AWS Glue verrà utilizzato un ruolo di sicurezza Snowflake.

Utilizza le seguenti proprietà per configurare una connessione a un endpoint Snowflake ospitato in Amazon VPC utilizzando AWS PrivateLink.

VPC

Scegli il nome del Virtual Private Cloud (VPC) che contiene l'archivio dati. La console AWS Glue elenca tutti i VPC per la regione corrente.

Sottorete

Scegli la sottorete all'interno della VPC che contiene l'archivio dati. La console AWS Glue elenca tutte le sottoreti per l'archivio dati nel VPC.

Gruppi di sicurezza

Scegli i gruppi di sicurezza associati agli archivi dati. AWS Glue richiede uno o più gruppi di sicurezza con una regola di origine per il traffico in entrata che permette a AWS Glue di connettersi. La console AWS Glue elenca tutti i gruppi di sicurezza cui è concesso accesso in entrata al VPC. AWS Glue associa questi gruppi di sicurezza all'interfaccia di rete elastica collegata alla sottorete VPC.

Connessione Vertica

Utilizzate le seguenti proprietà per configurare una connessione Vertica per AWS Glue i lavori ETL.

Host Vertica

Il nome host dell'installazione di Vertica.

Porta Vertica

La porta tramite cui è disponibile l'installazione di Vertica.

AWS Segreto

Il nome segreto di un segreto in AWS Secrets Manager. AWS Glue si connetterà a Vertica usando le chiavi del tuo segreto.

Utilizza le seguenti proprietà per configurare una connessione a un endpoint Vertica ospitato in Amazon VPC utilizzando.

VPC

Scegli il nome del Virtual Private Cloud (VPC) che contiene l'archivio dati. La console AWS Glue elenca tutti i VPC per la regione corrente.

Sottorete

Scegli la sottorete all'interno della VPC che contiene l'archivio dati. La console AWS Glue elenca tutte le sottoreti per l'archivio dati nel VPC.

Gruppi di sicurezza

Scegli i gruppi di sicurezza associati agli archivi dati. AWS Glue richiede uno o più gruppi di sicurezza con una regola di origine per il traffico in entrata che permette a AWS Glue di connettersi. La console AWS Glue elenca tutti i gruppi di sicurezza cui è concesso accesso in entrata al VPC. AWS Glue associa questi gruppi di sicurezza all'interfaccia di rete elastica collegata alla sottorete VPC.

Connessione SAP HANA

Utilizza le seguenti proprietà per configurare una connessione SAP HANA per AWS Glue i lavori ETL.

URL SAP HANA

UN URL JDBC SAP.

Gli URL SAP HANA JDBC sono nel modulo `jdbc:sap://saphanaHostname:saphanaPort/?databaseName=saphanaDBname,ParameterNa`

AWS Glue richiede i seguenti parametri URL JDBC:

- `databaseName`: un database predefinito in SAP HANA a cui connettersi.

AWS Segreto

Il nome segreto di un segreto in AWS Secrets Manager. AWS Glue si connetterà a SAP HANA utilizzando le chiavi del tuo segreto.

Utilizza le seguenti proprietà per configurare una connessione a un endpoint SAP HANA ospitato in Amazon VPC:

VPC

Scegli il nome del Virtual Private Cloud (VPC) che contiene l'archivio dati. La console AWS Glue elenca tutti i VPC per la regione corrente.

Sottorete

Scegli la sottorete all'interno della VPC che contiene l'archivio dati. La console AWS Glue elenca tutte le sottoreti per l'archivio dati nel VPC.

Gruppi di sicurezza

Scegli i gruppi di sicurezza associati agli archivi dati. AWS Glue richiede uno o più gruppi di sicurezza con una regola di origine per il traffico in entrata che permette a AWS Glue di connettersi. La console AWS Glue elenca tutti i gruppi di sicurezza cui è concesso accesso in entrata al VPC. AWS Glue associa questi gruppi di sicurezza all'interfaccia di rete elastica collegata alla sottorete VPC.

Connessione Azure SQL

Usa le seguenti proprietà per configurare una connessione Azure SQL per AWS Glue i processi ETL.

URL Azure SQL

L'URL JDBC di un endpoint Azure SQL.

L'elenco deve essere nel seguente formato: `jdbc:sqlserver://`*`databaseServerName`*`:`*`databasePort`*`;databaseName=`*`azuresqlDBname`*`;`.

AWS Glue richiede le seguenti proprietà URL:

- `databaseName`: un database predefinito in Azure SQL a cui connettersi.

Per altre informazioni sugli URL JDBC per le istanze gestite di Azure SQL, consulta la documentazione di Microsoft.

AWS Segreto

Il nome segreto di un segreto in AWS Secrets Manager. AWS Glue si connetterà ad Azure SQL usando le chiavi del tuo segreto.

Connessione Teradata Vantage

Usa le seguenti proprietà per configurare una connessione Teradata Vantage per i lavori ETL. AWS Glue

URL Teradata

Per connetterti a un'istanza Teradata, specifica il nome host dell'istanza del database e i parametri Teradata pertinenti:

```
jdbc:teradata://teradataHostname/ParameterName=ParameterValue,ParameterName=Pa
```

AWS Glue supporta i seguenti parametri URL JDBC:

- `DATABASE_NAME`: un database predefinito in Teradata a cui connettersi.
- `DBS_PORT`: specifica la porta Teradata, se non standard.

AWS Segreto

Il nome segreto di un segreto in AWS Secrets Manager. AWS Glue si connetterà a Teradata Vantage utilizzando le chiavi del tuo segreto.

Utilizza le seguenti proprietà per configurare una connessione a un endpoint Teradata Vantage ospitato in Amazon VPC:

VPC

Scegli il nome del Virtual Private Cloud (VPC) che contiene l'archivio dati. La console AWS Glue elenca tutti i VPC per la regione corrente.

Sottorete

Scegli la sottorete all'interno della VPC che contiene l'archivio dati. La console AWS Glue elenca tutte le sottoreti per l'archivio dati nel VPC.

Gruppi di sicurezza

Scegli i gruppi di sicurezza associati agli archivi dati. AWS Glue richiede uno o più gruppi di sicurezza con una regola di origine per il traffico in entrata che permette a AWS Glue di connettersi. La console AWS Glue elenca tutti i gruppi di sicurezza cui è concesso accesso in entrata al VPC. AWS Glue associa questi gruppi di sicurezza all'interfaccia di rete elastica collegata alla sottorete VPC.

OpenSearch Connessione al servizio

Utilizzate le seguenti proprietà per configurare una connessione di OpenSearch servizio per i lavori AWS Glue ETL.

Endpoint di dominio

Un endpoint OpenSearch di dominio Amazon Service avrà il seguente modulo predefinito, https://search - *`DomainName`* -. *`unstructuredIdContent regione .es.amazonaws.com.`* Per ulteriori informazioni sull'identificazione dell'endpoint del tuo dominio, consulta Creazione e gestione dei domini Amazon OpenSearch Service nella documentazione di Amazon OpenSearch Service.

Porta

La porta aperta sull'endpoint.

AWS Segreto

Il nome segreto di un segreto in AWS Secrets Manager. AWS Glue si connetterà al OpenSearch Servizio utilizzando le chiavi del tuo segreto.

Utilizza le seguenti proprietà per configurare una connessione a un endpoint di OpenSearch servizio ospitato in Amazon VPC:

VPC

Scegli il nome del Virtual Private Cloud (VPC) che contiene l'archivio dati. La console AWS Glue elenca tutti i VPC per la regione corrente.

Sottorete

Scegli la sottorete all'interno della VPC che contiene l'archivio dati. La console AWS Glue elenca tutte le sottoreti per l'archivio dati nel VPC.

Gruppi di sicurezza

Scegli i gruppi di sicurezza associati agli archivi dati. AWS Glue richiede uno o più gruppi di sicurezza con una regola di origine per il traffico in entrata che permette a AWS Glue di connettersi. La console AWS Glue elenca tutti i gruppi di sicurezza cui è concesso accesso in entrata al VPC. AWS Glue associa questi gruppi di sicurezza all'interfaccia di rete elastica collegata alla sottorete VPC.

Connessione Azure Cosmos

Usa le seguenti proprietà per configurare una connessione Azure Cosmos per i job ETL. AWS Glue

URI dell'endpoint dell'account di Azure Cosmos DB

L'endpoint utilizzato per connettersi ad Azure Cosmos. Per ulteriori informazioni, consulta la documentazione relativa ad Azure.

AWS Segreto

Il nome segreto di un segreto in AWS Secrets Manager. AWS Glue si connetterà ad Azure Cosmos usando le chiavi del tuo segreto.

Proprietà della connessione SSL AWS Glue

Di seguito sono riportati i dettagli della proprietà Require SSL connection (Richiedi connessione SSL).

Se non si richiede una connessione SSL, AWS Glue ignora gli errori quando utilizza SSL per crittografare una connessione a un archivio dati. Per istruzioni di configurazione, consulta la documentazione dell'archivio dati. Quando selezioni questa opzione, se AWS Glue non è in grado di connettersi, l'esecuzione del processo, del crawler o delle istruzioni ETL in un endpoint di sviluppo ha esito negativo.

> **Note**
>
> Snowflake supporta una connessione SSL per impostazione predefinita, quindi questa proprietà non è applicabile per Snowflake.

Questa opzione è convalidata in AWS Glue lato client. Per le connessioni JDBC, AWS Glue si connette solo tramite SSL con convalida del certificato e del nome host. Il supporto per la connessione SSL è disponibile per:

- Oracle Database
- Microsoft SQL Server
- PostgreSQL
- Amazon Redshift
- MySQL (solo istanze di Amazon RDS)
- Amazon Aurora MySQL (solo istanze di Amazon RDS)
- Amazon Aurora PostgreSQL (solo istanze Amazon RDS)
- Kafka, che include Amazon Managed Streaming for Apache Kafka
- MongoDB

> **Note**
>
> Per permettere a un archivio dati Amazon RDS Oracle di usare l'opzione Require SSL connection (Richiedi connessione SSL), devi creare e allegare un gruppo di opzioni all'istanza Oracle.
>
> 1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
> 2. Aggiungi un Option group (Gruppo di opzioni) all'istanza Amazon RDS Oracle. Per ulteriori informazioni su come aggiungere un gruppo di opzioni nella console Amazon RDS, consulta la sezione Creazione di un gruppo di opzioni
> 3. Aggiungere un'opzione al gruppo di opzioni per SSL. Il valore specificato in Port (Porta) per SSL verrà usato più avanti per creare un URL di connessione JDBC di AWS Glue per l'istanza Amazon RDS Oracle. Per ulteriori informazioni su come aggiungere un'opzione nella console Amazon RDS, consulta Aggiunta di un'opzione a un gruppo di opzioni nella Guida per l'utente di Amazon RDS. Per ulteriori informazioni sull'opzione SSL di Oracle, consulta Oracle SSL nella Guida per l'utente di Amazon RDS.
> 4. Nella console AWS Glue, crea una connessione all'istanza Amazon RDS Oracle. Nella definizione della connessione, seleziona Require SSL connection (Richiedi connessione

SSL). Quando richiesto, inserisci la Port (Porta) utilizzata nell'opzione Amazon RDS Oracle SSL.

Le seguenti proprietà facoltative aggiuntive sono disponibili quando è selezionata l'opzione Require SSL connection (Richiedi connessione SSL) per una connessione:

Certificato JDBC personalizzato in S3

Se disponi di un certificato che stai attualmente utilizzando per la comunicazione SSL con i database locali o cloud, puoi utilizzare tale certificato per le connessioni SSL alle origini dati o alle destinazioni AWS Glue. Immetti una posizione Amazon Simple Storage Service (Amazon S3) contenente un certificato root personalizzato. AWS Glue utilizza questo certificato per stabilire una connessione SSL al database. AWS Glue gestisce solo i certificati X.509. Il certificato deve essere codificato DER e fornito in formato PEM codificato Base64.

Se questo campo è lasciato vuoto, viene utilizzato il certificato predefinito.

Stringa di certificato JDBC personalizzata

Immetti le informazioni del certificato specifiche del database JDBC. Questa stringa viene utilizzata per la corrispondenza del dominio o la corrispondenza del nome distinto (DN). Per Oracle Database, questa stringa viene mappata al parametro `SSL_SERVER_CERT_DN` nella sezione di protezione del file `tnsnames.ora`. Per Microsoft SQL Server, questa stringa viene utilizzata come `hostNameInCertificate`.

Di seguito è riportato un esempio per il parametro `SSL_SERVER_CERT_DN` di Oracle Database.

```
cn=sales,cn=OracleContext,dc=us,dc=example,dc=com
```

Posizione del certificato emesso da una CA Kafka privata

Se disponi di un certificato che stai attualmente utilizzando per la comunicazione SSL con l'archivio dati Kafka, puoi utilizzare tale certificato con la connessione AWS Glue. Questa opzione è obbligatoria per gli archivi dati Kafka e facoltativa per Amazon Managed Streaming for Apache Kafka gli archivi dati. Immetti una posizione Amazon Simple Storage Service (Amazon S3) contenente un certificato root personalizzato. AWS Glue utilizza questo certificato per stabilire una connessione SSL all'archivio dati Kafka. AWS Glue gestisce solo i certificati X.509. Il certificato deve essere codificato DER e fornito in formato PEM codificato Base64.

Ignora convalida certificato

Seleziona la casella di controllo Skip certificate validation (Ignora convalida del certificato) per ignorare la convalida del certificato personalizzato da AWS Glue. Se scegli di convalidare, AWS Glue convalida l'algoritmo di firma e l'algoritmo di chiave pubblica oggetto per il certificato. Se la convalida del certificato non va a buon fine, qualsiasi processo ETL o crawler che utilizza la connessione ha esito negativo.

Gli unici algoritmi di firma consentiti sono SHA256withRSA, SHA384withRSA o SHA512withRSA. Per l'algoritmo della chiave pubblica oggetto, la lunghezza della chiave deve essere almeno 2048.

Posizione keystore del client Kafka

La posizione Amazon S3 del file keystore del client per l'autenticazione lato client Kafka. Il percorso deve essere nel formato s3://bucket/prefix/filename.jks. Deve terminare con il nome del file e l'estensione .jks.

Password del keystore del client Kafka (facoltativa)

La password per accedere al keystore fornito.

Password della chiave del client Kafka (facoltativa)

Un keystore può essere costituito da più chiavi, quindi questa è la password per accedere alla chiave client da utilizzare con la chiave lato server Kafka.

Proprietà della connessione Apache Kafka per l'autenticazione client

AWS Glue supporta il framework SASL (Simple Authentication and Security Layer) per l'autenticazione quando si crea una connessione Apache Kafka. Il framework SASL supporta vari meccanismi di autenticazione e AWS Glue offre i protocolli SCRAM (nome utente e password), GSSAPI (protocollo Kerberos) e PLAIN.

Utilizzare AWS Glue Studio per configurare uno dei seguenti metodi di autenticazione client. Per ulteriori informazioni, vedere Creazione di connessioni per i connettori nella guida per l' AWS Glue Studio utente.

- Nessuno: nessuna autenticazione. Questo è utile se si crea una connessione a scopo di test.
- SASL/SCRAM-SHA-512: la scelta di questo metodo di autenticazione consentirà di specificare le credenziali di autenticazione. Sono disponibili due opzioni:

  - Usa AWS Secrets Manager (consigliato): se selezioni questa opzione, puoi memorizzare il nome utente e la password in AWS Secrets Manager e AWS Glue consentirne l'accesso quando necessario. Specifica il segreto che memorizza le credenziali di autenticazione SSL o SASL. Per ulteriori informazioni, consulta Archiviazione delle credenziali di connessione in AWS Secrets Manager.
  - Inserisci direttamente un nome utente e una password.
- SASL/GSSAPI (Kerberos): selezionando questa opzione, è possibile selezionare la posizione del file keytab, il file krb5.conf e inserire il nome principale Kerberos e il nome del servizio Kerberos. Le posizioni per il file keytab e il file krb5.conf devono trovarsi in una posizione Amazon S3. Poiché MSK non supporta ancora SASL/GSSAPI, questa opzione è disponibile solo per i cluster Apache Kafka gestiti dal cliente. Per ulteriori informazioni, consulta la Documentazione di MIT Kerberos: keytab.
- SASL/PLAIN: scegli questo metodo di autenticazione per specificare le credenziali di autenticazione. Sono disponibili due opzioni:
  - Usa AWS Secrets Manager (consigliato): se selezioni questa opzione, puoi memorizzare le tue credenziali in AWS Secrets Manager e consentire AWS Glue l'accesso alle informazioni quando necessario. Specifica il segreto che memorizza le credenziali di autenticazione SSL o SASL.
  - Fornisci direttamente nome utente e password.
- Autenticazione client SSL: selezionando questa opzione, è possibile selezionare la posizione del keystore client Kafka navigando su Amazon S3. Facoltativamente, è possibile inserire la password del keystore del client Kafka e la password della chiave del client Kafka.

BigQuery Connessione a Google

Le seguenti proprietà vengono utilizzate per configurare una BigQuery connessione Google utilizzata nei lavori AWS Glue ETL. Per ulteriori informazioni, consulta the section called “Connessioni BigQuery”.

AWS Segreto

Il nome segreto di un segreto in AWS Secrets Manager. AWS Glue ETL jobs si connetterà a Google BigQuery utilizzando la `credentials` chiave del tuo segreto.

Connessione Vertica

Le seguenti proprietà vengono utilizzate per configurare una connessione Vertica utilizzata nei lavori AWS Glue ETL. Per ulteriori informazioni, consulta the section called “Connessioni Vertica”.

Archiviazione delle credenziali di connessione in AWS Secrets Manager

Ti consigliamo di fornire le credenziali di connessione per il tuo archivio dati tramite AWS Secrets Manager. Utilizzando Secrets Manager in questo modo, concedi a AWS Glue l'accesso al tuo segreto in fase di runtime durante l'esecuzione di crawler e processi ETL, proteggendo le tue credenziali.

Prerequisiti

Per utilizzare Secrets Manager con AWS Glue, devi concedere al tuo ruolo IAM per AWS Glue l'autorizzazione necessaria per recuperare i valori segreti. La policy AWS gestita da `AWSGlueServiceRole` non include autorizzazioni AWS Secrets Manager. Per le policy IAM di esempio, consulta Esempio: Autorizzazione per recuperare valori segreti nella Guida per l'utente di AWS Secrets Manager.

In base all'impostazione della rete, potrebbe essere necessario anche creare un endpoint VPC per stabilire una connessione privata tra il VPC e Secrets Manager. Per ulteriori informazioni, consulta Utilizzo di un endpoint VPC AWS Secrets Manager.

Creazione di un segreto per AWS Glue

1. Segui le istruzioni in Creazione e gestione di segreti nella Guida per l'utente di AWS Secrets Manager. L'esempio JSON seguente mostra come specificare le credenziali nella scheda Plaintext quando crei un segreto per AWS Glue.

   ```
   {
     "username": "EXAMPLE-USERNAME",
     "password": "EXAMPLE-PASSWORD"
   }
   ```

2. Associa il segreto a una connessione utilizzando l'interfaccia di AWS Glue Studio. Per ulteriori informazioni, consulta la pagina Creating connections for connectors nella Guida per l'utente di AWS Glue Studio.

Aggiunta di una connessione AWS Glue

Puoi connetterti alle origini dati in AWS Glue per Spark a livello di programmazione. Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark

Puoi anche utilizzare la console AWS Glue per aggiungere, modificare, eliminare e testare le connessioni. Per informazioni sulle connessioni AWS Glue, consulta Connessione ai dati.

Per aggiungere una connessione AWS Glue

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione, in catalogo dati (Catalogo dati), seleziona Connections (Connessioni).
3. Scegli Add connection (Aggiungi connessione) e completa la procedura guidata, immettendo le proprietà di connessione come descritto in the section called “Proprietà della connessione AWS Glue”.

Connessione a Redshift in AWS Glue Studio

> **Note**
>
> È possibile utilizzare AWS Glue per Spark per leggere e scrivere su tabelle nei database Amazon Redshift al di fuori di AWS Glue Studio. Per configurare Amazon Redshift con i processi AWS Glue a livello di programmazione, consulta la pagina Connessioni Redshift.

AWS Glue fornisce il supporto integrato per Amazon Redshift. AWS Glue Studio fornisce un'interfaccia visiva per connettersi a Amazon Redshift, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

- Creazione di una connessione Amazon Redshift
- Creazione di un nodo di origine Amazon Redshift
- Creazione di un nodo di destinazione Amazon Redshift
- Opzioni avanzate

Creazione di una connessione Amazon Redshift

Autorizzazioni necessarie

Sono necessarie autorizzazioni aggiuntive per utilizzare i cluster Amazon Redshift e gli ambienti serverless Amazon Redshift. Per ulteriori informazioni su come aggiungere autorizzazioni ai processi ETL, consulta la pagina Review IAM permissions needed for ETL jobs.

- redshift:DescribeClusters
- redshift-serverless:ListWorkgroups
- redshift-serverless:ListNamespaces

Panoramica

Quando si aggiunge una connessione Amazon Redshift, è possibile scegliere una connessione Amazon Redshift esistente o creare una nuova connessione quando si aggiunge un nodo Origine dati - Redshift in AWS Glue Studio.

AWS Glue supporta sia i cluster Amazon Redshift sia gli ambienti serverless Amazon Redshift. Quando si crea una connessione, gli ambienti serverless Amazon Redshift mostrano l'etichetta Serverless accanto all'opzione di connessione.

Per ulteriori informazioni su come creare una connessione Amazon Redshift, consulta la pagina Moving data to and from Amazon Redshift.

Creazione di un nodo di origine Amazon Redshift

Autorizzazioni necessarie

I processi AWS Glue Studio che utilizzano origini dati Amazon Redshift richiedono autorizzazioni aggiuntive. Per ulteriori informazioni su come aggiungere autorizzazioni ai processi ETL, consulta la pagina Review IAM permissions needed for ETL jobs.

Le seguenti autorizzazioni sono necessarie per utilizzare una connessione Amazon Redshift.

- redshift-data:ListSchemas
- redshift-data:ListTables
- redshift-data:DescribeTable
- redshift-data:ExecuteStatement

- redshift-data:DescribeStatement
- redshift-data:GetStatementResult

Aggiunta di un'origine dati Amazon Redshift

Per aggiungere un nodo Origine dati: Amazon Redshift:

1. Scegli il tipo di accesso Amazon Redshift:

   - Connessione dati diretta (consigliata): scegli questa opzione se desideri accedere direttamente ai tuoi dati Amazon Redshift. Questa è l'opzione consigliata nonché quella predefinita.
   - Data Catalog tables: scegli questa opzione se hai delle tabelle di Catalogo dati che desideri utilizzare.

2. Se scegli Connessione dati diretta, scegli la connessione per la tua origine dati Amazon Redshift. Ciò presuppone che la connessione esista già e che sia possibile effettuare una selezione tra le connessioni esistenti. Se devi creare una connessione, scegli Crea connessione Redshift. Per ulteriori informazioni, consulta la pagina Overview of using connectors and connections.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà. Le informazioni sulla connessione sono visibili, tra cui URL, gruppi di sicurezza, sottorete, zona di disponibilità, descrizione, nonché timestamp di creazione (UTC) e ultimo aggiornamento (UTC).

3. Scegli un'opzione di origine Amazon Redshift:

   - Scegli una singola tabella: questa è la tabella che contiene i dati a cui desideri accedere da una singola tabella Amazon Redshift.
   - Inserisci una query personalizzata: ti consente di accedere a un set di dati da più tabelle Amazon Redshift in base alla tua query personalizzata.

4. Se hai scelto una singola tabella, scegli lo schema Amazon Redshift. L'elenco degli schemi disponibili tra cui scegliere è determinato dalla tabella selezionata.

   In alternativa, scegli Inserisci query personalizzata. Scegli questa opzione per accedere a un set di dati personalizzato da più tabelle Amazon Redshift. Se scegli questa opzione, inserisci la query Amazon Redshift.

   Quando ti connetti a un ambiente serverless Amazon Redshift, aggiungi la seguente autorizzazione alla query personalizzata:

```
GRANT SELECT ON ALL TABLES IN <schema> TO PUBLIC
```

Puoi scegliere Acquisisci schema per leggere lo schema in base alla query che hai inserito. È inoltre possibile scegliere Apri editor di query Redshift per inserire una query Amazon Redshift. Per ulteriori informazioni, consulta la pagina Querying a database using the query editor.

5. In Prestazioni e sicurezza, scegli la directory di gestione temporanea di Amazon S3 e il ruolo IAM.
   - Directory di gestione temporanea di Amazon S3: scegli la posizione Amazon S3 per la gestione temporanea dei dati.
   - Ruolo IAM: scegli il ruolo IAM che può scrivere nella posizione Amazon S3 che hai selezionato.
6. In Parametri Redshift personalizzati - facoltativo, inserisci il parametro e il valore.

Creazione di un nodo di destinazione Amazon Redshift

Autorizzazioni necessarie

I processi AWS Glue Studio che utilizzano destinazioni dati Amazon Redshift richiedono autorizzazioni aggiuntive. Per ulteriori informazioni su come aggiungere autorizzazioni ai processi ETL, consulta la pagina Review IAM permissions needed for ETL jobs.

Le seguenti autorizzazioni sono necessarie per utilizzare una connessione Amazon Redshift.

- redshift-data:ListSchemas
- redshift-data:ListTables

Aggiunta di un nodo di destinazione Amazon Redshift

Per creare un nodo di destinazione Amazon Redshift:

1. Scegli una tabella Amazon Redshift esistente come destinazione o inserisci un nuovo nome per la tabella.
2. Quando utilizzi il nodo di destinazione Destinazione dati - Redshift, puoi scegliere tra le seguenti opzioni:

- AGGIUNGI: se esiste già una tabella, scarica tutti i nuovi dati nella tabella come inserto. Se la tabella non esiste, procedi alla sua creazione e quindi inserisci tutti i nuovi dati.

  Inoltre, seleziona la casella se desideri aggiornare (UPSERT) i record esistenti nella tabella di destinazione. La tabella deve già esistere, altrimenti l'operazione avrà esito negativo.
- MERGE: AWS Glue aggiorna o aggiunge i dati alla tabella di destinazione in base alle condizioni specificate.

  > **Note**
  >
  > Per utilizzare l'operazione di merge in AWS Glue, è necessario abilitare la funzionalità di merge di Amazon Redshift. Per istruzioni su come abilitare il merge per un'istanza Amazon Redshift, consulta la pagina MERGE (preview).

  Scegli le opzioni:

  - Scegli chiavi e operazioni semplici: scegli le colonne da utilizzare come chiavi di corrispondenza tra i dati di origine e il set di dati di destinazione.

    Specifica le seguenti opzioni in caso di corrispondenza:

    - Aggiorna il record nel set di dati di destinazione con i dati dell'origine.
    - Elimina il record nel set di dati di destinazione.

    Specifica le seguenti opzioni in caso di mancata corrispondenza:

    - Inserisci i dati di origine come nuova riga nel set di dati di destinazione.
    - Non fare nulla.
  - Inserisci un'istruzione MERGE personalizzata: puoi quindi scegliere Convalida l'istruzione MERGE per verificare che l'istruzione sia valida o non valida.
- TRUNCATE: se esiste già una tabella, tronca i dati della tabella cancellando prima il contenuto della tabella di destinazione. Se il troncamento ha esito positivo, inserisci tutti i dati. Se la tabella non esiste, procedi alla sua creazione e quindi inserisci tutti i dati. Se il troncamento non va a buon fine, l'operazione non andrà a buon fine.
- DROP: se esiste già una tabella, elimina i metadati e i dati della tabella. Se l'eliminazione ha esito positivo, inserisci tutti i dati. Se la tabella non esiste, procedi alla sua creazione e quindi inserisci tutti i dati. Se l'eliminazione non va a buon fine, l'operazione non andrà a buon fine.

- CREATE: crea una nuova tabella con il nome predefinito. Se il nome della tabella esiste già, crea una nuova tabella aggiungendo il suffisso nel formato `job_datetime` al nome per renderlo unico. Questo inserirà tutti i dati nella nuova tabella. Se la tabella esiste già, al nome finale della tabella verrà aggiunto il suffisso. Se la tabella non esiste, verrà creata una tabella. In entrambi i casi, verrà creata una nuova tabella.

Opzioni avanzate

Consulta la pagina Using the Amazon Redshift Spark connector on AWS Glue.

Connessione a Snowflake in AWS Glue Studio

> **Note**
>
> È possibile utilizzare AWS Glue per Spark per leggere e scrivere su tabelle in Snowflake in AWS Glue 4.0 e versioni successive. Per configurare una connessione Snowflake con i processi AWS Glue a livello di programmazione, consulta la pagina Connessioni Redshift.

AWS Glue fornisce il supporto integrato per Snowflake. AWS Glue Studio fornisce un'interfaccia visiva per connettersi a Snowflake, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

Creazione di una connessione Snowflake

Quando si aggiunge un nodo Origine dati - Snowflake in AWS Glue Studio, è possibile scegliere una connessione AWS Glue Snowflake esistente o crearne una nuova. È necessario scegliere un tipo di connessione `SNOWFLAKE` e non un tipo di connessione `JDBC`configurato per la connessione a Snowflake. Per creare una connessione AWS Glue Snowflake, utilizza la procedura seguente:

Creazione di una connessione Snowflake

1. In Snowflake, genera un utente, *sowflakeUser*, e una password, *snowflakePassword*.
2. Determina con quale warehouse Snowflake interagirà questo utente, *snowflakeWarehouse*. Puoi impostarlo come `DEFAULT_WAREHOUSE` per *snowflakeUser* in Snowflake o ricordarlo per il passaggio successivo.
3. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali Snowflake. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.
   - Quando selezioni le coppie chiave/valore, crea una coppia per *snowflakeUser* con la chiave `sfUser`.
   - Quando selezioni le coppie chiave/valore, crea una coppia per *snowflakePassword* con la chiave `sfPassword`.
   - Quando selezioni le coppie chiave/valore, crea una coppia per *snowflakeWarehouse* con la chiave `sfWarehouse`. Questo non è necessario se in Snowflake è impostato un valore predefinito.
4. In Catalogo dati AWS Glue, crea una connessione seguendo i passaggi descritti in Adding an AWS Glue connection. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per il passaggio successivo.
   - In Tipo di connessione, seleziona Snowflake.
   - In URL Snowflake, fornisci il nome host dell'istanza Snowflake. L'URL utilizzerà un nome host nel modulo *account_identifier*`.snowflakecomputing.com`.
   - Quando selezioni il Segreto AWS, fornisci *secretName*.

Creazione di un nodo di origine Snowflake

Autorizzazioni necessarie

I processi AWS Glue Studio che utilizzano origini dati Snowflake richiedono autorizzazioni aggiuntive. Per ulteriori informazioni su come aggiungere autorizzazioni ai processi ETL, consulta la pagina Review IAM permissions needed for ETL jobs.

Le connessioni SNOWFLAKE di AWS Glue utilizzano un segreto AWS Secrets Manager per fornire informazioni sulle credenziali. I tuoi ruoli di processo e di anteprima dei dati AWS Glue Studio devono essere autorizzati a leggere questo segreto.

Aggiunta di un'origine dati Snowflake

Prerequisiti:

- Un segreto AWS Secrets Manager per le tue credenziali Snowflake
- Una connessione a Catalogo dati AWS Glue di tipo Snowflake

Per aggiungere un nodo Origine dati: Snowflake:

1. Scegli la connessione per la tua origine dati Snowflake. Ciò presuppone che la connessione esista già e che sia possibile effettuare una selezione tra le connessioni esistenti. Se hai bisogno di creare una connessione, scegli Crea connessione Snowflake. Per ulteriori informazioni, consulta la pagina Overview of using connectors and connections.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà. Le informazioni sulla connessione sono visibili, tra cui URL, gruppi di sicurezza, sottorete, zona di disponibilità, descrizione, nonché timestamp di creazione (UTC) e ultimo aggiornamento (UTC).
2. Scegli un'opzione di origine Snowflake:

   - Scegli una singola tabella: questa è la tabella che contiene i dati a cui desideri accedere da una singola tabella Snowflake.
   - Inserisci una query personalizzata: ti consente di accedere a un set di dati da più tabelle Snowflake in base alla tua query personalizzata.
3. Se hai scelto una singola tabella, inserisci il nome di uno schema Snowflake.

   In alternativa, scegli Inserisci query personalizzata. Scegli questa opzione per accedere a un set di dati personalizzato da più tabelle Snowflake. Se scegli questa opzione, inserisci la query Snowflake.
4. In Prestazioni e sicurezza (facoltativo),

   - Abilita il push down delle query: scegli se vuoi trasferire il processo sull'istanza Snowflake.
5. In Proprietà personalizzate di Snowflake (facoltativo), inserisci i parametri e i valori necessari.

Creazione di un nodo di destinazione Snowflake

Autorizzazioni necessarie

I processi AWS Glue Studio che utilizzano origini dati Snowflake richiedono autorizzazioni aggiuntive. Per ulteriori informazioni su come aggiungere autorizzazioni ai processi ETL, consulta la pagina Review IAM permissions needed for ETL jobs.

Le connessioni `SNOWFLAKE` di AWS Glue utilizzano un segreto AWS Secrets Manager per fornire informazioni sulle credenziali. I tuoi ruoli di processo e di anteprima dei dati AWS Glue Studio devono essere autorizzati a leggere questo segreto.

Aggiunta di una destinazione dati Snowflake

Per creare un nodo di destinazione Snowflake:

1. Scegli una tabella Snowflake esistente come destinazione o inserisci un nuovo nome per la tabella.
2. Quando utilizzi il nodo di destinazione Destinazione dati - Snowflake, puoi scegliere tra le seguenti opzioni:

   - AGGIUNGI: se esiste già una tabella, scarica tutti i nuovi dati nella tabella come inserto. Se la tabella non esiste, procedi alla sua creazione e quindi inserisci tutti i nuovi dati.
   - MERGE: AWS Glue aggiorna o aggiunge i dati alla tabella di destinazione in base alle condizioni specificate.

     Scegli le opzioni:

     - Scegli chiavi e operazioni semplici: scegli le colonne da utilizzare come chiavi di corrispondenza tra i dati di origine e il set di dati di destinazione.

       Specifica le seguenti opzioni in caso di corrispondenza:

       - Aggiorna il record nel set di dati di destinazione con i dati dell'origine.
       - Elimina il record nel set di dati di destinazione.

       Specifica le seguenti opzioni in caso di mancata corrispondenza:

       - Inserisci i dati di origine come nuova riga nel set di dati di destinazione.
       - Non fare nulla.
     - Inserisci un'istruzione MERGE personalizzata: puoi quindi scegliere Convalida l'istruzione MERGE per verificare che l'istruzione sia valida o non valida.

- TRUNCATE: se esiste già una tabella, tronca i dati della tabella cancellando prima il contenuto della tabella di destinazione. Se il troncamento ha esito positivo, inserisci tutti i dati. Se la tabella non esiste, procedi alla sua creazione e quindi inserisci tutti i dati. Se il troncamento non va a buon fine, l'operazione non andrà a buon fine.
- DROP: se esiste già una tabella, elimina i metadati e i dati della tabella. Se l'eliminazione ha esito positivo, inserisci tutti i dati. Se la tabella non esiste, procedi alla sua creazione e quindi inserisci tutti i dati. Se l'eliminazione non va a buon fine, l'operazione non andrà a buon fine.

Opzioni avanzate

Consulta la pagina Snowflake connections nella Guida per gli sviluppatori di AWS Glue.

Connessione a BigQuery in AWS Glue Studio

> **Note**
>
> È possibile utilizzare AWS Glue per Spark per leggere e scrivere su tabelle in Google BigQuery in AWS Glue Glue 4.0 e versioni successive. Per configurare Google BigQuery con processi AWS Glue a livello di programmazione, consulta Connessioni BigQuery.

AWS Glue Studio fornisce un'interfaccia visiva per connettersi a BigQuery, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

Creazione di una connessione BigQuery

Per connetterti a Google BigQuery da AWS Glue, dovrai creare e archiviare le tue credenziali di Google Cloud Platform in un segreto AWS Secrets Manager, quindi associare tale segreto a una connessione AWS Glue Google BigQuery.

Per configurare una connessione a BigQuery:

1. In Google Cloud Platform, crea e identifica le risorse pertinenti:

   - Crea o identifica un progetto GCP contenente tabelle BigQuery a cui desideri connetterti.
   - Abilita l'API BigQuery. Per ulteriori informazioni, consulta la pagina Utilizzo dell'API di lettura dell'archiviazione BigQuery per leggere i dati delle tabelle .

2. In Google Cloud Platform, crea ed esporta le credenziali dell'account del servizio:

   Puoi utilizzare la procedura guidata per le credenziali di BigQuery per accelerare questo passaggio: Crea credenziali.

   Per creare un account di servizio in GCP, segui il tutorial disponibile in Creazione di account di servizio.

   - Quando selezioni il progetto, seleziona il progetto contenente la tua tabella BigQuery.
   - Quando selezioni i ruoli IAM di GCP per il tuo account di servizio, aggiungi o crea un ruolo che conceda le autorizzazioni appropriate per eseguire processi BigQuery per leggere, scrivere o creare tabelle BigQuery.

   Per creare le credenziali per il tuo account di servizio, segui il tutorial disponibile in Creazione della chiave di un account di servizio.

   - Quando selezioni il tipo di chiave, seleziona JSON.

   Ora dovresti avere scaricato un file JSON con le credenziali per il tuo account di servizio. La schermata visualizzata dovrebbe risultare simile a quella nell'immagine seguente:

```
{
  "type": "service_account",
  "project_id": "*****",
  "private_key_id": "*****",
  "private_key": "*****",
  "client_email": "*****",
  "client_id": "*****",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "*****",
```

```
  "universe_domain": "googleapis.com"
}
```

3. base64 codifica il tuo file di credenziali scaricato. In una sessione AWS CloudShell o simile, puoi farlo dalla riga di comando eseguendo `cat` *`credentialsFile.json`* `| base64 -w 0`. Conserva l'output di questo comando, *`credentialString`*.
4. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali di Google Cloud Platform. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `credentials` con il valore *`credentialString`*.
5. Nel Catalogo dati AWS Glue, crea una connessione seguendo i passaggi riportati in https://docs.aws.amazon.com/glue/latest/dg/console-connections.html. Dopo aver creato la connessione, prendi nota del nome, *`connectionName`*, per il passaggio successivo.
   - Quando selezioni un tipo di connessione, seleziona Google BigQuery.
   - Quando selezioni il Segreto AWS, fornisci *`secretName`*.
6. Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *`secretName`*.
7. Nella configurazione del processo AWS Glue, fornisci *`connectionName`* come Connessione di rete aggiuntiva.

Creazione di un nodo di origine BigQuery

Prerequisiti necessari

- Una connessione al Catalogo dati AWS Glue di tipo BigQuery
- Un segreto AWS Secrets Manager per le credenziali di Google BigQuery, utilizzato dalla connessione.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Il nome e il set di dati della tabella e del progetto Google Cloud corrispondente che si desidera leggere.

Aggiunta di un'origine dati BigQuery

Per aggiungere un nodo Origine dati: BigQuery:

1. Scegli la connessione per la tua origine dati BigQuery. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione BigQuery. Per ulteriori informazioni, consulta la pagina Overview of using connectors and connections.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Identifica quali dati BigQuery vorresti leggere, quindi scegli un'opzione Origine BigQuery
   - Scegli una singola tabella: ti consente di estrarre tutti i dati da una tabella.
   - Inserisci una query personalizzata: ti consente di personalizzare i dati recuperati fornendo una query.
3. Descrivi i dati che desideri leggere

   (Obbligatorio) imposta il Progetto padre sul progetto contenente la tabella o su un progetto padre di fatturazione, se pertinente.

   Se hai scelto una singola tabella, imposta Tabella sul nome di una tabella di Google BigQuery nel seguente formato: `[dataset].[table]`

   Se hai scelto una query, forniscila a Query. Nella tua query, fai riferimento alle tabelle con il loro nome di tabella completo, nel formato: `[project].[dataset].[tableName]`.
4. Fornisci le proprietà di BigQuery

   Se hai scelto una singola tabella, non è necessario fornire proprietà aggiuntive.

   Se hai scelto una query, devi fornire le seguenti proprietà personalizzate di Google BigQuery:
   - Imposta `viewsEnabled` su true.
   - Imposta `materializationDataset` su un set di dati. Il principale GCP autenticato dalle credenziali fornite tramite la connessione AWS Glue deve essere in grado di creare tabelle in questo set di dati.

Creazione di un nodo di destinazione BigQuery

Prerequisiti necessari

- Una connessione al Catalogo dati AWS Glue di tipo BigQuery
- Un segreto AWS Secrets Manager per le credenziali di Google BigQuery, utilizzato dalla connessione.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Il nome e il set di dati della tabella e del progetto Google Cloud corrispondente su cui desideri scrivere.

Aggiunta di una destinazione dati BigQuery

Per aggiungere un nodo destinazione dati: BigQuery:

1. Scegli la connessione per la tua destinazione dati BigQuery. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione BigQuery. Per ulteriori informazioni, consulta la pagina Overview of using connectors and connections.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Identifica la tabella BigQuery su cui vorresti scrivere, quindi scegli un metodo di scrittura.

   - Diretto: scrive direttamente su BigQuery utilizzando l'API BigQuery Storage Write.
   - Indiretto: scrive su Google Cloud Storage, quindi copia su BigQuery.

   Se desideri scrivere in modo indiretto, fornisci una posizione GCS di destinazione con un bucket GCS temporaneo. Dovrai fornire una configurazione aggiuntiva nella tua connessione AWS Glue. Per ulteriori informazioni, consulta la pagina Utilizzo della scrittura indiretta con Google BigQuery.
3. Descrivi i dati che desideri leggere

   (Obbligatorio) imposta il Progetto padre sul progetto contenente la tabella o su un progetto padre di fatturazione, se pertinente.

Se hai scelto una singola tabella, imposta Tabella sul nome di una tabella di Google BigQuery nel seguente formato: `[dataset].[table]`

Opzioni avanzate

È possibile fornire opzioni avanzate durante la creazione di un nodo BigQuery. Queste opzioni sono le stesse disponibili durante la programmazione di AWS Glue per gli script Spark.

Consulta il riferimento alle opzioni di connessione BigQuery nella guida per sviluppatori AWS Glue.

Connessione a Vertica in AWS Glue Studio

AWS Glue fornisce il supporto integrato per Vertica. AWS Glue Studiofornisce un'interfaccia visiva per connettersi a Vertica, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

Creazione di una connessione Vertica

Prerequisiti:

- Un bucket o una cartella Amazon S3 da utilizzare per l'archiviazione temporanea durante la lettura e la scrittura sul database, a cui fa riferimento *`tempS3Path`*.

> **Note**
>
> Quando utilizzi Vertica nelle anteprime dei dati del processo di AWS Glue, i file temporanei potrebbero non essere rimossi automaticamente da *`tempS3Path`*. Per garantire la rimozione dei file temporanei, interrompi direttamente la sessione di anteprima dei dati scegliendo Termina sessione nel riquadro Anteprima dei dati.
> Se non sei in grado di terminare direttamente la sessione di anteprima dei dati, valuta la possibilità di impostare la configurazione del ciclo di vita di Amazon S3 per rimuovere i dati

obsoleti. Consigliamo di rimuovere i dati più vecchi di 49 ore, in base al runtime massimo del processo in aggiunta a un margine. Per ulteriori informazioni sulla configurazione del ciclo di vita di Amazon S3, consulta Gestione del ciclo di vita dello storage nella documentazione di Amazon S3.

- Una policy IAM con autorizzazioni appropriate per il percorso Amazon S3 che puoi associare al ruolo di processo AWS Glue.
- Se la tua istanza VPC si trova in un Amazon VPC, configura Amazon VPC per consentire al processo AWS Glue di comunicare con l'istanza Vertica senza che il traffico attraversi la rete Internet pubblica.

  In Amazon VPC, identifica o crea un VPC, una sottorete e un gruppo di sicurezza che AWS Glue utilizzerà durante l'esecuzione del processo. Inoltre, assicurati che Amazon VPC sia configurato per consentire il traffico di rete tra l'istanza Vertica e questa posizione. Il tuo processo dovrà stabilire una connessione TCP con la tua porta del client Vertica, (per impostazione predefinita, 5433). In base al layout di rete, potrebbero essere necessarie modifiche alle regole dei gruppi di sicurezza, alle liste di controllo accessi di rete, ai gateway NAT e alle connessioni peering.

Per configurare una connessione a Vertica:

1. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali Vertica, *verticaUsername* e *verticaPassword*. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `user` con il valore *verticaUsername*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *verticaPassword*.
2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per il passaggio successivo.

   - In Tipo di connessione, seleziona Vertica.
   - In Host Vertica, fornisci il nome host dell'installazione Vertica.

- In Porta Vertica, indica la porta tramite cui è disponibile l'installazione di Vertica.
- Quando selezioni il Segreto AWS, fornisci *secretName*.

3. Nelle seguenti situazioni, potresti aver bisogno di una configurazione aggiuntiva:

   - Per le istanze Vertica ospitate su AWS in un Amazon VPC
     - Fornisci le informazioni di connessione Amazon VPC alla connessione AWS Glue che definisce le credenziali di sicurezza Vertica. Durante la creazione o l'aggiornamento della connessione, imposta VPC, sottorete e Gruppi di sicurezza nelle opzioni di rete.

Prima di eseguire il processo AWS Glue, è necessario eseguire le seguenti operazioni:

- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso per *tempS3Path*.
- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *secretName*.

Creazione di un nodo di origine Vertica

Prerequisiti necessari

- Una connessione a Catalogo dati AWS Glue di tipo Vertica, *connectionName* e una posizione Amazon S3 temporanea, *tempS3Path*, come descritto nella sezione precedente the section called “Creazione di una connessione Vertica”.
- Una tabella Vertica da cui desideri leggere, *tableName* o interrogare *targetQuery*.

Aggiunta di un'origine dati Vertica

Per aggiungere un nodo origine dati: Vertica:

1. Scegli la connessione per la tua origine dati Vertica. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione Vertica. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione Vertica” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Scegli il Database contenente la tabella.
3. Scegli l'area di staging in Amazon S3, inserisci un URI S3A in *tempS3Path*.

4. Scegli Origine Vertica.

   - Scegli una singola tabella: accedi a tutti i dati da un'unica tabella.
   - Inserisci una query personalizzata: accedi a un set di dati da più tabelle in base alla tua query personalizzata.

5. Se hai scelto una singola tabella, inserisci *`tableName`* e, facoltativamente, seleziona uno schema

   Se hai scelto Inserisci una query personalizzata, inserisci una query SQL SELECT e, facoltativamente, seleziona uno schema.

6. In Proprietà personalizzate di Vertica, inserisci i parametri e i valori necessari.

Creazione di un nodo di destinazione Vertica

Prerequisiti necessari

- Una connessione a Catalogo dati AWS Glue di tipo Vertica, *`connectionName`* e una posizione Amazon S3 temporanea, *`tempS3Path`*, come descritto nella sezione precedente the section called “Creazione di una connessione Vertica”.

Aggiunta di una destinazione dati Vertica

Per aggiungere un nodo destinazione dati: Vertica:

1. Scegli la connessione per la tua origine dati Vertica. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione Vertica. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione Vertica” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.

2. Scegli il Database contenente la tabella.
3. Scegli l'area di staging in Amazon S3, inserisci un URI S3A in *`tempS3Path`*.
4. Immetti *`tableName`* e, facoltativamente, seleziona uno schema.
5. In Proprietà personalizzate di Vertica, inserisci i parametri e i valori necessari.

Opzioni avanzate

È possibile fornire opzioni avanzate durante la creazione di un nodo Vertica. Queste opzioni sono le stesse disponibili durante la programmazione di AWS Glue per gli script Spark.

Per informazioni, consultare the section called “Connessioni Vertica”.

Connessione a SAP HANA in AWS Glue Studio

AWS Glue fornisce il supporto integrato per SAP HANA. AWS Glue Studio fornisce un'interfaccia visiva per connettersi a SAP HANA, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

Creazione di una connessione SAP HANA

Per connetterti a SAP HANA da AWS Glue, dovrai creare e archiviare le tue credenziali SAP HANA in un segreto AWS Secrets Manager, quindi associare tale segreto a una connessione AWS Glue SAP HANA. Dovrai configurare la connettività di rete tra il tuo servizio SAP HANA e AWS Glue.

Prerequisiti:

- Se il tuo servizio SAP HANA si trova in un Amazon VPC, configura Amazon VPC per consentire al processo AWS Glue di comunicare con il servizio SAP HANA senza che il traffico attraversi la rete Internet pubblica.

  In Amazon VPC, identifica o crea un VPC, una sottorete e un gruppo di sicurezza che AWS Glue utilizzerà durante l'esecuzione del processo. Inoltre, assicurati che Amazon VPC sia configurato per consentire il traffico di rete tra l'endpoint SAP HANA e questa posizione. Il tuo processo dovrà stabilire una connessione TCP con la tua porta SAP HANA JDBC. Per ulteriori informazioni sulle porte SAP HANA, consulta la documentazione SAP HANA. In base al layout di rete, potrebbe richiedere modifiche alle regole dei gruppi di sicurezza, alle liste di controllo accessi di rete, ai gateway NAT e alle connessioni peering.

Per configurare una connessione a SAP HANA:

1. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali SAP HANA. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `user` con il valore *saphanaUsername*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *saphanaPassword*.

2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per l'uso futuro in AWS Glue.

   - In Tipo di connessione, seleziona SAP HANA.
   - Quando fornisci l'URL SAP HANA, fornisci l'URL per la tua istanza.

     Gli URL SAP HANA JDBC sono nel modulo `jdbc:sap://`*saphanaHostname*`:`*saphanaPort*`/?`*databaseName*`=`*saphanaDBname*`,`*Paramete*

     AWS Glue richiede i seguenti parametri URL JDBC:

     - `databaseName` - Un database predefinito in SAP HANA a cui connettersi.
   - Quando selezioni il Segreto AWS, fornisci *secretName*.

Dopo aver creato una connessione AWS Glue SAP HANA, è necessario eseguire le seguenti operazioni prima di eseguire il processo AWS Glue:

- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *secretName*.

Creazione di un nodo di origine SAP HANA

Prerequisiti necessari

- Una connessione SAP HANA AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione SAP HANA”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.

- Una tabella SAP HANA da cui desideri leggere, *tableName* o interrogare *targetQuery*.

  Una tabella può essere specificata con un nome di tabella SAP HANA e di schema, nel modulo *schemaName*.*tableName*. Il nome dello schema e il separatore "." non sono necessari se la tabella si trova nello schema predefinito, "pubblico". Chiamalo *tableIdentifier*. Il database viene fornito come parametro URL JDBC in `connectionName`.

Aggiunta di un'origine dati SAP HANA

Per aggiungere un nodo origine dati: SAP HANA:

1. Scegli la connessione per la tua origine dati SAP HANA. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione SAP HANA. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione SAP HANA” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Scegli un'opzione origine SAP HANA:

   - Scegli una singola tabella: accedi a tutti i dati da un'unica tabella.
   - Inserisci una query personalizzata: accedi a un set di dati da più tabelle in base alla tua query personalizzata.
3. Se hai scelto una singola tabella, inserisci *tableName*.

   Se hai scelto Inserisci una query personalizzata, inserisci una query SQL SELECT.
4. In Proprietà personalizzate di SAP HANA, inserisci i parametri e i valori necessari.

Creazione di un nodo di destinazione SAP HANA

Prerequisiti necessari

- Una connessione SAP HANA AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione SAP HANA”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Una tabella SAP HANA su cui scrivere, *tableName*.

Una tabella può essere specificata con un nome di tabella SAP HANA e di schema, nel modulo *schemaName*.*tableName*. Il nome dello schema e il separatore "." non sono necessari se la tabella si trova nello schema predefinito, "pubblico". Chiamalo *tableIdentifier*. Il database viene fornito come parametro URL JDBC in `connectionName`.

Aggiunta di una destinazione di dati SAP HANA

Per aggiungere un nodo destinazione dati: SAP HANA:

1. Scegli la connessione per la tua origine dati SAP HANA. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione SAP HANA. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione SAP HANA” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Configura il nome della tabella fornendo *tableName*.
3. In Proprietà personalizzate di Teradata, inserisci i parametri e i valori necessari.

Opzioni avanzate

È possibile fornire opzioni avanzate durante la creazione di un nodo SAP HANA. Queste opzioni sono le stesse disponibili durante la programmazione di AWS Glue per gli script Spark.

Per informazioni, consultare the section called “Connessioni SAP HANA”.

Connessione ad Azure SQL in AWS Glue Studio

AWS Glue fornisce il supporto integrato per Azure SQL. AWS Glue Studio fornisce un'interfaccia visiva per connettersi ad Azure SQL, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

- Creazione di una connessione Azure SQL
- Creazione di un nodo sorgente di Azure SQL
- Creazione di un nodo destinazione di Azure SQL

- Opzioni avanzate

Creazione di una connessione Azure SQL

Per connetterti ad Azure SQL da AWS Glue, dovrai creare e archiviare le tue credenziali Azure SQL in un segreto AWS Secrets Manager, quindi associare quel segreto a una connessione Azure SQL AWS Glue.

Per configurare una connessione ad Azure SQL:

1. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali Azure SQL. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `user` con il valore *`azuresqlUsername`*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *`azuresqlPassword`*.

2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *`connectionName`*, per l'uso futuro in AWS Glue.

   - In Tipo di connessione, seleziona Azure SQL.
   - Quando fornisci l'URL SQL di Azure, fornisci un URL di endpoint JDBC.

     L'elenco deve essere nel seguente formato:
     `jdbc:sqlserver://`*`databaseServerName`*`:`*`databasePort`*`;databaseName=`*`azuresqlDBnam`*

     AWS Glue richiede le seguenti proprietà URL:

     - `databaseName` - Un database predefinito in Azure SQL a cui connettersi.

     Per altre informazioni sugli URL JDBC per le istanze gestite di Azure SQL, consulta la documentazione di Microsoft.

   - Quando selezioni il Segreto AWS, fornisci *`secretName`*.

Creazione di un nodo sorgente di Azure SQL

Prerequisiti necessari

- Una connessione Azure SQL AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione Azure SQL”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Una tabella Azure SQL da cui si desidera leggere, *`tableName`*.

  Una tabella SQL di Azure è identificata dal database, dallo schema e dal nome. È necessario fornire il nome del database e della tabella durante la connessione ad Azure SQL. È inoltre necessario fornire lo schema se diverso da quello predefinito, "pubblico". Il database viene fornito tramite una proprietà URL in *`connectionName`*, il nome dello schema e della tabella tramite `dbtable`.

Aggiungere un'origine dati di Azure SQL

Per aggiungere un nodo di origine dati: Azure SQL:

1. Scegli la connessione per la tua origine dati Azure SQL. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea una connessione Azure SQL. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione Azure SQL” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Scegli un'opzione Origine Azure SQL:

   - Scegli una singola tabella: accedi a tutti i dati da un'unica tabella.
   - Inserisci una query personalizzata: accedi a un set di dati da più tabelle in base alla tua query personalizzata.
3. Se hai scelto una singola tabella, inserisci *`tableName`*.

   Se hai scelto Inserisci una query personalizzata, inserisci una query TransactSQL SELECT.
4. In Proprietà personalizzate di Azure SQL, inserisci i parametri e i valori necessari.

Creazione di un nodo destinazione di Azure SQL

Prerequisiti necessari

- Una connessione Azure SQL AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione Azure SQL”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Una tabella di Azure SQL su cui scrivere, *`tableName`*.

  Una tabella SQL di Azure è identificata dal database, dallo schema e dal nome. È necessario fornire il nome del database e della tabella durante la connessione ad Azure SQL. È inoltre necessario fornire lo schema se diverso da quello predefinito, "pubblico". Il database viene fornito tramite una proprietà URL in *`connectionName`*, il nome dello schema e della tabella tramite `dbtable`.

Aggiungere una destinazione dati di Azure SQL

Per aggiungere un nodo di destinazione dati: Azure SQL:

1. Scegli la connessione per la tua origine dati Azure SQL. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea una connessione Azure SQL. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione Azure SQL” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Configura il nome della tabella fornendo *`tableName`*.
3. In Proprietà personalizzate di Azure SQL, inserisci i parametri e i valori necessari.

Opzioni avanzate

È possibile fornire opzioni avanzate durante la creazione di un nodo Azure SQL. Queste opzioni sono le stesse disponibili durante la programmazione di AWS Glue per gli script Spark.

Per informazioni, consultare the section called “Connessioni Azure SQL”.

Connessione a MongoDB in AWS Glue Studio

AWS Glue fornisce il supporto integrato per MongoDB. AWS Glue Studio fornisce un'interfaccia visiva per connettersi a MongoDB, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

Creazione di una connessione MongoDB

Prerequisiti:

- Se la tua istanza MongoDB si trova in un Amazon VPC, configura Amazon VPC per consentire al processo AWS Glue di comunicare con l'istanza MongoDB senza che il traffico attraversi la rete Internet pubblica.

  In Amazon VPC, identifica o crea un VPC, una sottorete e un gruppo di sicurezza che AWS Glue utilizzerà durante l'esecuzione del processo. Inoltre, assicurati che Amazon VPC sia configurato per consentire il traffico di rete tra l'istanza MongoDB e questa posizione. In base al layout di rete, potrebbe richiedere modifiche alle regole dei gruppi di sicurezza, alle liste di controllo accessi di rete, ai gateway NAT e alle connessioni peering.

Per configurare una connessione a MongoDB:

1. Facoltativamente, in AWS Secrets Manager, crea un segreto utilizzando le tue credenziali MongoDB. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `username` con il valore *`mongodbUser`*.

Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *`mongodbPass`*.

2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *`connectionName`*, per l'uso futuro in AWS Glue.

   - Quando selezioni un tipo di connessione, seleziona MongoDB o MongoDB Atlas.
   - Quando selezioni l'URL MongoDB o URL MongoDB Atlas, fornisci il nome host dell'istanza MongoDB.

     Un URL MongoDB viene fornito nel formato `mongodb://`*`mongoHost`*`:`*`mongoPort`*`/`*`mongoDBname`*.

     Un URL MongoDB Atlas viene fornito nel formato `mongodb+srv://`*`mongoHost`*`:`*`mongoPort`*`/`*`mongoDBname`*.

     Fornendo il database predefinito per la connessione, *`mongoDBname`* è facoltativo.
   - Se hai scelto di creare un segreto di Secrets Manager, scegli il tipo di credenziali AWS Secrets Manager.

     Quindi, in Segreto di AWS, fornisci *`secretName`*.
   - Se scegli di fornire nome utente e password, fornisci *`mongodbUser`* e *`mongodbPass`*.
3. Nelle seguenti situazioni, potresti aver bisogno di una configurazione aggiuntiva:

   - Per le istanze MongoDB ospitate su AWS in un Amazon VPC
     - Dovrai fornire le informazioni di connessione Amazon VPC alla connessione AWS Glue che definisce le credenziali di sicurezza MongoDB. Durante la creazione o l'aggiornamento della connessione, imposta VPC, sottorete e Gruppi di sicurezza nelle opzioni di rete.

Dopo aver creato una connessione AWS Glue MongoDB, è necessario eseguire le seguenti operazioni prima di eseguire il processo AWS Glue:

- Quando lavori con processi AWS Glue nell'editor visivo, devi fornire le informazioni sulla connessione Amazon VPC affinché il processo possa connettersi a MongoDB. Identifica una posizione adatta in Amazon VPC e forniscila alla tua connessione MongoDB AWS Glue.

- Se hai scelto di creare un segreto di Secrets Manager, concedi al ruolo IAM associato al processo AWS Glue il permesso di leggere *`secretName`*.

Creazione di un nodo di origine MongoDB

Prerequisiti necessari

- Una connessione MongoDB AWS Glue, come descritto nella sezione precedente the section called “Creazione di una connessione MongoDB”.
- Se hai scelto di creare un segreto di Secrets Manager, autorizzazioni appropriate sul tuo processo per leggere il segreto usato dalla connessione.
- Una raccolta MongoDB da cui desideri leggere. Avrai bisogno delle informazioni di identificazione per la raccolta.

  Una raccolta MongoDB è identificata da un nome di database e di raccolta, *`mongodbName`*, *`mongodbCollection`*.

Aggiunta di un'origine dati MongoDB

Per aggiungere un nodo origine dati: MongoDB:

1. Scegli la connessione per la tua origine dati MongoDB. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione MongoDB. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione MongoDB” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Scegli un Database. Inserisci *`mongodbName`*.
3. Scegli una Raccolta. Inserisci *`mongodbCollection`*.
4. Scegli il tuo partizionatore, la dimensione della partizione (MB) e la chiave di partizione. Per ulteriori informazioni sui parametri di partizione, consulta the section called “"connectionType": "mongodb" come sorgente”.
5. In Proprietà personalizzate di MongoDB, inserisci i parametri e i valori necessari.

Creazione di un nodo di destinazione MongoDB

Prerequisiti necessari

- Una connessione MongoDB AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione MongoDB”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Una tabella MongoDB su cui scrivere, *`tableName`*.

Aggiunta di una destinazione dati MongoDB

Per aggiungere un nodo di destinazione dati: MongoDB:

1. Scegli la connessione per la tua origine dati MongoDB. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione MongoDB. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione MongoDB” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Scegli un Database. Inserisci *`mongodbName`*.
3. Scegli una Raccolta. Inserisci *`mongodbCollection`*.
4. Scegli il tuo partizionatore, la dimensione della partizione (MB) e la chiave di partizione. Per ulteriori informazioni sui parametri di partizione, consulta the section called “"connectionType": "mongodb" come sorgente”.
5. Se lo desideri, scegli Riprova a scrivere.
6. In Proprietà personalizzate di MongoDB, inserisci i parametri e i valori necessari.

Opzioni avanzate

È possibile fornire opzioni avanzate durante la creazione di un nodo MongoDB. Queste opzioni sono le stesse disponibili durante la programmazione di AWS Glue per gli script Spark.

Per informazioni, consultare the section called “Connessione MongoDB”.

Connessione ad Azure Cosmos DB in AWS Glue Studio

AWS Glue fornisce il supporto integrato per Azure Cosmos DB. AWS Glue Studio fornisce un'interfaccia visiva per connettersi ad Azure Cosmos DB per NoSQL, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

Creazione di una connessione Azure Cosmos DB

Prerequisiti:

- In Azure, dovrai identificare o generare una chiave di Azure Cosmos DB da usare da AWS Glue, `cosmosKey`. Per altre informazioni, consulta Accesso sicuro ai dati in Azure Cosmos DB nella documentazione di Azure.

Per configurare una connessione ad Azure Cosmos DB:

1. In AWS Secrets Manager, crea un segreto utilizzando la tua chiave Azure Cosmos DB. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `spark.cosmos.accountKey` con il valore *`cosmosKey`*.
2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *`connectionName`*, per l'uso futuro in AWS Glue.
   - In Tipo di connessione, seleziona Azure Cosmos DB.
   - Quando selezioni il Segreto AWS, fornisci *`secretName`*.

Creazione di un nodo sorgente di Azure Cosmos DB

Prerequisiti necessari

- Una connessione Azure Cosmos DB AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione Azure Cosmos DB”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Un container Azure Cosmos DB per NoSQL da cui desideri leggere. Avrai bisogno delle informazioni di identificazione per il container.

  Un container Azure Cosmos per NoSQL è identificato dal database e dal container. È necessario fornire i nomi del database, *cosmosDBName*, e del container, *cosmosContainerName*, quando ci si connette all'API di Azure Cosmos per NoSQL.

Aggiungere un'origine dati di Azure Cosmos DB

Per aggiungere un nodo di origine dati: Azure Cosmos DB:

1. Scegli la connessione per la tua origine dati Azure Cosmos DB. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea una connessione Azure Cosmos DB. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione Azure Cosmos DB” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Scegli un nome del database Cosmos DB: fornisci il nome del database da cui desideri leggere, *CosmosDBName*.
3. Scegli il container Azure Cosmos DB: fornisci il nome del container da cui vuoi leggere, *cosmosContainerName*.
4. Facoltativamente, scegli Query personalizzata per Azure Cosmos DB: fornisci una query SQL SELECT per recuperare informazioni specifiche da Azure Cosmos DB.
5. In Proprietà personalizzate di Azure Cosmos, inserisci i parametri e i valori necessari.

Creazione di un nodo destinazione di Azure Cosmos DB

Prerequisiti necessari

- Una connessione Azure Cosmos DB AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione Azure Cosmos DB”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Una tabella di Azure Cosmos DB su cui scrivere. Avrai bisogno delle informazioni di identificazione per il container. È necessario creare il container prima di chiamare il metodo di connessione.

  Un container Azure Cosmos per NoSQL è identificato dal database e dal container. È necessario fornire i nomi del database, *`cosmosDBName`*, e del container, *`cosmosContainerName`*, quando ci si connette all'API di Azure Cosmos per NoSQL.

Aggiungere una destinazione dati di Azure Cosmos DB

Per aggiungere un nodo di destinazione dati: Azure Cosmos DB:

1. Scegli la connessione per la tua origine dati Azure Cosmos DB. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea una connessione Azure Cosmos DB. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione Azure Cosmos DB” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Scegli un nome del database Cosmos DB: fornisci il nome del database da cui desideri leggere, *`CosmosDBName`*.
3. Scegli il container Azure Cosmos DB: fornisci il nome del container da cui vuoi leggere, *`cosmosContainerName`*.
4. In Proprietà personalizzate di Azure Cosmos, inserisci i parametri e i valori necessari.

Opzioni avanzate

È possibile fornire opzioni avanzate durante la creazione di un nodo Azure Cosmos DB. Queste opzioni sono le stesse disponibili durante la programmazione di AWS Glue per gli script Spark.

Per informazioni, consultare the section called “Connessioni Azure Cosmos DB”.

Connessione a Teradata Vantage in AWS Glue Studio

AWS Glue fornisce il supporto integrato per Teradata Vantage. AWS Glue Studio fornisce un'interfaccia visiva per connettersi a Teradata, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio.

Argomenti

Creazione di una connessione Teradata Vantage

Per connettersi a Teradata Vantage da AWS Glue, è necessario creare e archiviare le credenziali Teradata in modo AWS Secrets Manager segreto, quindi associare tale segreto a una connessione Teradata. AWS Glue

Prerequisiti:

- Se accedi al tuo ambiente Teradata tramite Amazon VPC, configura Amazon VPC per consentire al tuo AWS Glue job di comunicare con l'ambiente Teradata. Sconsigliamo l'accesso all'ambiente Teradata tramite la rete Internet pubblica.

  In Amazon VPC, identifica o crea un VPC, una sottorete e un gruppo di sicurezza da utilizzare durante l'esecuzione del AWS Glue lavoro. Inoltre, assicurati che Amazon VPC sia configurato per consentire il traffico di rete tra l'istanza Teradata e questa posizione. Il tuo processo dovrà stabilire una connessione TCP con la tua porta del client Teradata. Per ulteriori informazioni sulle porte Teradata, consulta la documentazione di Teradata.

  In base al layout di rete, la connettività VPC sicura potrebbe richiedere modifiche ad Amazon VPC e ad altri servizi di rete. Per ulteriori informazioni sulla AWS connettività, consulta le opzioni di AWS connettività nella documentazione di Teradata.

Per configurare una connessione AWS Glue Teradata:

1. *Nella configurazione Teradata, identifica o crea un utente e la password si AWS Glue connetterà con TeradataUser e TeraDataPassword.* Per ulteriori informazioni, consulta Vantage Security Overview nella documentazione di Teradata.
2. Nel, crea un segreto utilizzando le AWS Secrets Manager tue credenziali Teradata. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Crea un AWS Secrets Manager segreto nella AWS Secrets Manager documentazione. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `user` con il valore *teradataUsername*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *teradataPassword*.
3. Nella AWS Glue console, crea una connessione seguendo la procedura riportata di seguitothe section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per il passaggio successivo.

   - In Tipo di connessione, seleziona Snowflake.
   - Quando fornisci JDBC URL, fornisci l'URL per la tua istanza. Puoi anche codificare determinati parametri di connessione, separati da virgole, nel tuo URL JDBC. L'URL deve rispettare il seguente formato: `jdbc:teradata://`*teradataHostname*`/`*ParameterName*`=`*ParameterValue*`,`*ParameterName*

     I parametri URL supportati includono:

     - `DATABASE`: nome del database sull'host a cui accedere per impostazione predefinita.
     - `DBS_PORT`: la porta del database, utilizzata con una porta non standard.
   - *Quando selezioni un* ***tipo di credenziali****, seleziona* ***AWS Secrets Manager****, quindi imposta* ***Segreto di AWS*** *su secretName*.
4. Nelle seguenti situazioni, potresti aver bisogno di una configurazione aggiuntiva:

   - Per le istanze Teradata ospitate su AWS un Amazon VPC
     - Dovrai fornire le informazioni di connessione Amazon VPC alla AWS Glue connessione che definisce le tue credenziali di sicurezza Teradata. Durante la creazione o l'aggiornamento della connessione, imposta VPC, sottorete e Gruppi di sicurezza nelle opzioni di rete.

Creazione di un nodo di origine Teradata

Prerequisiti necessari

- Una connessione Teradata Vantage AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione Teradata Vantage”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Una tabella Teradata da cui desideri leggere, *`tableName`* o interrogare *`targetQuery`*.

Aggiunta di un'origine dati Teradata

Per aggiungere un nodo origine dati: Teradata:

1. Scegli la connessione per la tua origine dati Teradata. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea una nuova connessione. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione Teradata Vantage” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Scegli un'opzione Origine Teradata:

   - Scegli una singola tabella: accedi a tutti i dati da un'unica tabella.
   - Inserisci una query personalizzata: accedi a un set di dati da più tabelle in base alla tua query personalizzata.
3. Se hai scelto una singola tabella, inserisci *`tableName`*.

   Se hai scelto Inserisci una query personalizzata, inserisci una query SQL SELECT.
4. In Proprietà personalizzate di Teradata, inserisci i parametri e i valori necessari.

Creazione di un nodo di destinazione Teradata

Prerequisiti necessari

- Una connessione Teradata Vantage AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione Teradata Vantage”.

- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Una tabella Teradata su cui scrivere, *`tableName`*.

Aggiunta di una destinazione dati Teradata

Per aggiungere un nodo destinazione dati: Teradata:

1. Scegli la connessione per la tua origine dati Teradata. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione Teradata. Per ulteriori informazioni, consulta la pagina Overview of using connectors and connections.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Configura il nome della tabella fornendo *`tableName`*.
3. In Proprietà personalizzate di Teradata, inserisci i parametri e i valori necessari.

Opzioni avanzate

È possibile fornire opzioni avanzate durante la creazione di un nodo Teradata. Queste opzioni sono le stesse disponibili durante la programmazione di AWS Glue per gli script Spark.

Per informazioni, consultare the section called “Connessioni Teradata Vantage”.

Connessione al servizio OpenSearch in AWS Glue Studio

AWS Glue fornisce il supporto integrato per il servizio OpenSearch di Amazon. AWS Glue Studio fornisce un'interfaccia visiva per connettersi al servizio OpenSearch di Amazon, creare processi di integrazione dei dati ed eseguirli sul runtime Spark serverless AWS Glue Studio. Questa funzionalità non è compatibile con il servizio OpenSearch serverless.

Argomenti

- Creazione di una connessione OpenSearch al servizio
- Creazione di un nodo di origine del servizio OpenSearch
- Creazione di un nodo di destinazione del servizio OpenSearch
- Opzioni avanzate

Creazione di una connessione OpenSearch al servizio

Prerequisiti:

- Identifica l'endpoint del dominio, *AOSEndpoint* e la porta, *AOSport* da cui desideri leggere o crea la risorsa seguendo le istruzioni nella documentazione di Amazon Service. OpenSearch Per ulteriori informazioni sulla creazione di un dominio, consulta Creazione e gestione di domini Amazon OpenSearch Service nella documentazione di Amazon OpenSearch Service.

  Un endpoint OpenSearch di dominio Amazon Service avrà il seguente modulo predefinito, https:// search - *DomainName* -. *unstructuredIdContent regione .es.amazonaws.com.* Per ulteriori informazioni sull'identificazione dell'endpoint del tuo dominio, consulta Creazione e gestione dei domini Amazon OpenSearch Service nella documentazione di Amazon OpenSearch Service.

  Identifica o genera credenziali di autenticazione di base HTTP, aosUser e aosPassword per il tuo dominio.

Per configurare una connessione al OpenSearch servizio:

1. InAWS Secrets Manager, crea un segreto utilizzando le tue credenziali OpenSearch di servizio. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `opensearch.net.http.auth.user` con il valore *aosUser*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `opensearch.net.http.auth.pass` con il valore *aosPassword*.
2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per l'uso futuro in AWS Glue.

   - Quando selezioni un tipo di connessione, seleziona OpenSearch Servizio.
   - Quando selezioni un endpoint di dominio, fornisci *aosEndpoint*.
   - Quando selezioni una porta, fornisci *aosPort*.
   - Quando selezioni il Segreto AWS, fornisci *secretName*.

Creazione di un nodo di origine del servizio OpenSearch

Prerequisiti necessari

- Una connessione al servizio OpenSearch AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione OpenSearch al servizio”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Un indice del servizio OpenSearch da cui desideri leggere, *aosIndex*.

Aggiunta di un'origine dati del servizio OpenSearch

Per aggiungere un nodo origine dati: servizio OpenSearch:

1. Scegli la connessione per la tua origine dati del servizio OpenSearch. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione servizio OpenSearch. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione OpenSearch al servizio” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Fornisci Indice, l'indice che desideri leggere.
3. Facoltativamente, fornisci Query, una query OpenSearch per fornire risultati più specifici. Per ulteriori informazioni sull'utilizzo della funzione OpenSearch, consulta the section called “Leggi dal servizio OpenSearch ”.
4. In Proprietà personalizzate del servizio OpenSearch, inserisci i parametri e i valori necessari.

Creazione di un nodo di destinazione del servizio OpenSearch

Prerequisiti necessari

- Una connessione al servizio OpenSearch AWS Glue, configurata con un segreto AWS Secrets Manager, come descritto nella sezione precedente the section called “Creazione di una connessione OpenSearch al servizio”.
- Autorizzazioni appropriate sul processo per leggere il segreto utilizzato dalla connessione.
- Un indice del servizio OpenSearch su cui scrivere, *aosIndex*.

Aggiunta di una destinazione di dati del servizio OpenSearch

Per aggiungere un nodo destinazione dati: servizio OpenSearch:

1. Scegli la connessione per la tua origine dati del servizio OpenSearch. Dato che l'hai creato, dovrebbe essere disponibile nel menu a discesa. Se devi creare una connessione, scegli Crea connessione servizio OpenSearch. Per ulteriori informazioni, consulta la sezione the section called “Creazione di una connessione OpenSearch al servizio” precedente.

   Dopo aver scelto una connessione, puoi visualizzare le proprietà della connessione facendo clic su Visualizza proprietà.
2. Fornisci Indice, l'indice che desideri leggere.
3. In Proprietà personalizzate del servizio OpenSearch, inserisci i parametri e i valori necessari.

Opzioni avanzate

È possibile fornire opzioni avanzate durante la creazione di un nodo di OpenSearch Service. Queste opzioni sono le stesse disponibili durante la programmazione di AWS Glue per gli script Spark.

Per informazioni, consultare the section called “OpenSearch Connessioni di servizio”.

Utilizzo di connettori e connessioni con AWS Glue Studio

> **Note**
>
> Per impostazione predefinita, le nuove istanze di database Amazon RDS utilizzeranno il nuovo certificato. `rds-ca-rsa2048-g1` AWS Gluejob e Test Connection si basano attualmente su. `certificate rds-ca-2019` Per connettere nuove istanze Amazon RDS con AWS Glue job o Test Connection, imposta l'istanza per utilizzare il certificato `rds-ca-2019` tramite la AWS console o. AWS CLI Per ulteriori informazioni, consulta la sezione Utilizzo di SSL/TLS per crittografare una connessione a un'istanza DB nella guida per l'utente di Amazon RDS per una guida dettagliata.

AWS Glue fornisce supporto integrato per gli archivi dati più comuni (ad esempio Amazon Redshift, Amazon Aurora, Microsoft SQL Server, MySQL, MongoDB e PostgreSQL) utilizzando le connessioni JDBC. AWS Glue consente inoltre di usare driver JDBC personalizzati nei processi di estrazione,

trasformazione e caricamento (ETL). Per gli archivi dati non supportati in modo nativo, ad esempio le applicazioni SaaS, è possibile utilizzare i connettori.

Un connettore è un pacchetto di codice opzionale che aiuta con l'accesso ai datastore in AWS Glue Studio. Puoi iscriverti a diversi connettori offerti in Marketplace AWS.

Durante la creazione di processi ETL, puoi utilizzare un datastore supportato nativamente, un connettore da Marketplace AWS o i tuoi connettori personalizzati. Se utilizzi un connettore, è innanzitutto necessario creare una connessione. Una connessione contiene le proprietà necessarie per connettersi a un particolare datastore. È possibile utilizzare la connessione con le tue origini dati e destinazioni dati nel processo ETL. Connettori e connessioni funzionano insieme per facilitare l'accesso ai datastore.

Argomenti

Panoramica sull'utilizzo di connettori e connessioni

Una connessione contiene le proprietà necessarie per connettersi a un particolare datastore. Quando crei una connessione, questa viene archiviata in AWS Glue Data Catalog. Scegli un connettore e quindi crea una connessione basata su di esso.

Puoi sottoscrivere i connettori per gli archivi dati non supportati in modo nativo in Marketplace AWS e quindi utilizzare tali connettori durante la creazione di connessioni. Gli sviluppatori possono anche creare i propri connettori ed è possibile utilizzarli durante la creazione di connessioni.

> **Note**
>
> Le connessioni create utilizzando connettori personalizzati o Marketplace AWS in AWS Glue Studio vengono visualizzati nella console AWS Glue con tipo impostato su `UNKNOWN`.

La procedura riportata di seguito illustra il processo generale di utilizzo dei connettori in AWS Glue Studio:

1. Sottoscrivi un connettore in Marketplace AWS o sviluppa il tuo connettore e caricalo su AWS Glue Studio. Per ulteriori informazioni, consulta Aggiunta di connettori a AWS Glue Studio.
2. Esamina le informazioni sull'utilizzo del connettore. Puoi trovare queste informazioni nella scheda Usage (Utilizzo) nella pagina prodotto del connettore. Ad esempio, se fai clic sulla scheda Utilizzo in questa pagina di prodotto, AWS GlueConnector for Google BigQuery, puoi vedere nella sezione Risorse aggiuntive un link a un blog sull'utilizzo di questo connettore. Altri connettori potrebbero contenere collegamenti alle istruzioni contenute nella sezione Overview (Panoramica), come mostrato nella pagina prodotto per Cloudwatch Logs connector for AWS Glue (Connettore Cloudwatch Logs per AWS Glue).
3. Crea una connessione. Puoi scegliere quale connettore utilizzare e fornire informazioni aggiuntive per la connessione, ad esempio le credenziali di accesso, le stringhe URI e le informazioni sul cloud privato virtuale (VPC). Per ulteriori informazioni, consulta Creazione di connessioni per i connettori.
4. Creare un ruolo IAM per il processo. Il processo assume le autorizzazioni del ruolo IAM specificate al momento della creazione. Questo ruolo IAM deve avere le autorizzazioni necessarie per autenticare, estrarre e scrivere dati nei datastore.
5. Crea un processo ETL e configura le proprietà dell'origine dati per il processo ETL. Fornire le opzioni di connessione e le informazioni di autenticazione secondo le istruzioni fornite dal provider di connettori personalizzati. Per ulteriori informazioni, consulta Creazione di processi con connettori personalizzati.
6. Personalizza il processo ETL aggiungendo trasformazioni o datastore aggiuntivi, come descritto in ETL visivo con AWS Glue Studio.
7. Se usi un connettore per la destinazione dati, configura le proprietà della destinazione dati per il processo ETL. Fornire le opzioni di connessione e le informazioni di autenticazione secondo le istruzioni fornite dal provider di connettori personalizzati. Per ulteriori informazioni, consulta the section called “Creazione di processi con connettori personalizzati”.

8. Personalizza l'ambiente di esecuzione configurando le proprietà del processo, come descritto in Modificare le proprietà del processo.
9. Esegui il processo.

Aggiunta di connettori a AWS Glue Studio

Un connettore è una parte di codice che facilita la comunicazione tra il datastore e AWS Glue. Puoi effettuare la sottoscrizione a un connettore offerto in Marketplace AWS oppure puoi creare un connettore personalizzato.

Argomenti

- Sottoscrizione ai connettori Marketplace AWS
- Creazione di connettori personalizzati

Sottoscrizione ai connettori Marketplace AWS

AWS Glue Studio semplifica l'aggiunta di connettori da Marketplace AWS.

Come aggiungere un connettore da Marketplace AWS ad AWS Glue Studio

1. Nella console AWS Glue Studio, scegli Connectors (Connettori) nel pannello di navigazione della console.
2. Nella pagina Connectors (Connectors), scegli Go to Marketplace AWS (Vai su Marketplace AWS).
3. In Marketplace AWS, in Featured products (Prodotti in evidenza), scegli il connettore che vuoi utilizzare. Puoi scegliere uno dei connettori in evidenza o utilizzare la ricerca. Puoi eseguire la ricerca in base al nome o al tipo di connettore e utilizzare le opzioni per perfezionare i risultati della ricerca.

   Se desideri utilizzare uno dei connettori in evidenza, scegli View product (Visualizza prodotto). Se hai usato la ricerca per trovare un connettore, scegli il nome del connettore.
4. Nella pagina prodotto del connettore, utilizza le schede per visualizzare le informazioni sul connettore. Se decidi di acquistare il connettore, scegli Continue to Subscribe (Continua con la sottoscrizione).
5. Inserisci le informazioni di pagamento, quindi scegli Continue to Configure (Continua con la configurazione).

6. Nella pagina Configure this software (Configurazione software), scegli il metodo di implementazione e la versione del connettore da utilizzare. Quindi, scegli Continue to Launch (Continua con l'avvio).
7. Nella pagina Launch this software (Avvia software), puoi rivedere le istruzioni di utilizzo fornite dal provider del connettore. Quando vuoi continuare, scegli Attiva connessione in AWS Glue Studio.

   Dopo un breve periodo di tempo, la console mostra la pagina Create marketplace connection (Crea connessione al marketplace) in AWS Glue Studio.
8. Crea una connessione che utilizza questo connettore, come descritto in Creazione di connessioni per i connettori.

   In alternativa, ora puoi scegliere Activate connector only (Attiva solo connettore) per ignorare la creazione di una connessione. Devi creare una connessione in un secondo momento prima di poter utilizzare il connettore.

Creazione di connettori personalizzati

Puoi anche costruire un connettore personalizzato e caricare il codice del connettore su AWS Glue Studio.

I connettori personalizzati sono integrati in AWS Glue Studio attraverso l'API del runtime di Spark AWS Glue. l'API del runtime di Spark AWS Glue consente di collegare qualsiasi connettore compatibile con l'interfaccia Spark, Athena o JDBC. Consente di aggiungere qualsiasi opzione di connessione disponibile per il connettore personalizzato.

È possibile incapsulare tutte le proprietà di connessione con le connessioni AWS Glue e fornire il nome della connessione al processo ETL. L'integrazione con le connessioni del catalogo dati consente di utilizzare le stesse proprietà di connessione per più chiamate in una singola applicazione Spark o in applicazioni diverse.

Puoi specificare ulteriori opzioni per la connessione. Lo script di processo che AWS Glue Studio genera contiene una voce `Datasource` che utilizza la connessione per collegare il connettore con le opzioni di connessione specificate. Ad esempio:

```
Datasource = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"dbTable":"Account","connectionName":"my-custom-
jdbc-
connection"}, transformation_ctx = "DataSource0")
```

Per aggiungere un connettore ad AWS Glue Studio

1. Crea il codice per il connettore personalizzato. Per ulteriori informazioni, consulta Sviluppo di connettori personalizzati.
2. Aggiungi il supporto per le caratteristiche AWS Glue al connettore. Di seguito sono riportati alcuni esempi di queste funzionalità e di come vengono utilizzate all'interno dello script di processo generato da AWS Glue Studio:

   - Mappatura dei dati: il connettore può eseguire il typecast le colonne durante la lettura dal datastore sottostante. Ad esempio, una `dataTypeMapping` di `{"INTEGER":"STRING"}` converte tutte le colonne di tipo `Integer` in colonne di tipo `String` durante l'analisi dei record e la costruzione del `DynamicFrame`. In questo modo gli utenti possono eseguire il cast delle colonne nel tipo desiderato.

     ```
     DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type
     = "custom.jdbc", connection_options = {"dataTypeMapping":{"INTEGER":"STRING"}",
     connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
     ```

   - Partizionamento per letture parallele: AWS Glue consente la lettura dei dati paralleli dal datastore partizionando i dati su una colonna. È necessario specificare la colonna, il limite inferiore e il limite superiore della partizione e il numero di partizioni. Questa funzione consente di utilizzare il parallelismo dei dati e più executor Spark allocati per l'applicazione Spark.

     ```
     DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type
     = "custom.jdbc", connection_options = {"upperBound":"200","numPartitions":"4",
     "partitionColumn":"id","lowerBound":"0","connectionName":"test-connection-jdbc"},
     transformation_ctx = "DataSource0")
     ```

   - Uso di AWS Secrets Manager per l'archiviazione delle credenziali: la connessione al catalogo dati può contenere anche un `secretId` per un segreto archiviato in AWS Secrets Manager. Il segreto AWS può archiviare in modo sicuro le informazioni di autenticazione e le credenziali e fornirle a AWS Glue in fase di esecuzione. In alternativa, è possibile specificare l'`secretId` dallo script Spark come segue:

     ```
     DataSource = glueContext.create_dynamic_frame.from_options(connection_type
     = "custom.jdbc", connection_options = {"connectionName":"test-connection-jdbc",
      "secretId"-> "my-secret-id"}, transformation_ctx = "DataSource0")
     ```

   - Filtraggio dei dati di origine con predicati di riga e proiezioni di colonna: il runtime di Spark AWS Glue consente inoltre agli utenti di trasferire le query SQL per filtrare i dati all'origine

con predicati di riga e proiezioni di colonna. Ciò permette al processo ETL di caricare i dati filtrati più velocemente dagli archivi dati che supportano push-down. Un esempio di query SQL trasferita in un'origine dati JDBC è: `SELECT id, name, department FROM department WHERE id < 200.`

```
DataSource = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"query":"SELECT id, name, department FROM
 department
WHERE id < 200","connectionName":"test-connection-jdbc"}, transformation_ctx =
"DataSource0")
```

- Segnalibri di processo: AWS Glue supporta il caricamento incrementale dei dati da origini JDBC. AWS Glue tiene traccia dell'ultimo record elaborato dal datastore ed elabora nuovi record di dati nelle successive esecuzioni del processo ETL. I segnalibri di processo utilizzano la chiave primaria come colonna predefinita per il tasto segnalibro, a condizione che questa colonna aumenti o diminuisca in sequenza. Per ulteriori informazioni sui segnalibri di processo, consulta Segnalibri di processo nella Guida per sviluppatori di AWS Glue.

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"jobBookmarkKeys":["empno"],
 "jobBookmarkKeysSortOrder"
:"asc", "connectionName":"test-connection-jdbc"}, transformation_ctx =
 "DataSource0")
```

3. Impacchetta il connettore personalizzato come file JAR e carica il file su Amazon S3.
4. Testa il connettore personalizzato. Per ulteriori informazioni, consulta le istruzioni su Glue Custom Connectors: GitHub Local Validation Tests Guide.
5. Nella console AWS Glue Studio, scegli Connectors (Connettori) nel pannello di navigazione della console.
6. Nella pagina Connectors (Connettori), seleziona Create custom connector (Crea connettore personalizzato).
7. Nella pagina Create custom connecto (Crea connettore personalizzato), immetti le seguenti informazioni:

   - Il percorso della posizione del file JAR di codice personalizzato in Amazon S3.
   - Un nome per il connettore che verrà utilizzato da AWS Glue Studio.
   - Il tipo di connettore, che può essere uno tra JDBC, Spark o Athena.

- Il nome del punto di ingresso all'interno del codice personalizzato che AWS Glue Studio chiama per utilizzare il connettore.
  - Per i connettori JDBC, questo campo deve essere il nome della classe del driver JDBC.
  - Per i connettori Spark, questo campo deve essere il nome completo della classe dell'origine dati, o il relativo alias, che si utilizza quando si carica l'origine dati Spark con l'operatore `format`.
- (Solo JDBC) L'URL di base utilizzato dalla connessione JDBC per l'archivio dati.
- (Facoltativo) Una descrizione del connettore personalizzato.

8. Scegli Create connector (Crea connettore).
9. Dalla pagina Connectors (Connettori), crea una connessione che utilizza questo connettore, come descritto in Creazione di connessioni per i connettori.

Connessioni disponibili

Le seguenti connessioni sono disponibili durante la creazione di connessioni per connettori:

- Amazon Aurora— un motore di database relazionale scalabile e ad alte prestazioni con sicurezza, backup e ripristino integrati e accelerazione in memoria.
- Amazon DocumentDB: un servizio di database di documenti scalabile, altamente disponibile e completamente gestito, che supporta le API MongoDB e SQL.
- Amazon Redshift: un servizio di database di documenti scalabile, altamente disponibile e completamente gestito, che supporta le API MongoDB e SQL.
- Google BigQuery: un data warehouse cloud serverless per l'esecuzione di query SQL veloci su set di dati di grandi dimensioni.
- JDBC: un sistema di gestione di database relazionale (RDBMS) che utilizza un'API Java per connettersi e interagire con le connessioni dati.
- Kafka: una piattaforma di elaborazione di flussi open-source, utilizzata per lo streaming e la messaggistica di dati in tempo reale.
- MariaDB: un fork di MySQL sviluppato dalla comunità che offre prestazioni, scalabilità e funzionalità migliorate.
- MongoDB: un database orientato ai documenti multipiattaforma che offre scalabilità, flessibilità e prestazioni elevate.
- MongoDB Atlas: un'offerta di database as a service (DBaaS) basata su cloud di MongoDB che semplifica la gestione e la scalabilità delle implementazioni di MongoDB.

- Microsoft SQL Server: un sistema di gestione di database relazionale (RDBMS) di Microsoft che offre solide funzionalità di archiviazione, analisi e reporting dei dati.
- MySQL: un sistema di gestione di database relazionale (RDBMS) open-source ampiamente utilizzato nelle applicazioni Web e noto per la sua affidabilità e scalabilità.
- Rete: un'origine dati di rete rappresenta una risorsa o un servizio accessibile in rete a cui è possibile accedere tramite una piattaforma di integrazione dei dati.
- OpenSearch: un'origine dati OpenSearch è un'applicazione a cui OpenSearch può connettersi e da cui può importare dati.
- Oracle: un sistema di gestione di database relazionale (RDBMS) di Oracle Corporation che offre solide funzionalità di archiviazione, analisi e reporting dei dati.
- PostgreSQL: un sistema di gestione di database relazionale (RDBMS) open-source che offre solide funzionalità di archiviazione, analisi e reporting dei dati.
- Snowflake: un data warehouse basato su cloud che fornisce servizi di archiviazione e analisi dei dati scalabili e ad alte prestazioni.
- Teradata: un sistema di gestione di database relazionale (RDBMS) che offre funzionalità di archiviazione, analisi e reporting dei dati ad alte prestazioni.
- Vertica: un data warehouse analitico orientato alle colonne progettato per l'analisi di big data che offre prestazioni di query rapide, analisi avanzate e scalabilità.
- SAP HANA: un database in memoria e una piattaforma di analisi che fornisce elaborazione rapida dei dati, analisi avanzate e integrazione dei dati in tempo reale.
- Azure SQL: un servizio di database relazionale basato su cloud di Microsoft Azure che offre funzionalità di archiviazione e gestione dei dati scalabili, affidabili e sicure.
- Cosmos DB: un servizio di database cloud distribuito a livello globale di Microsoft Azure che offre funzionalità di archiviazione e interrogazione di dati scalabili e ad alte prestazioni.

Creazione di connessioni per i connettori

Una connessione AWS Glue è un oggetto Data Catalog che memorizza le informazioni di connessione per un particolare datastore. Le connessioni archiviano le credenziali di accesso, le stringhe URI, le informazioni sul cloud privato virtuale (VPC) e altro ancora. La creazione di connessioni in Data Catalog consente di evitare di dover specificare tutti i dettagli della connessione ogni volta che si crea un processo.

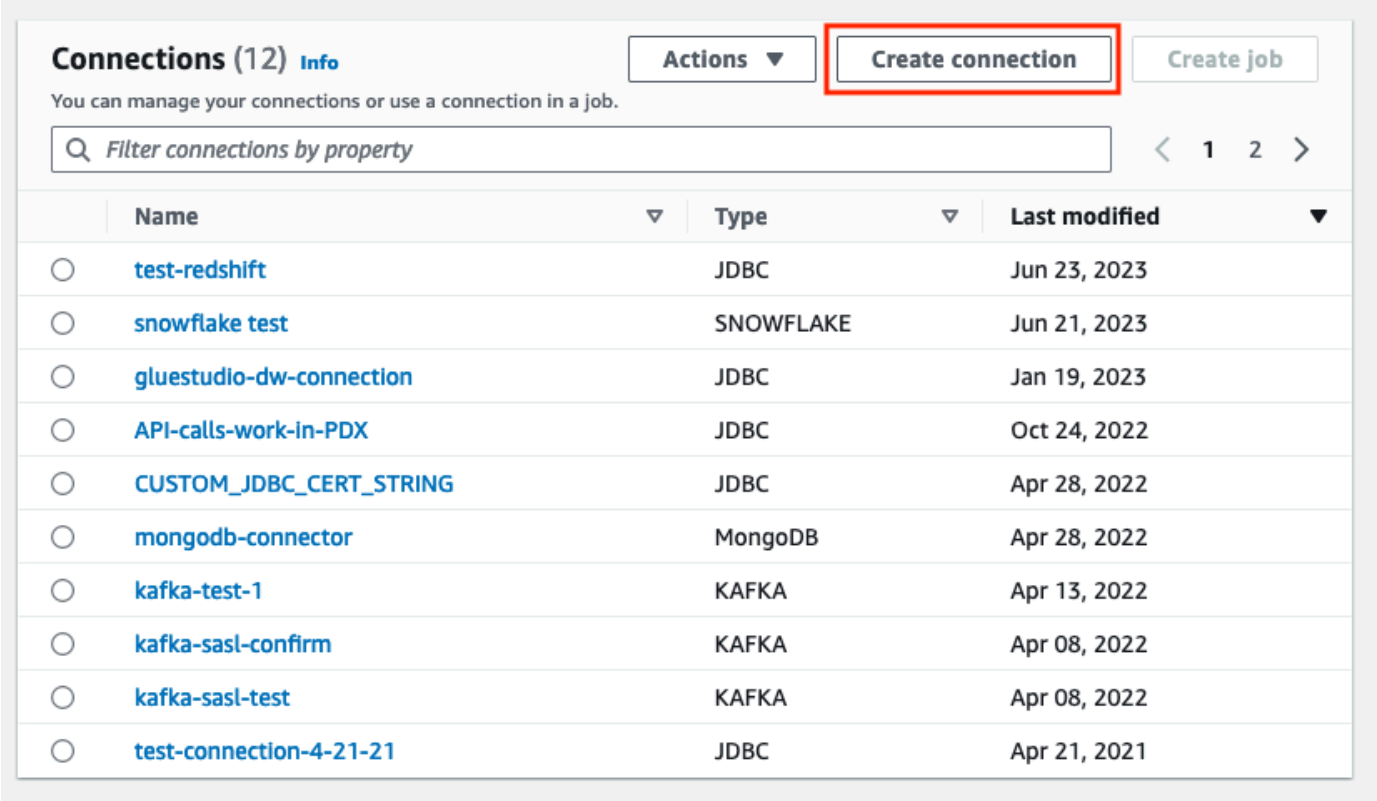
Per creare una connessione per un connettore

1. Nella console AWS Glue Studio, scegli Connectors (Connettori) nel pannello di navigazione della console. Nella sezione Connessioni, scegli Crea connessione.
2. Scegli l'origine dati per la quale desideri creare una connessione nel passaggio 1 della procedura guidata Crea connessione dati. Sono disponibili diversi modi per visualizzare le origini dati disponibili, inclusi i seguenti:

   - Filtra le origini dati disponibili scegliendo una scheda. Per impostazione predefinita, è selezionata l'opzione Tutti i connettori.
   - Attiva Elenco per visualizzare le origini dati sotto forma di elenco o torna a Griglia per visualizzare i connettori disponibili nel layout a griglia.
   - Utilizza la barra di ricerca per restringere l'elenco delle origini dati. Durante la digitazione, i risultati di ricerca vengono visualizzati e le fonti non corrispondenti vengono rimosse dalla visualizzazione.

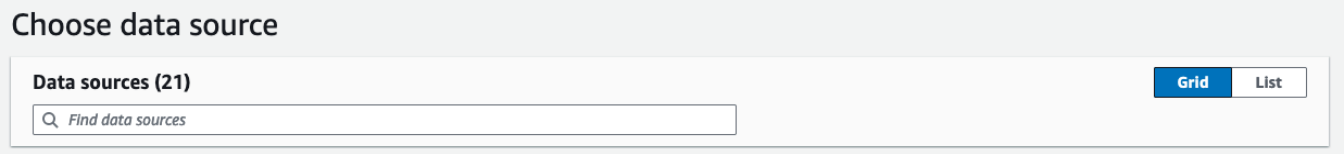
   Dopo aver scelto l'origine dati, scegli Avanti.
3. Configura la connessione nel passaggio 2 della procedura guidata.

   Inserisci i dettagli della connessione. A seconda del tipo di connettore selezionato, viene richiesto di inserire ulteriori informazioni:



4. Scegli l'origine dati per la quale desideri creare una connessione nel passaggio 1 della procedura guidata Crea connessione dati. Sono disponibili diversi modi per visualizzare le origini dati disponibili. Per impostazione predefinita, tutte le origini dati disponibili vengono visualizzate in un layout a griglia. Puoi anche:

   - Attiva Elenco per visualizzare le origini dati sotto forma di elenco o torna a Griglia per visualizzare i connettori disponibili nel layout a griglia.
   - Utilizza la barra di ricerca per restringere l'elenco delle origini dati. Durante la digitazione, i risultati di ricerca vengono visualizzati e le fonti non corrispondenti vengono rimosse dalla visualizzazione.

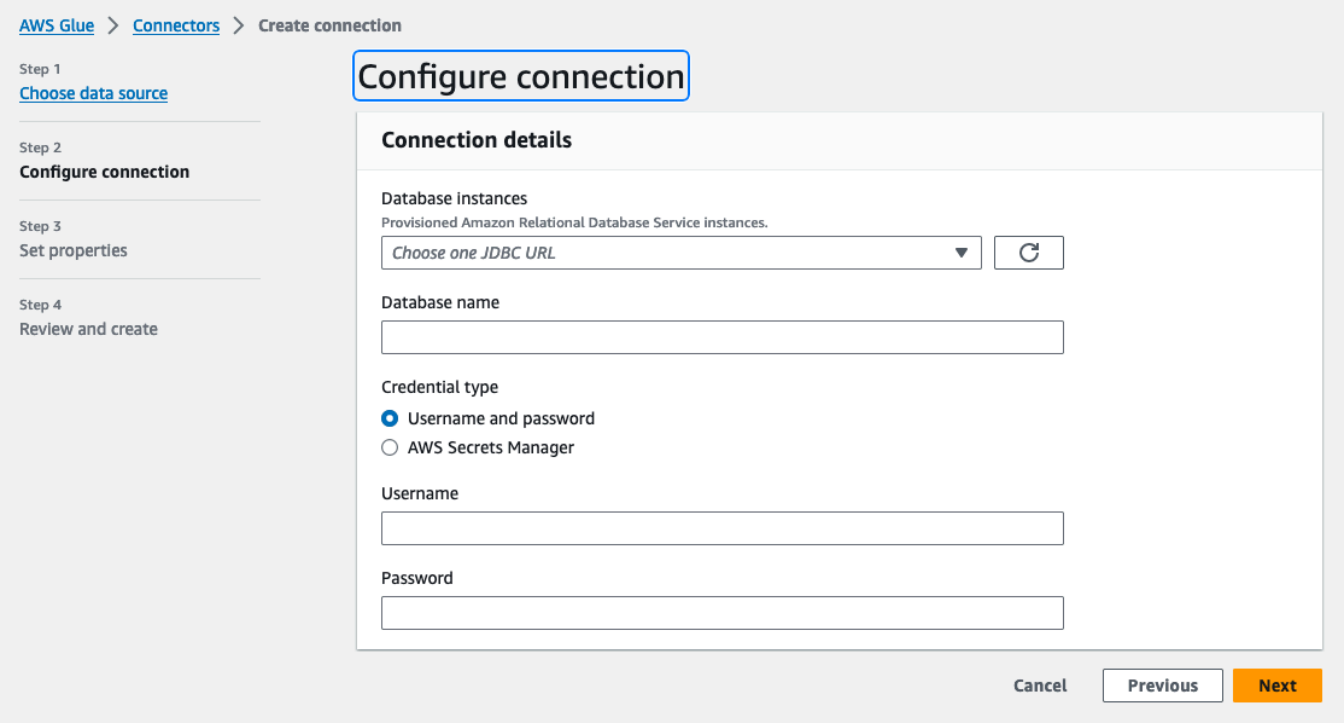
   

   Dopo aver scelto l'origine dati, scegli Avanti.

5. Configura la connessione nel passaggio 2 della procedura guidata.

Inserisci i dettagli della connessione. A seconda del tipo di connettore selezionato, potrebbe essere richiesto di inserire ulteriori informazioni di connessione: Sono inclusi:

- Dettagli di connessione: questi campi cambieranno a seconda dell'origine dati a cui ti stai connettendo. Ad esempio, se ti connetti a database Amazon DocumentDB, inserirai l'URL di Amazon DocumentDB. Se ti connetti a Amazon Aurora, sceglierai l'istanza del database e inserirai il nome del database. Di seguito, sono riportati i dettagli di connessione necessari per Amazon Aurora:



- Tipo di credenziale: scegli tra nome utente e password o AWS Secrets Manager. Inserisci le informazioni di autenticazione richieste.
- Per i connettori che utilizzano JDBC, inserisci le informazioni necessarie per creare l'URL JDBC per il datastore.
- Se usi un cloud privato virtuale (VPC), inserisci le relative informazioni di rete.

6. Impostare le proprietà di connessione nel passaggio 3 della procedura guidata. È possibile aggiungere descrizione e tag come parte opzionale di questo passaggio. Il nome è obbligatorio ed è precompilato con un valore predefinito. Seleziona Successivo.
7. Controlla l'origine, i dettagli e le proprietà della connessione. Per apportare modifiche, scegli Modifica per il passaggio della procedura guidata. Quando è tutto pronto, scegli Crea connessione.

Scegli Create connection (Crea connessione).

Vieni reindirizzato alla pagina Connectors (Connettori) e il banner informativo indica la connessione creata. Ora puoi utilizzare la connessione nei tuoi processi AWS Glue Studio.

Creazione di una connessione Kafka

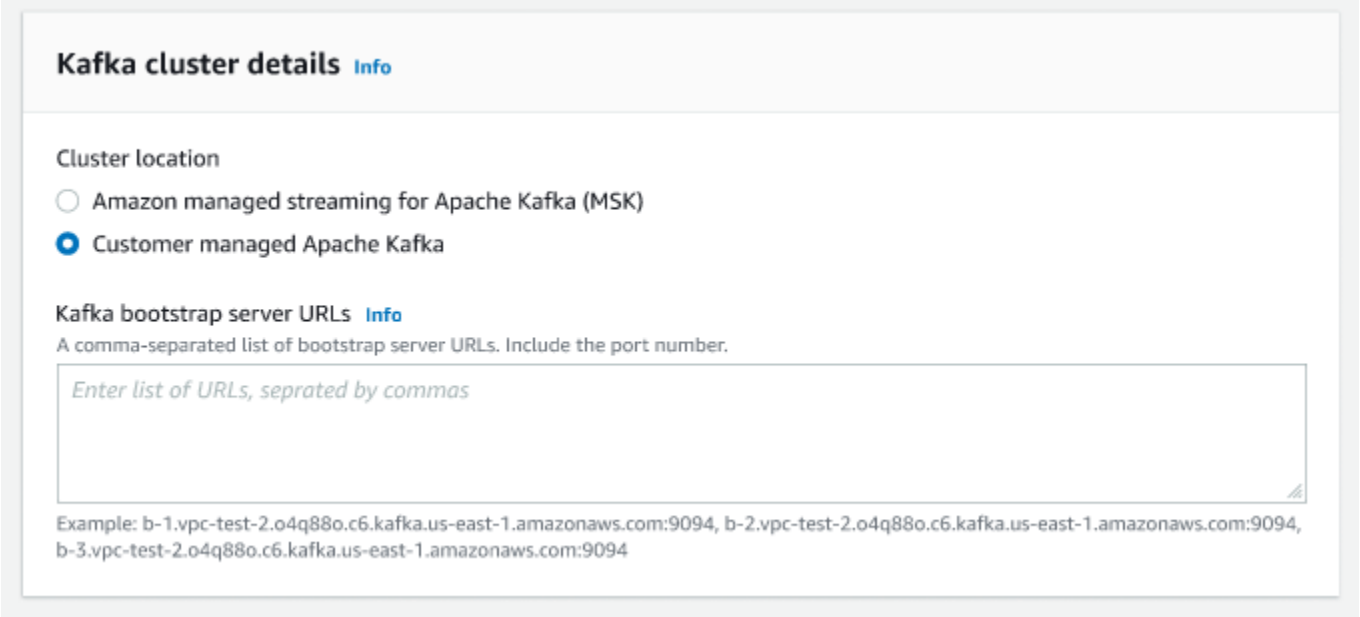
Quando crei una connessione Kafka, selezionando Kafka dal menu a discesa visualizzerai impostazioni aggiuntive da configurare:

- Dettagli del cluster Kafka
- Autenticazione
- Crittografia
- Opzioni di rete

Configurazione dei dettagli del cluster Kafka

1. Scegli la posizione del cluster. Puoi scegliere tra un cluster Amazon Managed Streaming for Apache Kafka (MSK) o un cluster Apache Kafka gestito dal cliente. Per ulteriori informazioni sullo streaming Amazon Managed for Apache Kafka, consulta Amazon Managed Streaming for Apache Kafka (MSK).

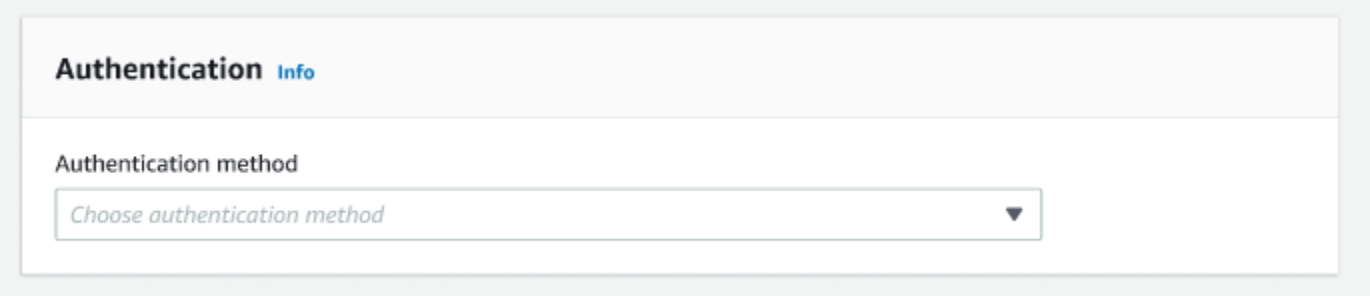
   > **Note**
   >
   > Amazon Managed Streaming for Apache Kafka supporta solo i metodi di autenticazione TLS e SASL/SCRAM-SHA-512.



2. Inserisci gli URL per i server bootstrap Kafka. È possibile inserirne più di uno separando ciascun server con una virgola. Includi il numero della porta alla fine dell'URL aggiungendo `:<port number>`.

   Ad esempio: `b-1.vpc-test-2.034a88o.kafka-us-east-1.amazonaws.com:9094`

Selezionare i metodi di autenticazione

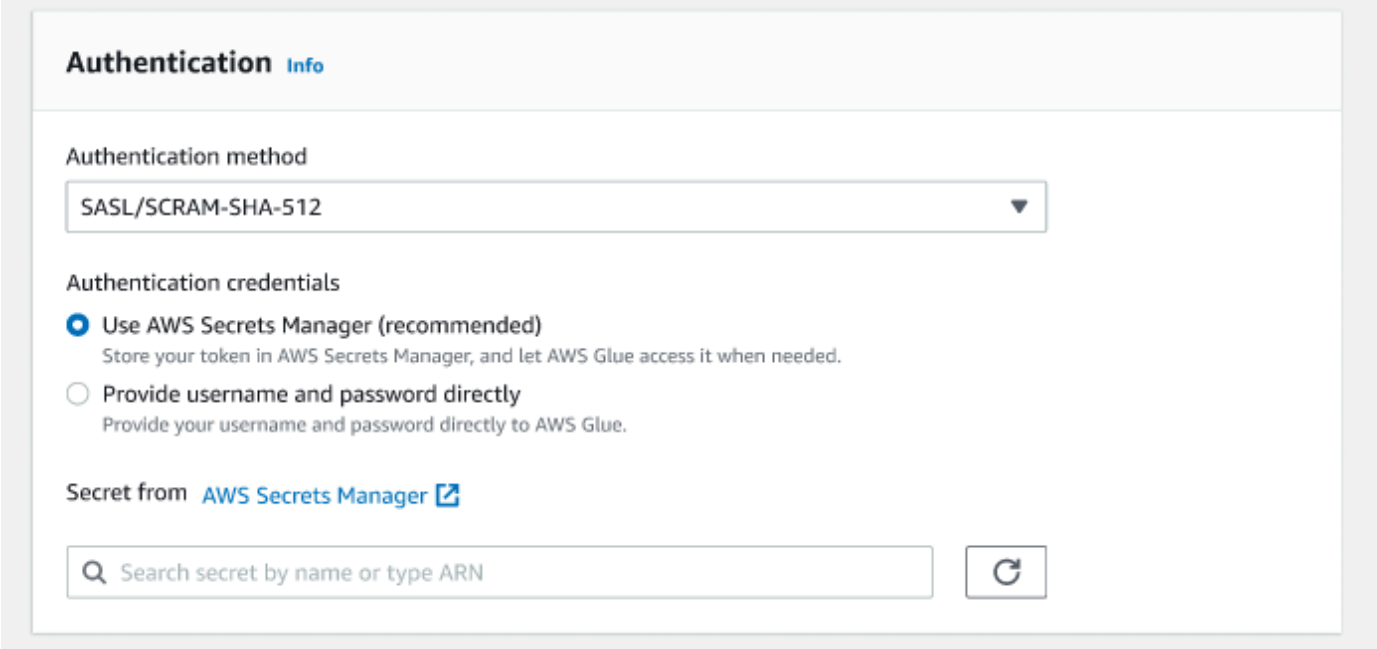


AWS Glue supporta il framework Simple Authentication Security Layer (SASL) per l'autenticazione. Il framework SASL supporta vari meccanismi di autenticazione e AWS Glue offre i protocolli SCRAM (nome utente e password), GSSAPI (protocollo Kerberos) e PLAIN (nome utente e password).

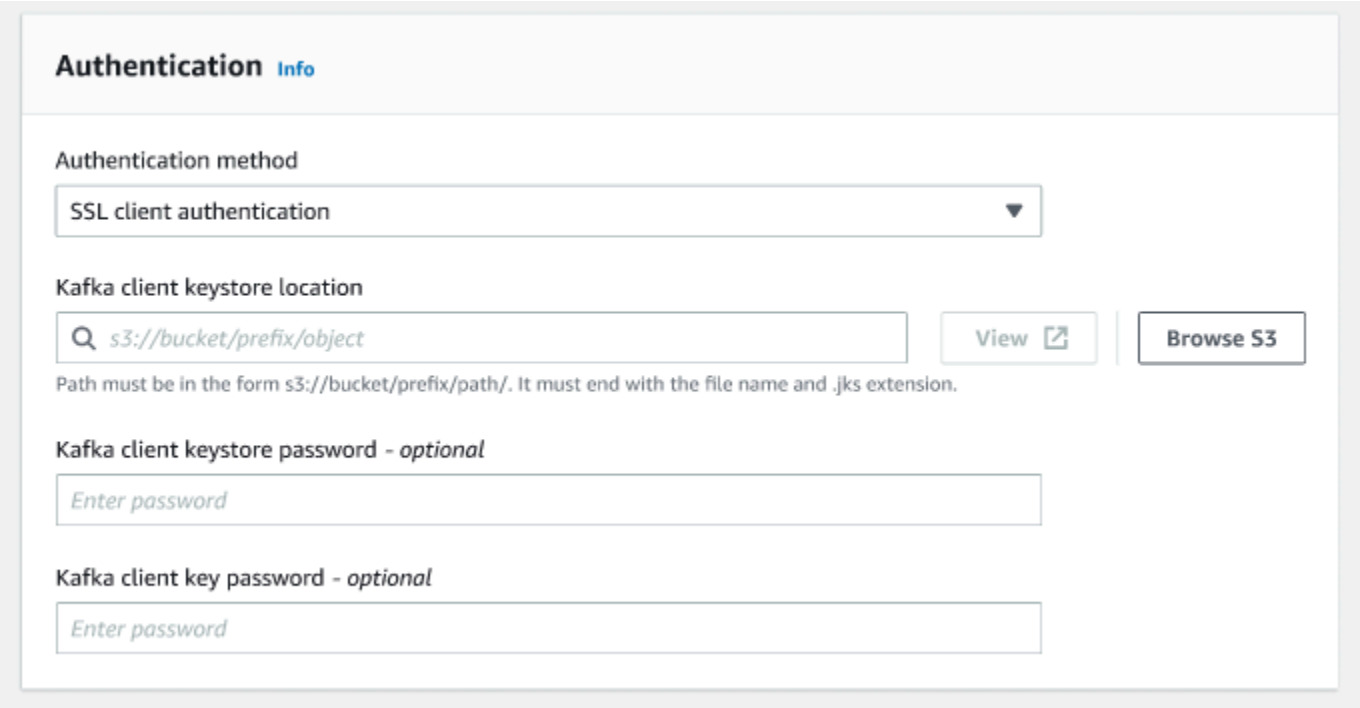
Quando si sceglie un metodo di autenticazione dal menu a discesa, è possibile selezionare i seguenti metodi di autenticazione client:

- Nessuno: nessuna autenticazione. Questo è utile se si crea una connessione a scopo di test.

- SASL/SCRAM-SHA-512: scegli questo metodo di autenticazione per specificare le credenziali di autenticazione. Sono disponibili due opzioni:
  - Usa AWS Secrets Manager (consigliato): se selezioni questa opzione, puoi memorizzare le tue credenziali in AWS Secrets Manager e consentire AWS Glue l'accesso alle informazioni quando necessario. Specifica il segreto che memorizza le credenziali di autenticazione SSL o SASL.



  - Fornisci direttamente nome utente e password.
- SASL/GSSAPI (Kerberos): selezionando questa opzione, è possibile selezionare la posizione del file keytab, il file krb5.conf e inserire il nome principale Kerberos e il nome del servizio Kerberos. Le posizioni per il file keytab e il file krb5.conf devono trovarsi in una posizione Amazon S3. Poiché MSK non supporta ancora SASL/GSSAPI, questa opzione è disponibile solo per i cluster Apache Kafka gestiti dal cliente. Per ulteriori informazioni, consulta la Documentazione di MIT Kerberos: keytab.
- SASL/PLAIN: scegli questo metodo di autenticazione per specificare le credenziali di autenticazione. Sono disponibili due opzioni:
  - Usa AWS Secrets Manager (consigliato): se selezioni questa opzione, puoi memorizzare le tue credenziali in AWS Secrets Manager e consentire AWS Glue l'accesso alle informazioni quando necessario. Specifica il segreto che memorizza le credenziali di autenticazione SSL o SASL.
  - Fornisci direttamente nome utente e password.
- Autenticazione client SSL: selezionando questa opzione, è possibile selezionare la posizione del keystore client Kafka navigando su Amazon S3. Facoltativamente, è possibile inserire la password del keystore del client Kafka e la password della chiave del client Kafka.



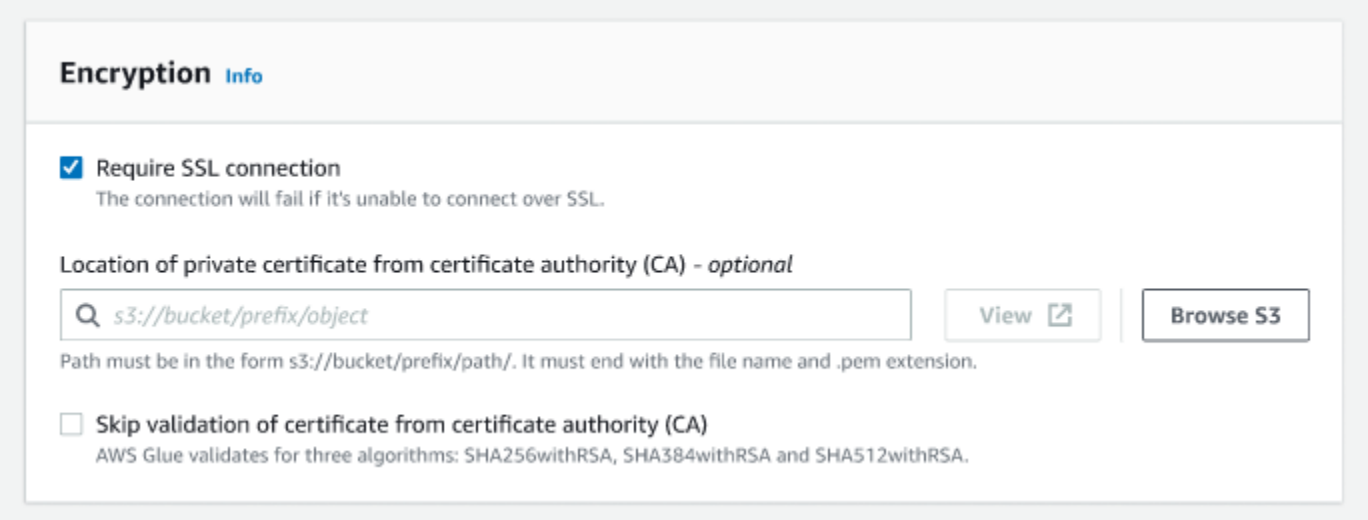
Configurazione delle impostazioni di crittografia

1. Se la connessione Kafka richiede una connessione SSL, seleziona la casella di controllo per Require SSL connection (Connessione SSL necessaria). Tieni presente che la connessione non riesce se non può connettersi tramite SSL. SSL per la crittografia può essere utilizzato con qualsiasi metodo di autenticazione (SASL/SCRAM-SHA-512, SASL/GSSAPI, SASL/PLAIN o SSL Client Authentication) ed è facoltativo.

   Se il metodo di autenticazione è impostato su Autenticazione client SSL, questa opzione verrà selezionata automaticamente e verrà disattivata per evitare eventuali modifiche.
2. (Facoltativo). Scegli la posizione del certificato privato dell'autorità di certificazione (CA). Tieni presente che la posizione della certificazione deve trovarsi in una sede S3. Scegli Browse (Sfoglia) per scegliere il file da un bucket S3 collegato. Il percorso deve essere nel formato `s3://bucket/prefix/filename.pem`. Deve terminare con il nome del file e l'estensione .pem.
3. È possibile scegliere di saltare la convalida del certificato da un'autorità di certificazione (CA). Scegli la casella di controlloSkip validation of certificate from certificate authority (CA) (Salta la convalida del certificato da un'autorità di certificazione [CA]). Se questa casella non è selezionata, AWS Glue convalida i certificati per tre algoritmi:

   - SHA256withRSA

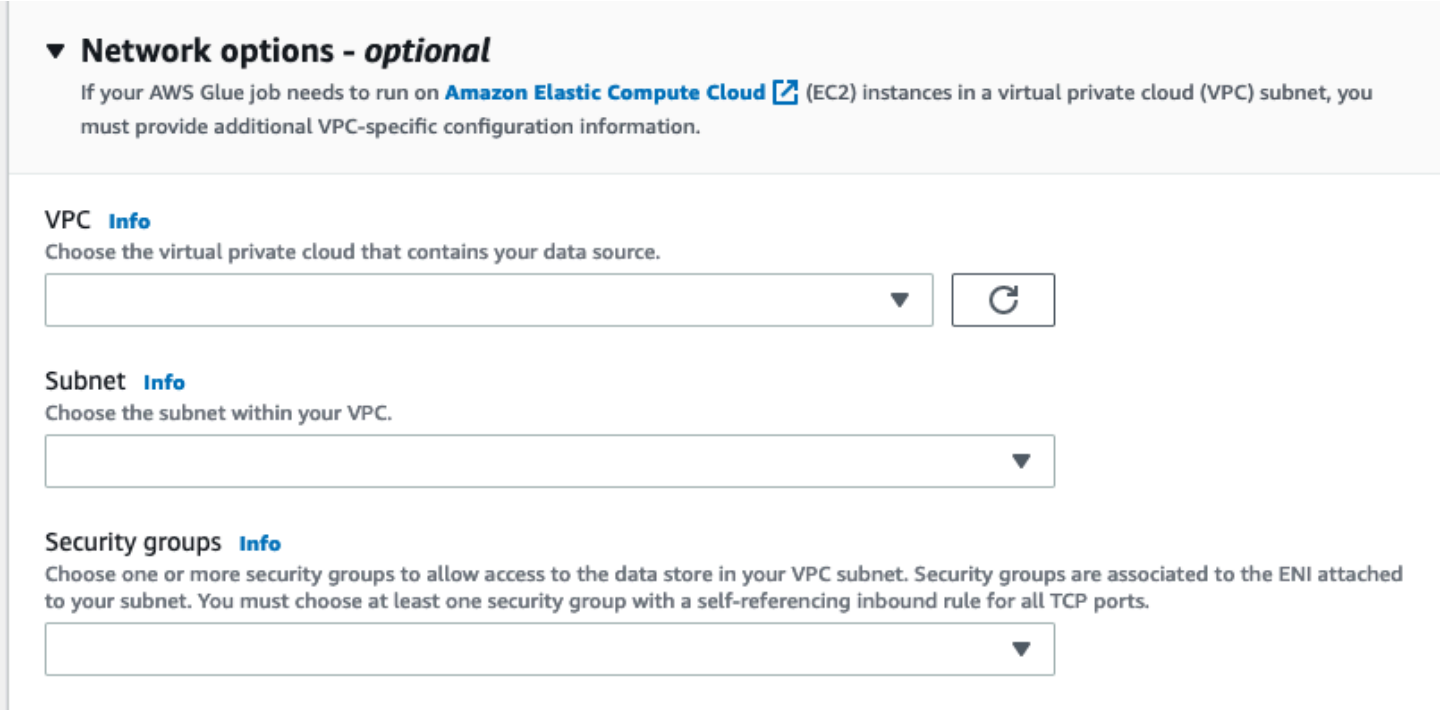
- SHA384 con RSA
- SHA512withRSA



(Facoltativo) Opzioni di rete

Di seguito sono riportate i passaggi opzionali per configurare VPC, sottorete e gruppi di sicurezza. Se il processo AWS Glue deve essere eseguito su istanze Amazon EC2 in una sottorete di cloud privato virtuale (VPC), è necessario fornire ulteriori informazioni di configurazione specifiche per il VPC.

1. Scegli il VPC (cloud privato virtuale) che contiene l'origine dati.
2. Scegli la sottorete nel VPC.
3. Scegli uno o più gruppi di sicurezza per consentire l'accesso all'archivio dati nella sottorete VPC. I gruppi di sicurezza sono associati all'ENI collegata alla sottorete. È necessario scegliere almeno un gruppo di sicurezza con una regola in entrata autoreferenziale per tutte le porte TCP.



Creazione di processi con connettori personalizzati

Puoi utilizzare connettori e connessioni sia per i nodi di origine dati che per i nodi di destinazione dati in AWS Glue Studio.

Argomenti

- Creare processi che utilizzano un connettore per l'origine dati
- Configurare le proprietà di origine per i nodi che utilizzano connettori
- Configurare le proprietà di destinazione per i nodi che utilizzano connettori

Creare processi che utilizzano un connettore per l'origine dati

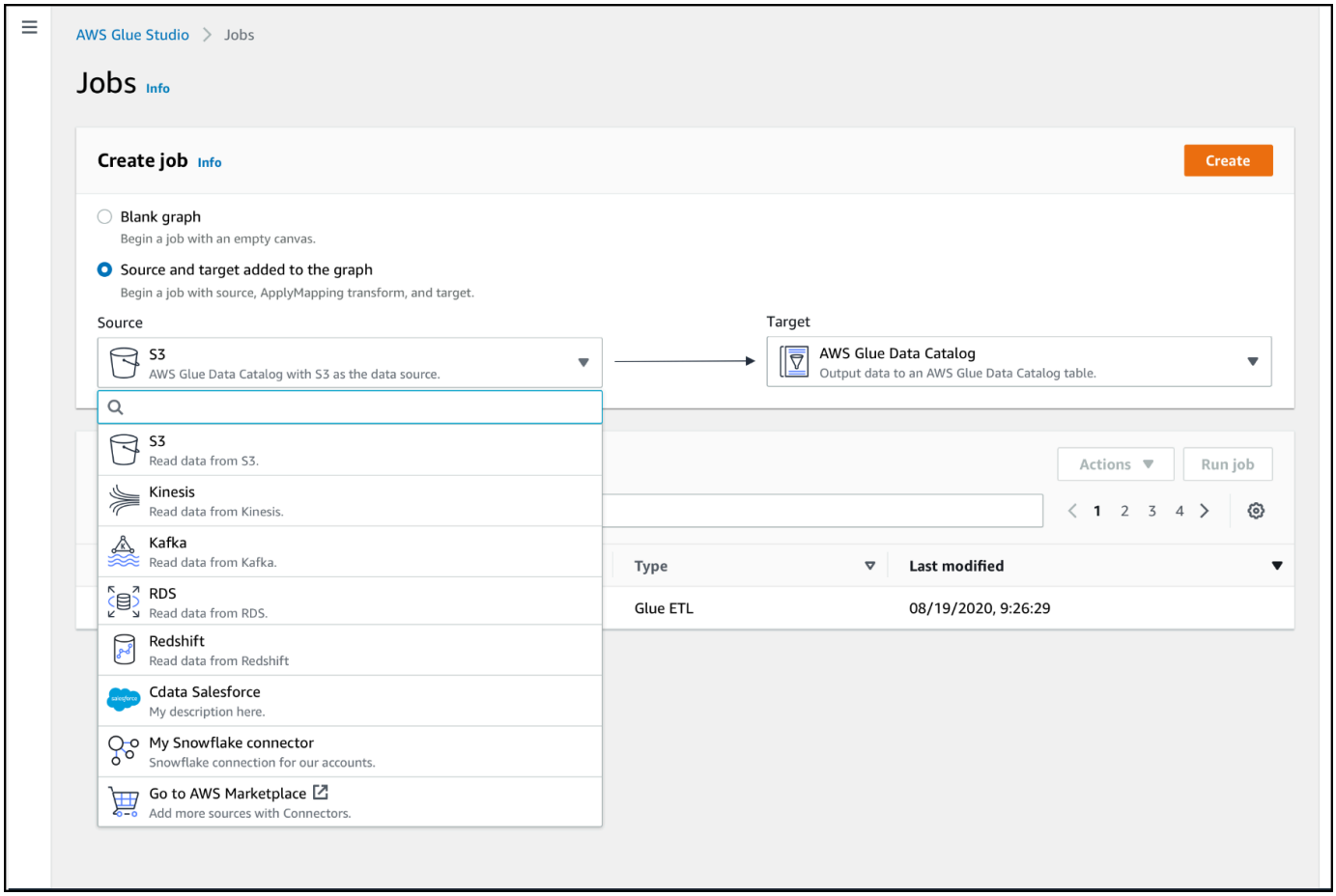
Quando si crea un nuovo processo, puoi scegliere un connettore per l'origine dati e le destinazioni dati.

Per creare un processo che utilizza connettori per l'origine dati o la destinazione dati

1. Accedi alla AWS Management Console e apri la console AWS Glue Studio all'indirizzo https://console.aws.amazon.com/gluestudio/.

2. Nella pagina Connectors (Connettori), nell'elenco di risorse Your connections (Le tue connessioni), scegli la connessione da utilizzare nel processo, quindi scegli Create job (crea processo).

   In alternativa, nella pagina Jobs (Processi) di AWS Glue Studio, sotto Create job (Crea processo), scegli Source and target added to the graph (Origine e destinazione aggiunti al grafico). Nell'elenco a discesa Source (Origine), scegli il connettore personalizzato che desideri utilizzare nel processo. Puoi anche scegliere un connettore per Target (Destinazione).



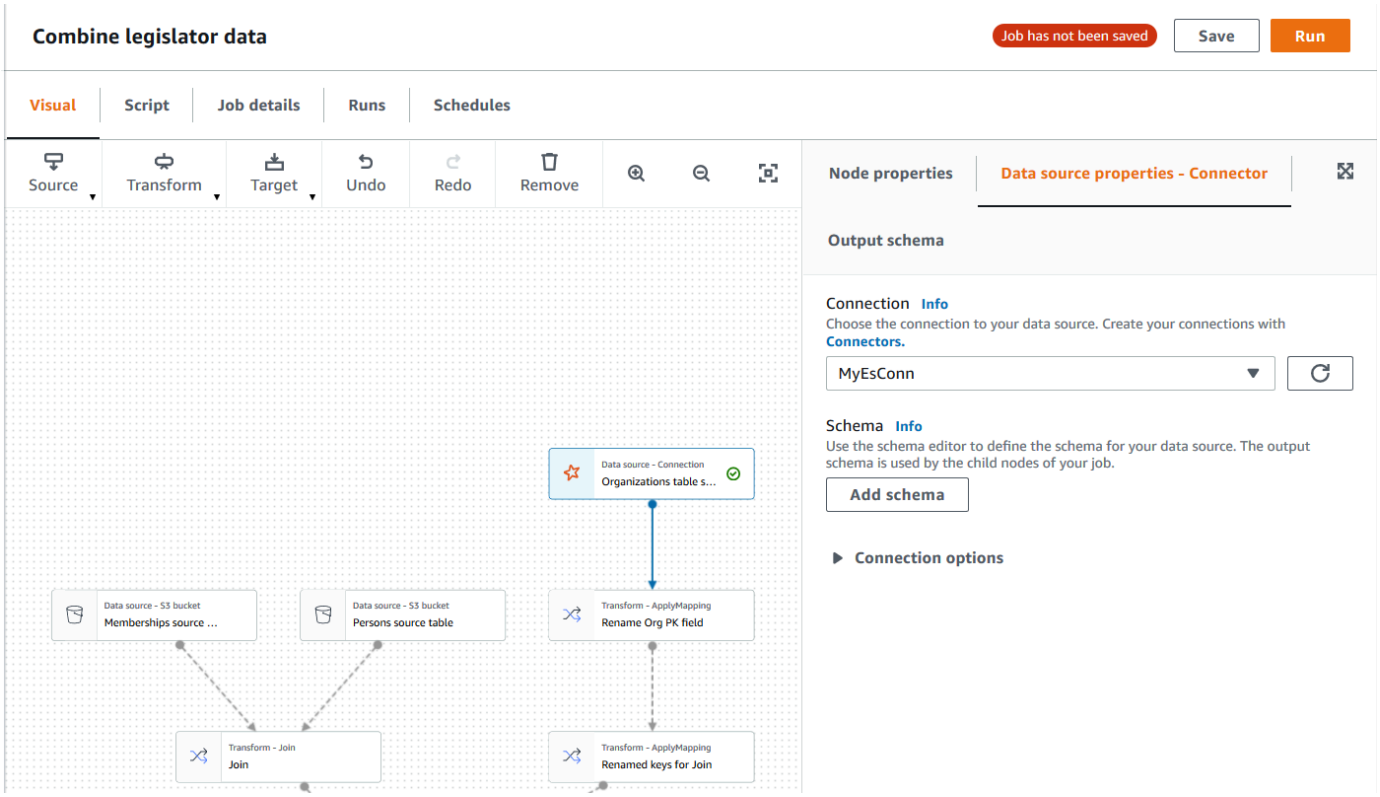
3. Scegli quindi Create (Crea) per aprire l'editor visivo dei processi.
4. Configura il nodo di origine dati, come descritto in Configurare le proprietà di origine per i nodi che utilizzano connettori.
5. Continua a creare il processo ETL aggiungendo trasformazioni, datastore aggiuntivi e destinazioni dati, come descritto in ETL visivo con AWS Glue Studio.
6. Personalizza l'ambiente di esecuzione configurando le proprietà del processo, come descritto in Modificare le proprietà del processo.
7. Salva ed esegui il processo.

Configurare le proprietà di origine per i nodi che utilizzano connettori

Dopo aver creato un processo che utilizza un connettore per l'origine dati, l'editor visivo dei processi mostra un grafico del processo con un nodo di origine dati configurato per il connettore. Devi configurare le proprietà dell'origine dati per tale nodo.

Per configurare le proprietà di un nodo di origine dati che utilizza un connettore

1. Scegli il nodo dell'origine dati del connettore nel grafico del processo oppure aggiungi un nuovo nodo e scegli il connettore per Node type (Tipo di nodo). Quindi, sulla destra, nel pannello dei dettagli dei nodi, scegli la scheda Data source properties (Proprietà dell'origine dati) se non è già selezionata.



2. Nella scheda Data source properties (Proprietà dell'origine dati), scegli la connessione da utilizzare per questo processo.

   Inserisci le informazioni aggiuntive necessarie per ciascun tipo di connessione:

JDBC

- Data source input type (Tipo di input dell'origine dati): scegli di specificare un nome di tabella o una query SQL come origine dati. A seconda del tipo, devi fornire le seguenti informazioni aggiuntive:
  - Table name (Nome tabella): il nome della tabella nell'origine dati. Se l'origine dati non utilizza il termine table (tabella), fornisci il nome di una struttura dati appropriata, come indicato dalle informazioni sull'utilizzo del connettore personalizzato (disponibili in Marketplace AWS).
  - Filter predicate (Predicato di filtro): una clausola di condizione da utilizzare durante la lettura dell'origine dati, simile a una clausola `WHERE`, che viene utilizzata per recuperare un sottoinsieme dei dati.
  - Query code (Codice query): inserisci una query SQL da utilizzare per recuperare un set di dati specifico dall'origine dati. Un esempio di query SQL di base è:

    ```
    SELECT column_list FROM
                            table_name WHERE where_clause
    ```

- Schema: poiché AWS Glue Studio utilizza le informazioni archiviate nella connessione per accedere all'origine dati anziché recuperare le informazioni sui metadati da una tabella del Data Catalog, devi fornire i metadati dello schema per l'origine dati. Scegli Add schema (Aggiungi schema) per aprire l'editor dello schema.

  Per istruzioni su come utilizzare l'editor dello schema, consulta Modifica dello schema in un nodo di trasformazione personalizzato.

- Partition column (Colonna di partizione): (facoltativo) puoi scegliere di partizionare le letture dei dati fornendo valori per Partition column (Colonna di partizione), Lower bound (Limite inferiore), Upper bound (Limite superiore) e Number of partitions (Numero di partizioni).

  I valori `lowerBound` e `upperBound` vengono utilizzati per decidere lo stride della partizione, non per filtrare le righe nella tabella. Tutte le righe della tabella vengono partizionate e restituite.

> **Note**
>
> Il partizionamento delle colonne aggiunge una condizione di partizionamento aggiuntiva alla query utilizzata per leggere i dati. Quando si utilizza una query anziché un nome di tabella, è necessario verificare che la query funzioni con la condizione di partizionamento specificata. Ad esempio:
>
> - Se il formato della query è `"SELECT col1 FROM table1"`, testa la query aggiungendo una clausola `WHERE` alla fine della query che utilizza la colonna della partizione.
> - Se il formato della query è `"SELECT col1 FROM table1 WHERE col2=val"`, testa la query estendendo la clausola `WHERE` con `AND` e un'espressione che utilizza la colonna della partizione.

- Data type casting (Casting del tipo di dati): se l'origine dati utilizza tipi di dati non disponibili in JDBC, utilizza questa sezione per specificare come convertire un tipo di dati dell'origine dati in tipi di dati JDBC. Puoi specificare fino a 50 conversioni di tipi di dati diverse. Tutte le colonne dell'origine dati che utilizzano lo stesso tipo di dati vengono convertite nello stesso modo.

  Ad esempio, se nell'origine dati sono presenti tre colonne che utilizzano il tipo di dati `Float` e si indica che il tipo di dati `Float` deve essere convertito nel tipo di dati `String` JDBC, tutte e tre le colonne che utilizzano il tipo di dati `Float` vengono convertite in `String`.

- Job bookmark keys (Chiavi di segnalibro di processo): i segnalibri del processo aiutano AWS Glue a mantenere le informazioni sullo stato e prevenire la rielaborazione dei dati precedenti. Specifica un'altra o più colonne come chiavi di segnalibro. AWS Glue Studio utilizza i tasti di segnalibro per tenere traccia dei dati già elaborati durante una precedente esecuzione del processo ETL. Tutte le colonne utilizzate per le chiavi di segnalibro personalizzati devono aumentare o diminuire in modo rigorosamente monotonico, ma sono ammessi spazi.

  Se inserisci più chiavi di segnalibro, queste vengono combinate per formare una singola chiave composta. Una chiave di segnalibro di processo composta non deve contenere colonne duplicate. Se non si specificano le chiavi di segnalibro, AWS Glue Studio per impostazione predefinita utilizza la chiave primaria come chiave di segnalibro, a condizione che sia in ordine crescente o decrescente sequenzialmente (senza spazi vuoti). Se la tabella non dispone di una chiave primaria, ma la proprietà segnalibro di processo è

abilitata, devi fornire chiavi personalizzate di segnalibro di processo. In caso contrario, la ricerca delle chiavi primarie da utilizzare come impostazione predefinita avrà esito negativo e l'esecuzione del processo avrà non riuscirà.

- Job bookmark keys sorting order (Ordinamento delle chiavi di segnalibro di processo): scegli se i valori chiave vengono aumentati o diminuiti in sequenza.

Spark

- Schema: poiché AWS Glue Studio utilizza le informazioni archiviate nella connessione per accedere all'origine dati anziché recuperare le informazioni sui metadati da una tabella del Data Catalog, devi fornire i metadati dello schema per l'origine dati. Scegli Add schema (Aggiungi schema) per aprire l'editor dello schema.

  Per istruzioni su come utilizzare l'editor dello schema, consulta Modifica dello schema in un nodo di trasformazione personalizzato.
- Connection options (Opzioni di connessione): inserisci ulteriori coppie chiave-valore in base alle esigenze per fornire ulteriori informazioni o opzioni di connessione. Ad esempio, puoi inserire un nome di database, un nome di tabella, un nome utente e una password.

  Ad esempio, per OpenSearch, si inseriscono le seguenti coppie chiave-valore, come descritto in: the section called “ Tutorial: utilizzo del connettore AWS Glue per Elasticsearch ”
  - `es.net.http.auth.user` : *`username`*
  - `es.net.http.auth.pass` : *`password`*
  - `es.nodes` : `https://`*`<Elasticsearch endpoint>`*
  - `es.port` : `443`
  - `path`: *`<Elasticsearch resource>`*
  - `es.nodes.wan.only` : `true`

Per un esempio delle opzioni di connessione minime da usare, vedete lo script di test di esempio MinimalSparkConnectorTest.scala on GitHub, che mostra le opzioni di connessione che normalmente fornireste in una connessione.

Athena

- Table name (Nome tabella): il nome della tabella nell'origine dati. Se utilizzi un connettore per leggere i log di CloudWatch Athena-log, devi inserire il nome della tabella. `all_log_streams`
- Athena schema name (Nome schema Athena): scegli lo schema nell'origine dati Athena corrispondente al database che contiene la tabella. Se si utilizza un connettore per la lettura da CloudWatch Athena-logs, è necessario immettere un nome di schema simile a. `/aws/glue/name`
- Schema: poiché AWS Glue Studio utilizza le informazioni archiviate nella connessione per accedere all'origine dati anziché recuperare le informazioni sui metadati da una tabella del Data Catalog, devi fornire i metadati dello schema per l'origine dati. Scegli Add schema (Aggiungi schema) per aprire l'editor dello schema.

  Per istruzioni su come utilizzare l'editor dello schema, consulta Modifica dello schema in un nodo di trasformazione personalizzato.
- Additional connection options (Opzioni di connessione aggiuntive): inserisci ulteriori coppie chiave-valore in base alle esigenze per fornire ulteriori informazioni o opzioni di connessione.

Per un esempio, consultate il `README.md` file all'indirizzo https://github.com/aws-samples/aws-glue-samples /tree/master/ GlueCustomConnectors /development/ATHENA. Nei passaggi di questo documento, il codice di esempio mostra le opzioni di connessione minime richieste, che sono `tableName`, `schemaName` e `className`. L'esempio di codice specifica queste opzioni come parte della variabile `optionsMap`, ma puoi specificarle per la connessione e quindi utilizzarla.

3. (Facoltativo) Dopo aver configurato le proprietà del nodo e dell'origine dati, puoi visualizzare lo schema dei dati risultante per l'origine dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. Lo schema visualizzato in questa scheda viene utilizzato da tutti i nodi figlio aggiunti al grafico del processo.
4. (Facoltativo) Dopo aver configurato le proprietà del nodo e dell'origine dati, puoi visualizzare il set di dati dall'origine dati scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Configurare le proprietà di destinazione per i nodi che utilizzano connettori

Se usi un connettore per il tipo di destinazione dati, devi configurare le proprietà del nodo di destinazione dati.

Per configurare le proprietà di un nodo di destinazione dati che utilizza un connettore

1. Scegli il nodo di destinazione dati del connettore nel grafico del processo. Quindi, sulla destra, nel pannello dei dettagli dei nodi, scegli la scheda Data target properties (Proprietà della destinazione dati) se non è già selezionata.
2. Nella scheda Data target properties (Proprietà della destinazione dati), scegli la connessione da utilizzare per la scrittura nella destinazione.

   Inserisci le informazioni aggiuntive necessarie per ciascun tipo di connessione:

   JDBC

   - Connection (Connessione): scegli la connessione da utilizzare con il connettore. Per informazioni su come creare una connessione, vedi Creazione di connessioni per i connettori.
   - Table name (Nome tabella): il nome della tabella nella destinazione dati. Se la destinazione dati non utilizza il termine table (tabella), fornisci il nome di una struttura dati appropriata, come indicato dalle informazioni sull'utilizzo del connettore personalizzato (disponibili in Marketplace AWS).
   - Batch size (Dimensione batch): (facoltativo): immetti il numero di righe o record da inserire nella tabella di destinazione in un'unica operazione. Il valore predefinito è 1000 righe.

   Spark

   - Connection (Connessione): scegli la connessione da utilizzare con il connettore. Se non hai creato una connessione in precedenza, scegli Create connection (Crea connessione) per crearne una. Per informazioni su come creare una connessione, vedi Creazione di connessioni per i connettori.
   - Connection options (Opzioni di connessione): inserisci ulteriori coppie chiave-valore in base alle esigenze per fornire ulteriori informazioni o opzioni di connessione. Puoi inserire un nome di database, un nome di tabella, un nome utente e una password.

Ad esempio, for, si inseriscono le seguenti coppie OpenSearch chiave-valore, come descritto in: the section called “ Tutorial: utilizzo del connettore AWS Glue per Elasticsearch ”

- `es.net.http.auth.user` : *username*
- `es.net.http.auth.pass` : *password*
- `es.nodes` : `https://`*<Elasticsearch endpoint>*
- `es.port` : `443`
- `path`: *<Elasticsearch resource>*
- `es.nodes.wan.only` : `true`

Per un esempio delle opzioni di connessione minime da usare, vedete lo script di test di esempio MinimalSparkConnectorTest.scala on GitHub, che mostra le opzioni di connessione che normalmente fornireste in una connessione.

3. Dopo aver configurato le proprietà del nodo e dell'origine dati, puoi visualizzare lo schema dei dati risultante per l'origine dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo.

Gestione di connettori e connessioni

Per gestire i connettori e le connessioni, utilizza la pagina Connessioni di AWS Glue.

Argomenti

- Visualizzazione dei dettagli dei connettori e delle connessioni
- Modifica di connettori e connessioni
- Eliminazione di connettori e connessioni
- Annullare una sottoscrizione per un connettore

Visualizzazione dei dettagli dei connettori e delle connessioni

Puoi visualizzare le informazioni di riepilogo sui connettori e sulle connessioni nelle tabelle delle risorse Your connectors (I tuoi connettori) e Your connections (Le tue connessioni) nella scheda Connectors (Connettori). Per visualizzare le informazioni dettagliate, procedi come segue.

Per visualizzare i dettagli del connettore o della connessione

1. Nella console AWS Glue Studio, scegli Connectors (Connettori) nel pannello di navigazione della console.
2. Scegli il connettore o la connessione di cui visualizzare le informazioni dettagliate.
3. Scegli Actions (Operazioni) e quindi View details (Visualizza i dettagli) per aprire la pagina relativa ai dettagli del connettore o della connessione.
4. Nella pagina prodotto, puoi scegliere di modificare o eliminare il connettore o la connessione.

   - Per i connettori, puoi scegliere Create connection (Crea connessione) per creare una nuova connessione che utilizza il connettore.
   - Per i connettori, puoi scegliere Create job (Crea processo) per creare un processo che utilizza il connettore.

Modifica di connettori e connessioni

Per modificare le informazioni archiviate nei connettori e nelle connessioni, devi utilizzare la pagina Connectors (Connettori).

Per modificare un connettore o una connessione

1. Nella console AWS Glue Studio, scegli Connectors (Connettori) nel pannello di navigazione della console.
2. Scegli il connettore o la connessione da modificare.
3. Seleziona Actions (Operazioni), quindi scegli Edit (Modifica).

   Puoi anche scegliere View details (Visualizza i dettagli) e nella pagina dei dettagli del connettore o della connessione scegliere Edit (Modifica).
4. Nella pagina Edit connector (Modifica connettore) o Edit connection (Modifica connessione), aggiorna le informazioni, quindi scegli Save (Salva).

Eliminazione di connettori e connessioni

Per eliminare connettori e connessioni, devi utilizzare la pagina Connectors (Connettori). Eliminando un connettore, è necessario eliminare anche tutte le connessioni create per tale connettore.

Come rimuovere i connettori da AWS Glue Studio

1. Nella console AWS Glue Studio, scegli Connectors (Connettori) nel pannello di navigazione della console.
2. Scegli il connettore o la connessione da eliminare.
3. Scegli Actions (Operazioni), quindi Delete (Elimina).

   Puoi anche scegliere View details (Visualizza i dettagli) e nella pagina dei dettagli del connettore o della connessione scegliere Delete (Elimina).
4. Conferma di voler rimuovere il connettore o la connessione inserendo **`Delete`** e quindi scegli Delete (Elimina).

   Eliminando un connettore, è necessario eliminare anche tutte le connessioni create per tale connettore.

I processi che utilizzano una connessione eliminata non funzioneranno più. Puoi modificare i processi per utilizzare un datastore diverso oppure rimuoverli. Per informazioni su come eliminare un processo, consulta Eliminazione dei processi.

Se elimini un connettore, la sottoscrizione per il connettore non viene annullata in Marketplace AWS. Per rimuovere una sottoscrizione per un connettore eliminato, segui le istruzioni riportate in Annullare una sottoscrizione per un connettore.

Annullare una sottoscrizione per un connettore

Dopo aver eliminato le connessioni e il connettore da AWS Glue Studio, se il connettore non è più necessario, puoi annullare la sottoscrizione in Marketplace AWS.

> **Note**
>
> Se annulli la sottoscrizione a un connettore, il connettore o la connessione non vengono rimossi dall'account. Qualsiasi processo che utilizza il connettore e le connessioni correlate non sarà più in grado di utilizzare il connettore e avrà esito negativo.
> Prima di annullare o eseguire nuovamente la sottoscrizione a un connettore da Marketplace AWS, è necessario eliminare le connessioni e i connettori esistenti associati a tale prodotto Marketplace AWS.

Per annullare la sottoscrizione a un connettore in Marketplace AWS

1. Accedi alla console Marketplace AWS all'indirizzo https://console.aws.amazon.com/marketplace
2. Scegli Manage subscriptions (Gestisci sottoscrizioni).
3. Nella pagina Manage subscriptions (Gestione degli abbonamenti), scegli Manage (Gestisci) accanto alla sottoscrizione del connettore che vuoi annullare.
4. Scegli Actions (Operazioni), quindi Cancel Subscriptions (Annulla sottoscrizione).
5. Seleziona la casella di controllo per acconsentire che le istanze in esecuzione siano addebitate all'account, quindi scegli Yes, cancel subscription (Sì, annulla sottoscrizione).

Sviluppo di connettori personalizzati

Puoi scrivere il codice che legge o scrive i dati nel datastore e formatta i dati per l'utilizzo con i processi AWS Glue Studio. Puoi creare connettori per datastore Spark, Athena e JDBC. Il codice di esempio pubblicato su GitHub fornisce una panoramica delle interfacce di base da implementare.

Per creare il codice del connettore è necessario un ambiente di sviluppo locale. Puoi usare qualsiasi IDE o anche solo un editor della riga di comando per scrivere il connettore. Esempi di ambienti di sviluppo includono:

- Un ambiente Scala locale con una libreria Maven ETL AWS Glue locale, come descritto in Sviluppo in locale con Scala nella Guida per gli sviluppatori di AWS Glue.
- IntelliJ IDE, scaricando l'IDE da https://www.jetbrains.com/idea/.

Argomenti

- Sviluppo dei connettori Spark
- Sviluppo di connettori Athena
- Sviluppo di connettori JDBC
- Esempi di utilizzo di connettori personalizzati con AWS Glue Studio
- Sviluppo di connettori AWS Glue per Marketplace AWS

Sviluppo dei connettori Spark

Puoi creare un connettore Spark con Spark DataSource API V2 (Spark 2.4) per leggere i dati.

Per creare un connettore Spark personalizzato

Segui i passaggi indicati nella libreria di AWS Glue GitHub esempio per lo sviluppo di connettori Spark, che si trova all'indirizzo https://github.com/aws-samples/ /tree/master/ /development/Spark/ README.md. aws-glue-samples GlueCustomConnectors

Sviluppo di connettori Athena

Puoi creare un connettore Athena utilizzabile da AWS Glue e AWS Glue Studio per eseguire query su un'origine dati personalizzata.

Per creare un connettore Athena personalizzato

Segui i passaggi nella libreria di AWS Glue GitHub esempio per lo sviluppo di connettori Athena, che si trova all'indirizzo https://github.com/aws-samples/ aws-glue-samples /tree/master/ GlueCustomConnectors /development/Athena.

Sviluppo di connettori JDBC

Puoi creare un connettore che utilizza JDBC per accedere ai datastore.

Per creare un connettore JDBC personalizzato

1. Installa le librerie runtime Spark AWS Glue nel tuo ambiente di sviluppo locale. Fate riferimento alle istruzioni nella libreria di esempio all'indirizzo https://github.com/aws-samples/ /tree/master/ /development/ /README.md. AWS Glue GitHub aws-glue-samples GlueCustomConnectors GlueSparkRuntime
2. Implementa il driver JDBC responsabile del recupero dei dati dall'origine dati. Fai riferimento alla documentazione Java per Java SE 8.

   Crea un punto di ingresso all'interno del tuo codice che AWS Glue Studio utilizza per individuare il connettore. Il campo Class name (Nome classe) dovrebbe essere il percorso completo del driver JDBC.
3. Usa l'API `GlueContext` per leggere i dati con il connettore. Se necessario, gli utenti possono aggiungere altre opzioni di input nella console AWS Glue Studio per configurare la connessione all'origine dati. Per un esempio di codice che mostra come leggere e scrivere in un database JDBC con un connettore JDBC personalizzato, consulta Valori di personalizzazione e connectionType Marketplace AWS.

Esempi di utilizzo di connettori personalizzati con AWS Glue Studio

Fai riferimento ai seguenti blog per esempi di utilizzo di connettori personalizzati:

- Sviluppo, test e implementazione di connettori personalizzati per gli archivi dati con AWS Glue
- Apache Hudi: Scrittura in tabelle Apache Hudi usando il connettore personalizzato AWS Glue
- Google BigQuery: migrazione dei dati da Google BigQuery ad Amazon S3 AWS Glue tramite connettori personalizzati
- Snowflake (JDBC): Esecuzione di trasformazione dati con Snowflake e AWS Glue
- SingleStore: Creazione di un sistema ETL rapido utilizzando e SingleStore AWS Glue
- Salesforce: Acquisizione di dati Salesforce in Amazon S3 con il connettore personalizzato CData JDBC con AWS Glue -
- MongoDB: Creazione di processi AWS Glue Spark ETL con Amazon DocumentDB (con compatibilità MongoDB) e MongoDB
- Amazon Relational Database Service (Amazon RDS): Creazione di processi AWS Glue Spark ETL con i propri driver JDBC per Amazon RDS
- MySQL (JDBC): https://github.com/aws-samples/ /blob/master/ /development/spark/ sql.Scala aws-glue-samples GlueCustomConnectors SparkConnectorMy

Sviluppo di connettori AWS Glue per Marketplace AWS

In qualità di partner AWS, puoi creare connettori personalizzati e caricarli su Marketplace AWS per la vendita ai clienti AWS Glue.

Il processo per lo sviluppo del codice del connettore è lo stesso dei connettori personalizzati, ma il processo di caricamento e verifica del codice del connettore è più dettagliato. Consulta le istruzioni contenute nella sezione Creazione di connettori disponibili sul sito Web. Marketplace AWS GitHub

Restrizioni per l'utilizzo di connettori e connessioni in AWS Glue Studio

Quando utilizzi connettori personalizzati o connettori da Marketplace AWS, tieni presente le seguenti restrizioni:

- L'API TestConnection non è supportata con le connessioni create per i connettori personalizzati.
- La crittografia delle password di connessione al catalogo dati non è supportata con connettori personalizzati.
- Non è possibile utilizzare i segnalibri di processo se specifichi un predicato di filtro per un nodo di origine dati che utilizza un connettore JDBC.
- La creazione di una connessione Marketplace non è supportata al di fuori dell'interfaccia utente di AWS Glue Studio.

Connessione alle origini dati tramite processi ETL visivi

Durante la creazione di un nuovo processo, è possibile utilizzare le connessioni per connettersi ai dati durante la modifica dei processi visivi ETL in AWS Glue. È possibile farlo aggiungendo nodi di origine che utilizzano connettori per leggere i dati e nodi di destinazione per specificare la posizione in cui scrivere i dati.

Argomenti

Modifica delle proprietà di un nodo di origine dati

Per specificare le proprietà di origine dati, è innanzitutto necessario scegliere un nodo di origine dati nel diagramma del processo. Quindi, sul lato destro nel pannello dei dettagli del nodo, puoi configurare le proprietà del nodo.

Per modificare le proprietà di un nodo di origine dati

1. Vai all'editor visivo per un processo nuovo o salvato.
2. Scegli un nodo di origine dati nel diagramma del processo.
3. Seleziona Node properties (Proprietà del nodo) nel pannello dei dettagli del nodo, quindi inserisci le seguenti informazioni:
   - Name (Nome): (facoltativo) immetti un nome da associare al nodo nel diagramma del processo. Questo nome deve essere univoco tra tutti i nodi per questo processo.
   - Node type (Tipo di nodo): il tipo di nodo determina l'azione eseguita dal nodo. Nell'elenco delle opzioni per Node type (Tipo di nodo), scegli uno dei valori elencati sotto l'intestazione Data source (Origine dati).
4. Configura le informazioni di Data source properties (Proprietà dell'origine dati). Per ulteriori informazioni, consulta le sezioni seguenti:

- Utilizzo delle tabelle del catalogo dati per l'origine dati
- Utilizzo di un connettore per l'origine dati
- Utilizzo di file in Amazon S3 per l'origine dati
- Utilizzo di un'origine dati di streaming

5. (Facoltativo) Dopo aver configurato le proprietà del nodo e dell'origine dati, puoi visualizzare lo schema per l'origine dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
6. (Facoltativo) Dopo aver configurato le proprietà del nodo e dell'origine dati, puoi visualizzare il set di dati dall'origine dati scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Utilizzo delle tabelle del catalogo dati per l'origine dati

Per tutte le origini dati ad eccezione di Amazon S3 e dei connettori, è necessario che esista una tabella in AWS Glue Data Catalog per il tipo di origine scelto. AWS Glue non crea la tabella in Data Catalog.

Per configurare un nodo di origine dati basato su una tabella del catalogo dati

1. Vai all'editor visivo per un processo nuovo o salvato.
2. Scegli un nodo di origine dati nel diagramma del processo.
3. Seleziona la scheda Data source properties (Proprietà dell'origine dati), quindi immetti le informazioni riportate di seguito:

   - S3 source type (Tipo di origine S3): (solo per origini dati Amazon S3) scegli l'opzione Select a Catalog table (Seleziona una tabella del catalogo) per utilizzare una tabella di AWS Glue Data Catalog.
   - Database: scegli il database nel catalogo dati contenente la tabella di origine da utilizzare per questo processo. Puoi utilizzare il campo di ricerca per cercare un database per nome.

- Table (Tabella): scegli dall'elenco la tabella associata ai dati di origine. Questa tabella deve esistere già in AWS Glue Data Catalog. Puoi utilizzare il campo di ricerca per cercare una tabella per nome.
- Partition predicate (Predicato di partizione): (solo per origini dati Amazon S3) inserisci un'espressione booleana basata su Spark SQL che includa solo le colonne di partizionamento. Ad esempio: `"(year=='2020' and month=='04')"`
- Temporary directory (Directory temporanea): (solo per le origini dati Amazon Redshift) inserisci un percorso per la posizione di una directory di processo in Amazon S3 in cui il processo ETL può scrivere risultati intermedi temporanei.
- Role associated with the cluster (Ruolo associato al cluster): (solo per le origini dati Amazon Redshift) inserisci un ruolo da utilizzare per il processo ETL che contiene le autorizzazioni per i cluster Amazon Redshift. Per ulteriori informazioni, consulta the section called “Autorizzazioni origine dati e destinazione dati”.

Utilizzo di un connettore per l'origine dati

Se per Node type (Tipo di nodo) selezioni un connettore, segui le istruzioni in Creazione di processi con connettori personalizzati per completare la configurazione delle proprietà dell'origine dati.

Utilizzo di file in Amazon S3 per l'origine dati

Se scegli Amazon S3 come origine dati, puoi scegliere:

- Un database e una tabella del catalogo dati.
- Un bucket, una cartella o un file in Amazon S3.

Se utilizzi un bucket Amazon S3 come origine dati, AWS Glue rileva lo schema dei dati nella posizione specificata da uno dei file o utilizzando il file specificato come file di esempio. Il rilevamento dello schema si verifica quando si utilizza il pulsante Infer schema (Deduci schema). Se modifichi la posizione di Amazon S3 o il file di esempio, devi selezionare nuovamente Infer schema (Deduci schema) per eseguire il rilevamento dello schema utilizzando le nuove informazioni.

Per configurare un nodo origine dati che legge direttamente dai file in Amazon S3

1. Vai all'editor visivo per un processo nuovo o salvato.
2. Scegli un nodo di origine dati nel diagramma del processo per un'origine Amazon S3.

3. Seleziona la scheda Data source properties (Proprietà dell'origine dati), quindi immetti le informazioni riportate di seguito:

   - TS3 source type (Tipo di origine S3): (solo per origini dati Amazon S3) scegli l'opzione S3 location (Posizione S3).
   - S3 URL (URL S3): inserisci il percorso del bucket, della cartella o del file Amazon S3 che contiene i dati per il processo. Puoi scegliere Browse S3 (Sfoglia S3) per selezionare il percorso dalle posizioni disponibili per il tuo account.
   - Recursive (Ricorsiva): scegli questa opzione se vuoi che AWS Glue legga i dati dai file nelle cartelle figlio nella posizione S3.

     Se le cartelle figlio contengono dati partizionati, AWS Glue non aggiunge le informazioni di partizione specificate nei nomi delle cartelle al Data Catalog. Considera, ad esempio, le seguenti cartelle in Amazon S3:

     ```
     S3://sales/year=2019/month=Jan/day=1
     S3://sales/year=2019/month=Jan/day=2
     ```

     Scegliendo Recursive (Ricorsiva) e selezionando `sales` come posizione S3, AWS Glue legge i dati in tutte le cartelle figlio, ma non crea partizioni per anno, mese o giorno.
   - Data format (Formato dei dati): scegli il formato in cui sono memorizzati i dati. Puoi scegliere JSON, CSV o Parquet. Il valore selezionato indica al processo AWS Glue come leggere i dati dal file di origine.

     > **Note**
     >
     > Se non selezioni il formato dei dati corretto, AWS Glue potrebbe dedurre lo schema correttamente, ma il processo non sarà in grado di analizzare correttamente i dati dal file di origine.

     Puoi immettere opzioni di configurazione aggiuntive, a seconda del formato scelto.

     - JSON (JavaScript Object Notation)
       - JsonPath (Percorso JSON): inserisci un percorso JSON che faccia riferimento a un oggetto usato per definire uno schema di tabella. Le espressioni di percorso JSON fanno sempre riferimento a una struttura JSON nello stesso modo in cui le espressioni XPath vengono utilizzate in combinazione con un documento XML. L' "oggetto membro root"

nel percorso JSON è sempre indicato come $, anche se si tratta di un oggetto o di una matrice. È possibile scrivere il percorso JSON in notazione punto o in notazione parentesi.

Per ulteriori informazioni sul percorso JSON, consulta JsonPath su GitHub.

- Records in source files can span multiple lines (I registri nei file di origine possono estendersi su più righe): seleziona questa opzione se un singolo registro può estendersi su più righe nel file CSV.

- CSV (valori separati da virgola)
  - Delimiter (Delimitatore): immetti un carattere per indicare il separatore di ogni voce di colonna nella riga, ad esempio `;` o `,`.
  - Escape character (Carattere di escape): immetti un carattere utilizzato come carattere di escape. Questo carattere indica che il carattere che segue immediatamente il carattere di escape deve essere preso alla lettera e non deve essere interpretato come un delimitatore.
  - Quote character (Carattere virgolette): immetti il carattere utilizzato per raggruppare stringhe separate in un singolo valore. Ad esempio, devi scegliere Double quote (") (virgolette doppie ["]) se nel file CSV sono presenti valori `"This is a single value"`.
  - Records in source files can span multiple lines (I registri nei file di origine possono estendersi su più righe): seleziona questa opzione se un singolo registro può estendersi su più righe nel file CSV.
  - First line of source file contains column headers (La prima riga del file di origine contiene le intestazioni di colonna): scegli questa opzione se la prima riga del file CSV contiene intestazioni di colonna anziché dati.
- Parquet (storage a colonne Apache Parquet)

  Non ci sono impostazioni aggiuntive da configurare per i dati memorizzati in formato Parquet.

- Partition predicate (Predicato di partizione): per partizionare i dati letti dall'origine dati, inserisci un'espressione booleana basata su Spark SQL che includa solo le colonne di partizionamento. Ad esempio: `"(year=='2020' and month=='04')"`
- Advanced options (Opzioni avanzate): espandi questa sezione se vuoi che AWS Glue rilevi lo schema dei dati in base a un file specifico.
  - Schema inference (Deduzione dello schema): seleziona l'opzione Choose a sample file from S3 (Scegli un file di esempio da S3) se vuoi utilizzare un file specifico invece di lasciare che AWS Glue scelga un file.

- Auto-sampled file (File con campionatura automatica): inserisci il percorso del file in Amazon S3 da utilizzare per dedurre lo schema.

Se stai apportando modifiche a un nodo dell'origine dati e al file di esempio selezionato, scegli Reload schema (Ricarica schema) per rilevare lo schema utilizzando il nuovo file di esempio.

4. Seleziona il pulsante Infer schema (Seleziona schema) per rilevare lo schema dai file di origine in Amazon S3. Se modifichi la posizione di Amazon S3 o il file di esempio, devi selezionare nuovamente Infer schema (Deduci schema) per rilevare lo schema utilizzando le nuove informazioni.

Utilizzo di un'origine dati di streaming

È possibile creare processi in streaming di estrazione, trasformazione e caricamento (ETL) che vengono eseguiti continuamente e consumano dati da origini di streaming in Amazon Kinesis Data Streams, Apache Kafka e Amazon Managed Streaming for Apache Kafka (Amazon MSK).

Per configurare le proprietà per un'origine dati di streaming

1. Vai all'editor grafico visivo per un processo nuovo o salvato.
2. Scegli un nodo origine dati nel grafico per Kafka o Kinesis Data Streams.
3. Seleziona la scheda , quindi immetti le informazioni seguenti:

   Kinesis

   - Kinesis source type (Tipo sorgente Kinesis): scegli l'opzione Stream details (Dettagli streaming) per utilizzare l'accesso diretto alla sorgente di streaming o Data Catalog table (Tabella Data Catalog) per utilizzare invece le informazioni archiviate in questa posizione.

     Se scegli Stream details (Dettagli streaming), specifica le seguenti informazioni aggiuntive.

     - Posizione del flusso di dati: scegli se il flusso di dati è associato all'utente corrente o se è associato a un altro utente.
     - Region (Regione): scegli la Regione AWS dove si trova il flusso di dati. Queste informazioni vengono utilizzate per costruire l'ARN per l'accesso al flusso di dati.
     - Stream ARN (ARN del flusso di dati): l'Amazon Resource Name (ARN) per l'endpoint del flusso di dati Kinesis. Se il flusso di dati si trova nell'account corrente, è possibile selezionarne il nome dall'elenco a discesa. Puoi utilizzare il campo di ricerca per cercare un flusso dei dati per nome o per ARN.

- Data format (Formato dei dati): scegli il formato utilizzato dal flusso di dati dall'elenco.

  AWS Glue rileva automaticamente lo schema dai dati in streaming.

  Se scegli Data Catalog table (Tabella Data Catalog), specifica le seguenti informazioni aggiuntive.

  - Database: (facoltativo) scegli il database nel Data Catalog di AWS Glue che contiene la tabella associata all'origine dati in streaming. Puoi utilizzare il campo di ricerca per cercare un database per nome.
  - Table (Tabella): (facoltativo) scegli dall'elenco la tabella associata ai dati di origine. Questa tabella deve esistere già nel AWS Glue Data Catalog. Puoi utilizzare il campo di ricerca per cercare una tabella per nome.
  - Detect schema (Rileva schema): scegli questa opzione per permettere ad AWS Glue Glue Studio di rilevare lo schema dai dati di streaming, anziché archiviare le informazioni sullo schema in una tabella di Data Catalog. Se scegli l'opzione Stream details (Dettagli streaming), questa opzione è abilitata automaticamente.

- Starting position (Posizione di inizio): per impostazione predefinita, il processo ETL utilizza l'opzione Earliest (Primo), il che significa che legge i dati a partire dal registro più vecchio disponibile nel flusso di dati. Puoi invece scegliere Latest (Più recente), che indica che il processo ETL dovrebbe iniziare a leggere subito dopo il registro più recente nel flusso di dati.
- Window size (Dimensione finestra): per impostazione predefinita, il processo ETL elabora e scrive i dati in finestre di 100 secondi. Ciò consente di elaborare i dati in modo efficiente e di eseguire aggregazioni su dati che arrivano più tardi del previsto. Puoi modificare questa dimensione della finestra per aumentare la tempestività o la precisione dell'aggregazione.

  I processi di streaming AWS Glue utilizzano i checkpoint anziché i segnalibri di processo per tenere traccia dei dati letti.
- Connection options (Opzioni di connessione): espandi questa sezione per aggiungere coppie chiave-valore per specificare opzioni di connessione aggiuntive. Per informazioni sulle opzioni che è possibile specificare qui, consulta "connectionType": "kinesis" nella Guida per gli sviluppatori di AWS Glue.

Kafka

- Apache Kafka source (Origine Apache Kafka): scegli l'opzione Stream details (Dettagli streaming) per utilizzare l'accesso diretto alla sorgente di streaming o Data Catalog table (Tabella Data Catalog) per utilizzare invece le informazioni archiviate in questa posizione.

  Se scegli Data Catalog table (Tabella Data Catalog), specifica le seguenti informazioni aggiuntive.

  - Database: (facoltativo) scegli il database nel Data Catalog di AWS Glue che contiene la tabella associata all'origine dati in streaming. Puoi utilizzare il campo di ricerca per cercare un database per nome.
  - Table (Tabella): (facoltativo) scegli dall'elenco la tabella associata ai dati di origine. Questa tabella deve esistere già nel AWS Glue Data Catalog. Puoi utilizzare il campo di ricerca per cercare una tabella per nome.
  - Detect schema (Rileva schema): scegli questa opzione per permettere ad AWS Glue di rilevare lo schema dai dati di streaming, anziché archiviare le informazioni sullo schema in una tabella di Data Catalog. Se scegli l'opzione Stream details (Dettagli streaming), questa opzione è abilitata automaticamente.

  Se scegli Stream details (Dettagli streaming), specifica le seguenti informazioni aggiuntive.

  - Connection name (Nome della connessione): scegli la connessione AWS Glue che contiene le informazioni di accesso e autenticazione per il flusso dei dati Kafka. È necessario utilizzare una connessione con le origini dati in streaming di Kafka. Se non esiste una connessione, per creare una connessione per il flusso di dati Kafka è possibile utilizzare la console AWS Glue.
  - Topic name (Nome argomento): inserisci il nome dell'argomento da cui leggere.
  - Data format (Formato dei dati): scegli il formato da utilizzare durante la lettura dei dati dal flusso di eventi Kafka.
- Starting position (Posizione di inizio): per impostazione predefinita, il processo ETL utilizza l'opzione Earliest (Primo), il che significa che legge i dati a partire dal registro più vecchio disponibile nel flusso di dati. Puoi invece scegliere Latest (Più recente), che indica che il processo ETL dovrebbe iniziare a leggere subito dopo il registro più recente nel flusso di dati.
- Window size (Dimensione finestra): per impostazione predefinita, il processo ETL elabora e scrive i dati in finestre di 100 secondi. Ciò consente di elaborare i dati in modo efficiente e

di eseguire aggregazioni su dati che arrivano più tardi del previsto. Puoi modificare questa dimensione della finestra per aumentare la tempestività o la precisione dell'aggregazione.

I processi di streaming AWS Glue utilizzano i checkpoint anziché i segnalibri di processo per tenere traccia dei dati letti.

- Connection options (Opzioni di connessione): espandi questa sezione per aggiungere coppie chiave-valore per specificare opzioni di connessione aggiuntive. Per informazioni sulle opzioni che è possibile specificare qui, consulta "connectionType": "kafka" nella Guida per gli sviluppatori di AWS Glue.

> **Note**
>
> Le anteprime dei dati non sono attualmente supportate per le origini dati di streaming.

Riferimenti

Best practice

- Creazione di una pipeline di servizi ETL per caricare i dati in modo incrementale da Amazon S3 a Amazon Redshift utilizzando AWS Glue

Programmazione ETL

- Tipi e opzioni di connessione per ETL in AWS Glue
- Valori di connectionType JDBC
- Opzioni avanzate per lo spostamento dei dati da e verso Amazon Redshift

Aggiunta di una connessione JDBC utilizzando i propri driver JDBC

Quando si utilizza una connessione JDBC è possibile utilizzare il proprio driver JDBC. Quando il driver predefinito utilizzato dal crawler AWS Glue non è in grado di connettersi a un database, è possibile utilizzare il proprio driver JDBC. Ad esempio, se desideri utilizzare SHA-256 con il tuo database Postgres e i driver Postgres precedenti non lo supportano, puoi utilizzare il tuo driver JDBC.

Origini dati supportate

Origini dati supportate	Origini dati non supportate
MySQL	Snowflake
Postgres	
Oracle	
Redshift	
SQL Server	
Aurora*	

* Supportato se si utilizza il driver JDBC nativo. Non è possibile avvalersi di tutte le funzionalità del driver.

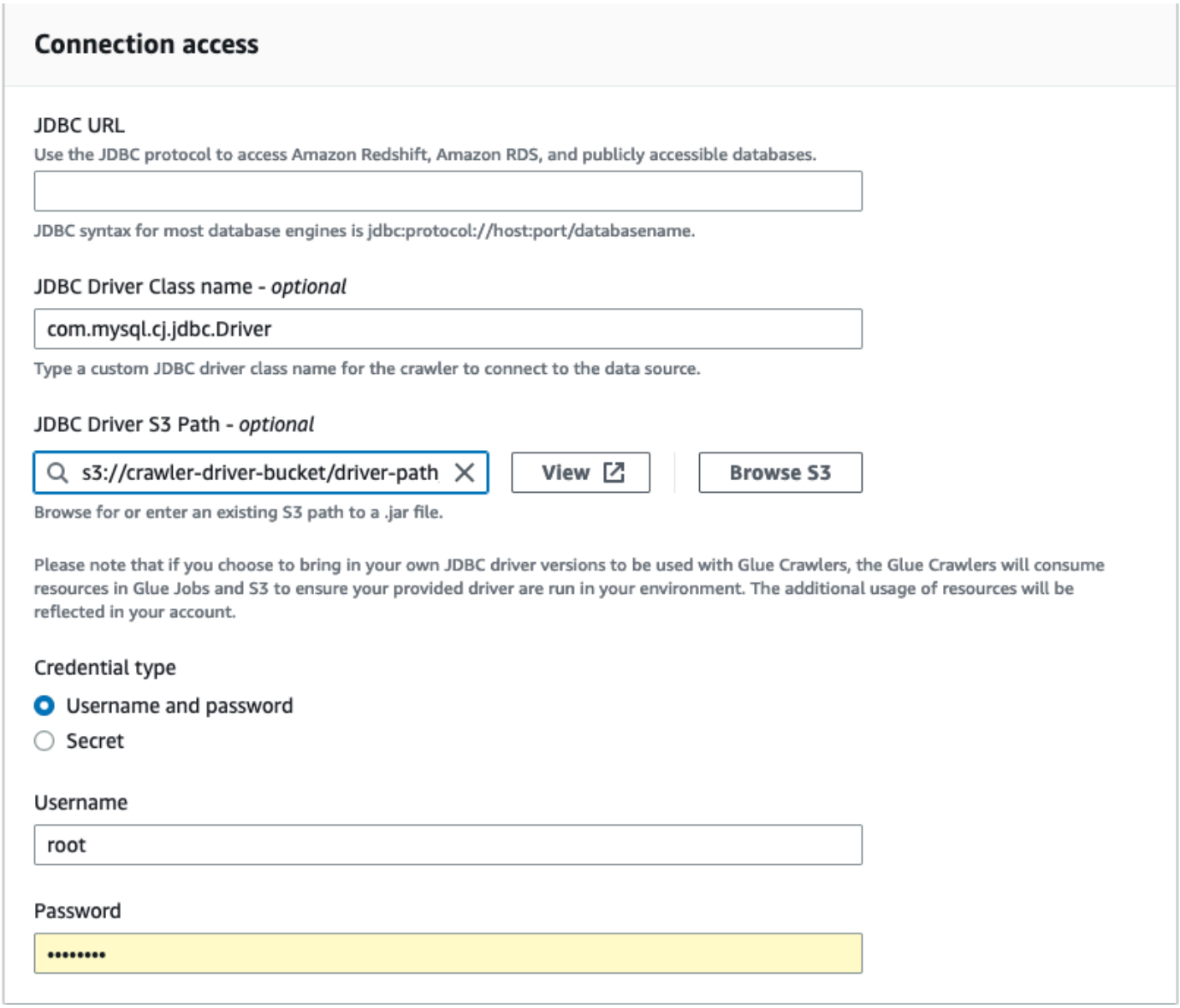
Aggiunta del driver JDBC a una connessione JDBC

> **Note**
>
> Se scegli di importare le tue versioni dei driver JDBC, i crawler AWS Glue consumeranno risorse nei processi AWS Glue e nei bucket Amazon S3 per garantire che i driver forniti vengano eseguiti nel tuo ambiente. L'utilizzo aggiuntivo delle risorse si rifletterà nel tuo account. Il costo dei crawler e dei processi AWS Glue rientra nella categoria AWS Glue in fattura. Inoltre, è importante sottolineare che anche se si fornisce il proprio driver JDBC, ciò non implica automaticamente che il crawler possa sfruttare tutte le funzionalità offerte da tale driver.

Per aggiungere il proprio driver JDBC a una connessione JDBC:

1. Aggiungi il file del driver JDBC a una posizione Amazon S3. È possibile creare un bucket e/o una cartella o utilizzare un bucket e/o una cartella esistente.
2. Nella console AWS Glue, scegli Connessioni nel menu a sinistra sotto Catalogo dati, quindi crea una nuova connessione.

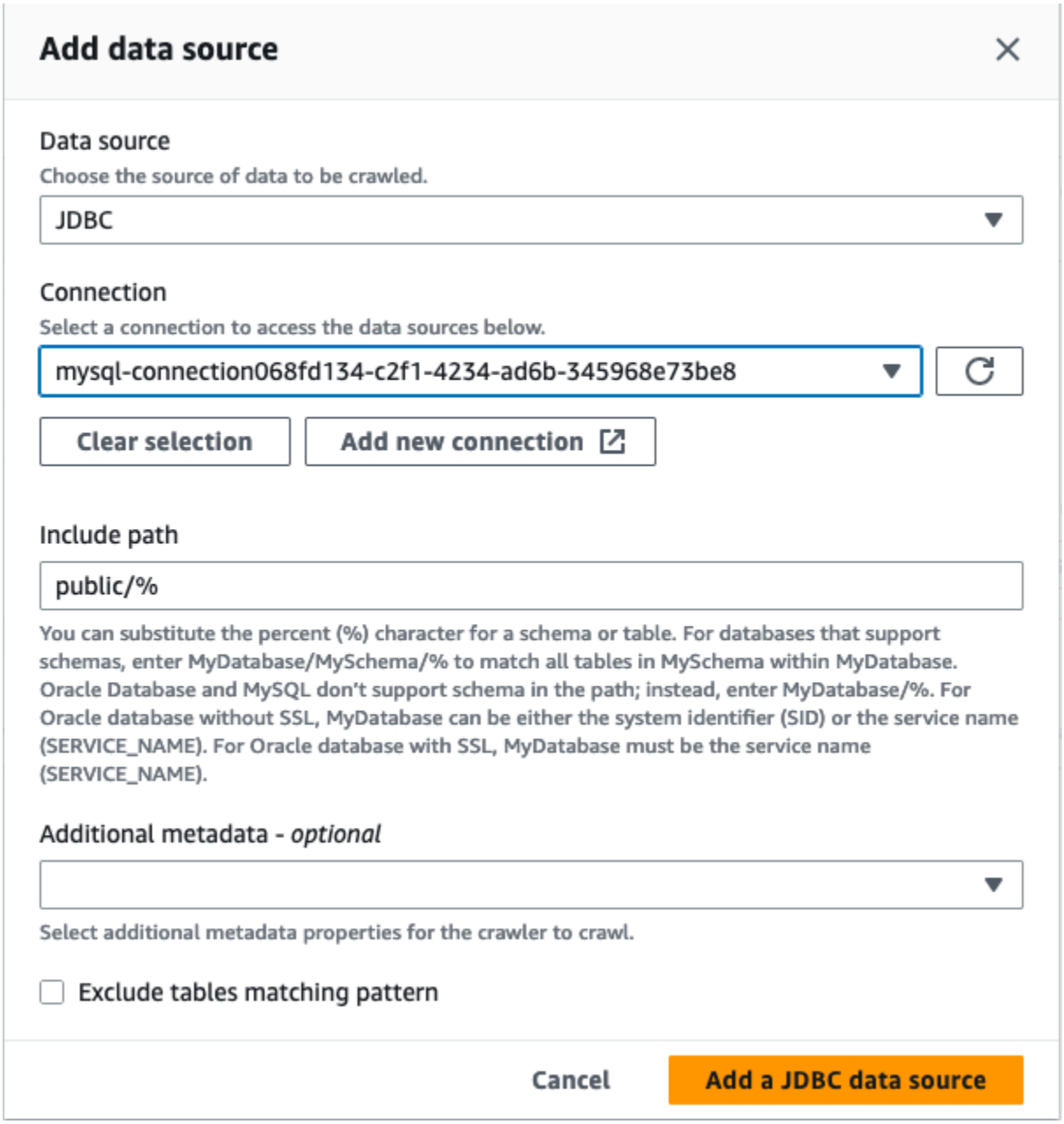
3. Completa i campi per le Proprietà di connessione e scegli JDBC per il Tipo di connessione.
4. In Accesso alla connessione, inserisci l'URL JDBC e il Nome della classe del driver JDBC - facoltativo. Il nome della classe del driver deve riferirsi a un'origine dati supportata dai crawler AWS Glue.



5. Scegli il percorso Amazon S3 in cui si trova il driver JDBC nel campo Percorso del driver JDBC Amazon S3 - facoltativo.
6. Se inserisci un nome utente e una password o un segreto, completa i campi per Tipo di credenziale. Al termine, scegli Crea connessione.

> **Note**
>
> Il test delle connessioni personalizzate non è attualmente supportato. Quando esegui il crawling dell'origine dati con un driver JDBC fornito da te, il crawler salta questo passaggio.

7. Aggiungi la connessione appena creata a un crawler. Nella console AWS Glue, scegli Crawler nel menu a sinistra sotto Catalogo dati, quindi crea un nuovo crawler.
8. Nella procedura guidata Aggiungi crawler, nel passaggio 2 scegli Aggiungi un'origine dati.



9. Scegli JDBC come origine dati e scegli la connessione creata nei passaggi precedenti. Completa
10. Per utilizzare il tuo driver JDBC con un crawler AWS Glue, aggiungi le seguenti autorizzazioni al ruolo utilizzato dal crawler:

    - Concedi le autorizzazioni per le seguenti operazioni di processo: `CreateJob`, `DeleteJob`, `GetJob`, `GetJobRun`, `StartJobRun`.
    - Concedi le autorizzazioni per le operazioni IAM: `iam:PassRole`
    - Concedi le autorizzazioni per le operazioni di Amazon S3: `s3:DeleteObjects`, `s3:GetObject`, `s3:ListBucket`, `s3:PutObject`.
    - Concedi l'accesso principale del servizio al bucket/cartella nella policy IAM.

    Policy IAM di esempio:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name/driver-parent-folder/driver.jar",
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

11. Se si utilizza un VPC, è necessario consentire l'accesso all'endpoint AWS Glue creando l'endpoint dell'interfaccia e aggiungendolo alla tabella di routing. Per ulteriori informazioni, consulta la pagina Creating an interface VPC endpoint for AWS Glue

12. Se utilizzi la crittografia nel tuo Catalogo dati, crea l'endpoint di interfaccia AWS KMS e aggiungilo alla tabella di routing. Per ulteriori informazioni, consulta la pagina Creating a VPC endpoint for AWS KMS.

Test di una connessione AWS Glue

> **Note**
>
> Per testare una connessione, è necessario utilizzare la pagina Connessioni della console AWS Glue. Il test delle connessioni personalizzate non è supportato.
> Per impostazione predefinita, le nuove istanze di database Amazon RDS utilizzeranno il nuovo certificato. `rds-ca-rsa2048-g1` AWS Gluejob e Test Connection si basano attualmente su. `certificate rds-ca-2019` Per connettere nuove istanze Amazon RDS con AWS Glue job o Test Connection, imposta l'istanza per utilizzare il certificato `rds-ca-2019` tramite la AWS console o. AWS CLI Per ulteriori informazioni, consulta la sezione Utilizzo di SSL/TLS per crittografare una connessione a un'istanza DB nella guida per l'utente di Amazon RDS per una guida dettagliata.

Come best practice, prima di utilizzare una connessione AWS Glue in un processo ETL, utilizza la console AWS Glue per testare la connessione. AWS Glue utilizza i parametri nella connessione per confermare che può accedere al datastore e segnala eventuali errori. Per informazioni sulle connessioni AWS Glue, consulta Connessione ai dati.

Per testare una connessione AWS Glue

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione, in Catalogo dati, seleziona Connessioni. È inoltre possibile scegliere Connessioni dati sopra Catalogo dati nel riquadro di navigazione.
3. In Connessioni, seleziona la casella di controllo accanto alla connessione desiderata, quindi scegli Operazioni. Nel menu a discesa, seleziona Testa connessione.
4. Nella finestra di dialogo Test connection (Connessione di prova), seleziona un ruolo o scegli Create IAM role (Crea ruolo IAM) per passare alla console AWS Identity and Access Management (IAM) e creare un nuovo ruolo. Il ruolo deve disporre delle autorizzazioni per il datastore.

5. Scegli Conferma.

   Il test inizia e può richiedere diversi minuti per completare. Se il test fallisce, scegli Risoluzione dei problemi per visualizzare i passaggi per risolvere il problema.
6. Scegli Log per visualizzare i log in. CloudWatch Devi disporre delle autorizzazioni IAM necessarie per visualizzare i log. Per ulteriori informazioni, consulta AWSManaged (Predefined) Policies for CloudWatch Logs nella Amazon CloudWatch Logs User Guide.

Configurazione delle chiamate AWS affinché passino attraverso il tuo VPC

Il parametro speciale del processo `disable-proxy-v2` consente di instradare le chiamate ai servizi come Amazon S3, CloudWatch e AWS Glue tramite il tuo VPC. Per impostazione predefinita, AWS Glue utilizza un proxy locale per inviare traffico attraverso il VPC AWS Glue per scaricare script e librerie da Amazon S3, inviare richieste a CloudWatch per la pubblicazione di log e metriche e inviare richieste a AWS Glue per l'accesso ai cataloghi dati. Questo proxy permette al processo di funzionare normalmente anche se il VPC non configura un routing corretto per altri servizi AWS come Amazon S3, CloudWatch e AWS Glue. AWS Glue ora offre un parametro per disattivare questo comportamento. Per ulteriori informazioni, consulta Parametri di processo utilizzati da AWS Glue. AWS Glue continuerà a utilizzare il proxy locale per pubblicare i registri di CloudWatch dei processi AWS Glue.

> **Note**
>
> - Questa funzionalità è supportata per processi AWS Glue con versione AWS Glue 2.0 e versioni successive. Quando utilizzi questa caratteristica, devi assicurarti che il tuo VPC abbia configurato un percorso verso Amazon S3, tramite un endpoint NAT o VPC di servizio.
> - Il parametro del processo obsoleto `disable-proxy` inoltra le tue chiamate ad Amazon S3 solo per il download di script e librerie tramite il tuo VPC. Consigliamo invece di utilizzare il nuovo parametro `disable-proxy-v2`.

Esempio di utilizzo

Creare un processo AWS Glue con `disable-proxy-v2`:

```
aws glue create-job \
    --name no-proxy-job \
    --role GlueDefaultRole \
    --command "Name=glueetl,ScriptLocation=s3://my-bucket/glue-script.py" \
    --connections Connections="traffic-monitored-connection" \
    --default-arguments '{"--disable-proxy-v2" : "true"}'
```

La connessione a un archivio dati JDBC in un VPC

In genere, le risorse vengono create in Amazon Virtual Private Cloud (Amazon VPC) per impedirne l'accesso tramite rete internet pubblica. Per impostazione predefinita, AWS Glue non è in grado di accedere alle risorse all'interno di un VPC. Per permettere ad AWS Glue di accedere a risorse all'interno del VPC, devi fornire altre informazioni di configurazione specifiche di VPC, tra cui gli ID sottorete VPC e gli ID gruppo di sicurezza. AWS Glue usa queste informazioni per configurare interfacce di rete elastiche che permettono alla funzione di connettersi in modo sicuro ad altre risorse nel VPC privato.

Quando utilizzi un endpoint VPC, aggiungilo alla tabella di routing. Per ulteriori informazioni, consulta le pagine Creating an interface VPC endpoint for AWS Glue e Prerequisiti.

Quando utilizzi la crittografia in Catalogo dati, crea l'endpoint dell'interfaccia KMS e aggiungilo alla tabella di routing. Per ulteriori informazioni, consulta la pagina Creating a VPC endpoint for AWS KMS.

Accesso a dati VPC mediante interfacce di rete elastiche

Quando AWS Glue si connette a un archivio dati JDBC in un VPC, AWS Glue crea un'interfaccia di rete elastica (con il prefisso `Glue_`) nel tuo account per accedere ai dati VPC. Non puoi eliminare questa interfaccia di rete finché è collegata ad AWS Glue. Come parte della creazione dell'interfaccia di rete elastica, AWS Glue associa all'interfaccia uno o più gruppi di sicurezza. Per permettere ad AWS Glue di creare l'interfaccia di rete, i gruppi di sicurezza associati alla risorsa devono permettere l'accesso in entrata con una regola di origine. Questa regola contiene un gruppo di sicurezza associato alla risorsa. In questo modo, l'interfaccia di rete elastica accede all'archivio dati con lo stesso gruppo di sicurezza.

Per permettere ad AWS Glue di comunicare con i suoi componenti, specifica un gruppo di sicurezza con una regola autoreferenziale per il traffico in entrata per tutte le porte TCP. Se crei una regola autoreferenziata, puoi limitare l'origine allo stesso gruppo di sicurezza nel VPC e non aprirla a tutte le

reti. Il gruppo di sicurezza predefinito per il tuo VPC potrebbe già avere una regola autoreferenziata in entrata per `ALL Traffic`.

Puoi creare regole nella console Amazon VPC. Per aggiornare le impostazioni della regola tramite AWS Management Console, passa alla console VPC (https://console.aws.amazon.com/vpc/) e seleziona il gruppo di sicurezza appropriato. Specifica la regola in entrata per `ALL TCP` così da avere come origine lo stesso nome gruppo di sicurezza. Per ulteriori informazioni sulle regole dei gruppi di sicurezza, consulta Gruppi di sicurezza per il tuo VPC.

A ogni interfaccia di rete elastica viene assegnato un indirizzo IP privato dall'intervallo di indirizzi IP nelle sottoreti che hai specificato. All'interfaccia di rete non viene assegnato alcun indirizzo IP pubblico. AWS Glue richiede l'accesso a Internet, ad esempio per accedere ai servizi AWS che non hanno endpoint VPC. È·possibile configurare un'istanza NAT (Network Address Translation) all'interno del VPC oppure puoi usare il gateway NAT Amazon VPC. Per ulteriori informazioni, consulta Gateway NAT nella Guida per l'utente di Amazon VPC. Non puoi utilizzare direttamente un gateway internet collegato al VPC come percorso nella tabella di instradamento della sottorete perché ciò richiede che l'interfaccia di rete abbia indirizzi IP pubblici.

Gli attributi di rete VPC `enableDnsHostnames` e `enableDnsSupport` devono essere impostati su true. Per ulteriori informazioni, consulta Utilizzo del DNS con il tuo VPC.

> **Important**
>
> Non inserire l'archivio dati in una sottorete pubblica o in una sottorete privata che non abbia accesso a Internet. Collegalo invece solo alle sottoreti private che hanno accesso a Internet tramite un'istanza NAT o un gateway NAT Amazon VPC.

Proprietà dell'interfaccia di rete elastica

Per creare l'interfaccia di rete elastica, sono necessarie le seguenti proprietà:

VPC

Il nome del VPC che contiene l'archivio dati.

Sottorete

Il nome della sottorete nel VPC che contiene l'archivio dati.

Gruppi di sicurezza

Gruppi di sicurezza associati all'archivio dati. AWS Glue associa questi gruppi di sicurezza all'interfaccia di rete elastica collegata alla sottorete VPC. Per consentire ai componenti di AWS Glue di comunicare e di impedire l'accesso da altre reti, almeno un gruppo di sicurezza a scelta deve specificare una regola in entrata autoreferenziale per tutte le porte TCP.

Per informazioni sulla gestione di un VPC con Amazon Redshift, consulta Gestione di cluster in Amazon Virtual Private Cloud (VPC).

Per ulteriori informazioni sulla gestione di un VPC con Amazon Relational Database Service (Amazon RDS), consulta Uso di un'istanza database Amazon RDS in un VPC.

Utilizzo di una connessione MongoDB o MongoDB Atlas

Dopo aver creato una connessione per MongoDB o MongoDB Atlas, puoi utilizzarla nel processo ETL. Crei una tabella in AWS Glue Data Catalog e specifichi la connessione MongoDB o MongoDB Atlas per l'attributo `connection` della tabella.

AWS Glue memorizza la connessione `url` e le credenziali nella connessione MongoDB. I formati di URI di connessione sono i seguenti:

- Per MongoDB: mongodb://host:port/database. L'host può essere un nome host, un indirizzo IP o un socket di dominio UNIX. Se la stringa di connessione non specifica una porta, utilizza la porta MongoDB predefinita, 27017.
- Per MongoDB Atlas: mongodb+srv://server.example.com/database. L'host può essere un nome host che corrisponde a un record DNS SRV. Il formato SRV non richiede una porta e utilizzerà la porta MongoDB predefinita, 27017.

Altre opzioni possono essere specificate nello script di processo. Per ulteriori informazioni, consulta the section called “Connessione MongoDB”.

Crawling di un archivio di dati Amazon S3 utilizzando un endpoint VPC

Per motivi di sicurezza, audit o controllo, puoi consentire l'accesso all'archivio dati Amazon S3 o alle tabelle Catalogo dati supportate da Amazon S3 solo tramite un ambiente Amazon Virtual Private

Cloud (Amazon VPC). In questo argomento viene descritto come creare e testare una connessione all'archivio dati Amazon S3 o alle tabelle Catalogo dati supportate da Amazon S3 in un endpoint VPC utilizzando il tipo di connessione `Network`.

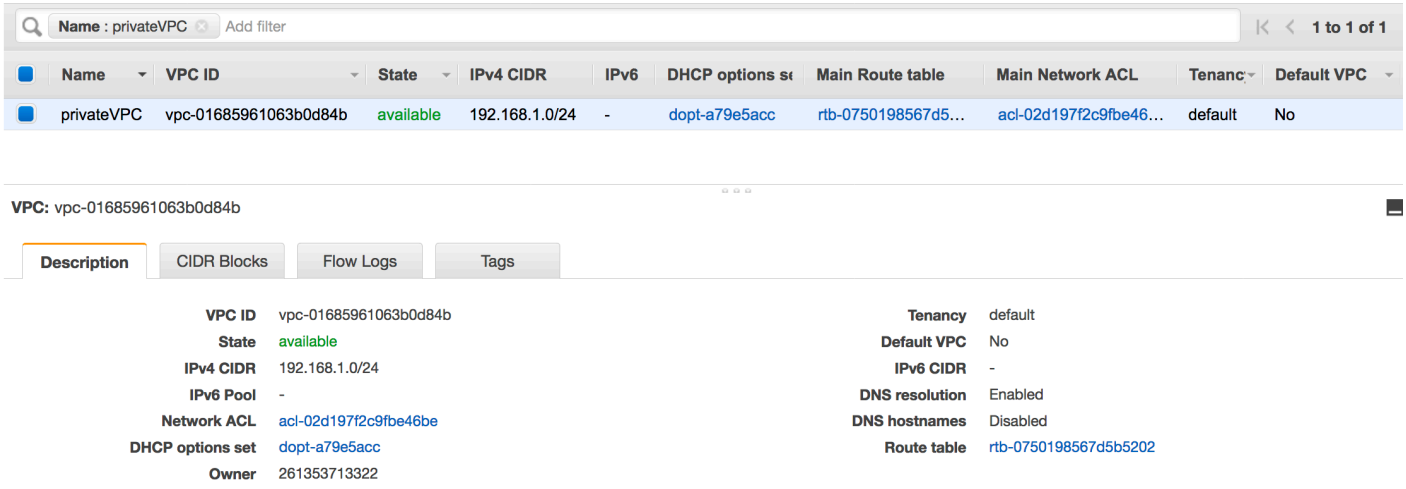
Esegui le attività seguenti per eseguire un crawler nell'archivio dati:

- the section called “Prerequisiti”
- the section called “Creazione della connessione ad Amazon S3”
- the section called “Test della connessione ad Amazon S3”
- the section called “Creazione di un crawler per un archivio di dati Amazon S3”
- the section called “Esecuzione di un crawler”

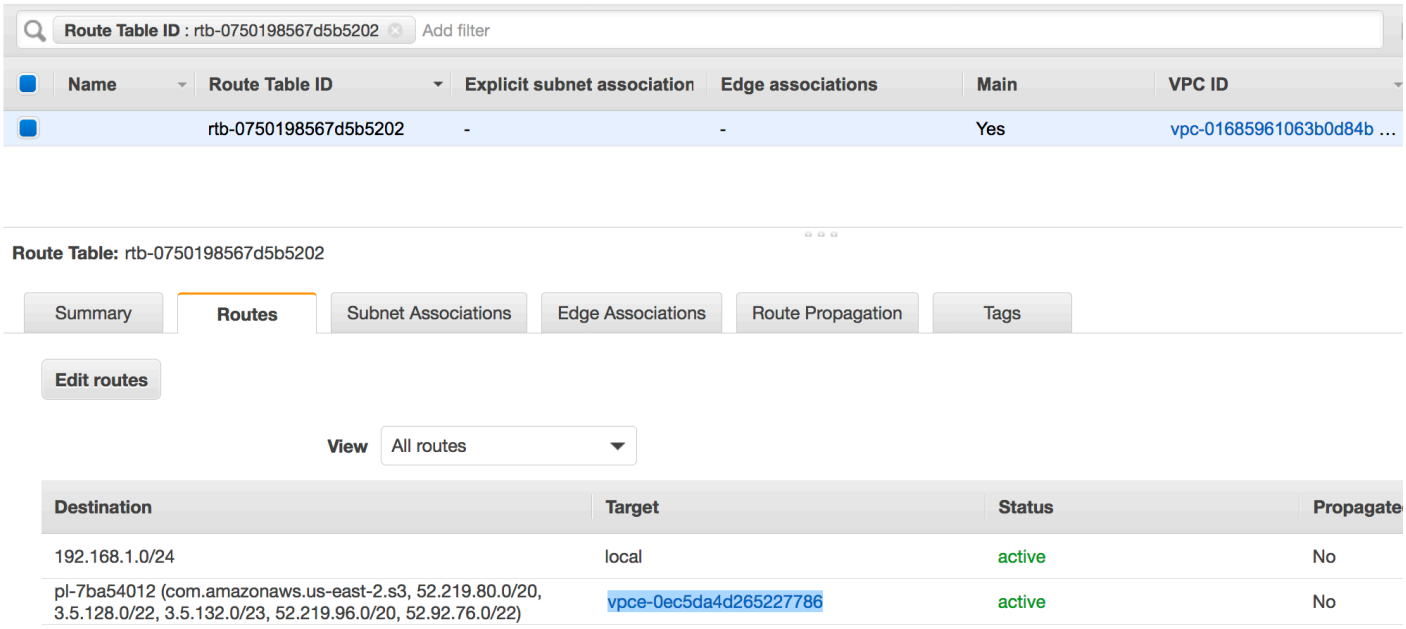
Prerequisiti

Verifica di aver soddisfatto questi prerequisiti per configurare il datastore Amazon S3 affinché vi si possa accedere solo tramite un ambiente Amazon Virtual Private Cloud (Amazon VPC).

- Un VPC configurato. Ad esempio: vpc-01685961063b0d84b. Per ulteriori informazioni, consulta le Nozioni di base su Amazon VPC nella Guida per l'utente di Amazon VPC.
- Un endpoint Amazon S3 collegato al VPC. Ad esempio: vpc-01685961063b0d84b. Per ulteriori informazioni, consulta Endpoint per Amazon S3 nella Guida per l'utente di Amazon VPC.



- Una voce route che punta all'endpoint VPC. Ad esempio vpce-0ec5da4d265227786 nella tabella di routing utilizzata dall'endpoint VPC (vpce-0ec5da4d265227786).



- Una lista di controllo degli accessi di rete collegata al VPC consente il traffico.
- Un gruppo di sicurezza collegato al VPC consente il traffico.

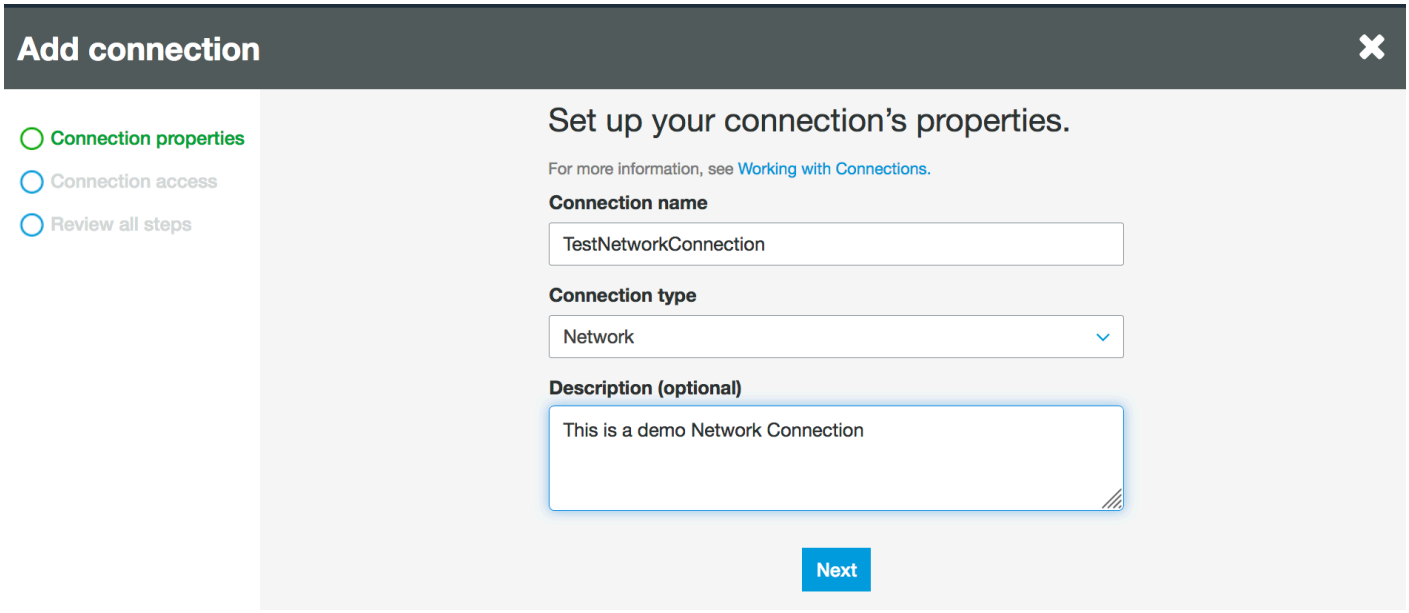
Creazione della connessione ad Amazon S3

In genere, le risorse vengono create in Amazon Virtual Private Cloud (Amazon VPC) per impedirne l'accesso tramite rete internet pubblica. Per impostazione predefinita, AWS Glue non è in grado di accedere alle risorse all'interno di un VPC. Per permettere a AWS Glue di accedere alle risorse nel VPC, devi fornire informazioni di configurazione specifiche VPC aggiuntive che includano ID di sottorete VPC e ID dei gruppi di sicurezza. Per creare una connessione `Network`, è necessario specificare le informazioni seguenti:

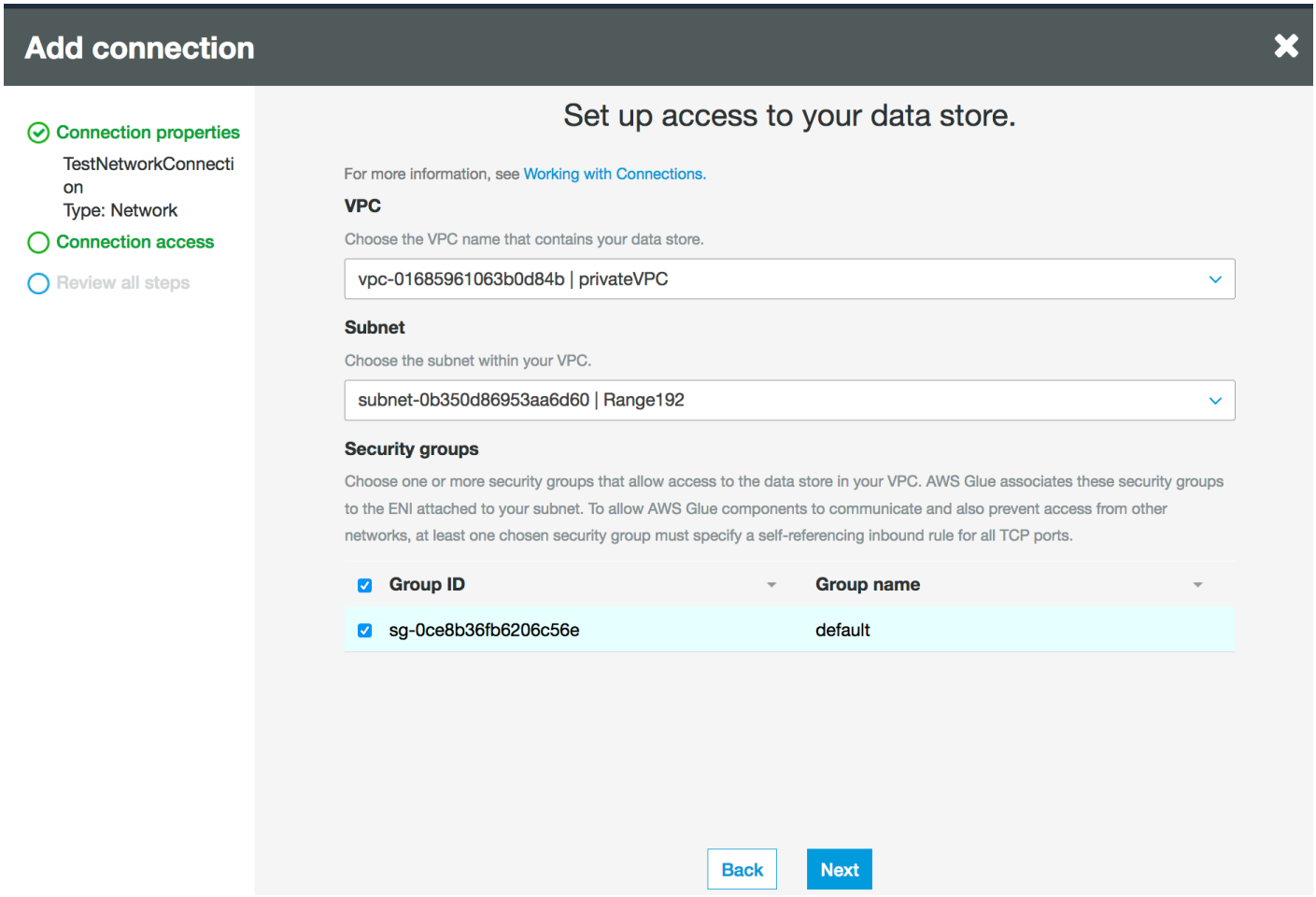
- Un ID VPC
- Una sottorete all'interno del VPC
- Un gruppo di sicurezza

Per impostare una connessione `Network`:

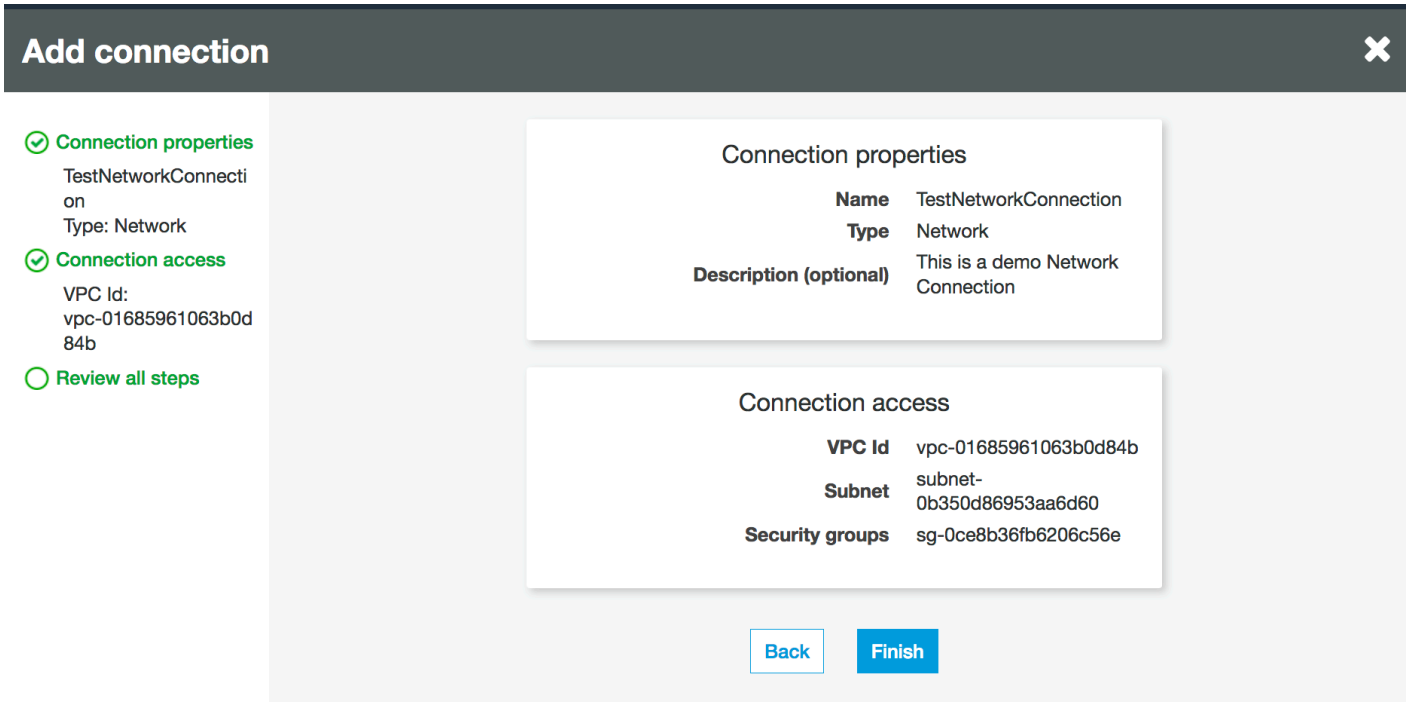
1. Scegli Add connection (Aggiungi connessione) nel pannello di navigazione della console AWS Glue.
2. Inserisci il nome della connessione e scegli Network (Rete) come tipo di connessione. Seleziona Successivo.



3. Configura le informazioni su VPC, sottorete e gruppi di sicurezza.
   - VPC: scegli il nome del VPC che contiene l'archivio dati.
   - Sottorete: scegli una sottorete nel VPC.
   - Gruppi di sicurezza: scegli uno o più gruppi di protezione che consentono l'accesso all'archivio dati nel VPC.



4. Seleziona Next (Successivo).
5. Verifica le informazioni di connessione e scegli Finish (Termina).



Test della connessione ad Amazon S3

Una volta creata la connessione di `Network`, puoi testare la connettività al tuo archivio dati Amazon S3 in un endpoint VPC.

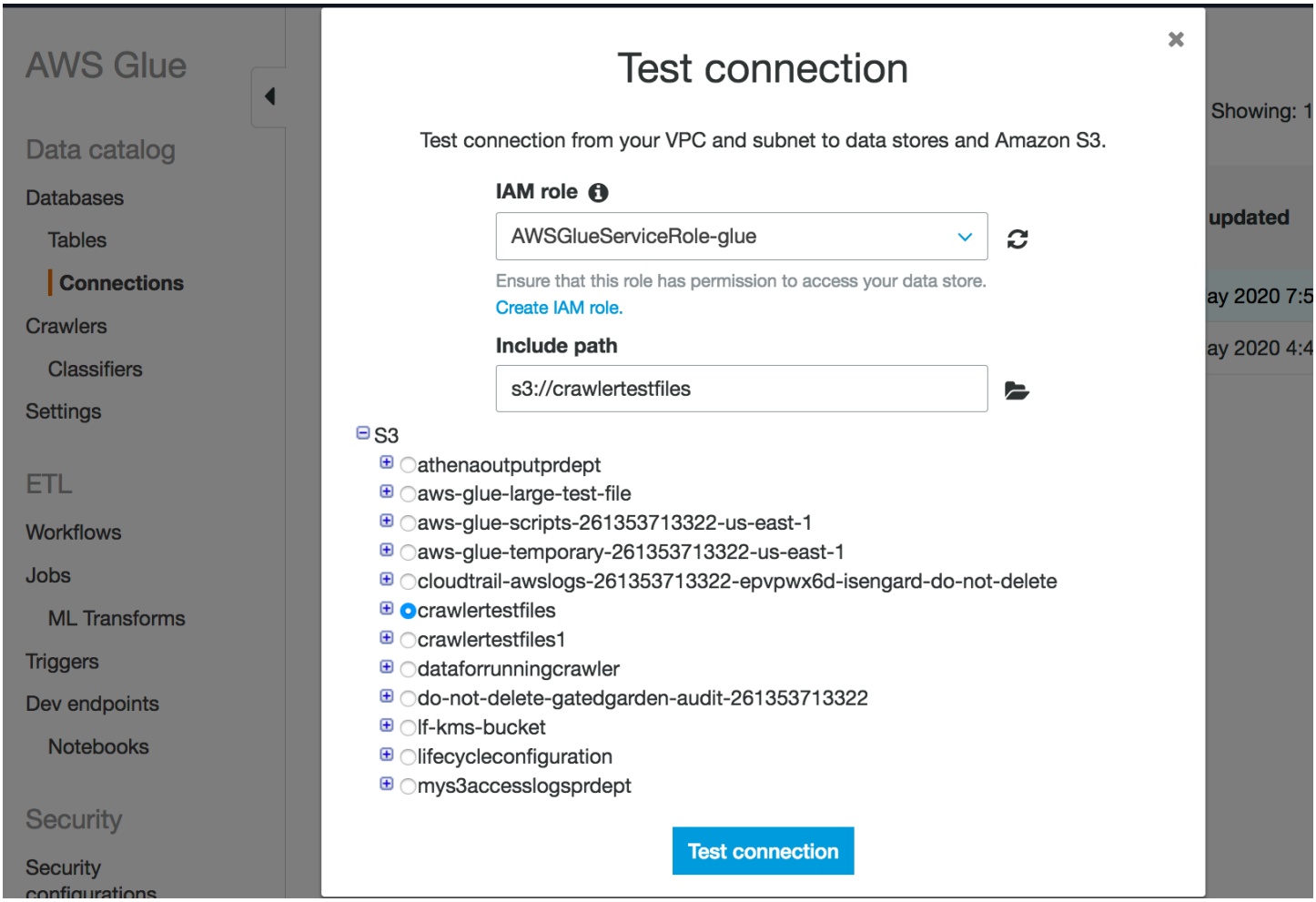
Durante il test di una connessione possono verificarsi i seguenti errori:

- ERRORE DI CONNESSIONE A INTERNET: indica un problema di connessione a Internet
- ERRORE DI BUCKET NON VALIDO: indica un problema con il bucket Amazon S3
- ERRORE DI CONNESSIONE S3: indica un errore di connessione ad Amazon S3
- ERRORE DI TIPO DI CONNESSIONE: indica che il tipo di connessione non ha il valore previsto, `NETWORK`
- TIPO DI TEST DI CONNESSIONE NON VALIDO: indica un problema con il tipo di test della connessione di rete
- DESTINAZIONE NON VALIDA: indica che il bucket Amazon S3 non è stato specificato correttamente

Per testare una connessione `Network`:

1. Seleziona la connessione Network (Rete) nella console AWS Glue.

2. Scegli Test Connection (Connessione di prova).
3. Scegli il ruolo IAM creato nel passaggio precedente e specifica un bucket Amazon S3.
4. Per verificare la connessione, scegli Test connection (Testa connessione). Potrebbero essere necessari alcuni istanti prima che il risultato venga visualizzato.



Se viene visualizzato un errore, controlla quanto segue:

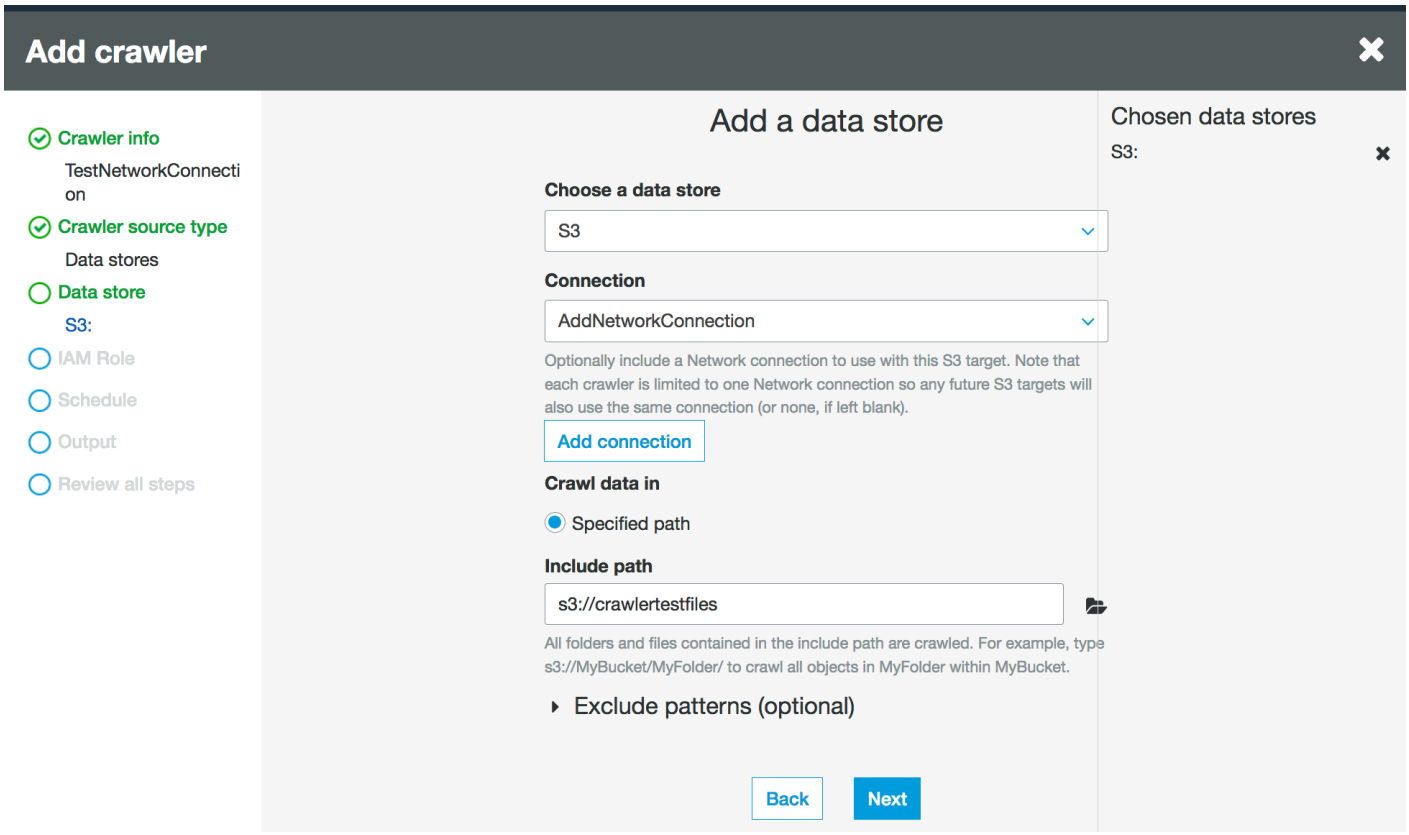
- I privilegi corretti vengono forniti al ruolo selezionato.
- Viene fornito il bucket Amazon S3 corretto.
- I gruppi di sicurezza e la lista di controllo degli accessi di rete consentono il traffico in entrata e in uscita necessario.
- Il VPC specificato è connesso a un endpoint VPC Amazon S3.

Dopo aver testato correttamente la connessione, è possibile creare un crawler.

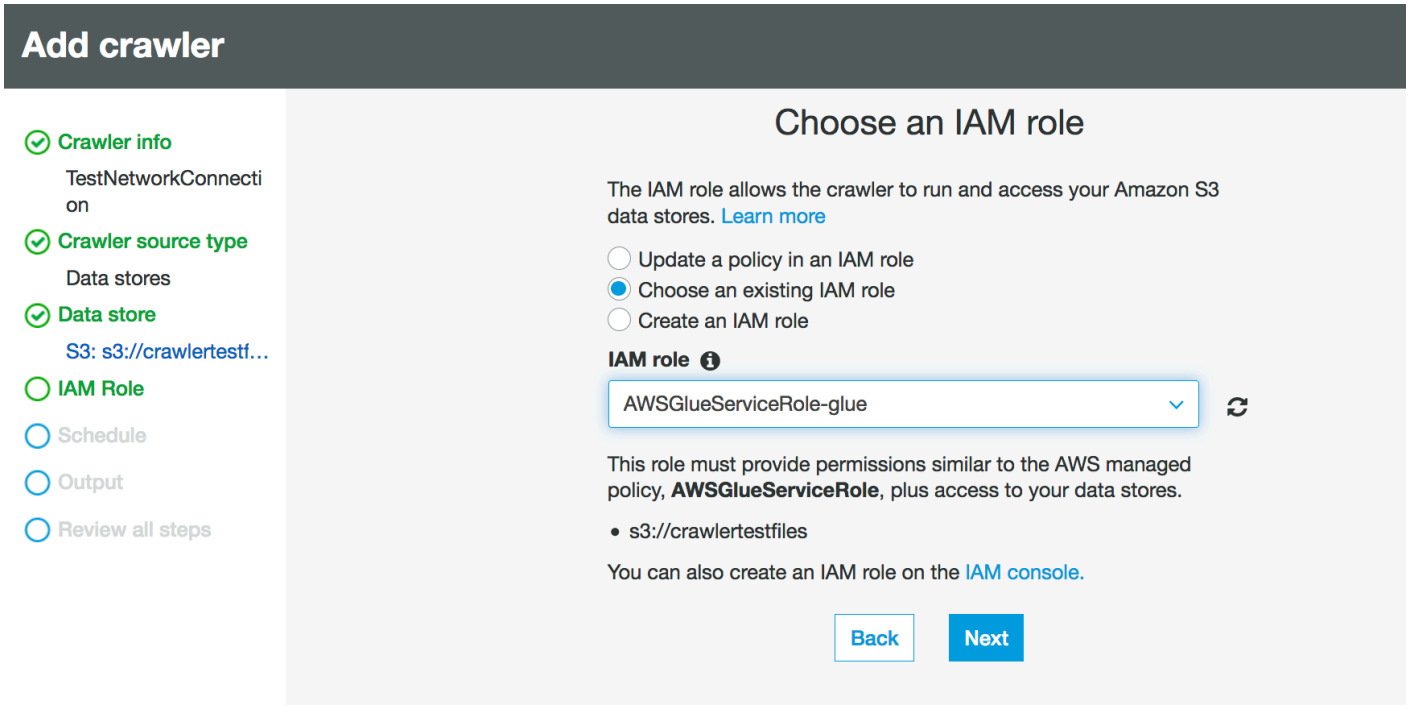
Creazione di un crawler per un archivio di dati Amazon S3

Ora è possibile creare un crawler che specifichi la connessione di `Network` creata. Per ulteriori dettagli sulla creazione di un crawler, consulta the section called “Uso di crawler nella console”.

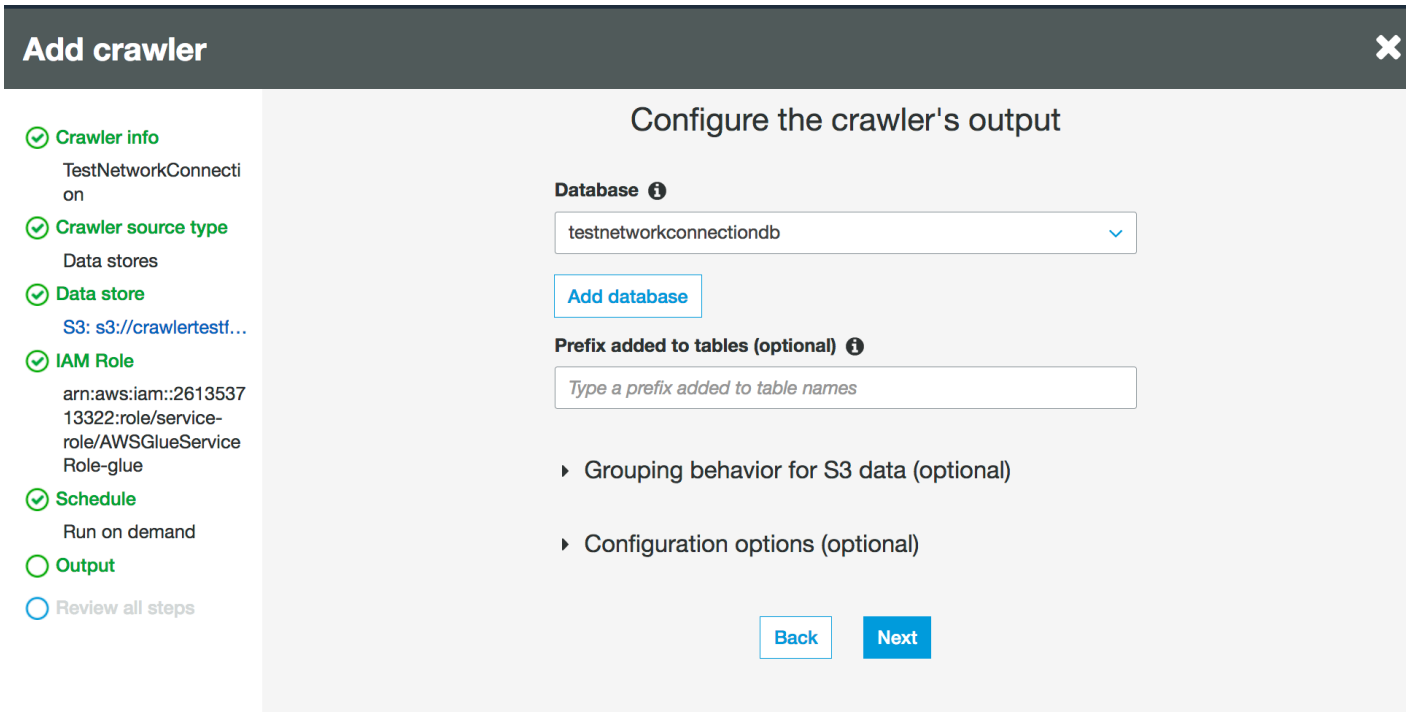
1. Inizia scegliendo Crawlers (Crawler) nel pannello di navigazione nella console AWS Glue.
2. Scegli Add crawler (Aggiungi crawler).
3. Specifica il nome del crawler, quindi scegli Next (Avanti).
4. Quando viene richiesto di specificare l'origine dati, seleziona S3 e specifica il prefisso del bucket Amazon S3 e la connessione creata in precedenza.



5. Se necessario, aggiungi un altro archivio dati sulla stessa connessione di rete.
6. Scegli il ruolo IAM. Il ruolo IAM deve consentire l'accesso al servizio AWS Glue e al bucket Amazon S3. Per ulteriori informazioni, consulta the section called “Uso di crawler nella console”.



7. Definisci la pianificazione per il crawler.
8. Scegli un database esistente nel catalogo dati oppure crea una nuova voce del database.

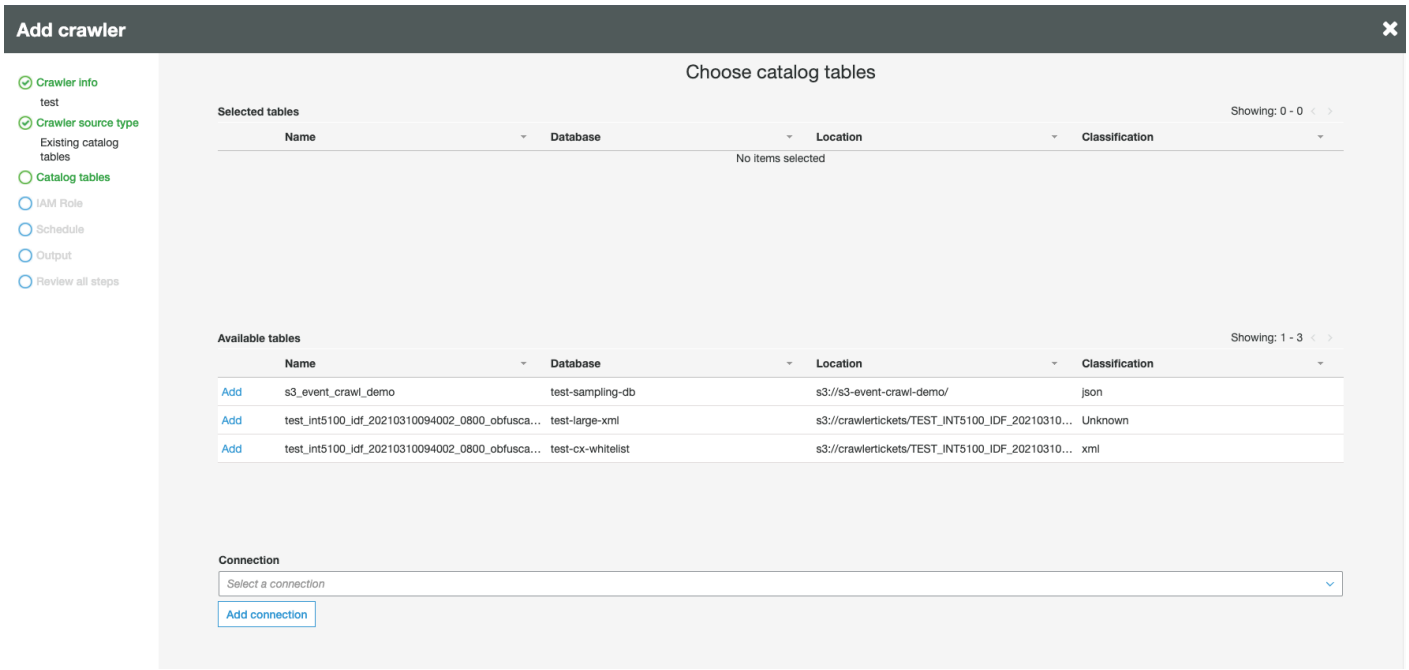


9. Completa la configurazione rimanente.

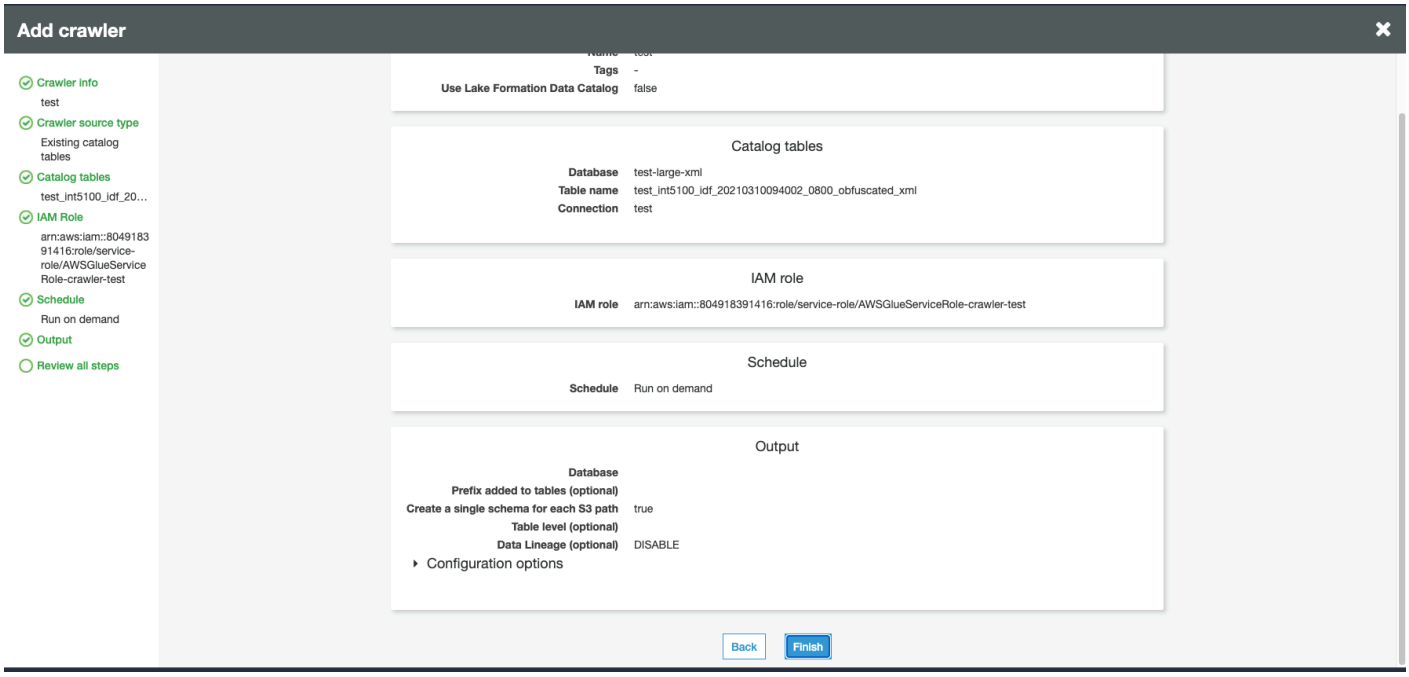
Creazione di un crawler per le tabelle del Catalogo dati supportate da Amazon S3

Ora è possibile creare un crawler che specifichi la connessione di `Network` creata e il tipo di fonte del Catalogo. Per ulteriori dettagli sulla creazione di un crawler, consulta the section called “Uso di crawler nella console”.

1. Inizia scegliendo Crawlers (Crawler) nel pannello di navigazione nella console AWS Glue.
2. Scegli Add crawler (Aggiungi crawler).
3. Specifica il nome del crawler, quindi scegli Next (Avanti).
4. Quando viene richiesto il tipo di origine crawler, scegliere Existing catalog tables (Tabelle di catalogo esistenti) e specificare le tabelle di catalogo esistenti da eseguire per il crawling dall'elenco delle tabelle disponibili.

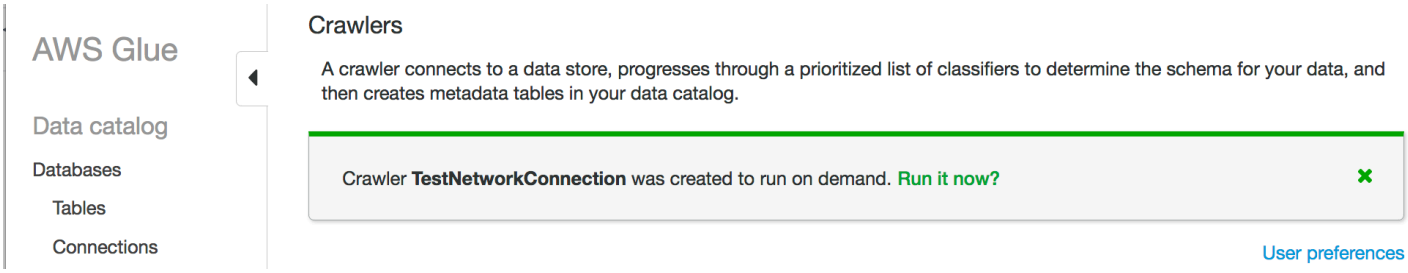


5. Scegli il ruolo IAM. Il ruolo IAM deve consentire l'accesso al servizio AWS Glue e al bucket Amazon S3. Per ulteriori informazioni, consulta the section called “Uso di crawler nella console”.
6. Definisci la pianificazione per il crawler.
7. Scegli un database esistente nel catalogo dati oppure crea una nuova voce del database.
8. Completa la configurazione rimanente e rivedi i passaggi.



Esecuzione di un crawler

Esegui il crawler.



Risoluzione dei problemi

Per la risoluzione dei problemi relativi ai bucket Amazon S3 che utilizzano un gateway VPC, consulta l'argomento relativo alle difficoltà di connessione a un bucket S3 usando un endpoint VPC gateway.

Risoluzione dei problemi di connessione in AWS Glue

Quando un crawler o un processo AWS Glue usa le proprietà di connessione per accedere a un datastore, potrebbero verificarsi errori durante un tentativo di connessione. AWS Glue usa indirizzi IP privati nella sottorete quando crea interfacce di rete elastiche nel cloud privato virtuale (VPC, Virtual Private Cloud) e nella sottorete specificati. I gruppi di sicurezza specificati nella connessione vengono

applicati a ciascuna delle interfacce di rete elastiche. Verifica se i gruppi di sicurezza consentono l'accesso in uscita e la connettività al cluster database.

Inoltre, Apache Spark richiede la connettività bidirezionale tra nodi driver ed executor. Uno dei gruppi di sicurezza deve consentire le regole in ingresso su tutte le porte TCP. Puoi evitare che vengano aperte a tutti limitando l'origine del gruppo di sicurezza a sé stesso con un gruppo di sicurezza autoreferenziata.

Di seguito sono elencate alcune operazioni tipiche che puoi eseguire per risolvere i problemi di connessione:

- Controlla l'indirizzo di porta della tua connessione.
- Controlla la stringa del nome utente e della password nella connessione o nel segreto.
- Per un archivio dati JDBC, verifica che consenta le connessioni in entrata.
- Verifica che il tuo archivio dati sia accessibile all'interno di VPC.
- Se memorizzi le credenziali di connessione utilizzando AWS Secrets Manager, verifica che il ruolo IAM per AWS Glue disponga dell'autorizzazione necessaria per accedere al tuo segreto. Per ulteriori informazioni, consulta la pagina Esempio: Autorizzazione per recuperare valori segreti nella Guida per l'utente di AWS Secrets Manager. In base alla configurazione della rete, potrebbe essere necessario anche creare un endpoint VPC per stabilire una connessione privata tra il VPC e Secrets Manager. Per ulteriori informazioni, consulta Utilizzo di un endpoint VPC AWS Secrets Manager.

Tutorial: utilizzo del connettore AWS Glue per Elasticsearch

Elasticsearch è un diffuso motore di ricerca e analisi dei dati open source per casi d'uso come analisi dei dati dei log, monitoraggio delle applicazioni in tempo reale e analisi dei dati di clickstream. Puoi utilizzare OpenSearch come archivio dati per i processi di estrazione, trasformazione e caricamento (ETL) configurando il connettore AWS Glue per Elasticsearch in AWS Glue Studio. Questo connettore è disponibile gratuitamente da Marketplace AWS.

> **Note**
>
> Marketplace AWS Elasticsearch Spark Connector è diventato obsoleto. Utilizza invece un Connettore AWS Glue per Elasticsearch.

In questo tutorial, mostreremo come connettersi ai nodi del servizio Amazon OpenSearch con un numero minimo di passaggi.

Argomenti

Prerequisiti

Per utilizzare questo tutorial, è necessario disporre di quanto segue:

- Accesso a AWS Glue Studio.
- Accesso a un cluster OpenSearch in AWS Cloud
- (Facoltativo) Accesso a AWS Secrets Manager.

Fase 1: (facoltativo) creare un segreto AWS per le informazioni sul cluster OpenSearch

Per archiviare e utilizzare in modo sicuro le credenziali di connessione, salvale in AWS Secrets Manager. Il segreto creato verrà utilizzato più avanti nel tutorial dalla connessione. Le coppie chiave-valore delle credenziali verranno inserite nel connettore AWS Glue per Elasticsearch come normali opzioni di connessione.

Per ulteriori informazioni sulla creazione dei segreti, consulta Creazione e gestione di segreti conAWS Secrets Manager nella Guida per l'utente di AWS Secrets Manager.

Per creare un segreto AWS

1. Accedi alla console AWS Secrets Manager.
2. Nella pagina di introduzione del servizio o nella pagina dell'elenco Secrets (Segreti), scegli Store a new secret (Archivia un nuovo segreto).

3. Nella pagina Store a new secret (Archivia un nuovo segreto), scegli Other type of secret (Altro tipo di segreto). Questa opzione indica che devi fornire la struttura e i dettagli del tuo segreto.
4. Aggiungi coppia chiave e valore per il nome utente del cluster OpenSearch. Ad esempio:

   `es.net.http.auth.user`: *`nomeutente`*
5. Scegli + Add row (+ Aggiungi riga) e inserisci un'altra coppia chiave-valore per la password. Ad esempio:

   `es.net.http.auth.pass` *`password`*
6. Seleziona Next (Successivo).
7. Immetti il nome di un segreto. Ad esempio: my-es-secret. Facoltativamente, puoi inserire una descrizione.

   Registra il nome del segreto, che viene utilizzato più avanti in questo tutorial, quindi scegli Next (Successivo).
8. Scegli di nuovo Next (Successivo), quindi scegli Store (Archivia) per creare il segreto.

Approfondimenti

Fase 2: sottoscrizione al connettore

Fase 2: sottoscrizione al connettore

Il connettore AWS Glue per Elasticsearch è disponibile gratuitamente su Marketplace AWS.

Sottoscrizione del connettore AWS Glue per Elasticsearch su Marketplace AWS

1. Se hai ancora configurato l'account AWS per utilizzare License Manager, procedi come segue:

   a. Apri la console AWS License Manager all'indirizzo https://console.aws.amazon.com/license-manager.
   b. Scegli Create customer managed license (Crea una licenza gestita dal cliente).
   c. Nella finestra IAM permissions (one-time setup) (Autorizzazioni IAM [configurazione una tantum]), scegli I grant AWS License Manager the required permissions (Concedo le autorizzazioni AWS License Manager richieste), quindi scegli Grant permissions (Concedi le autorizzazioni).

      Se non vedi questa finestra, hai già configurato le autorizzazioni necessarie.

2. Accedi alla console AWS Glue Studio all'indirizzo https://console.aws.amazon.com/gluestudio/.
3. Nella console AWS Glue Studio, espandi l'icona del menu (☰), quindi scegli Connectors (Connettori) nel pannello di navigazione.
4. Nella pagina Connectors (Connectors), scegli Go to Marketplace AWS (Vai su Marketplace AWS).
5. In Marketplace AWS, nella sezione Cerca prodotti AWS Glue Studio, inserisci Connettore AWS Glue per Elasticsearch nel campo di ricerca, quindi premi Invio.
6. Seleziona il nome del connettore, Connettore AWS Glue per Elasticsearch.
7. Nella pagina prodotto del connettore, utilizza le schede per visualizzare le relative informazioni. Quando vuoi continuare, scegli Continue to Subscribe (Continua con la sottoscrizione).
8. Rivedi i termini di utilizzo del software. Fai clic su Accetta termini.
9. Al termine del processo di sottoscrizione, verrà visualizzata una notifica: "Grazie per esserti registrato a questo prodotto! Adesso puoi configurare il software". Sopra il banner ci sarà il pulsante Passa alla configurazione. Scegli Continue to Configuration (Passa alla configurazione).
10. Scegli l'opzione Fulfillment (Compimento) sulla pagina Configure this software (Configura questo software). Puoi scegliere tra AWS Glue 1.0/2.0 o AWS Glue 3.0. Quindi, scegli Continue to Launch (Continua con l'avvio).

Approfondimenti

Fase 3: attivazione del connettore in AWS Glue Studio e creazione di una connessione

Fase 3: attivazione del connettore in AWS Glue Studio e creazione di una connessione

Dopo aver selezionato Continue to Launch (Continua con l'avvio), viene visualizzata la pagina Launch this Software (Avvia software) in Marketplace AWS. Dopo aver utilizzato il collegamento per attivare il connettore in AWS Glue Studio, si crea una connessione.

Come implementare il connettore e creare una connessione in AWS Glue Studio

1. Nella pagina Launch this Software (Avvia software) nella console Marketplace AWS, scegli Usage Instructions (Istruzioni di utilizzo), quindi scegli il collegamento nella finestra visualizzata.

Il browser è reindirizzato alla pagina della console AWS Glue StudioCreate marketplace connection (Crea connessione al marketplace).

2. Inserisci un nome per la connessione. Ad esempio: my-es-connection.
3. Nella sezione Connection access (Accesso alla connessione), per Connection credential type (Tipo di credenziali di connessione), scegli User name and password (Nome utente e password).
4. Nel campo AWS secret (Segreto AWS), inserisci il nome del tuo segreto. Ad esempio: my-es-secret.
5. Nella sezione Network options (Opzioni di rete), inserisci le informazioni sul VPC per connetterti al cluster OpenSearch.
6. Scegli Create connection and activate connector (Crea una connessione e attiva il connettore).

Approfondimenti

Fase 4: configurazione di un ruolo IAM per il processo ETL

Fase 4: configurazione di un ruolo IAM per il processo ETL

Quando si crea il processo ETL AWS Glue, si specifica un ruolo AWS Identity and Access Management (IAM) per il processo da utilizzare. Il ruolo deve concedere l'accesso a tutte le risorse utilizzate dal processo, inclusi Amazon S3 (per qualsiasi origine, destinazione, script, file driver e directory temporanea) e gli oggetti di AWS Glue Data Catalog.

Il ruolo IAM assunto per il processo ETL AWS Glue deve anche avere accesso al segreto che è stato creato nella sezione precedente. Per impostazione predefinita, il ruolo gestito da AWS `AWSGlueServiceRole` non ha accesso al segreto. Per impostare il controllo dell'accesso per i tuoi segreti, consulta Autenticazione e controllo degli accessi perAWS Secrets Manager e Limitazione dell'accesso a segreti specifici.

Per configurare un ruolo IAM per il processo ETL

1. Configura le autorizzazioni descritte in the section called “Esaminare le autorizzazioni IAM necessarie per i processi ETL”.
2. Configura le autorizzazioni aggiuntive necessarie quando utilizzi connettori con AWS Glue Studio, come descritto in the section called “Autorizzazioni richieste per l'utilizzo dei connettori”.

Approfondimenti

Fase 5: creazione di un processo che utilizza la connessione OpenSearch

Fase 5: creazione di un processo che utilizza la connessione OpenSearch

Dopo aver creato un ruolo per il tuo processo ETL, puoi creare un processo in AWS Glue Studio che utilizzi la connessione e il connettore per Open Spark ElasticSearch.

Se il processo viene eseguito all'interno di un Amazon Virtual Private Cloud (Amazon VPC), verifica che questo sia configurato correttamente. Per ulteriori informazioni, consulta the section called “Configurazione di un VPC per il tuo processo ETL”.

Per creare un processo che utilizza il connettore Spark Elasticsearch

1. In AWS Glue Studio, scegli Connectors (Connettori).
2. Nell'elenco Your connections (Le tue connessioni), seleziona la connessione appena creata e scegli Create job (Crea processo).
3. Nell'editor visivo dei processi, scegli il nodo di origine dati. A destra, nella scheda Data source properties - Connector (Proprietà origine dati - Connettore), configura ulteriori informazioni per il connettore.

    a. Scegli Add Schema (Aggiungi schema) e inserisci lo schema del set di dati nell'origine dati. Le connessioni non utilizzano tabelle archiviate in Data Catalog, il che significa che AWS Glue Studio non conosce lo schema dei dati. Devi fornire queste informazioni sullo schema manualmente. Per istruzioni su come utilizzare l'editor dello schema, consulta the section called “Modifica dello schema in un nodo di trasformazione personalizzato”.

    b. Espandi Connection options (Opzioni di connessione).

    c. Scegli (Aggiungi nuova opzione) e inserisci le informazioni necessarie per il connettore non inserite nella casella del segreto AWS:

    - es.nodes: https://<endpoint dominio OpenSearch>
    - es.port: 443
    - path: test
    - es.nodes.wan.only: true

Per una spiegazione di queste opzioni di connessione, fai riferimento a: https://www.elastic.co/guide/en/elasticsearch/hadoop/current/configuration.html.

4. Aggiungi un nodo di destinazione al grafico.

   La destinazione dati può essere Amazon S3 oppure le informazioni provenienti da un AWS Glue Data Catalog o un connettore possono essere usate per scrivere dati in una posizione diversa. Ad esempio, è possibile utilizzare una tabella del catalogo dati per scrivere in un database in Amazon RDS oppure utilizzare un connettore come destinazione dati per scrivere in archivi dati non supportati in modo nativo in AWS Glue.

   Se si sceglie un connettore per la destinazione dati, è necessario scegliere una connessione creata per tale connettore. Inoltre, se richiesto dal provider del connettore, è necessario aggiungere opzioni per fornire ulteriori informazioni al connettore. Se si utilizza una connessione che contiene informazioni per un segreto AWS, non è necessario fornire l'autenticazione con nome utente e password nelle opzioni di connessione.
5. Facoltativamente, aggiungi ulteriori origini dati e uno o più nodi di trasformazione come descritto in the section called “Modifica dei nodi di trasformazione dei dati gestiti da AWS Glue”.
6. Configura le proprietà del processo come descritto in the section called “Modificare le proprietà del processo”, iniziando dalla fase 3, e salva il lavoro.

Approfondimenti

Fase 6: esecuzione del processo

Fase 6: esecuzione del processo

Dopo aver salvato il processo, puoi eseguire il processo per eseguire le operazioni ETL.

Esecuzione del processo creato per il connettore AWS Glue per Elasticsearch

1. Utilizzando la console AWS Glue Studio, nella pagina dell'editor visivo, scegli Run (Esegui).
2. Nel banner che indica l'esito positivo, scegli Run Details (Dettagli esecuzione), oppure puoi scegliere la scheda Runs (Esecuzioni) dell'editor visivo per visualizzare le informazioni sull'esecuzione del processo.

Creazione di processi AWS Glue con sessioni interattive

Grazie alle sessioni interattive in AWS Glue, i data engineer possono creare processi AWS Glue con una velocità e una semplicità mai viste prima.

Argomenti

Panoramica delle sessioni interattive in AWS Glue

Con le sessioni interattive di AWS Glue, puoi creare, testare ed eseguire rapidamente applicazioni di preparazione e analisi dei dati. Le sessioni interattive forniscono un'interfaccia programmatica e visiva per la creazione e il test di script di estrazione, trasformazione e caricamento (ETL) per la preparazione dei dati. Le sessioni interattive eseguono applicazioni di analisi dei dati Apache Spark e forniscono accesso on demand a un ambiente di runtime Spark remoto. AWS Glue gestisce in modo trasparente Spark serverless per queste sessioni interattive.

Le sessioni interattive sono flessibili, pertanto ti permettono di creare e testare le applicazioni dall'ambiente che preferisci. Puoi creare e utilizzare le sessioni interattive tramite la AWS Command Line Interface e l'API. Puoi utilizzare i notebook compatibili con Jupyter per creare e testare visivamente gli script. Le sessioni interattive forniscono un kernel Jupyter open source che si integra quasi ovunque lo faccia Jupyter, inclusa l'integrazione con IDE come PyCharm, IntelliJ e VS Code. Ciò consente di creare codice nell'ambiente locale ed eseguirlo senza problemi sul backend delle sessioni interattive.

Utilizzando l'API delle sessioni interattive, i clienti possono eseguire in modo programmatico applicazioni che utilizzano l'analisi dei dati di Apache Spark senza dover gestire l'infrastruttura Spark. È possibile eseguire una o più istruzioni Spark in una singola sessione interattiva.

Le sessioni interattive forniscono quindi un modo più rapido, economico e flessibile per creare ed eseguire applicazioni di preparazione e analisi dei dati. Per informazioni sull'utilizzo di sessioni interattive, consulta la documentazione in questa sezione. Magic supportati da AWS Glue

Restrizioni

Le sessioni interattive e i notebook non sono attualmente disponibili nelle regioni Medio Oriente (Emirati Arabi Uniti) (`me-central-1`), Europa (Spagna) (`eu-south-2`) o Europa (Zurigo) (`eu-central-2`), ma potrebbero essere disponibili in seguito.

I segnalibri del processo non sono supportati nelle sessioni interattive.

Nozioni di base sulle sessioni interattive AWS Glue

Queste sezioni descrivono come eseguire le sessioni interattive AWS Glue localmente.

Prerequisiti per impostare le sessioni interattive a livello locale

Di seguito sono indicati i prerequisiti per l'installazione delle sessioni interattive:

- Sono supportate le versioni di Python dalla 3.6 alla 3.10 e successive.
- Vedere le sezioni riportate di seguito per le istruzioni per MacOS/Linux e Windows.

Installazione di Jupyter e delle sessioni interattive Jupyter Kernel di AWS Glue

Utilizza quanto segue per installare il kernel localmente.

Il comando `install-glue-kernels` installa il kernelspec jupyter sia per i kernel pyspark sia per quelli spark e installa anche i loghi nella directory corretta.

```
pip3 install --upgrade jupyter boto3 aws-glue-sessions
```

```
install-glue-kernels
```

Esecuzione di Jupyter

Completa i seguenti passaggi per eseguire Jupyter Notebook.

1. Per avviare il notebook Jupyter utilizzare il seguente comando.

```
jupyter notebook
```

2. Scegliere New (Nuovo), quindi scegliere uno dei kernel AWS Glue per iniziare a creare codice rispetto a AWS Glue.

Configurazione delle credenziali di sessione e della regione

Istruzioni per MacOS/Linux

Le sessioni interattive di AWS Glue richiedono le stesse autorizzazioni IAM di Processi e Dev Endpoint di AWS Glue. Specificare il ruolo utilizzato con le sessioni interattive in uno dei due modi seguenti:

1. Con `%iam_role` e `%region` magic
2. Con una linea aggiuntiva in `~/.aws/config`

Configurazione di un ruolo di sessione con magic

Nella prima cella digita `%iam_role <YourGlueServiceRole>` nella prima cella eseguita.

Configurazione di un ruolo di sessione con **`~/.aws/config`**

Il ruolo di servizio AWS Glue per le sessioni interattive può essere specificato nel notebook stesso o memorizzato insieme alla configurazione AWS CLI. Se hai un ruolo che normalmente usi con processi AWS Glue, questo sarà questo ruolo. Se non disponi ancora di un ruolo per i processi AWS Glue, segui la guida Configuring IAM permissions for AWS Glue per configurarne uno.

Per impostare questo ruolo come ruolo predefinito per le sessioni interattive:

1. Con un editor di testo, apri `~/.aws/config`.
2. Cerca il profilo che usi per AWS Glue. Se non usi un profilo, usa il profilo `[Default]`.

3. Aggiungi una riga nel profilo per il ruolo che intendi usare come `glue_role_arn=<AWSGlueServiceRole>`.
4. [Facoltativo]: se sul tuo profilo non è impostata una regione predefinita, è consigliabile aggiungerne una con `region=us-east-1`, sostituendo `us-east-1` con la regione desiderata.
5. Salvare la configurazione.

Per ulteriori informazioni, consulta Sessioni Interattive con IAM.

Istruzioni per Windows

Le sessioni interattive di AWS Glue richiedono le stesse autorizzazioni IAM di Processi e Dev Endpoint di AWS Glue. Specificare il ruolo utilizzato con le sessioni interattive in uno dei due modi seguenti:

1. Con `%iam_role` e `%region` magic
2. Con una linea aggiuntiva in `~/.aws/config`

Configurazione di un ruolo di sessione con magic

Nella prima cella digita `%iam_role <YourGlueServiceRole>` nella prima cella eseguita.

Configurazione di un ruolo di sessione con **`~/.aws/config`**

Il ruolo di servizio AWS Glue per le sessioni interattive può essere specificato nel notebook stesso o memorizzato insieme alla configurazione AWS CLI. Se hai un ruolo che normalmente usi con processi AWS Glue, questo sarà questo ruolo. Se non disponi ancora di un ruolo per processi AWS Glue, segui questa guida,Impostazione delle autorizzazioni IAM per AWS Glue, per configurarne uno.

Per impostare questo ruolo come ruolo predefinito per le sessioni interattive:

1. Con un editor di testo, apri `~/.aws/config`.
2. Cerca il profilo che usi per AWS Glue. Se non usi un profilo, usa il profilo `[Default]`.
3. Aggiungi una riga nel profilo per il ruolo che intendi usare come `glue_role_arn=<AWSGlueServiceRole>`.
4. [Facoltativo]: se sul tuo profilo non è impostata una regione predefinita, è consigliabile aggiungerne una con `region=us-east-1`, sostituendo `us-east-1` con la regione desiderata.
5. Salvare la configurazione.

Per ulteriori informazioni, consulta Sessioni Interattive con IAM.

Aggiornamento dall'anteprima delle sessioni interattive

Il kernel è stato aggiornato con nuovi nomi quando è stato rilasciato con la versione 0.27. Per pulire le versioni di anteprima dei kernel, esegui quanto segue da un terminale o PowerShell.

> **Note**
>
> Se fai parte di qualsiasi altra anteprima di AWS Glue che richiede un modello di servizio personalizzato, rimuovendo il kernel sarà rimosso anche questo modello.

```
# Remove Old Glue Kernels
jupyter kernelspec remove glue_python_kernel
jupyter kernelspec remove glue_scala_kernel

# Remove Custom Model
cd ~/.aws/models
rm -rf glue/
```

Utilizzo di sessioni interattive con SageMaker Studio

AWS Glue Interactive Sessions è un ambiente di runtime Apache Spark on-demand e senza server che data scientist e ingegneri possono utilizzare per creare, testare ed eseguire rapidamente applicazioni di preparazione e analisi dei dati. È possibile avviare una sessione AWS Glue interattiva avviando un notebook Studio Classic. Amazon SageMaker

Per ulteriori informazioni, consulta Preparare i dati utilizzando sessioni AWS Glue interattive.

Utilizzo di sessioni interattive con codice Microsoft Visual Studio

Prerequisiti

- Installa sessioni interattive AWS Glue e verifica che funzioni con notebook Jupyter.
- Scarica e installa Visual Studio Code con Jupyter. Per informazioni dettagliate, consulta Notebook Jupyter in VS Code

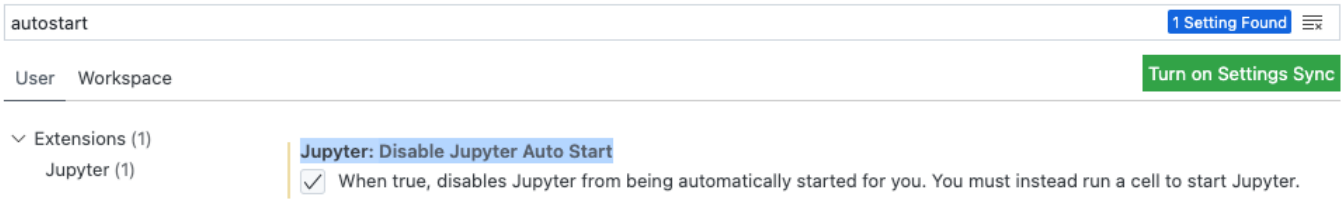
Nozioni di base sulle sessioni interattive con VSCode

1. Disattiva Jupyter AutoStart in VS Code.

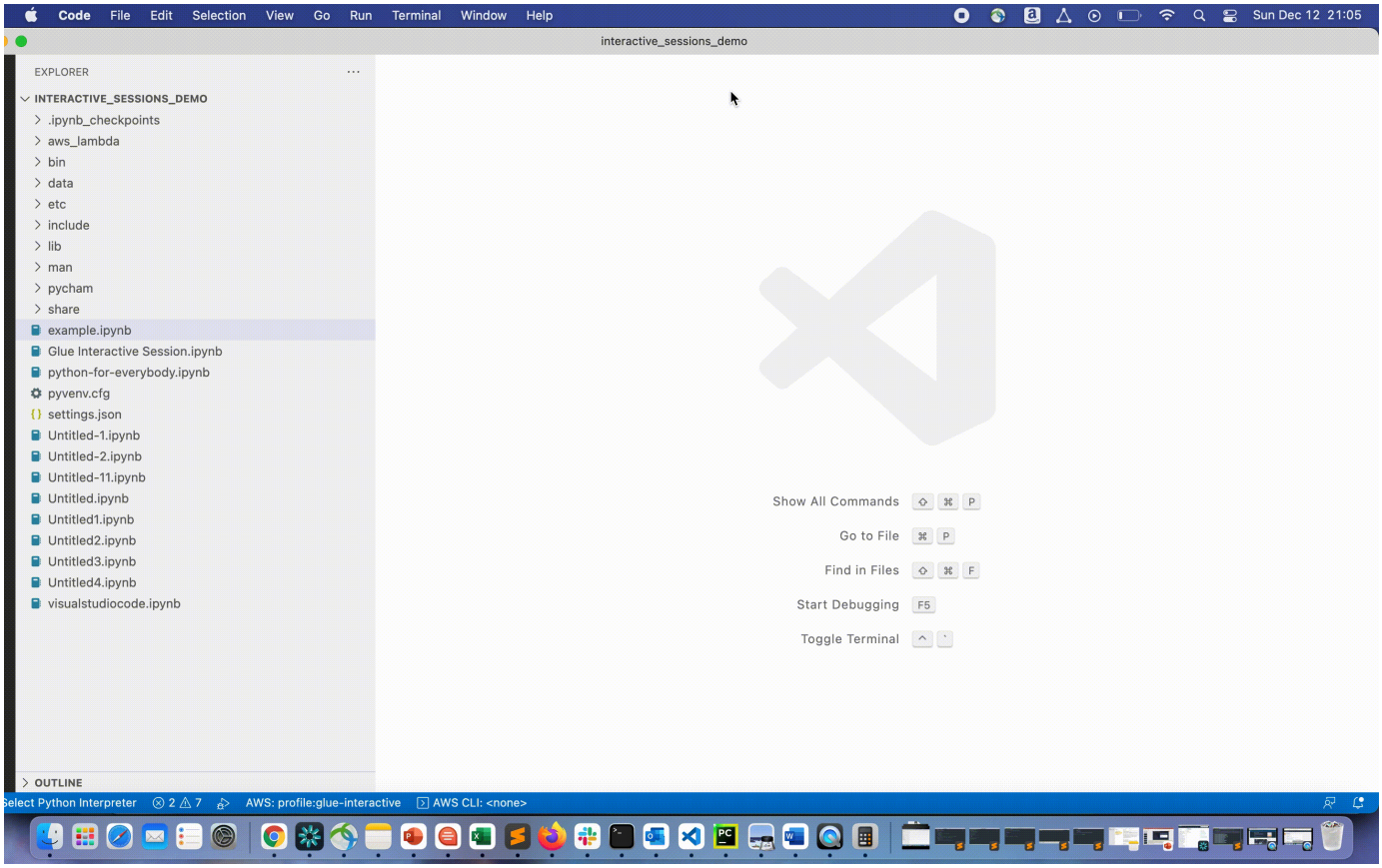
   In Visual Studio Code, i kernel di Jupyter vengono avviati automaticamente. Questo impedisce ai magic di entrare in vigore poiché la sessione è già stata avviata. Per disabilitare Auto Start su Windows, apri File > Preferences > Extensions > Jupyter, fai clic con il pulsante destro del mouse su Jupyter, quindi scegli Extension Settings.

   Su macOS, apri Code > Settings > Extensions > Jupyter, fai clic con il pulsante destro del mouse su Jupyter, quindi scegli Extension Settings.

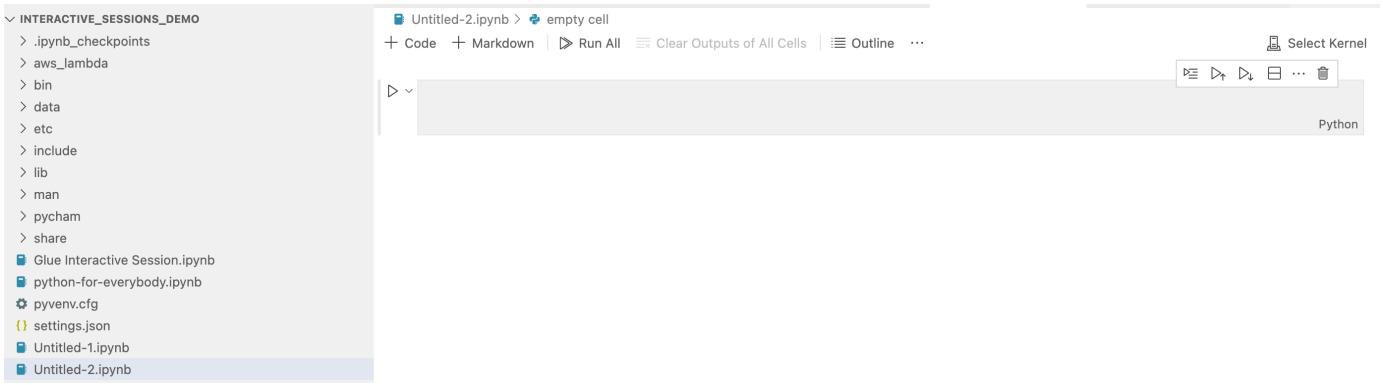
   Scorri verso il basso fino a visualizzare Jupyter: Disable Jupyter Auto Start. Seleziona la casella "When true, disables Jupyter from being automatically started for you. È necessario invece eseguire una cella per avviare Jupyter."



2. Vai su File (File) > New File (Nuovo file) > Save (Salva) per salvare questo file con il nome di tua scelta come estensione `.ipynb` o seleziona jupyter sotto Select a language (Seleziona una lingua) e salva il file.



3. Fare doppio clic sul file. Viene visualizzata la shell di Jupyter e verrà aperto un notebook.



4. Su Windows, quando crei un file per la prima volta, per impostazione predefinita, non è selezionato alcun kernel. Clicca su Select Kernel (Seleziona kernel) per visualizzare un elenco di kernel disponibili. Scegli Glue PySpark.

   Su macOS, se non vedi il PySpark kernel Glue, prova i seguenti passaggi:

   1. Esegui una sessione locale di Jupyter per ottenere l'URL.

      Ad esempio, per avviare il notebook Jupyter, utilizza il seguente comando.

```
jupyter notebook
```

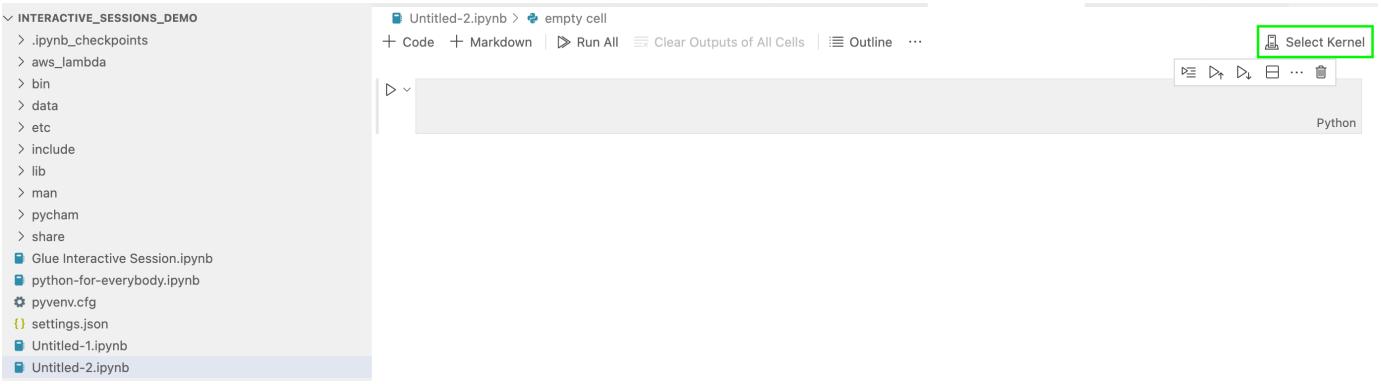
Quando il notebook viene eseguito per la prima volta, verrà visualizzato un URL simile a `http://localhost:8888/?token=3398XXXXXXXXXXXXXXX`.

Copiare l'URL.

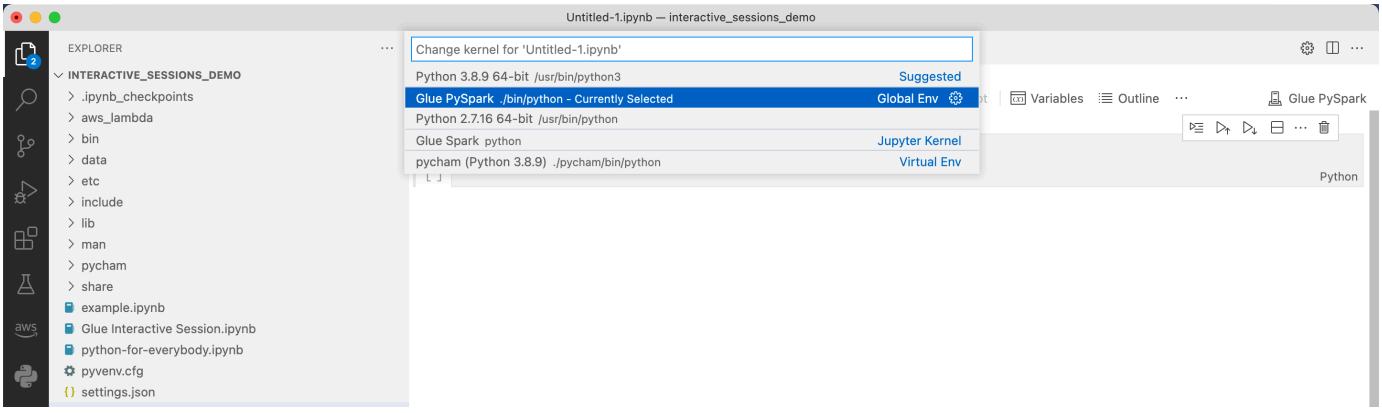
2. In VS Code, fai clic sul kernel corrente, quindi seleziona Select Another Kernel... e successivamente Existing Jupyter Server.... Incolla l'URL che hai copiato dal passaggio precedente.

   Se ricevi un messaggio di errore, consulta il wiki VS Code Jupyter.

3. In caso di successo, questo imposterà il kernel su PySparkGlue.



Scegli il kernel Glue PySpark o Glue Spark (rispettivamente per Python e Scala).



Se non vedi AWS Glue PySparki kernel AWS GlueSpark nell'elenco a discesa, assicurati di aver installato il AWS Glue kernel nel passaggio precedente o che

l'`python.defaultInterpreterPath`impostazione in Visual Studio Code sia corretta. Per ulteriori informazioni, vedi python. defaultInterpreterPath descrizione dell'impostazione.

5. Crea una sessione interattiva AWS Glue. Procedi alla creazione di una sessione nello stesso modo in cui è stato fatto nel notebook Jupyter. Specifica qualsiasi magic nella parte superiore della prima cella ed esegui un'istruzione di codice.

Configurazione delle sessioni interattive di AWS Glue per Jupyter e notebook AWS Glue Studio

Introduzione ai magic di Jupyter

I magic di Jupyter sono comandi che possono essere eseguiti all'inizio di una cella o come un intero corpo di una cella. I magic di linea iniziano per `%`, i magic di cella per `%%`. I magic di linea come `%region` e `%connections` possono essere eseguiti con più magic in una cella o con codice incluso nel corpo della cella come nell'esempio seguente.

```
%region us-east-2
%connections my_rds_connection
dy_f = glue_context.create_dynamic_frame.from_catalog(database='rds_tables',
 table_name='sales_table')
```

I magic di cella devono utilizzare l'intera cella e possono avere il comando esteso su più righe. Un esempio di `%%sql` è riportato di seguito.

```
%%sql
select * from rds_tables.sales_table
```

Magic supportati dalle sessioni interattive di AWS Glue per Jupyter

Di seguito sono elencati i magic che puoi utilizzare con le sessioni interattive AWS Glue per Jupyter Notebook.

Sessioni Magic

Nome	Type	Descrizione
`%help`	N/A	Restituisce un elenco di descrizioni e tipi di input per tutti i comandi magic.
`%profile`	Stringa	Specifica un profilo nella tua configura zione AWS da utilizzare come provider di credenziali.
`%region`	Stringa	Specificare l'Regione AWS in cui inizializ zare la sessione. Impostazione predefinita da `~/.aws/configure`. Esempio: `%region us-west-1`
`%idle_timeout`	Int	Il numero di minuti di inattività dopo i quali una sessione andrà in timeout in seguito all'esecuzione di una cella. Il valore predefinito del timeout di inattività per le sessioni Spark ETL è il timeout predefini to, pari a 2.880 minuti (48 ore). Per altri tipi di sessione, consulta la documentazione relativa al tipo di sessione specifico. Esempio: `%idle_timeout 3000`
`%session_id`	Stringa	Restituisce l'ID della sessione in esecuzion e. Se viene fornito un elemento Stringa, verrà impostato come ID di sessione per la successiva sessione in esecuzione. Quando esegui un notebook Jupyter in AWS Glue Studio, questo magic restituis ce un valore di sola lettura che non puoi modificare.
`%session_id_prefix`	Stringa	Definire una stringa che precederà tutti gli ID di sessione nel formato [session_ id_prefix]-[session_id]. Se non viene fornito

Nome	Type	Descrizione
		un ID di sessione, verrà generato un UUID casuale. Questo magic non è supportat o quando esegui un notebook Jupyter in AWS Glue Studio. Esempio: `%session_id_prefix 001`
`%status`		Restituisce lo stato dell'attuale sessione AWS Glue inclusi la durata, la configura zione e l'utente/ruolo che la esegue.
`%stop_session`		Arresta la sessione corrente.
`%list_sessions`		Elenca tutte le sessioni attualmente in esecuzione per nome e ID.
`%session_type`	Stringa	Imposta il tipo di sessione su Flussi di dati, ETL o Ray. Esempio: `%session_type Streaming`
`%glue_version`	Stringa	La versione di AWS Glue che questa sessione dovrà utilizzare. Esempio: `%glue_version 3.0`

Magic per la selezione dei tipi di processo

Nome	Type	Descrizione
`%streaming`	Stringa	Cambia il tipo di sessione in Streaming AWS Glue.
`%etl`	Stringa	Cambia il tipo di sessione in ETL AWS Glue.

Nome	Type	Descrizione
%glue_ray	Stringa	Cambia il tipo di sessione in AWS Glue per Ray. Consulta la pagina Magics supported by AWS Glue Ray interactive sessions.

Magic per la configurazione di AWS Glue per Spark

Il magic `%%configure` è un dizionario formattato json composto da tutti i parametri di configurazione per una sessione. Ciascun parametro può essere specificato qui o tramite magic individuali.

Nome	Type	Descrizione
`%%configure`	Dizionario	Specifica un dizionario formattato JSON composto da tutti i parametri di configurazione per una sessione. Ciascun parametro può essere specificato qui o tramite magic individuali. Per un elenco di parametri ed esempi di utilizzo di `%%configure` , consulta la tabella seguente: Using %%configure.
`%iam_role`	Stringa	Specifica un ruolo IAM ARN con cui eseguire la sessione. Predefinito da ~/.aws/configure. Esempio: `%iam_role AWSGlueServiceRole`
`%number_of_workers`	Int	Il numero di dipendenti di un specifico worker-type allocati quando viene eseguito un processo. Deve essere impostato anche `worker_type` . Il `number_of_workers` predefinito è 5. Esempio: `%number_of_workers 2`

Nome	Type	Descrizione
`%additional_python_modules`	Elenco	Elenco separato da virgole di moduli Python aggiuntivi da includere nel cluster (possono provenire da PyPI o S3). Esempio: `%additional_python_modules pandas, numpy` .

Nome	Type	Descrizione

`%%tags`	Stringa	Aggiunge tag a una sessione. Specifica i tag tra parentesi graffe { }. Ogni coppia di nomi di tag è racchiusa tra parentesi (" ") e separata da una virgola (,).
		
		`%%tags`
		`{"billing":"Data-Platform",`
		`"team":"analytics"}`
		
		Utilizza il magic `%status` per visualizzare i tag associati alla sessione.
		
		`%status`
		
		`Session ID: <sessionId>`
		`Status: READY`
		`Role: <example-role>`
		`CreatedOn: 2023-05-26 11:12:17.`
		`056000-07:00`
		`GlueVersion: 3.0`
		`Job Type: glueetl`
		`Tags: {'owner':'example-owner',`
		`'team':'analytics', 'billing'`
		`:'Data-Platform'}`
		`Worker Type: G.4X`
		`Number of Workers: 5`
		`Region: us-west-2`
		`Applying the following default`
		`arguments:`
		`--glue_kernel_version 0.38.0`
		`--enable-glue-datacatalog true`
		`Arguments Passed: ['--glue_`
		`kernel_version: 0.38.0', '--`
		`enable-glue-datacatalog: true']`

Nome	Type	Descrizione

`%%assume_role`	Dizionario	Specifica un dizionario in formato json o una stringa ARN del ruolo IAM per creare una sessione per l'accesso multi-account.
		
		Esempio con ARN:
		
		`%%assume_role`
		`{`
		`  'arn:aws:iam::XXXXXXXXXXXX:`
		`role/AWSGlueServiceRole'`
		`}`
		
		Esempio con credenziali:
		
		` %%assume_role`
		`{{`
		`    "aws_access_key_id" =`
		` "XXXXXXXXXXXX",`
		`    "aws_secret_access_key" =`
		` "XXXXXXXXXXXX",`
		`    "aws_session_token" =`
		` "XXXXXXXXXXXX"`
		`}}`

argomenti del magic di cella %%configure

Il magic `%%configure` è un dizionario formattato json composto da tutti i parametri di configurazione per una sessione. Ciascun parametro può essere specificato qui o tramite magic individuali. Di seguito sono riportati alcuni esempi di argomenti supportati dal magic di cella `%%configure`. Utilizza il `--` prefisso per gli argomenti di esecuzione specificati per il lavoro. Esempio:

```
%%configure
{
   "--user-jars-first": "true",
   "--enable-glue-datacatalog": "false"
}
```

Per ulteriori informazioni sui parametri del processo, vedereParametri del processo.

Configurazione della sessione

Parametro	Tipo	Descrizione

`max_retries`	Int	Il numero massimo di tentativi per riprovare il processo se ha esito negativo.
		`%%configure`
		`{`
		`  "max_retries": "0"`
		`}`

`max_concurrent_runs`	Int	Il numero massimo di esecuzioni simultane e consentite per un processo.
		Esempio:
		`%%configure`
		`{`
		`  "max_concurrent_runs": "3"`
		`}`

Parametri della sessione

Parametro	Tipo	Descrizione

`--enable-spark-ui`	Booleano	Abilita l'interfaccia utente di Spark per monitorare ed eseguire il debug dei processi ETL AWS Glue.
		`%%configure`
		`{`
		`  "--enable-spark-ui": "true"`

Parametro	Tipo	Descrizione
		`}`

`--spark-event-logs-path`	Stringa	Specifica un percorso Amazon S3. Quando si utilizza la funzionalità di monitoraggio dell'interfaccia utente Spark.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--spark-event-logs-path":`
		` "s3://path/to/event/logs/"`
		`}`

`--scriptLocation`	Stringa	Specifica il percorso S3 per uno script che esegue un processo.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--scriptLocation": "s3://new-folder-here"`
		`}`

Parametro	Tipo	Descrizione

`--SECURITY_CONFIGURATION`	Stringa	Il nome di una configurazione AWS Glue di sicurezza
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--SECURITY_CONFIGURATION": "security-configuration-name",`
		`}`

`--job-language`	Stringa	Il linguaggio di programmazione script. Accetta un valore di "scala" o "python". L'impostazione predefinita è "python".
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--job-language": "scala"`
		`}`

`--class`	Stringa	La classe Scala che funge da punto di accesso per lo script Scala. L'impostazione predefinita è null.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--class": "className"`
		`}`

Parametro	Tipo	Descrizione

`--user-jars-first`	Booleano	Assegna la priorità ai file JAR aggiuntivi del cliente nel classpath. L'impostazione predefinita è null.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--user-jars-first": "true"`
		`}`

`--use-postgres-driver`	Booleano	Assegna la priorità al driver JDBC Postgre nel classpath per evitare un conflitto con il driver JDBC di Amazon Redshift. L'imposta zione predefinita è null.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--use-postgres-driver": "true"`
		`}`

Parametro	Tipo	Descrizione

`--extra-files`	List(string)	I percorsi Amazon S3 dei file aggiuntivi, come file di configurazione, che AWS Glue copia nella directory di lavoro del tuo script prima di eseguirlo.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--extra-files": "s3://path/to/additional/files/"`
		`}`

`--job-bookmark-option`	Stringa	Controlla il comportamento di un segnalibro del processo. Accetta il valore 'job-bookmark-enable', 'job-bookmark-disable' o 'job-bookmark-pause'. L'impostazione predefinita è 'job-bookmark-disable'.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--job-bookmark-option": "job-bookmark-enable"`
		`}`

Parametro	Tipo	Descrizione

`--temp-dir`	Stringa	Specifica un percorso Amazon S3 a un bucket utilizzabile come directory temporanea per il processo. L'impostazione predefinita è null.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--temp-dir": "s3://path/to/temp/dir"`
		`}`

`--enable-s3-parquet-optimized-committer`	Booleano	Abilita il committer ottimizzato EMRFS Amazon S3 per la scrittura dei dati Parquet in Amazon S3. Il valore predefinito è "true".
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--enable-s3-parquet-optimized-committer": "false"`
		`}`

Parametro	Tipo	Descrizione

`--enable-rename-algorithm-v2`	Booleano	Imposta la versione dell'algoritmo di ridenominazione EMRFS alla versione 2. Il valore predefinito è "true".
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--enable-rename-algorithm-v2": "true"`
		`}`

`--enable-glue-datacatalog`	Booleano	Consente di utilizzare Catalogo dati AWS Glue come metastore Apache Spark Hive.
		
		Esempio:
		
		`%%configure`
		`{`
		`  --"enable-glue-datacatalog": "true"`
		`}`

`--enable-metrics`	Booleano	Abilita la raccolta di parametri per la profilatura del processo per l'esecuzione. Il valore predefinito è "false".
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--enable-metrics": "true"`
		`}`

Parametro	Tipo	Descrizione

`--enable-continuous-cloudwatch-log`	Booleano	Abilita la registrazione continua in tempo reale per i processi AWS Glue. Il valore predefinito è "false".
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--enable-continuous-cloudwatch-log": "true"`
		`}`

`--enable-continuous-log-filter`	Booleano	Specifica un filtro standard o nessun filtro durante la creazione o la modifica di un processo abilitato per la registrazione continua. Il valore predefinito è "true".
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--enable-continuous-log-filter": "true"`
		`}`

Parametro	Tipo	Descrizione

`--continuous-log-stream-prefix`	Stringa	Specifica un prefisso del flusso di log Amazon CloudWatch personalizzato per un processo abilitato per la registrazione continua. L'impostazione predefinita è null.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--continuous-log-stream-prefix": "prefix"`
		`}`

`--continuous-log-conversionPattern`	Stringa	Specifica un modello di log di conversione personalizzato per un processo abilitato per la registrazione continua. L'impostazione predefinita è null.
		
		Esempio:
		
		`%%configure`
		`{`
		`  "--continuous-log-conversionPattern": "pattern"`
		`}`

Parametro	Tipo	Descrizione

`--conf`	Stringa	Controlla i parametri di configurazione di Spark. È per casi d'uso avanzati. Utilizzare `--conf` prima di ogni parametro. Esempio:
		`%%configure`
		`{`
		`"--conf": "spark.hadoop.hive`
		`.metastore.glue.catalogid=1`
		`23456789012 --conf hive.meta`
		`store.client.factory.class=`
		`com.amazonaws.glue.catalog.`
		`metastore.AWSGlueDataCatalo`
		`gHiveClientFactory --conf`
		` hive.metastore.schema.verif`
		`ication=false"`
		`}`

Magic per processi Spark (ETL e flussi di dati)

Nome	Type	Descrizione
`%worker_type`	Stringa	Standard, G.1X o G.2X. Anche `number_of_workers` deve essere impostato. Il worker_type predefinito è G.1X.

`%connections`	Elenco	Specifica un elenco separato da virgola di connessioni da utilizzare nella sessione.
		Esempio:
		`%connections my_rds_connection`
		`dy_f =`
		` glue_context.create_dynamic`
		`_frame.from_catalog(databas`

Nome	Type	Descrizione
		`e='rds_tables', table_nam e='sales_table')`
`%extra_py_files`	Elenco	Specifica un elenco separato da virgola di file Python aggiuntivi da Amazon S3.
`%extra_jars`	Elenco	Specifica un elenco separato da virgola di jar aggiuntivi da includere nel cluster.
`%spark_conf`	Stringa	Specifica le configurazioni Spark personalizzate per la sessione. Ad esempio, `%spark_conf spark.ser ializer=org.apache.spark.se rializer.KryoSerializer` .

Magic per processi Ray

Nome	Type	Descrizione
`%min_workers`	Int	Il numero minimo di worker allocati a un processo Ray. Default: 1. Esempio: `%min_workers 2`
`%object_memory_head`	Int	La percentuale di memoria libera sul nodo principale dell'istanza dopo un avvio a caldo. Minimo: 0 Massimo: 100 Esempio: `%object_memory_head 100`
%object_memory_worker	Int	La percentuale di memoria libera sui nodi worker dell'istanza dopo un avvio a caldo. Minimo: 0 Massimo: 100

Nome	Type	Descrizione
		Esempio: `%object_memory_worker 100`

Magic operativi

Nome	Type	Descrizione
`%%sql`	Stringa	Eseguire codice SQL. Tutte le righe dopo che il magic `%%sql` iniziale verrà passato come parte del codice SQL. Esempio: `%%sql select * from rds_tables.sales_table`

`%matplot`	Figura matplotlib	Visualizza i dati utilizzando la libreria matplotlib.
		
		Esempio:
		
		`import matplotlib.pyplot as plt`
		
		`# Set X-axis and Y-axis values`
		`x = [5, 2, 8, 4, 9]`
		`y = [10, 4, 8, 5, 2]`
		
		`# Create a bar chart`
		`plt.bar(x, y)`
		
		`# Show the plot`
		`%matplot plt`

`%plotly`	Figura plotly	Visualizza i dati utilizzando la libreria plotly. Esempio: `import plotly.express as px`

Nome	Type	Descrizione

		`#Create a graphical figure`
		`fig = px.line(x=["a","b","c"],`
		` y=[1,3,2], title="sample`
		` figure")`
		
		`#Show the figure`
		`%plotly fig`

Sessioni di denominazione

Le sessioni interattive AWS Glue sono risorse AWS e richiedono un nome. I nomi devono essere univoci per ogni sessione e possono essere limitati dagli amministratori IAM. Per ulteriori informazioni, consulta Sessioni Interattive con IAM. Il kernel Jupyter genera automaticamente nomi di sessione univoci per tuo conto. Tuttavia, le sessioni possono essere nominate manualmente in due modi:

1. Utilizzo del file di configurazione AWS Command Line Interface situato in `~.aws/config`. Consulta Configurazione di AWS Config con AWS Command Line Interface.
2. Utilizzo dei magic `%session_id_prefix`. Per informazioni, consulta Magic supportati dalle sessioni interattive di AWS Glue per Jupyter.

Il nome di una sessione viene generato come segue:

- Quando vengono forniti il prefisso e session_id: il nome della sessione sarà {prefix}-{UUID}.
- Quando non viene fornito nulla: il nome della sessione sarà {UUID}.

Il prefisso dei nomi delle sessioni consente di riconoscere una sessione quando viene elencata nella AWS CLI o nella console.

Specifica di un ruolo IAM per le sessioni interattive

È necessario specificare un ruolo Identity and Access Management (IAM) AWS da utilizzare con il Codice ETL AWS Glue eseguito con le sessioni interattive.

Il ruolo deve disporre delle stesse autorizzazioni IAM di quelle necessarie per l'esecuzione dei processi AWS Glue. Consulta Creazione di un ruolo IAM per AWS Glue per ulteriori informazioni sulla creazione di un ruolo per processi AWS Glue e sessioni interattive.

I ruoli IAM possono essere specificati in due modi:

- Utilizzo del file di configurazione AWS Command Line Interface situato in `~.aws/config` (Suggerito). Per ulteriori informazioni, consulta Configurazione di sessioni con ~/.aws/config .

  > **Note**
  >
  > Quando il magic `%profile` è in uso, la configurazione per `glue_iam_role` di quel profilo è mantenuta.

- Usando il magic %iam_role. Per ulteriori informazioni, consulta Magic supportati dalle sessioni interattive di AWS Glue per Jupyter.

Configurazione di sessioni con profili denominati

Le sessioni interattive AWS Glue utilizzano le stesse credenziali di AWS Command Line Interface o boto3. Le sessioni interattive onorano e lavorano con profili denominati come il AWS CLI che si trova in `~/.aws/config` (Linux e macOS) o `%USERPROFILE%\.aws\config` (Windows). Per ulteriori informazioni, consulta la sezione Using named profiles.

Le sessioni interattive sfruttano i vantaggi dei profili denominati consentendo al prefisso ruolo di servizio e ID sessione AWS Glue di essere specificato in un profilo. Per configurare un ruolo di profilo, aggiungere una riga per la chiave `iam_role` e/o `session_id_prefix` al tuo profilo denominato come mostrato di seguito. Il valore `session_id_prefix` non richiede virgolette. Ad esempio, se desideri aggiungere un `session_id_prefix`, inserisci il valore di `session_id_prefix=myprefix`.

```
[default]
region=us-east-1
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
glue_iam_role=arn:aws:iam::<AccountID>:role/<GlueServiceRole>
session_id_prefix=<prefix_for_session_names>

[user1]
```

```
region=eu-west-1
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
glue_iam_role=arn:aws:iam::<AccountID>:role/<GlueServiceRoleUser1>
session_id_prefix=<prefix_for_session_names_for_user1>
```

Se si dispone di un metodo personalizzato per generare credenziali, è anche possibile configurare il profilo in modo che utilizzi il parametro `credential_process` nel file `~/.aws/config`. Ad esempio:

```
[profile developer]
region=us-east-1
credential_process = "/Users/Dave/generate_my_credentials.sh" --username helen
```

Puoi trovare ulteriori dettagli sulle credenziali di approvvigionamento tramite il parametro `credential_process` qui: Credenziali di approvvigionamento con un processo esterno.

Se nel profilo che state usando non sono impostate una regione o un `iam_role`, dovete specificarli usando le `%region` e i `%iam_role` magic nella prima cella che eseguite.

Guida introduttiva alle AWS Glue sessioni interattive di For Ray (anteprima)

Warning

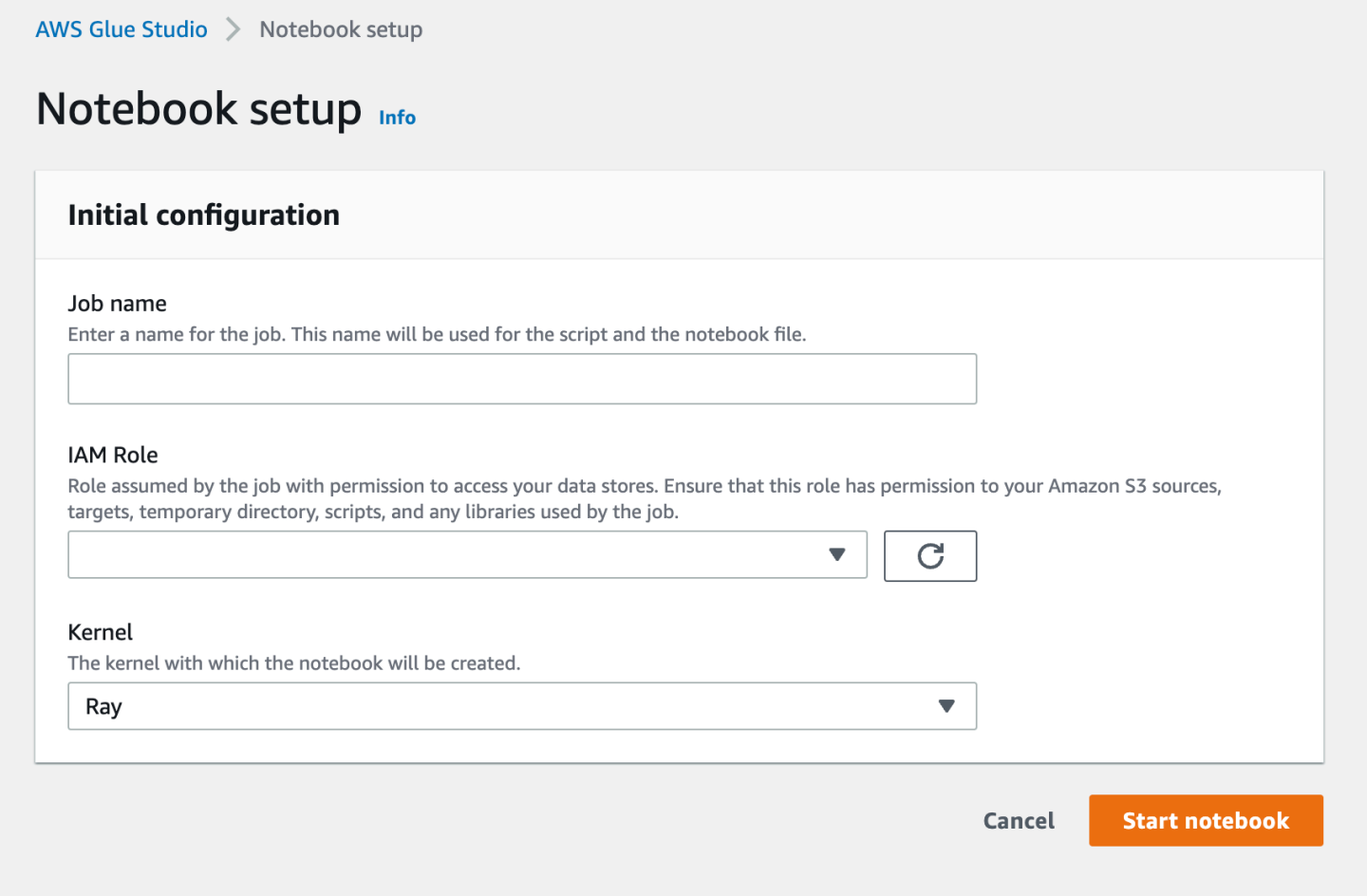
L'anteprima delle sessioni interattive di AWS Glue for Ray terminerà il 30 aprile 2024. Dopo questa data, non sarà più possibile creare nuove sessioni interattive su AWS Glue for Ray.

Note

AWS Glue for Ray è disponibile negli Stati Uniti orientali (Virginia settentrionale), Stati Uniti orientali (Ohio), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo) ed Europa (Irlanda).

Sessioni interattive con Ray nella console AWS Glue Studio

Nella pagina Jobs della AWS Glue Studio Console, seleziona l'opzione Jupyter Notebook esistente. Si aprirà una pagina Notebook setup (Configurazione del notebook) in cui è possibile selezionare il kernel. Seleziona il kernel Ray per iniziare una sessione interattiva di Ray. Per ulteriori informazioni, sulle sessioni interattive e su come vengono utilizzate, consulta the section called “Nozioni di base sulle sessioni interattive AWS Glue”.



Sessioni interattive di Ray con il kernel Jupyter

Per utilizzare il kernel Ray al di fuori della AWS Glue Studio console, è necessario installare il `aws-glue-sessions` pacchetto, che pubblichiamo su PyPI. Per ulteriori informazioni sull'utilizzo del pacchetto del kernel, consulta la documentazione di the section called “Nozioni di base sulle sessioni interattive AWS Glue”.

Per aggiornare o installare il kernel, esegui `pip install --upgrade aws-glue-sessions`. Per usare il kernel Ray, avrai bisogno della versione 3.7 o successiva.

Le sessioni interattive di Ray hanno accesso alle stesse librerie e versioni di Ray dei processi Ray. Nell'anteprima, tutte le sessioni interattive di Ray utilizzeranno Ray 2.4.0.

Impostazioni predefinite del timeout della sessione interattiva di Ray

- Timeout (per sessione) predefinito: 8 ore.
- Tempo di inattività predefinito: 1 ora.

Magic supportati dalle sessioni interattive di AWS Glue Ray

I magic per il kernel Jupyter di AWS Glue quando è alla base delle sessioni interattive di Ray sono simili a quelli delle sessioni Spark. Come riferimento, consulta the section called " Configurazione delle sessioni interattive di AWS Glue per Jupyter e notebook AWS Glue Studio".

Sessioni Magic

I magic delle sessioni sono per lo più invariati rispetto a prima dell'anteprima AWS Glue per Ray. Per ulteriori informazioni sulle sessioni dei magic al di fuori di questa anteprima, consulta the section called "Magic supportati dalle sessioni interattive di AWS Glue per Jupyter". Introduciamo un nuovo magic per impostare il tipo di sessione su AWS Glue per Ray.

Nome	Type (Tipo)	Descrizione
`%glue_ray`	Stringa	Cambia il tipo di sessione in AWS Glue per Ray.

AWS Glue config magics

I magic per configurare AWS Glue in una sessione interattiva possono essere diversi tra i tipi di sessione. Attualmente, supportiamo questo sottoinsieme di magic esistenti solo quando si utilizza AWS Glue per Ray.

> **Warning**
>
> Problema noto: **`additional_python_modules`**
> Nell'anteprima delle sessioni interattive di Ray, l'utilizzo del magic di cella `additional_python_modules` causerà problemi durante il salvataggio del notebook. Per

configurare i moduli Python per le sessioni Ray, utilizza il magic `%%configure` per impostare il parametro `pip-install` definito in the section called “Parametri dei processi Ray”.

Nome	Type (Tipo)	Descrizione
`%%configure`	Dizionario	Specifica un dizionario formattato JSON composto da tutti i parametri di configurazione per una sessione. Ciascun parametro può essere specificato qui o tramite magic individuali.
`%iam_role`	Stringa	Specifica un ruolo IAM ARN con cui eseguire la sessione. Predefinito da ~/.aws/configure
`%number_of_workers`	int	Il numero di dipendenti di un specifico worker-type allocati quando viene eseguito un processo. Deve essere impostato anche `worker_type` .
`%worker_type`	Stringa	Nell'anteprima di AWS Glue per Ray, l'unico tipo di worker supportato è Z.2X.
`%additional_python_modules`	Elenco	Elenco separato da virgole di moduli Python aggiuntivi da includere nel cluster (può essere di Pypi o S3).

Magic operativi

Le sessioni Ray di AWS Glue non supportano alcun magic di operazione.

Sessioni Interattive con IAM

Queste sezioni descrivono le considerazioni sulla sicurezza delle sessioni interattive di AWS Glue.

Argomenti

Principali IAM utilizzati con le sessioni interattive

Con le sessioni interattive AWS Glue vengono utilizzati due principali IAM.

- Principale client: il principale del client (un utente o un ruolo) autorizza le operazioni dell'API per sessioni interattive da un client AWS Glue configurato con le credenziali basate sull'identità del principale. Ad esempio, potrebbe trattarsi di un ruolo IAM che in genere si utilizza per accedere alla console di AWS Glue. Potrebbe anche trattarsi di un ruolo fornito a un utente IAM le cui credenziali vengono utilizzate per la AWS Command Line Interface o di un client AWS Glue utilizzato dal kernel Jupyter delle sessioni interattive.
- Ruolo Runtime: il Ruolo runtime è un ruolo IAM che il principale del client passa alle operazioni API delle sessioni interattive. AWS Glue utilizza questo ruolo per eseguire istruzioni nella sessione. Ad esempio, questo ruolo potrebbe essere quello utilizzato per l'esecuzione dei processi ETL di AWS Glue.

  Per ulteriori informazioni, consulta Configurazione di un ruolo runtime.

Configurazione di un principale del client

È necessario allegare una policy di identità al principale del client per consentirgli di chiamare l'API delle sessioni interattive. Questo ruolo deve avere l'accesso `iam:PassRole` al ruolo di esecuzione che si passa per le operazioni API delle sessioni interattive come `CreateSession`. Ad esempio, puoi collegare la policy AWSGlueConsoleFullAccessgestita a un ruolo IAM che consente agli utenti del tuo account con la policy allegata di accedere a tutte le sessioni create nel tuo account (come l'istruzione di runtime o l'istruzione di cancellazione).

Se desideri proteggere la tua sessione e renderla privata solo per determinati ruoli IAM, come quelli associati all'utente che ha creato la sessione, puoi utilizzare il Tag Based Authorization Control di AWS Glue Interactive Session chiamato TagOnCreate. Per ulteriori informazioni, scopri Rendi privata la tua sessione con TagOnCreate come una policy gestita con ambito e basata su tag

proprietari può rendere privata la tua sessione con. TagOnCreate Per ulteriori informazioni sulle politiche basate sull'identità, consulta Politiche basate sull'identità per. AWS Glue

Configurazione di un ruolo runtime

È necessario passare un ruolo IAM all'operazione CreateSession API per consentire l'assunzione e l'esecuzione di istruzioni in AWS Glue sessioni interattive. Il ruolo deve disporre delle stesse autorizzazioni IAM necessarie per l'esecuzione dei processi AWS Glue tipici. Ad esempio, puoi creare un ruolo di servizio utilizzando la AWSGlueServiceRolepolitica che consente di AWS Glue chiamare AWS i servizi per tuo conto. Se utilizzi la console AWS Glue, la console creerà automaticamente un ruolo di servizio per tuo conto o ne utilizzerà uno esistente. Puoi anche creare il tuo ruolo IAM personalizzato e allegare la policy IAM per concedere autorizzazioni simili.

Se desideri proteggere la tua sessione e renderla privata solo per l'utente che ha creato la sessione, puoi utilizzare il controllo di autorizzazione basato su tag di AWS Glue Interactive Session chiamato TagOnCreate. Per ulteriori informazioni, scopri Rendi privata la tua sessione con TagOnCreate in che modo una politica gestita basata su tag proprietari e basata su tag del proprietario può rendere privata la tua sessione con. TagOnCreate Per ulteriori informazioni sulle policy basate sull'identità, consulta la pagina Policy basate su identità per AWS Glue. Se stai creando il ruolo di esecuzione da solo dalla console IAM e desideri rendere privato il tuo servizio con TagOnCreate funzionalità, segui i passaggi seguenti.

1. Creare un ruolo IAM con il tipo di ruolo impostato su `Glue`.
2. Allega questa policy AWS Glue gestita: AwsGlueSessionUserRestrictedServiceRole
3. Prefissa il nome del ruolo con il nome della politica. AwsGlueSessionUserRestrictedServiceRole Ad esempio, puoi creare un ruolo con name AwsGlueSessionUserRestrictedServiceRole-myrole e allegare AWS Glue una policy gestita. AwsGlueSessionUserRestrictedServiceRole
4. Allegare una policy di attendibilità come di seguito per consentire a AWS Glue di assumere il ruolo:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
```

```
        ]
      },
      "Action": [
        "sts:AssumeRole"
      ]
    }
  ]
}
```

Per un kernel Jupyter a sessioni interattive, puoi specificare la chiave `iam_role` nel tuo profilo AWS Command Line Interface. Per ulteriori informazioni, consulta Configurazione di sessioni con ~/.aws/config . Se stai interagendo con sessioni interattive usando un notebook AWS Glue, puoi passare il ruolo di esecuzione in `%iam_role` magic nella prima cella che esegui.

Rendi privata la tua sessione con TagOnCreate

Le sessioni interattive di AWS Glue supportano il tagging e l'autorizzazione basata sui tag (TBAC) per le sessioni interattive come risorsa denominata. Oltre all'utilizzo TagResource delle UntagResource API da parte di TBAC, le sessioni AWS Glue interattive supportano la TagOnCreate funzionalità di «taggare» una sessione con un determinato tag solo durante la creazione della sessione con operazione. CreateSession Ciò significa anche che quei tag verranno rimossi il, alias. DeleteSession UntagOnDelete

TagOnCreate offre un potente meccanismo di sicurezza per rendere la sessione privata al creatore della sessione. Ad esempio, puoi allegare una policy IAM con «owner» RequestTag e valore ${aws:userId} a un client principal (come un utente) per consentire la creazione di una sessione solo se su richiesta viene fornito un tag «owner» con il valore corrispondente dell'userID del chiamante. CreateSession Questa policy consente alle sessioni interattive di AWS Glue di creare una risorsa di sessione e taggare la sessione con il tag userId solo durante il tempo di creazione della sessione. Inoltre, puoi limitare l'accesso (ad esempio le istruzioni in esecuzione) alla tua sessione solo al creatore (alias tag owner con valore ${aws:userId}) della sessione allegando una policy IAM con «owner» ResourceTag al ruolo di esecuzione che hai passato durante CreateSession.

Per semplificare l'utilizzo della TagOnCreate funzionalità che rende privata una sessione per il creatore della sessione, AWS Glue fornisce politiche gestite e ruoli di servizio specializzati.

Se desideri creare una sessione AWS Glue interattiva utilizzando un AssumeRole principale IAM (ovvero utilizzando credenziali fornite assumendo un ruolo IAM) e desideri rendere la sessione privata per il creatore, utilizza politiche simili rispettivamente a e.

AWSGlueSessionUserRestrictedNotebookPolicyAWSGlueSessionUserRestrictedNotebookServiceRole Queste politiche consentono di AWS Glue utilizzare $ {aws:PrincipalTag} per estrarre il valore del tag owner. Ciò richiede il passaggio di un tag userID con valore $ {aws:userId} come nella credenziale di assunzione del ruolo. SessionTag Consulta Tag della sessione ID. Se utilizzi un'istanza Amazon EC2 con un profilo di istanza che vende la credenziale e desideri creare una sessione o interagire con la sessione dall'interno dell'istanza Amazon EC2, dovrai passare un tag userId con valore $ {aws:userId} come nella credenziale di assunzione del ruolo. SessionTag

Ad esempio, se stai creando una sessione utilizzando una credenziale AssumeRole principale IAM e desideri rendere privato il tuo servizio con TagOnCreate funzionalità, procedi nel seguente modo.

1. Crea tu stesso un ruolo runtime dalla console IAM. Allega questo AwsGlueSessionUserRestrictedNotebookServiceruolo di policy AWS Glue gestita e aggiungi al nome del ruolo il nome della policy Role come prefisso. AwsGlueSessionUserRestrictedNotebookService Ad esempio, puoi creare un ruolo con il nome AwsGlueSessionUserRestrictedNotebookServiceRole-MyRole e allegare AWS Glue il ruolo della policy gestita. AwsGlueSessionUserRestrictedNotebookService
2. Allega una policy di attendibilità come di seguito per consentire ad AWS Glue di assumere il ruolo:

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

3. Crea un altro ruolo denominato con un prefisso AwsGlueSessionUserRestrictedNotebookPolicye allega la policy AWS Glue gestita AwsGlueSessionUserRestrictedNotebookPolicyper rendere

privata la sessione. Oltre alla policy gestita, allega la seguente policy in linea per consentire iam: PassRole al ruolo che hai creato nel passaggio 1.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/
AwsGlueSessionUserRestrictedNotebookServiceRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

4. Allega una policy di attendibilità come di seguito al AWS Glue IAM riportato qui sopra per consentirgli di assumere il ruolo.

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

```
    }]
}
```

> **Note**
>
> Facoltativamente, puoi utilizzare un singolo ruolo (ad esempio, il ruolo notebook) e allegare entrambe le politiche gestite sopra riportate AwsGlueSessionUserRestrictedNotebookServiceRole e. AwsGlueSessionUserRestrictedNotebookPolicy Allega anche la policy inline aggiuntiva per consentire `iam:passrole` del tuo ruolo di AWS Glue. E infine allega la policy di fiducia di cui sopra per consentire `sts:AssumeRole` e `sts:TagSession`.

AWSGlueSessionUserRestrictedNotebookPolicy

AWSGlueSessionUserRestrictedNotebookPolicy Fornisce l'accesso per creare una sessione AWS Glue interattiva da un notebook solo se il tag chiave «proprietario» e il valore corrispondono AWS all'ID utente del principale (utente o ruolo). Per ulteriori informazioni, consulta Casi in cui è possibile utilizzare le variabili di policy. Questa policy è collegata al principale (utente o ruolo IAM) che crea notebook di sessioni interattive AWS Glue da AWS Glue Studio. Questa policy, inoltre, consente un accesso sufficiente al notebook AWS Glue Studio per l'interazione con le risorse della sessione interattiva di AWS Glue Studio create con il valore del tag "proprietario" corrispondente all'ID utente AWS del principale. Questa policy nega l'autorizzazione di modificare o rimuovere i tag "proprietario" da una risorsa di sessione AWS Glue dopo la creazione della sessione.

AWSGlueSessionUserRestrictedNotebookServiceRole

AWSGlueSessionUserRestrictedNotebookServiceRoleFornisce un accesso sufficiente al AWS Glue Studio notebook per interagire con le risorse della sessione AWS Glue interattiva create con il valore del tag «owner» che corrisponde all'ID AWS utente del principale (utente o ruolo) del creatore del notebook. Per ulteriori informazioni, consulta Casi in cui è possibile utilizzare le variabili di policy. Questa policy relativa ai ruoli di servizio è allegata al ruolo che viene passato per magia a un notebook o passato come ruolo di esecuzione all' CreateSession API. Questa policy, inoltre, consente al principale di creare una sessione interattiva di AWS Glue da un notebook solo se una chiave di tag "proprietario" e il valore corrispondono all'ID utente AWS del principale. Questa policy nega l'autorizzazione di modificare o rimuovere i tag "proprietario" da una risorsa di sessione AWS Glue dopo la creazione della sessione. Questa policy include anche le autorizzazioni per la scrittura e

la lettura da bucket Amazon S3, la CloudWatch scrittura di log, la creazione e l'eliminazione di tag per le risorse Amazon EC2 utilizzate da. AWS Glue

Rendere private le sessioni con gli utenti

Puoi collegare i AWSGlueSessionUserRestrictedPolicydue ruoli IAM associati a ciascuno degli utenti del tuo account per impedire loro di creare una sessione solo con un tag proprietario con un valore corrispondente al proprio $ {aws:userId}. Invece di utilizzare AWSGlueSessionUserRestrictedNotebookPolicye, è AWSGlueSessionUserRestrictedNotebookServiceRolenecessario utilizzare politiche simili a e rispettivamente. AWSGlueSessionUserRestrictedPolicyAWSGlueSessionUserRestrictedServiceRole Per ulteriori informazioni, consulta la pagina Using identity based policies. Questa policy limita l'accesso a una sessione solo al creatore, il valore ${aws:userID} dell'utente che ha creato la sessione con tag proprietario svelando il proprio ${aws:userID}. Se hai creato tu stesso il ruolo di esecuzione utilizzando la console IAM seguendo i passaggi indicatiConfigurazione di un ruolo runtime, oltre ad allegare la policy AwsGlueSessionUserRestrictedPolicygestita, collega anche la seguente policy in linea a ciascuno degli utenti del tuo account `iam:PassRole` per consentire il ruolo di esecuzione che hai creato in precedenza.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": [
            "arn:aws:iam::*:role/AwsGlueSessionUserRestrictedServiceRole*"
        ],
        "Condition": {
            "StringLike": {
                "iam:PassedToService": [
                    "glue.amazonaws.com"
                ]
            }
        }
    }]
}
```

AWSGlueSessionUserRestrictedPolicy

AWSGlueSessionUserRestrictedPolicyFornisce l'accesso alla creazione di una sessione AWS Glue interattiva utilizzando l' CreateSession API solo se vengono forniti una chiave di tag «proprietario» e un valore corrispondenti al AWS relativo ID utente. Questa politica di identità è allegata all'utente che richiama l' CreateSession API. Questa policy, inoltre, consente l'interazione con le risorse della sessione interattiva di AWS Glue create con un tag "proprietario" e un valore corrispondente al relativo ID utente AWS. Questa policy nega l'autorizzazione di modificare o rimuovere i tag "proprietario" da una risorsa di sessione AWS Glue dopo la creazione della sessione.

AWSGlueSessionUserRestrictedServiceRole

AWSGlueSessionUserRestrictedServiceRoleFornisce l'accesso completo a tutte le AWS Glue risorse ad eccezione delle sessioni e consente agli utenti di creare e utilizzare solo le sessioni interattive associate all'utente. Questa policy include anche altre autorizzazioni necessarie ad AWS Glue per gestire le risorse Glue in altri servizi AWS. La policy permette anche l'aggiunta di tag a risorse AWS Glue in altri servizi AWS.

Considerazioni sulle policy IAM

Le sessioni interattive sono risorse IAM in AWS Glue. Poiché si tratta di risorse IAM, l'accesso e l'interazione a una sessione sono regolati dalle policy IAM. In base alle policy IAM allegate a un principale del client o al ruolo di esecuzione configurato da un amministratore, un principale del client (utente o ruolo) sarà in grado di creare nuove sessioni e interagire con le proprie e con altre.

Se un amministratore ha allegato una policy IAM come AWSGlueConsoleFullAccess o AWSGlueServiceRole che consente l'accesso a tutte le AWS Glue risorse di quell'account, il responsabile del cliente sarà in grado di collaborare tra loro. Ad esempio, un utente sarà in grado di interagire con le sessioni create da altri utenti se le policy lo consentono.

Per configurare una policy su misura per le tue esigenze specifiche, consulta la documentazione IAM sulla configurazione delle risorse per una policy. Ad esempio, per isolare le sessioni che appartengono a un utente, puoi utilizzare la TagOnCreate funzionalità supportata dalle sessioni AWS Glue interattive. Per informazioni, consulta Rendi privata la tua sessione con TagOnCreate .

Le sessioni interattive supportano la limitazione della creazione di sessioni in base a determinate condizioni del VPC. Per informazioni, consulta Policy di controllo che controllano le impostazioni utilizzando le chiavi di condizione.

Conversione di uno script o di un notebook in un AWS Glue processo Glue

È possibile convertire uno script o un notebook in un processo AWS Glue in due modi:

- Utilizza nbconvert per convertire il tuo file di documento notebook Jupyter `.ipynb` in un file `.py`. Per ulteriori informazioni, consulta nbconvert: Convert Notebooks to other formats (nbconvert: convertire notebook in altri formati).
- Carica il file nei notebook di AWS Glue Studio.
  - Nella console AWS Glue Studio, dal menu di navigazione a sinistra scegliere Processi.
  - Nella sezione Create job (Crea processo), scegli Jupyter Notebook (Notebook Jupyter).
  - Nella sezione Options (Opzioni), scegli Upload and edit an existing notebook (Carica e modifica un notebook esistente).
  - Seleziona Choose file (Scegli file) per effettuare il caricamento di un file `.ipynb`.

Sessioni interattive AWS Glue per lo streaming

Commutazione del tipo di sessione streaming

Utilizzo della configurazione magic sessioni interattive AWS Glue, `%streaming`, per definire il processo che si sta eseguendo e inizializzare una sessione interattiva in streaming.

Flusso di input di campionamento per lo sviluppo interattivo

Uno strumento che abbiamo sviluppato per migliorare l'esperienza interattiva nelle sessioni AWS Glue interattive è l'aggiunta di un nuovo metodo `GlueContext` per ottenere un'istantanea di uno stream in modalità statica DynamicFrame. `GlueContext`consente di ispezionare, interagire e implementare il flusso di lavoro.

Con l'istanza di classe `GlueContext`, sarai in grado di localizzare il metodo `getSampleStreamingDynamicFrame`. Gli argomenti richiesti per questo metodo sono:

- `dataFrame`: Lo Spark Streaming DataFrame
- `options`: vedi le opzioni disponibili di seguito

Le opzioni disponibili includono:

- windowSize: viene chiamato anche durata del microbatch. Questo parametro determinerà la durata di attesa di una query di streaming dopo l'attivazione del batch precedente. Il valore del parametro deve essere inferiore a `pollingTimeInMs`.
- pollingTimeInMs: La durata totale dell'esecuzione del metodo. Esso spedirà almeno un micro batch per ottenere registri campione dal flusso di input.
- recordPollingLimit: Questo parametro consente di limitare il numero totale di record che verranno esaminati dallo stream.
- (Facoltativo) È possibile utilizzare anche `writeStreamFunction` per applicare questa funzione personalizzata a ogni funzione di campionamento del registro. Vedi di seguito alcuni esempi in Scala e Python.

Scala

```
val sampleBatchFunction = (batchDF: DataFrame, batchId: Long) => {//Optional but
 you can replace your own forEachBatch function here}
val jsonString: String = s"""{"pollingTimeInMs": "10000", "windowSize": "5
 seconds"}"""
val dynFrame = glueContext.getSampleStreamingDynamicFrame(YOUR_STREAMING_DF,
 JsonOptions(jsonString), sampleBatchFunction)
dynFrame.show()
```

Python

```
def sample_batch_function(batch_df, batch_id):
       //Optional but you can replace your own forEachBatch function here
options = {
            "pollingTimeInMs": "10000",
            "windowSize": "5 seconds",
        }
glue_context.getSampleStreamingDynamicFrame(YOUR_STREAMING_DF, options,
 sample_batch_function)
```

> **Note**
>
> Se il `DynFrame` d'esempio è vuoto, le ragioni possono essere varie:

- La fonte di streaming è impostata su "Più recente" e non sono stati inseriti nuovi dati durante il periodo di campionamento.
- Il tempo del polling non è sufficiente per elaborare i registri importati. I dati non verranno visualizzati a meno che l'intero batch non sia stato elaborato.

Esecuzione di applicazioni di streaming in sessioni interattive

Nelle sessioni interattive AWS Glue, è possibile eseguire un'applicazione di streaming AWS Glue nello stesso modo in cui si creerebbe un'applicazione di streaming nella Console AWS Glue. Poiché le sessioni interattive sono basate su sessione, l'individuazione di eccezioni nel runtime non provoca l'interruzione della sessione. Ora abbiamo l'ulteriore vantaggio di sviluppare iterativamente la funzione batch. Ad esempio:

```
def batch_function(data_frame, batch_id):
    log.info(data_frame.count())
    invalid_method_call()
glueContext.forEachBatch(frame=streaming_df, batch_function = batch_function, options =
 {**})
```

Nell'esempio precedente, abbiamo incluso un utilizzo non valido di un metodo e a differenza dei normali processi AWS Glue che usciranno dall'intera applicazione, le definizioni e il contesto di codifica dell'utente vengono conservati interamente e la sessione è ancora operativa. Non è necessario avviare un nuovo cluster e rieseguire tutte le trasformazioni precedenti. Ciò consente di concentrarsi sull'iterazione rapida delle implementazioni delle funzioni batch per ottenere risultati desiderati.

È importante notare che le sessioni interattive valutano ogni istruzione in modo bloccante per far sì che la sessione esegua una sola istruzione alla volta. Poiché le query di streaming sono continue e senza fine, le sessioni con query di streaming attive non saranno in grado di gestire alcuna istruzione di follow-up a meno che non vengano interrotte. Puoi emettere il comando di interruzione direttamente da Jupyter Notebook e il nostro kernel gestirà la cancellazione per te.

Prendi come esempio la seguente sequenza di istruzioni in attesa dell'esecuzione:

```
Statement 1:
```

```
      val number = df.count()
      #Spark Action with deterministic result
      Result: 5

Statement 2:
      streamingQuery.start().awaitTermination()
      #Spark Streaming Query that will be executing continously
      Result: Constantly updated with each microbatch

Statement 3:
      val number2 = df.count()
      #This will not be executed as previous statement will be running indefinitely
```

Sviluppo e test di script di processo AWS Glue in locale

Quando sviluppi e testi gli script di processo Spark per AWS Glue, hai a disposizione molteplici opzioni:

- Console AWS Glue Studio
  - Visual editor (Editor visivo)
  - Editor di script
  - Notebook AWS Glue Studio
- Sessioni interattive
  - Jupyter Notebook
- immagine Docker
  - Sviluppo locale
  - Sviluppo remoto
- Libreria ETL AWS Glue Studio
  - Sviluppo locale

Puoi scegliere una delle opzioni sopra elencate in base alle tue esigenze.

Se preferisci un'esperienza priva di codice o con meno codice da gestire, l'editor visivo AWS Glue Studio è un'ottima scelta.

Se preferisci un'esperienza interattiva con notebook, AWS Glue Studio notebook è un'ottima scelta. Per ulteriori informazioni, consulta Utilizzare Notebooks with AWS Glue Studio e AWS Glue Se desideri utilizzare il tuo ambiente locale, le sessioni interattive sono un'ottima scelta. Per ulteriori informazioni, consulta Utilizzare le sessioni interattive con AWS Glue.

Se preferisci un'esperienza di sviluppo locale/remoto, l'immagine Docker è un'ottima scelta. Ciò ti consente di sviluppare e testare gli script di processo Spark per AWS Glue in qualsiasi ambiente, senza dover sostenere i costi di AWS Glue.

Se preferisci uno sviluppo locale senza Docker, l'installazione della directory della libreria ETL AWS Glue in locale è un'ottima scelta.

Sviluppo utilizzando AWS Glue Studio

L'editor visivo AWS Glue Studio è un'interfaccia grafica che consente di creare, eseguire e monitorare in modo semplice processi di estrazione, trasformazione e caricamento (ETL) in AWS Glue. È possibile comporre visivamente flussi di lavoro per la trasformazione dei dati ed eseguirli senza problemi sul motore ETL serverless basato su Apache Spark di AWS Glue. È possibile esaminare lo schema e i risultati dei dati in ogni fase del processo. Per ulteriori informazioni, consulta la Guida per l'utente di AWS Glue Studio.

Sviluppo tramite le sessioni interattive

Le sessioni interattive consentono di creare e testare le applicazioni dall'ambiente che preferisci. Per ulteriori informazioni, consulta Utilizzare le sessioni interattive con AWS Glue.

Sviluppo tramite un'immagine Docker

> **Note**
>
> Le istruzioni contenute in questa sezione non sono state testate sui sistemi operativi Microsoft Windows.
> Per lo sviluppo e i test locali su piattaforme Windows, consulta il blog Come costruire una pipeline ETL AWS Glue localmente senza un account AWS

Per una piattaforma dati pronta per la produzione, il processo di sviluppo e la pipeline CI/CD per i processi AWS Glue sono un argomento chiave. È possibile sviluppare e testare in modo flessibile i processi AWS Glue in un container Docker. AWS Glue ospita immagini Docker su Docker Hub per

configurare l'ambiente di sviluppo con utility aggiuntive. È possibile utilizzare il tuo IDE, notebook o REPL preferito utilizzando la libreria ETL AWS Glue. Questo argomento illustra come sviluppare e testare i processi della versione 4.0 di AWS Glue su un container Docker utilizzando l'immagine Docker.

Le seguenti immagini Docker sono disponibili per AWS Glue su Docker Hub.

- Per la versione 4.0 di AWS Glue: `amazon/aws-glue-libs:glue_libs_4.0.0_image_01`
- Per AWS Glue versione 3.0: `amazon/aws-glue-libs:glue_libs_3.0.0_image_01`
- Per AWS Glue versione 2.0: `amazon/aws-glue-libs:glue_libs_2.0.0_image_01`

Queste immagini sono per x86_64. Si consiglia di eseguire il test su questa architettura. Tuttavia, potrebbe essere possibile rielaborare una soluzione di sviluppo locale su immagini di base non supportate.

In questo esempio viene descritto l'utilizzo di `amazon/aws-glue-libs:glue_libs_4.0.0_image_01` e l'esecuzione del container su un computer locale. Questa immagine del container è stata testata per i processi della versione 3.3 di Spark di AWS Glue. Questa immagine contiene quanto segue:

- Amazon Linux
- Libreria ETL AWS Glue (aws-glue-libs)
- Apache Spark 3.3.0
- Spark History Server
- Jupyter Lab
- Livy
- Altre dipendenze della libreria (lo stesso set di quelle del sistema di processi AWS Glue)

Completa una delle seguenti sezioni in base alle tue esigenze:

- Configurazione del container per l'utilizzo di spark-submit
- Configurazione del container per l'utilizzo della shell REPL (PySpark)
- Configurazione del container per l'utilizzo di Pytest
- Configurazione del container per l'utilizzo di Jupyter Lab
- Configurazione del container per l'utilizzo di Visual Studio Code

Prerequisiti

Prima di iniziare, verifica che Docker sia installato e che il daemon Docker sia in esecuzione. Per istruzioni sull'installazione, consulta la documentazione Docker per Mac o Linux. Il computer che esegue Docker ospita il container AWS Glue. Inoltre, verifica di avere almeno 7 GB di spazio su disco per l'immagine sull'host su cui è in esecuzione Docker.

Per ulteriori informazioni sulle restrizioni durante lo sviluppo locale del codice AWS Glue, consulta Local Development Restrictions (restrizioni sullo sviluppo locale).

Configurazione di AWS

Per abilitare le chiamate API AWS dal container, configura le credenziali AWS seguendo i passaggi di seguito. Nelle sezioni seguenti, useremo questo profilo denominato AWS.

1. Configura AWS CLI, impostando un profilo denominato. Per ulteriori informazioni sulla configurazione di AWS CLI, consulta Impostazioni del file di configurazione e delle credenziali nella documentazione di AWS CLI.
2. Esegui il comando seguente in un terminale:

```
PROFILE_NAME="<your_profile_name>"
```

Potrebbe anche essere necessario impostare la variabile di ambiente AWS_REGION per specificare la Regione AWS a cui inviare le richieste.

Configurazione ed esecuzione del container

La configurazione del container per eseguire il codice PySpark tramite il comando spark-submit include i seguenti passaggi di alto livello:

1. Estrarre l'immagine da Docker Hub.
2. Eseguire il container.

Estrazione dell'immagine da Docker Hub

Esegui il comando seguente per estrarre l'immagine da Docker Hub:

```
docker pull amazon/aws-glue-libs:glue_libs_4.0.0_image_01
```

Esecuzione del container

Ora puoi eseguire un container utilizzando questa immagine. Puoi scegliere una delle opzioni seguenti in base alle tue esigenze.

spark-submit

Puoi eseguire uno script di processo di AWS Glue eseguendo il comando `spark-submit` sul container.

1. Scrivi lo script e salvalo come `sample1.py` nella directory `/local_path_to_workspace`. Il codice di esempio è incluso come appendice in questo argomento.

```
$ WORKSPACE_LOCATION=/local_path_to_workspace
$ SCRIPT_FILE_NAME=sample.py
$ mkdir -p ${WORKSPACE_LOCATION}/src
$ vim ${WORKSPACE_LOCATION}/src/${SCRIPT_FILE_NAME}
```

2. Esegui il comando seguente per eseguire il comando `spark-submit` sul container per inviare una nuova applicazione Spark:

```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $WORKSPACE_LOCATION:/
home/glue_user/workspace/ -e AWS_PROFILE=$PROFILE_NAME -e DISABLE_SSL=true
 --rm -p 4040:4040 -p 18080:18080 --name glue_spark_submit amazon/aws-glue-
libs:glue_libs_4.0.0_image_01 spark-submit /home/glue_user/workspace/src/
$SCRIPT_FILE_NAME
...22/01/26 09:08:55 INFO DAGScheduler: Job 0 finished: fromRDD at
 DynamicFrame.scala:305, took 3.639886 s
root
|-- family_name: string
|-- name: string
|-- links: array
| |-- element: struct
| | |-- note: string
| | |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
| |-- element: struct
| | |-- scheme: string
| | |-- identifier: string
|-- other_names: array
| |-- element: struct
```

```
 | | |-- lang: string
 | | |-- note: string
 | | |-- name: string
 |-- sort_name: string
 |-- images: array
 | |-- element: struct
 | | |-- url: string
 |-- given_name: string
 |-- birth_date: string
 |-- id: string
 |-- contact_details: array
 | |-- element: struct
 | | |-- type: string
 | | |-- value: string
 |-- death_date: string

 ...
```

3. (Facoltativo) Configura `spark-submit` in modo che corrisponda all'ambiente in uso. Ad esempio, puoi trasferire le dipendenze con la configurazione `--jars`. Per ulteriori informazioni, consulta Avvio delle applicazioni con spark-submit nella documentazione di Spark.

Shell REPL (Pyspark)

Puoi eseguire la shell REPL (read-eval-print loop) per lo sviluppo interattivo.

Esegui il seguente comando per eseguire il comando PySpark sul container per avviare la shell REPL:

```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -e AWS_PROFILE=$PROFILE_NAME -e
 DISABLE_SSL=true --rm -p 4040:4040 -p 18080:18080 --name glue_pyspark amazon/aws-glue-
libs:glue_libs_4.0.0_image_01 pyspark
...
 ____ __
 / __/__ ___ _____/ /__
 _\ \/ _ \/ _ `/ __/ '_/
 /__ / .__/\_,_/_/ /_/\_\  version 3.1.1-amzn-0
 /_/

Using Python version 3.7.10 (default, Jun 3 2021 00:02:01)
Spark context Web UI available at http://56e99d000c99:4040
Spark context available as 'sc' (master = local[*], app id = local-1643011860812).
SparkSession available as 'spark'.
```

```
>>>
```

Pytest

Per l'unit test, puoi utilizzare pytest per gli script di processo Spark di AWS Glue.

Esegui i comandi seguenti per la preparazione.

```
$ WORKSPACE_LOCATION=/local_path_to_workspace
$ SCRIPT_FILE_NAME=sample.py
$ UNIT_TEST_FILE_NAME=test_sample.py
$ mkdir -p ${WORKSPACE_LOCATION}/tests
$ vim ${WORKSPACE_LOCATION}/tests/${UNIT_TEST_FILE_NAME}
```

Esegui il comando seguente per eseguire `pytest` nella suite di test:

```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $WORKSPACE_LOCATION:/home/glue_user/
workspace/ -e AWS_PROFILE=$PROFILE_NAME -e DISABLE_SSL=true --rm -p 4040:4040 -p
 18080:18080 --name glue_pytest amazon/aws-glue-libs:glue_libs_4.0.0_image_01 -c
 "python3 -m pytest"
starting org.apache.spark.deploy.history.HistoryServer,
 logging to /home/glue_user/spark/logs/spark-glue_user-
org.apache.spark.deploy.history.HistoryServer-1-5168f209bd78.out
*============================================================= test session starts
 ==============================================================
*platform linux -- Python 3.7.10, pytest-6.2.3, py-1.11.0, pluggy-0.13.1
rootdir: /home/glue_user/workspace
plugins: anyio-3.4.0
*collected 1 item  *

tests/test_sample.py . [100%]

============================================================== warnings summary
 ===============================================================
tests/test_sample.py::test_counts
 /home/glue_user/spark/python/pyspark/sql/context.py:79: DeprecationWarning: Deprecated
 in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
 DeprecationWarning)

-- Docs: https://docs.pytest.org/en/stable/warnings.html
======================================================== 1 passed, *1 warning* in
 21.07s =========================================================
```

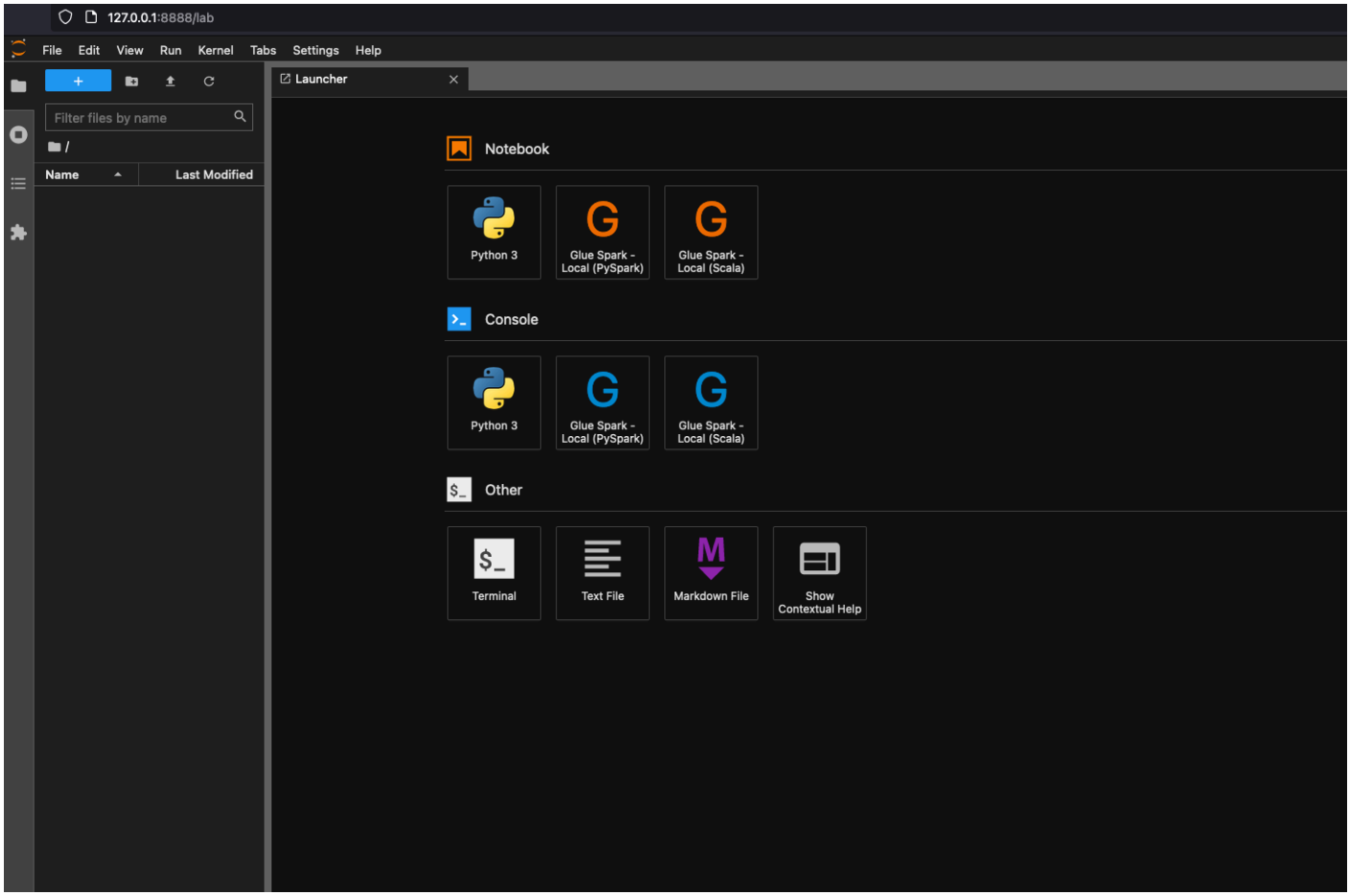
Jupyter Lab

Puoi avviare Jupyter per lo sviluppo interattivo e le query ad hoc sui notebook.

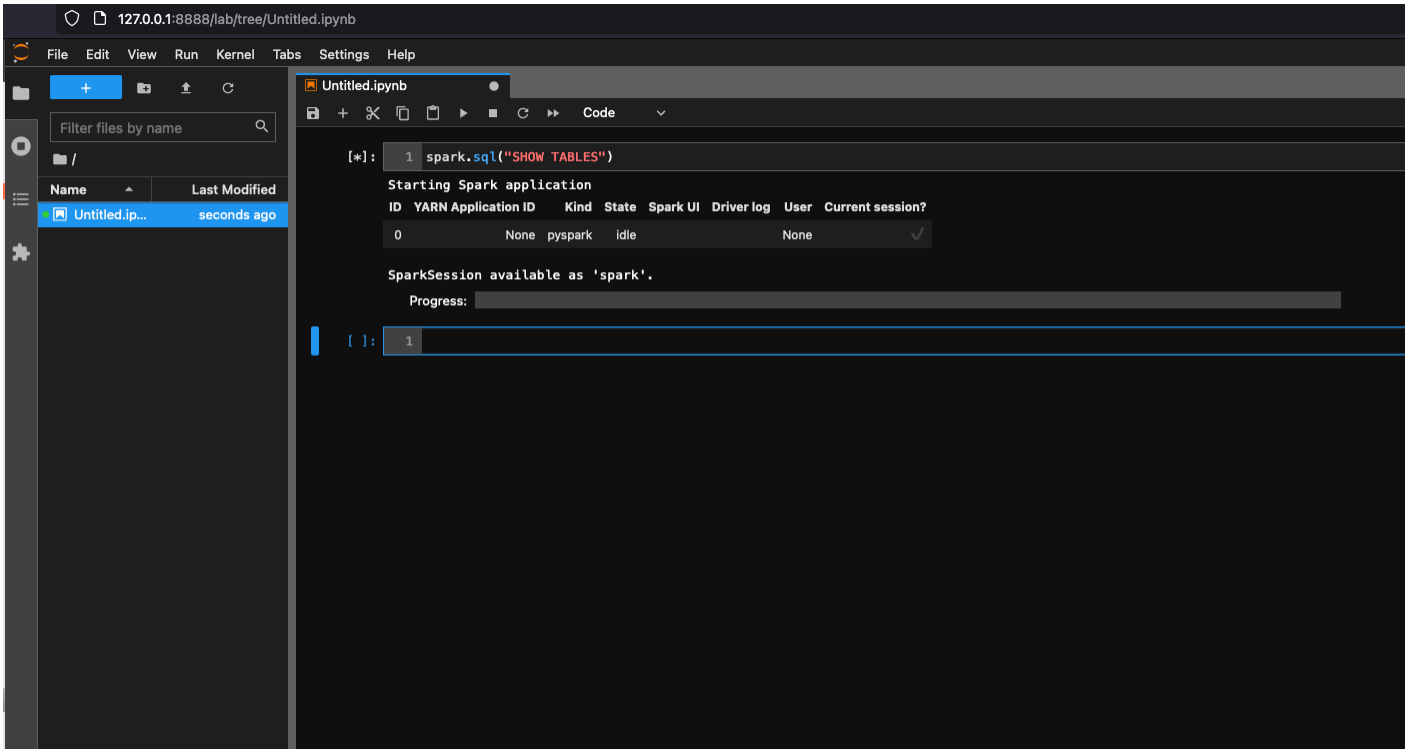
1. Esegui il comando seguente per avviare Jupyter Lab:

```
$ JUPYTER_WORKSPACE_LOCATION=/local_path_to_workspace/jupyter_workspace/
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $JUPYTER_WORKSPACE_LOCATION:/
home/glue_user/workspace/jupyter_workspace/ -e AWS_PROFILE=$PROFILE_NAME -e
 DISABLE_SSL=true --rm -p 4040:4040 -p 18080:18080 -p 8998:8998 -p 8888:8888 --name
 glue_jupyter_lab amazon/aws-glue-libs:glue_libs_4.0.0_image_01 /home/glue_user/
jupyter/jupyter_start.sh
...
[I 2022-01-24 08:19:21.368 ServerApp] Serving notebooks from local directory: /home/
glue_user/workspace/jupyter_workspace
[I 2022-01-24 08:19:21.368 ServerApp] Jupyter Server 1.13.1 is running at:
[I 2022-01-24 08:19:21.368 ServerApp] http://faa541f8f99f:8888/lab
[I 2022-01-24 08:19:21.368 ServerApp] or http://127.0.0.1:8888/lab
[I 2022-01-24 08:19:21.368 ServerApp] Use Control-C to stop this server and shut down
 all kernels (twice to skip confirmation).
```

2. Apri http://127.0.0.1:8888/lab nel browser web del tuo computer locale, per visualizzare l'interfaccia utente di Jupyter lab.



3. Scegli Glue Spark Local (PySpark) in Notebook. Puoi iniziare a sviluppare codice nell'interfaccia utente interattiva del notebook Jupyter.



Configurazione del container per l'utilizzo di Visual Studio Code

Prerequisiti:

1. Installare Visual Studio Code.
2. Installare Python.
3. Installare Visual Studio Code Remote - Containers (Visual Studio Code Remote - Container)
4. Aprire la cartella del WorkSpace in Visual Studio Code.
5. Selezionare Settings (Impostazioni).
6. Scegliere WorkSpace.
7. Scegliere Open Settings (JSON) (Apri impostazioni [JSON]).
8. Incollare il seguente codice JSON e salvarlo.

```
{
    "python.defaultInterpreterPath": "/usr/bin/python3",
    "python.analysis.extraPaths": [
        "/home/glue_user/aws-glue-libs/PyGlue.zip:/home/glue_user/spark/python/lib/
py4j-0.10.9-src.zip:/home/glue_user/spark/python/",
    ]
```

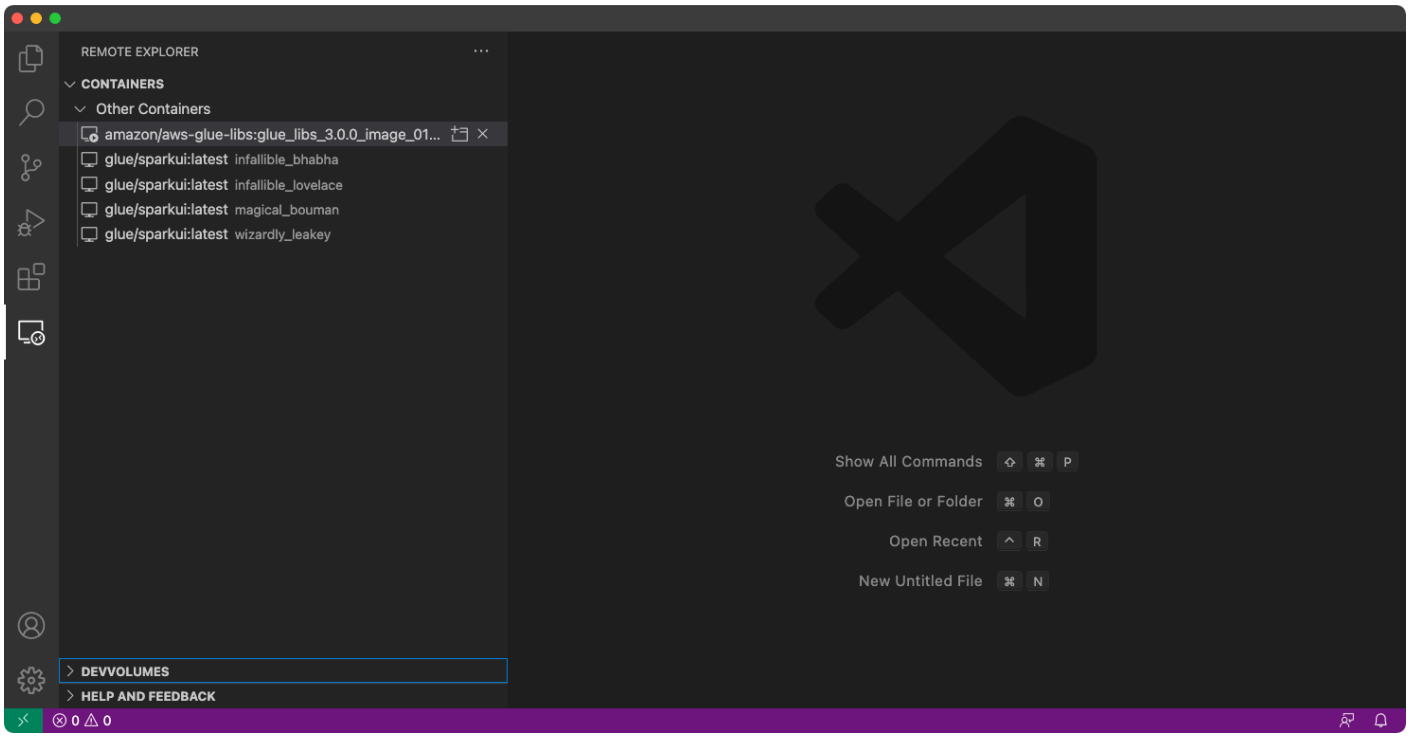
```
}
```

Fasi:

1. Eseguire il container Docker.

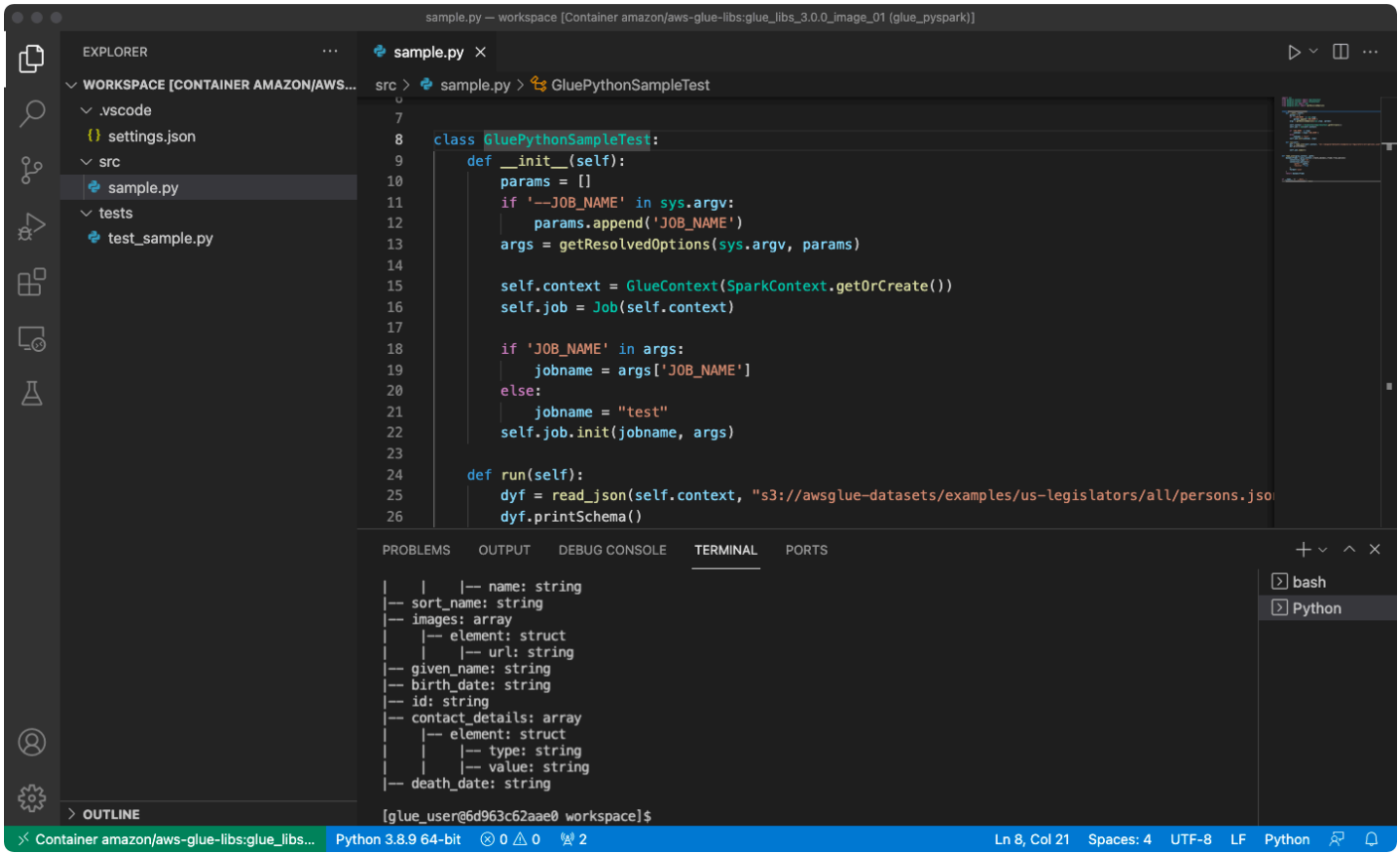
```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $WORKSPACE_LOCATION:/
home/glue_user/workspace/ -e AWS_PROFILE=$PROFILE_NAME -e DISABLE_SSL=true
 --rm -p 4040:4040 -p 18080:18080 --name glue_pyspark amazon/aws-glue-
libs:glue_libs_4.0.0_image_01 pyspark
```

2. Avviare Visual Studio Code.
3. Scegliere Remote Explorer nel menu a sinistra, quindi `amazon/aws-glue-libs:glue_libs_4.0.0_image_01`.



4. Cliccare con il tasto destro e scegliere Attach to Container (Allega al container). Se viene visualizzata una finestra di dialogo, scegliere Got it (Fatto).
5. Aprire `/home/glue_user/workspace/`.
6. Creare uno script Glue PySpark e scegliere Run (Esegui).

   Visualizzerai l'esecuzione corretta dello script.



Appendice: codice di esempio di processo di AWS Glue per i test

Questa appendice fornisce script come codice di esempio di processo di AWS Glue per motivi di test.

sample.py: codice di esempio per utilizzare la libreria ETL AWS Glue con una chiamata API di Amazon S3

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions


class GluePythonSampleTest:
    def __init__(self):
        params = []
        if '--JOB_NAME' in sys.argv:
            params.append('JOB_NAME')
        args = getResolvedOptions(sys.argv, params)
```

```
        self.context = GlueContext(SparkContext.getOrCreate())
        self.job = Job(self.context)

        if 'JOB_NAME' in args:
            jobname = args['JOB_NAME']
        else:
            jobname = "test"
        self.job.init(jobname, args)

    def run(self):
        dyf = read_json(self.context, "s3://awsglue-datasets/examples/us-legislators/
all/persons.json")
        dyf.printSchema()

        self.job.commit()


def read_json(glue_context, path):
    dynamicframe = glue_context.create_dynamic_frame.from_options(
        connection_type='s3',
        connection_options={
            'paths': [path],
            'recurse': True
        },
        format='json'
    )
    return dynamicframe


if __name__ == '__main__':
    GluePythonSampleTest().run()
```

Il codice di cui sopra richiede le autorizzazioni Amazon S3 in AWS IAM. È necessario concedere la policy gestita da IAM `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess` o una policy personalizzata IAM che consente di chiamare `ListBucket` e `GetObject` per il percorso Amazon S3.

test_sample.py: codice di esempio per l'unit test di sample.py.

```
import pytest
from pyspark.context import SparkContext
from awsglue.context import GlueContext
```

```
from awsglue.job import Job
from awsglue.utils import getResolvedOptions
import sys
from src import sample


@pytest.fixture(scope="module", autouse=True)
def glue_context():
    sys.argv.append('--JOB_NAME')
    sys.argv.append('test_count')

    args = getResolvedOptions(sys.argv, ['JOB_NAME'])
    context = GlueContext(SparkContext.getOrCreate())
    job = Job(context)
    job.init(args['JOB_NAME'], args)

    yield(context)

    job.commit()


def test_counts(glue_context):
    dyf = sample.read_json(glue_context, "s3://awsglue-datasets/examples/us-
legislators/all/persons.json")
    assert dyf.toDF().count() == 1961
```

Sviluppo tramite la libreria ETL AWS Glue

La libreria ETL AWS Glue è disponibile in un bucket Amazon S3 pubblico e può essere utilizzata dal sistema di compilazione Apache Maven. Questo consente di sviluppare e testare gli script di estrazione, trasformazione e caricamento (ETL) Python e Scala in locale, senza la necessità di una connessione di rete. Consigliamo lo sviluppo locale con l'immagine Docker, in quanto fornisce un ambiente correttamente configurato per l'uso di questa libreria.

Lo sviluppo locale è disponibile per tutte le versioni di AWS Glue, tra cui le versioni di AWS Glue 0.9, 1.0, 2.0 e successive. Per informazioni sulle versioni di Python e Apache Spark disponibili con AWS Glue, consulta Glue version job property.

La libreria viene rilasciata con la licenza software Amazon (https://aws.amazon.com/asl).

Limitazioni di sviluppo locali

Tieni presente le seguenti limitazioni quando utilizzi la libreria Scala di AWS Glue per sviluppare localmente.

- Evita di creare un jar di assembly ("fat jar" o "uber jar") con la libreria AWS Glue, in quanto causa la disabilitazione delle seguenti caratteristiche:
  - Segnalibri di processo
  - Writer AWS Glue Parquet (Utilizzo del formato Parquet in AWS Glue)
  - Trasformazione FillMissingValues (Scala o Python)

  Queste caratteristiche sono disponibili solo all'interno del sistema di processi AWS Glue.
- La trasformazione FindMatches non è supportata con lo sviluppo locale.
- Il lettore CSV SIMD vettorializzato non è supportato nello sviluppo locale.
- La proprietà customJdbcDriverS3Path per il caricamento del driver JDBC dal percorso S3 non è supportata nello sviluppo locale. In alternativa, puoi scaricare il driver JDBC in locale e caricarlo da lì.
- Qualità dei dati di Glue non è supportato nello sviluppo locale.

Sviluppo in locale con Python

Completa alcune fasi preliminari, quindi utilizza le utilità AWS Glue per testare e inviare lo script ETL Python.

Prerequisiti per lo sviluppo in locale con Python

Completa queste fasi per preparare allo sviluppo di Python locale:

1. Clona il repository Python AWS Glue da GitHub (https://github.com/awslabs/aws-glue-libs).
2. Completa una delle seguenti operazioni:
   - Per AWS Glue versione 0.9, controlla il ramo `glue-0.9`.
   - Per AWS Glue versione 1.0, controlla il ramo `glue-1.0`. Tutte le versioni successive a AWS Glue 0.9 supportano Python 3.
   - Per AWS Glue versione 2.0, controlla il ramo `glue-2.0`.
   - Per AWS Glue versione 3.0, controlla il ramo `glue-3.0`.
   - Per AWS Glue versione 4.0, controlla il ramo `master`.

3. Installa Apache Maven dal seguente percorso: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-common/apache-maven-3.6.0-bin.tar.gz.
4. Installa la distribuzione Apache Spark da uno dei seguenti percorsi:
   - Per AWS Glue versione 0.9: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-0.9/spark-2.2.1-bin-hadoop2.7.tgz
   - Per AWS Glue versione 1.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-1.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Per AWS Glue versione 2.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-2.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Per AWS Glue versione 3.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-3.0/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3.tgz
   - Per la versione 4.0 di AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-4.0/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0.tgz
5. Esporta la variabile di ambiente `SPARK_HOME` impostandola sulla posizione della radice estratta dall'archivio Spark. Ad esempio:
   - Per AWS Glue versione 0.9: `export SPARK_HOME=/home/$USER/spark-2.2.1-bin-hadoop2.7`
   - Per AWS Glue versione 1.0 e 2.0: `export SPARK_HOME=/home/$USER/spark-2.4.3-bin-spark-2.4.3-bin-hadoop2.8`
   - Per AWS Glue versione 3.0: `export SPARK_HOME=/home/$USER/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3`
   - Per la versione 4.0 di AWS Glue: `export SPARK_HOME=/home/$USER/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0`

Esecuzione dello script ETL Python

Con i file jar AWS Glue disponibili per lo sviluppo locale, puoi eseguire il pacchetto AWS Glue Python in locale.

Utilizza le seguenti utilità e framework per testare ed eseguire lo script Python. I comandi elencati nella tabella seguente vengono eseguiti dalla directory root del pacchetto AWS Glue Python.

Utility	Comando	Descrizione
AWS Glue Shell	`./bin/gluepyspark`	Inserisci ed esegui script Python in una shell che si integra con le librerie ETL AWS Glue.
Invio AWS Glue	`./bin/gluesparksubmit`	Invia uno script Python completo per l'esecuzione.
Pytest	`./bin/gluepytest`	Scrivi ed esegui unit test del codice Python. Il modulo pytest deve essere installato e disponibile in PATH. Per ulteriori informazioni, consulta la documentazione del pytest.

Sviluppo in locale con Scala

Completa alcune fasi preliminari, quindi emetti un comando Maven per eseguire lo script ETL Scala in locale.

Prerequisiti per lo sviluppo in locale con Scala

Completa queste fasi per la preparazione allo sviluppo Scala locale.

Fase 1: installare il software

In questa fase devi installare il software e impostare la variabile di ambiente richiesta.

1. Installa Apache Maven dal seguente percorso: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-common/apache-maven-3.6.0-bin.tar.gz.
2. Installa la distribuzione Apache Spark da uno dei seguenti percorsi:
   - Per AWS Glue versione 0.9: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-0.9/spark-2.2.1-bin-hadoop2.7.tgz
   - Per AWS Glue versione 1.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-1.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Per AWS Glue versione 2.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-2.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Per AWS Glue versione 3.0: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-3.0/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3.tgz

- Per la versione 4.0 di AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-4.0/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0.tgz

3. Esporta la variabile di ambiente `SPARK_HOME` impostandola sulla posizione della radice estratta dall'archivio Spark. Ad esempio:
   - Per AWS Glue versione 0.9: `export SPARK_HOME=/home/$USER/spark-2.2.1-bin-hadoop2.7`
   - Per AWS Glue versione 1.0 e 2.0: `export SPARK_HOME=/home/$USER/spark-2.4.3-bin-spark-2.4.3-bin-hadoop2.8`
   - Per AWS Glue versione 3.0: `export SPARK_HOME=/home/$USER/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3`
   - Per la versione 4.0 di AWS Glue: `export SPARK_HOME=/home/$USER/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0`

Fase 2: configurare il progetto Maven

Utilizza il file `pom.xml` seguente come un modello per le applicazioni Scala AWS Glue. Contiene gli elementi obbligatori `plugins`, `dependencies` e `repositories`. Sostituisci la stringa `Glue version` con uno dei seguenti:

- `4.0.0` per AWS Glue versione 4.0
- `3.0.0` per AWS Glue versione 3.0
- `1.0.0` per AWS Glue versione 1.0 o 2.0
- `0.9.0` per AWS Glue versione 0.9

```
  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.amazonaws</groupId>
    <artifactId>AWSGlueApp</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>${project.artifactId}</name>
    <description>AWS ETL application</description>

        <properties>
```

```
            <scala.version>2.11.1 for AWS Glue 2.0 or below, 2.12.7 for AWS Glue 3.0
 and 4.0</scala.version>
            <glue.version>Glue version with three numbers (as mentioned earlier)</
glue.version>
        </properties>
    <dependencies>
        <dependency>
            <groupId>org.scala-lang</groupId>
            <artifactId>scala-library</artifactId>
            <version>${scala.version}</version>
   <!-- A "provided" dependency, this will be ignored when you package your application
 -->
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>AWSGlueETL</artifactId>
   <version>${glue.version}</version>
            <!-- A "provided" dependency, this will be ignored when you package your
 application -->
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <repositories>
        <repository>
            <id>aws-glue-etl-artifacts</id>
            <url>https://aws-glue-etl-artifacts.s3.amazonaws.com/release/</url>
        </repository>
    </repositories>
    <build>
        <sourceDirectory>src/main/scala</sourceDirectory>
        <plugins>
            <plugin>
                <!-- see http://davidb.github.com/scala-maven-plugin -->
                <groupId>net.alchim31.maven</groupId>
                <artifactId>scala-maven-plugin</artifactId>
                <version>3.4.0</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
                        </goals>
```

```
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>1.6.0</version>
            <executions>
                <execution>
                    <goals>
                    <goal>java</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
            <systemProperties>
                <systemProperty>
                    <key>spark.master</key>
                    <value>local[*]</value>
                </systemProperty>
                <systemProperty>
                    <key>spark.app.name</key>
                    <value>localrun</value>
                </systemProperty>
                <systemProperty>
                    <key>org.xerial.snappy.lib.name</key>
                    <value>libsnappyjava.jnilib</value>
                </systemProperty>
            </systemProperties>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-enforcer-plugin</artifactId>
            <version>3.0.0-M2</version>
            <executions>
                <execution>
                    <id>enforce-maven</id>
                    <goals>
                        <goal>enforce</goal>
                    </goals>
                    <configuration>
                        <rules>
                            <requireMavenVersion>
```

```
                                    <version>3.5.3</version>
                                </requireMavenVersion>
                            </rules>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
    <!-- The shade plugin will be helpful in building a uberjar or fatjar.
    You can use this jar in the AWS Glue runtime environment. For more information, see
 https://maven.apache.org/plugins/maven-shade-plugin/ -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>3.2.4</version>
                <configuration>
                    <!-- any other shade configurations -->
                </configuration>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Esecuzione dello script ETL Scala

Esegui il comando seguente dalla directory root del progetto Maven per eseguire lo script ETL Scala.

```
mvn exec:java -Dexec.mainClass="mainClass" -Dexec.args="--JOB-NAME jobName"
```

Sostituisci *mainClass* con il nome della classe completo della classe principale dello script. Sostituisci *jobName* con il nome del processo desiderato.

Configurazione di un ambiente di test

Per esempi di configurazione di un ambiente di test locale, consulta i seguenti articoli del blog:

- Come costruire una pipeline ETL AWS Glue localmente senza un account AWS
- Sviluppo di processi ETL AWS Glue localmente usando un container

Per utilizzare endpoint di sviluppo o notebook per testare gli script ETL, consulta Utilizzo di endpoint per lo sviluppo di script.

> **Note**
>
> Gli endpoint di sviluppo non sono supportati per l'utilizzo con i processi AWS Glue versione 2.0. Per ulteriori informazioni, consulta Esecuzione di processi ETL Spark con tempi di avvio ridotti.

Endpoint di sviluppo

> **Note**
>
> L'esperienza su console per gli endpoint di sviluppo è stata rimossa a partire dal 31 marzo 2023. La creazione, l'aggiornamento e il monitoraggio degli endpoint di sviluppo sono ancora disponibili tramite l'API endpoint di sviluppo e la CLI AWS Glue.

Consigliamo vivamente di eseguire la migrazione dagli endpoint di sviluppo alle sessioni interattive per i motivi elencati di seguito. Per le azioni necessarie per la migrazione dagli endpoint di sviluppo alle sessioni interattive, consulta Migrazione dagli endpoint di sviluppo alle sessioni interattive.

Descrizione	Endpoint dev	Sessioni interattive
Supporto per la versione Glue	Supporta AWS Glue versioni 0.9 e 1.0	Supporta AWS Glue versione 2.0 e successive
Gli endpoint di sviluppo non sono disponibili nelle Regioni Asia Pacifico (Giacarta) (`ap-southeast-3`), Medio Oriente (Emirati Arabi Uniti) (`me-central-1`), Europa	Le sessioni interattive non sono attualmente disponibili in Medio Oriente (Emirati Arabi Uniti) (`me-central-1`), ma potrebbero essere rese disponibili in futuro	

Descrizione	Endpoint dev	Sessioni interattive
(Spagna) (`eu-south-2`), Europa (Zurigo) (`eu-central-2`) o altre nuove Regioni in futuro		
Metodo di accesso al cluster Spark	Supporta SSH, shell (interprete di comandi) REPL, notebook Jupyter, IDE (ad esempio, PyCharm)	supporta notebook AWS Glue Studio, notebook Jupyter, vari IDE (ad esempio, Visual Studio Code, PyCharm) e notebook SageMaker
Tempo per la prima query	Richiede 10-15 minuti per la configurazione di un cluster Spark	Può richiedere fino a 1 minuto per la configurazione di un cluster Spark temporaneo
Modello dei prezzi	AWS addebita gli endpoint di sviluppo in base al tempo di fornitura dell'endpoint e al numero di DPU. Gli endpoint di sviluppo non scadono. È prevista una durata di fatturazione minima di 10 minuti per ogni endpoint di sviluppo fornito. Inoltre, AWS addebita il notebook Jupyter sulle istanze Amazon EC2 e i notebook SageMaker quando si configurano con un endpoint di sviluppo.	AWS addebita le sessioni interattive in base al tempo di attività della sessione e al numero di DPU. Le sessioni interattive hanno timeout di inattività configurabili. I notebook AWS Glue Studio forniscono un'interfaccia integrata per le sessioni interattive e sono offerti senza costi aggiuntivi. È prevista una durata di fatturazione minima di 1 minuto per ogni sessione interattiva. I notebook AWS Glue Studio forniscono un'interfaccia integrata per sessioni interattive e sono offerti senza costi aggiuntivi
Esperienza della console	Disponibile solo tramite CLI e API	Disponibile tramite console AWS Glue, CLI e API

Migrazione dagli endpoint di sviluppo alle sessioni interattive

Utilizza la seguente lista di controllo per determinare il metodo appropriato per eseguire la migrazione dagli endpoint di sviluppo alle sessioni interattive.

Il tuo script dipende da funzionalità specifiche di AWS Glue 0.9 o 1.0 (ad esempio, HDFS, YARN, ecc.)?

Se la risposta è sì, consulta Migrazione dei processi AWS Glue a AWS Glue versione 3.0 per scoprire come migrare da Glue 0.9 o 1.0 a Glue 3.0 e versioni successive.

Quale metodo utilizzi per accedere all'endpoint di sviluppo?

Se usi questo metodo	Allora completa le seguenti operazioni
Notebook SageMaker, notebook Jupyter o Jupyter Lab	Esegui la migrazione al notebook AWS Glue Studio scaricando i file `.ipynb` su Jupyter e crea un nuovo processo del notebook AWS Glue Studio caricando il file `.ipynb`. In alternativa, puoi usare anche SageMaker Studio e selezionare il kernel AWS Glue.
Notebook Zeppelin	Converti il notebook in un notebook Jupyter manualmente copiando e incollando il codice o automaticamente utilizzando un convertitore di terze parti come ze2nb. Quindi, usa il notebook nel notebook AWS Glue Studio o in SageMaker Studio.
IDE	Consulta Creazione di processi AWS Glue con PyCharm tramite sessioni interattive di AWS Glue o Utilizzo di sessioni interattive con Microsoft Visual Studio Code.
REPL	Installa `aws-glue-session package` in locale, quindi esegui il comando seguente: • Per Python: `jupyter console --kernal glue_pyspark`

Se usi questo metodo	Allora completa le seguenti operazioni
	• Per Scala: `jupyter console --kernal glue_spark`
SSH	Non esiste un'opzione corrispondente per le sessioni interattive. In alternativa, puoi usare un'immagine Docker. Per ulteriori informazioni, consulta Sviluppare utilizzando un'immagine Docker

Le seguenti sezioni forniscono informazioni sull'utilizzo degli endpoint di sviluppo per lo sviluppo di processi in AWS Glue versione 1.0.

Argomenti

Utilizzo di endpoint per lo sviluppo di script

> **Note**
>
> Gli endpoint di sviluppo sono supportati solo per le versioni di AWS Glue precedenti alla 2.0. Per un ambiente interattivo in cui è possibile creare e testare script di ETL, utilizza Notebook su AWS Glue Studio.

AWS Glue può creare un ambiente, noto come endpoint di sviluppoche puoi utilizzare per sviluppare e testare iterativamente gli script ETL (Extract, Transform and Load). Puoi creare, modificare ed eliminare endpoint di sviluppo utilizzando la console AWS Glue o l'API.

Gestione dell'ambiente di sviluppo

Quando crei un endpoint di sviluppo, devi fornire i valori di configurazione per effettuare il provisioning dell'ambiente di sviluppo. Questi valori indicano a AWS Glue come configurare la rete in modo da poter accedere all'endpoint di sviluppo in tutta sicurezza e che l'endpoint possa accedere agli archivi dati.

Quindi, crea un notebook che si connette all'endpoint di sviluppo e utilizza il notebook per creare e testare lo script ETL. Quando sei soddisfatto dei risultati del processo di sviluppo, è possibile creare un processo ETL che esegue il tuo script. Con questo processo è possibile aggiungere funzioni ed eseguire il debug dello script in modo interattivo.

Segui i tutorial in questa sezione per ulteriori informazioni su come utilizzare l'endpoint di sviluppo con i notebook.

Argomenti

Flusso di lavoro degli endpoint di sviluppo

Per utilizzare un endpoint di sviluppo AWS Glue, puoi seguire questo flusso di lavoro.

1. Crea un endpoint di sviluppo utilizzando l'API. Questo endpoint viene lanciato nel tuo cloud privato virtuale (VPC, Virtual Private Cloud) con i gruppi di sicurezza da te definiti.
2. L'API è in grado di eseguire il polling dell'endpoint di sviluppo fino a quando non viene allocato ed è pronto per l'utilizzo. Quando è pronto, connettiti all'endpoint di sviluppo utilizzando uno dei seguenti metodi per creare e testare gli script AWS Glue.
    - Crea un notebook SageMaker nel tuo account. Per ulteriori informazioni su come creare un notebook, consulta the section called “Creazione di codice con notebook AWS Glue Studio”.
    - È possibile aprire una finestra terminale per connettersi direttamente a un endpoint di sviluppo.

- Se disponi dell'edizione Professional di JetBrains PyCharm Python IDE, collegalo a un endpoint di sviluppo e utilizzalo per sviluppare interattivamente. Se inserisci dichiarazioni `pydevd` nello script, PyCharm è in grado di supportare breakpoint remoti.

3. Una volta terminato il debug e il test dell'endpoint di sviluppo, è possibile eliminarlo.

Come funzionano gli endpoint di sviluppo AWS Glue con i notebook SageMaker

Uno dei modi più comuni per accedere agli endpoint di sviluppo è usare Jupyter sui notebook SageMaker. Il notebook Jupyter è un'applicazione web open source ampiamente utilizzata nella visualizzazione, nell'analisi dei dati, nel machine learning, ecc. Un notebook SageMaker AWS Glue offre un'esperienza notebook Jupyter con endpoint di sviluppo AWS Glue. Nel notebook SageMaker AWS Glue, l'ambiente notebook Jupyter è preconfigurato con SparkMagic, un plugin Jupyter open source per inviare i processi Spark a un cluster Spark remoto. Apache Livy è un servizio che consente l'interazione con un cluster Spark remoto tramite una REST API. Nel notebook SageMakerAWS Glue, SparkMagic è configurato per richiamare la REST API su un server Livy in esecuzione su un endpoint di sviluppo AWS Glue.

Il seguente flusso di testo spiega come funziona ogni componente:

AWS GlueNotebook SageMaker: (Jupyter → SparkMagic) → (rete) → endpoint di sviluppo AWS Glue: (Apache Livy → Apache Spark)

Una volta eseguito lo script Spark scritto in ogni paragrafo su un notebook Jupyter, il codice Spark viene inviato al server Livy tramite SparkMagic, quindi un processo Spark denominato "livy-session-N" viene eseguito sul cluster Spark. Questo processo è denominato sessione Livy. Il processo Spark verrà eseguito mentre la sessione del notebook è attiva. Il processo Spark verrà terminato quando il kernel Jupyter viene arrestato dal notebook o quando la sessione è scaduta. Viene avviato un processo Spark per ogni file notebook (.ipynb).

È possibile utilizzare un singolo endpoint di sviluppo AWS Glue con più istanze notebook SageMaker. È possibile creare più file notebook in ogni istanza notebook di SageMaker. Quando apri un file di ciascun notebook ed esegui i paragrafi, viene avviata una sessione Livy per ogni file notebook nel cluster Spark tramite SparkMagic. Ogni sessione Livy corrisponde a un singolo processo Spark.

Funzionamento predefinito per endpoint di sviluppo AWS Glue e notebook SageMaker

I processi Spark vengono eseguiti in base alla configurazione di Spark. Esistono diversi modi per impostare la configurazione di Spark (ad esempio, configurazione cluster Spark, configurazione di SparkMagic, ecc.).

Per impostazione predefinita, Spark alloca le risorse del cluster a una sessione Livy in base alla configurazione del cluster Spark. Nell'endpoint di sviluppo AWS Glue, la configurazione del cluster dipende dal tipo di worker. Ecco una tabella che spiega le configurazioni comuni per tipo di worker.

	Standard	G.1X	G.2X
`spark.driver.memory`	5G	10G	20G
`spark.executor.memory`	5G	10G	20G
`spark.executor.cores`	4	8	16
`spark.dynamicAllocation.enabled`	TRUE	TRUE	TRUE

Il numero massimo di executor Spark viene calcolato automaticamente in base alla combinazione di DPU (o `NumberOfWorkers`) e il tipo di dipendente.

	Standard	G.1X	G.2X
Il numero massimo di executor Spark	`(DPU - 1) * 2 - 1`	`(NumberOfWorkers - 1)`	`(NumberOfWorkers - 1)`

Ad esempio, se l'endpoint di sviluppo ha 10 worker e il tipo di worker è `G.1X`, allora si avranno 9 executor Spark e l'intero cluster avrà 90G di memoria dell'executor, poiché ogni executor avrà 10G di memoria.

Indipendentemente dal tipo di worker specificato, verrà attivata l'allocazione dinamica delle risorse Spark. Se un set di dati è abbastanza grande, Spark potrebbe allocare tutti gli executor a una singola sessione Livy poiché `spark.dynamicAllocation.maxExecutors` non è configurata per impostazione predefinita. Ciò significa che altre sessioni Livy sullo stesso endpoint di sviluppo attenderanno per l'avvio di nuovi executor. Se il set di dati è piccolo, Spark sarà in grado di allocare gli esecutori a più sessioni Livy contemporaneamente.

> **Note**
>
> Per ulteriori informazioni sull'allocazione delle risorse in diversi casi d'uso e su come impostare una configurazione che modifichi il funzionamento, consulta Configurazione avanzata: condivisione degli endpoint di sviluppo tra più utenti.

Aggiunta di un endpoint di sviluppo

Utilizza gli endpoint di sviluppo per lo sviluppo iterativo e per il test di script di estrazione, trasformazione e caricamento (ETL) in AWS Glue. È possibile utilizzare gli endpoint di sviluppo solo tramite la AWS Command Line Interface.

1. In una finestra a riga di comando, immetti un comando simile al seguente.

   ```
   aws glue create-dev-endpoint --endpoint-name "endpoint1" --role-arn
    "arn:aws:iam::account-id:role/role-name" --number-of-nodes "3" --glue-version
    "1.0" --arguments '{"GLUE_PYTHON_VERSION": "3"}' --region "region-name"
   ```

   Questo comando specifica AWS Glue versione 1.0. Poiché questa versione supporta sia Python 2 sia Python 3, puoi utilizzare il parametro `arguments` per indicare la versione di Python desiderata. Se il parametro `glue-version` viene omesso, viene utilizzato AWS Glue versione 0.9. Per ulteriori informazioni sulle versioni di AWS Glue, consulta Glue version job property.

   Per ulteriori informazioni sui parametri della riga di comando aggiuntivi, consulta create-dev-endpoint nel riferimento ai comandi AWS CLI.

2. (Facoltativo) Immetti il comando seguente per controllare lo stato dell'endpoint di sviluppo. Quando lo stato cambia in `READY`, l'endpoint di sviluppo è pronto per l'uso.

   ```
   aws glue get-dev-endpoint --endpoint-name "endpoint1"
   ```

Accesso all'endpoint di sviluppo

Quando crei un endpoint di sviluppo in un cloud privato virtuale (VPC, Virtual Private Cloud), AWS Glue restituisce solo un indirizzo IP privato. Il campo dell'indirizzo IP pubblico non è compilato. Quando crei un endpoint non di sviluppo non VPC, AWS Glue restituisce solo un indirizzo IP pubblico.

Se l'endpoint di sviluppo dispone di un Public address (Indirizzo pubblico), conferma che sia raggiungibile con la chiave privata SSH per l'endpoint di sviluppo, come nell'esempio seguente.

```
ssh -i dev-endpoint-private-key.pem glue@public-address
```

Supponi che l'endpoint di sviluppo disponga di un Private address (Indirizzo privato), che la sottorete VPC sia instradabile dalla rete Internet pubblica e che i relativi gruppi di sicurezza consentano l'accesso in entrata dal client. In questo caso, segui questa procedura per collegare un indirizzo IP elastico a un endpoint di sviluppo per consentire l'accesso da Internet.

> **Note**
>
> Se desideri utilizzare indirizzi IP elastici, la sottorete utilizzata richiede un gateway Internet associato tramite la tabella di routing.

Per accedere a un endpoint di sviluppo collegando un indirizzo IP elastico

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, scegli Dev endpoints (Endpoint di sviluppo) e passa alla pagina dei dettagli dell'endpoint di sviluppo. Registra il Private address (Indirizzo privato) da utilizzare nella fase successiva.
3. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
4. Nel pannello di navigazione, in Network & Security (Rete e sicurezza), scegli Network Interfaces (Interfacce di rete).
5. Cerca l'indirizzo Private DNS (IPv4) (DNS privato - IPv4) che corrisponde al valore di Private address (Indirizzo privato) nella pagina dei dettagli dell'endpoint di sviluppo della console AWS Glue.

Potrebbe essere necessario modificare quali colonne vengono visualizzate nella console Amazon EC2. Prendi nota del Network interface ID (ID interfaccia di rete, ENI) per questo indirizzo (ad esempio, `eni-12345678`).

6. Nella console Amazon EC2, in Network & Security (Rete e sicurezza), scegli Elastic IPs (IP elastici).
7. Scegli Allocate new address (Alloca nuovo indirizzo), quindi seleziona Allocate (Alloca) per allocare un nuovo indirizzo IP elastico.
8. Nella pagina Elastic IPs (IP elastici), scegli il nuovo Elastic IP (IP elastico) assegnato. Quindi scegli Actions (Operazioni), Associate address (Associa indirizzo).
9. Nella pagina Associate address (Associa indirizzo), procedi come segue:
   - Per il Resource type (Tipo di risorsa), scegli Network interface (Interfaccia di rete).
   - Nella casella Network interface (Interfaccia di rete) , immetti il Network interface ID (ID interfaccia di rete, ENI) dell'indirizzo privato.
   - Selezionare Associate (Associa).
10. Conferma che l'indirizzo IP elastico appena associato sia raggiungibile con la chiave privata SSH associata all'endpoint di sviluppo, come nell'esempio seguente.

```
ssh -i dev-endpoint-private-key.pem glue@elastic-ip
```

Per ulteriori informazioni sull'utilizzo di un bastion host per ottenere l'accesso SSH all'indirizzo privato dell'endpoint di sviluppo, consulta il post del blog sulla sicurezza AWS Securely Connect to Linux Instances Running in a Private Amazon VPC.

Tutorial: configurare un notebook Jupyter in JupyterLab per testare ed eseguire il debug degli script ETL

In questo tutorial, collegherai un notebook Jupyter in JupyterLab in esecuzione sul computer locale a un endpoint di sviluppo. Questa operazione serve per poter eseguire, eseguire il debug e testare in modo interattivo l'estrazione, la trasformazione e il caricamento di script (ETL) AWS Glue prima di distribuirli. Questo tutorial usa l'inoltro della porta Secure Shell (SSH) per connettere il computer locale a un endpoint di sviluppo AWS Glue. Per ulteriori informazioni, consulta Inoltro della porta in Wikipedia.

Fase 1: installare JupyterLab e Sparkmagic

Puoi installare JupyterLab tramite `conda` o `pip`. `conda` è un sistema open source di gestione di pacchetti e di ambienti eseguibile in Windows, macOS e Linux. `pip`è il programma di installazione dei pacchetti per Python.

Se esegui l'installazione su macOS, devi avere Xcode installato prima di poter installare Sparkmagic.

1. Installa JupyterLab, Sparkmagic e le relative estensioni.

```
$ conda install -c conda-forge jupyterlab
$ pip install sparkmagic
$ jupyter nbextension enable --py --sys-prefix widgetsnbextension
$ jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

2. Controlla la directory `sparkmagic` da `Location`.

```
$ pip show sparkmagic | grep Location
Location: /Users/username/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-
packages
```

3. Cambia la tua directory con quella restituita per `Location` e installa i kernel per Scala e PySpark.

```
$ cd /Users/username/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-packages
$ jupyter-kernelspec install sparkmagic/kernels/sparkkernel
$ jupyter-kernelspec install sparkmagic/kernels/pysparkkernel
```

4. Scarica un esempio di file `config`.

```
$ curl -o ~/.sparkmagic/config.json https://raw.githubusercontent.com/jupyter-
incubator/sparkmagic/master/sparkmagic/example_config.json
```

   In questo file di configurazione, è possibile configurare parametri relativi a Spark come `driverMemory` e `executorCores`.

Fase 2: avviare JupyterLab

Quando avvii JupyterLab, il browser Web predefinito viene aperto automaticamente e viene visualizzato l'URL `http://localhost:8888/lab/workspaces/{workspace_name}`.

```
$ jupyter lab
```

Fase 3: avviare l'inoltro alla porta SSH per la connessione all'endpoint di sviluppo

Usa quindi l'inoltro porta locale SSH per inoltrare una porta locale (qui, 8998) alla destinazione remota definita da AWS Glue (`169.254.76.1:8998`).

1. Apri una finestra separata del terminale che ti consente di accedere a SSH. Su Microsoft Windows, puoi utilizzare la shell BASH fornita da Git for Windows o installare Cygwin.
2. Esegui il comando SSH seguente, modificato come segue:

   - Sostituisci *`private-key-file-path`* con un percorso al file `.pem` contenente la chiave privata corrispondente alla chiave pubblica utilizzata per creare l'endpoint di sviluppo.
   - Se stai inoltrando una porta diversa da 8998, sostituisci 8998 con il numero di porta che stai effettivamente usando in locale. L'indirizzo `169.254.76.1:8998` è la porta remota e non è modificata da te.
   - Sostituisci *`dev-endpoint-public-dns`* con l'indirizzo DNS pubblico del tuo endpoint di sviluppo. Per trovare questo indirizzo, passa all'endpoint di sviluppo nella console AWS Glue, scegli il nome e copia il valore di Public address (Indirizzo pubblico) elencato nella pagina Endpoint details (Dettagli endpoint).

   ```
   ssh -i private-key-file-path -NTL 8998:169.254.76.1:8998 glue@dev-endpoint-public-dns
   ```

   È probabile che vedrai un messaggio di avviso, come il seguente:

   ```
   The authenticity of host 'ec2-xx-xxx-xxx-xx.us-west-2.compute.amazonaws.com
    (xx.xxx.xxx.xx)'
   can't be established.  ECDSA key fingerprint is SHA256:4e97875Brt+1wKzRko
   +JflSnp21X7aTP3BcFnHYLEts.
   Are you sure you want to continue connecting (yes/no)?
   ```

   Inserisci **`yes`** e lascia aperta la finestra del terminale mentre usi JupyterLab.
3. Verifica che l'inoltro porta SSH funzioni correttamente con l'endpoint di sviluppo.

   ```
   $ curl localhost:8998/sessions
   ```

```
{"from":0,"total":0,"sessions":[]}
```

Fase 4: eseguire un semplice frammento di script in un paragrafo del notebook

Ora il notebook in JupyterLab dovrebbe funzionare con il tuo endpoint di sviluppo. Digita il frammento di script seguente nel notebook ed eseguilo.

1. Verifica che Spark sia in esecuzione correttamente. Il comando seguente indica a Spark di calcolare 1 e quindi stampare il valore.

   ```
   spark.sql("select 1").show()
   ```

2. Controlla se l'integrazione AWS Glue Data Catalog sta funzionando. Il comando seguente elenca le tabelle nel catalogo dati.

   ```
   spark.sql("show tables").show()
   ```

3. Verifica che un frammento di script semplice che utilizza le librerie AWS Glue funzioni.

   Lo script seguente utilizza i metadati della tabella `persons_json` in AWS Glue Data Catalog per creare un`DynamicFrame`dai dati di esempio. Quindi, verranno stampati il conteggio item e lo schema di questi dati.

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create a Glue context
glueContext = GlueContext(SparkContext.getOrCreate())

# Create a DynamicFrame using the 'persons_json' table
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")

# Print out information about *this* data
print("Count:  ", persons_DyF.count())
persons_DyF.printSchema()
```

L'output dello script è il seguente.

```
Count:  1961
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

Risoluzione dei problemi

- Durante l'installazione di JupyterLab, se il computer è protetto da un proxy aziendale o da un firewall, potrebbero verificarsi errori HTTP e SSL dovuti a profili di sicurezza personalizzati gestiti dai reparti IT aziendali.

  Di seguito è riportato un esempio di errore tipico che si verifica quando `conda` non è in grado di connettersi ai propri repository:

```
CondaHTTPError: HTTP 000 CONNECTION FAILED for url <https://repo.anaconda.com/pkgs/
main/win-64/current_repodata.json>
```

Questo errore potrebbe verificarsi perché la tua azienda può bloccare le connessioni a repository ampiamente utilizzati nelle community Python e JavaScript. Per ulteriori informazioni, consulta Installation problems sul sito Web di JupyterLab.

- Se si verifica un errore di connessione rifiutata quando provi a connetterti all'endpoint di sviluppo, potresti utilizzare un endpoint di sviluppo obsoleto. Prova a creare un nuovo endpoint di sviluppo e a riconnetterti.

Tutorial: utilizzo di un notebook SageMaker con l'endpoint di sviluppo

In AWS Glue, puoi creare un endpoint di sviluppo e quindi creare un notebook SageMaker per sviluppare script ETL e Machine Learning. Un notebook SageMaker è un'istanza di calcolo di Machine Learning completamente gestita che esegue l'applicazione Jupyter Notebook.

1. Nella console AWS Glue seleziona Dev endpoints (Endpoint di sviluppo) per passare all'elenco degli endpoint di sviluppo.
2. Seleziona la casella di controllo accanto al nome di un endpoint di sviluppo che desideri utilizzare e nel menu Action (Azione), scegli Create SageMaker notebook (Creazione di un notebook SageMaker).
3. Compilare la pagina Create and configure a notebook (Crea e configura un notebook) come segue:

   a. Immettere il nome di un notebook.

   b. In Attach to development endpoint (Collega a endpoint di sviluppo), verificare l'endpoint di sviluppo.

   c. Creare o scegliere un ruolo AWS Identity and Access Management (IAM).

      Si consiglia di creare un ruolo. Se si utilizza un ruolo esistente, assicurarsi di avere le autorizzazioni necessarie. Per ulteriori informazioni, consulta the section called “Fase 6: creare una policy IAM per i notebook SageMaker”.

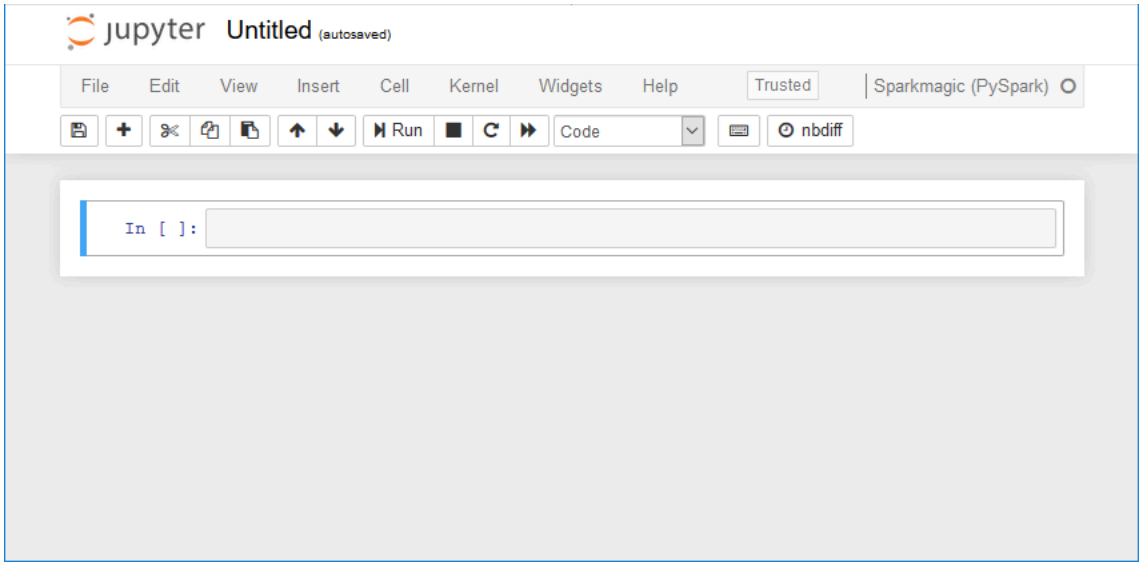
   d. (Facoltativo) Scegliere un VPC, una sottorete e uno o più gruppi di sicurezza.

   e. (Facoltativo) Scegliere una chiave di crittografia AWS Key Management Service.

   f. (Facoltativo) Aggiungere i tag per l'istanza del notebook.

4. Seleziona Create Notebook (Crea notebook). Sulla pagina Notebooks (Notebook), scegli l'icona di aggiornamento in alto a destra e continua fino a quando la finestra Status (Stato) non mostra `Ready`.
5. Selezionare la casella di controllo accanto al nuovo nome del notebook, quindi scegliere Open notebook (Apri notebook).
6. Creare un nuovo notebook: nella pagina jupyter scegliere New (Nuovo), quindi scegliere Sparkmagic (PySpark).

   La schermata dovrebbe essere simile alla seguente:



7. (Facoltativo) Nella parte superiore della pagina, scegliere Untitled (Senza titolo) e assegnare un nome al notebook.
8. Per avviare un'applicazione Spark, immettere il seguente comando nel notebook e quindi nella barra degli strumenti scegliere Run (Esegui).

```
spark
```

   Dopo una breve attesa, viene visualizzata la seguente risposta:

```
In [1]: spark
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1576209965005_0001	pyspark	idle	Link	Link	✓

```
SparkSession available as 'spark'.
<pyspark.sql.session.SparkSession object at 0x7f3d54913550>
```

9. Creare un frame dinamico ed eseguirvi una query: copiare, incollare ed eseguire il codice seguente, che restituisce il conteggio e lo schema della tabella `persons_json`.

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *
glueContext = GlueContext(SparkContext.getOrCreate())
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")
print ("Count:  ", persons_DyF.count())
persons_DyF.printSchema()
```

Tutorial: utilizzo di una shell (interprete di comandi) REPL con l'endpoint di sviluppo

In AWS Glue puoi creare un endpoint di sviluppo e quindi richiamare una shell (interprete di comandi) REPL (Read–Evaluate–Print Loop) per eseguire il codice PySpark in modo incrementale per poter eseguire il debug interattivo degli script ETL prima di distribuirli.

Per utilizzare una REPL su un endpoint di sviluppo, è necessario disporre dell'autorizzazione SSH all'endpoint.

1. Sul tuo computer locale, apri una finestra terminale che possa eseguire comandi SSH e incolla il comando SSH modificato. Esegui il comando .

   Se hai accettato AWS Glue versione 1.0 con Python 3 per l'endpoint di sviluppo, l'aspetto dell'output sarà simile al seguente:

```
Python 3.6.8 (default, Aug  2 2019, 17:42:44)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/share/aws/glue/etl/jars/glue-assembly.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/spark/jars/slf4j-log4j12-1.7.16.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
 setLogLevel(newLevel).
```

```
2019-09-23 22:12:23,071 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading
 libraries under SPARK_HOME.
2019-09-23 22:12:26,562 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same name resource file:/usr/lib/spark/python/lib/pyspark.zip added multiple
 times to distributed cache
2019-09-23 22:12:26,580 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same path resource file:///usr/share/aws/glue/etl/python/PyGlue.zip added
 multiple times to distributed cache.
2019-09-23 22:12:26,581 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same path resource file:///usr/lib/spark/python/lib/py4j-src.zip added multiple
 times to distributed cache.
2019-09-23 22:12:26,581 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same path resource file:///usr/share/aws/glue/libs/pyspark.zip added multiple
 times to distributed cache.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.3
      /_/

Using Python version 3.6.8 (default, Aug  2 2019 17:42:44)
SparkSession available as 'spark'.
>>>
```

2. Verifica che la REPL shell funzioni correttamente digitando l'istruzione `print(spark.version)`. Finché visualizza la versione Spark, la REPL è pronta per l'uso.
3. Ora puoi provare a eseguire il seguente script semplice, riga per riga, nella shell:

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *
glueContext = GlueContext(SparkContext.getOrCreate())
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")
print ("Count:  ", persons_DyF.count())
persons_DyF.printSchema()
```

Tutorial: configurazione di PyCharm Professional con un endpoint di sviluppo

Questo tutorial mostra come connettere l'IDE Python PyCharm Professional in esecuzione nel computer locale a un endpoint di sviluppo in modo da poter eseguire, sottoporre a debug e testare in modo interattivo script ETL (Extract, Transfer and Load, estrazione, trasferimento e caricamento) AWS Glue prima di distribuirli. Le istruzioni e le schermate nel tutorial sono basate su PyCharm Professional versione 2019.3.

Per connettersi a un endpoint di sviluppo in modo interattivo, è necessario che PyCharm Professional sia installato. Non puoi farlo usando l'edizione gratuita.

> **Note**
>
> Il tutorial utilizza Amazon S3 come origine dati. Se invece desideri utilizzare un'origine dati JDBC, devi eseguire l'endpoint di sviluppo in un cloud privato virtuale (VPC). Per connetterti con SSH a un endpoint di sviluppo in un VPC, devi creare un tunnel SSH. Questo tutorial non include le istruzioni per la creazione di un tunnel SSH. Per informazioni sull'utilizzo di SSH per connetterti a un endpoint di sviluppo in un VPC, consulta Securely Connect to Linux Instances Running in a Private Amazon VPC nel blog AWS sulla sicurezza.

Argomenti

Connettere PyCharm Professional a un endpoint di sviluppo

1. Crea un nuovo progetto in Python puro all'interno di PyCharm denominato `legislators`.
2. Crea un file denominato `get_person_schema.py` nel progetto con il seguente contenuto:

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext


def main():
```

```
    # Create a Glue context
    glueContext = GlueContext(SparkContext.getOrCreate())

    # Create a DynamicFrame using the 'persons_json' table
    persons_DyF =
 glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")

    # Print out information about this data
    print("Count:  ", persons_DyF.count())
    persons_DyF.printSchema()


if __name__ == "__main__":
    main()
```

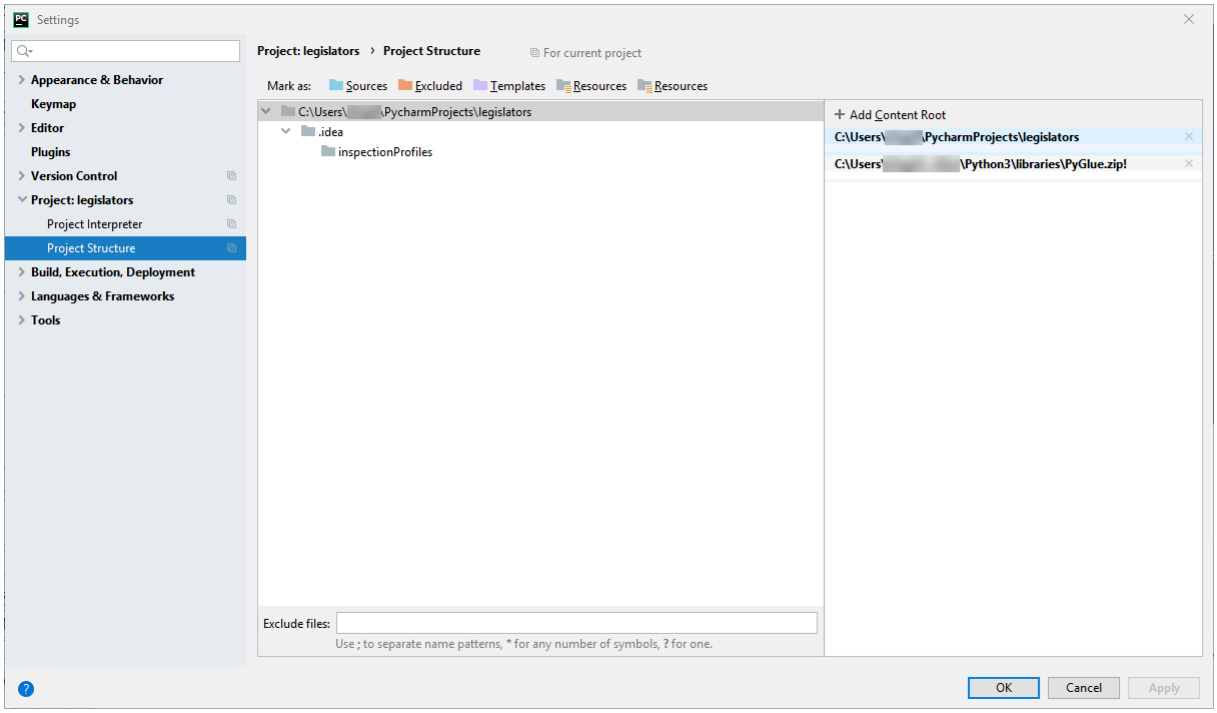
3. Completa una delle seguenti operazioni:

   - Per AWS Glue versione 0.9, scarica il file della libreria Python AWS Glue, `PyGlue.zip`, da `https://s3.amazonaws.com/aws-glue-jes-prod-us-east-1-assets/etl/python/PyGlue.zip` a un percorso appropriato nel computer locale.
   - Per AWS Glue versione 1.0, scarica il file della libreria Python AWS Glue, `PyGlue.zip`, da `https://s3.amazonaws.com/aws-glue-jes-prod-us-east-1-assets/etl-1.0/python/PyGlue.zip` a un percorso appropriato nel computer locale.

4. Aggiungi `PyGlue.zip` come content root per il tuo progetto in PyCharm:

   - In PyCharm, scegli File (File), Settings (Impostazioni) per aprire la finestra di dialogo Settings (Impostazioni). Puoi anche premere `Ctrl+Alt+S`.
   - Espandi il progetto `legislators` e scegli Project Structure (Struttura del progetto). Quindi nel riquadro di destra, scegli + Add Content Root (+ Aggiungi Content Root).
   - Seleziona il percorso in cui hai salvato `PyGlue.zip`, selezionalo, quindi scegli Apply (Applica).

   La schermata Settings (Impostazioni) deve avere un aspetto simile al seguente:



Lascia aperta la finestra di dialogo Settings (Impostazioni) dopo aver scelto Apply (Applica).

5. Configura le opzioni di distribuzione per caricare gli script locali sull'endpoint di sviluppo utilizzando SFTP (questa caratteristica è disponibile solo in PyCharm Professional):

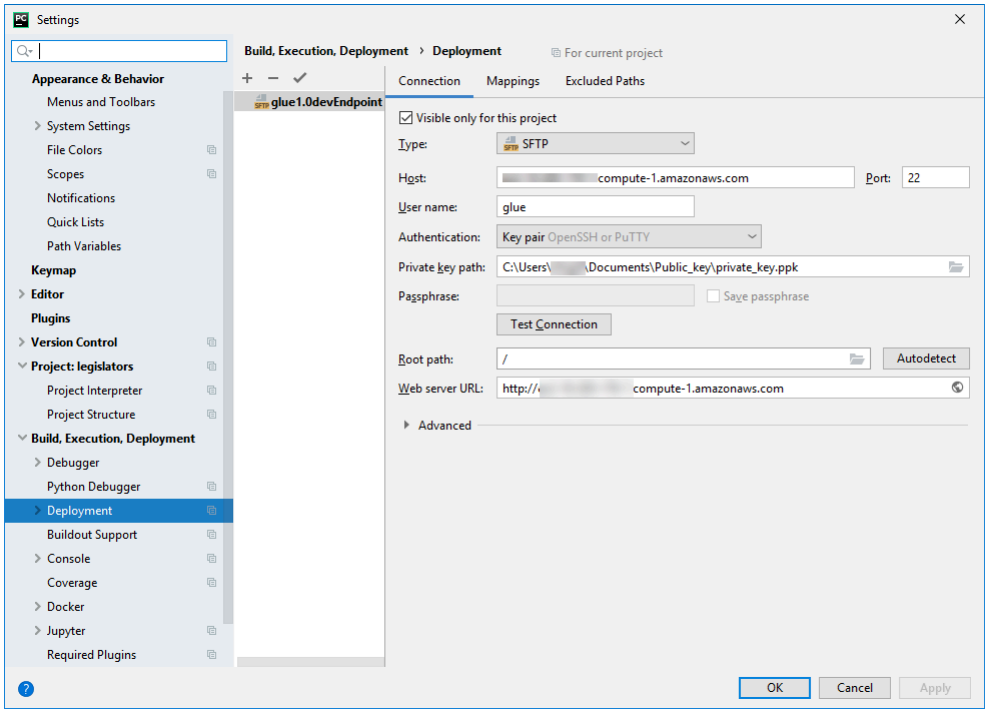
   - Nella finestra di dialogo Settings (Impostazioni), espandi la sezione Build, Execution, Deployment (Creazione, esecuzione e distribuzione). Scegli la sottosezione Deployment (Distribuzione).
   - Scegli l'icona + in alto nel riquadro centrale per aggiungere un nuovo server. Imposta il Type (Tipo) su `SFTP` e assegna un nome.
   - Imposta SFTP host (Host SFTP) sull'indirizzo pubblico dell'endpoint di sviluppo, come indicato nella pagina dei dettagli. Scegli il nome dell'endpoint di sviluppo nella console AWS Glue per visualizzare la pagina dei dettagli. Per un endpoint di sviluppo in esecuzione in un VPC, imposta l'host SFTP sull'indirizzo host e la porta locale del tunnel SSH sull'endpoint di sviluppo.
   - Imposta lo User name (Nome utente) su `glue`.
   - Imposta l'Auth type (Tipo di autenticazione) per la Key pair (OpenSSH or Putty) (Coppia di chiavi, OpenSSH o Putty). Imposta il Private key file (File della chiave privata) cercando il percorso in cui si trova il file della chiave privata dell'endpoint di sviluppo. Da notare che PyCharm supporta i tipi di chiavi DSA, RSA e ECDSA OpenSSH e non accetta le chiavi in

formato privato Putty. È possibile utilizzare una versione aggiornata di `ssh-keygen` per generare un tipo di coppia di chiavi accettata da PyCharm, utilizzando la sintassi che segue:

```
ssh-keygen -t rsa -f <key_file_name> -C "<your_email_address>"
```

- Scegli Test connection (Test connessione) e consenti il test della connessione. Se la connessione viene stabilita, scegli Apply (Applica).

La schermata Settings (Impostazioni) ora deve avere un aspetto simile al seguente:

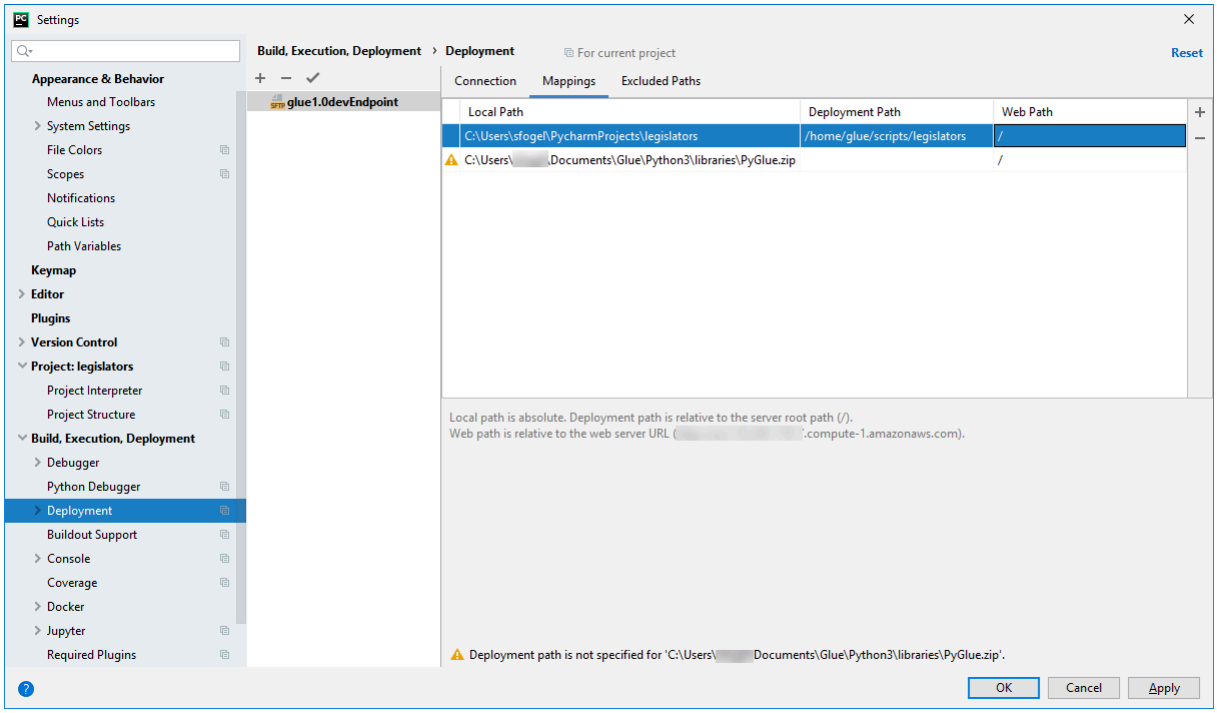


Lascia nuovamente aperta la finestra di dialogo Settings (Impostazioni) dopo aver scelto Apply (Applica).

6. Imposta la directory locale su una directory remota per la distribuzione:

   - Nel riquadro a destra della pagina Deployment (Distribuzione), scegli la scheda centrale nella parte superiore, contrassegnata dall'etichetta Mappings (Mappe).
   - Nella colonna Deployment Path (Percorso di distribuzione), immetti un percorso sotto `/home/glue/scripts/` per la distribuzione del percorso del progetto. Ad esempio: `/home/glue/scripts/legislators`.
   - Seleziona Apply (Applica).

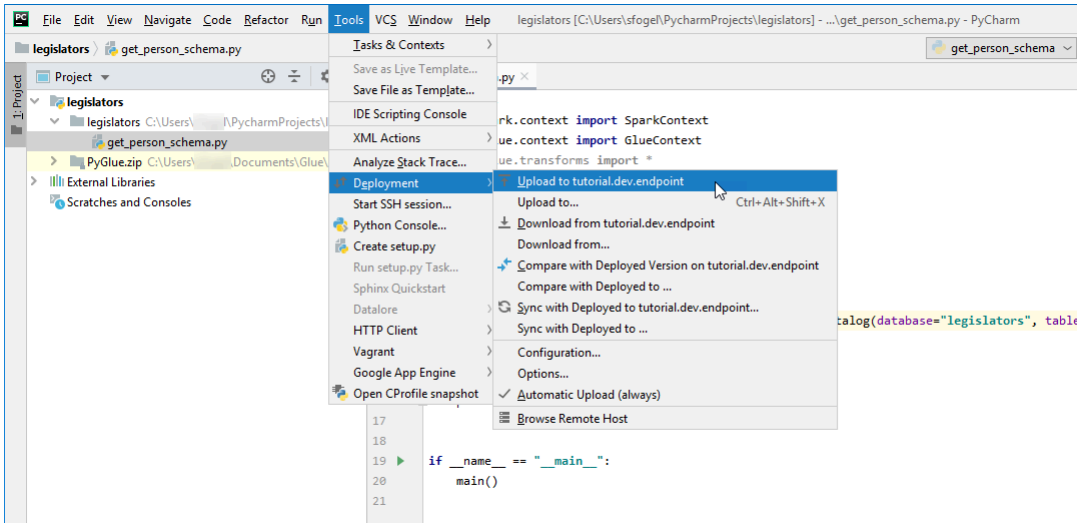
La schermata Settings (Impostazioni) ora deve avere un aspetto simile al seguente:



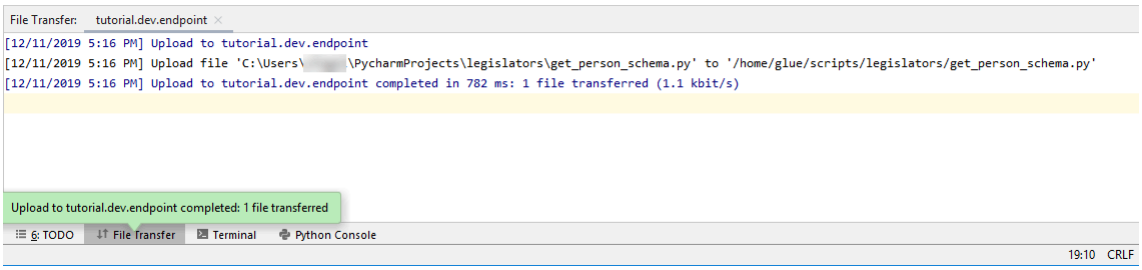
Scegli OK per chiudere la finestra di dialogo Settings (Impostazioni).

Distribuzione dello script nell'endpoint di sviluppo

1. Scegli Tools (Strumenti), Deployment (Distribuzione), quindi scegli il nome con cui hai configurato l'endpoint di sviluppo, come illustrato nell'immagine seguente:



Dopo che lo script è stato distribuito, la parte inferiore della schermata deve avere un aspetto simile al seguente:



2. Nella barra dei menu scegli Tools (Strumenti), Deployment (Distribuzione), Automatic Upload (always) (Caricamento automatico (sempre)). Accertati che venga visualizzato un segno di spunta accanto a Automatic Upload (always) (Caricamento automatico (sempre)).

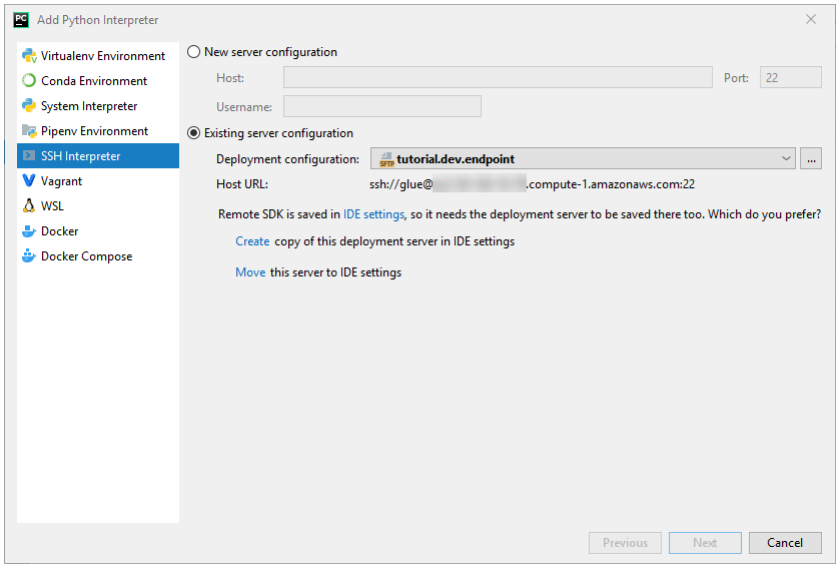
   Quando questa opzione è abilitata, PyCharm carica automaticamente i file modificati nell'endpoint di sviluppo.

Configurazione di un interprete remoto

Configura PyCharm per utilizzare l'interprete Python nell'endpoint di sviluppo.

1. Nel menu File, scegli Settings (Impostazioni).
2. Espandi i legislators (legislatori) del progetto e scegli Project Interpreter (Interprete di progetto).
3. Scegli l'icona a forma di ingranaggio accanto all'elenco Project Interpreter (Interprete di progetto) quindi scegli Add (Aggiungi).
4. Nella finestra di dialogo Add Python Interpreter (Aggiungi interprete Python) nel riquadro di sinistra, scegli SSH Interpreter (Interprete SSH).
5. Scegli Existing server configuration (Configurazione server esistente) e nell'elenco Deployment configuration (Configurazione distribuzione) scegli la configurazione.

   La schermata ora deve avere un aspetto simile all'immagine seguente.



6. Scegli "Move this server to IDE settings (Sposta il server nelle impostazioni IDE", quindi seleziona Next (Successivo).
7. Nel campo Interpreter (Interprete) cambia il percorso in `/usr/bin/gluepython` se stai usando Python 2 o in `/usr/bin/gluepython3` se stai usando Python 3. Quindi scegli Finish (Fine).

Esecuzione dello script sull'endpoint di sviluppo

Per eseguire lo script:

- Nel riquadro di sinistra, fai clic con il pulsante destro del mouse sul nome del file e scegli Run '***<nomefile>***'.

  Dopo una serie di messaggi, l'output finale mostra il conteggio e lo schema.

```
Count:  1961
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
```

```
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string


Process finished with exit code 0
```

Ora hai effettuato la configurazione per eseguire il debug dello script in remoto sul tuo endpoint di sviluppo.

Configurazione avanzata: condivisione degli endpoint di sviluppo tra più utenti

Questa sezione spiega come sfruttare gli endpoint di sviluppo con i notebook SageMaker in casi d'uso tipici per condividere gli endpoint di sviluppo tra più utenti.

Configurazione tenancy singola

Nei casi d'uso a tenant singolo, per semplificare l'esperienza degli sviluppatori ed evitare conflitti per le risorse, è consigliabile che ogni sviluppatore utilizzi il proprio endpoint di sviluppo per il progetto su cui sta lavorando. Ciò semplifica anche le decisioni relative al tipo di worker e al conteggio DPU, lasciandole a discrezione dello sviluppatore e del progetto su cui sta lavorando.

Non dovrai occuparti dell'allocazione delle risorse a meno che non vengano eseguiti simultaneamente più file notebook. Se esegui il codice in più file notebook contemporaneamente, verranno avviate contemporaneamente più sessioni Livy. Per separare le configurazioni del cluster

Spark al fine di eseguire più sessioni Livy contemporaneamente, puoi seguire i passaggi introdotti nei casi d'uso multi-tenant.

Ad esempio, se l'endpoint di sviluppo ha 10 worker e il tipo di worker è `G.1X`, allora si avranno 9 executor Spark e l'intero cluster avrà 90G di memoria dell'executor, poiché ogni executor avrà 10G di memoria.

Indipendentemente dal tipo di worker specificato, verrà attivata l'allocazione dinamica delle risorse Spark. Se un set di dati è abbastanza grande, Spark potrebbe allocare tutti gli executor a una singola sessione Livy poiché `spark.dynamicAllocation.maxExecutors` non è configurata per impostazione predefinita. Ciò significa che altre sessioni Livy sullo stesso endpoint di sviluppo attenderanno per l'avvio di nuovi executor. Se il set di dati è piccolo, Spark sarà in grado di allocare gli esecutori a più sessioni Livy contemporaneamente.

> **Note**
>
> Per ulteriori informazioni sull'allocazione delle risorse in diversi casi d'uso e su come impostare una configurazione che modifichi il funzionamento, consulta Configurazione avanzata: condivisione degli endpoint di sviluppo tra più utenti.

Configurazione tenancy multipla

> **Note**
>
> Tieni presente che gli endpoint di sviluppo hanno lo scopo di emulare l'ambiente ETL AWS Glue come ambiente a tenant singolo. Sebbene l'utilizzo di multi-tenant sia possibile, si tratta di un caso d'uso avanzato e alla maggior parte degli utenti si consiglia di mantenere un modello di tenancy singolo per ogni endpoint di sviluppo.

Nei casi d'uso multi-tenant, potresti doverti occupare dell'allocazione delle risorse. Il fattore chiave è il numero di utenti che utilizzano contemporaneamente un notebook Jupyter. Se il tuo team lavora in un flusso di lavoro "follow-the-sun" con un solo utente Jupyter per ogni fuso orario, il numero di utenti simultanei è uno solo, quindi non dovrai preoccuparti dell'allocazione delle risorse. Tuttavia, se il notebook è condiviso tra più utenti e ogni utente invia il codice ad hoc, sarà necessario considerare i seguenti punti.

Per suddividere le risorse del cluster Spark tra più utenti, è possibile utilizzare le configurazioni di SparkMagic. È possibile configurare SparkMagic in due diversi modi.

(A) Utilizzo della direttiva %%configure -f

Se vuoi modificare la configurazione per sessione Livy dal notebook, puoi eseguire la direttiva `%%configure -f` sul paragrafo di notebook.

Ad esempio, se vuoi eseguire l'applicazione Spark su 5 executor, puoi eseguire il comando seguente nel paragrafo di notebook.

```
%%configure -f
{"numExecutors":5}
```

Vedrai quindi solo 5 executor in esecuzione per il processo sull'interfaccia utente di Spark.

Si consiglia di limitare il numero massimo di executor per l'allocazione dinamica delle risorse.

```
%%configure -f
{"conf":{"spark.dynamicAllocation.maxExecutors":"5"}}
```

(B) Modifica del file di configurazione di SparkMagic

SparkMagic funziona in base all'API Livy. SparkMagic crea sessioni Livy con configurazioni come `driverMemory`, `driverCores`, `executorMemory`, `executorCores`, `numExecutors`, `conf`, e così via. Questi sono i fattori chiave che determinano la quantità di risorse utilizzate dall'intero cluster Spark. SparkMagic consente di fornire un file di configurazione per specificare i parametri che vengono inviati a Livy. Puoi vedere un file di configurazione di esempio in questo repository Github.

Se vuoi modificare la configurazione di tutte le sessioni Livy da un notebook, puoi modificare `/home/ec2-user/.sparkmagic/config.json` per aggiungere `session_config`.

Per modificare il file di configurazione in un'istanza notebook di SageMaker, segui la seguente procedura.

1. Apri un notebook SageMaker.
2. Apri il kernel del terminale.
3. Esegui i comandi seguenti:

```
sh-4.2$ cd .sparkmagic
```

```
sh-4.2$ ls
config.json logs
sh-4.2$ sudo vim config.json
```

Ad esempio, puoi aggiungere queste righe a `/home/ec2-user/.sparkmagic/config.json` e riavviare il kernel Jupyter dal notebook.

```
  "session_configs": {
    "conf": {
      "spark.dynamicAllocation.maxExecutors":"5"
    }
  },
```

Linee guida e best practice

Per evitare questo tipo di conflitto di risorse, puoi utilizzare alcuni approcci di base come:

- Avere un cluster Spark più grande aumentando il `NumberOfWorkers` (dimensionamento orizzontale) e aggiornando il `workerType` (dimensionamento verticale)
- Allocare di meno risorse per utente (meno risorse per sessione Livy)

Il tuo approccio dipenderà dal caso d'uso. Se disponi di un endpoint di sviluppo più grande e non vi è una quantità enorme di dati, la possibilità di un conflitto di risorse diminuirà significativamente perché Spark può allocare risorse in base a una strategia di allocazione dinamica.

Come descritto sopra, il numero massimo di executor Spark viene calcolato automaticamente in base alla combinazione di DPU (o `NumberOfWorkers`) e il tipo di worker. Ogni applicazione Spark avvia un driver e più executor. Per il calcolo è necessario il `NumberOfWorkers` = `NumberOfExecutors` + 1. La matrice seguente illustra la capacità necessaria nell'endpoint di sviluppo in base al numero di utenti simultanei.

Numero di utenti simultanei del notebook	Numero di executor Spark che vuoi allocare per utente	Numero totale di worker per l'endpoint di sviluppo
3	5	18
10	5	60

Numero di utenti simultanei del notebook	Numero di executor Spark che vuoi allocare per utente	Numero totale di worker per l'endpoint di sviluppo
50	5	300

Se vuoi allocare meno risorse per utente, `spark.dynamicAllocation.maxExecutors` (o `numExecutors`) è il parametro più semplice da configurare come parametro di sessione Livy. Impostando la seguente configurazione in `/home/ec2-user/.sparkmagic/config.json`, SparkMagic assegnerà un massimo di 5 executor per sessione Livy. Questo aiuterà a separare le risorse per sessione Livy.

```
"session_configs": {
    "conf": {
      "spark.dynamicAllocation.maxExecutors":"5"
    }
  },
```

Supponiamo che ci sia un endpoint di sviluppo con 18 worker (G.1X) e 3 utenti del notebook contemporaneamente. Se la configurazione della sessione ha `spark.dynamicAllocation.maxExecutors=5`, ogni utente può utilizzare 1 driver e 5 executor. Anche eseguendo più paragrafi di notebook simultaneamente, non si verificheranno conflitti di risorse.

Pro e contro

Con questa configurazione di sessione `"spark.dynamicAllocation.maxExecutors":"5"`, potrai evitare errori di conflitto di risorse e non sarà necessario attendere l'allocazione delle risorse in caso di accessi utente simultanei. Tuttavia, anche quando ci sono molte risorse gratuite (ad esempio, non ci sono altri utenti simultanei), Spark non può assegnare più di 5 executor per la sessione Livy.

Altre note

È buona prassi interrompere il kernel Jupyter quando si smette di usare un notebook. Ciò consentirà di liberare risorse che altri utenti del notebook potranno utilizzare immediatamente, senza dover attendere la scadenza del kernel (spegnimento automatico).

Problemi comuni

Anche quando si seguono le linee guida, è possibile che si verifichino alcuni problemi.

Sessione non trovata

Se provi a eseguire un paragrafo del notebook anche se la sessione Livy è già stata terminata, verrà visualizzato il messaggio seguente. Per attivare la sessione Livy, devi riavviare il kernel Jupyter scegliendo Kernel >Restart (Riavvia) nel menu Jupyter, quindi eseguire nuovamente il paragrafo del notebook.

```
An error was encountered:
Invalid status code '404' from http://localhost:8998/sessions/13 with error payload:
 "Session '13' not found."
```

Risorse YARN insufficienti

Se provi a eseguire un paragrafo del notebook anche se il cluster Spark non dispone di risorse sufficienti per avviare una nuova sessione Livy, verrà visualizzato il messaggio seguente. Seguendo le linee guida è spesso possibile evitare questo problema, tuttavia potrebbe verificarsi. Per risolvere il problema, puoi verificare se sono presenti sessioni Livy attive non necessarie. In caso affermativo, sarà necessario terminarle per liberare le risorse del cluster. Per ulteriori informazioni, consulta la prossima sezione.

```
Warning: The Spark session does not have enough YARN resources to start.
The code failed because of a fatal error:
    Session 16 did not start up in 60 seconds..

Some things to try:
a) Make sure Spark has enough available resources for Jupyter to create a Spark
 context.
b) Contact your Jupyter administrator to make sure the Spark magics library is
 configured correctly.
c) Restart the kernel.
```

Monitoraggio e debug

In questa sezione vengono descritte le tecniche per il monitoraggio delle risorse e delle sessioni.

Monitoraggio e debug dell'allocazione delle risorse nel cluster

È possibile visualizzare l'interfaccia utente di Spark per monitorare il numero di risorse allocate per ogni sessione Livy e quali sono le configurazioni Spark effettive sul processo. Per attivare l'interfaccia utente di Spark, consulta Abilitazione dell'interfaccia utente Web di Apache Spark per endpoint di sviluppo.

(Facoltativo) Se è necessaria una visualizzazione in tempo reale dell'interfaccia utente di Spark, puoi configurare un tunnel SSH sul server di cronologia Spark in esecuzione sul cluster Spark.

```
ssh -i <private-key.pem> -N -L 8157:<development endpoint public address>:18080
 glue@<development endpoint public address>
```

Apri http://localhost:8157 nel browser per visualizzare l'interfaccia utente di Spark in locale.

Sessioni libere Livy non necessarie

Consulta queste procedure per arrestare le sessioni Livy non necessarie da un notebook o da un cluster Spark.

(a). Terminare le sessioni Livy da un notebook

Puoi chiudere il kernel su un notebook Jupyter per terminare sessioni Livy non necessarie.

(b). Terminare le sessioni Livy da un cluster Spark

Se sono ancora in esecuzione sessioni Livy non necessarie, puoi arrestare le sessioni Livy nel cluster Spark.

Come prerequisito per eseguire questa procedura, è necessario configurare la chiave pubblica SSH per l'endpoint di sviluppo.

Per accedere al cluster Spark, esegui il comando seguente:

```
$ ssh -i <private-key.pem> glue@<development endpoint public address>
```

Per visualizzare le sessioni attive Livy, esegui il comando seguente:

```
$ yarn application -list
20/09/25 06:22:21 INFO client.RMProxy: Connecting to ResourceManager at
 ip-255-1-106-206.ec2.internal/172.38.106.206:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED,
 RUNNING]):2
Application-Id Application-Name Application-Type User Queue State Final-State Progress
 Tracking-URL
application_1601003432160_0005 livy-session-4 SPARK livy default RUNNING UNDEFINED 10%
 http://ip-255-1-4-130.ec2.internal:41867
application_1601003432160_0004 livy-session-3 SPARK livy default RUNNING UNDEFINED 10%
 http://ip-255-1-179-185.ec2.internal:33727
```

Puoi quindi chiudere la sessione Livy con il seguente comando:

```
$ yarn application -kill application_1601003432160_0005
20/09/25 06:23:38 INFO client.RMProxy: Connecting to ResourceManager at
 ip-255-1-106-206.ec2.internal/255.1.106.206:8032
Killing application application_1601003432160_0005
20/09/25 06:23:39 INFO impl.YarnClientImpl: Killed application
 application_1601003432160_0005
```

Gestione di notebook

> **Note**
>
> Gli endpoint di sviluppo sono supportati solo per le versioni di AWS Glue precedenti alla 2.0. Per un ambiente interattivo in cui è possibile creare e testare script di ETL, utilizza Notebook su AWS Glue Studio.

Un notebook consente lo sviluppo e il testing interattivi degli script di estrazione, trasformazione e caricamento (ETL) su un endpoint di sviluppo. AWS Glue fornisce un'interfaccia per i notebook Jupyter di SageMaker. Con AWS Glue, puoi creare e gestire i notebook SageMaker. Puoi anche aprire i notebook SageMaker dalla console AWS Glue.

Inoltre, puoi utilizzare Apache Spark con SageMaker sugli endpoint di sviluppo AWS Glue che supportano SageMaker (ma non i processi ETL AWS Glue). SageMaker Spark è una libreria Apache Spark open source per SageMaker. Per ulteriori informazioni, consulta la pagina Utilizzo di Apache Spark con Amazon SageMaker.

> **Important**
>
> La gestione di notebook SageMaker con gli endpoint di sviluppoAWS Glue è disponibile nelle seguenti regioni AWS:

Regione	Codice
Stati Uniti orientali (Ohio)	`us-east-2`
Stati Uniti orientali (Virginia settentrionale)	`us-east-1`

Regione	Codice
Stati Uniti occidentali (California settentrionale)	`us-west-1`
Stati Uniti occidentali (Oregon)	`us-west-2`
Asia Pacifico (Tokyo)	`ap-northeast-1`
Asia Pacific (Seul)	`ap-northeast-2`
Asia Pacifico (Mumbai)	`ap-south-1`
Asia Pacifico (Singapore)	`ap-southeast-1`
Asia Pacifico (Sydney)	`ap-southeast-2`
Canada (Centrale)	`ca-central-1`
Europa (Francoforte)	`eu-central-1`
Europa (Irlanda)	`eu-west-1`
Europa (Londra)	`eu-west-2`

Creazione di processi ETL visivi con AWS Glue Studio

Un processo AWS Glue incapsula uno script che si connette ai dati di origine, lo elabora e quindi lo scrive nella destinazione dati. Di solito un processo esegue script di estrazione, trasformazione e caricamento (ETL). I processi possono eseguire script progettati per ambienti di runtime Apache Spark e Ray. I processi possono anche eseguire script Python generici (processi shell Python). AWS Glue I trigger possono avviare processi in base a una pianificazione, un evento o su richiesta. È possibile monitorare le esecuzioni dei processi per comprendere i parametri di runtime come esito positivo, durata e ora di inizio.

È possibile utilizzare gli script generati da AWS Glue oppure è possibile fornire i propri. Con uno schema di origine e una posizione o uno schema di destinazione, il generatore di AWS Glue Studio codice può creare automaticamente uno script Apache Spark API (PySpark). Puoi usare questo script come punto di partenza e modificarlo per soddisfare gli obiettivi.

AWS Glue può scrivere file di output in diversi formati di dati. Ogni tipo di processo può supportare diversi formati di output. Per alcuni formati di dati, possono essere scritti formati comuni di compressione.

Accesso alla console AWS Glue

Un job in AWS Glue è costituito dalla logica aziendale che esegue il lavoro di estrazione, trasformazione e caricamento (ETL). Puoi creare processi nella sezione ETL della console AWS Glue.

Per visualizzare i lavori esistenti, accedi AWS Management Console e apri la AWS Glue console all'indirizzo https://console.aws.amazon.com/glue/. Quindi scegli scheda Jobs (Processi) in AWS Glue. L'elenco Jobs (Processi) mostra l'ubicazione dello script associato a ciascun processo quando il processo è stato modificato e l'opzione di segnalibro del processo attuale.

Durante la creazione di un nuovo processo o dopo averlo salvato, è possibile utilizzare AWS Glue Studio per modificare i processi ETL. Poi farlo modificando i nodi nell'editor visivo o modificando lo script del processo in modalità sviluppatore. È inoltre possibile aggiungere e rimuovere nodi nell'editor visivo per creare processi ETL più complicati.

Passaggi successivi per la creazione di un processo in AWS Glue Studio

Puoi utilizzare l'editor visivo dei processi per configurare i nodi per il processo. Ogni nodo rappresenta un'azione, ad esempio la lettura di dati dalla posizione di origine o l'applicazione di una trasformazione ai dati. Ogni nodo aggiunto al processo dispone di proprietà che forniscono informazioni sulla posizione dei dati o sulla trasformazione.

I passaggi successivi per la creazione e la gestione dei lavori sono:

- ETL visivo con AWS Glue Studio
- Visualizzare lo script del processo
- Modificare le proprietà del processo
- Salvare il lavoro
- Avviare un'esecuzione del processo
- Visualizzare le informazioni sulle esecuzioni dei processi recenti
- Accesso al pannello di controllo di monitoraggio dei processi

ETL visivo con AWS Glue Studio

Puoi utilizzare la semplice interfaccia visiva di AWS Glue Studio per creare i processi ETL. Puoi la pagina Jobs (Processi) per creare nuovi processi. È inoltre possibile usare un editor di script o notebook per lavorare direttamente con il codice nello script del processo ETL di AWS Glue Studio.

Nella pagina Jobs (Processi) puoi visualizzare tutti i processi creati con AWS Glue Studio o AWS Glue. In questa pagina puoi visualizzare, gestire ed eseguire i processi.

Vedi anche il tutorial del blog per un altro esempio di come creare processi ETL con AWS Glue Studio.

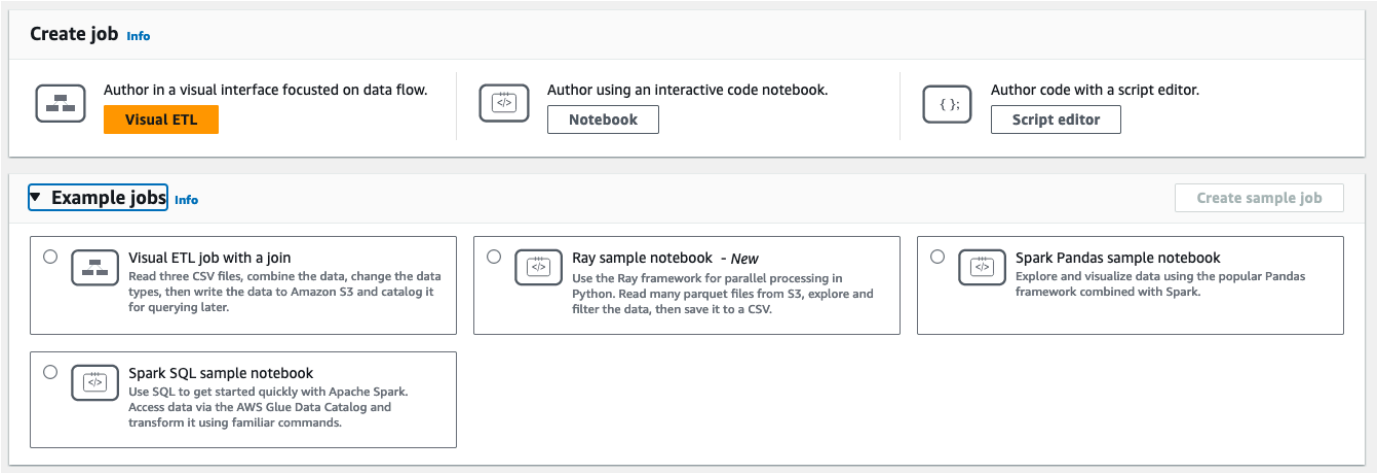
Avvio di processi in AWS Glue Studio

AWS Glue consente di creare un processo tramite un'interfaccia visiva, un notebook di codice interattivo o un editor di script. È possibile avviare un processo facendo clic su una delle opzioni o creare un nuovo processo basato su un processo di esempio.

I processi di esempio creano processi con lo strumento che preferisci. Ad esempio, i processi di esempio consentono di creare un processo ETL visivo che esegue il join di file CSV in una tabella di catalogo, creare un processo in un notebook di codice interattivo con AWS Glue per Ray o AWS Glue per Spark quando si lavora con panda, oppure creare un processo in un notebook di codice interattivo con SparkSQL.

Creare un lavoro AWS Glue Studio partendo da zero

1. Accedi AWS Management Console e apri la AWS Glue Studio console all'indirizzo https://console.aws.amazon.com/gluestudio/.
2. Nel riquadro di navigazione, seleziona Processi ETL.
3. Nella sezione Crea processo, scegli un'opzione di configurazione per il processo.



Opzioni per creare un processo da zero:

- ETL visivo: crea il processo in un'interfaccia visiva incentrata sul flusso di dati
- Crea processi utilizzando un notebook a codice interattivo: crea processi in modo interattivo in un'interfaccia notebook basata su notebook Jupyter

  Prima di selezionare questa opzione e creare una sessione di creazione di processi tramite notebook, è necessario fornire informazioni aggiuntive. Per ulteriori informazioni su come specificare queste informazioni, consulta Nozioni di base sui notebook in AWS Glue Studio.
- Crea codice con un editor di script: se hai familiarità con la programmazione e la scrittura di script ETL, scegli questa opzione per creare un nuovo processo ETL di Spark. Scegli il motore: shell Python, Ray, Spark (Python) o Spark (Scala). Quindi, scegli Inizia da zero o Carica script per caricare uno script esistente da un file locale. Se scegli di utilizzare l'editor di script, per progettare o modificare il tuo processo, non potrai utilizzare l'editor visivo dei processi.

Un processo Spark viene eseguito in un ambiente Apache Spark gestito da AWS Glue. Per impostazione predefinita, i nuovi script sono codificati in Python. Per scrivere un nuovo script Scala, consulta Creazione e modifica di script Scala in AWS Glue Studio.

Creazione di un lavoro a AWS Glue Studio partire da un lavoro di esempio

Puoi scegliere di creare un processo da un processo di esempio. Nella sezione Processi di esempio, scegli un processo di esempio, quindi scegli Crea processo di esempio. La creazione di un processo di esempio da una delle opzioni fornisce un modello rapido per iniziare a lavorare.

1. Accedi AWS Management Console e apri la AWS Glue Studio console all'indirizzo https://console.aws.amazon.com/gluestudio/.
2. Nel riquadro di navigazione, seleziona Processi ETL.
3. Seleziona un'opzione per creare un processo da un processo di esempio:
   - Processo ETL visivo per eseguire il join di più origini: leggi tre file CSV, combina i dati, modifica i tipi di dati, quindi scrivi i dati su Amazon S3 e catalogali per le query successive.
   - Notebook Ray per la parallelizzazione di Python: utilizza il framework Ray per l'elaborazione parallela in Python. Leggi i file parquet da Amazon S3, esplora e filtra i dati, quindi salva in un file CSV.
   - Notebook Spark con Pandas: esplora e visualizza i dati utilizzando il popolare framework Pandas combinato con Spark.
   - Notebook Spark con SQL: inizia rapidamente a utilizzare Apache Spark tramite SQL. Accedi ai dati tramite Catalogo dati AWS Glue e trasformali utilizzando comandi familiari.
4. Scegli Crea un processo di esempio.

Caratteristiche dell'editor dei processi

L'editor di processi offre le seguenti caratteristiche per la creazione e la modifica di processi.

- Un diagramma visivo del processo, con un nodo per ogni attività: nodi di origine dati per la lettura dei dati; nodi di trasformazione per la modifica dei dati; nodi di destinazione dati per la scrittura dei dati.

  È possibile visualizzare e configurare le proprietà di ciascun nodo nel diagramma del processo. È inoltre possibile visualizzare lo schema e i dati di esempio per ogni nodo nel diagramma del

processo. Queste caratteristiche consentono di verificare che il processo stia modificando e trasformando i dati nel modo corretto, senza doverlo eseguire

- Una scheda di visualizzazione e modifica degli script, in cui è possibile modificare il codice generato per il processo.
- Una scheda per i dettagli del processo, in cui è possibile configurare diverse impostazioni per personalizzare l'ambiente in cui viene eseguito il processo ETL AWS Glue.
- Una scheda per le esecuzioni, in cui è possibile visualizzare le esecuzioni correnti e precedenti del processo, lo stato dell'esecuzione del processo e accedere ai registri per l'esecuzione del processo.
- Una scheda per la qualità dei dati, in cui è possibile applicare le regole sulla qualità dei dati al processo.
- Una scheda per le pianificazioni, in cui è possibile configurare l'ora di inizio del processo o impostare le esecuzioni del processo ricorrenti.
- Una scheda per il controllo della versione, in cui è possibile configurare un servizio Git da utilizzare con il processo.

Utilizzo delle anteprime dello schema nell'editor visivo dei processi

Durante la creazione o la modifica del processo, è possibile utilizzare la scheda Output schema (Schema di output) per visualizzare lo schema dei dati.

Prima di poter visualizzare lo schema, l'editor dei processi necessita delle autorizzazioni per accedere all'origine dati. È possibile specificare un ruolo IAM nella scheda dei dettagli del processo dell'editor o nella scheda Output schema (Schema di output) per un nodo. Se il ruolo IAM dispone di tutte le autorizzazioni necessarie per accedere all'origine dati, è possibile visualizzare lo schema nella scheda Output schema (Schema di output) per un nodo.

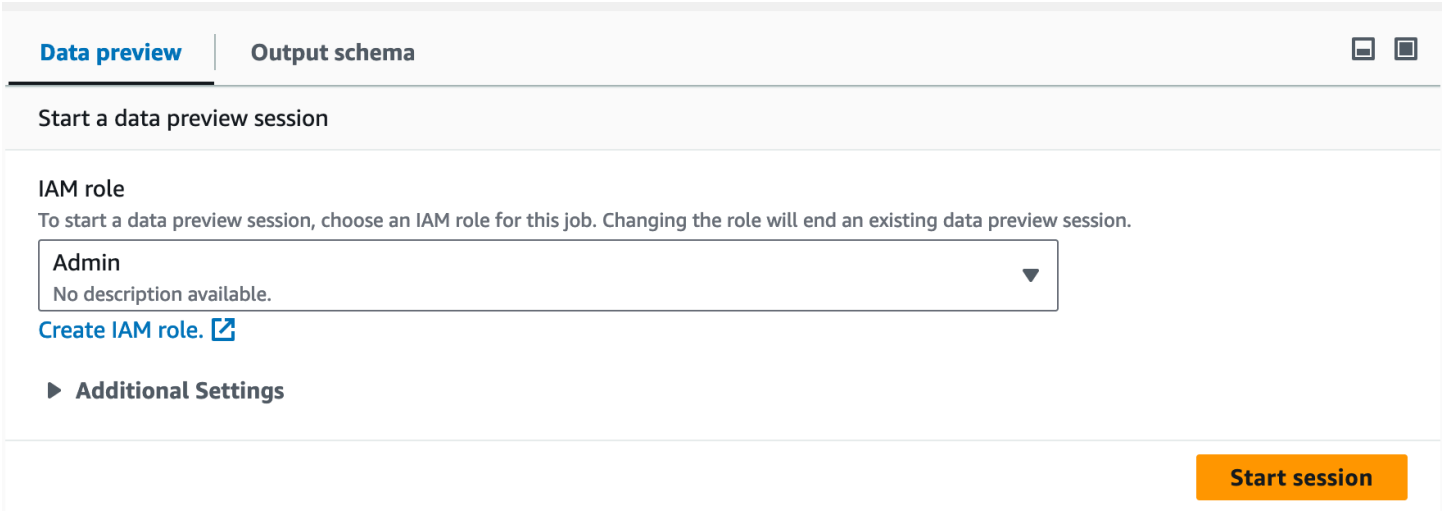
Utilizzo delle anteprime dei dati nell'editor visivo dei processi

Le anteprime dei dati consentono di creare e testare il processo, usando un esempio dei dati, senza doverlo eseguire ripetutamente. Utilizzando l'anteprima dei dati, puoi:

- Verifica un ruolo IAM per assicurarti di avere accesso alle origini dati o alle destinazioni dati.
- Controlla che la trasformazione stia modificando i dati nel modo previsto. Ad esempio, se utilizzi un filtro di trasformazione, puoi accertarti che il filtro stia selezionando il sottoinsieme di dati corretto.

- Controlla i dati. Se il set di dati contiene colonne con valori di più tipi, nell'anteprima dei dati viene visualizzato un elenco di tuple per tali colonne. Ogni tupla contiene il tipo di dato e il suo valore.

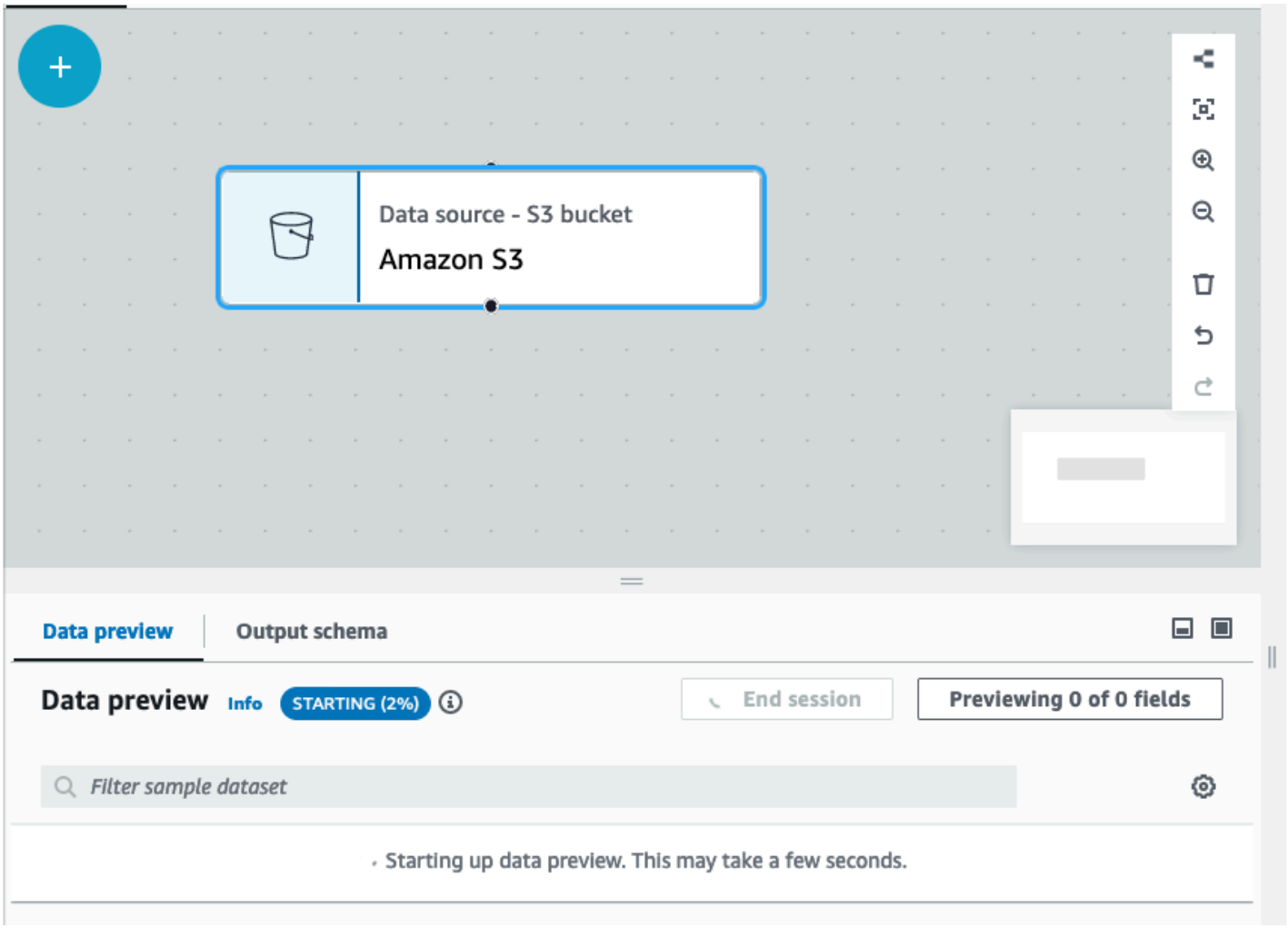
Durante la creazione o la modifica del processo, è possibile utilizzare la scheda Anteprima dei dati sotto il canvas del processo per visualizzare un campione dei dati. Una nuova sessione di anteprima dei dati verrà avviata automaticamente quando il ruolo è già configurato sul processo o è stato impostato un ruolo IAM predefinito nell'account. Se un ruolo non è stato configurato in precedenza, puoi avviare una sessione selezionando il ruolo.



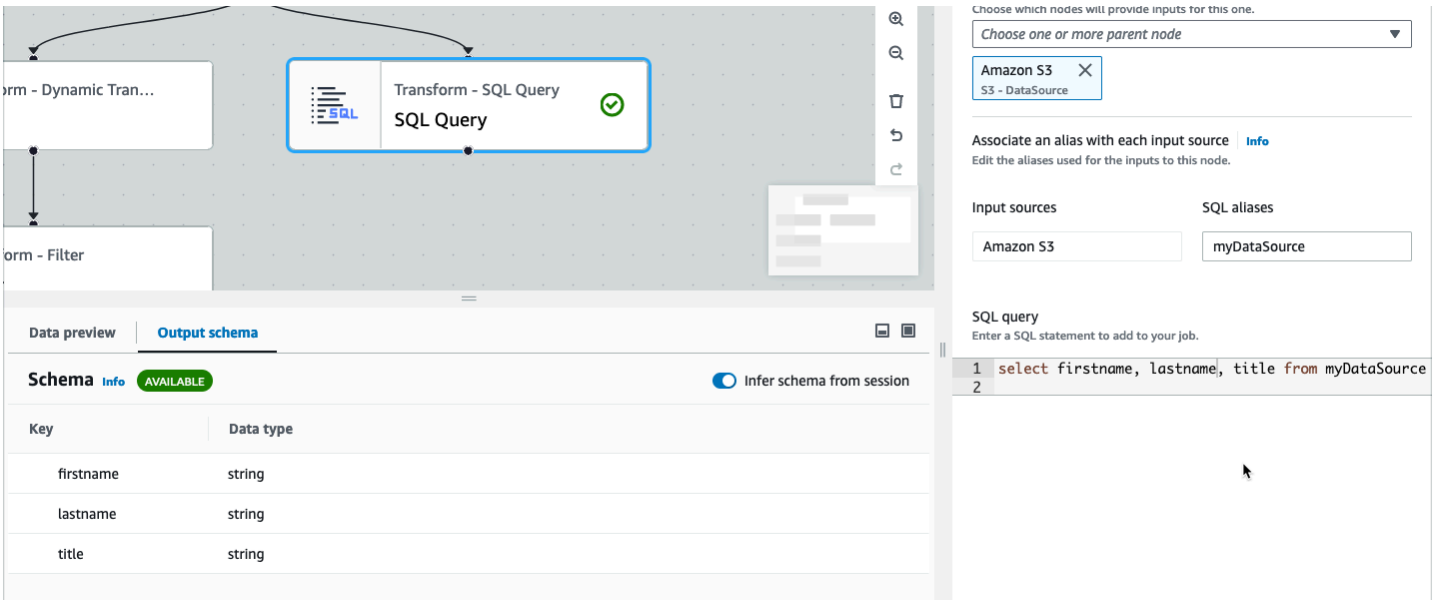
> **Note**
>
> Il ruolo scelto per la sessione di anteprima dei dati verrà utilizzato anche per il processo.

Puoi vedere lo stato e l'avanzamento della sessione, nonché i dettagli della sessione, facendo clic sull'icona delle informazioni.

Quando la sessione è pronta, AWS Glue Studio caricherà i dati per il nodo selezionato. È possibile visualizzare la percentuale di completamento man mano che procede.



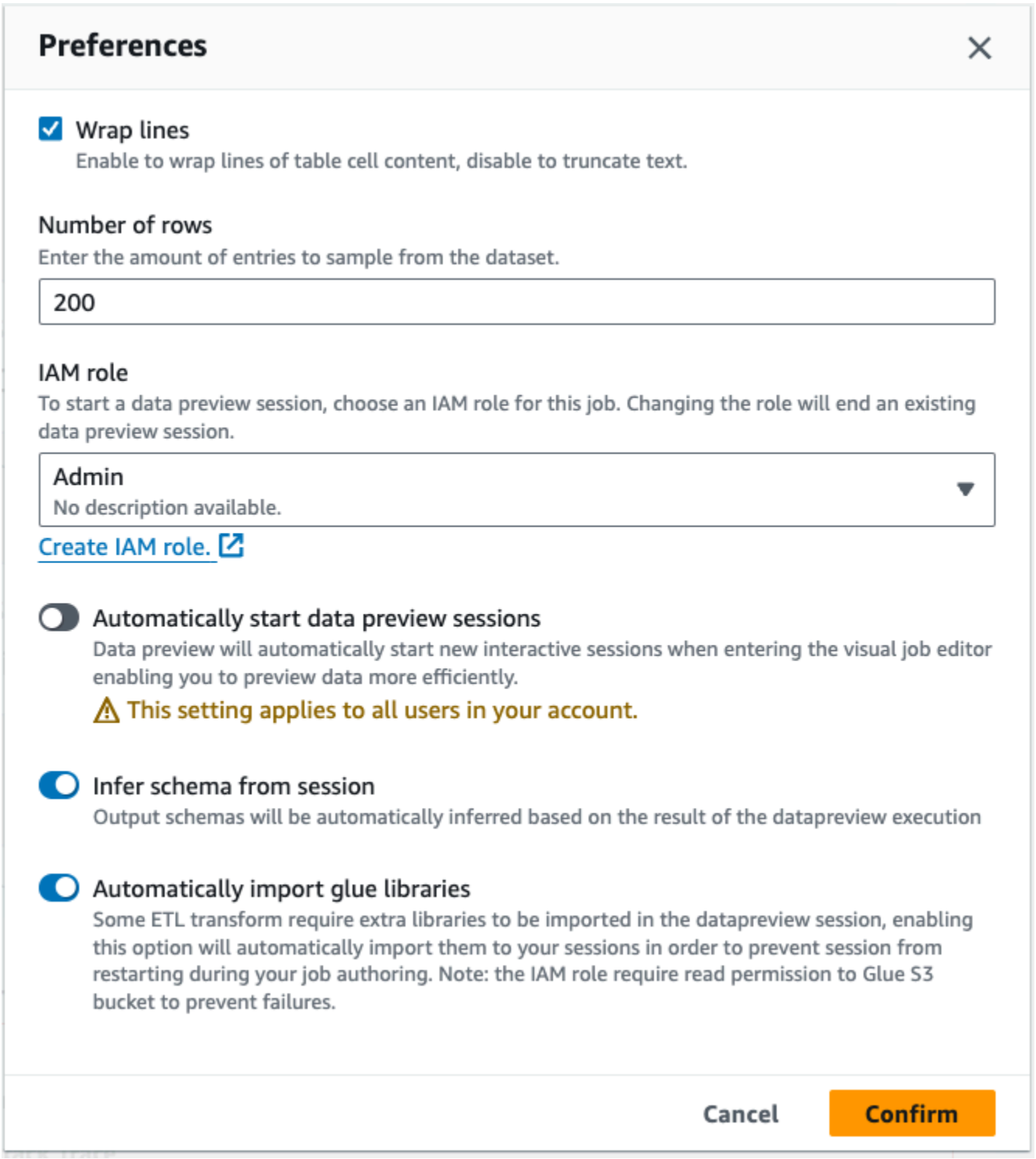
Durante la creazione del processo visivo, AWS Glue Studio aggiornerà automaticamente lo schema per il nodo selezionato quando si attiva Deduci schema dalla sessione nella scheda Schema di output.



Per configurare le preferenze di anteprima dei dati:

Scegliere l'icona delle impostazioni (simbolo dell'ingranaggio) per configurare le preferenze per le anteprime dei dati. Queste impostazioni si applicano a tutti i nodi del diagramma del processo. È possibile:

- Scegliere di avvolgere il testo da una riga all'altra. Per impostazione predefinita, questa opzione è abilitata.
- Modifica il numero di righe (il valore predefinito è 200)
- Scegli un ruolo IAM o creare un ruolo IAM, se necessario
- Scegli di avviare automaticamente una nuova sessione quando si crea un processo. Questo fornisce una nuova sessione interattiva durante la creazione dei processi. Questa impostazione si applica a livello di account. Una volta configurata, verrà applicata a tutti gli utenti dell'account durante la modifica di qualsiasi processo.
- Scegliere di dedurre automaticamente lo schema. Gli schemi di output verranno dedotti automaticamente per il nodo selezionato
- Scegli di importare automaticamente le librerie AWS Glue. Questo è utile perché impedirà che l'anteprima dei dati riavvii nuove sessioni quando si aggiungono nuove trasformazioni che richiedono il riavvio della sessione

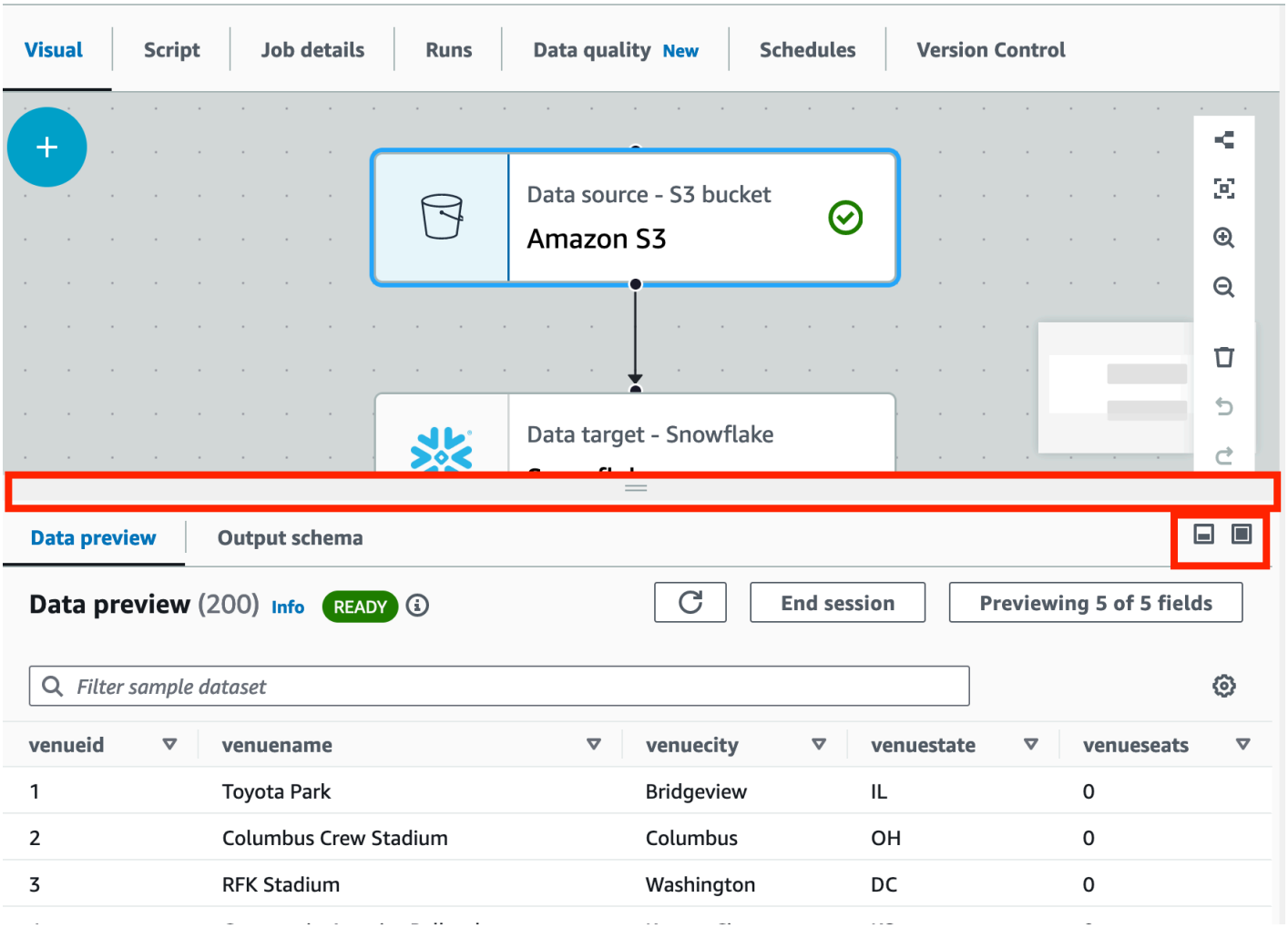


Le funzionalità aggiuntive includono la possibilità di:

- Seleziona Previewing x of y fields (Anteprima dei campi x di y) per selezionare le colonne (campi) da visualizzare in anteprima. Quando si visualizzano in anteprima i dati utilizzando le impostazioni

di default, l'editor dei processi mostra le prime 5 colonne del set di dati. È possibile modificare questa impostazione per mostrare tutte o nessuna (non consigliato).

- Scorri la finestra di anteprima dei dati sia orizzontalmente che verticalmente.
- Per visualizzare meglio i dati e le strutture dei dati, utilizzare il pulsante di ingrandimento per espandere la scheda Anteprima dati e sovrapporre il grafico del processo. Allo stesso modo, utilizzare il pulsante di riduzione al minimo per ridurre al minimo la scheda Anteprima dei dati. È possibile anche selezionare la maniglia del riquadro e trascinarla verso l'alto per espandere la scheda Anteprima dei dati.



- Usa Termina sessione per interrompere l'anteprima dei dati. Quando interrompi la sessione, puoi scegliere un nuovo ruolo IAM e selezionare impostazioni aggiuntive (come attivare o disattivare le impostazioni) per avviare automaticamente una nuova sessione, dedurre lo schema o importare librerie AWS Glue, e riavviare la sessione.

Restrizioni nell'utilizzo delle anteprime dei dati

Quando utilizzi le anteprime dati, potresti riscontrare le seguenti restrizioni o limitazioni.

- Selezionando la scheda Data preview (Anteprima dei dati) per la prima volta, ti verrà richiesto di scegliere un ruolo IAM. Questo ruolo deve disporre delle autorizzazioni necessarie per accedere ai dati e alle altre risorse necessarie per creare le anteprime dei dati.
- Dopo aver fornito un ruolo IAM, è necessario un po' di tempo prima che i dati siano disponibili per la visualizzazione. Per i set di dati con meno di 1 GB di dati, può essere necessario fino a un minuto. Se disponi di un set di dati di grandi dimensioni, utilizza le partizioni per ridurre il tempo di caricamento. Il caricamento dei dati direttamente da Amazon S3 offre le prestazioni migliori.
- Se disponi di un set di dati molto grande e sono necessari più di 15 minuti per eseguire query sui dati per l'anteprima, la richiesta scadrà. Le anteprime dei dati hanno un timeout di inattività di 30 minuti. Per ovviare a questo problema, riduci le dimensioni del set di dati per utilizzare le anteprime dei dati.
- Per impostazione predefinita, vengono visualizzate le prime 50 colonne nella scheda Anteprima dei dati. Se le colonne non contengono valori di dati, verrà visualizzato un messaggio che indica che non sono presenti dati da visualizzare. Puoi aumentare il numero di righe campionate o di colonne selezionate per visualizzare i valori dei dati.
- Le anteprime dei dati non sono attualmente supportate per le origini dati in streaming o per le origini dati che utilizzano connettori personalizzati.
- Gli errori su un nodo influiscono sull'intero processo. Se un nodo presenta un errore con le anteprime dei dati, l'errore verrà visualizzato su tutti i nodi finché non lo si corregge.
- Se si modifica un'origine dati per il processo, potrebbe essere necessario aggiornare i nodi figlio dell'origine dati in modo che corrispondano al nuovo schema. Ad esempio, se si dispone di un nodo ApplyMapping che modifica una colonna e la colonna non esiste nell'origine dati sostitutiva, sarà necessario aggiornare il nodo di trasformazione ApplyMapping.
- Se visualizzi la scheda Data preview (Anteprima dei dati) per un nodo di trasformazione della query SQL e la query SQL utilizza un nome di campo non corretto, nella scheda viene visualizzato un errore.

Generazione di codice dello script

Quando si utilizza l'editor visivo per creare un processo, il codice ETL viene generato automaticamente per te. AWS Glue Studio crea uno script di processo funzionale e completo e lo salva in una posizione Amazon S3.

Esistono due forme di codice generate da AWS Glue Studio: la versione originale o classica e una versione più recente e semplificata. Per impostazione predefinita, il nuovo generatore di codice viene utilizzato per creare lo script del processo. È possibile generare uno script di processo utilizzando il generatore di codice classico sulla scheda Script scegliendo il pulsante di attivazione Generate classic script (Genera script classico).

Alcune delle differenze nella nuova versione del codice generato includono:

- I blocchi di commenti di grandi dimensioni non vengono più aggiunti allo script
- Le strutture di output nel codice utilizzano il nome del nodo specificato nell'editor visivo. Nello script di classe, le strutture di output sono semplicemente denominate `DataSource0`, `DataSource1`, `Transform0`, `Transform1`, `DataSink0`, `DataSink1` e così via.
- I comandi lunghi sono divisi su più righe per eliminare la necessità di scorrere la pagina per visualizzare l'intero comando.

Le nuove caratteristiche di AWS Glue Studio richiedono la nuova versione di generazione del codice e non funziona con il classico script di codice. Quando si tenta di eseguire questi processi, viene richiesto di aggiornarli.

Modifica dei nodi di trasformazione dei dati gestiti da AWS Glue

AWS Glue Studio offre due tipi di trasformazioni:

- Trasformazioni native di AWS Glue: gestite da AWS Glue e disponibili per tutti gli utenti.
- Trasformazioni visive personalizzate: consente di caricare le proprie trasformazioni da utilizzare in AWS Glue Studio

Nodi di trasformazione dei dati gestiti da AWS Glue

AWS Glue Studio offre un set di trasformazioni predefinite che puoi usare per elaborare i dati. I dati passano da un nodo nel diagramma di processo a un altro in una struttura di dati denominata `DynamicFrame`, che è un'estensione di un SQL Apache Spark `DataFrame`.

Nel diagramma precompilato per un processo, tra i nodi di origine dati e di destinazione dati si trova il nodo di trasformazione Modifica schema. È possibile configurare questo nodo di trasformazione per modificare i dati oppure utilizzare ulteriori trasformazioni.

Le seguenti trasformazioni predefinite sono disponibili con AWS Glue Studio:

- ChangeSchema: mappa le chiavi di proprietà dei dati nell'origine dati alle chiavi di proprietà dei dati nella destinazione dati. È possibile rinominare le chiavi, modificare i tipi di dati per le chiavi e scegliere le chiavi da eliminare dal set di dati.
- SelectFields: scegli le chiavi di proprietà dei dati da conservare.
- DropFields: scegli le chiavi di proprietà dei dati da eliminare.
- RenameField: rinomina una singola chiave di proprietà dati.
- Spigot: scrivi esempi dei dati in un bucket Amazon S3.
- Join: esegui il join di due set di dati in un set di dati utilizzando una frase di confronto sulle chiavi di proprietà dei dati specificate. È possibile utilizzare inner, outer, left, right, left semi e left anti join.
- Union: combina righe provenienti da più di un'origine dati che hanno lo stesso schema.
- SplitFields: suddivide le chiavi di proprietà dei dati in due `DynamicFrames`. Output è una raccolta di `DynamicFrames`: uno con le chiavi di proprietà dei dati selezionate e uno con le chiavi di proprietà dei dati rimanenti.
- SelectFromCollection: scegli un `DynamicFrame` da una raccolta di `DynamicFrames`. L'output è il `DynamicFrame` selezionato.
- FillMissingValues: individua i registri nel set di dati con valori mancanti e aggiunge un nuovo campo con un valore determinato dall'imputazione
- Filter: divide un set di dati in due, in base a una condizione di filtro.
- DropNullFields: rimuove le colonne dal set di dati se tutti i valori nella colonna sono "null".
- Elimina i duplicati: rimuove le righe dall'origine dati consentendo di scegliere se abbinare righe intere o specificare le chiavi.
- SQL: inserisce il codice SparkSQL in un campo di inserimento testo per utilizzare una query SQL e trasformare i dati. L'output è un singolo `DynamicFrame`.
- Aggregate: esegue un calcolo (ad esempio media, somma, min, max) su campi e righe selezionati e crea un nuovo campo con i valori appena calcolati.
- Flatten: estrae i campi all'interno delle strutture in campi di primo livello.
- UUID: aggiunge una colonna con un identificatore univoco universale per ogni riga.
- Identifier: aggiunge una colonna con un identificatore numerico per ogni riga.
- To timestamp: converte una colonna in un tipo di timestamp.
- Format timestamp: converte una colonna di timestamp in una stringa formattata.

- Conditional Router transform: applica più condizioni ai dati in ingresso. Ogni riga dei dati in ingresso viene valutata in base a una condizione di filtro di gruppo ed elaborata nel gruppo corrispondente.
- Trasformazione Concatena colonne: crea una nuova colonna di stringhe utilizzando i valori di altre colonne con un distanziatore opzionale.
- Trasformazione Dividi stringa: suddividi una stringa in un array di token utilizzando un'espressione regolare per definire come viene eseguita la suddivisione.
- Trasformazione Array a colonne: estrai alcuni o tutti gli elementi di una colonna di tipo array in nuove colonne.
- Trasformazione Aggiungi timestamp corrente: contrassegna le righe con l'ora in cui i dati sono stati elaborati. Ciò è utile per scopi di controllo o per tenere traccia della latenza nella pipeline di dati.
- Trasformazione Pivot: righe a colonne: aggrega una colonna numerica ruotando valori univoci su colonne selezionate che diventano nuove colonne. Se sono selezionate più colonne, i valori vengono concatenati per denominare le nuove colonne.
- Trasformazione Elimina pivot: righe a colonne: converti le colonne in valori di nuove colonne generando una riga per ogni valore univoco.
- Trasformazione Bilancia automaticamente elaborazione: ridistribuisci i dati tra i worker per migliorare le prestazioni. Ciò è utile nei casi in cui i dati non sono bilanciati o, poiché provengono dall'origine, non consentono un'elaborazione parallela sufficiente.
- Trasformazione Colonna derivata: definisci una nuova colonna basata su una formula matematica o un'espressione SQL in cui è possibile utilizzare altre colonne nei dati, oltre a costanti e valori letterali.
- Trasformazione Ricerca: aggiungi colonne da una tabella di catalogo definita quando le chiavi corrispondono alle colonne di ricerca definite nei dati.
- Trasformazione Espandi array o mappa: estrae i valori da una struttura annidata in singole righe più facili da manipolare.
- Trasformazione Corrispondenza dei record: richiama una trasformazione di classificazione dei dati di machine learning Corrispondenza dei record esistente.
- Trasformazione Rimuovi righe nulle: rimuove dal set di dati le righe che hanno tutte le colonne come nulle o vuote.
- Trasformazione Analizza colonna JSON: analizza una colonna di stringhe contenente dati JSON e convertila in una struttura o in una colonna di array, a seconda che il JSON sia rispettivamente un oggetto o un array.
- Trasformazione Estrai percorso JSON: estrai nuove colonne da una colonna di stringhe JSON.

- Trasformazione Estrai frammenti di stringa con un'espressione regolare: estrai frammenti di stringa utilizzando un'espressione regolare e crea a partire da essa una nuova colonna o anche più colonne, se si utilizzano gruppi di espressioni regolari.
- Custom transform: inserisce il codice in un campo di inserimento testo per utilizzare le trasformazioni personalizzate. L'output è una raccolta di `DynamicFrames`.

Utilizzo di una ricetta di preparazione dei dati in AWS Glue Studio

AWS Glue Studio consente di utilizzare una ricetta AWS Glue DataBrew in un flusso di lavoro visivo. Ciò consente di eseguire le ricette AWS Glue DataBrew di un cliente in un processo AWS Glue insieme ad altri nodi AWS Glue Studio.

In DataBrew, una ricetta è l'insieme dei passaggi di trasformazione dei dati. Le ricette DataBrew prescrivono come trasformare i dati che sono già stati letti e non descrivono come e dove leggere i dati, né come e dove scrivere i dati. Questo è configurato nei nodi di origine e destinazione in AWS Glue Studio. Per ulteriori informazioni sulle ricette, consulta Creating and using AWS Glue DataBrew recipes.

Il nodo Ricetta di preparazione dei dati è disponibile nel pannello Risorse. È possibile connettere il nodo Ricetta di preparazione dei dati a un altro nodo del flusso di processo visivo, che si tratti di un nodo Origine dati o di un altro nodo di trasformazione. Dopo aver scelto una ricetta AWS Glue DataBrew e una versione, i passaggi applicati nella ricetta sono visibili nella scheda delle proprietà del nodo.

Prerequisiti

- Hai creato una ricetta AWS Glue DataBrew in AWS Glue DataBrew.
- Disponi delle autorizzazioni IAM necessarie, come descritto nella sezione seguente.

Autorizzazioni IAM per AWS Glue DataBrew

Questo argomento fornisce informazioni sulle operazioni e le risorse che un amministratore IAM può utilizzare in una policy AWS Identity and Access Management (IAM) per la trasformazione Ricetta di preparazione dei dati.

Per ulteriori informazioni sulla sicurezza in AWS Glue, consulta la pagina Access Management.

La tabella seguente elenca le autorizzazioni richieste a un utente per eseguire determinate operazioni per utilizzare la trasformazione Ricetta di preparazione dei dati.

Azioni di trasformazione di Ricetta di preparazione dei dati

Azione	Descrizione
`databrew:ListRecipes`	Concede l'autorizzazione per recuperare le ricette AWS Glue DataBrew.
`databrew:ListRecipeVersions`	Concede l'autorizzazione per recuperare le versioni delle ricette AWS Glue DataBrew.
`databrew:DescribeRecipe`	Concede l'autorizzazione per recuperare la descrizione delle ricette AWS Glue DataBrew.

Il ruolo che stai utilizzando per accedere a questa funzionalità dovrebbe disporre di una policy che consenta diverse AWS Glue DataBrew. A tale scopo, puoi utilizzare una policy `AWSGlueConsoleFullAccess` che includa le operazioni necessarie o aggiungere la seguente policy inline al tuo ruolo:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "databrew:ListRecipes",
                "databrew:ListRecipeVersions",
                "databrew:DescribeRecipe"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Per utilizzare la trasformazione Ricetta di preparazione dei dati, devi aggiungere l'operazione `IAM:PassRole` alla policy delle autorizzazioni.

Autorizzazioni aggiuntive richieste

Azione	Descrizione
`iam:PassRole`	Concede a IAM l'autorizzazione per consentire all'utente di trasmettere i ruoli approvati.

Senza queste autorizzazioni si verifica il seguente errore:

```
"errorCode": "AccessDenied"
"errorMessage": "User: arn:aws:sts::account_id:assumed-role/AWSGlueServiceRole is not
authorized to perform: iam:PassRole on resource: arn:aws:iam::account_id:role/service-
role/AWSGlueServiceRole
because no identity-based policy allows the iam:PassRole action"
```

Limitazioni

- Non tutte le ricette AWS Glue DataBrew sono supportate da AWS Glue. Alcune ricette non potranno essere eseguite in AWS Glue Studio.
  - Le ricette con trasformazioni UNION e JOIN non sono supportate, tuttavia AWS Glue Studio dispone già di nodi di trasformazione "Join" e "Union" che possono essere utilizzati prima o dopo un nodo Ricetta di preparazione dei dati.
- I nodi Ricetta di preparazione dei dati sono supportati per i processi a partire da AWS Glue versione 4.0. Questa versione verrà selezionata automaticamente dopo l'aggiunta di un nodo Ricetta di preparazione dei dati al processo.
- I nodi Ricetta di preparazione dei dati richiedono Python. Viene impostato automaticamente quando il nodo Ricetta di preparazione dei dati viene aggiunto al processo.
- Quando si utilizza Anteprima dati, è necessario riavviare la sessione di anteprima dei dati dopo aver aggiunto un nodo Ricetta di preparazione dei dati al processo.

Come utilizzare le ricette AWS Glue DataBrew in AWS Glue Studio

Per utilizzare le ricette AWS Glue DataBrew in AWS Glue Studio, inizia creando le ricette in AWS Glue DataBrew. Se disponi di ricette che desideri utilizzare, puoi ignorare questo passaggio.

Per creare una nuova ricetta AWS Glue DataBrew in AWS Glue DataBrew:

1. Scrivi una ricetta in AWS Glue DataBrew. Per ulteriori informazioni, consulta la pagina Getting started with AWS Glue DataBrew.
2. Salva la ricetta.
3. Pubblica la ricetta. Questo pubblicherà la tua ricetta come versione 1.0.

Per utilizzare un nodo Ricetta di preparazione dei dati in AWS Glue Studio:

È possibile utilizzare più di un nodo Ricetta di preparazione dei dati in un processo ETL visivo. A tale scopo, aggiungi un nodo Ricetta di preparazione dei dati completando i passaggi seguenti e aggiungi un altro nodo Ricetta di preparazione dei dati al processo. Ad esempio, un flusso di lavoro potrebbe seguire questo modello:

- Origine dati 1 > ricetta 1 > output 1
- Origine dati 2 > ricetta 2 > output 2
- output 1, output 2 > JOIN

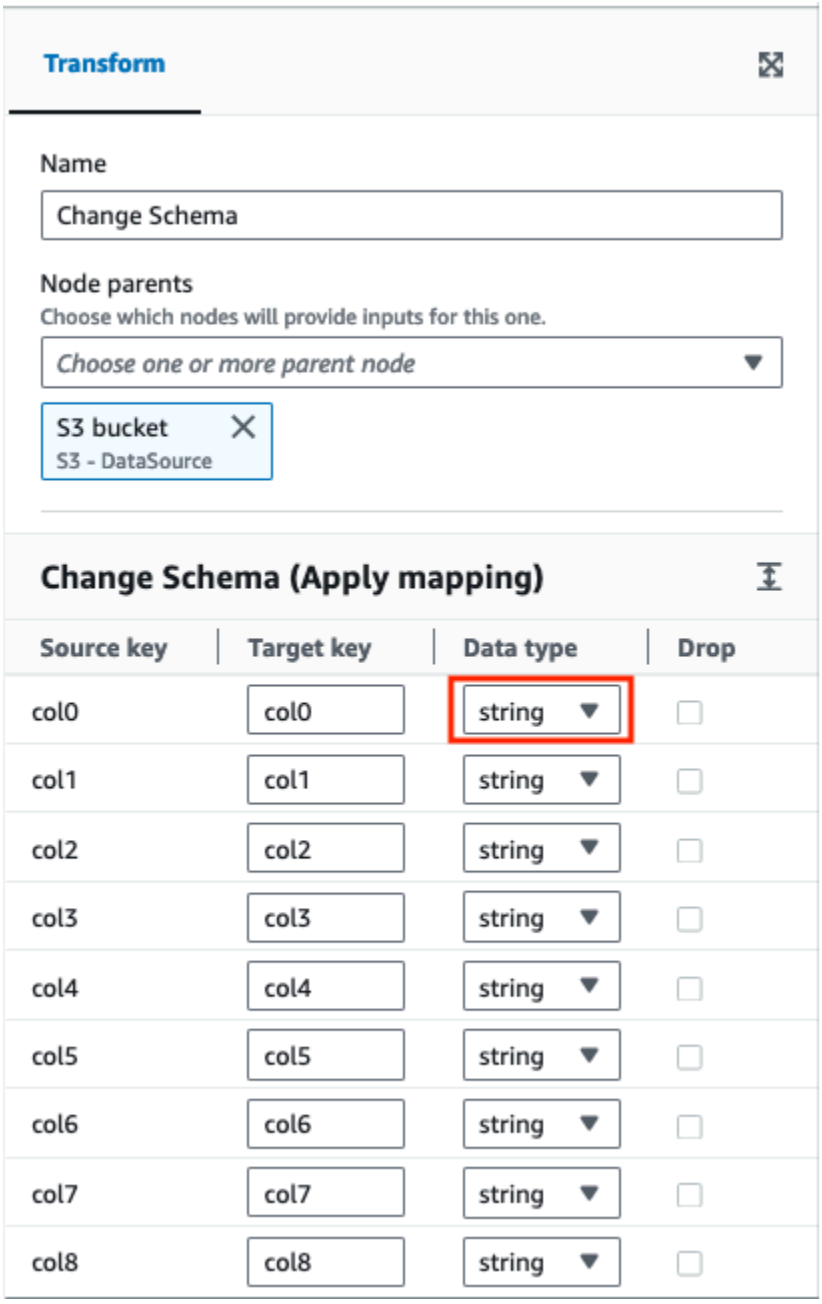
1. Avvia un processo AWS Glue con un'origine dati AWS Glue Studio.
2. Aggiungi il nodo Ricetta di preparazione dei dati all'origine dati.
3. Filtra la ricetta per nome digitando il nome della ricetta nel campo di ricerca.
4. Scegli la versione pubblicata. Sono disponibili solo le versioni pubblicate.
5. Completa la creazione del processo aggiungendo altri nodi di trasformazione secondo necessità e aggiungi i nodi di destinazione dei dati per salvare l'output del processo.
6. Apporta le modifiche di configurazione necessarie nella scheda Dettagli del processo, ad esempio assegnando un nome al processo e regolando la capacità allocata in base alle esigenze, e salva il processo.
7. Esegui il processo selezionando Esegui dal menu a discesa Operazioni.

Per modificare lo schema se l'origine dati è Amazon S3 e il formato dei dati è CSV:

Se tutte le colonne di un file CSV vengono inizialmente caricate come tipo di dati stringa in AWS Glue Studio, devi assicurarti che il tipo di dati della colonna sia compatibile con i restanti passaggi nella ricetta AWS Glue DataBrew.

Le ricette di AWS Glue DataBrew prescrivono solamente come trasformare i dati che sono già stati letti. Non descrive dove e come leggere i dati.

1. Aggiungi un nodo Modifica schema prima del nodo della ricetta Multi-fase.
2. Fai clic sul nodo Modifica schema e modifica lo schema in modo che sia uguale ai tipi di dati nella colonna AWS Glue DataBrew selezionando il nuovo tipo di dati in Trasforma per le colonne, se necessario.



Per modificare lo schema se l'origine dati è senza intestazione:

Le ricette di AWS Glue DataBrew prescrivono solamente come trasformare i dati che sono già stati letti. Non descrive dove e come leggere i dati.

Quando si caricano set di dati senza intestazione in AWS Glue Studio, i nomi delle intestazioni predefiniti sono diversi da quelli caricati in AWS Glue DataBrew.

1. Nel processo ETL, aggiungi un nodo Modifica schema prima del nodo Ricetta di preparazione dei dati.
2. Scegli il nodo Modifica schema e modifica i nomi delle colonne utilizzando gli stessi nomi usati nella ricetta AWS Glue DataBrew.

Utilizzo di Modifica schema per mappare nuovamente le chiavi delle proprietà dei dati

Una trasformazione Modifica schema mappa nuovamente le chiavi di proprietà dei dati di origine nella configurazione desiderata per i dati di destinazione. In un nodo di trasformazione Modifica schema, puoi:

- Modificare il nome di più chiavi di proprietà dati.
- Modificare il tipo di dati delle chiavi di proprietà dei dati, se il nuovo tipo di dati è supportato e esiste un percorso di trasformazione tra i due tipi di dati.
- Scegliere un sottoinsieme di chiavi di proprietà dei dati indicando quali chiavi di proprietà dei dati si desidera eliminare.

È possibile aggiungere ulteriori nodi Modifica schema al diagramma del processo in base alle esigenze, ad esempio per modificare origini dati aggiuntive o in seguito a una trasformazione Join.

> **Note**
>
> La trasformazione Modifica schema non fa distinzione tra maiuscole e minuscole.

Aggiunta di un nodo di trasformazione Modifica schema al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli Modifica schema per aggiungere una nuova trasformazione al diagramma di processo, se necessario.

2. Nel pannello Proprietà del nodo, inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Trasforma nel pannello Proprietà del nodo.
4. Modifica lo schema di input:

   - Per rinominare una chiave di proprietà dati, inserisci il nuovo nome della chiave nel campo Target key (Chiave di destinazione).
   - Per modificare il tipo di dati per una chiave di proprietà dei dati, scegli il nuovo tipo di dati per la chiave dall'elenco Data type (Tipo di dati).
   - Per rimuovere una chiave di proprietà dati dallo schema di destinazione, scegli la casella di controllo Drop (Elimina) per quella chiave.
5. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
6. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Utilizzo di Elimina duplicati

La trasformazione Elimina duplicati offre due opzioni per rimuovere le righe dall'origine dati. È possibile scegliere di rimuovere le righe duplicate interamente uguali oppure selezionare alcuni campi e rimuovere le righe corrispondenti solo in base ai campi scelti.

Ad esempio, in questo set di dati sono presenti righe duplicate in cui tutti i valori in alcune righe sono esattamente gli stessi di un'altra riga mentre altri sono uguali o diversi.

Riga	Nome	E-mail	Età	Stato	Nota
1	Joy	joy@gmail	33	NY	

Riga	Nome	E-mail	Età	Stato	Nota
2	Tim	tim@gmail	45	OH	
3	Rose	rose@gmail	23	NJ	
4	Tim	tim@gmail	42	OH	
5	Rose	rose@gmail	23	NJ	
6	Tim	tim@gmail	42	OH	Questa è una riga duplicata e corrisponde completamente in tutti i valori alla riga n. 4
7	Rose	rose@gmail	23	NJ	Questa è una riga duplicata e corrisponde completamente in tutti i valori alla riga n. 5

Se scegli di abbinare righe intere, le righe 6 e 7 verranno rimosse dal set di dati. Il set di dati ora è:

Riga	Nome	E-mail	Età	Stato
1	Joy	joy@gmail	33	NY
2	Tim	tim@gmail	45	OH
3	Rose	rose@gmail	23	NJ
4	Tim	tim@gmail	42	OH

Riga	Nome	E-mail	Età	Stato
5	Rose	rose@gmail	23	NJ

Se hai scelto di specificare le chiavi, puoi scegliere di rimuovere le righe che corrispondono a "nome" ed "e-mail". In questo modo puoi esercitare un maggiore controllo su che cosa si intende per "riga duplicata" per il tuo set di dati. Specificando "nome" ed "e-mail", il set di dati ora è:

Riga	Nome	E-mail	Età	Stato
1	Joy	joy@gmail	33	NY
2	Tim	tim@gmail	45	OH
3	Rose	rose@gmail	23	NJ

Alcune cose da tenere a mente:

- Affinché le righe vengano riconosciute come duplicate, i valori fanno distinzione tra maiuscole e minuscole. Tutti i valori nelle righe devono avere la stessa successione di maiuscole e minuscole. Questo vale per entrambe le opzioni scelte (Abbina righe intere o Specifica le chiavi).
- Tutti i valori vengono letti come stringhe.
- La trasformazione Elimina duplicati utilizza il comando dropDuplicates di Spark.
- Quando si utilizza la trasformazione Elimina duplicati, la prima riga viene mantenuta e le altre righe vengono eliminate.
- La trasformazione Elimina duplicati non modifica lo schema del dataframe. Se scegli di specificare le chiavi, tutti i campi vengono conservati nel dataframe risultante.

Utilizzo di SelectFields per rimuovere la maggior parte delle chiavi di proprietà dei dati

Puoi creare un sottoinsieme di chiavi di proprietà dei dati dal set di dati utilizzando la trasformazione SelectFields. Puoi indicare quali chiavi di proprietà dei dati conservare e le altre vengono rimosse dal set di dati.

> **Note**
>
> La trasformazione SelectFields fa distinzione tra maiuscole e minuscole. Utilizza ApplyMapping se ti serve un modo per selezionare i campi che faccia distinzione tra maiuscole e minuscole.

Per aggiungere un nodo di trasformazione SelectFields al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli SelectFields per aggiungere una nuova trasformazione al diagramma di processo, se necessario.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Transform (Trasformazione) nel pannello dei dettagli del nodo.
4. Sotto l'intestazione SelectFields, scegli le chiavi di proprietà dei dati nel set di dati da conservare. Tutte le chiavi di proprietà dei dati non selezionate vengono eliminate dal set di dati.

   Puoi anche selezionare la casella di controllo accanto all'intestazione di colonna Field (Campo) per scegliere automaticamente tutte le chiavi di proprietà dei dati nel set di dati. Quindi puoi deselezionare singolarmente le chiavi di proprietà dei dati per rimuoverle dal set di dati.
5. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
6. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Utilizzo di DropFields per conservare la maggior parte delle chiavi di proprietà dati

Puoi creare un sottoinsieme di chiavi di proprietà dei dati dal set di dati utilizzando la trasformazione DropFields. Puoi indicare quali chiavi di proprietà dei dati rimuovere dal dataset e le altre vengono conservate.

> **Note**
>
> La trasformazione DropFields fa distinzione tra maiuscole e minuscole. Utilizza Modifica schema se ti serve un modo per selezionare i campi che non faccia distinzione tra maiuscole e minuscole.

Per aggiungere un nodo di trasformazione DropFields al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli DropFields per aggiungere una nuova trasformazione al diagramma di processo, se necessario.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Transform (Trasformazione) nel pannello dei dettagli del nodo.
4. Sotto l'intestazione DropFields, scegli le chiavi di proprietà dei dati da eliminare dall'origine dati.

   Puoi anche selezionare la casella di controllo accanto all'intestazione di colonna Field (Campo) per scegliere automaticamente tutte le chiavi di proprietà dei dati nel set di dati. Quindi puoi deselezionare singolarmente le chiavi di proprietà dei dati per mantenerle nel set di dati.
5. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
6. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Rinominare un campo nel set di dati

Puoi utilizzare la trasformazione RenameField per modificare il nome di una singola chiave di proprietà nel set di dati.

> **Note**
>
> La trasformazione RenameField fa distinzione tra maiuscole e minuscole. Utilizza ApplyMapping se ti serve una trasformazione che faccia distinzione tra maiuscole e minuscole.

> **Tip**
>
> Utilizzando la trasformazione Modifica schema è possibile rinominare più chiavi di proprietà dei dati nel set di dati con una singola trasformazione.

Per aggiungere un nodo di trasformazione RenameField al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli RenameField per aggiungere una nuova trasformazione al diagramma di processo, se necessario.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Transform (Trasformazione).
4. Sotto l'intestazione Data field (Campo dati), scegli una chiave di proprietà dai dati di origine, quindi inserisci un nuovo nome nel campo New field name (Nuovo nome campo).
5. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
6. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del

processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Utilizzo di Spigot per campionare il set di dati

Per testare le trasformazioni eseguite dal processo, è possibile ottenere un campione dei dati, allo scopo di verificare che la trasformazione funzioni come previsto. La trasformazione Spigot scrive un sottoinsieme di regitri dal set di dati in un file JSON in un bucket Amazon S3. Il metodo di campionamento dei dati può essere un numero specifico di registri dall'inizio del file o un fattore di probabilità utilizzato per selezionare i registri.

Per aggiungere un nodo di trasformazione Spigot al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli Spigot per aggiungere una nuova trasformazione al diagramma di processo, se necessario.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Transform (Trasformazione) nel pannello dei dettagli del nodo.
4. Inserisci un percorso Amazon S3 o scegli Browse S3 (Sfoglia S3) per scegliere una posizione in Amazon S3. Questa è la posizione in cui il processo scrive il file JSON che contiene l'esempio di dati.
5. Inserisci le informazioni per il metodo di campionamento. Puoi specificare un valore per Number of records (Numero di registri) da scrivere a partire dall'inizio del set di dati e una Probability threshold (Soglia di probabilità) (inserita sotto forma di valore decimale con un valore massimo di 1) di scelta di un dato registro.

   Ad esempio, per scrivere i primi 50 registri dal set di dati, è necessario impostare Number of records (Numero di registri) su 50 e Probability threshold (Soglia di probabilità) su 1 (100%).

Unione di set di dati

La trasformazione Join consente di combinare due set di dati in uno. È possibile specificare i nomi delle chiavi nello schema di ogni set di dati da confrontare. L'output `DynamicFrame` contiene righe in cui le chiavi soddisfano la condizione di join. Le righe in ogni set di dati che soddisfano la condizione di join vengono combinate in una singola riga nell'output `DynamicFrame`, che contiene tutte le colonne trovate in entrambi i set di dati.

Per aggiungere un nodo di trasformazione Join al diagramma di processo

1. Se è disponibile una sola origine dati, è necessario aggiungere un nuovo nodo di origine dati al diagramma di processo.
2. Scegli uno dei nodi di origine per il join. Apri il pannello Risorse, quindi scegli Join per aggiungere una nuova trasformazione al diagramma del processo.
3. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo.
4. Nella scheda Node properties (Proprietà del nodo), sotto l'intestazione Node parents (Nodi padre), aggiungi un nodo padre in modo che ci siano due set di dati che forniscono input per il join. Il padre può essere un nodo di origine dati o un nodo di trasformazione.

   > **Note**
   >
   > Un join può avere solo due nodi padre.

5. Seleziona la scheda Transform (Trasformazione).

   Se viene visualizzato un messaggio che indica che esistono nomi di chiavi in conflitto, è possibile:

   - Scegliere Resolve it (Risolvi) per aggiungere automaticamente un nodo di trasformazione ApplyMapping nel diagramma del processo. Il nodo ApplyMapping aggiunge un prefisso a tutte le chiavi del set di dati che hanno lo stesso nome di una chiave nell'altro set di dati. Ad esempio, se utilizzi il valore predefinito di **`right`**, tutte le chiavi nel set di dati destro che hanno lo stesso nome di una chiave nel set di dati sinistro verranno rinominate in `(right)`*`key name`*.
   - Aggiungere manualmente un nodo di trasformazione in precedenza nel diagramma del processo per rimuovere o rinominare le chiavi in conflitto.
6. Scegli il tipo di join nell'elenco Join type (Tipo di join).

   - Inner join: restituisce una riga con colonne di entrambi i set di dati per ogni corrispondenza in base alla condizione di join. Le righe che non soddisfano la condizione di join non vengono restituite.
   - Left join: tutte le righe del set di dati sinistro e solo le righe del set di dati destro che soddisfano la condizione di join.

- Right join: tutte le righe del set di dati destro e solo le righe del set di dati sinistro che soddisfano la condizione di join.
- Outer join: tutte le righe di entrambi i set di dati.
- Left semi join: tutte le righe del set di dati sinistro che hanno una corrispondenza nel set di dati destro in base alla condizione di join.
- Right semi join: tutte le righe del set di dati sinistro che non hanno una corrispondenza nel set di dati destro in base alla condizione di join.

7. Nella scheda Transform (Trasformazione), sotto l'intestazione Join conditions (Condizioni di join), scegli Add condition (Aggiungi condizione). Scegli una chiave di proprietà da ciascun set di dati da confrontare. Le chiavi di proprietà sul lato sinistro dell'operatore di confronto vengono definite come set di dati sinistro e le chiavi di proprietà a destra vengono definite come set di dati destro.

   Per condizioni di join più complesse, è possibile aggiungere ulteriori chiavi di corrispondenza scegliendo Add condition (Aggiungi condizione) più di una volta. Se si aggiunge accidentalmente una condizione, è possibile selezionare l'icona di eliminazione ( ) per rimuoverla.

8. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
9. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa funzionalità e la fatturazione inizia non appena si fornisce un ruolo IAM.

Per un esempio di schema di output del join, considera un join tra due set di dati con le seguenti chiavi di proprietà:

```
Left: {id, dept, hire_date, salary, employment_status}
Right: {id, first_name, last_name, hire_date, title}
```

Il join è configurato in modo che corrisponda alle chiavi `id` e `hire_date` utilizzando l'operatore di confronto =.

Perché entrambi i set di dati contengono le chiavi `id` e `hire_date`, scegli Resolve it (Risolvi) per aggiungere automaticamente il prefisso **`right`** alle chiavi nel set di dati giusto.

Le chiavi nello schema di output sarebbero:

```
{id, dept, hire_date, salary, employment_status,
(right)id, first_name, last_name, (right)hire_date, title}
```

Utilizzo di Union per combinare le righe

Il nodo di trasformazione Union si utilizza quando si desidera combinare righe provenienti da più di un'origine dati aventi il medesimo schema.

Esistono due tipi di trasformazioni Union:

1. ALL: quando si applica ALL, l'unione risultante non rimuove le righe duplicate.
2. DISTINCT: quando si applica DISTINCT, l'unione risultante rimuove le righe duplicate.

Union e Join: differenze

Si utilizza Union per combinare le righe. Si utilizza Join per combinare le colonne.

Utilizzo della trasformazione Union nel canvas di ETL visivo

1. Aggiungi più di un'origine dati per eseguire una trasformazione Union. Per aggiungere un'origine dati, apri il pannello Risorse, quindi scegli l'origine dati dalla scheda Origini. Prima di utilizzare la trasformazione Union, devi assicurarti che tutte le origini dati coinvolte nell'unione abbiano lo stesso schema e la stessa struttura.
2. Quando hai almeno due origini dati che desideri combinare utilizzando la trasformazione Union, crea la trasformazione Union aggiungendola al canvas. Apri il pannello Risorse sul canvas e cerca "Union". In alternativa, scegli la scheda Trasformazioni nel pannello Risorse, scorri verso il basso fino a trovare la trasformazione Union, quindi scegli Union.
3. Seleziona il nodo Union nel canvas del processo. Nella finestra Proprietà del nodo, scegli i nodi padri da connettere alla trasformazione Union.
4. AWS Glue verifica la compatibilità per assicurarsi che la trasformazione Union possa essere applicata a tutte le origini dati. Se lo schema delle origini dati è lo stesso, l'operazione sarà

consentita. Se le origini dati non hanno lo stesso schema, viene visualizzato un messaggio di errore: "The input schemas of this union are not the same. Consider using ApplyMapping to match the schemas." Per risolvere questo problema, scegli Utilizza ApplyMapping.

5. Scegli il tipo di Union.

    1. All: per impostazione predefinita, è selezionato il tipo All Union; ciò comporterà la duplicazione delle righe, se presenti nella combinazione di dati.
    2. Distinct: scegli Distinct se desideri che le righe duplicate vengano rimosse dalla combinazione di dati risultante.

Utilizzo di SplitFields per dividere un set di dati in due

La trasformazione SplitFields consente di scegliere alcune delle chiavi di proprietà dei dati nel set di dati di input e inserirle in un set di dati, inserendo le chiavi non selezionate in un set di dati separato. L'output di questa trasformazione è una raccolta di `DynamicFrames`.

> **Note**
>
> È necessario utilizzare una trasformazione SelectFromCollection per convertire la raccolta di `DynamicFrames` in un singolo `DynamicFrame` prima di poter inviare l'output a una posizione di destinazione.

La trasformazione SplitFields fa distinzione tra maiuscole e minuscole. Aggiungi una trasformazione ApplyMapping come nodo padre se sono necessari nomi di chiavi di proprietà senza distinzione tra maiuscole e minuscole.

Per aggiungere un nodo di trasformazione SplitFields al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli SplitFields per aggiungere una nuova trasformazione al diagramma di processo, se necessario.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Transform (Trasformazione).
4. Scegli le chiavi di proprietà da inserire nel primo set di dati. Le chiavi non scelte vengono inserite nel secondo set di dati.

5. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
6. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa funzionalità e la fatturazione inizia non appena si fornisce un ruolo IAM.
7. Configurare una trasformazione SelectFromCollection per elaborare i set di dati risultanti.

Panoramica della trasformazione SelectFromCollection

Alcune trasformazioni, come SplitFields, hanno più set di dati come output invece di un singolo set di dati. La trasformazione SelectFromCollection seleziona un set di dati (`DynamicFrame`) da una raccolta di set di dati (una matrice di `DynamicFrames`). L'output per la trasformazione è il `DynamicFrame` selezionato.

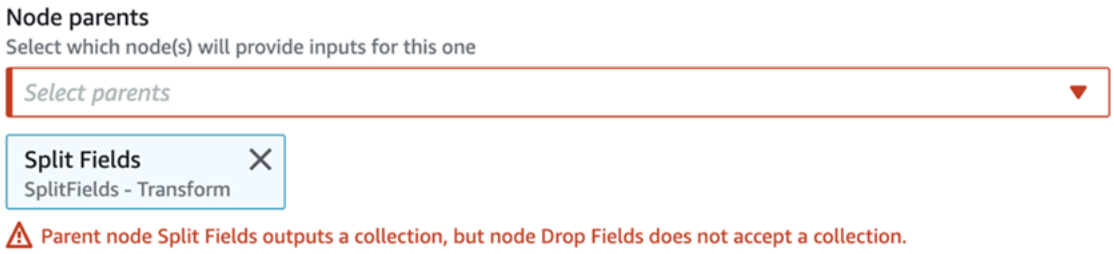
È necessario utilizzare questa trasformazione dopo aver utilizzato una trasformazione che crea una raccolta di `DynamicFrames`, come ad esempio:

- Trasformazioni di codice personalizzate
- SplitFields

Se dopo una di queste trasformazioni non si aggiunge un nodo di trasformazione SelectFromCollection al diagramma di processo, si otterrà un errore.

Il nodo padre per questa trasformazione deve essere un nodo che restituisce una raccolta di `DynamicFrames`. Se per questo nodo di trasformazione si sceglie un padre che restituisce un singolo `DynamicFrame`, ad esempio Join, il processo restituisce un errore.

Similmente, se si utilizza un nodo SelectFromCollection nel diagramma di processo come padre per una trasformazione che si aspetta un singolo `DynamicFrame` come input, il processo restituisce un errore.



Utilizzo di SelectFromCollection per scegliere quale set di dati conservare

Utilizzo della trasformazione SelectFromCollection per convertire una raccolta di `DynamicFrames` in un singolo `DynamicFrame`.

Per aggiungere un nodo di trasformazione SelectFromCollection al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli SelectFromCollection per aggiungere una nuova trasformazione al diagramma di processo, se necessario.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Transform (Trasformazione).
4. Sotto l'intestazione Frame index (Indice del frame), scegli il numero di indice della matrice che corrisponde al `DynamicFrame` da selezionare dalla raccolta di `DynamicFrames`.

   Ad esempio, se il nodo padre per questa trasformazione è una trasformazione SplitFields, nella scheda Output schema (Schema di output) di quel nodo puoi vedere lo schema per ogni `DynamicFrame`. Per mantenere il `DynamicFrame` associato allo schema per Output 2, devi selezionare **1** per il valore di Frame index (Indice di frame), che è il secondo valore nell'elenco.

   Solo il `DynamicFrame` scelto è incluso nell'output.
5. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
6. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel

pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Trovare e riempi i valori mancanti in un set di dati

Puoi utilizzare la trasformazione FillMissingValues per individuare i registri nel set di dati con valori mancanti e aggiungere un nuovo campo con un valore determinato dall'imputazione. Il set di dati di input viene utilizzato per addestrare il modello di Machine Learning (ML) che determina quale dovrebbe essere il valore mancante. Se si utilizzano set di dati incrementali, ogni set incrementale viene utilizzato come dati di addestramento per il modello ML, pertanto i risultati potrebbero non essere molto accurati.

Per aggiungere un nodo di trasformazione FillMissingValues al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli FillMissingValues per aggiungere una nuova trasformazione al diagramma di processo, se necessario.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Transform (Trasformazione).
4. Per Data field (Campo dati), scegli il nome della colonna o del campo dai dati di origine da analizzare per i valori mancanti.
5. (Facoltativo) Nel campo New field name (Nuovo nome campo), inserisci un nome per il campo aggiunto a ciascun registro che conterrà il valore di sostituzione stimato per il campo analizzato. Se nel campo analizzato non ci sono valori mancanti, il valore nel campo analizzato viene copiato nel nuovo campo.

   Se non specifichi un nome per il nuovo campo, il nome predefinito è il nome della colonna analizzata con aggiunta di `_filled`. Ad esempio, se inserisci **`Age`** per Data field (Campo dati) senza specificare un valore per New field name (Nuovo nome campo), a ogni registro viene aggiunto un nuovo campo denominato **`Age_filled`**.
6. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato

un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.

7. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Filtro delle chiavi all'interno di un set di dati

Utilizzo della trasformazione Filter per creare un nuovo set di dati filtrando i registri dal set di dati di input in base a un'espressione regolare. Le righe che non soddisfano la condizione di filtro vengono rimosse dall'output.

- Per i tipi di dati stringa, è possibile filtrare le righe in cui il valore della chiave corrisponde a una stringa specificata.
- Per i tipi di dati numerici, è possibile filtrare le righe confrontando il valore della chiave con un valore specificato utilizzando gli operatori di confronto <, >, =, !=, <= e >=.

Se si specificano più condizioni di filtro, i risultati vengono combinati utilizzando AND per impostazione predefinita, ma è possibile anche scegliere OR.

La trasformazione Filter fa distinzione tra maiuscole e minuscole. Aggiungi una trasformazione ApplyMapping come nodo padre se sono necessari nomi di chiavi di proprietà senza distinzione tra maiuscole e minuscole.

Per aggiungere un nodo di trasformazione Filter al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli Filtra per aggiungere una nuova trasformazione al diagramma di processo, se necessario.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Seleziona la scheda Transform (Trasformazione).
4. Scegli Globale AND o Global OR. Questo determina il modo in cui vengono combinate più condizioni di filtro. Tutte le condizioni sono combinate usando le operazioni AND o OR. Se hai una condizione di filtro singolo, puoi sceglierne una delle due.

5. Seleziona il pulsante Add condition (Aggiungi condizione) nella sezione Filter condition (Condizione di filtro) per aggiungere una condizione di filtro.

   Nel campo Key (Chiave), scegli il nome di una chiave di proprietà dal set di dati. Nel campo Operation (Operazione), seleziona l'operatore di confronto. Nel campo Value (Valore), inserisci il valore di confronto. Di seguito sono riportate alcuni esempi di condizioni di filtro.

   - `year >= 2018`
   - `State matches 'CA*'`

   Quando si filtrano i valori di stringa, è necessario assicurarsi che il valore di confronto utilizzi un formato di espressione regolare che corrisponda al linguaggio di script selezionato nelle proprietà del processo (Python o Scala).
6. Aggiungi ulteriori condizioni di filtro, se necessario.
7. (Facoltativo) Dopo aver configurato le proprietà del nodo di trasformazione, puoi visualizzare lo schema modificato per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.
8. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

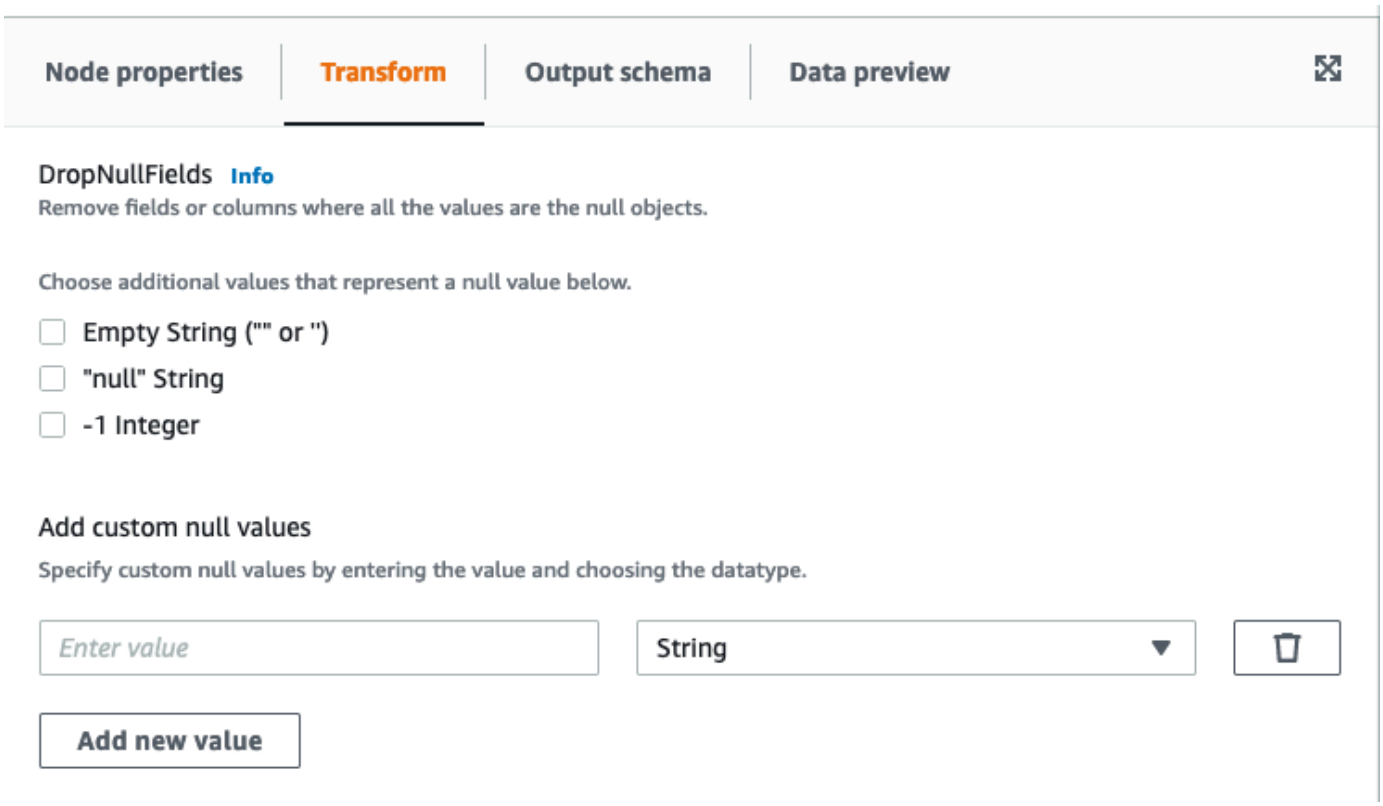
Utilizzo di DropNullFields per rimuovere campi con valori nulli

Utilizzo della trasformazione DropNullFields per rimuovere i campi dal dataset se tutti i valori nel campo sono "null". Per impostazione predefinita, AWS Glue Studio riconoscerà gli oggetti nulli, ma alcuni valori come stringhe vuote, stringhe "null", -1 interi o altri segnaposto come zeri, non verranno riconosciuti automaticamente come null.

Come utilizzare DropNullFields

1. Aggiunta di un nodo DropNullFields al diagramma del processo.

2. Nella scheda Node properties (Proprietà del nodo), scegli valori aggiuntivi che rappresentano un valore nullo. È possibile scegliere di selezionare nessuno o tutti i valori:



   - Stringa vuota ("" or ''): i campi che contengono stringhe vuote verranno eliminati
   - "stringa null": i campi che contengono la stringa con la parola "null" verranno eliminati
   - -1 numero intero: i campi che contengono un numero intero -1 (negativo) verranno eliminati
3. Se necessario, è anche possibile specificare valori nulli personalizzati. Si tratta di valori nulli che potrebbero essere univoci per il set di dati. Per aggiungere un valore nullo personalizzato, scegli Add new value (Aggiungi nuovo valore).
4. Inserisci il valore nullo personalizzato. Ad esempio, questo può essere zero o qualsiasi valore utilizzato per rappresentare un valore nullo nel set di dati.
5. Scegli il tipo di dati nel campo a discesa. I tipi di dati possono essere String o Integer.

> **Note**
>
> I valori nulli personalizzati e i relativi tipi di dati devono corrispondere esattamente, affinché i campi vengano riconosciuti come valori nulli e vengano quindi eliminati.

Corrispondenze parziali in cui solo il valore nullo personalizzato corrisponde, ma il tipo di dati non comporta l'eliminazione dei campi.

Utilizzo di una query SQL per trasformare i dati

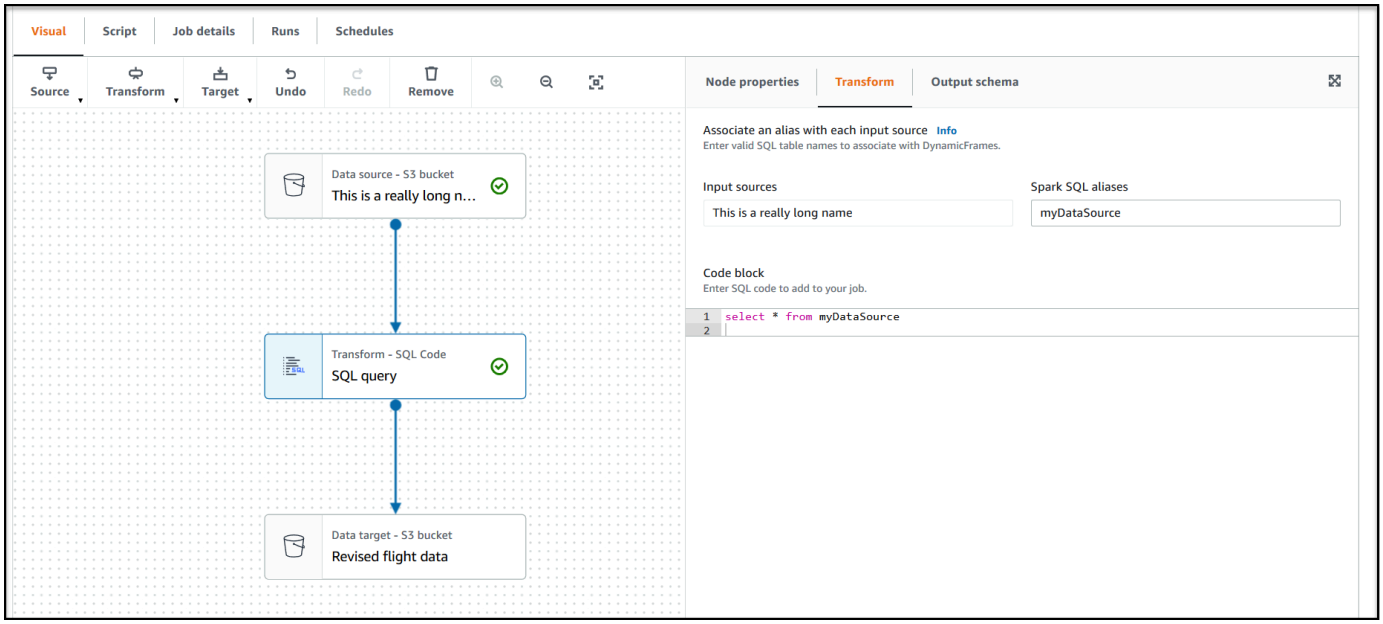
Puoi utilizzare una trasformazione SQL per scrivere la tua trasformazione sotto forma di query SQL.

Un nodo di trasformazione SQL può avere più set di dati come input, ma produce solo un singolo set di dati come output. Contiene un campo di testo, in cui puoi inserire la query Apache SparkSQL. Puoi assegnare alias a ciascun set di dati utilizzato come input, in modo da semplificare la query SQL. Per ulteriori informazioni sulla sintassi SQL, consulta la documentazione di Spark SQL.

> **Note**
>
> Se utilizzi una trasformazione SQL Spark con un'origine dati situata in un VPC, aggiungi un endpoint VPC AWS Glue al VPC che contiene l'origine dati. Per ulteriori informazioni sulla configurazione degli endpoint di sviluppo, consulta Aggiunta di un endpoint di sviluppo, Impostazione dell'ambiente per endpoint di sviluppo e Accesso all'endpoint di sviluppo nella Guida per gli sviluppatori di AWS Glue.

Per aggiungere un nodo di trasformazione SQL al diagramma di processo

1. (Facoltativo) Aggiungi un nodo di trasformazione al diagramma di processo, se necessario. Scegli Spark SQL per il tipo di nodo.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, o se desideri più input per la trasformazione SQL, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione. Aggiungi nodi padre aggiuntivi in base alle esigenze.
3. Seleziona la scheda Transform (Trasformazione) nel pannello dei dettagli del nodo.
4. I set di dati di origine per la query SQL sono identificati dai nomi specificati nel campo Name (Nome) per ogni nodo. Se non vuoi utilizzare questi nomi o se i nomi non sono adatti per una query SQL, puoi associare un nome a ciascun set di dati. La console fornisce alias predefiniti, ad esempio `MyDataSource`.



Ad esempio, se un nodo padre per il nodo di trasformazione SQL è denominato `Rename Org PK field`, è possibile associare il nome `org_table` a questo set di dati. Questo alias può quindi essere utilizzato nella query SQL al posto del nome del nodo.

5. Nel campo di immissione testo sotto l'intestazione Code block (Blocco di codice), incolla o immetti la query SQL. Il campo di testo mostra la sintassi SQL evidenziata e i suggerimenti per le parole chiave.
6. Con il nodo di trasformazione SQL selezionato, scegli l'opzione Output schema (Schema di output), quindi scegli Edit (Modifica). Specifica le colonne e i tipi di dati che descrivono i campi di output della query SQL.

   Specifica lo schema utilizzando le azioni seguenti nella sezione Output schema (Schema di output) della pagina:

   - Per rinominare una colonna, posiziona il cursore nella casella di testo Key (Chiave) per la colonna (nota anche come field (campo) o property key (chiave di proprietà) e inserisci il nuovo nome.
   - Per modificare il tipo di dati per una colonna, seleziona il nuovo tipo di dati per la colonna dall'elenco a discesa.
   - Per aggiungere una nuova colonna di livello superiore allo schema, scegli l'opzione Overflow (⋯), quindi scegli Add root key (Aggiungi chiave root). Vengono aggiunte nuove colonne nella parte superiore dello schema.

- Per rimuovere una colonna dallo schema, scegli l'icona di eliminazione
  ( )
  all'estrema destra del nome della chiave.

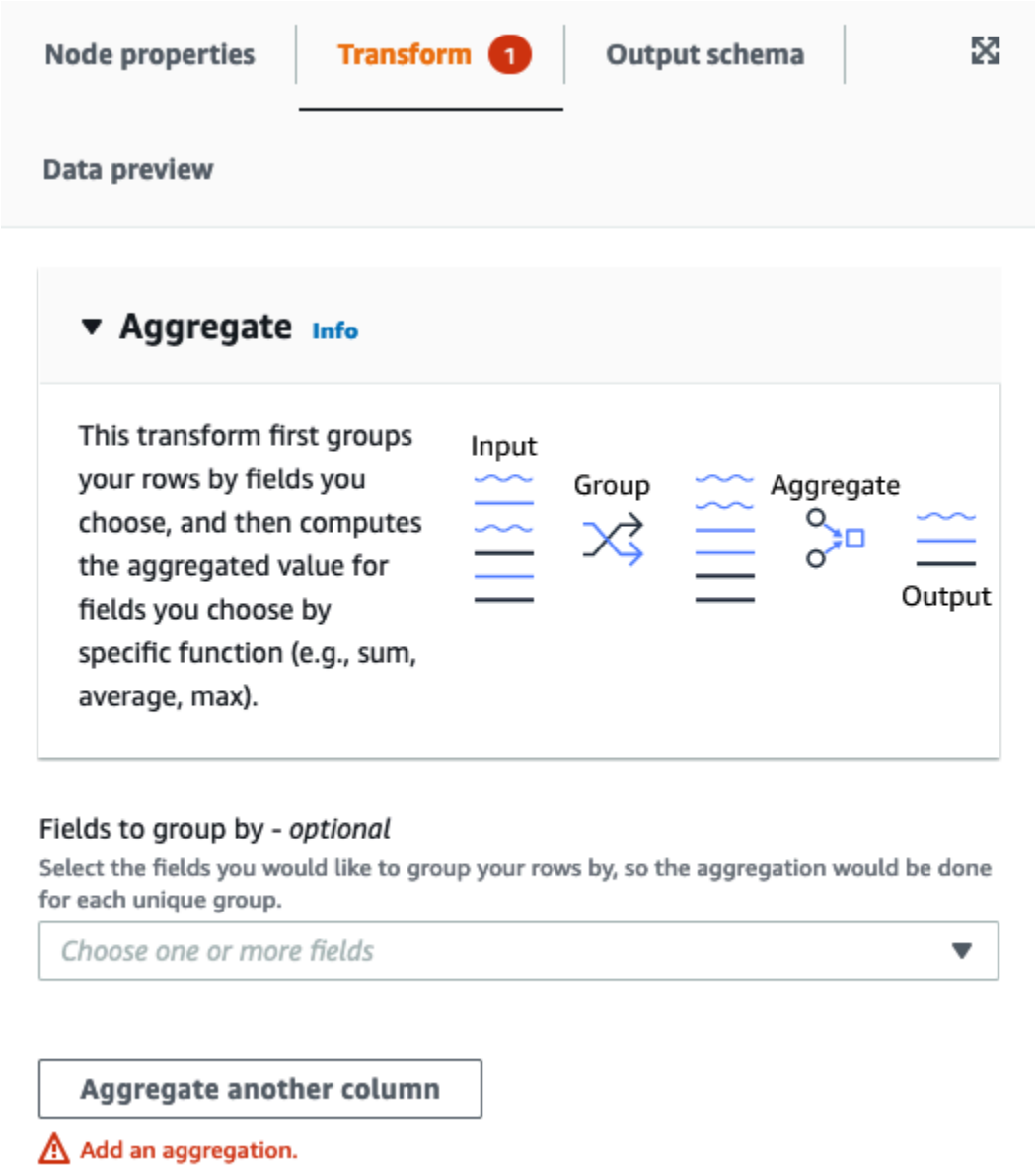
7. Una volta terminato di specificare lo schema di output, scegli Apply (Applica) per salvare le modifiche e uscire dall'editor dello schema. Se non vuoi salvare le modifiche, scegli Cancel (Annulla) per modificare l'editor dello schema.
8. (Facoltativo) Dopo aver configurato le proprietà del nodo e le proprietà di trasformazione, puoi visualizzare il set di dati modificato scegliendo la scheda Data preview (Anteprima dei dati) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Esiste un costo per l'utilizzo di questa caratteristica e la fatturazione inizia non appena si fornisce un ruolo IAM.

Utilizzo di Aggregate per eseguire calcoli di riepilogo sui campi selezionati

Utilizzo della trasformazione Aggregate

1. Aggiungi il nodo Aggregate al diagramma del processo.
2. Nella scheda Node properties (Proprietà del nodo), scegli i campi da raggruppare selezionando il campo a discesa (facoltativo). È possibile selezionare più di un campo alla volta o cercare il nome del campo digitando nella barra di ricerca.

   Quando i campi sono selezionati, vengono visualizzati il nome e il tipo di dati. Per eliminare un campo, selezionare "X" sul campo.



3. Scegli Aggregate another column (Aggrega un'altra colonna). È necessario selezionare almeno un campo.

4. Scegli un campo nel menu a discesa Field to aggregate (Campo da aggregare).

5. Scegli la funzione di aggregazione da applicare al campo scelto:

   - avg. calcola la media
   - countDistinct: calcola il numero di valori univoci non nulli
   - count: calcola il numero di valori non null
   - first: restituisce il primo valore che soddisfa i criteri "raggruppa per"
   - last: restituisce l'ultimo valore che soddisfa i criteri "raggruppa per"
   - kurtosis: calcola la nitidezza del picco di una curva di distribuzione della frequenza
   - max: restituisce il valore più alto che soddisfa i criteri "raggruppa per"
   - min: restituisce il valore più basso che soddisfa i criteri "raggruppa per"
   - skewness: misura l'asimmetria della distribuzione di probabilità di una distribuzione normale
   - stddev_pop: calcola la deviazione standard della popolazione e restituisce la radice quadrata della varianza di popolazione
   - sum: la somma di tutti i valori del gruppo
   - SumDistinct: la somma di valori distinti nel gruppo
   - var_samp: la varianza campione del gruppo (ignora i valori nulli)
   - var_pop: la varianza di popolazione del gruppo (ignora i valori nulli)

Appiattimento di strutture annidate

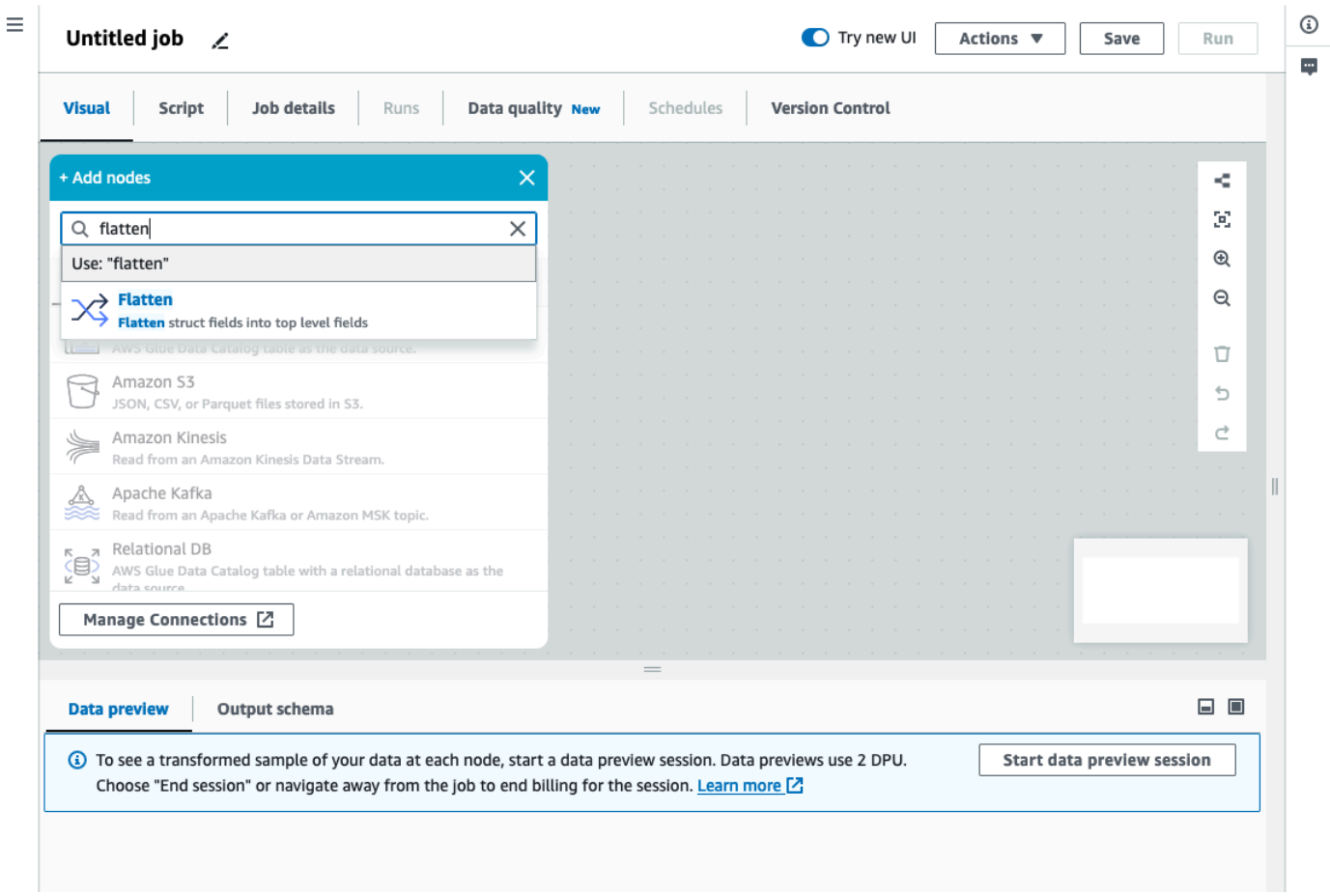
Appiattisci i campi delle strutture annidate nei dati, in modo che diventino campi di primo livello. I nuovi campi vengono denominati utilizzando il nome del campo preceduto dai nomi dei campi della struttura per raggiungerlo, separati da punti.

Ad esempio, prendiamo un caso in cui i dati hanno un campo di tipo Struct denominato "phone_numbers", che tra gli altri campi ne ha uno di tipo Struct denominato "home_phone" con due campi: "country_code" e "number". Una volta appiattiti, questi due campi diventeranno campi di primo livello denominati rispettivamente "phone_numbers.home_phone.country_code" e "phone_numbers.home_phone.number".

Aggiunta di un nodo di trasformazione Appiattisci nel diagramma di processo

1. Apri il pannello Risorse, scegli la scheda Trasformazioni, quindi Appiattisci per aggiungere una nuova trasformazione al diagramma di processo. È inoltre possibile digitare "Appiattisci" nella

barra di ricerca e poi fare clic sul nodo Appiattisci. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.



2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. (Facoltativo) Nella scheda Trasforma, puoi limitare l'appiattimento del livello massimo di annidamento. Ad esempio, se si imposta tale valore su 1 significa che solo le strutture di primo livello verranno appiattite. Impostando il valore massimo su 2 verrà appiattito il primo livello e le strutture direttamente sottostanti.

Aggiunta di una colonna UUID

Quando aggiungi una colonna UUID (Universally Unique Identified), a ogni riga verrà assegnata una stringa univoca di 36 caratteri.

Aggiunta di un nodo di trasformazione UUID al diagramma di processo

1. Apri il pannello Risorse, quindi scegli UUID per aggiungere una nuova trasformazione al diagramma di processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. (Facoltativo) Nella scheda Trasforma, puoi personalizzare il nome della nuova colonna. Per impostazione predefinita, si chiamerà "uuid".

Aggiunta di una colonna identificativa

Assegna un Identificatore numerico per ogni riga del set di dati.

Aggiunta di un nodo di trasformazione Identificatore nel diagramma di processo

1. Apri il pannello Risorse, quindi scegli Identificatore per aggiungere una nuova trasformazione al diagramma di processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. (Facoltativo) Nella scheda Trasforma, puoi personalizzare il nome della nuova colonna. Per impostazione predefinita, verrà denominata "id".
4. (Facoltativo) Se il processo elabora e archivia i dati in modo incrementale, è necessario evitare che gli stessi ID vengano riutilizzati tra le esecuzioni del processo.

   Nella scheda Trasforma, seleziona l'opzione della casella di controllo Univoco. Includerà il timestamp del processo nell'identificatore, rendendolo univoco tra più esecuzioni. Per consentire l'inserimento di un numero maggiore, la colonna anziché di tipo lungo sarà decimale.

Conversione di una colonna in tipo timestamp

È possibile utilizzare la trasformazione A timestamp per modificare il tipo di dati di una colonna numerica o di stringa in un timestamp, in modo da consentirne l'archiviazione con quel tipo di dati o l'applicazione ad altre trasformazioni che richiedono un timestamp.

Aggiunta di un nodo di trasformazione A timestamp nel diagramma di processo

1. Apri il pannello Risorse, quindi scegli A timestamp per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, inserisci il nome della colonna da convertire.
4. Nella scheda Trasforma, definisci come analizzare la colonna selezionata scegliendo il tipo.

   Se il valore è un numero, può essere espresso in secondi (timestamp Unix/Python), millisecondi o microsecondi, scegli l'opzione corrispondente.

   Se il valore è una stringa formattata, scegli il tipo "iso"; la stringa deve essere conforme a una delle varianti del formato ISO, ad esempio: "2022-11-02T14:40:59.915Z".

   Se a questo punto non conosci il tipo oppure righe diverse utilizzano tipi diversi, puoi scegliere "rilevamento automatico" e il sistema genererà la sua ipotesi migliore, a un costo di prestazioni ridotto.
5. (Facoltativo) Nella scheda Trasforma, invece di convertire la colonna selezionata, puoi crearne una nuova e mantenere l'originale inserendo un nome per la nuova colonna.

Conversione di una colonna di timestamp in una stringa formattata

Formatta una colonna di timestamp in una stringa in base a uno schema. Puoi utilizzare Formatta timestamp per ottenere data e ora come stringa con il formato desiderato. Puoi definire il formato utilizzando la sintassi della data Spark e la maggior parte dei codici di data Python.

Ad esempio, se desideri che la stringa della data sia formattata come "2023-01-01 00:00", puoi definire tale formato usando la sintassi Spark come "yyyy-MM-dd HH:mm" o i codici di data Python equivalenti come "%Y-%m-%d %H:%M"

Aggiunta di un nodo di trasformazione Formatta timestamp nel diagramma di processo

1. Apri il pannello Risorse, quindi scegli Formatta timestamp per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.

2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, inserisci il nome della colonna da convertire.
4. Nella scheda Trasforma, inserisci il modello di formato timestamp da utilizzare, espresso utilizzando la sintassi della data Spark o i codici di data Python.
5. (Facoltativo) Nella scheda Trasforma, invece di convertire la colonna selezionata, puoi crearne una nuova e mantenere l'originale inserendo un nome per la nuova colonna.

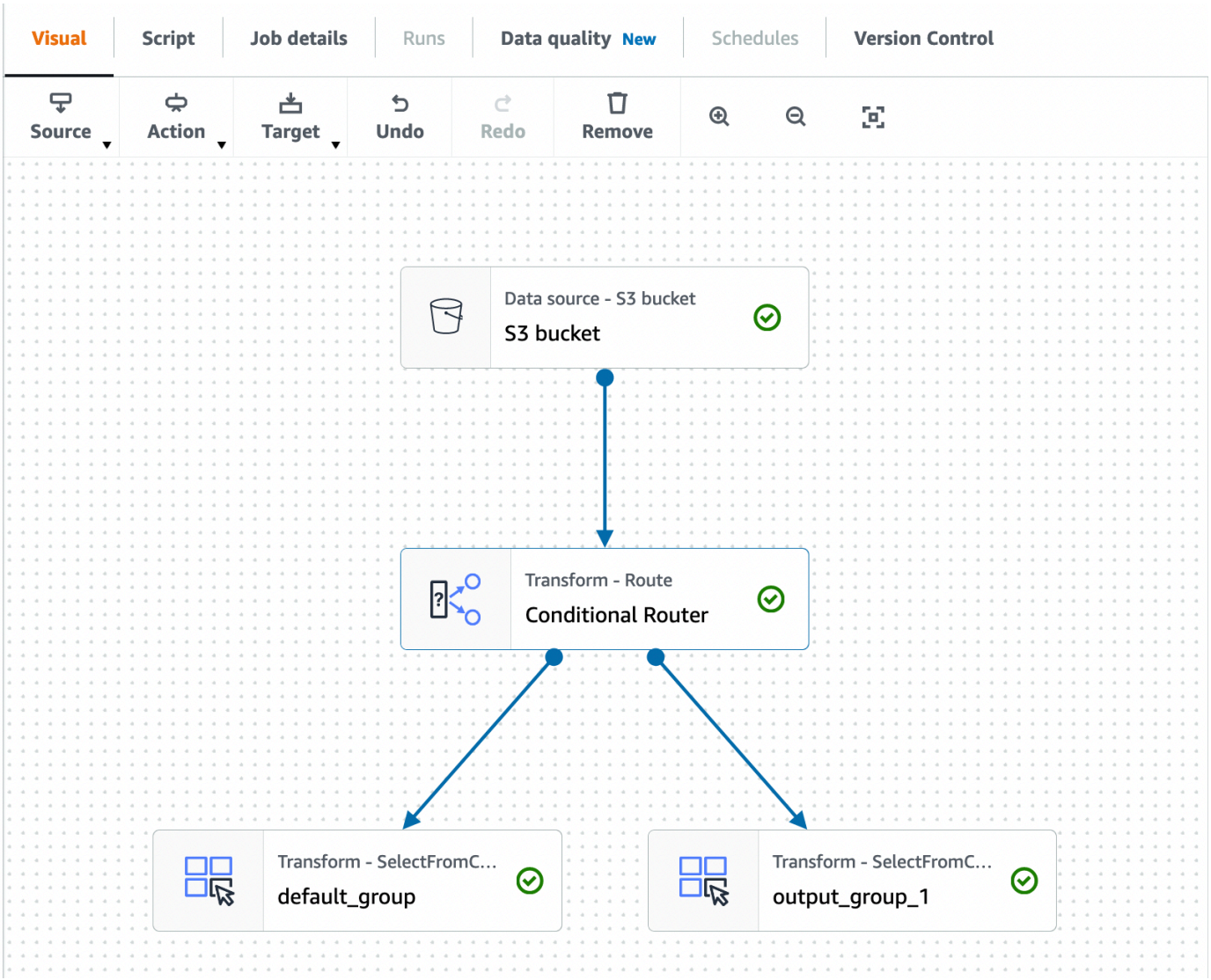
Creazione di una trasformazione router condizionale

La trasformazione router condizionale consente di applicare più condizioni ai dati in ingresso. Ogni riga dei dati in ingresso viene valutata in base a una condizione di filtro di gruppo ed elaborata nel gruppo corrispondente. Se una riga soddisfa più di una condizione di filtro di gruppo, la trasformazione passa la riga a più gruppi. Se una riga non soddisfa alcuna condizione, può essere eliminata o indirizzata a un gruppo di output predefinito.

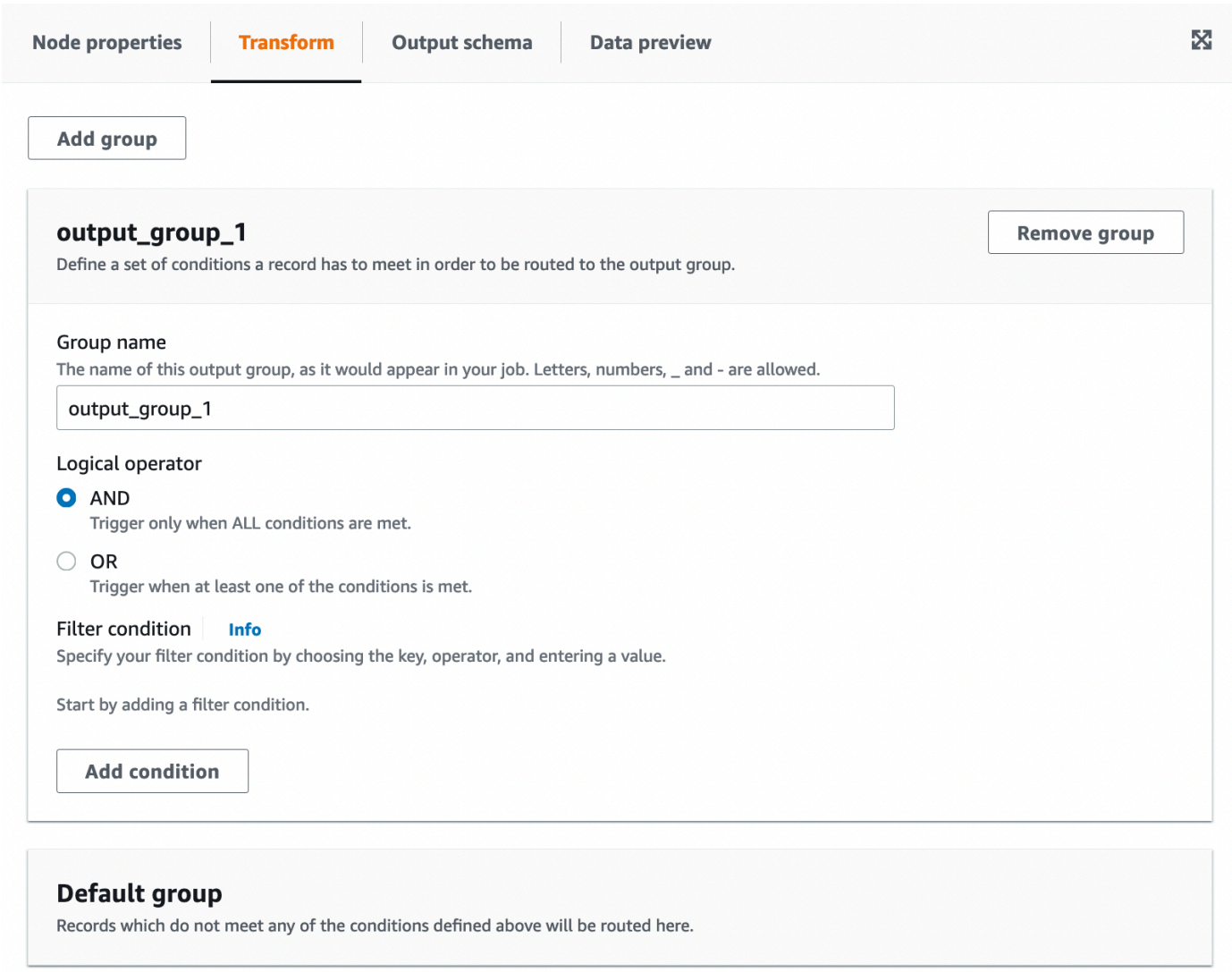
Questa trasformazione è simile alla trasformazione di filtro, ma utile per gli utenti che desiderano testare gli stessi dati di input su più condizioni.

Per aggiungere una trasformazione router condizionale:

1. Scegli un nodo in cui eseguire la trasformazione router condizionale. Può essere un nodo di origine o un'altra trasformazione.
2. Scegli Azione, quindi usa la barra di ricerca per trovare e scegliere "Router condizionale". Viene aggiunta una trasformazione Router condizionale insieme a due nodi di output. Un nodo di output, "Gruppo predefinito", contiene record che non soddisfano nessuna delle condizioni definite negli altri nodi di output. Il gruppo predefinito non può essere modificato.

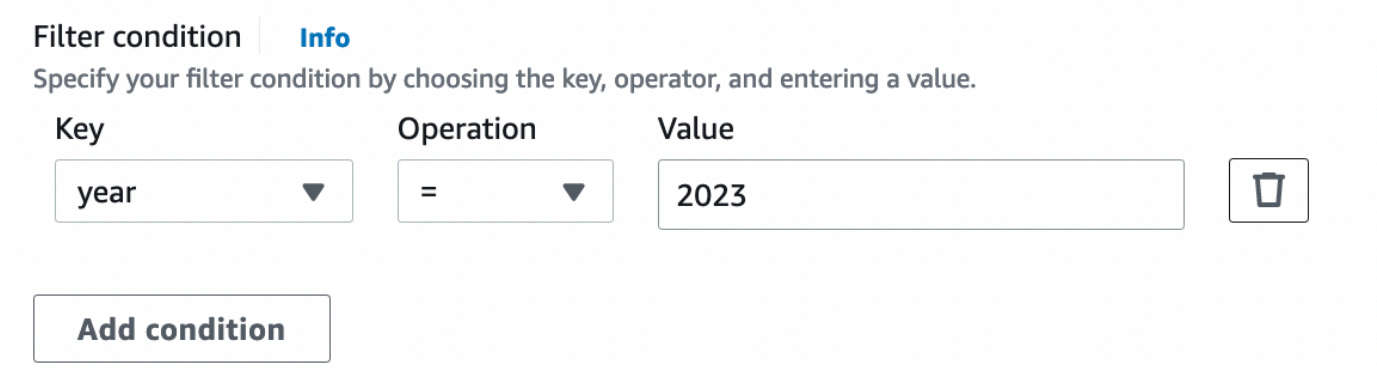


Puoi aggiungere altri gruppi di output scegliendo Aggiungi gruppo. Per ogni gruppo di output, è possibile assegnare un nome al gruppo e aggiungere condizioni di filtro e un operatore logico.



3. Rinomina il nome del gruppo di output inserendo un nuovo nome per il gruppo. AWS Glue Studio assegnerà automaticamente un nome ai tuoi gruppi (ad esempio, "output_group_1").
4. Scegli un operatore logico (AND, OR) e aggiungi una condizione di filtro specificando la chiave, l'operazione e il valore. Gli operatori logici consentono di implementare più di una condizione di filtro ed eseguire l'operatore logico su ogni condizione di filtro specificata.

   Quando si specifica la chiave, è possibile scegliere tra le chiavi disponibili nello schema. È quindi possibile scegliere l'operazione disponibile in base al tipo di chiave selezionata. Ad esempio, se il tipo di chiave è "stringa", l'operazione disponibile tra cui scegliere è "corrispondenze".



5. Inserisci il valore nel campo Valore. Per aggiungere condizioni di filtro aggiuntive, scegli Aggiungi condizione . Per rimuovere condizioni di filtro, scegli l'icona del cestino.

Utilizzo della trasformazione Concatena colonne per aggiungere colonne

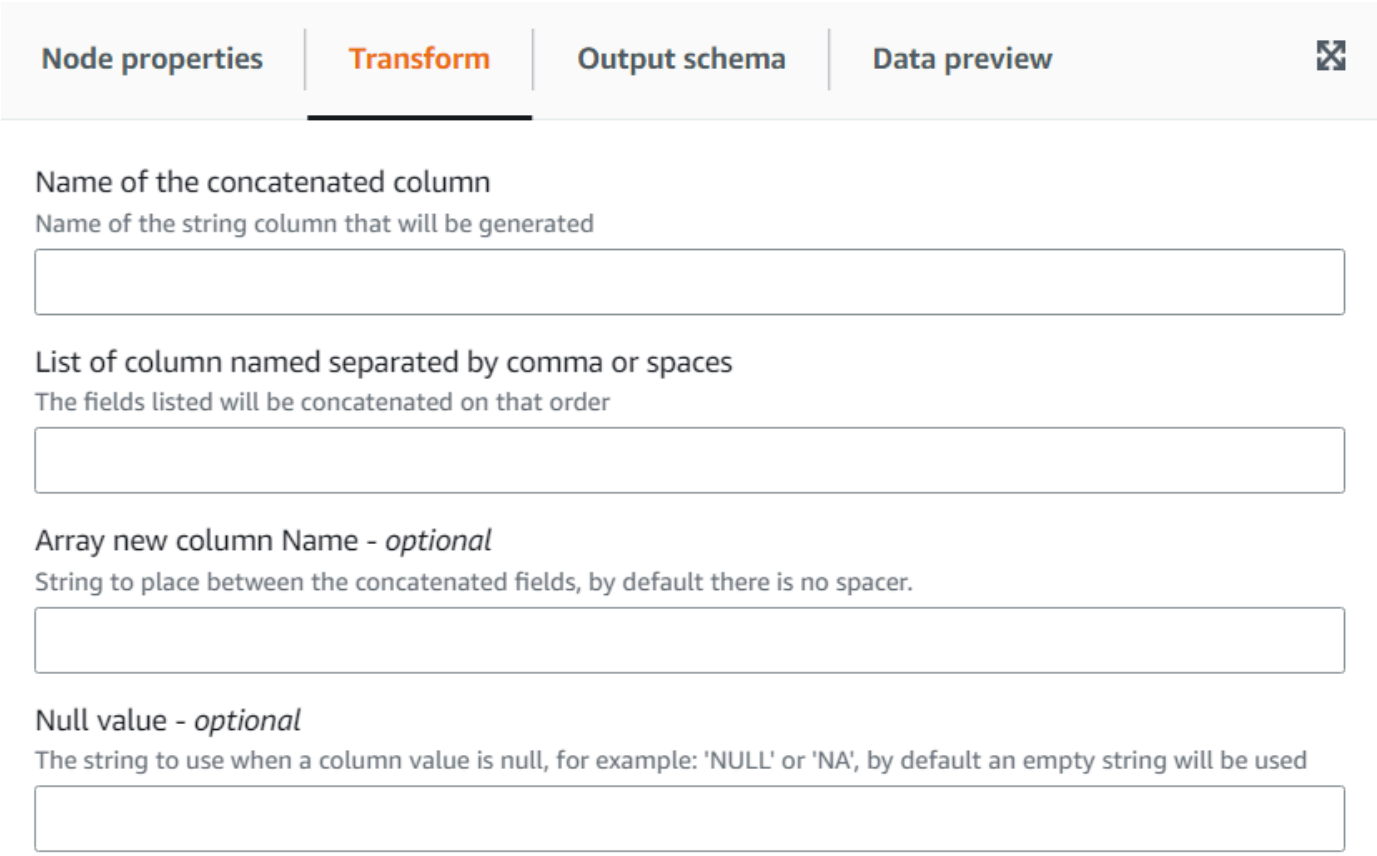
La trasformazione Concatena consente di creare una nuova colonna di stringhe utilizzando i valori di altre colonne con un distanziatore opzionale. Ad esempio, se definiamo una colonna concatenata "data" come concatenazione di "anno", "mese" e "giorno" (in quest'ordine) con "-" come spaziatore, otterremmo:

giorno	mese	anno	date
01	01	2020	2020-11-01
02	01	2020	02-07-2020
03	01	2020	2020-06-03
04	01	2020	2020-11-04

Per aggiungere una trasformazione Concatena:

1. Apri il pannello Risorse. Quindi, scegli Concatena colonne per aggiungere una nuova trasformazione al diagramma di processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.

3. Nella scheda Trasforma, inserisci il nome della colonna che conterrà la stringa concatenata e le colonne da concatenare. L'ordine in cui selezioni le colonne nel menu a discesa sarà l'ordine utilizzato.



4. Spaziatore - facoltativo: inserisci una stringa da inserire tra i campi concatenati. Per impostazione predefinita, non sono previsti spaziatori.
5. Valore nullo - facoltativo: inserisci una stringa da utilizzare quando il valore di una colonna è nullo. Per impostazione predefinita, nei casi in cui le colonne hanno il valore "NULL" o "NA", viene utilizzata una stringa vuota.

Utilizzo della trasformazione Dividi stringa per suddividere una colonna di stringhe

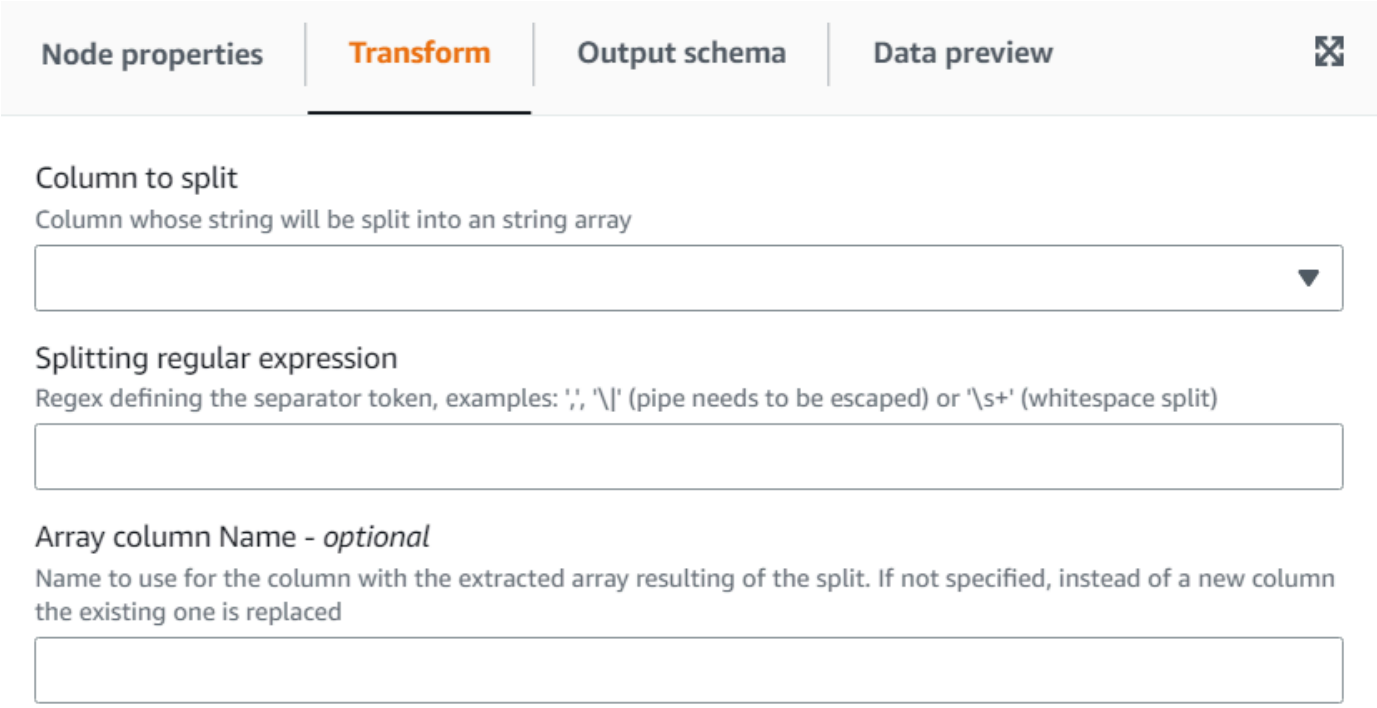
La trasformazione Dividi stringa consente di suddividere una stringa in un array di token utilizzando un'espressione regolare per definire come viene eseguita la suddivisione. È quindi possibile mantenere la colonna come tipo array o applicare una trasformazione Array a colonne successivamente a questa per estrarre i valori dell'array in campi di primo livello, supponendo che ogni token abbia un significato che conosciamo in precedenza. Inoltre, se l'ordine dei token è irrilevante (ad esempio, un insieme di categorie), è possibile utilizzare la trasformazione Espandi per generare una riga separata per ogni valore.

Ad esempio, è possibile dividere una colonna "categories" utilizzando una virgola come modello per aggiungere una colonna "categories_arr".

product_id	categories	categories_arr
1	sport,inverno	[sport, inverno]
2	giardino, attrezzi	[giardino, attrezzi]
3	videogiochi	[videogiochi]
4	gioco,gioco da tavolo,gioco di società	[gioco, gioco da tavolo, gioco di società]

Per aggiungere una trasformazione Dividi stringa:

1. Apri il pannello Risorse, quindi scegli Dividi stringa per aggiungere una nuova trasformazione al diagramma di processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, scegli la colonna da dividere e inserisci il modello da utilizzare per dividere la stringa. Nella maggior parte dei casi puoi semplicemente inserire i caratteri, a meno che non abbiano un significato speciale come espressione regolare e debbano contenere caratteri di escape. I caratteri che richiedono escape sono `\.[]{}()<>*+-=!?^$|` e occorre aggiungere una barra rovesciata davanti al carattere. Ad esempio, se vuoi utilizzare un punto (".") come separatore, devi inserire `\.`. Tuttavia, la virgola non ha un significato speciale e può essere specificata così com'è: `,`.



4. (Facoltativo) Se desideri mantenere la colonna di stringhe originale, puoi inserire un nome per una nuova colonna di array: potrai così mantenere sia la colonna di stringhe originale sia la nuova colonna di array tokenizzata.

Utilizzo della trasformazione Array a colonne per estrarre gli elementi di un array in colonne di primo livello

La trasformazione Array a colonne consente di estrarre alcuni o tutti gli elementi di una colonna di tipo array in nuove colonne. La trasformazione riempirà le nuove colonne il più possibile se l'array ha un numero sufficiente di valori da estrarre, prendendo facoltativamente gli elementi nelle posizioni specificate.

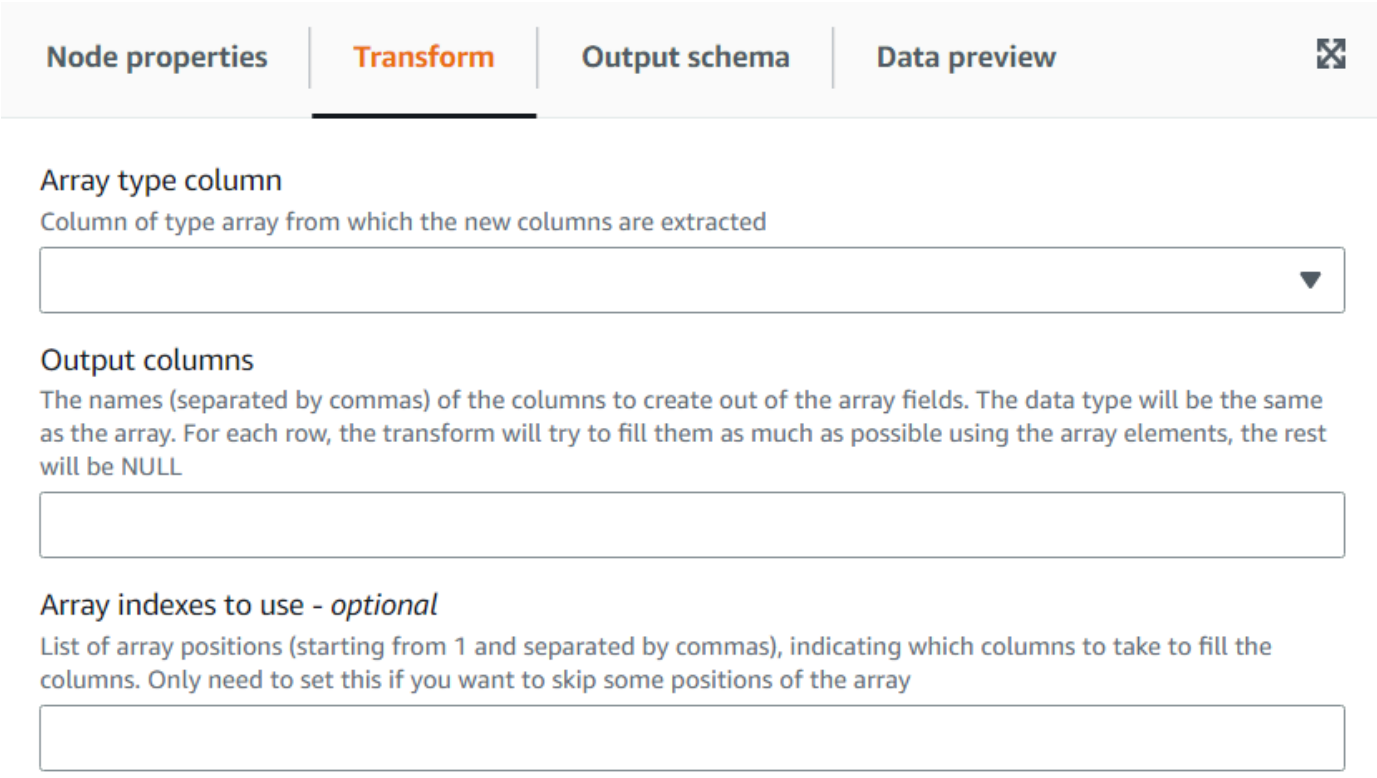
Ad esempio, se hai una colonna di array "subnet", che è il risultato dell'applicazione della trasformazione "Dividi stringa" su una sottorete ip v4, puoi estrarre la prima e la quarta posizione nelle nuove colonne "first_octect" e "fourth_octect". L'output della trasformazione in questo esempio sarebbe il seguente; nota che le ultime due righe hanno array più corti del previsto:

sottorete	first_octect	fourth_octect
[54, 240, 197, 238]	54	238

sottorete	first_octect	fourth_octect
[192, 168, 0, 1]	192	1
[192, 168]	192	
[]		

Per aggiungere una trasformazione Array a colonne:

1. Apri il pannello Risorse, quindi scegli Array a colonne per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, scegli la colonna dell'array da estrarre e inserisci l'elenco delle nuove colonne per i token estratti.



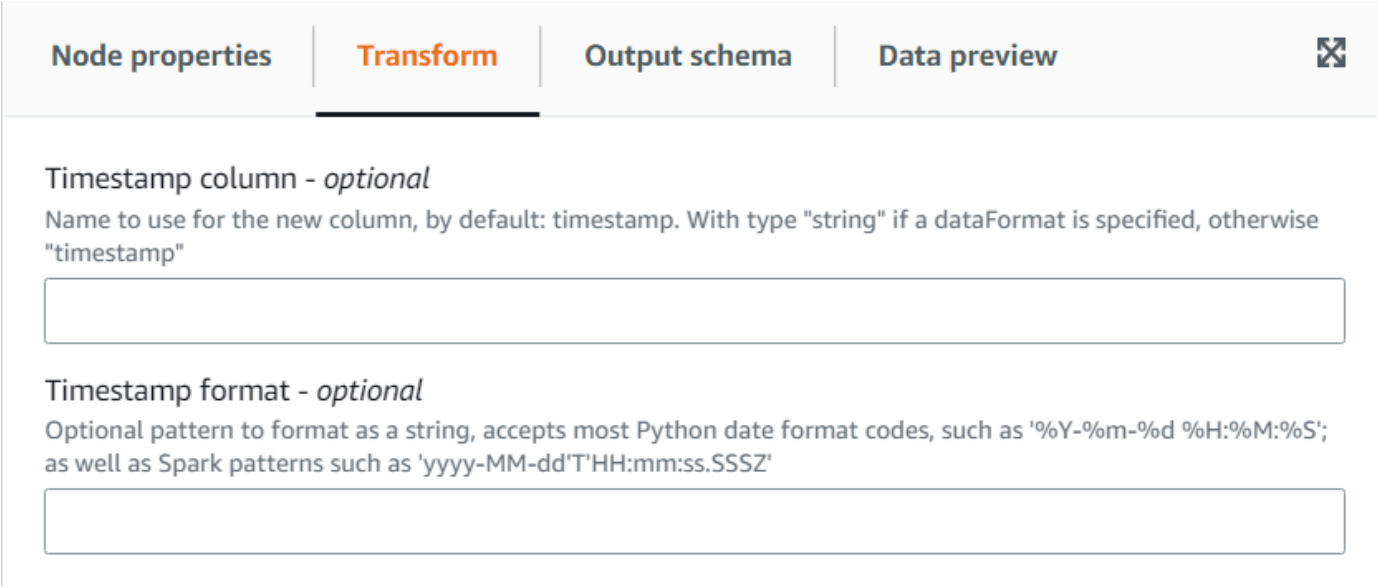
4. (Facoltativo) Se non vuoi prendere i token dell'array per assegnarli alle colonne, puoi specificare gli indici da prendere che verranno assegnati all'elenco di colonne nello stesso ordine specificato. Ad esempio, se le colonne di output sono "column1, column2, column3" e gli indici "4, 1, 3", il quarto elemento dell'array andrà alla column1, il primo alla column2 e il terzo alla column3 (se l'array è più corto del numero di indice, verrà impostato un valore NULL).

Utilizzo della trasformazione Aggiungi timestamp corrente

La trasformazione Aggiungi timestamp corrente consente di contrassegnare le righe con l'ora in cui i dati sono stati elaborati. Ciò è utile per scopi di controllo o per tenere traccia della latenza nella pipeline di dati. È possibile aggiungere questa nuova colonna come tipo di dati timestamp o come stringa formattata.

Per aggiungere una trasformazione Aggiungi timestamp corrente:

1. Apri il pannello Risorse, quindi scegli Aggiungi timestamp corrente per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.



3. (Facoltativo) Nella scheda Trasforma, inserisci un nome personalizzato per la nuova colonna e un formato se preferisci che la colonna sia una stringa di data formattata.

Utilizzo della trasformazione Pivot: righe a colonne

La trasformazione Pivot: righe a colonne consente di aggregare una colonna numerica ruotando valori univoci su colonne selezionate che diventano nuove colonne. Se sono selezionate più colonne, i valori vengono concatenati per denominare le nuove colonne. In questo modo, le righe vengono consolidate pur avendo più colonne con aggregazioni parziali per ogni valore univoco. Ad esempio, se disponi di questo set di dati sulle vendite per mese e paese (ordinato per facilitare la comprensione):

anno	mese	country	amount
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64
2020	Feb	uk	67
2020	Feb	de	4
2020	Feb	de	7
2020	Feb	us	6
2020	Feb	us	12
2020	Jan	us	90

Se utilizzi quantità e paese come colonne di aggregazione, vengono create nuove colonne a partire dalla colonna paese originale. Nella tabella seguente sono presenti nuove colonne per de, uk e us anziché la colonna paese.

anno	mese	de	uk	us
2020	Jan	42	32	64
2020	Jan	11	67	18

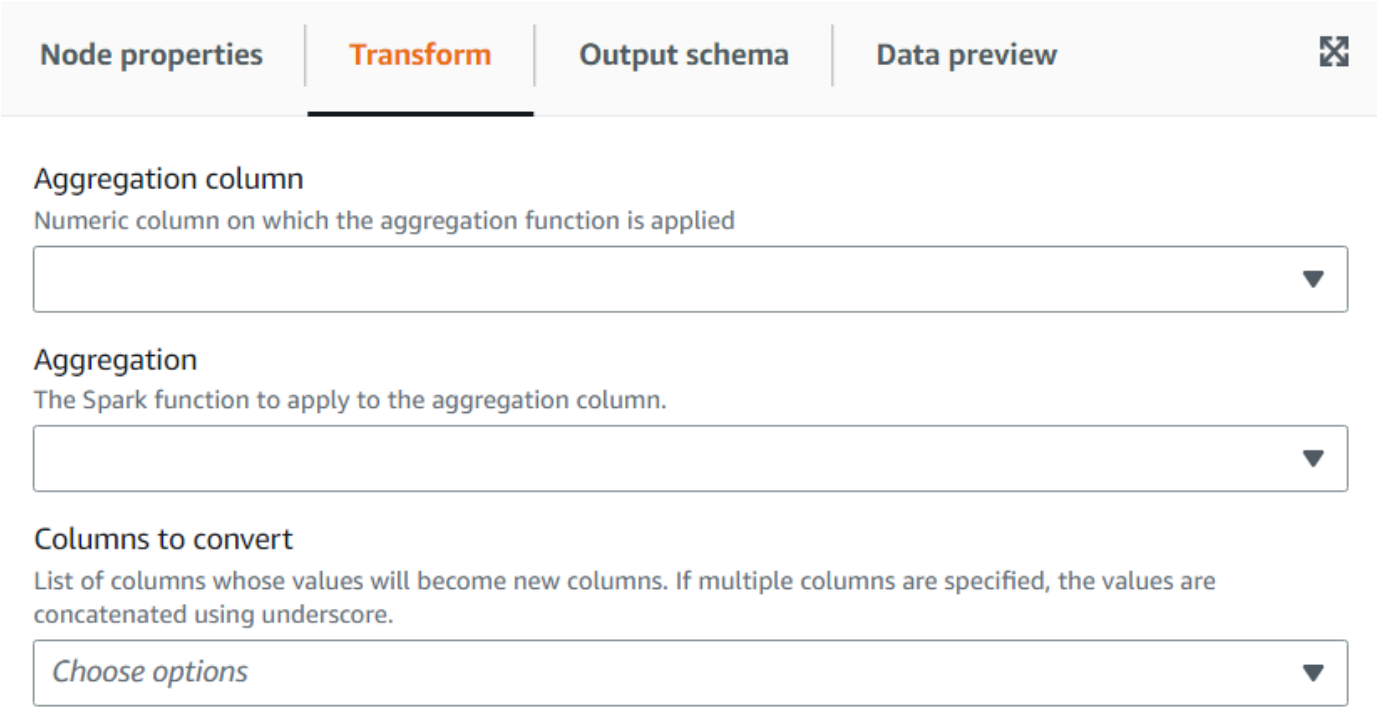
anno	mese	de	uk	us
2021	Jan			90

Se invece vuoi eseguire il pivot sia sul mese sia sul paese, otterrai una colonna per ogni combinazione dei valori di tali colonne:

anno	Jan_de	Jan_uk	Jan_us	Feb_de	Feb_uk	Feb_us
2020	42	32	64	11	67	18
2021			90			

Per aggiungere una trasformazione Pivot: righe a colonne:

1. Apri il pannello Risorse, quindi scegli Pivot: righe a colonne per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, sceglie la colonna numerica che verrà aggregata per produrre i valori per le nuove colonne, la funzione di aggregazione da applicare e le colonne per convertire i valori univoci in nuove colonne.



Utilizzo della trasformazione Elimina pivot: colonne a righe

La trasformazione Elimina pivot consente di convertire le colonne in valori di nuove colonne generando una riga per ogni valore univoco. È l'opposto del pivot, ma tieni presente che non è equivalente, in quanto non può separare le righe con valori identici che sono state aggregate o suddividere le combinazioni nelle colonne originali. Per fare queste operazioni, puoi utilizzare in seguito una trasformazione Dividi. Ad esempio, in presenza della tabella seguente:

anno	mese	de	uk	us
2020	Jan	42	32	64
2020	Feb	11	67	18
2021	Jan			90

È possibile eliminare il pivot dalle colonne "de", "uk" e "us" in una colonna "country" con il valore "amount" e ottenere quanto segue (ordinato qui a scopo illustrativo):

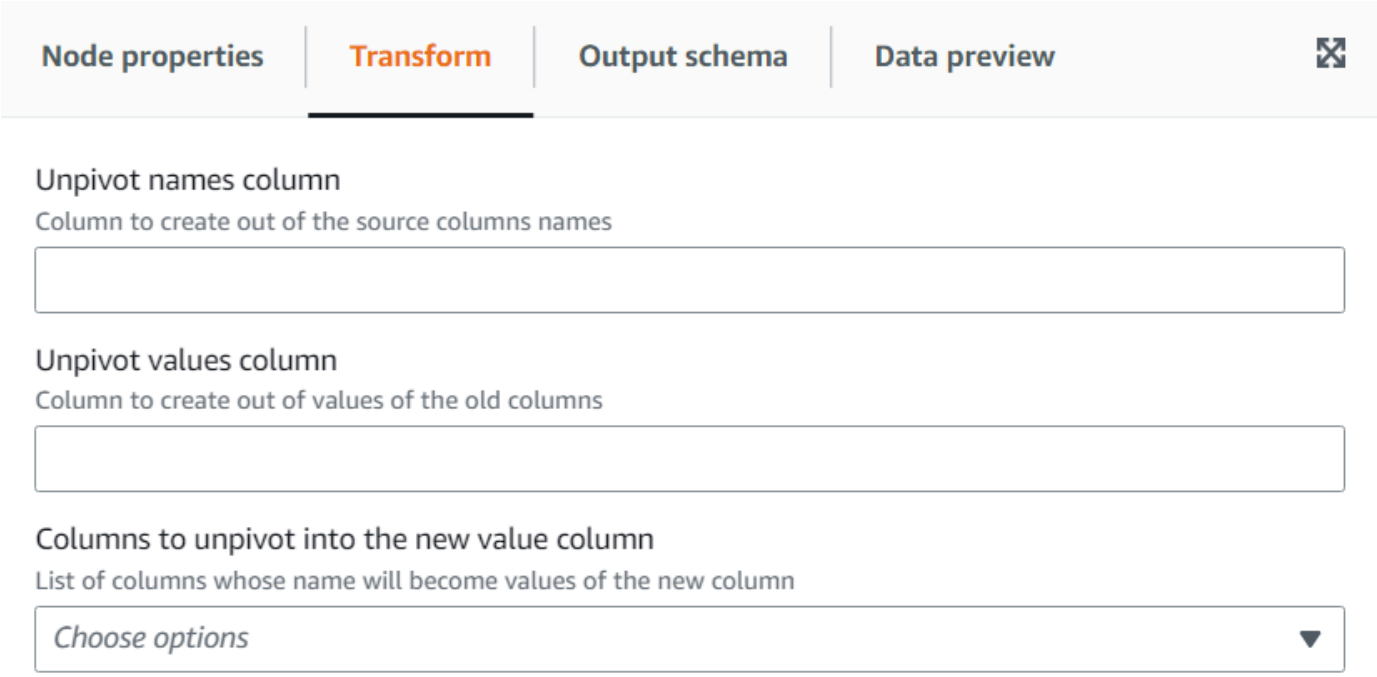
anno	mese	country	amount
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64
2020	Feb	uk	67
2020	Feb	de	11
2020	Feb	us	18
2021	Jan	us	90

Nota che le colonne con un valore NULL ("de" e "uk" di gennaio 2021) non vengono generate per impostazione predefinita. È possibile abilitare questa opzione per ottenere:

anno	mese	country	amount
2020	Jan	uk	32
2020	Jan	de	42
2020	Jan	us	64
2020	Feb	uk	67
2020	Feb	de	11
2020	Feb	us	18
2021	Jan	us	90
2021	Jan	de	
2021	Jan	uk	

Per aggiungere una trasformazione Elimina pivot: colonne a righe:

1. Apri il pannello Risorse, quindi scegli Elimina pivot: colonne a righe per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, inserisci le nuove colonne da creare per contenere i nomi e i valori delle colonne dalle quali intendi eliminare il pivot.

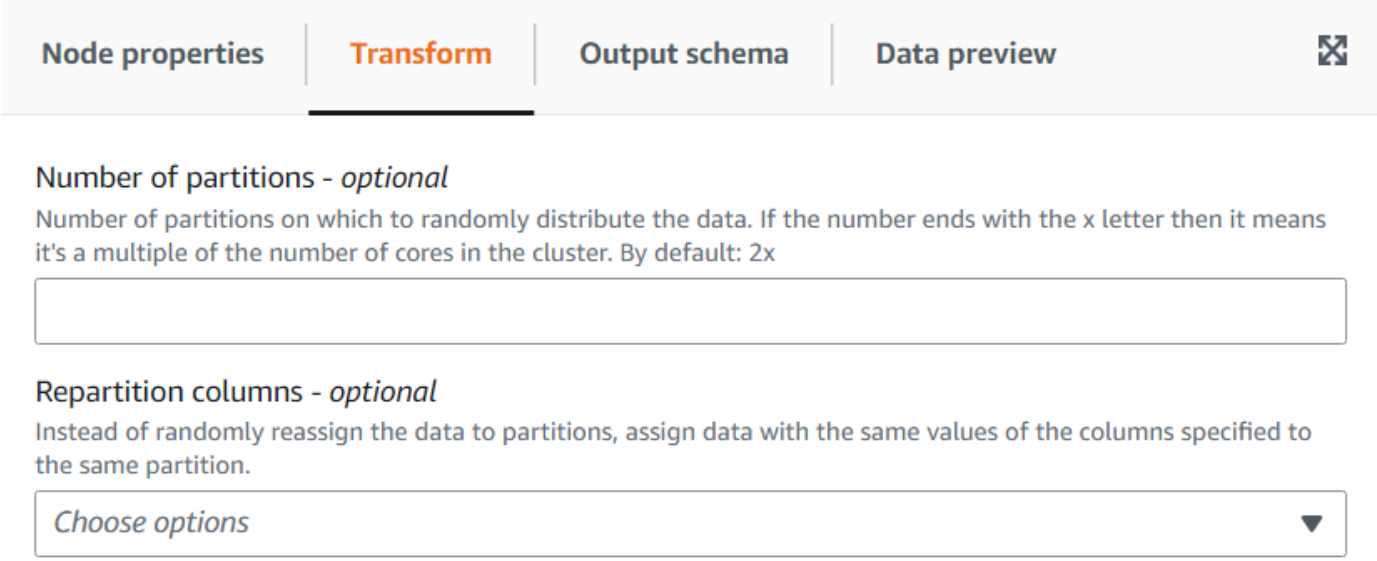


Utilizzo della trasformazione Bilancia automaticamente elaborazione per ottimizzare il runtime

La trasformazione Bilancia automaticamente elaborazione ridistribuisce i dati tra i worker per migliorare le prestazioni. Ciò è utile nei casi in cui i dati non sono bilanciati o, poiché provengono dall'origine, non consentono un'elaborazione parallela sufficiente. Questo è comune quando l'origine è compressa con gzip o è JDBC. La ridistribuzione dei dati ha un costo prestazionale modesto, quindi l'ottimizzazione potrebbe non sempre compensare tale sforzo se i dati fossero già ben bilanciati. A un livello più basso, la trasformazione utilizza la ripartizione Apache Spark per riassegnare in modo casuale i dati tra una serie di partizioni ottimali per la capacità del cluster. Per gli utenti esperti, è

possibile inserire una serie di partizioni manualmente. Inoltre, può essere utilizzato per ottimizzare la scrittura di tabelle partizionate riorganizzando i dati in base a colonne specificate. Ciò si traduce in file di output più consolidati.

1. Apri il pannello Risorse, quindi scegli Bilancia automaticamente elaborazione per aggiungere una nuova trasformazione al diagramma di processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. (Facoltativo) Nella scheda Trasforma è possibile inserire un numero di partizioni. In generale, si consiglia di lasciare che sia il sistema a decidere questo valore, tuttavia è possibile regolare il moltiplicatore o inserire un valore specifico se è necessario controllarlo. Se intendi salvare i dati partizionati per colonne, puoi scegliere le stesse colonne come colonne di ripartizione. In questo modo ridurrà al minimo il numero di file su ciascuna partizione ed eviterà di avere molti file per partizione, il che ostacolerebbe le prestazioni degli strumenti che eseguono query su tali dati.



Utilizzo della trasformazione Colonna derivata per combinare altre colonne

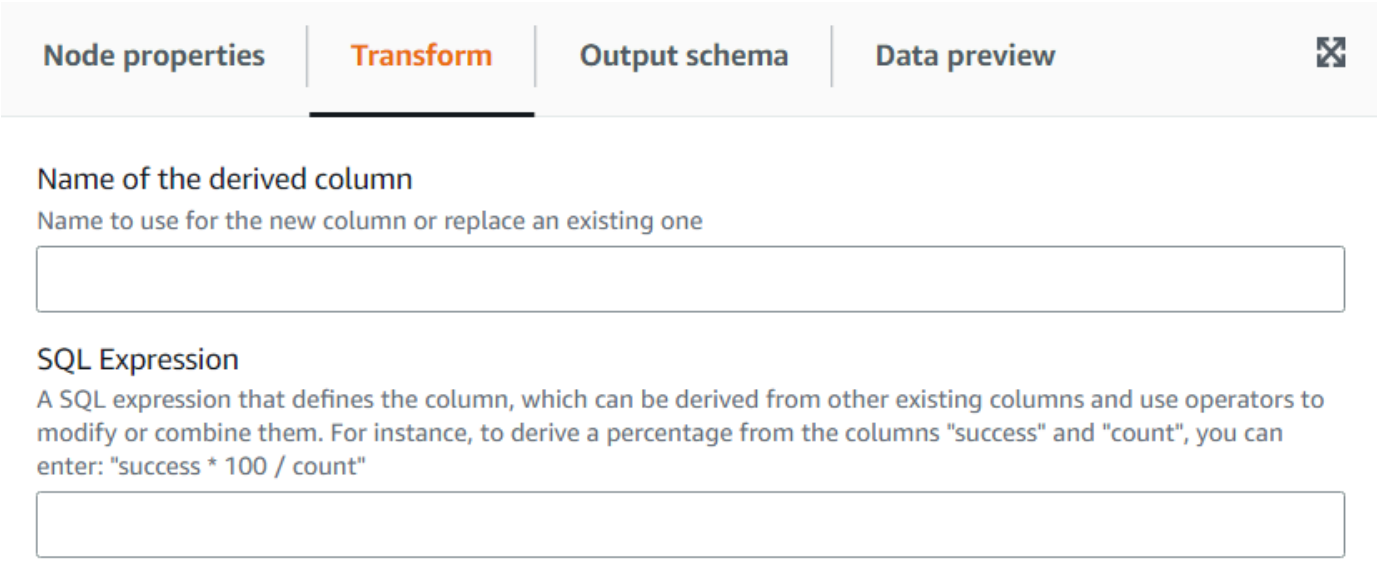
La trasformazione Colonna derivata consente di definire una nuova colonna basata su una formula matematica o un'espressione SQL in cui è possibile utilizzare altre colonne nei dati, oltre a costanti e valori letterali. Ad esempio, per ricavare una colonna "percentage" dalle colonne "success" e "count", puoi inserire l'espressione SQL: "success * 100/count || '%'".

Risultato dell'esempio:

success	count	percentage
14	100	14%
6	20	3%
3	40	7,5%

Per aggiungere una trasformazione Colonna derivata:

1. Apri il pannello Risorse, quindi scegli Colonna derivata per aggiungere una nuova trasformazione al diagramma di processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, inserisci il nome della colonna e l'espressione per il relativo contenuto.



Utilizzo della trasformazione Ricerca per aggiungere dati corrispondenti da una tabella di catalogo

La trasformazione Ricerca consente di aggiungere colonne da una tabella di catalogo definita quando le chiavi corrispondono alle colonne di ricerca definite nei dati. Ciò equivale a eseguire un join esterno sinistro tra i dati e la tabella di ricerca, utilizzando come condizioni le colonne corrispondenti.

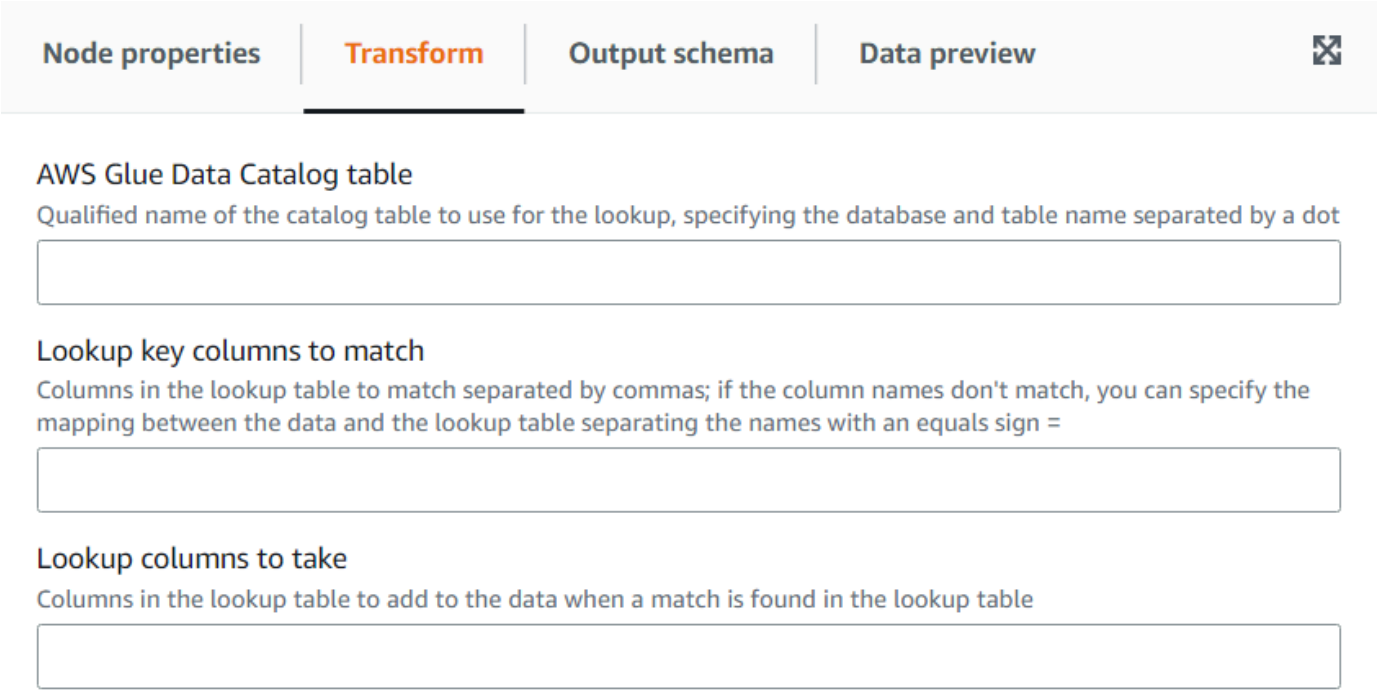
Per aggiungere una trasformazione Ricerca:

1. Apri il pannello Risorse, quindi scegli Ricerca per aggiungere una nuova trasformazione al diagramma di processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, inserisci il nome completo della tabella di catalogo da utilizzare per eseguire le ricerche. Ad esempio, se il tuo database è "mydb" e la tua tabella "mytable", inserisci "mydb.mytable". Inserisci quindi i criteri per trovare una corrispondenza nella tabella di ricerca, se la chiave di ricerca è composta. Inserisci l'elenco delle colonne chiave separate da virgole. Se una o più colonne chiave non hanno lo stesso nome, devi definire la mappatura delle corrispondenze.

   Ad esempio, se le colonne di dati sono "user_id" e "region" e nella tabella utenti le colonne corrispondenti sono denominate "id" e "region", nel campo Colonne da abbinare, inserisci: "user_id=id, region". Potresti anche utilizzare region=region, ma non è necessario poiché sono uguali.
4. Infine, inserisci le colonne da estrarre dalla riga corrispondente nella tabella di ricerca per incorporarle nei dati. Se non viene trovata alcuna corrispondenza, tali colonne verranno impostate su NULL.

> **Note**
>
> Sotto la trasformazione Ricerca, viene utilizzato un join a sinistra a fini di efficienza. Se la tabella di ricerca ha una chiave primaria composita, assicurati di impostare le colonne di corrispondenza in modo che includano tutte le colonne che fanno parte di tale chiave, così da ottenere una sola corrispondenza. Altrimenti, più righe nella tabella di ricerca

corrisponderanno, il che porterà ad aggiungere righe duplicate per ogni corrispondenza trovata.



Utilizzo della trasformazione Espandi array o mappa in righe

La trasformazione Espandi consente di estrarre valori da una struttura nidificata in singole righe più facili da manipolare. Nel caso di un array, la trasformazione genererà una riga per ogni valore dell'array, replicando i valori per le altre colonne della riga. Nel caso di una mappa, la trasformazione genererà una riga per ogni voce con la chiave e il valore come colonne più ogni altra colonna presente nella riga.

Ad esempio, se abbiamo questo set di dati che ha una colonna di array "categoria" con più valori.

product_id	category
1	[sport, inverno]
2	[giardino, attrezzi]
3	[videogiochi]

product_id	category
4	[gioco, gioco da tavolo, gioco di società]
5	[]

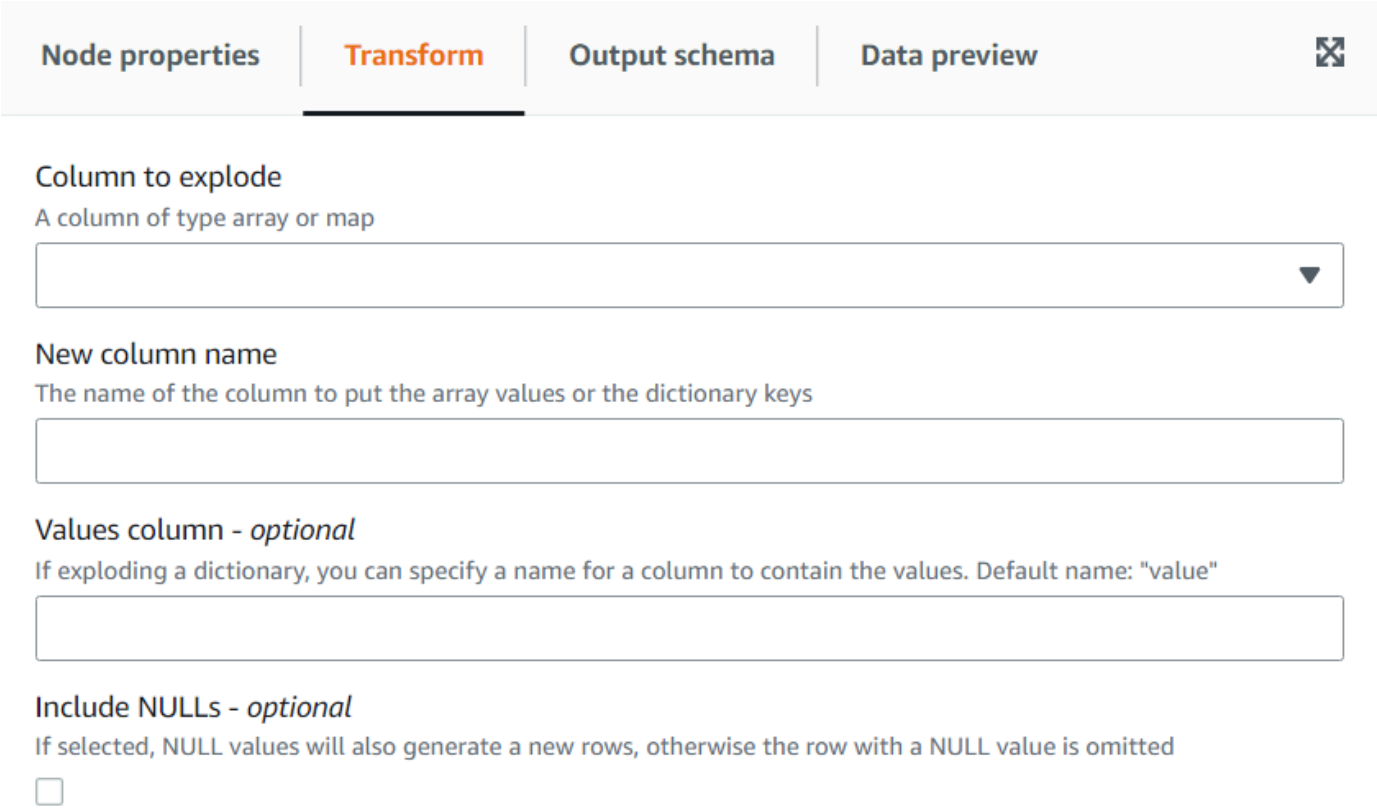
Se espandi la colonna "category" in una colonna con lo stesso nome, sovrascriverai la colonna. Puoi selezionare che desideri includere i NULL per ottenere quanto segue (ordinato a scopo illustrativo):

product_id	category
1	sport
1	inverno
2	giardino
2	strumento
3	videogiochi
4	game
4	gioco da tavolo
4	gioco di società
5	

Per aggiungere una trasformazione Espandi array o mappa in righe:

1. Apri il pannello Risorse, quindi scegli Espandi array o mappa in righe per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. (Facoltativo) Nella scheda Proprietà del nodo, puoi inserire un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.

3. Nella scheda Trasforma, scegli la colonna da espandere (deve essere di tipo array o mappa). Quindi, specifica un nome per la colonna relativa agli elementi dell'array oppure i nomi delle colonne per chiavi e valori nel caso di espansione di una mappa.
4. (Facoltativo) Nella scheda Trasforma, per impostazione predefinita, se la colonna da espandere è NULL o ha una struttura vuota, verrà omessa nel set di dati espanso. Se vuoi mantenere la riga (con le nuove colonne come NULL), seleziona "Includi NULL".

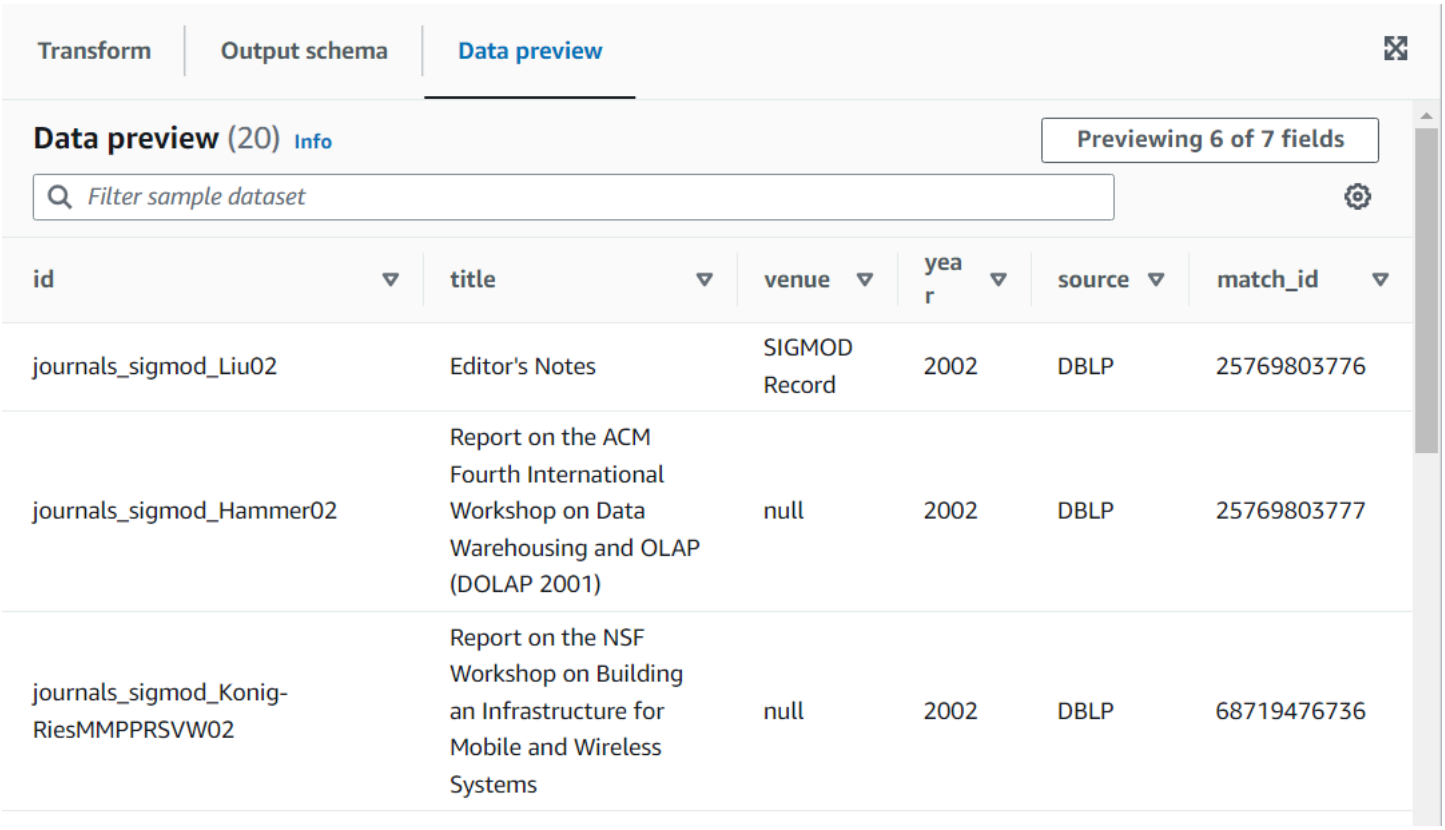


Utilizzo della trasformazione Corrispondenza dei record per richiamare una trasformazione di classificazione dei dati esistente

Questa trasformazione richiama una trasformazione di classificazione dei dati di machine learning Corrispondenza dei record esistente.

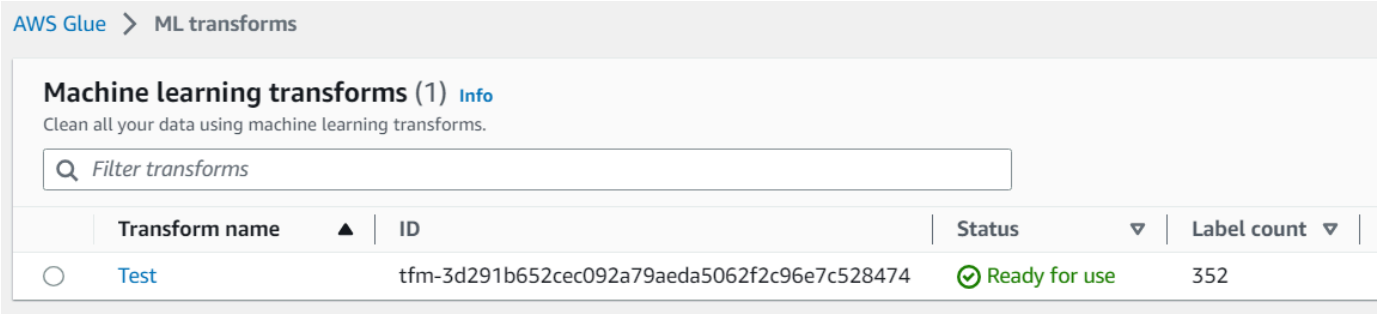
La trasformazione valuta i dati correnti rispetto al modello addestrato sulla base di etichette. Viene aggiunta una colonna "match_id" per assegnare ogni riga a un gruppo di elementi considerati equivalenti in base all'addestramento dell'algoritmo. Per ulteriori informazioni, consulta la pagina Record matching with Lake Formation FindMatches.

> **Note**
>
> La versione di AWS Glue utilizzata dal processo visivo deve corrispondere alla versione che AWS Glue ha utilizzato per creare la trasformazione Corrispondenza dei record.



Aggiunta di un nodo di trasformazione Corrispondenza dei record al diagramma di processo

1. Apri il pannello Risorse, quindi scegli Corrispondenza dei record per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. Nel pannello Proprietà del nodo, è possibile assegnare al nodo un nome nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, inserisci l'ID ottenuto dalla pagina delle Trasformazioni di Machine learning:



4. (Facoltativo) Nella scheda Trasforma, puoi selezionare l'opzione per aggiungere i punteggi di affidabilità. Al costo di un calcolo aggiuntivo, il modello stimerà un punteggio di affidabilità per ogni corrispondenza sotto forma di colonna aggiuntiva.

Rimozione di righe nulle

Questa trasformazione rimuove dal set di dati le righe che hanno tutte le colonne come nulle. Inoltre, è possibile estendere questi criteri per includere anche i campi vuoti, in modo da mantenere le righe in cui almeno una colonna non è vuota.

Aggiunta di un nodo di trasformazione Rimuovi righe nulle al diagramma di processo

1. Apri il pannello Risorse, quindi scegli Rimuovi righe nulle per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. Nel pannello Proprietà del nodo, è possibile assegnare al nodo un nome nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. (Facoltativo) Nella scheda Trasforma, seleziona l'opzione Estesa se desideri che le righe siano, oltre che non nulle, anche non vuote; in questo modo le stringhe, gli array o le mappe vuote verranno considerati nulli ai fini di questa trasformazione.

Analisi di una colonna di stringhe contenente dati JSON

Questa trasformazione analizza una colonna di stringhe contenente dati JSON e la converte in una struttura o in una colonna di array, a seconda che il JSON sia rispettivamente un oggetto o un array. Facoltativamente, puoi mantenere sia la colonna analizzata sia quella originale.

Lo schema JSON può essere fornito o dedotto (nel caso di oggetti JSON), con campionamento opzionale.

Aggiunta di un nodo di trasformazione Analizza colonna JSON al diagramma di processo

1. Apri il pannello Risorse, quindi scegli Analizza colonna JSON per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. Nel pannello Proprietà del nodo, è possibile assegnare al nodo un nome nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, seleziona la colonna contenente la stringa JSON.
4. (Facoltativo) Nella scheda Trasforma, inserisci lo schema seguito dai dati JSON utilizzando la sintassi SQL, ad esempio "field1 STRING, field2 INT" nel caso di un oggetto oppure "ARRAY<STRING>" nel caso di un array.

   Nel caso di un array, lo schema è richiesto, ma nel caso di un oggetto, se lo schema non è specificato, verrà dedotto utilizzando i dati. Per ridurre l'impatto dell'inferenza dello schema, specialmente su un set di dati di grandi dimensioni, puoi evitare di leggere l'intero dato due volte inserendo un Rapporto di campioni da utilizzare per dedurre lo schema. Se il valore è inferiore a 1, viene utilizzato il rapporto corrispondente di campioni casuali per dedurre lo schema. Se i dati sono affidabili e l'oggetto è coerente tra le righe, è possibile utilizzare un rapporto ridotto, ad esempio 0,1, per migliorare le prestazioni.
5. (Facoltativo) Nella scheda Trasforma, puoi inserire un nuovo nome di colonna se desideri mantenere sia la colonna di stringa originale sia la colonna analizzata.

Estrazione di un percorso JSON

Questa trasformazione estrae nuove colonne da una colonna di stringhe JSON. Questa trasformazione è utile quando sono necessari solo pochi elementi di dati e non si desidera importare l'intero contenuto JSON nello schema della tabella.

Aggiunta di un nodo di trasformazione Estrai percorso JSON nel diagramma di processo

1. Apri il pannello Risorse, quindi scegli Estrai percorso JSON per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. Nel pannello Proprietà del nodo, è possibile assegnare al nodo un nome nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.

3. Nella scheda Trasforma, seleziona la colonna contenente la stringa JSON. Inserisci una o più espressioni di percorso JSON separate da virgole, ognuna delle quali fa riferimento a come estrarre un valore dall'array o dall'oggetto JSON. Ad esempio, se la colonna JSON contenesse oggetti con le proprietà "prop_1" e "prop2", puoi estrarli entrambi specificando i nomi "prop_1, prop_2".

   Se il campo JSON contiene caratteri speciali, ad esempio per estrarre da questo JSON la proprietà `{"a. a": 1}`, puoi utilizzare il percorso `$['a. a']`. L'eccezione è la virgola perché è riservata a percorsi separati. Inserisci quindi i nomi di colonna corrispondenti per ogni percorso, separati da virgole.

4. (Facoltativo) Nella scheda Trasforma, puoi selezionare di eliminare la colonna JSON una volta estratta. Ciò ha senso quando non hai bisogno del resto dei dati JSON dopo aver estratto le parti necessarie.

Estrazione di frammenti di stringa utilizzando un'espressione regolare

Questa trasformazione estrae frammenti di stringa utilizzando un'espressione regolare e crea a partire da essa una nuova colonna o anche più colonne, se si utilizzano gruppi di regex.

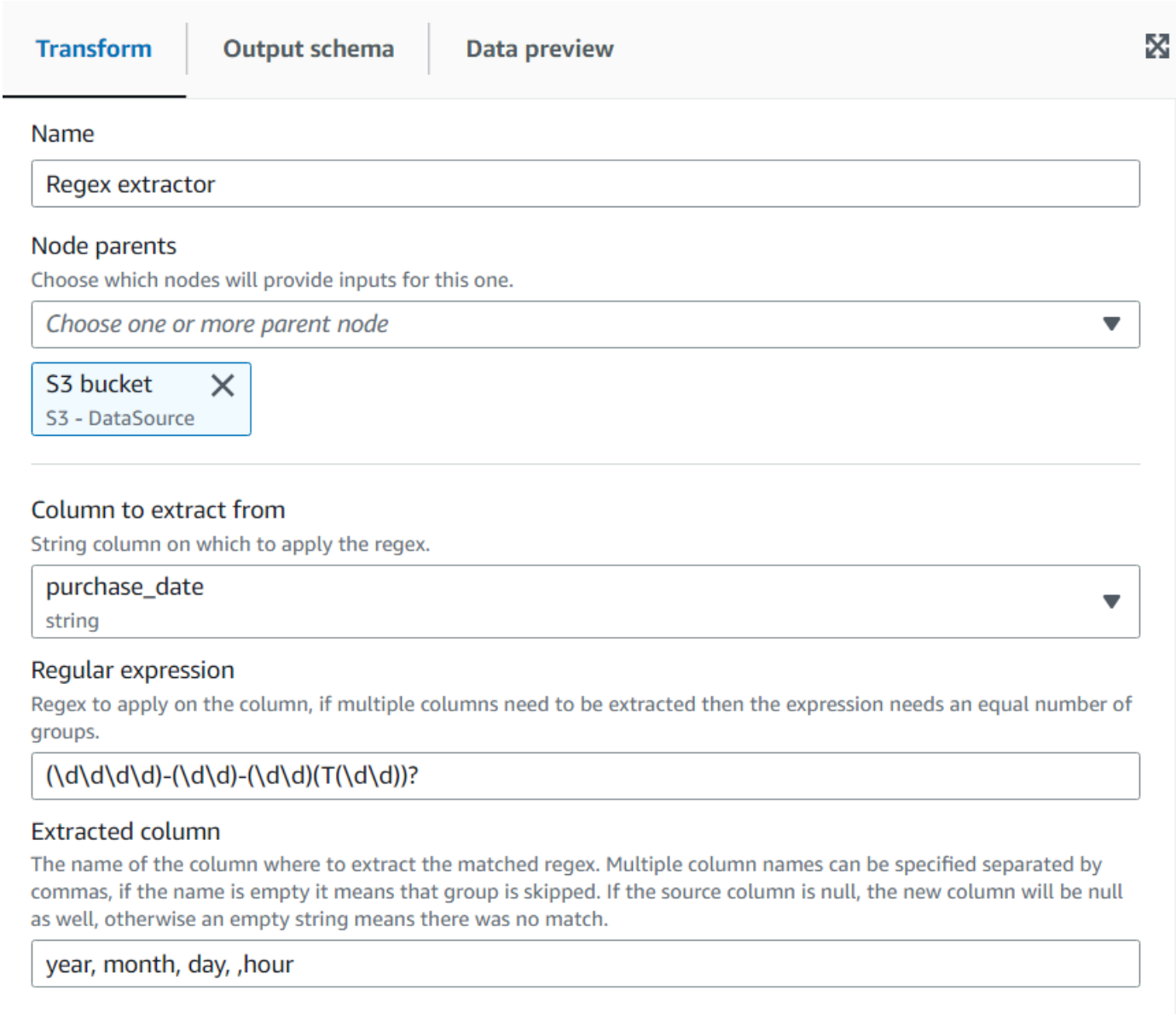
Aggiunta di un nodo di trasformazione Estrattore regex al diagramma di processo

1. Apri il pannello Risorse, quindi scegli Estrattore regex per aggiungere una nuova trasformazione al diagramma del processo. Il nodo selezionato al momento dell'aggiunta del nodo ne sarà il nodo padre.
2. Nel pannello Proprietà del nodo, è possibile assegnare al nodo un nome nel diagramma del processo. Se non è già selezionato un nodo padre, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.
3. Nella scheda Trasforma, inserisci l'espressione regolare e la colonna alla quale deve essere applicata. Quindi inserisci il nome della nuova colonna in cui archiviare la stringa corrispondente. La nuova colonna sarà nulla solo se la colonna di origine è nulla, mentre se l'espressione regolare non corrisponde la colonna sarà vuota.

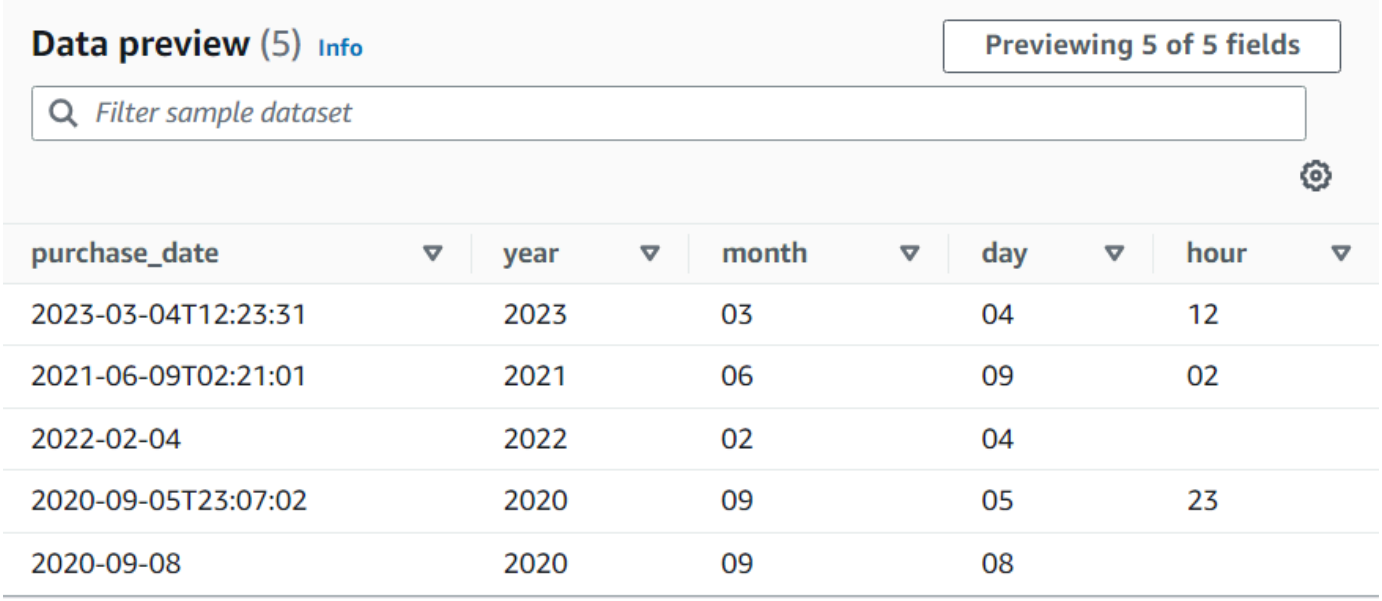
   Se l'espressione regolare utilizza gruppi, esiste un nome di colonna corrispondente separato da una virgola, ma è possibile saltare i gruppi lasciando vuoto il nome della colonna.

   Ad esempio, poniamo che tu abbia una colonna "purchase_date" con una stringa che utilizza formati di data ISO lunghi e brevi e voglia estrarre l'anno, il mese, il giorno e l'ora, se disponibili.

Nota che il gruppo delle ore è facoltativo, altrimenti, nelle righe in cui non è disponibile, tutti i gruppi estratti sarebbero stringhe vuote perché l'espressione regolare non corrisponde. In questo caso, non vogliamo che il gruppo renda facoltativo l'orario ma quello interno, quindi lasciamo il nome vuoto ed esso non verrà estratto (il gruppo includerebbe il carattere T).



Risultato dell'anteprima dei dati:



purchase_date	year	month	day	hour
2023-03-04T12:23:31	2023	03	04	12
2021-06-09T02:21:01	2021	06	09	02
2022-02-04	2022	02	04	
2020-09-05T23:07:02	2020	09	05	23
2020-09-08	2020	09	08	

Creazione di una trasformazione personalizzata

Se devi eseguire trasformazioni più complicate sui dati o se vuoi aggiungere chiavi di proprietà dei dati al set di dati, puoi aggiungere una trasformazione Custom code al diagramma di processo. Il nodo Custom code permette di immettere uno script che esegue la trasformazione.

Quando si utilizza il codice personalizzato, è necessario utilizzare un editor di schemi per indicare le modifiche apportate all'output tramite il codice personalizzato. Quando modifichi lo schema, puoi eseguire le seguenti operazioni:

- Aggiungere o rimuovere chiavi di proprietà dei dati
- Modificare il tipo di dati delle chiavi di proprietà dei dati
- Modificare il nome delle chiavi di proprietà dei dati.
- Ristrutturare una chiave di proprietà nidificata

Devi utilizzare una trasformazione SelectFromCollection per scegliere un singolo `DynamicFrame` dal risultato del nodo di trasformazione Custom prima di poter inviare l'output a una posizione di destinazione.

Usa i processi seguenti per aggiungere un nodo di trasformazione personalizzato al diagramma di processo.

Aggiunta di un nodo di trasformazione di codice personalizzato al diagramma di processo

Per aggiungere un nodo di trasformazione personalizzato al diagramma di processo

1. (Facoltativo) Apri il pannello Risorse, quindi scegli Trasformazione personalizzata per aggiungere una nuova trasformazione al diagramma del processo.
2. Nella scheda Node properties (Proprietà del nodo), inserisci un nome per il nodo nel diagramma del processo. Se non è già selezionato un nodo padre, o se desideri più input per la trasformazione personalizzata, scegli un nodo dall'elenco Node parents (Nodi padre) da utilizzare come origine di input per la trasformazione.

Immissione del codice per il nodo di trasformazione personalizzato

Puoi digitare o copiare il codice in un campo di input. Il processo utilizza questo codice per eseguire la trasformazione dei dati. Puoi fornire un frammento di codice in Python o Scala. Il codice richiede uno o più `DynamicFrames` come input e restituisce una raccolta di `DynamicFrames`.

Per inserire lo script per un nodo di trasformazione personalizzato

1. Con il nodo di trasformazione personalizzato selezionato nel diagramma di processo, scegli la casella Transform (Trasformazione).
2. Nel campo di immissione testo sotto l'intestazione Code block (Blocco di codice), incolla o immetti il codice per la trasformazione. Il codice utilizzato deve corrispondere al linguaggio specificato per il processo nella scheda Job details (Dettagli del processo).

   Quando si fa riferimento ai nodi di input nel codice, AWS Glue Studio assegna un nome ai `DynamicFrames` restituiti dai nodi del diagramma del processo in modo sequenziale in base all'ordine di creazione. Utilizza uno dei seguenti metodi di denominazione nel codice:

   - Generazione di codice classico: utilizza i nomi funzionali per fare riferimento ai nodi nel diagramma del processo.
     - Nodi di origine dati: `DataSource0`, `DataSource1`, `DataSource2` e così via.
     - Nodi di trasformazione : `Transform0`, `Transform1`, `Transform2` e così via.
   - Nuova generazione di codice: utilizza il nome specificato nella scheda Node properties (Proprietà del nodo) di un nodo, aggiunta con "`_node1`","`_node2`" e così via.
     Ad esempio, `S3bucket_node1`, `ApplyMapping_node2`, `S3bucket_node2`, `MyCustomNodeName_node1`.

Per ulteriori informazioni sul nuovo generatore di codice, consulta Generazione di codice dello script.

Gli esempi seguenti mostrano il formato del codice da inserire nella casella del codice:

Python

L'esempio seguente prende il primo `DynamicFrame` ricevuto, lo converte in un `DataFrame` per applicare il metodo di filtro nativo (conservando solo i registri che hanno più di 1000 voti), quindi prima di restituirlo lo converte di nuovo in un `DynamicFrame`.

```
def FilterHighVoteCounts (glueContext, dfc) -> DynamicFrameCollection:
    df = dfc.select(list(dfc.keys())[0]).toDF()
    df_filtered = df.filter(df["vote_count"] > 1000)
    dyf_filtered = DynamicFrame.fromDF(df_filtered, glueContext, "filter_votes")
    return(DynamicFrameCollection({"CustomTransform0": dyf_filtered}, glueContext))
```

Scala

L'esempio seguente prende il primo `DynamicFrame` ricevuto, lo converte in un `DataFrame` per applicare il metodo di filtro nativo (conservando solo i registri che hanno più di 1000 voti), quindi prima di restituirlo lo converte di nuovo in un `DynamicFrame`.

```
object FilterHighVoteCounts {
  def execute(glueContext : GlueContext, input : Seq[DynamicFrame]) :
 Seq[DynamicFrame] = {
    val frame = input(0).toDF()
    val filtered = DynamicFrame(frame.filter(frame("vote_count") > 1000),
 glueContext)
    Seq(filtered)
  }
}
```

Modifica dello schema in un nodo di trasformazione personalizzato

Quando si utilizza un nodo di trasformazione personalizzato, AWS Glue Studio non può dedurre automaticamente gli schemi di output creati dalla trasformazione. Devi utilizzare l'editor dello schema per descrivere le modifiche allo schema implementate dal codice di trasformazione personalizzato.

Un nodo di codice personalizzato può avere un numero qualsiasi di nodi padre, ognuno dei quali fornisce un `DynamicFrame` come input per il codice personalizzato. Un nodo di codice personalizzato restituisce una raccolta di `DynamicFrames`. Ogni `DynamicFrame` utilizzato come input ha uno schema associato. È necessario aggiungere uno schema che descriva ogni `DynamicFrame` restituito dal nodo di codice personalizzato.

> **Note**
>
> Quando imposti il tuo schema su una trasformazione personalizzata, AWS Glue Studio non eredita schemi dai nodi precedenti. Per aggiornare lo schema, seleziona il nodo di trasformazione personalizzata, quindi sceglie la scheda Data preview (anteprima dati. Una volta generata l'anteprima, scegli “Use Preview Schema” (usa schema di anteprima). Lo schema verrà quindi sostituito dallo schema utilizzando i dati di anteprima.

Per modificare gli schemi per un nodo di trasformazione personalizzato

1. Con il nodo di trasformazione personalizzato selezionato nel diagramma di processo, scegli la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo.
2. Scegli Edit (Modifica) per apportare modifiche allo schema.

   Se disponi di chiavi di proprietà dei dati nidificate, ad esempio una matrice o un oggetto, puoi scegliere l'icona Expand-Rows (Espandi righe) ( ) in alto a destra di ogni pannello dello schema per espandere l'elenco delle chiavi di proprietà dei dati figlio. Dopo aver selezionato l'icona, questa si trasforma nell'icona Collapse-Rows (Comprimi righe) ( ), che puoi selezionare per comprimere l'elenco delle chiavi di proprietà figlio.
3. Modifica lo schema utilizzando le seguenti operazioni nella sezione a destra della pagina:

   - Per rinominare una chiave di proprietà, posiziona il cursore nella casella di testo Key (Chiave) per la chiave di proprietà, quindi immetti il nuovo nome.
   - Per modificare il tipo di dati per una chiave di proprietà dei dati, usa l'elenco per scegliere il nuovo tipo di dati per la chiave di proprietà.
   - Per aggiungere una nuova chiave di proprietà di livello superiore allo schema, scegli l'opzione Overflow

(⋯)
sulla sinistra del pulsante Cancel (Annulla), quindi scegli Add root key (Aggiungi chiave root).

- Per aggiungere una chiave di proprietà figlio allo schema, scegli l'icona Add-Key (Aggiungi chiave)
  associata alla chiave padre. Inserisci un nome per la chiave figlio e scegli il tipo di dati.
- Per rimuovere una colonna dallo schema, scegli l'icona Remove (Elimina)
  ( )
  all'estrema destra del nome della chiave.

4. Se il codice di trasformazione personalizzato utilizza più `DynamicFrames`, è possibile aggiungere schemi di output aggiuntivi.

   - Per aggiungere un nuovo schema vuoto, scegli l'icona Overflow
     (⋯),
     quindi scegli Add output schema (Aggiungi schema di output).
   - Per copiare uno schema esistente in un nuovo schema di output, assicurati che lo schema da copiare sia visualizzato nel selettore dello schema. Seleziona l'icona Overflow
     (⋯),
     quindi scegli Duplicate (Duplica).

   Se vuoi rimuovere uno schema di output, assicurati che lo schema da copiare sia visualizzato nel selettore dello schema. Seleziona l'icona Overflow
   (⋯),
   quindi scegli Delete (Elimina).
5. Aggiungi nuove chiavi radice al nuovo schema o modifica le chiavi duplicate.
6. Quando modifichi gli lo schemi di output, scegli il pulsante Apply (Applica) per salvare le modifiche e uscire dall'editor dello schema.

   Se non vuoi salvare le modifiche, seleziona il pulsante Cancel (Annulla).

Configurare l'output della trasformazione personalizzata

Una trasformazione di codice personalizzata restituisce una raccolta di `DynamicFrames`, anche se nel set di risultati è presente solo un `DynamicFrame`.

Per elaborare l'output da un nodo di trasformazione personalizzato

1. Aggiungi una trasformazione SelectFromCollection, che ha il nodo di trasformazione personalizzato come nodo padre. Aggiorna questa trasformazione per indicare il set di dati da utilizzare. Per ulteriori informazioni, consulta Utilizzo di SelectFromCollection per scegliere quale set di dati conservare.
2. Aggiungi altre trasformazioni SelectFromCollection al diagramma di processo se vuoi utilizzare `DynamicFrames` aggiuntivi prodotti dal nodo di trasformazione personalizzato.

   Consideriamo uno scenario in cui aggiungi un nodo di trasformazione personalizzato per dividere un set di dati di volo in più set di dati, ma duplichi alcune delle chiavi di proprietà identificative in ciascuno schema di output, ad esempio la data di volo o il numero di volo. Aggiungi un nodo di trasformazione SelectFromCollection per ogni schema di output, con il nodo di trasformazione personalizzato come nodo padre.
3. (Facoltativo) Puoi quindi utilizzare ogni nodo di trasformazione SelectFromCollection come input per altri nodi nel processo o come padre per un nodo di destinazione dati.

Trasformazioni visive personalizzate di AWS Glue

Le trasformazioni visive personalizzate consentono di creare trasformazioni e renderle disponibili per l'uso nei processi AWS Glue Studio. Le trasformazioni visive personalizzate consentono agli sviluppatori ETL, che potrebbero non avere familiarità con la codifica, di cercare e utilizzare una libreria di trasformazioni in via di sviluppo tramite l'interfaccia AWS Glue Studio.

Puoi creare una trasformazione visiva personalizzata e caricarla su Amazon S3 per renderla disponibile all'uso tramite l'editor visivo in AWS Glue Studio e lavorare con questi processi.

Argomenti

- Esempi di script visivi personalizzati
- Video

Nozioni di base sulle trasformazioni visive personalizzate

Per creare una trasformazione visiva personalizzata, completa la seguente procedura.

- Fase 1. Creazione di un file di configurazione JSON
- Fase 2. Implementazione della logica di trasformazione
- Fase 3. Convalida della trasformazione visiva personalizzata
- Fase 4. Aggiornamento della trasformazione visiva personalizzata in base alle necessità
- Fase 5: Utilizzo della trasformazione visiva personalizzata in AWS Glue Studio

Inizia configurando il bucket Amazon S3 e continua con la Fase 1. Crea un file di configurazione JSON.

Prerequisiti

Le trasformazioni fornite dal cliente risiedono all'interno di un account AWS del cliente. Quell'account possiede le trasformazioni e quindi dispone di tutte le autorizzazioni per visualizzarle (ricerca e uso), modificarle o eliminarle.

Per utilizzare una trasformazione personalizzata in AWS Glue Studio, dovrai creare e caricare due file nel bucket delle risorse di Amazon S3 in quell'account AWS:

- File Python: contiene la funzione di trasformazione.
- File JSON: descrive la trasformazione. Questo è noto anche come file di configurazione necessario per definire la trasformazione.

Per accoppiare i file, utilizzate lo stesso nome per entrambi. Ad esempio:

- myTransform.json
- myTransform.py

Facoltativamente, puoi assegnare alla tua trasformazione visiva personalizzata un'icona personalizzata fornendo un file SVG contenente l'icona. Per accoppiare i file, utilizza lo stesso nome per l'icona:

- myTransform.svg

AWS Glue Studio li abbinerà automaticamente utilizzando i rispettivi nomi dei file. I nomi dei file non possono essere uguali per nessun modulo esistente.

Convenzione consigliata per il nome del file della trasformazione

AWS Glue Studio importerà il file come modulo (ad esempio, `import myTransform`) nello script del processo. Pertanto, il nome del file deve seguire le stesse regole di denominazione impostate per i nomi delle variabili python (identificatori). In particolare, devono iniziare con una lettera o un carattere di sottolineatura e quindi essere composti interamente da lettere, numeri e/o trattini bassi.

> **Note**
>
> Assicurati che il nome del file di trasformazione non sia in conflitto con i moduli python caricati esistenti (ad esempio, `sys, array, copy` ecc.) per evitare problemi di runtime imprevisti.

Configurazione del bucket Amazon S3

Le trasformazioni create vengono archiviate in Amazon S3 e sono di proprietà del tuo account AWS. È possibile creare nuove trasformazioni visive personalizzate semplicemente caricando i file (.json e .py) nella cartella delle risorse di Amazon S3 dove sono correntemente archiviati tutti gli script del processo (ad esempio, `s3://aws-glue-assets-<accountid>-<region>/transforms`). Se utilizzi un'icona personalizzata, carica anch'essa. Per impostazione predefinita, AWS Glue Studio leggerà tutti i file .json dalla cartella /transforms nello stesso bucket S3.

Fase 1. Creazione di un file di configurazione JSON

Un file di configurazione JSON è necessario per definire e descrivere la trasformazione visiva personalizzata. Lo schema per il file di configurazione è il seguente.

Struttura dei file JSON

Campi

- `name: string`: (obbligatorio) il nome del sistema di trasformazione utilizzato per identificare le trasformazioni. Segui le stesse regole di denominazione impostate per i nomi delle variabili python (identificatori). In particolare, devono iniziare con una lettera o un carattere di sottolineatura e quindi essere composti interamente da lettere, numeri e/o trattini bassi.
- `displayName: string`: (facoltativo) il nome della trasformazione visualizzata nell'editor di processi visivi di AWS Glue Studio. Se non viene specificato alcun `displayName`, come nome della trasformazione in AWS Glue Studio viene utilizzato `name`.
- `description: string`: (facoltativo) la descrizione della trasformazione viene visualizzata in AWS Glue Studio ed è ricercabile.
- `functionName: string`: (obbligatorio) il nome della funzione Python viene utilizzato per identificare la funzione da chiamare nello script Python.
- `path: string`: (facoltativo) il percorso completo di Amazon S3 del file sorgente di Python. Se non viene specificato, AWS Glue utilizza il nome del file corrispondente per accoppiare i file .json e .py. Ad esempio, il nome del file JSON, `myTransform.json`, verrà associato al file Python, `myTransform.py`, nella stessa posizione di Amazon S3.
- `parameters: Array of TransformParameter object`: (facoltativo) l'elenco dei parametri da visualizzare quando li si configura nell'editor visivo di AWS Glue Studio.

Campi di TransformParameter

- `name: string`: (obbligatorio) il nome del parametro che verrà passato alla funzione python come argomento denominato nello script del processo. Segui le stesse regole di denominazione impostate per i nomi delle variabili python (identificatori). In particolare, devono iniziare con una lettera o un carattere di sottolineatura e quindi essere composti interamente da lettere, numeri e/o trattini bassi.
- `displayName: string`: (facoltativo) il nome della trasformazione visualizzata nell'editor di processi visivi di AWS Glue Studio. Se non viene specificato alcun `displayName`, come nome della trasformazione in AWS Glue Studio viene utilizzato `name`.
- `type: string`: (obbligatorio) il tipo di parametro che accetta i tipi di dati Python comuni. I valori validi sono 'str' | 'int' | 'float' | 'list' | 'bool'.
- `isOptional: boolean`: (facoltativo) determina se il parametro è facoltativo. Per impostazione predefinita, tutti i parametri sono obbligatori.
- `description: string`: (facoltativo) viene visualizzata una descrizione in AWS Glue Studio per aiutare l'utente a configurare il parametro di trasformazione.

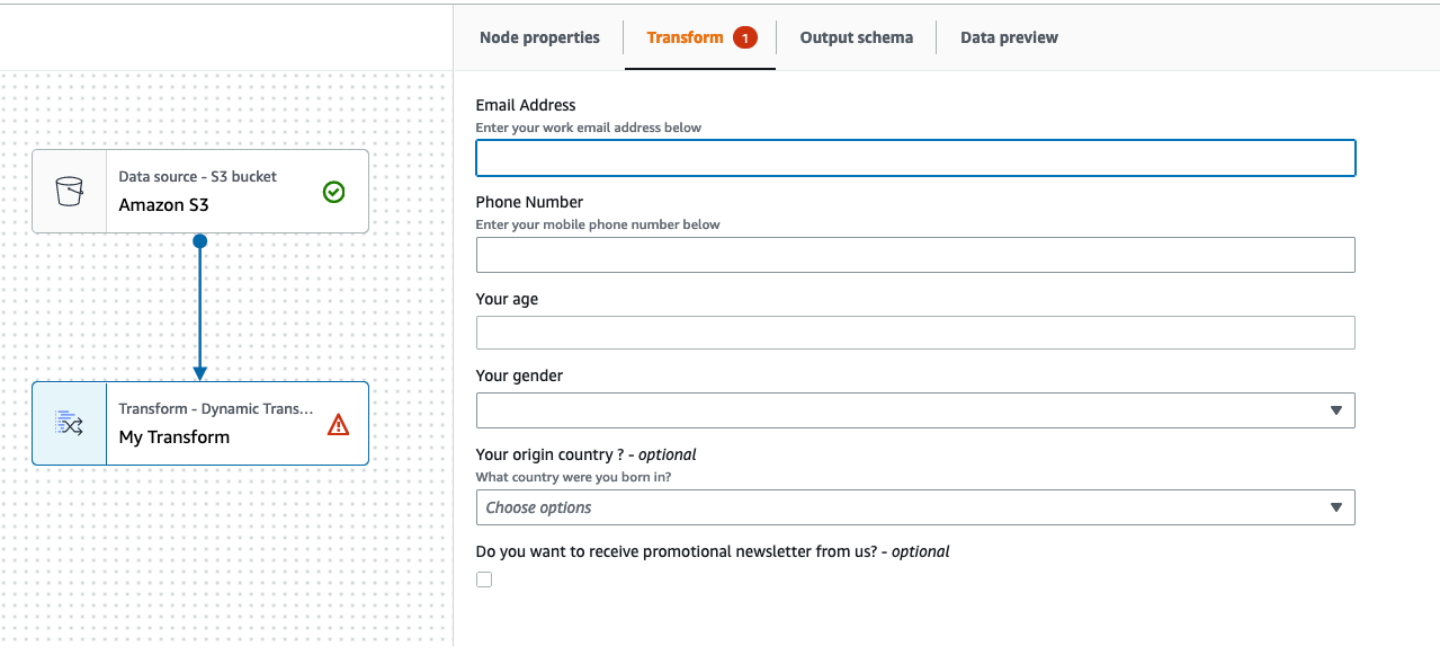
- `validationType: string`: (facoltativo) definisce il modo in cui questo parametro viene convalidato. Al momento, supporta solo le espressioni regolari. Per impostazione predefinita, il tipo di convalida è impostato su `RegularExpression`.
- `validationRule: string`: (facoltativo) l'espressione regolare utilizzata per convalidare l'input del modulo prima dell'invio quando `validationType` è impostato su `RegularExpression`. La sintassi delle espressioni regolari deve essere compatibile con le specifiche RegExp Ecmascript.
- `validationMessage: string`: (facoltativo) il messaggio da visualizzare quando la convalida non riesce.
- `listOptions: An array of TransformParameterListOption object` OPPURE una `string` o il valore di stringa "column": (facoltativo) le opzioni da visualizzare nel controllo Select o Multiselect dell'interfaccia utente. Accettazione di un elenco di valori separati da virgole o di un oggetto JSON fortemente tipizzato di tipo `TransformParameterListOption`. Può anche compilare dinamicamente l'elenco di colonne dello schema del nodo padre specificando il valore di stringa "column".
- `listType: string`: (facoltativo) definisci i tipi di opzioni per type = 'list'. I valori validi sono 'str' | 'int' | 'float' | 'list' | 'bool'. Tipo di parametro che accetta i tipi di dati Python comuni.

Campi TransformParameterListOption

- `value: string | int | float | bool`: (obbligatorio) il valore dell'opzione.
- `label: string` (facoltativo) l'etichetta dell'opzione visualizzata nel menu a discesa di selezione.

Trasformazione dei parametri in AWS Glue Studio

Per impostazione predefinita, i parametri sono obbligatori a meno che non siano contrassegnati come `isOptional` nel file .json. In AWS Glue Studio, i parametri vengono visualizzati nella scheda Transform (Trasforma). L'esempio mostra parametri definiti dall'utente come indirizzo e-mail, numero di telefono, età, sesso e paese di origine.

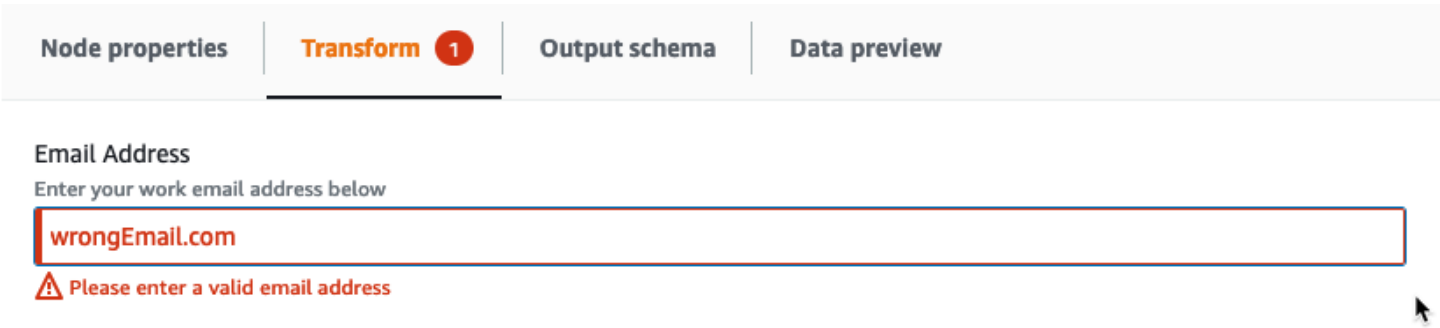


È possibile applicare alcune convalide in AWS Glue Studio utilizzando le espressioni regolari nel file .json specificando il parametro `validationRule` e specificando un messaggio di convalida in `validationMessage`.

```
    "validationRule": "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
    "validationMessage": "Please enter a valid US number"
```

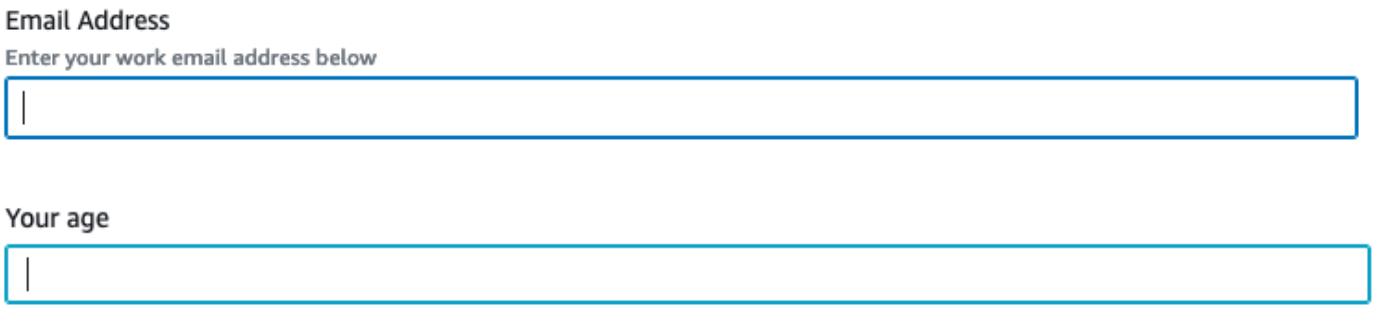
> **Note**
>
> Poiché la convalida avviene nel browser, la sintassi dell'espressione regolare deve essere compatibile con le specifiche RegExp Ecmascript. La sintassi di Python non è supportata per queste espressioni regolari.

L'aggiunta della convalida impedirà all'utente di salvare il processo con un input utente errato. AWS Glue Studio visualizza il messaggio di convalida come mostrato nell'esempio:

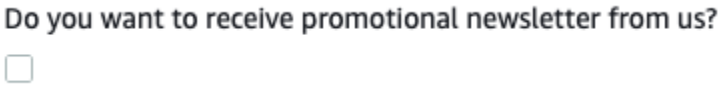


I parametri vengono visualizzati in AWS Glue Studio in base alla loro configurazione.

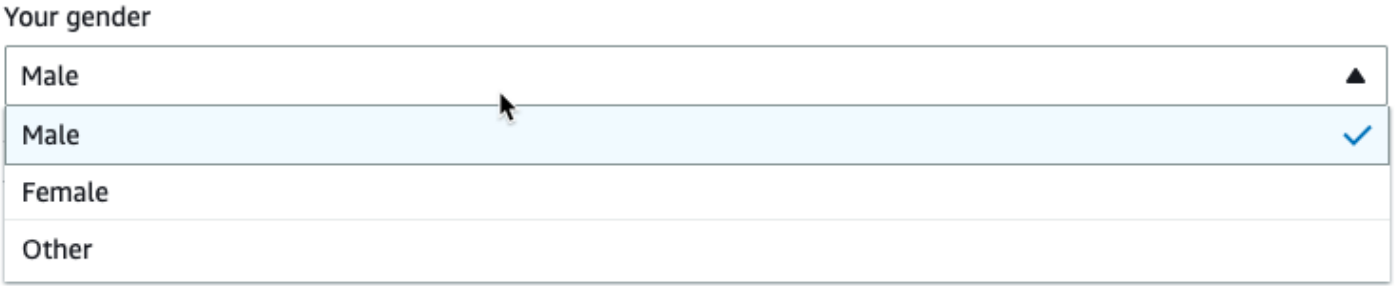
- Quando `type` è `str`, `int` o `float`, viene visualizzato un campo di immissione di testo. Ad esempio, la schermata mostra i campi di input per i parametri "Indirizzo e-mail" ed "Età".



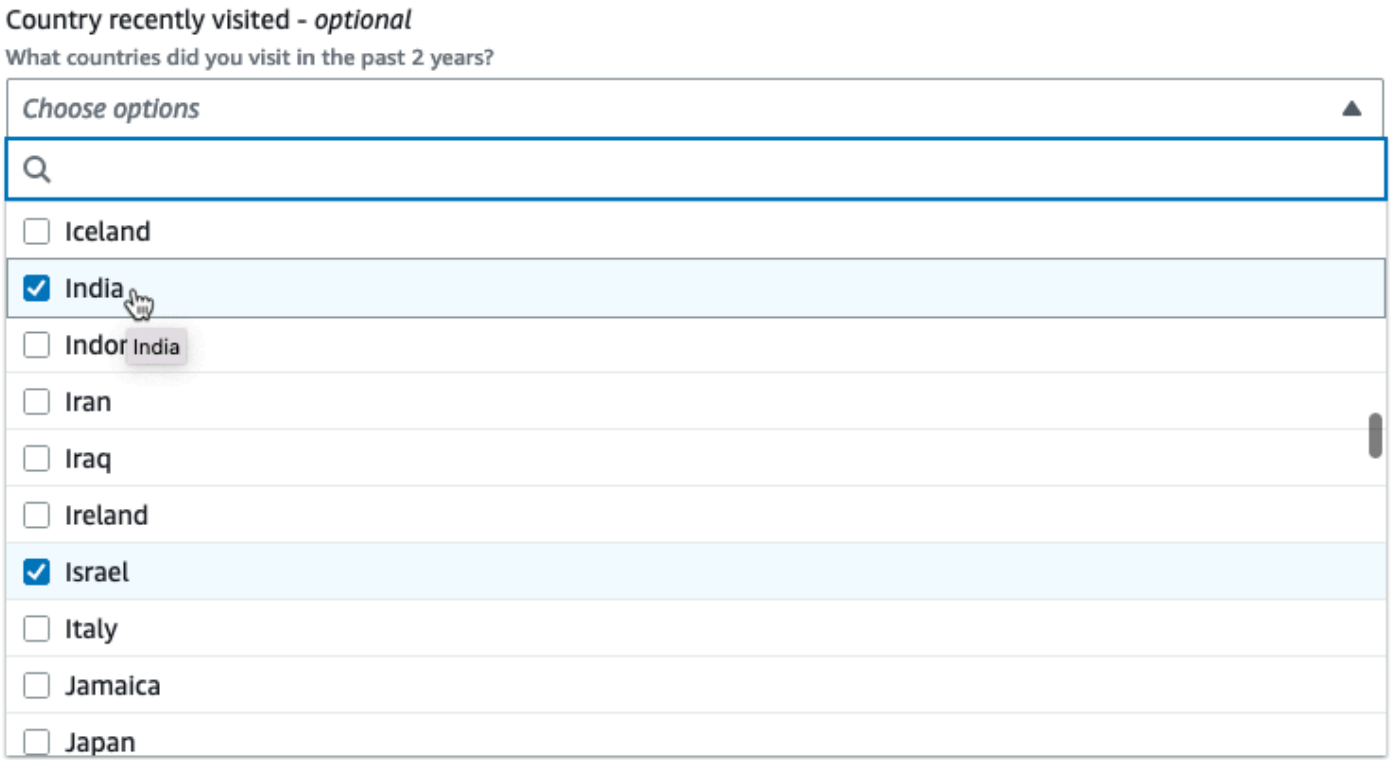
- Quando `type` è `bool`, viene visualizzata una casella di controllo.



- Quando `type` è `str` e viene fornito `listOptions`, viene visualizzato un unico elenco di selezione.



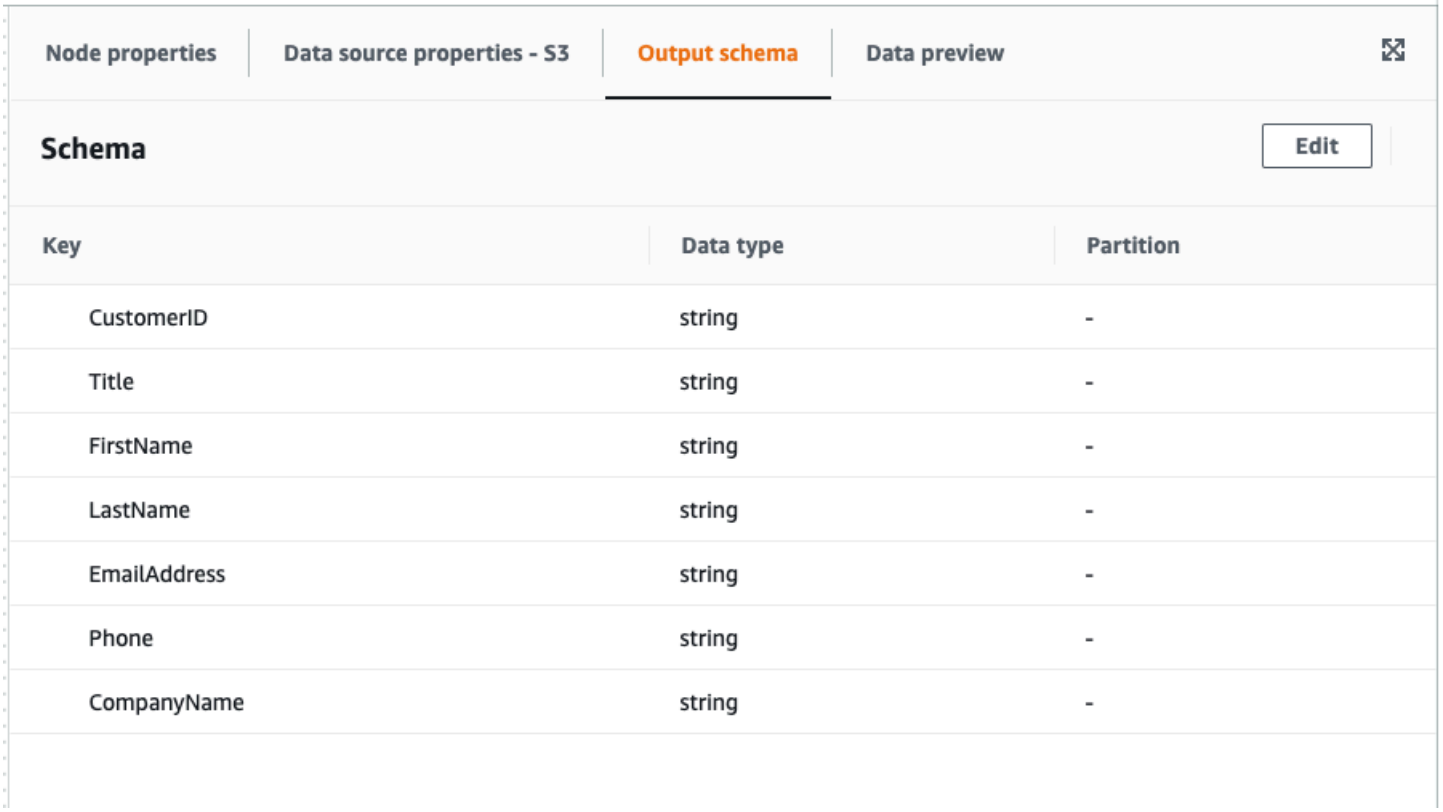
- Quando `type` è `list` e sono forniti `listOptions` e `listType`, viene visualizzato un elenco a selezione multipla.



Visualizzazione di un selettore di colonna come parametro

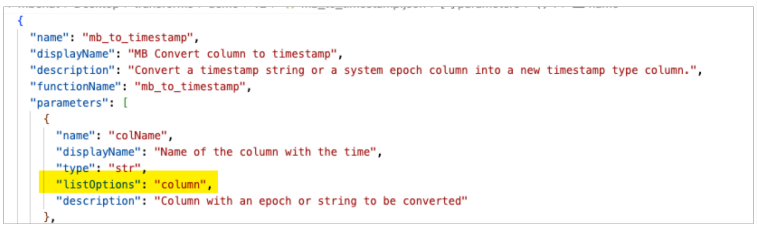
Se la configurazione richiede all'utente di scegliere una colonna dallo schema, è possibile visualizzare un selettore di colonna in modo che l'utente non debba digitarne il nome. Impostando il campo `listOptions` su "column", AWS Glue Studio visualizza dinamicamente un selettore di colonne basato sullo schema di output del nodo principale. AWS Glue Studio può visualizzare un selettore a colonna singola o multipla.

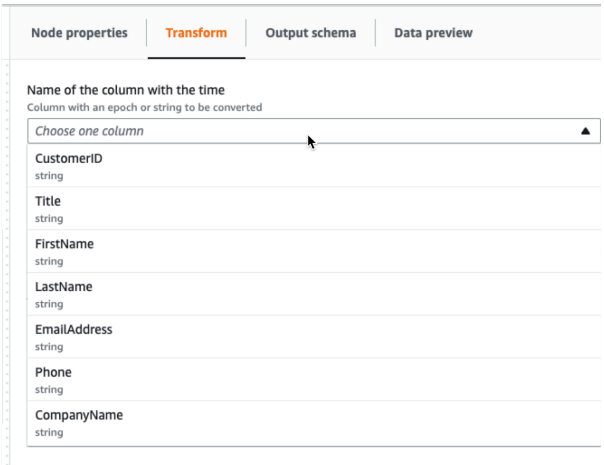
L'esempio seguente utilizza lo schema:



Per definire il parametro Trasformazione visiva personalizzata per visualizzare una singola colonna:

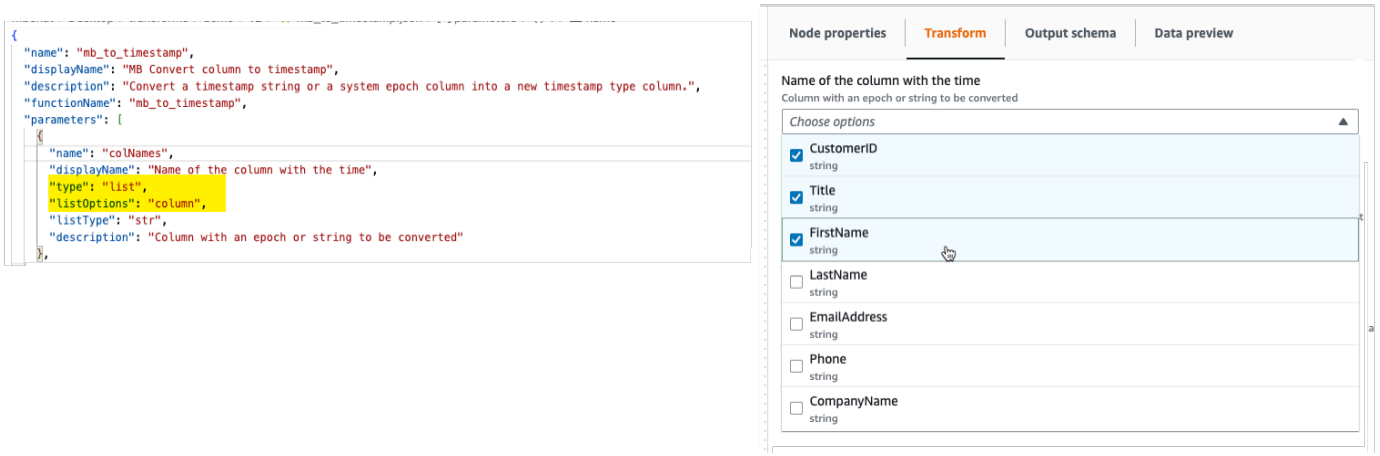
1. Nel file JSON, per l'oggetto `parameters`, imposta il valore di `listOptions` su "column". Ciò consente a un utente di scegliere una colonna da un elenco di selezione in AWS Glue Studio.





2. È inoltre possibile consentire la selezione di più colonne definendo il parametro come:

   - `listOptions: "column"`
   - `type: "list"`



Fase 2. Implementazione della logica di trasformazione

> **Note**
>
> Le trasformazioni visive personalizzate supportano solo gli script Python. Scala non è supportato.

Per aggiungere il codice che implementa la funzione definita dal file di configurazione .json, si consiglia di posizionare il file Python nella stessa posizione del file .json, con lo stesso nome ma con l'estensione ".py". AWS Glue Studio accoppia automaticamente i file .json e .py in modo che non sia necessario specificare il percorso del file Python nel file di configurazione.

Nel file Python, aggiungi la funzione dichiarata, con i parametri denominati configurati e registrala per l'uso in `DynamicFrame`. Di seguito è riportato un esempio di un file Python:

```
from awsglue import DynamicFrame

# self refers to the DynamicFrame to transform,
# the parameter names must match the ones defined in the config
# if it's optional, need to provide a default value
def myTransform(self, email, phone, age=None, gender="",
                      country="", promotion=False):
   resulting_dynf = # do some transformation on self
   return resulting_dynf

DynamicFrame.myTransform = myTransform
```

Si consiglia di utilizzare un notebook AWS Glue per sviluppare e testare il codice python in modo più rapido. consulta, Nozioni di base sui notebook in AWS Glue Studio.

Per illustrare come implementare la logica della trasformazione, la trasformazione visiva personalizzata nell'esempio seguente è una trasformazione per filtrare i dati in entrata e conservare solo i dati relativi a uno specifico stato degli Stati Uniti. Il file .json contiene il parametro per `functionName` come `custom_filter_state` e due argomenti ("state" e "colName" con tipo "str").

Il file di configurazione .json di esempio è:

```
{
"name": "custom_filter_state",
"displayName": "Filter State",
"description": "A simple example to filter the data to keep only the state indicated.",
"functionName": "custom_filter_state",
"parameters": [
   {
    "name": "colName",
    "displayName": "Column name",
    "type": "str",
    "description": "Name of the column in the data that holds the state postal code"
   },
   {
    "name": "state",
    "displayName": "State postal code",
    "type": "str",
    "description": "The postal code of the state whole rows to keep"
   }
  ]
}
```

Implementazione dello script complementare in Python

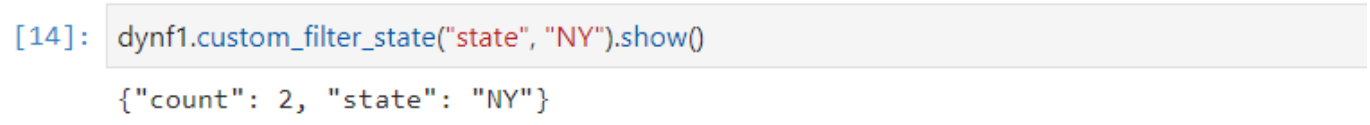
1. Avvia un notebook AWS Glue ed esegui la cella iniziale fornita per l'avvio della sessione. L'esecuzione della cella iniziale crea i componenti di base necessari.
2. Crea una funzione che esegua il filtraggio come descritto nell'esempio e registrala in `DynamicFrame`. Copia il codice seguente e incollalo in una cella del notebook AWS Glue.

```
from awsglue import DynamicFrame

def custom_filter_state(self, colName, state):
    return self.filter(lambda row: row[colName] == state)

DynamicFrame.custom_filter_state = custom_filter_state
```

3. Crea o carica dati di esempio per testare il codice nella stessa cella o in una nuova cella. Se aggiungi i dati di esempio in una nuova cella, non dimenticare di eseguire la cella. Per esempio:

```
# A few of rows of sample data to test
data_sample = [
    {"state": "CA", "count": 4},
    {"state": "NY", "count": 2},
    {"state": "WA", "count": 3}
]
df1 = glueContext.sparkSession.sparkContext.parallelize(data_sample).toDF()
dynf1 = DynamicFrame.fromDF(df1, glueContext, None)
```

4. Esegui test per convalidare "custom_filter_state" con diversi argomenti:



5. Dopo aver eseguito diversi test, salva il codice con l'estensione .py e assegna al file .py un nome che rispecchi il nome del file .json. I file .py e .json devono trovarsi nella stessa cartella di trasformazione.

   Copia il codice seguente e incollalo in un file, quindi rinominalo con l'estensione .py.

```
from awsglue import DynamicFrame

def custom_filter_state(self, colName, state):
    return self.filter(lambda row: row[colName] == state)

DynamicFrame.custom_filter_state = custom_filter_state
```

6. In AWS Glue Studio, apri un processo visivo e aggiungi una trasformazione selezionandola tra quelle disponibili nell'elenco Transforms (Trasformazioni).

   Per riutilizzare questa trasformazione in un codice script Python, aggiungi il percorso Amazon S3 al file .py nel processo nel "Percorso dei file di riferimento" e nello script, importa il nome del file python (senza estensione) aggiungendolo all'inizio del file. Ad esempio: `import` <nome del file (senza estensione)>

Fase 3. Convalidazione e risoluzione dei problemi delle trasformazioni visive personalizzate in AWS Glue Studio

AWS Glue Studio convalida il file di configurazione JSON prima che le trasformazioni visive personalizzate vengano caricate in AWS Glue Studio. La convalida include:

- Presenza di campi obbligatori
- Convalida del formato JSON
- Parametri non corretti o non validi
- Presenza dei file .py e .json nello stesso percorso Amazon S3
- Nomi di file corrispondenti per i file .py e .json

Se la convalida ha esito positivo, la trasformazione viene riportata tra quelle disponibili nell'elenco Actions (Operazioni) nell'editor visivo. Se è stata fornita un'icona personalizzata, dovrebbe essere visibile accanto a Operazione.

Se la convalida non riesce, AWS Glue Studio non carica la trasformazione visiva personalizzata.

Fase 4. Aggiornamento delle trasformazioni visive personalizzate in base alle necessità

Una volta creato e utilizzato, lo script di trasformazione può essere aggiornato purché la trasformazione segua la definizione json corrispondente:

- Il nome usato per l'assegnazione a DynamicFrame deve corrispondere al `functionName` json.
- Gli argomenti della funzione devono essere definiti nel file .json come descritto in Fase 1. Creazione di un file di configurazione JSON .
- Il percorso Amazon S3 del file Python non può cambiare, poiché i processi dipendono direttamente da esso.

> **Note**
>
> Se è necessario apportare aggiornamenti, assicurati che lo script e il file .json siano costantemente aggiornati e che tutti i processi visivi vengano nuovamente salvati correttamente con la nuova trasformazione. Se i processi visivi non vengono salvati dopo aver effettuato gli aggiornamenti, gli aggiornamenti non saranno applicati e convalidati. Se il file di script Python viene rinominato o non posizionato insieme al file .json, è necessario specificare il percorso completo nel file .json.

Icona personalizzata

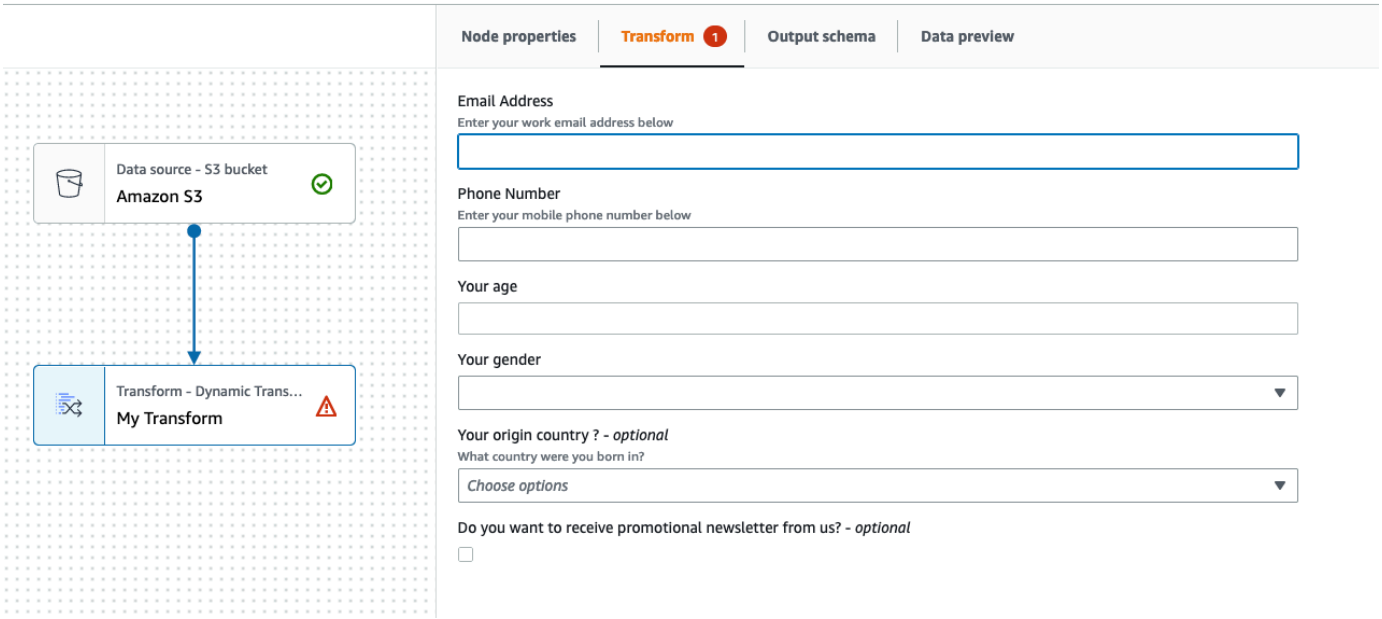
Se ritieni che l'icona predefinita per l'Operazione non la distingue visivamente come parte dei flussi di lavoro, puoi fornire un'icona personalizzata, come descritto nella sezione the section called “ Nozioni di base sulle trasformazioni visive personalizzate ”. È possibile aggiornare l'icona aggiornando il file SVG corrispondente ospitato in Amazon S3.

Per ottenere risultati ottimali, progetta l'immagine in modo che venga visualizzata a 32x32 pixel seguendo le linee guida del Cloudscape Design System. Per ulteriori informazioni sulle linee guida di Cloudscape, consulta la documentazione di Cloudscape.

Fase 5: Utilizzo della trasformazione visiva personalizzata in AWS Glue Studio

Per utilizzare una trasformazione visiva personalizzata in AWS Glue Studio, si devono caricare i file di configurazione e di origine, quindi selezionare la trasformazione dal menu Action (Operazione). Tutti i parametri che richiedono valori o un input sono disponibili nella scheda Transform (Trasforma).

1. Carica i due file (file di origine Python e file di configurazione JSON) nella cartella delle risorse di Amazon S3 in cui sono archiviati gli script di processo. Per impostazione predefinita, AWS Glue estrarrà tutti i file JSON dalla cartella /transforms nello stesso bucket Amazon S3.
2. Dal menu Action (Operazione), scegli la trasformazione visiva personalizzata. Viene denominato con la trasformazione `displayName` o il nome specificato nel file di configurazione .json.
3. Immetti i valori per tutti i parametri configurati nel file di configurazione.



Esempi di utilizzo

Di seguito è riportato un esempio di tutti i parametri possibili in un file di configurazione .json.

```
{
  "name": "MyTransform",
  "displayName": "My Transform",
  "description": "This transform description will be displayed in UI",
  "functionName": "myTransform",
  "parameters": [
      {
      "name": "email",
      "displayName": "Email Address",
      "type": "str",
      "description": "Enter your work email address below",
      "validationType": "RegularExpression",
      "validationRule": "^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*(\\.\\w{2,3})+$",
      "validationMessage": "Please enter a valid email address"
    },
    {
      "name": "phone",
      "displayName": "Phone Number",
      "type": "str",
      "description": "Enter your mobile phone number below",
      "validationRule": "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
      "validationMessage": "Please enter a valid US number"
```

```
    },
    {
      "name": "age",
      "displayName": "Your age",
      "type": "int",
      "isOptional": true
    },
    {
      "name": "gender",
      "displayName": "Your gender",
      "type": "str",
      "listOptions": [
            {"label": "Male", "value": "male"},
            {"label": "Female", "value": "female"},
            {"label": "Other", "value": "other"}
        ],
      "isOptional": true
    },
    {
      "name": "country",
      "displayName": "Your origin country ?",
      "type": "list",
      "listOptions": "Afghanistan,Albania,Algeria,American
 Samoa,Andorra,Angola,Anguilla,Antarctica,Antigua and
 Barbuda,Argentina,Armenia,Aruba,Australia,Austria,Azerbaijan,Bahamas,Bahrain,Bangladesh,Barbad
 and Herzegovina,Botswana,Bouvet Island,Brazil,British Indian Ocean Territory,Brunei
 Darussalam,Bulgaria,Burkina Faso,Burundi,Cambodia,Cameroon,Canada,Cape
 Verde,Cayman Islands,Central African Republic,Chad,Chile,China,Christmas
 Island,Cocos (Keeling Islands),Colombia,Comoros,Congo,Cook Islands,Costa
 Rica,Cote D'Ivoire (Ivory Coast),Croatia (Hrvatska,Cuba,Cyprus,Czech
 Republic,Denmark,Djibouti,Dominica,Dominican Republic,East Timor,Ecuador,Egypt,El
 Salvador,Equatorial Guinea,Eritrea,Estonia,Ethiopia,Falkland Islands (Malvinas),Faroe
 Islands,Fiji,Finland,France,France,Metropolitan,French Guiana,French Polynesia,French
 Southern
 Territories,Gabon,Gambia,Georgia,Germany,Ghana,Gibraltar,Greece,Greenland,Grenada,Guadeloupe,G
Bissau,Guyana,Haiti,Heard and McDonald Islands,Honduras,Hong
 Kong,Hungary,Iceland,India,Indonesia,Iran,Iraq,Ireland,Israel,Italy,Jamaica,Japan,Jordan,Kazak
 (North),Korea
 (South),Kuwait,Kyrgyzstan,Laos,Latvia,Lebanon,Lesotho,Liberia,Libya,Liechtenstein,Lithuania,Lu
 Islands,Martinique,Mauritania,Mauritius,Mayotte,Mexico,Micronesia,Moldova,Monaco,Mongolia,Mont
 Antilles,New Caledonia,New Zealand,Nicaragua,Niger,Nigeria,Niue,Norfolk
 Island,Northern Mariana Islands,Norway,Oman,Pakistan,Palau,Panama,Papua
 New Guinea,Paraguay,Peru,Philippines,Pitcairn,Poland,Portugal,Puerto
 Rico,Qatar,Reunion,Romania,Russian Federation,Rwanda,Saint Kitts and Nevis,Saint
```

```
 Lucia,Saint Vincent and The Grenadines,Samoa,San Marino,Sao Tome and Principe,Saudi
 Arabia,Senegal,Seychelles,Sierra Leone,Singapore,Slovak Republic,Slovenia,Solomon
 Islands,Somalia,South Africa,S. Georgia and S. Sandwich Isls.,Spain,Sri
 Lanka,St. Helena,St. Pierre and Miquelon,Sudan,Suriname,Svalbard and Jan Mayen
 Islands,Swaziland,Sweden,Switzerland,Syria,Tajikistan,Tanzania,Thailand,Togo,Tokelau,Tonga,Tri
 and Tobago,Tunisia,Turkey,Turkmenistan,Turks and Caicos
 Islands,Tuvalu,Uganda,Ukraine,United Arab Emirates,United Kingdom
 (Britain / UK),United States of America (USA),US Minor Outlying
 Islands,Uruguay,Uzbekistan,Vanuatu,Vatican City State (Holy See),Venezuela,Viet
 Nam,Virgin Islands (British),Virgin Islands (US),Wallis and Futuna Islands,Western
 Sahara,Yemen,Yugoslavia,Zaire,Zambia,Zimbabwe",
      "description": "What country were you born in?",
      "listType": "str",
      "isOptional": true
    },
    {
      "name": "promotion",
      "displayName": "Do you want to receive promotional newsletter from us?",
      "type": "bool",
      "isOptional": true
    }
  ]
}
```

Esempi di script visivi personalizzati

Gli esempi seguenti eseguono trasformazioni equivalenti. Tuttavia, il secondo esempio (SparkSQL) è il più pulito ed efficiente, seguito dall'UDF di pandas e infine dalla mappatura a basso livello nel primo esempio. L'esempio seguente è un esempio completo di una semplice trasformazione per sommare due colonne:

```
from awsglue import DynamicFrame

# You can have other auxiliary variables, functions or classes on this file, it won't
 affect the runtime
def record_sum(rec, col1, col2, resultCol):
    rec[resultCol] = rec[col1] + rec[col2]
    return rec


# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
```

```
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    # The mapping will alter the columns order, which could be important
    fields = [field.name for field in self.schema()]
    if resultCol not in fields:
        # If it's a new column put it at the end
        fields.append(resultCol)
    return self.map(lambda record: record_sum(record, col1, col2,
 resultCol)).select_fields(paths=fields)


# The name we assign on DynamicFrame must match the configured "functionName"
DynamicFrame.custom_add_columns = custom_add_columns
```

L'esempio seguente è una trasformazione equivalente che sfrutta l'API SparkSQL.

```
from awsglue import DynamicFrame

# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    df = self.toDF()
    return DynamicFrame.fromDF(
        df.withColumn(resultCol, df[col1] + df[col2]) # This is the conversion logic
        , self.glue_ctx, self.name)


# The name we assign on DynamicFrame must match the configured "functionName"
DynamicFrame.custom_add_columns = custom_add_columns
```

L'esempio seguente utilizza le stesse trasformazioni ma utilizzando un'UDF di pandas, che è più efficiente rispetto all'utilizzo di un'UDF semplice. Per ulteriori informazioni sulla scrittura di UDF di pandas, consulta la documentazione SQL di Apache Spark.

```
from awsglue import DynamicFrame
import pandas as pd
from pyspark.sql.functions import pandas_udf
```

```
# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    @pandas_udf("integer")  # We need to declare the type of the result column
    def add_columns(value1: pd.Series, value2: pd.Series) # pd.Series:
        return value1 + value2

    df = self.toDF()
    return DynamicFrame.fromDF(
        df.withColumn(resultCol, add_columns(col1, col2)) # This is the conversion
 logic
        , self.glue_ctx, self.name)

# The name we assign on DynamicFrame must match the configured "functionName"
DynamicFrame.custom_add_columns = custom_add_columns
```

Video

Il video seguente fornisce un'introduzione alle trasformazioni visive personalizzate e dimostra come utilizzarle.

Utilizzo dei framework data lake con AWS Glue Studio

Panoramica

I framework di data lake open source semplificano l'elaborazione incrementale dei dati per i file archiviati in data lake basati su Amazon S3. AWS Glue 3.0 e versioni successive supportano i seguenti framework di data lake open source:

- Apache Hudi
- Linux Foundation Delta Lake
- Apache Iceberg

A partire da AWS Glue 4.0, AWS Glue fornisce il supporto nativo per questi framework in modo che sia possibile leggere e scrivere i dati archiviati in Amazon S3 in modo coerente dal punto di vista transazionale. Non è necessario installare un connettore separato o completare passaggi di configurazione aggiuntivi per utilizzare questi framework nei processi di AWS Glue.

I framework Data Lake possono essere utilizzati come origine o destinazione all'interno di AWS Glue Studio tramite i processi dell'editor di script Spark. Per ulteriori informazioni sull'utilizzo di Apache Hudi, Apache Iceberg e Delta Lake, consulta Utilizzo di framework di data lake con processi AWS Glue ETL.

Creazione di formati di tabelle aperte da un'origine di streaming AWS Glue

I processi di streaming AWS Glue ETL consumano continuamente dati provenienti da origini di streaming, puliscono e trasformano i dati in corso e li rendono disponibili per l'analisi in pochi secondi.

AWS offre un'ampia selezione di servizi per soddisfare le tue esigenze. Un servizio di replica del database come Database Migration Service AWS può replicare i dati dai sistemi di origine su Amazon S3, che di solito ospita il livello di storage del data lake. Sebbene sia semplice applicare gli aggiornamenti su un sistema di gestione di database relazionale (RDBMS) che supporta un'applicazione di origine online, è difficile applicare questo processo CDC sui data lake. I framework di gestione dei dati open-source semplificano l'elaborazione incrementale dei dati e lo sviluppo di pipeline di dati e sono una buona opzione per risolvere questo problema.

Per ulteriori informazioni, consulta:

- Crea un data lake transazionale basato su Apache Hudi, quasi in tempo reale, utilizzando Streaming di AWS Glue
- Crea un data lake Apache Iceberg allineato al GDPR in tempo reale

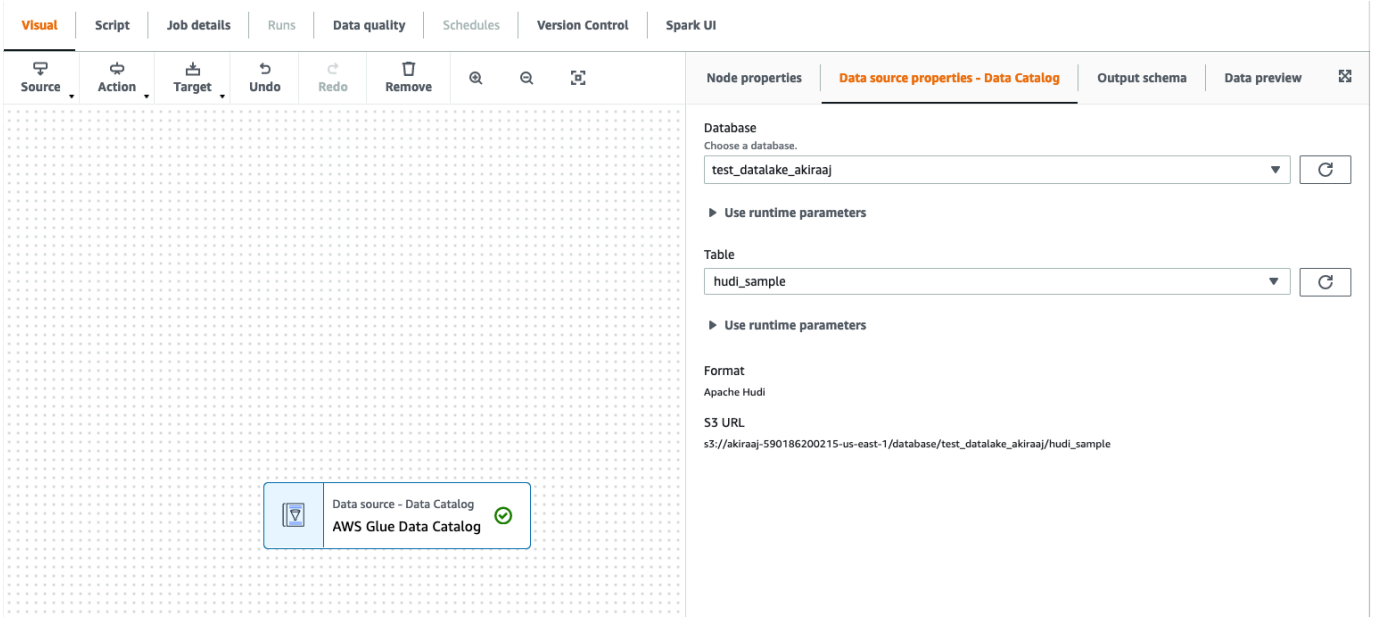
Utilizzo del framework Hudi in AWS Glue Studio

Quando crei o modifichi un processo, AWS Glue Studio aggiunge automaticamente le librerie Hudi corrispondenti a seconda della versione di AWS Glue che stai utilizzando. Per ulteriori informazioni, consulta Utilizzo del framework di Hudi in AWS Glue.

Utilizzo del framework di Apache Hudi nelle origini dati del catalogo dati

Per aggiungere un formato di origine dati di Hudi a un processo:

1. Dal menu Origine, scegli Catalogo dati AWS Glue Studio.
2. Nella scheda Proprietà dell'origine dati, scegli un database e una tabella.
3. AWS Glue Studio mostra il tipo di formato come Apache Hudi e l'URL di Amazon S3.



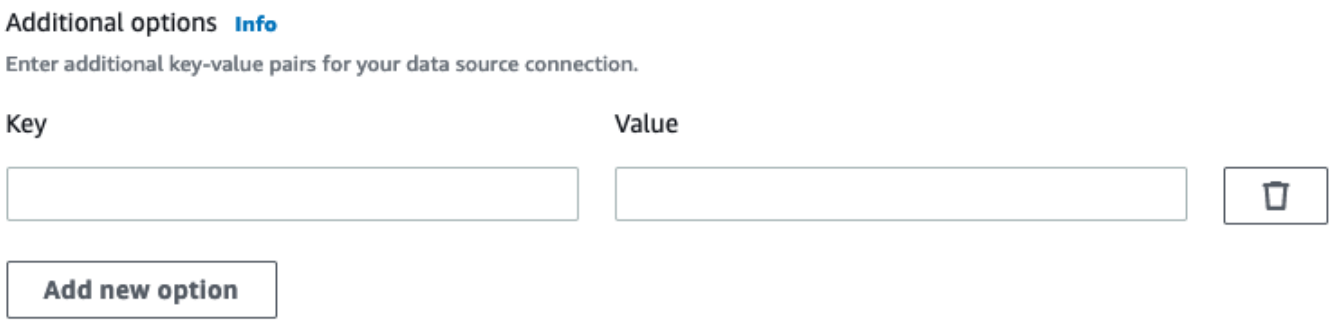
Utilizzo del framework di Hudi nelle origini dati di Amazon S3

1. Dal menu Origine, scegli Amazon S3.
2. Se scegli la tabella del catalogo dati come tipo di origine di Amazon S3, scegli un database e una tabella.
3. AWS Glue Studio mostra il formato come Apache Hudi e l'URL di Amazon S3.
4. Se scegli la posizione Amazon S3 come tipo di origine Amazon S3, scegli l'URL di Amazon S3 facendo clic su Sfoglia Amazon S3.
5. In Formato dati, seleziona Apache Hudi.

> **Note**
>
> Se AWS Glue Studio non riesce a inferire lo schema dalla cartella o dal file Amazon S3 che hai selezionato, scegli Opzioni aggiuntive per selezionare una nuova cartella o un nuovo file.
> In Opzioni aggiuntive, scegli tra le seguenti opzioni in Inferenza dello schema:
>
> - Lascia che AWS Glue Studio scelga automaticamente un file di esempio: AWS Glue Studio sceglierà un file di esempio nella posizione di Amazon S3 in modo da poter inferire lo schema. Nel campo File con campionatura automatica, puoi visualizzare il file che è stato selezionato automaticamente.

- Scegli un file di esempio da Amazon S3: scegli il file Amazon S3 da utilizzare facendo clic su Sfoglia Amazon S3.

6. Fai clic su Inferisci schema. A questo punto potrai visualizzare lo schema di output facendo clic sulla scheda Schema di output.
7. Scegli Opzioni aggiuntive per inserire una coppia chiave-valore.



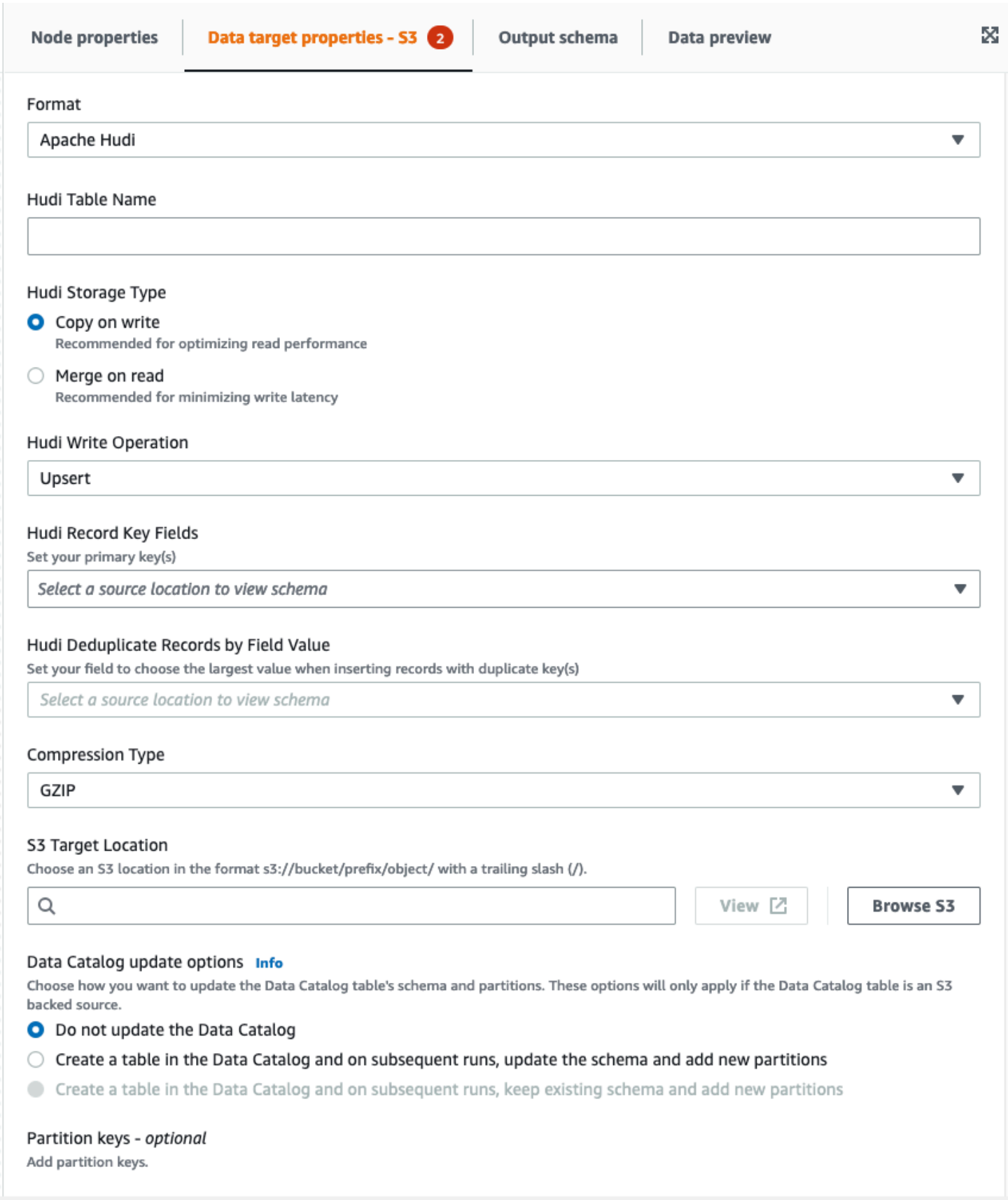
Utilizzo del framework di Apache Hudi nelle destinazioni dei dati

Utilizzo del framework di Apache Hudi nelle destinazioni dei dati del catalogo dati

1. Dal menu Destinazione, scegli Catalogo dati AWS Glue Studio.
2. Nella scheda Proprietà dell'origine dati, scegli un database e una tabella.
3. AWS Glue Studio mostra il tipo di formato come Apache Hudi e l'URL di Amazon S3.

Utilizzo del framework di Apache Hudi nelle destinazioni dei dati di Amazon S3

Inserisci valori o scegli tra le opzioni disponibili per configurare il formato di Apache Hudi. Per ulteriori informazioni su Apache Hudi, consulta la documentazione di Apache Hudi.


Node properties
Data target properties - S3 2
Output schema
Data preview
Format
Apache Hudi
Hudi Table Name
Hudi Storage Type
Copy on write
Recommended for optimizing read performance
Merge on read
Recommended for minimizing write latency
Hudi Write Operation
Upsert
Hudi Record Key Fields
Set your primary key(s)
Select a source location to view schema
Hudi Deduplicate Records by Field Value
Set your field to choose the largest value when inserting records with duplicate key(s)
Select a source location to view schema
Compression Type
GZIP
S3 Target Location
Choose an S3 location in the format s3://bucket/prefix/object/ with a trailing slash (/).
View
Browse S3
Data Catalog update options Info
Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog table is an S3 backed source.
Do not update the Data Catalog
Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions
Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions
Partition keys - optional
Add partition keys.

- Nome tabella Hudi: questo è il nome della tua tabella Hudi.
- Tipo di archiviazione Hudi - scegli tra due opzioni:
  - Copia in scrittura: consigliata per ottimizzare le prestazioni di lettura. È il tipo di archiviazione di Hudi predefinito. Ogni aggiornamento crea una nuova versione dei file durante una scrittura.
  - Unisci in lettura: consigliata per ridurre al minimo la latenza di scrittura. Gli aggiornamenti vengono registrati nei file delta basati su righe e vengono compattati come necessario per creare nuove versioni dei file colonnari.
- Operazione di scrittura di Hudi - scegli una delle seguenti opzioni:
  - Upsert: questa è l'operazione predefinita in cui i record di input vengono prima contrassegnati come inserimenti o aggiornamenti cercando l'indice. Consigliata laddove stai aggiornando dati esistenti.
  - Inserimento: inserisce i record ma non verifica i record esistenti e può generare duplicati.
  - Inserimento in blocco: consente di inserire record ed è consigliato per grandi quantità di dati.
- Campi chiave record Hudi: utilizza la barra di ricerca per cercare e scegliere le chiavi record primarie. I record in Hudi sono identificati da una chiave primaria che è una coppia composta da chiave di record e percorso di partizione a cui appartiene il record.
- Campo di precombinazione Hudi: questo è il campo utilizzato in "preCombining" prima della scrittura effettiva. Quando due record hanno lo stesso valore chiave, AWS Glue Studio seleziona quello con il valore più grande per il campo di precombinazione. Imposta un campo con valore incrementale (ad esempio updated_at) a cui appartiene.
- Tipo di compressione: scegli una delle opzioni per il tipo di compressione: Uncompressed, GZIP, LZO o Snappy.
- Posizione di destinazione di Amazon S3: scegli la posizione di destinazione di Amazon S3 facendo clic su Sfoglia S3.
- Opzioni di aggiornamento del catalogo dati - scegli una delle seguenti opzioni:
  - Do not update the Data Catalog (Non aggiornare il catalogo dati): (impostazione predefinita) scegli questa opzione se non vuoi che il processo aggiorni il catalogo dati, anche se lo schema viene modificato o sono aggiunte nuove partizioni.
  - Crea una tabella nel catalogo dati e, nelle esecuzioni successive, aggiorna lo schema e aggiungi nuove partizioni: se scegli questa opzione, il processo crea la tabella nel catalogo dati alla prima esecuzione. Nelle successive esecuzioni del processo, questo aggiorna la tabella del catalogo dati se lo schema viene modificato o sono aggiunte nuove partizioni.

    Devi inoltre selezionare un database dal catalogo dati e inserire un nome di tabella.

- Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions (Crea una tabella nel catalogo dati e, nelle esecuzioni successive, mantieni lo schema esistente e aggiungi nuove partizioni): se scegli questa opzione, il processo crea la tabella nel catalogo dati alla prima esecuzione. Nelle successive esecuzioni del processo, questo aggiorna la tabella del catalogo dati solo per aggiungere nuove partizioni.

  Devi inoltre selezionare un database dal catalogo dati e inserire un nome di tabella.

- Partition keys (Chiavi di partizione): scegli quali colonne utilizzare come chiavi di partizionamento nell'output. Per aggiungere altre chiavi di partizione, scegli Add a partition key (Aggiungi una chiave di partizione).
- Opzioni aggiuntive: inserisci una coppia chiave-valore, se necessario.

Generazione di codice tramite AWS Glue Studio

Quando il processo viene salvato, i seguenti parametri di processo vengono aggiunti al processo se viene rilevata un'origine o una destinazione Hudi:

- `--datalake-formats`: un elenco distinto di formati di data lake rilevati nel processo visivo (direttamente scegliendo un "Formato" o indirettamente selezionando una tabella di catalogo supportata da un data lake).
- `--conf` : generato in base al valore di `--datalake-formats`. Ad esempio, se il valore per `--datalake-formats` è "hudi", AWS Glue genera un valore di `spark.serializer=org.apache.spark.serializer.KryoSerializer –conf spark.sql.hive.convertMetastoreParquet=false` per questo parametro.

Sostituzione delle librerie fornite da AWS Glue

Per utilizzare una versione di Hudi che AWS Glue non supporta, puoi specificare i tuoi file JAR della libreria di Hudi. Per usare il tuo file JAR:

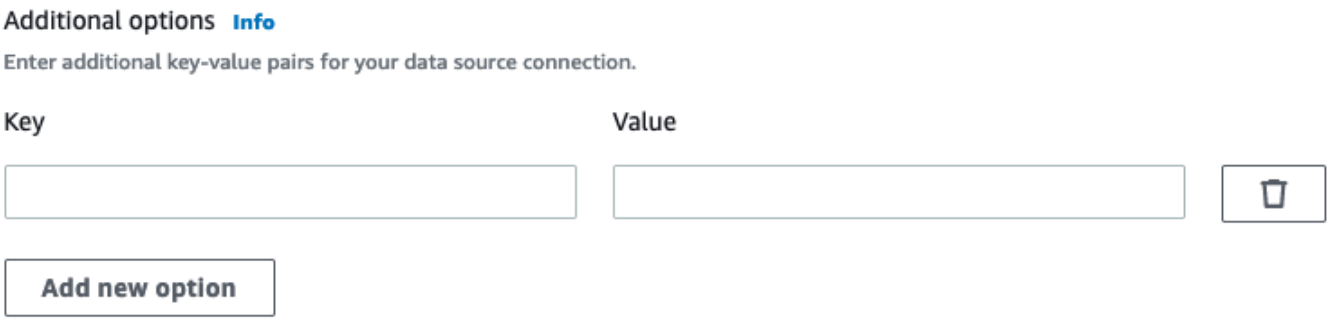
- utilizza il parametro del processo `--extra-jars`. Ad esempio, `'--extra-jars': 's3pathtojarfile.jar'`. Per ulteriori informazioni, consulta i parametri del processo AWS Glue.
- Non includere `hudi` come valore per il parametro del processo `--datalake-formats`. L'immissione di una stringa vuota come valore garantisce che nessuna libreria di data lake venga fornita automaticamente da AWS Glue. Per ulteriori informazioni, consulta Utilizzo del framework di Hudi in AWS Glue.

Utilizzo del framework Delta Lake in AWS Glue Studio

Utilizzo del framework Delta Lake in origini dati

Utilizzo del framework Delta Lake in origini dati Amazon S3

1. Dal menu Origine, scegli Amazon S3.
2. Se scegli la tabella del catalogo dati come tipo di origine di Amazon S3, scegli un database e una tabella.
3. AWS Glue Studio mostra il formato come Delta Lake e l'URL di Amazon S3.
4. Scegli Opzioni aggiuntive per inserire una coppia chiave-valore. Ad esempio, una coppia chiave-valore potrebbe essere: chiave: timestampAsOf e valore: 2023-02-24 14:16:18.



5. Se scegli la posizione Amazon S3 come tipo di origine Amazon S3, scegli l'URL di Amazon S3 facendo clic su Sfoglia Amazon S3.
6. In Formato data, scegli Delta Lake.

> **Note**
>
> Se AWS Glue Studio non riesce a inferire lo schema dalla cartella o dal file Amazon S3 che hai selezionato, scegli Opzioni aggiuntive per selezionare una nuova cartella o un nuovo file.
> In Opzioni aggiuntive, scegli tra le seguenti opzioni in Inferenza dello schema:
>
> - Lascia che AWS Glue Studio scelga automaticamente un file di esempio: AWS Glue Studio sceglierà un file di esempio nella posizione di Amazon S3 in modo da poter inferire lo schema. Nel campo File con campionatura automatica, puoi visualizzare il file che è stato selezionato automaticamente.

- Scegli un file di esempio da Amazon S3: scegli il file Amazon S3 da utilizzare facendo clic su Sfoglia Amazon S3.

7. Fai clic su Inferisci schema. A questo punto potrai visualizzare lo schema di output facendo clic sulla scheda Schema di output.

Utilizzo del framework Delta Lake in origini dati Catalogo dati

1. Dal menu Origine, scegli Catalogo dati AWS Glue Studio.
2. Nella scheda Proprietà dell'origine dati, scegli un database e una tabella.
3. AWS Glue Studio mostra il tipo di formato come Delta Lake e l'URL di Amazon S3.

> **Note**
>
> Se la tua origine Delta Lake non è ancora registrata come tabella del catalogo dati AWS Glue, hai due opzioni:
>
> 1. Creare un crawler AWS Glue per l'archivio dati Delta Lake. Per ulteriori informazioni, consulta Come specificare le opzioni di configurazione per un archivio dati Delta Lake.
> 2. Utilizzare un'origine dati Amazon S3 per selezionare la tua origine dati Delta Lake. Per informazioni, consultare Utilizzo del framework Delta Lake in origini dati Amazon S3 .

Utilizzo dei formati Delta Lake negli obiettivi dei dati

Utilizzo dei formati Delta Lake negli obiettivi dei dati del Catalogo dati

1. Dal menu Destinazione, scegli Catalogo dati AWS Glue Studio.
2. Nella scheda Proprietà dell'origine dati, scegli un database e una tabella.
3. AWS Glue Studio mostra il tipo di formato come Delta Lake e l'URL di Amazon S3.

Utilizzo dei formati Delta Lake nelle origini dati di Amazon S3

Inserisci valori o scegli tra le opzioni disponibili per configurare il formato di Delta Lake.

- Tipo di compressione: scegli una delle opzioni per il tipo di compressione: Uncompressed o Snappy.

- Posizione di destinazione di Amazon S3: scegli la posizione di destinazione di Amazon S3 facendo clic su Sfoglia S3.
- Opzioni di aggiornamento del Catalogo dati: l'aggiornamento del Catalogo dati non è supportato per questo formato nell'editor visivo di Glue Studio.
  - Do not update the Data Catalog (Non aggiornare il catalogo dati): (impostazione predefinita) scegli questa opzione se non vuoi che il processo aggiorni il catalogo dati, anche se lo schema viene modificato o sono aggiunte nuove partizioni.
  - Per aggiornare il Catalogo dati dopo l'esecuzione del processo AWS Glue, esegui o pianifica un crawler AWS Glue. Per ulteriori informazioni, consulta Come specificare le opzioni di configurazione per un archivio dati Delta Lake.
- Chiavi di partizione: scegli quali colonne utilizzare come chiavi di partizionamento nell'output. Per aggiungere altre chiavi di partizione, scegli Add a partition key (Aggiungi una chiave di partizione).
- Facoltativamente, scegli Opzioni aggiuntive per inserire una coppia chiave-valore. Ad esempio, una coppia chiave-valore potrebbe essere: chiave: timestampAsOf e valore: 2023-02-24 14:16:18.

Utilizzo del framework Apache Iceberg in AWS Glue Studio

Utilizzo del framework Apache Iceberg nelle destinazioni dati

Utilizzo del framework Apache Iceberg nelle destinazioni dati di Catalogo dati

1. Dal menu Destinazione, scegli Catalogo dati AWS Glue Studio.
2. Nella scheda Proprietà dell'origine dati, scegli un database e una tabella.
3. AWS Glue Studio mostra il tipo di formato come Apache Iceberg e l'URL di Amazon S3.

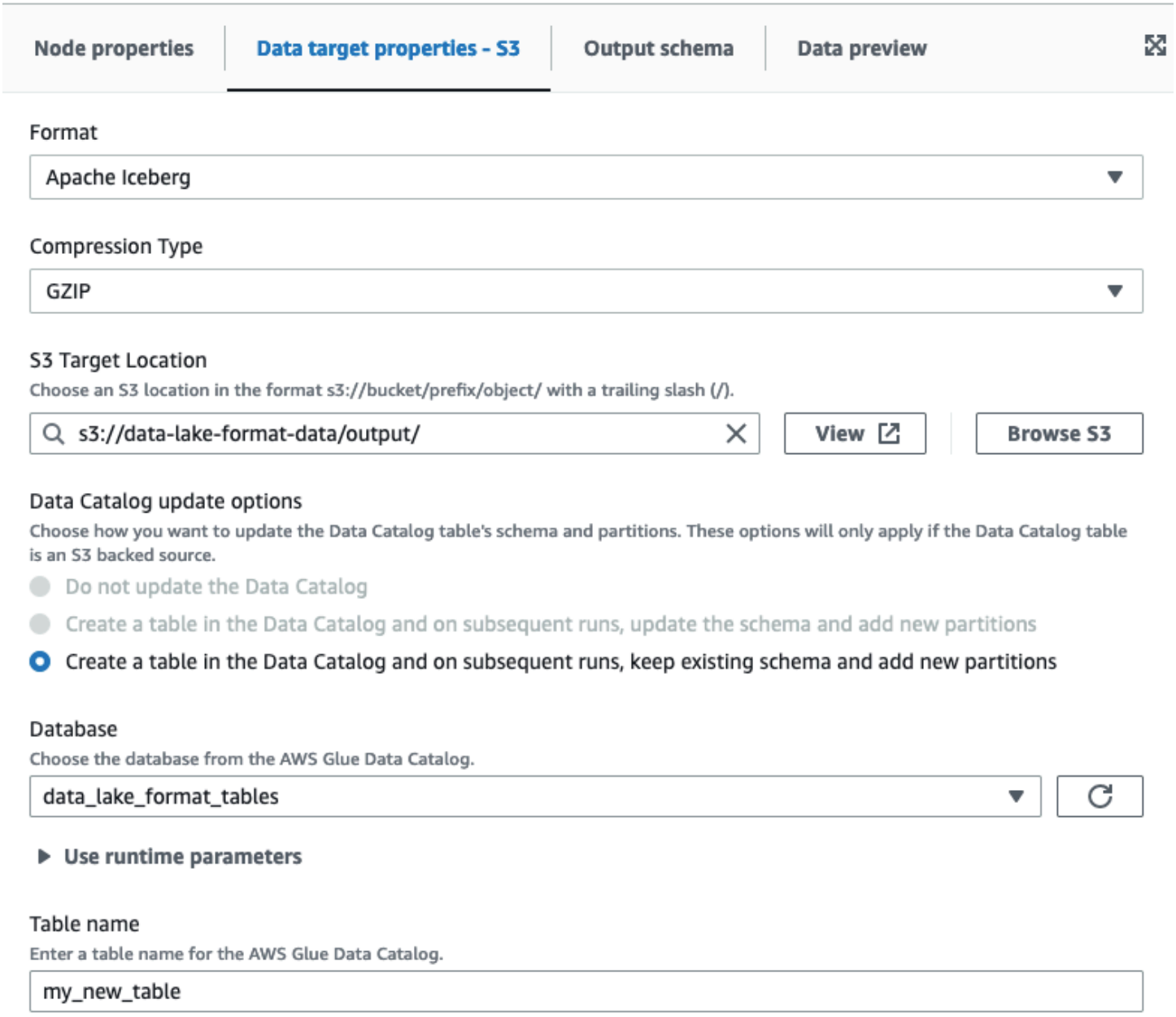
Utilizzo del framework Apache Iceberg nelle destinazioni dati di Amazon S3

Inserisci valori o scegli tra le opzioni disponibili per configurare il formato di Apache Iceberg.

- Formato: scegli Apache Iceberg dal menu a discesa.
- Posizione di destinazione di Amazon S3: scegli la posizione di destinazione di Amazon S3 facendo clic su Sfoglia S3.
- Opzioni di aggiornamento del Catalogo dati: è necessario selezionare Crea una tabella in Catalogo dati e, nelle esecuzioni successive, mantieni lo schema esistente e aggiungi nuove partizioni per procedere. La scrittura di una nuova tabella Iceberg utilizzando AWS Glue richiede che Data

Catalog venga configurato come catalogo per la tabella Iceberg. Per aggiornare una tabella Iceberg esistente che è stata registrata in Data Catalog, scegli Data Catalog come destinazione.

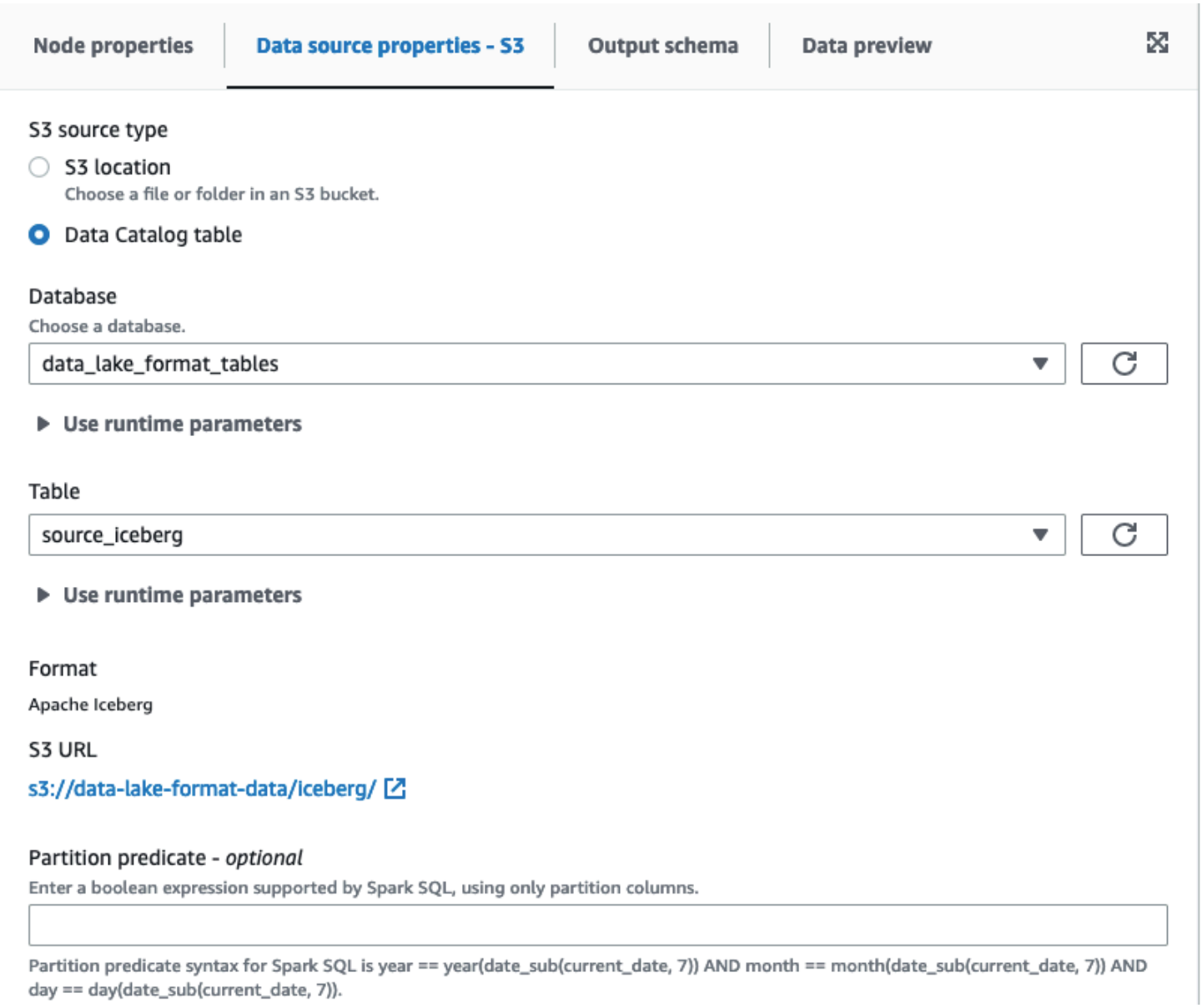
- Database: scegli il database dal Data Catalog.
- Nome tabella: inserisci un nome univoco per la tabella. I nomi delle tabelle di Apache Iceberg devono essere tutti in minuscolo. Utilizza i caratteri di sottolineatura, se necessario, poiché gli spazi non sono consentiti. Ad esempio "data_lake_format_tables".



Utilizzo del framework Apache Iceberg nelle origini dati di Amazon S3

Utilizzo del framework Apache Iceberg nelle origini dati di Catalogo dati

1. Dal menu Origine, scegli Catalogo dati AWS Glue Studio.
2. Nella scheda Proprietà dell'origine dati, scegli un database e una tabella.
3. AWS Glue Studio mostra il tipo di formato come Apache Iceberg e l'URL di Amazon S3.



Utilizzo del framework Apache Iceberg nelle origini dati di Amazon S3

Apache Iceberg non è disponibile come opzione dati per i nodi origine di Amazon S3 in AWS Glue Studio.

Configurazione dei nodi di destinazione dati

La destinazione dati è la posizione in cui il processo scrive i dati trasformati.

Panoramica delle opzioni di destinazione dati

La destinazione dati (chiamata anche sink dei dati) può essere:

- S3 – Il processo scrive i dati in un file nella posizione Amazon S3 scelta e nel formato specificato.

  Se configuri le colonne di partizione per la destinazione dati, il processo scrive il set di dati su Amazon S3 in directory basate sulla chiave di partizione.
- AWS Glue Data Catalog – Il processo utilizza le informazioni associate alla tabella nel catalogo dati per scrivere i dati di output in una posizione di destinazione.

  Puoi creare la tabella manualmente o con il crawler. Puoi utilizzare anche modelli AWS CloudFormation per creare tabelle nel catalogo dati.
- Un connettore – un connettore è una parte di codice che facilita la comunicazione tra l'archivio dati e AWS Glue. Il processo utilizza il connettore e la connessione associata per scrivere i dati di output in una posizione di destinazione. Puoi effettuare la sottoscrizione a un connettore offerto in Marketplace AWS oppure puoi creare un connettore personalizzato. Per ulteriori informazioni, consulta Aggiunta di connettori a AWS Glue Studio.

Puoi scegliere di aggiornare il catalogo dati quando il tuo processo scrive in una destinazione dati Amazon S3. Anziché richiedere a un crawler di aggiornare il catalogo dati quando lo schema o le partizioni cambiano, questa opzione semplifica l'aggiornamento delle tabelle. Questa opzione semplifica il processo che rende disponibili i dati per l'analisi aggiungendo facoltativamente nuove tabelle al catalogo dati, aggiornando le partizioni di tabella e aggiornando lo schema delle tabelle direttamente dal processo.

Modifica del nodo di destinazione dati

La destinazione dati è la posizione in cui il processo scrive i dati trasformati.

Per aggiungere o configurare un nodo di destinazione dati nel diagramma di processo

1. (Facoltativo) Se devi aggiungere un nodo di destinazione, scegli Target (Destinazione) nella barra degli strumenti nella parte superiore dell'editor visivo, quindi scegli S3 o Glue Data Catalog.

   - Se scegli S3 per la destinazione, il processo scrive il set di dati in uno o più file nella posizione Amazon S3 specificata.
   - Se scegli AWS Glue Data Catalog per la destinazione, il processo scrive in una posizione descritta dalla tabella selezionata dal catalogo dati.

2. Scegli un nodo di destinazione dati nel diagramma del processo. Quando scegli un nodo, il pannello dei dettagli del nodo viene visualizzato sul lato destro della pagina.
3. Seleziona la scheda Node properties (Proprietà del nodo), quindi inserisci le informazioni riportate di seguito:

   - Name (Nome): inserisci un nome da associare al nodo nel diagramma del processo.
   - Node type (Tipo di nodo): dovrebbe essere già selezionato un valore, ma è possibile modificarlo in base alle necessità.
   - Node parents (Nodi padre): il nodo padre è il nodo nel diagramma del processo che fornisce i dati di output da scrivere nella posizione di destinazione. Per un diagramma di processo precompilato, il nodo di destinazione deve già avere il nodo padre selezionato. Se non è visualizzato alcun nodo padre, scegline uno dall'elenco.

     Un nodo di destinazione ha un singolo nodo padre.

4. Configura le informazioni di Data target properties (Proprietà della destinazione dati). Per ulteriori informazioni, consulta le sezioni seguenti:

   - Uso di Amazon S3 per la destinazione dati
   - Utilizzo delle tabelle del catalogo dati per la destinazione dati
   - Utilizzo di un connettore per la destinazione dati

5. (Facoltativo) Dopo aver configurato le proprietà del nodo di destinazione dati, puoi visualizzare lo schema di output per i dati scegliendo la scheda Output schema (Schema di output) nel pannello dei dettagli del nodo. La prima volta che si sceglie questa scheda per qualsiasi nodo del processo, viene richiesto di fornire un ruolo IAM per accedere ai dati. Se non è stato specificato un ruolo IAM nella scheda Job details (Dettagli del processo), viene richiesto di immettere un ruolo IAM a questo punto.

Uso di Amazon S3 per la destinazione dati

Per tutte le origini dati ad eccezione di Amazon S3 e dei connettori, è necessario che esista una tabella in AWS Glue Data Catalog per il tipo di origine scelto. AWS Glue Studio non crea la tabella in Data Catalog.

Per configurare un nodo di destinazione dati che scrive su Amazon S3

1. Vai all'editor visivo per un processo nuovo o salvato.
2. Scegli un nodo di origine dati nel diagramma del processo.
3. Seleziona la scheda Data source properties (Proprietà dell'origine dati), quindi immetti le informazioni riportate di seguito:

   - Format (Formato): Scegli un formato dall'elenco. I tipi di formato disponibili per i risultati dei dati sono:
     - JSON: JavaScript Object Notation.
     - CSV: valori separati da virgola.
     - Avro: Apache Avro JSON binario.
     - Parquet: storage a colonne Apache Parquet.
     - Glue Parquet: un tipo personalizzato di writer Parquet ottimizzato per `DynamicFrames`come formato dei dati. Anziché richiedere uno schema precalcolato per i dati, calcola e modifica lo schema in modo dinamico.
     - ORC: formato Apache Optimized Row Columnar (ORC).

     Per ulteriori informazioni su queste opzioni di formato, consulta Opzioni di formato per gli input e output ETL in AWS Glue nella Guida per gli sviluppatori di AWS Glue.
   - Compression Type (Tipo di compressione): puoi scegliere di comprimere i dati tramite `gzip` o `bzip2`. L'impostazione predefinita non è alcuna compressione, oppure None (Nessuna).
   - S3 Target Location (Posizione di destinazione S3): il bucket Amazon S3 e la posizione per l'output dei dati. Puoi selezionare il pulsante Browse S3 (Sfoglia S3) per visualizzare i bucket Amazon S3 a cui hai accesso e sceglierne uno come destinazione.
   - Opzioni per l'aggiornamento del catalogo dati
     - Do not update the Data Catalog (Non aggiornare il catalogo dati): (impostazione predefinita) scegli questa opzione se non vuoi che il processo aggiorni il catalogo dati, anche se lo schema viene modificato o sono aggiunte nuove partizioni.

- Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions (Crea una tabella nel catalogo dati e, nelle esecuzioni successive, aggiorna lo schema e aggiungi nuove partizioni): se scegli questa opzione, il processo crea la tabella nel catalogo dati alla prima esecuzione. Nelle successive esecuzioni del processo, questo aggiorna la tabella del catalogo dati se lo schema viene modificato o sono aggiunte nuove partizioni.

  Devi inoltre selezionare un database dal catalogo dati e inserire un nome di tabella.
- Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions (Crea una tabella nel catalogo dati e, nelle esecuzioni successive, mantieni lo schema esistente e aggiungi nuove partizioni): se scegli questa opzione, il processo crea la tabella nel catalogo dati alla prima esecuzione. Nelle successive esecuzioni del processo, questo aggiorna la tabella del catalogo dati solo per aggiungere nuove partizioni.

  Devi inoltre selezionare un database dal catalogo dati e inserire un nome di tabella.
- Partition keys (Chiavi di partizione): scegli quali colonne utilizzare come chiavi di partizionamento nell'output. Per aggiungere altre chiavi di partizione, scegli Add a partition key (Aggiungi una chiave di partizione).

Utilizzo delle tabelle del catalogo dati per la destinazione dati

Per tutte le origini dati ad eccezione di Amazon S3 e dei connettori, è necessario che esista una tabella in AWS Glue Data Catalog per il tipo di destinazione scelto. AWS Glue Studio non crea la tabella in Data Catalog.

Per configurare le proprietà dei dati per una destinazione che utilizza una tabella del catalogo dati

1. Vai all'editor visivo per un processo nuovo o salvato.
2. Scegli un nodo di destinazione dati nel diagramma del processo.
3. Seleziona la scheda Data target properties (Proprietà della destinazione dati), quindi inserisci le informazioni riportate di seguito:

   - Database: scegli dall'elenco il database che contiene la tabella da utilizzare come destinazione. Questo database deve esistere già nel catalogo dati.
   - Table (Tabella): scegli la tabella che definisce lo schema dei dati di output dall'elenco. Questa tabella deve esistere già nel catalogo dati.

Una tabella nel catalogo dati contiene i nomi delle colonne, le definizioni dei tipi di dati, le informazioni sulle partizioni e altri metadati su un set di dati di destinazione. Il processo scrive in una posizione descritta da questa tabella nel catalogo dati.

Per ulteriori informazioni sulla creazione di tabelle nel catalogo dati, consulta Definizione di tabelle nel Catalogo dati nella Guida per gli sviluppatori di AWS Glue.

- Opzioni per l'aggiornamento del catalogo dati
  - Do not change table definition (Non modificare la definizione della tabella): (impostazione predefinita) scegli questa opzione se non vuoi che il processo aggiorni il catalogo dati, anche se lo schema viene modificato o sono aggiunte nuove partizioni.
  - Update schema and add new partitions (Aggiorna lo schema e aggiungi nuove partizioni): se scegli questa opzione, il processo aggiorna la tabella del catalogo dati se lo schema viene modificato o sono aggiunte nuove partizioni.
  - Keep existing schema and add new partitions (Mantieni lo schema esistente e aggiungi nuove partizioni): se scegli questa opzione, il processo aggiorna la tabella del catalogo dati solo per aggiungere nuove partizioni.
  - Partition keys (Chiavi di partizione): scegli quali colonne utilizzare come chiavi di partizionamento nell'output. Per aggiungere altre chiavi di partizione, scegli Add a partition key (Aggiungi una chiave di partizione).

Utilizzo di un connettore per la destinazione dati

Se per Node type (Tipo di nodo) selezioni un connettore, segui le istruzioni in Creazione di processi con connettori personalizzati per completare la configurazione delle proprietà della destinazione dati.

Modifica o caricamento di uno script del processo

Utilizzo dell'editor visivo AWS Glue Studio per modificare lo script del processo o caricare il proprio script.

È possibile utilizzare l'editor visivo per modificare i nodi di processo solo se i processi sono stati creati con AWS Glue Studio. Se il processo è stato creato utilizzando la console AWS Glue, tramite i comandi API o con l'interfaccia a riga di comando (CLI), puoi utilizzare l'editor di script in AWS Glue Studio per modificare lo script del processo, i parametri e la pianificazione. Puoi anche modificare lo script per un processo creato in AWS Glue Studio convertendo il processo in modalità solo script.

Per modificare lo script del processo o caricare il proprio script

1. Se crei un nuovo processo, nella pagina Jobs (Processi), seleziona l'opzione Spark script editor (Editor di script Spark) per creare un processo Spark o scegli l'opzione Python Shell script editor (Editor di script shell Python) per creare un processo shell Python. Puoi scrivere un nuovo script o caricare uno script esistente. Se scegli Spark script editor (Editor di script Spark), puoi scrivere o caricare uno script Scala o Python. Se scegli Python Shell script editor (Editor di script shell Python), puoi scrivere o caricare solo uno script Python.

   Dopo aver selezionato l'opzione per creare un nuovo processo, nella sezione Options (Opzioni) che appare, puoi scegliere di iniziare con uno script di inizio (Create a new script with boilerplate code [Crea un nuovo script con codice boilerplate]), oppure puoi caricare un file locale da utilizzare come script del processo.

   Se hai scelto Spark script editor (Editor di script Spark), puoi caricare un file script Python o Scala. Gli script Scala devono avere l'estensione di file `.scala`. Gli script Python devono essere riconosciuti come file di tipo Python. Se hai scelto Python Shell script editor (Editor di script shell Python), puoi caricare solo file di script Python.

   Una volta completate le scelte, seleziona Create (Crea) per creare il processo e aprire l'editor visivo.
2. Vai all'editor di processo visivo per il processo nuovo o salvato, quindi seleziona la scheda Script.
3. Se non hai creato un nuovo processo utilizzando una delle opzioni dell'editor di script e non hai mai modificato lo script per un processo esistente, la scheda Script mostra l'intestazione Script (Locked) (Script [bloccato]). Ciò significa che l'editor di script è in modalità di sola lettura. Scegli Edit script (Modifica script) per sbloccare lo script per la modifica.

   Per rendere lo script modificabile, AWS Glue Studio converte il processo da processo visivo a processo solo script. Sbloccando lo script per la modifica, non puoi più utilizzare l'editor visivo per questo processo dopo averlo salvato.

   Nella finestra di conferma, scegli Confirm (Conferma) per continuare o Cancel (Annulla) per mantenere il processo disponibile per la modifica visiva.

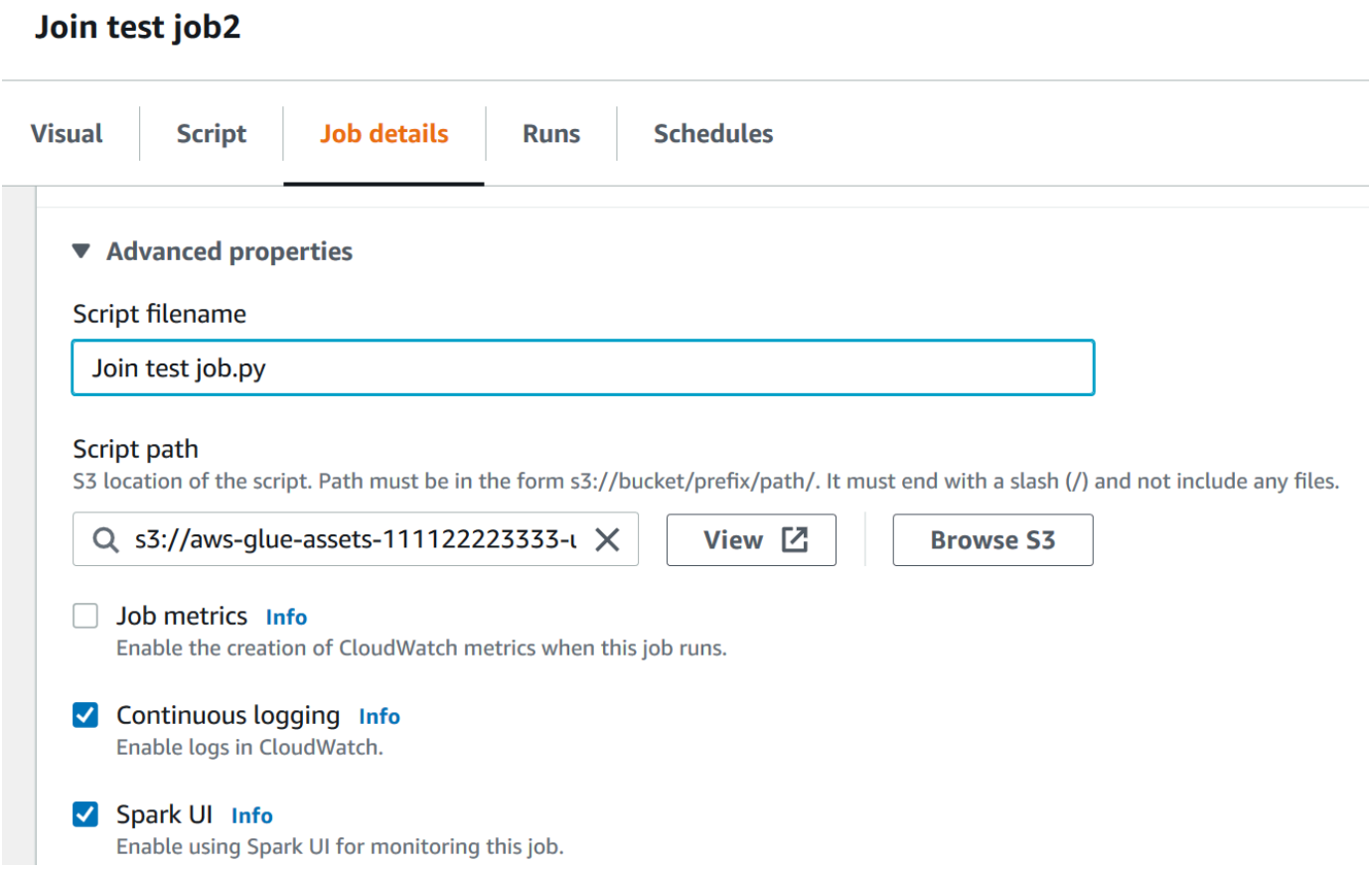
   Scegliendo Confirm (Conferma), la scheda Visual (Visivo) non viene più mostrata nell'editor. Puoi utilizzare AWS Glue Studio per modificare lo script utilizzando l'editor di script, modificare i dettagli o la pianificazione del processo o visualizzarne le esecuzioni.

> **Note**
>
> Fino a quando non salvi il processo, la conversione in un processo solo script non è permanente. Se aggiorni la pagina Web della console o chiudi il processo prima di salvarlo e lo riapri nell'editor visivo, potrai ancora modificare i singoli nodi nell'editor visivo.

4. Modifica lo script in base alle esigenze.

   Dopo aver modificato lo script, seleziona Save (Salva) per salvare il processo e convertirlo in modo permanente da visivo a solo script.

5. (Facoltativo) Puoi scaricare lo script dalla console AWS Glue Studio selezionando il pulsante Download (Scarica) nella scheda Script. Selezionando questo pulsante, si apre una nuova finestra del browser che mostra lo script dalla sua posizione in Amazon S3. I parametri Script filename (Nome del file di script) e Script path (Percorso dello script) nella scheda del processo Job details (Dettagli del processo) determinano il nome e la posizione del file di script in Amazon S3.



Quando salvi il processo, AWS Glue salva lo script del processo nella posizione specificata da questi campi. Se modifichi il file di script in questa posizione all'interno di Amazon S3, AWS Glue Studio caricherà lo script modificato alla successiva modifica del processo.

Creazione e modifica di script Scala in AWS Glue Studio

Quando scegli l'editor di script per la creazione di un processo, per impostazione predefinita, il linguaggio di programmazione dei processi è impostato su `Python 3`. Se scegli di scrivere un nuovo script invece di caricarne uno, AWS Glue Studio avvia un nuovo script con testo boilerplate scritto in Python. Se invece vuoi scrivere uno script Scala, devi prima configurare l'editor di script per utilizzare Scala.

> **Note**
>
> Se scegli Scala come linguaggio di programmazione per il processo e usi l'editor visivo per progettare il processo, lo script del processo generato viene scritto in Scala e non sono necessarie ulteriori azioni.

Come scrivere un nuovo script Scala in AWS Glue Studio

1. Crea un nuovo processo scegliendo l'opzione Spark script editor (Editor di script Spark).
2. Sotto Options (Opzioni), scegli Create a new script with boilerplate code (Crea un nuovo script con codice boilerplate).
3. Seleziona Job details (Dettagli del processo) e imposta Language (Linguaggio) su `Scala` (invece di `Python 3`).

   > **Note**
   >
   > La proprietà Type (Tipo) per il processo viene automaticamente impostata su `Spark` quando scegli l'opzione Spark script editor (Editor di script Spark) per creare un processo.

4. Seleziona la scheda Script.
5. Rimuovi il testo boilerplate Python. Puoi sostituirlo con il seguente testo boilerplate Scala.

```
import com.amazonaws.services.glue.{DynamicRecord, GlueContext}
```

```
import org.apache.spark.SparkContext
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job

object MyScript {
  def main(args: Array[String]): Unit = {
    val sc: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(sc)

    }
}
```

6. Scrivi lo script del processo Scala nell'editor. Aggiungi ulteriori istruzioni `import` in base alle esigenze.

Creazione e modifica di processi shell Python in AWS Glue Studio

Scegliendo l'editor di script shell Python per la creazione di un processo, puoi caricare uno script Python esistente o scriverne uno nuovo. Se scegli di scrivere un nuovo script, il codice boilerplate viene aggiunto al nuovo script del processo Python.

Per creare un nuovo processo shell Python

Fai riferimento alle istruzioni riportate in Avvio di processi in AWS Glue Studio.

Le proprietà del processo supportate per i processi shell Python non sono le stesse supportate per i processi Spark. Nell'elenco seguente vengono descritte le modifiche ai parametri di processo disponibili per i processi shell Python nella scheda Job details (Dettagli del processo).

- La proprietà Type (Tipo) per il processo viene automaticamente impostata su `Python Shell` e non può essere modificata.
- Invece di Language (Linguaggio), è presente la proprietà Python version (Versione di Python) per il processo. Al momento, i processi shell Python creati in AWS Glue Studio utilizzano Python 3.6.
- La proprietà Glue version (Versione Glue) non è disponibile, perché non applicabile ai processi shell Python.
- Invece di Worker type (Tipo di worker) e Number of workers (Numero di worker), è mostrata la proprietà Data processing units (Unità di elaborazione dati). Questa proprietà del processo determina quante unità di elaborazione dati (DPU) vengono utilizzate dalla shell Python durante l'esecuzione del processo.

- La proprietà Job bookmark (Segnalibro del processo) non è disponibile, perché non è supportata per i processi shell Python.
- Sotto Advanced properties (Proprietà avanzate), le seguenti proprietà non sono disponibili per i processi shell Python.
  - Parametri del processo
  - Registrazione continua
  - Spark UI (Interfaccia utente di Spark) e Spark UI logs path (Percorso dei log dell'interfaccia utente Spark)
  - Dependent jars path (Percorso file .jar dipendente), sotto la voce Libraries (Librerie).

Modifica dei nodi padre per un nodo nel diagramma del processo

Puoi modificare i nodi padre di un nodo per spostare i nodi all'interno del diagramma del processo o per modificare un'origine dati per un nodo.

Per modificare il nodo padre

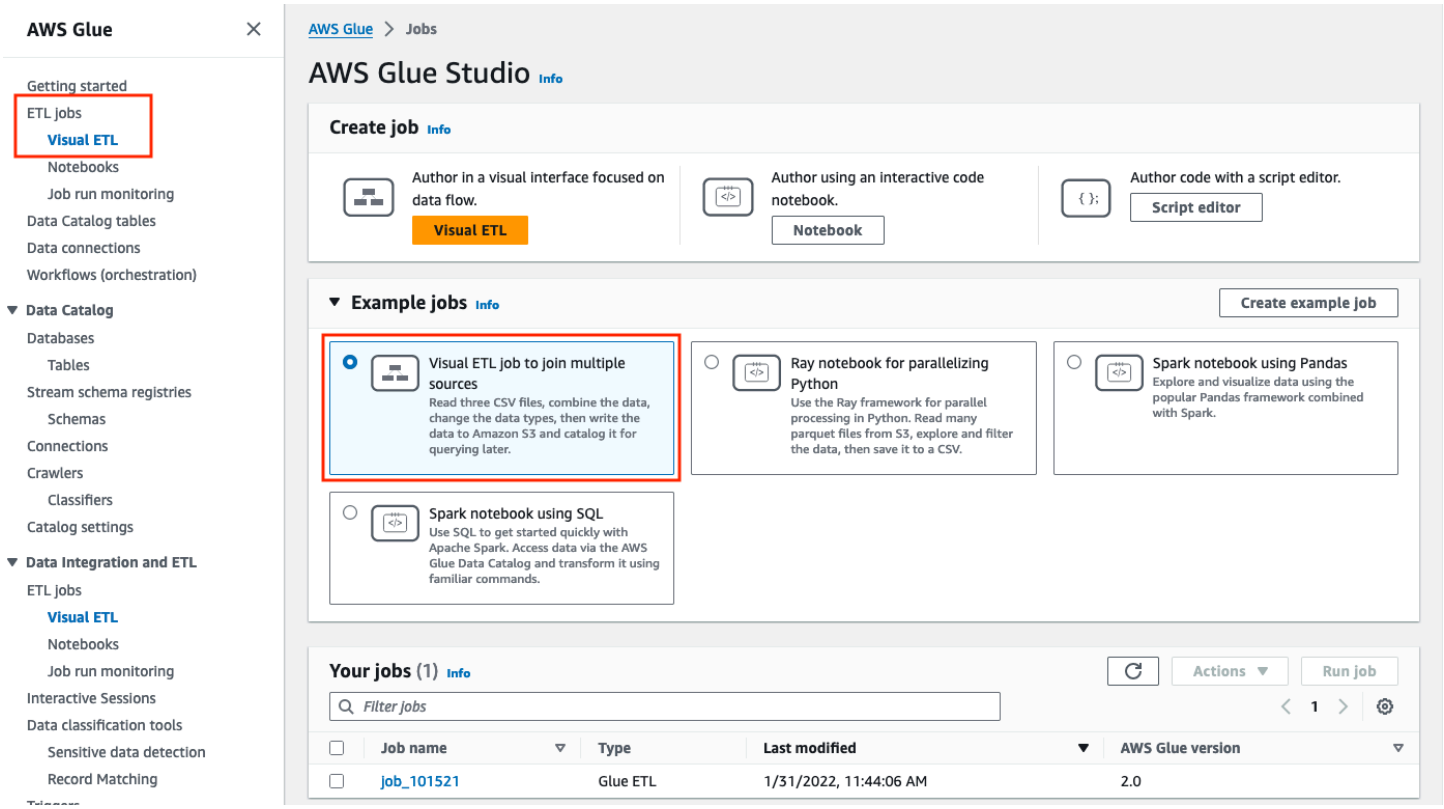
1. Scegli il nodo nel diagramma del processo da modificare.
2. Nel pannello dei dettagli del nodo, nella scheda Node properties (Proprietà del nodo), sotto l'intestazione Node parents (Nodi padre), rimuovi l'attuale padre per il nodo.
3. Scegli un nuovo nodo padre dall'elenco.
4. Modifica le altre proprietà del nodo in base alle esigenze in modo che corrispondano al nodo padre appena selezionato.

Se hai modificato un nodo per errore, puoi utilizzare il pulsante Undo (Annulla) nella barra degli strumenti per invertire l'operazione.

Eliminazione di nodi dal diagramma del processo

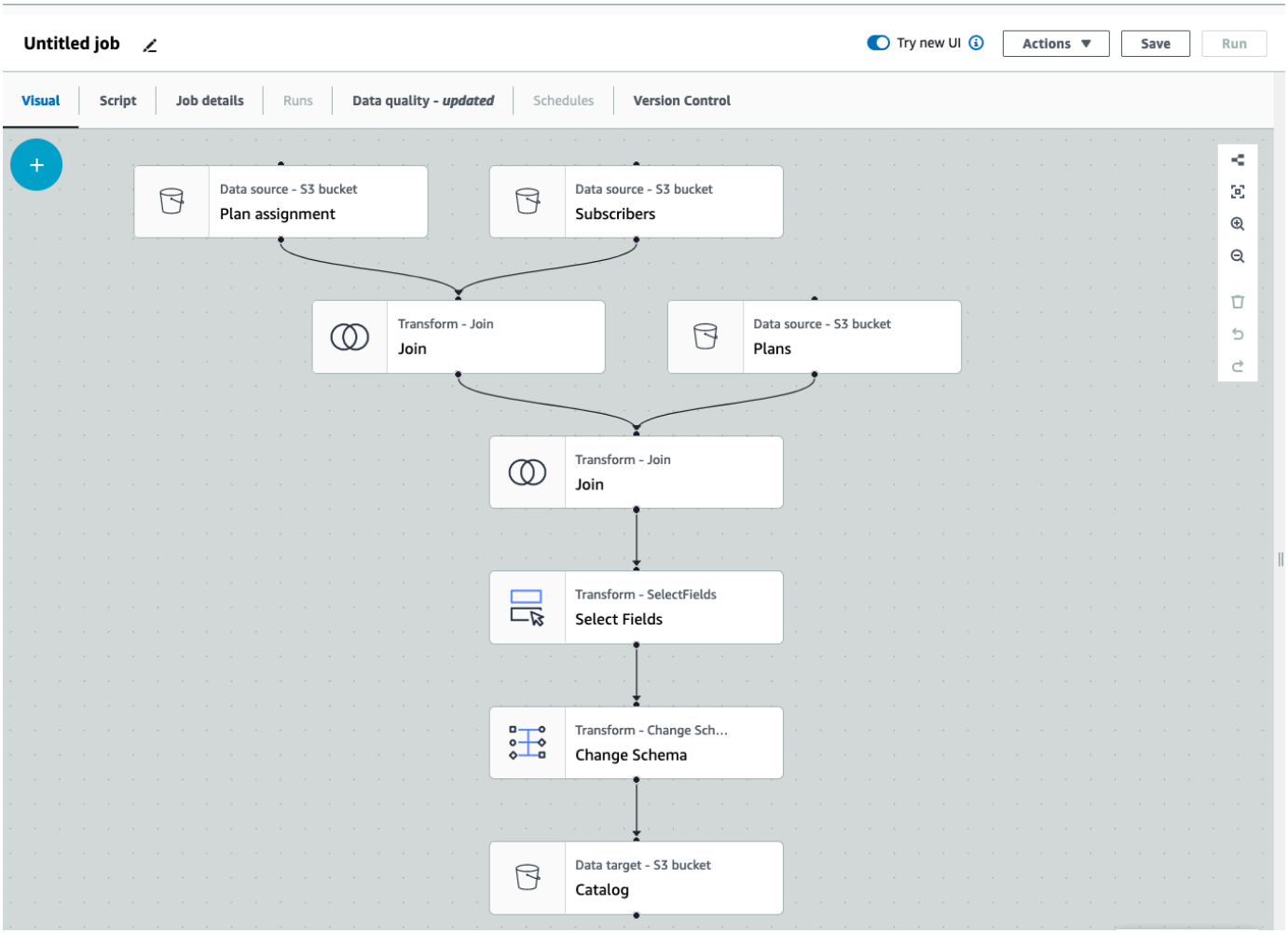
Quando si lavora con i job di Visual ETL, è possibile rimuovere i nodi dall'area di disegno senza dover aggiungere nuovamente o ristrutturare i nodi collegati al nodo rimosso.

Nell'esempio seguente, puoi seguire la procedura scegliendo ETL job > Visual ETL, quindi in Example jobs, scegliendo Visual ETL job per unire più fonti. Scegli Crea un lavoro di esempio per creare un lavoro e segui i passaggi seguenti.



Per rimuovere un nodo dall'area di disegno

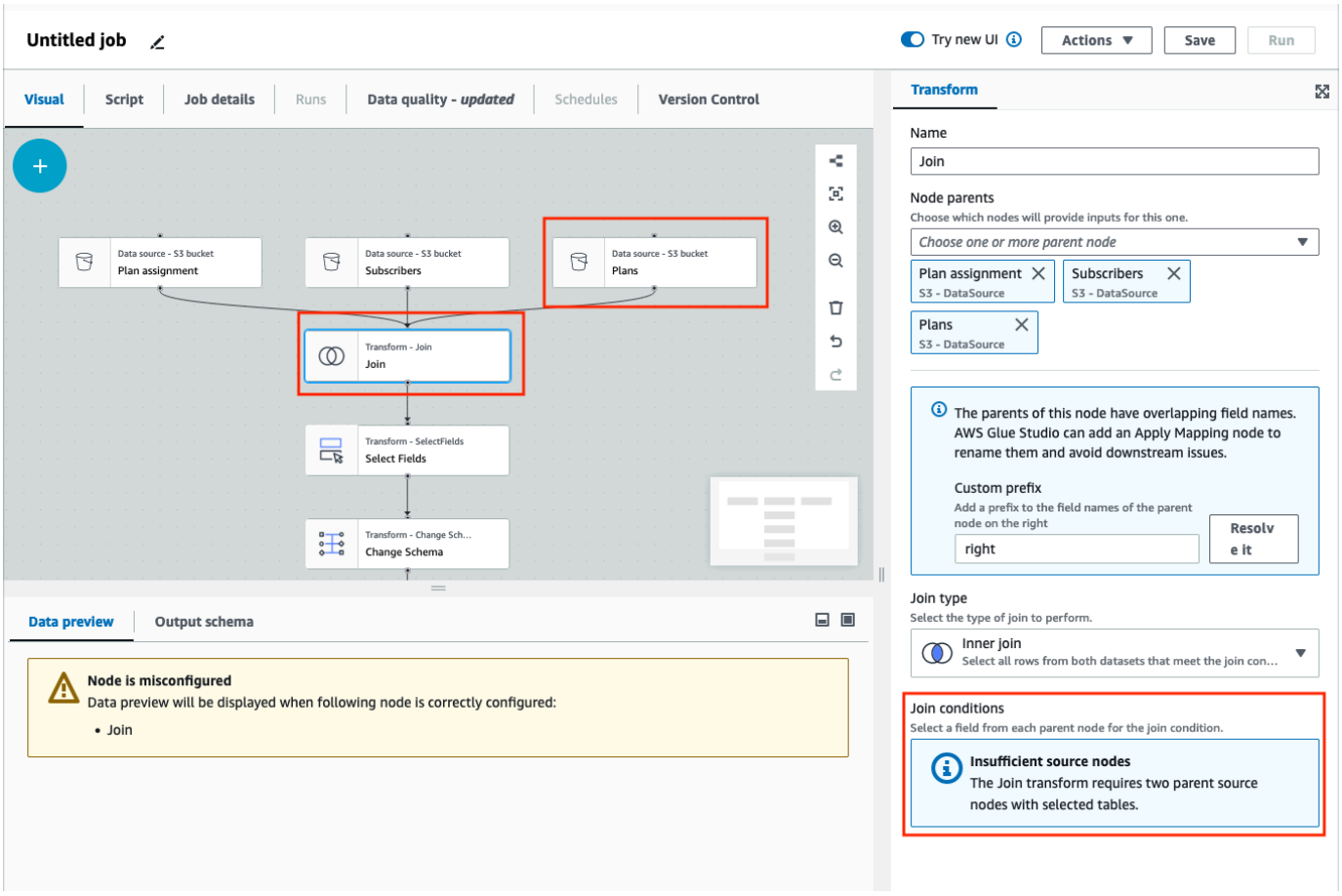
1. Dalla AWS Glue console, scegli Visual ETL dal menu di navigazione e scegli un lavoro esistente. L'area di disegno del lavoro mostra il lavoro di esempio, come illustrato di seguito.



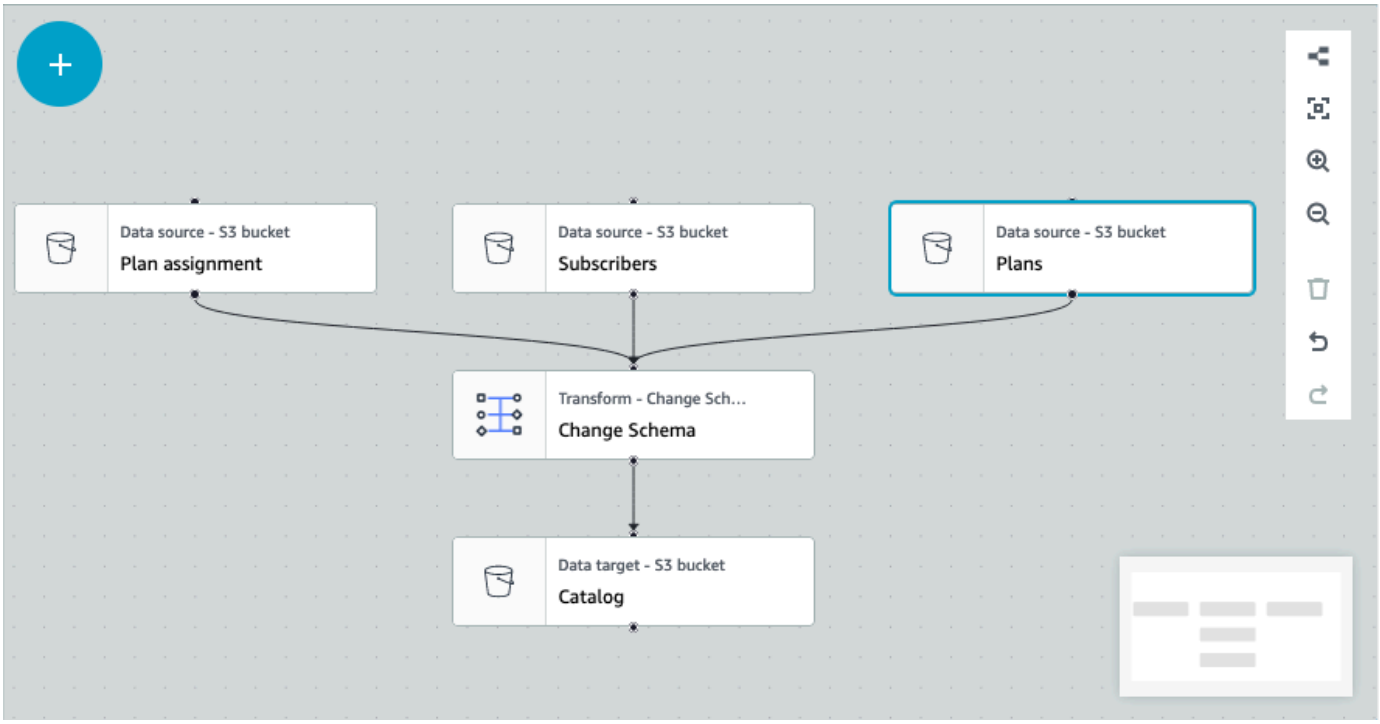
2. Scegli il nodo da rimuovere. La tela ingrandirà il nodo. Nella barra degli strumenti sul lato destro dell'area di disegno, scegli l'icona Cestino. Questo rimuoverà il nodo e qualsiasi nodo connesso al nodo verrà spostato per prendere il suo posto nel flusso di lavoro. In questo esempio, il primo nodo Join è stato eliminato dall'area di disegno.

   Se elimini un nodo dal flusso di lavoro, AWS Glue riorganizzerà i nodi in modo che siano organizzati in modo da non creare un flusso di lavoro non valido. Potrebbe comunque essere necessario correggere la configurazione di un nodo.

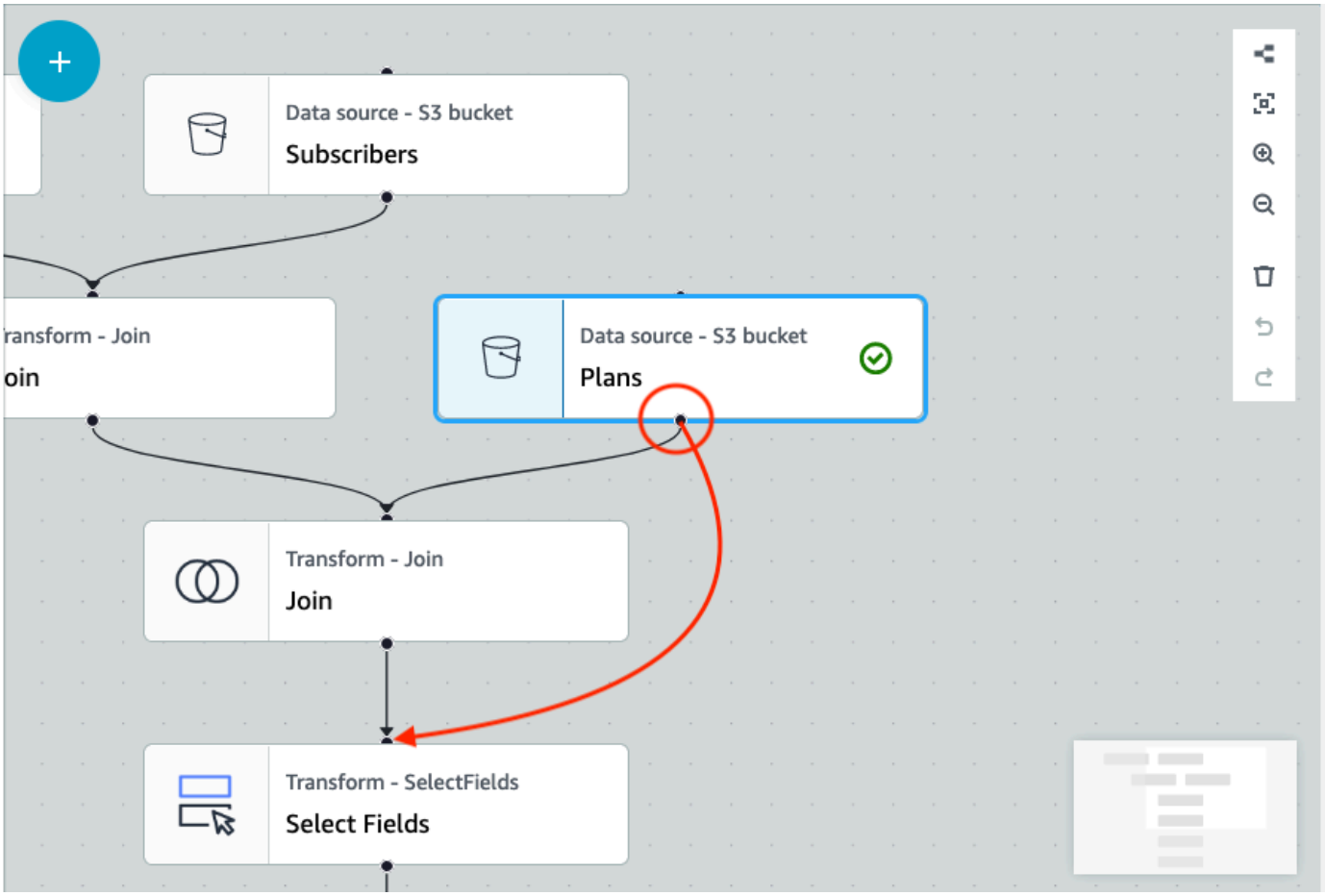
   Nell'esempio, il nodo Join sotto il nodo Subscribers è stato rimosso. Di conseguenza, il nodo sorgente Plans è stato spostato al livello superiore ed è ancora connesso al nodo Join secondario. Il nodo Join ora richiede una configurazione aggiuntiva poiché Join richiede due nodi di origine principali con tabelle selezionate. La scheda Trasforma a destra dell'area di disegno mostra il requisito mancante nelle condizioni Join.



3. Eliminare il secondo nodo Join e il nodo Select Fields. Una volta eliminati i nodi, il flusso di lavoro sarà simile all'esempio seguente.



4. Per modificare le connessioni dei nodi, fai clic sulla maniglia del nodo e trascina la connessione su un nuovo nodo. Ciò ti consentirà di eliminare i nodi e riorganizzarli in un flusso logico. Nell'esempio, viene effettuata una nuova connessione facendo clic sulla maniglia nel nodo Piani e trascinando la connessione al nodo Join, come illustrato dalla freccia rossa.



5. Se devi annullare un'azione, scegli l'icona Annulla direttamente sotto l'icona Cestino nella barra degli strumenti sul lato destro dell'area di disegno.

Aggiungendo parametri di origine e destinazione al nodo AWS Glue Data Catalog

AWS Glue Studio consente di parametrizzare i processi visivi. Poiché i nomi delle tabelle di catalogo nell'ambiente di produzione e sviluppo possono essere diversi, è possibile definire e selezionare i parametri di runtime per database e tabelle che verranno eseguiti durante l'esecuzione del processo.

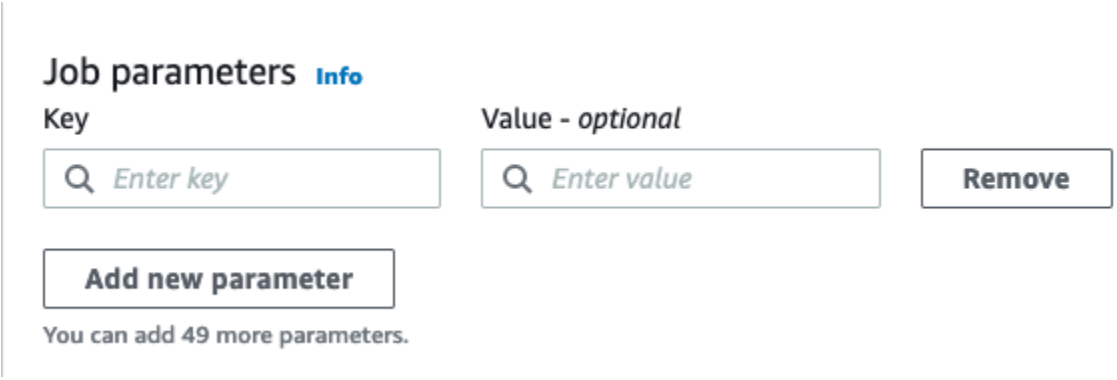
La parametrizzazione dei processi consente di parametrizzare origini e destinazioni e di salvare tali parametri sul processo quando usi il nodo AWS Glue Data Catalog. Quando si specificano origini e destinazioni come parametri, si abilita la riutilizzabilità dei processi, in particolare quando si utilizza lo stesso processo in più ambienti. Questo è utile quando si promuove il codice negli ambienti di implementazione, risparmiando tempo e fatica nella gestione delle origini e delle destinazioni. Inoltre,

i parametri personalizzati specificati sostituiranno qualsiasi argomento predefinito per esecuzioni specifiche di processi AWS Glue.

Aggiungere parametri di origine e destinazione

Sia che si stia usando il nodo AWS Glue Data Catalog come origine o destinazione, puoi definire i parametri di runtime nella sezione Proprietà avanzate nella scheda Dettagli del processo.

1. Scegli il nodo AWS Glue Data Catalog come nodo di origine o nodo di destinazione.
2. Seleziona la scheda Job details (Dettagli del processo).
3. Scegli Proprietà avanzate.
4. Nella sezione Parametri del processo, inserisci un valore chiave. Ad esempio,`--db.source` sarebbe il parametro per un database di origine. Puoi inserire qualsiasi nome come chiave, purché il nome della chiave sia seguito da “trattino trattino”.

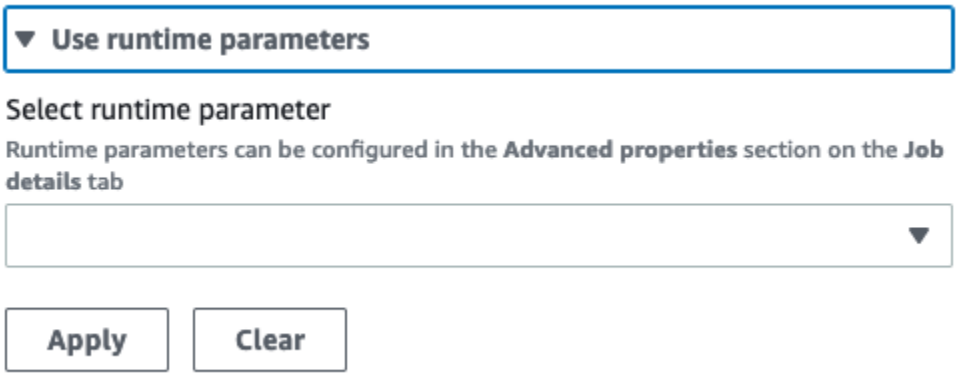


5. Inserire il valore. Ad esempio, `databasename` sarebbe il valore per la parametrizzazione del database.
6. Scegli Aggiunta nuovo parametro se si desidera aggiungere altri parametri. Sono consentiti fino a 50 parametri. Una volta definita la coppia di valori chiave, puoi utilizzare il parametro nel nodo AWS Glue Data Catalog.

Selezionare un parametro di runtime

> **Note**
>
> Il processo di selezione dei parametri di runtime per database e tabelle è lo stesso indipendentemente dal fatto che il nodo AWS Glue Data Catalog sia l'origine o la destinazione.

1. Scegli il nodo AWS Glue Data Catalog come nodo di origine o nodo di destinazione.
2. Nella scheda Proprietà dell'origine dei dati: Data Catalog , sotto Database, scegli Usa parametri di runtime.



3. Scegli un parametro dal menu a discesa. Ad esempio, quando si seleziona un parametro definito per un database di origine, il database verrà inserito automaticamente nel menu a discesa del database quando si sceglie Applicazione.
4. Nella sezione Tabella, scegli un parametro già definito come tabella di origine. Quando si sceglie Applicazione, la tabella viene inserita automaticamente come tabella da utilizzare.
5. Quando si salva e si esegue il processo, AWS Glue Studio farà riferimento ai parametri selezionati durante l'esecuzione del processo.

Utilizzo dei sistemi di controllo delle versioni Git in AWS Glue

> **Note**
>
> I notebook non sono attualmente supportati per il controllo della versione in AWS Glue Studio. Tuttavia, è supportato il controllo della versione per gli script del processo AWS Glue e i processi ETL visivi.

Se disponi di repository remoti e desideri gestire i tuoi AWS Glue lavori utilizzando i tuoi repository, puoi utilizzare AWS Glue Studio o AWS CLI per sincronizzare le modifiche ai tuoi repository e ai tuoi lavori in. AWS Glue Quando si sincronizzano le modifiche in questo modo, si sta inviando il processo da AWS Glue Studio al repository o lo si sta estraendo dal repository verso AWS Glue Studio.

Grazie all'integrazione di Git in AWS Glue Studio, è possibile:

- Integrazione con i sistemi di controllo delle versioni Git AWS CodeCommit, come, GitHub GitLab, e Bitbucket
- Modifica i processi AWS Glue in AWS Glue Studio indipendentemente dal fatto che utilizzi processi visivi o processi di script e li sincronizzi con un repository
- Parametrizza le origini e le destinazioni nei processi
- Estrai i processi da un repository e modificali in AWS Glue Studio
- Testa i processi estraendoli dai rami e/o inviandoli verso i rami utilizzando flussi di lavoro multiramo in AWS Glue Studio
- Scarica file da un repository e carica processi in AWS Glue Studio per la creazione di processi tra account
- Usa lo strumento di automazione che preferisci (ad esempio, Jenkins AWS CodeDeploy, ecc.)

Questo video dimostra come integrare AWS Glue con Git e creare una pipeline di codice continua e collaborativa.

Autorizzazioni IAM

Verifica che il processo abbia una delle seguenti autorizzazioni IAM. Per ulteriori informazioni su come configurare le autorizzazioni IAM, consulta Impostare le autorizzazioni IAM per AWS Glue Studio.

- `AWSGlueServiceRole`
- `AWSGlueConsoleFullAccess`

Come minimo, sono necessarie le seguenti operazioni per l'integrazione con Git:

- `glue:UpdateJobFromSourceControl` — essere in grado di aggiornare AWS Glue con un processo presente in un sistema di controllo delle versioni
- `glue:UpdateSourceControlFromJob` — essere in grado di aggiornare il sistema di controllo delle versioni con un processo archiviato in AWS Glue
- `s3:GetObject` — essere in grado di recuperare lo script per il processo mentre si invia al sistema di controllo delle versioni
- `s3:PutObject` — essere in grado di aggiornare lo script quando si estrae un processo da un sistema di controllo dell'origine

Prerequisiti

Per poter inviare i processi a un repository di controllo del codice sorgente, avrai bisogno di:

- un repository che è già stato creato dall'amministratore
- un ramo nel repository
- un token di accesso personale (per Bitbucket, questo è il Repository Access Token)
- il nome utente del proprietario del repository
- impostare le autorizzazioni nel repository per consentire ad AWS Glue Studio la lettura e la scrittura nel repository
  - GitLab— imposta gli ambiti dei token su api, read_repository e write_repository
  - Bitbucket: imposta le autorizzazioni su:
    - Iscrizione a Workspace: lettura e scrittura
    - Progetti: scrittura, lettura e amministratore
    - Repository: lettura, scrittura, eliminazione e amministratore

> **Note**
>
> Durante l'utilizzo AWS CodeCommit, non sono necessari il token di accesso personale e il proprietario del repository. Consulta Introduzione a Git e AWS CodeCommit.

Utilizzo dei processi dal tuo repository di controllo del codice sorgente in AWS Glue Studio

Per poter estrarre un processo dal repository per il controllo di origine che non è presente in AWS Glue Studio e utilizzare quel processo in AWS Glue Studio, i prerequisiti dipenderanno dal tipo di processo.

Per i processi visivi:

- sono necessari una cartella e un file JSON della definizione del processo che corrisponda al nome del processo

  Ad esempio, vedi la definizione del processo di seguito. Il ramo nel repository dovrebbe contenere un percorso `my-visual-job/my-visual-job.json` dove sia la cartella che il file JSON corrispondono al nome del processo

```
{
  "name" : "my-visual-job",
  "description" : "",
  "role" : "arn:aws:iam::aws_account_id:role/Rolename",
  "command" : {
    "name" : "glueetl",
    "scriptLocation" : "s3://foldername/scripts/my-visual-job.py",
    "pythonVersion" : "3"
  },
  "codeGenConfigurationNodes" : "{\"node-nodeID\":{\"S3CsvSource\":
{\"AdditionalOptions\":{\"EnableSamplePath\":false,\"SamplePath\":\"s3://notebook-
test-input/netflix_titles.csv\"},\"Escaper\":\"\",\"Exclusions\":[],\"Name\":\"Amazon
 S3\",\"OptimizePerformance\":false,\"OutputSchemas\":[{\"Columns\":[{\"Name\":
\"show_id\",\"Type\":\"string\"},{\"Name\":\"type\",\"Type\":\"string\"},{\"Name\":
\"title\",\"Type\":\"choice\"},{\"Name\":\"director\",\"Type\":\"string\"},{\"Name\":
\"cast\",\"Type\":\"string\"},{\"Name\":\"country\",\"Type\":\"string\"},{\"Name\":
\"date_added\",\"Type\":\"string\"},{\"Name\":\"release_year\",\"Type\":\"bigint\"},
{\"Name\":\"rating\",\"Type\":\"string\"},{\"Name\":\"duration\",\"Type\":\"string
\"},{\"Name\":\"listed_in\",\"Type\":\"string\"},{\"Name\":\"description\",\"Type
\":\"string\"}]}],\"Paths\":[\"s3://dalamgir-notebook-test-input/netflix_titles.csv
\"],\"QuoteChar\":\"quote\",\"Recurse\":true,\"Separator\":\"comma\",\"WithHeader
\":true}}}"
}
```

Per i processi di script:

- hai bisogno di una cartella, un file JSON con la definizione del processo e lo script
- la cartella e il file JSON devono corrispondere al nome del processo. Il nome dello script deve corrispondere alla `scriptLocation` nella definizione del processo insieme all'estensione del file

  Ad esempio, nella definizione del processo riportata di seguito, il ramo nel repository deve contenere un percorso `my-script-job/my-script-job.json` e `my-script-job/my-script-job.py`. Il nome dello script deve corrispondere al nome nella `scriptLocation` inclusa l'estensione dello script

```
{
  "name" : "my-script-job",
  "description" : "",
  "role" : "arn:aws:iam::aws_account_id:role/Rolename",
```

```
  "command" : {
    "name" : "glueetl",
    "scriptLocation" : "s3://foldername/scripts/my-script-job.py",
    "pythonVersion" : "3"
  }
}
```

Limitazioni

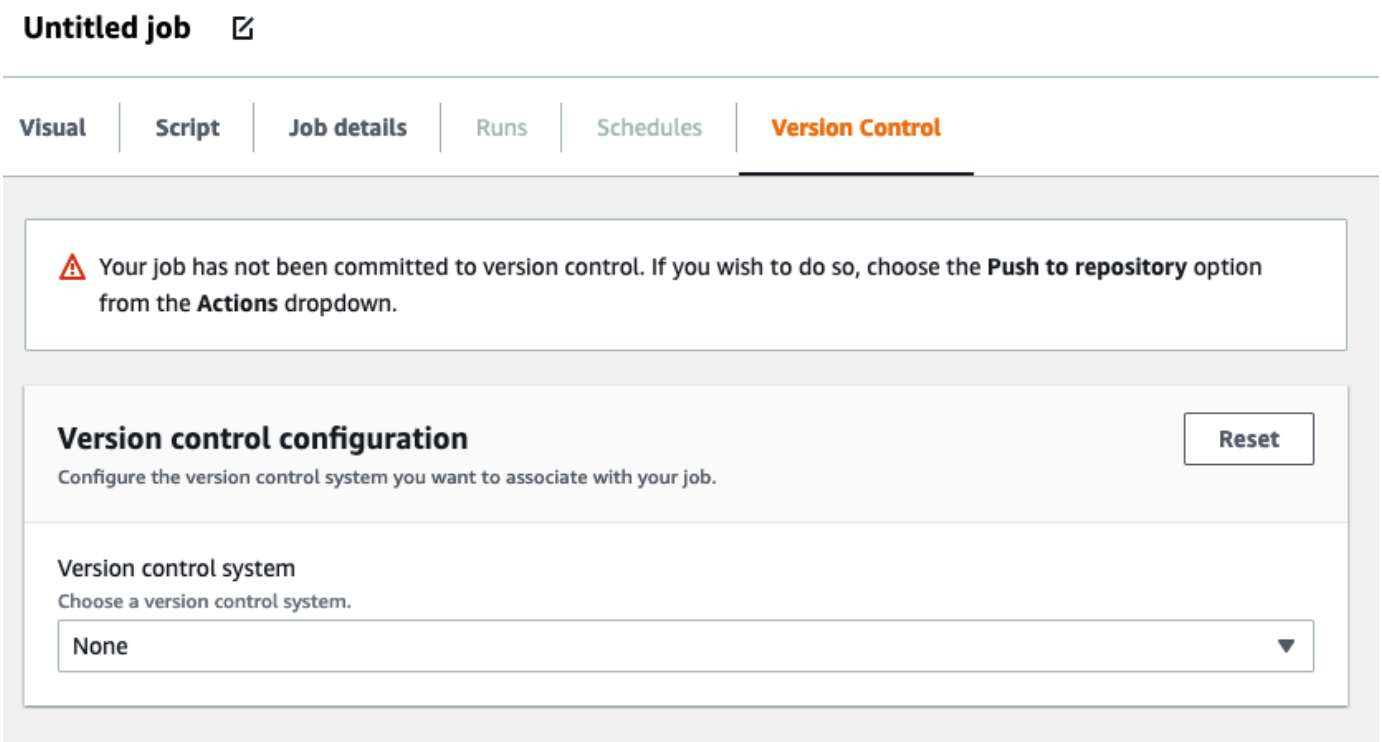
- AWS Glueattualmente non supporta la pressione o l'estrazione da -Groups. GitLab

Collegamento dei repository di controllo delle versioni a AWS Glue

È possibile inserire i dettagli del repository di controllo delle versioni e gestirli nella scheda Controllo delle versioni nell'editor dei processi AWS Glue Studio. Per integrarlo con il repository Git, bisogna connettersi al repository ogni volta che si accede a AWS Glue Studio.

Per connettere un sistema di controllo della versione Git:

1. In AWS Glue Studio, inizia un nuovo processo e scegli la scheda Controllo delle versioni.



2. In Sistema di controllo delle versioni, scegli Git Service tra le opzioni disponibili facendo clic sul menu a discesa.

   - AWS CodeCommit
   - GitHub
   - GitLab
   - Bitbucket

3. A seconda del sistema di controllo delle versioni Git scelto, si avranno diversi campi da completare.

   Per AWS CodeCommit:

   Completa la configurazione del repository selezionando il repository e il ramo per il processo:

   - Repository: se hai configurato dei repository in AWS CodeCommit, seleziona il repository dal menu a discesa. I repository verranno inseriti automaticamente nell'elenco
   - Ramo — seleziona il ramo dal menu a discesa
   - Cartella — facoltativo: inserisci il nome della cartella in cui salvare il processo. Se lasciato vuoto, viene creata automaticamente una cartella. Il nome predefinito della cartella è il nome del processo

   Per: GitHub

   Completa la GitHub configurazione completando i campi:

   - Token di accesso personale: è il token fornito dal GitHub repository. Per ulteriori informazioni sui token di accesso personali, consulta Docs GitHub
   - Proprietario del repository: è il proprietario del repository. GitHub

   Completa la configurazione del repository selezionando il repository e il ramo da. GitHub

   - Repository: se hai configurato dei repository in GitHub, seleziona il repository dal menu a discesa. I repository verranno inseriti automaticamente nell'elenco
   - Ramo — seleziona il ramo dal menu a discesa

- Cartella — facoltativo: inserisci il nome della cartella in cui salvare il processo. Se lasciato vuoto, viene creata automaticamente una cartella. Il nome predefinito della cartella è il nome del processo

Per: GitLab

> **Note**
>
> AWS Glueattualmente non supporta la spinta/estrazione da -Groups. GitLab

- Token di accesso personale: questo è il token fornito dal repository. GitLab Per ulteriori informazioni sui token di accesso personali, consulta Token di accesso GitLab personali
- Proprietario del repository: si tratta del proprietario del repository. GitLab

Completa la configurazione del repository selezionando il repository e il ramo da. GitLab

- Repository: se hai configurato dei repository in GitLab, seleziona il repository dal menu a discesa. I repository verranno inseriti automaticamente nell'elenco
- Ramo — seleziona il ramo dal menu a discesa
- Cartella — facoltativo: inserisci il nome della cartella in cui salvare il processo. Se lasciato vuoto, viene creata automaticamente una cartella. Il nome predefinito della cartella è il nome del processo

Per Bitbucket:

- Password dell'app: Bitbucket utilizza le password delle app e non i token di accesso al repository. Per ulteriori informazioni sulle password delle app, consulta Password delle app.
- Proprietario del repository: questo è il proprietario del repository Bitbucket. In Bitbucket, il proprietario è il creatore del repository.

Completa la configurazione del repository selezionando il workspace, il repository, il ramo e la cartella da Bitbucket.

- Workspace: se sono stati configurati i workspace in Bitbucket, selezionare il workspace dal menu a discesa. I workspace vengono popolati automaticamente
- Repository: se sono stati configurati i repository in Bitbucket, selezionare il repository dal menu a discesa. I repository vengono popolati automaticamente
- Ramo: seleziona il ramo dal menu a discesa. I tuoi rami vengono popolati automaticamente
- Cartella — facoltativo: inserisci il nome della cartella in cui salvare il processo. Se lasciato vuoto, viene creata automaticamente una cartella con il nome del processo.

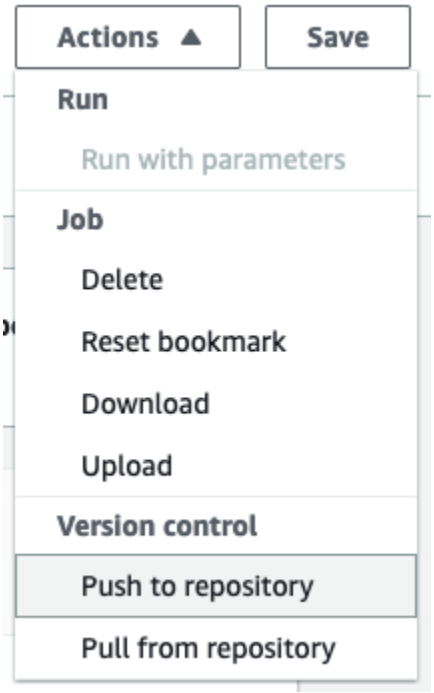
4. Scegli Save (Salva) nella parte superiore del processo AWS Glue Studio

Invio dei processi AWS Glue al repository di origine

Dopo aver inserito i dettagli del sistema di controllo delle versioni, puoi modificare i processi in AWS Glue Studio e inviarli al repository di origine. Se non conosci i concetti di Git come inviare ed estrarre, guarda questo tutorial Nozioni di base su Git e AWS CodeCommit.

Per inviare il processo a un repository, devi inserire i dettagli del sistema di controllo delle versioni e salvare il processo.

1. Nel processoAWS Glue Studio, scegli Operazioni. Questo aprirà opzioni di menu aggiuntive.



2. Scegli Invia a repository.

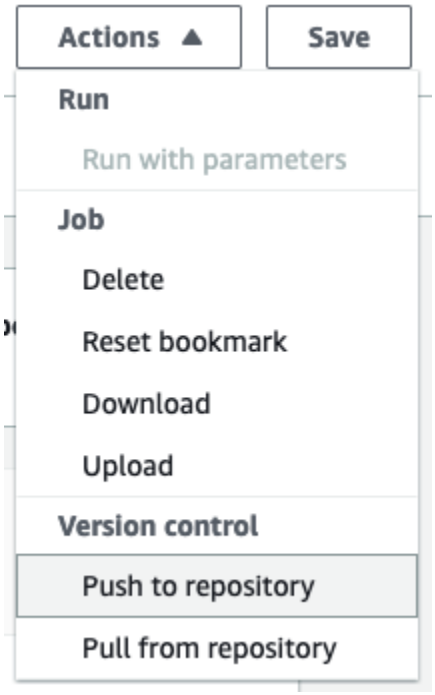
Questa operazione salverà il processo. Quando si esegue l'invio al repository,AWS Glue Studio invia l'ultima modifica salvata. Se il processo nel repository è stato modificato da te o da un altro utente e non è sincronizzato con il processo in AWS Glue Studio, quando si invia il processo da AWS Glue Studio, il processo nel repository viene sovrascritto con il processo salvato in AWS Glue Studio.

3. Scegli Conferma per completare l'operazione. Questo crea un nuovo commit nel repository. Se lo stai utilizzando AWS CodeCommit, un messaggio di conferma mostrerà un link all'ultimo commit on. AWS CodeCommit

Estrarre processi AWS Glue dal repository di origine

Dopo aver inserito i dettagli del repository Git nella scheda Controllo delle versioni, puoi anche estrarre i processi dal repository e modificarli in AWS Glue Studio.

1. Nel processo AWS Glue Studio, scegli Operazioni. Questo aprirà opzioni di menu aggiuntive.



2. Scegli Estrai dal repository.
3. Scegli Conferma. Questo prende il commit più recente dal repository e aggiorna il processo in AWS Glue Studio.
4. Modificare il processo in AWS Glue Studio. Se apporti delle modifiche, puoi sincronizzare il processo con il repository scegliendo Invia a repository dal menu a discesa Operazioni.

Creazione di codice con notebook AWS Glue Studio

I data engineer possono creare processi AWS Glue più velocemente e più facilmente rispetto a prima utilizzando l'interfaccia interattiva per notebook in AWS Glue Studio o le sessioni interattive in AWS Glue.

Argomenti

Panoramica sull'utilizzo dei notebook

AWS Glue Studio permette di creare processi in modo interattivo in un'interfaccia notebook basata su Jupyter Notebooks. Attraverso i notebook in AWS Glue Studio, è possibile modificare gli script di processo e visualizzare l'output senza dover eseguire un processo completo e modificare il codice di integrazione dei dati e visualizzare l'output senza dover eseguire un processo completo. Inoltre, è possibile aggiungere markdown e salvare i notebook come file con estensione .ipynb e script di processo. È possibile avviare un notebook senza installare software localmente o gestire server. Quando hai terminato di lavorare con il codice, AWS Glue Studio può convertire il notebook in un processo Glue con un semplice clic.

Alcuni dei vantaggi derivanti dall'utilizzo dei notebook sono:

- Nessun cluster di cui effettuare il provisioning o da gestire
- Nessun cluster inattivo da pagare
- Nessuna configurazione iniziale richiesta
- Non è richiesta l'installazione di notebook Jupyter
- Lo stesso tempo di esecuzione/piattaforma di AWS Glue ETL

All'avvio di un notebook tramite AWS Glue Studio, tutti i passaggi di configurazione vengono eseguiti per te, in modo che tu possa esplorare i dati e iniziare a sviluppare lo script del processo dopo pochi

secondi. AWS Glue Studio configura un notebook Jupyter con il kernel AWS Glue Jupyter. Per utilizzare questo notebook, non è necessario configurare VPC, connessioni di rete o endpoint di sviluppo.

Per creare processi utilizzando l'interfaccia notebook:

- configura le autorizzazioni IAM necessarie
- avvia una sessione notebook per creare un processo
- scrivi codice nelle celle del notebook
- esegui e testa il codice per visualizzare l'output
- salva il processo

Dopo aver salvato il notebook, questo sarà un processo AWS Glue completo. È possibile gestire tutti gli aspetti del processo, ad esempio la pianificazione delle esecuzioni e l'impostazione dei parametri del processo e la visualizzazione della cronologia dell'esecuzione del processo direttamente accanto al notebook.

Creazione di un processo ETL utilizzando i notebook in AWS Glue Studio

Per iniziare a utilizzare i notebook nella console AWS Glue Studio

1. Allega le policy AWS Identity and Access Management all'utente AWS Glue Studio e crea un ruolo IAM per il processo e il notebook ETL.
2. Configura la sicurezza IAM aggiuntiva per notebook, come descritto in Concessione di autorizzazioni per il ruolo IAM.
3. Accedi alla console AWS Glue Studio all'indirizzo https://console.aws.amazon.com/gluestudio/.

> **Note**
>
> Verifica che il tuo browser non blocchi i cookie di terzi. Qualsiasi browser che blocca i cookie di terze parti per impostazione predefinita o abilitata dall'utente impedirà l'avvio di notebook. Per ulteriori informazioni sulla gestione dei cookie, consulta:

- Chrome
- Firefox

- Safari

4. Scegli il link Jobs (Processi) nel menu di navigazione a sinistra.
5. Scegli Notebook Jupyter e quindi Create (Crea) per avviare una nuova sessione del notebook.
6. Nella pagina Create job in Jupyter notebook (Crea processo nel notebook Jupyter), specifica il nome del processo, il ruolo IAM da utilizzare. Scegli Create job (Crea processo).

   Dopo un breve periodo di tempo, viene visualizzato l'editor del notebook.
7. Dopo aver aggiunto il codice, è necessario eseguire la cella per avviare una sessione. Esistono diversi modi per eseguire la cella:

   - Premi il pulsante play.
   - Utilizza la scelta rapida da tastiera:
     - Su MacOS, Command + Invio per eseguire la cella.
     - Su Windows, Maius + Invio per eseguire la cella.

   Per informazioni sulla scrittura di codice utilizzando un'interfaccia per notebook Jupyter, vedi la Documentazione utente di Jupyter Notebook .
8. Per testare lo script, esegui l'intero script o le singole celle. Qualsiasi output di comando verrà visualizzato nell'area sotto la cella.
9. Dopo aver completato lo sviluppo del notebook, è possibile salvare il processo e quindi eseguirlo. Lo script è disponibile nella tabella Script. Tutte i magic aggiunti al taccuino verranno rimossi e non verranno salvati come parte dello script del processo AWS Glue. AWS Glue Studio aggiungerà automaticamente un `job.commit()` fino alla fine dello script generato dal contenuto di notebook.

   Per informazioni su come creare i processi, consulta Avviare un'esecuzione del processo.

Componenti per l'editor del notebook

L'interfaccia dell'editor del notebook ha le seguenti sezioni principali.

- Interfaccia notebook (pannello principale) e barra degli strumenti
- Schede di modifica dei processi

L'editor del notebook

L'editor del notebook AWS Glue Studio si basa sull'applicazione Jupyter Notebook. L'interfaccia del notebook AWS Glue Studio è simile a quella fornita da Juypter Notebook, descritta nella sezione Interfaccia utente per notebook. Il notebook utilizzato dalle sessioni interattive è un Jupyter Notebook.

Anche se il notebook AWS Glue Studio è simile a Juptyer Notebook, si differenzia in alcuni modi chiave:

- attualmente, il notebook AWS Glue Studio non può installare le estensioni
- non è possibile utilizzare più schede; esiste una relazione 1:1 tra un processo e un notebook
- il notebook AWS Glue Studio non ha lo stesso menu di file superiore presente in Jupyter Notebook
- attualmente, il notebook AWS Glue Studio funziona solo con il kernel AWS Glue Nota che non è possibile aggiornare il kernel da solo.

schede di modifica dei processi AWS Glue Studio

Le schede utilizzate per interagire con il processo ETL si trovano nella parte superiore della pagina del notebook. Sono simili alle schede che appaiono nell'editor dei processi visivi di AWS Glue Studio ed eseguono le stesse operazioni.

- Notebook: utilizza questa scheda per visualizzare lo script del processo utilizzando l'interfaccia del notebook.
- Job details (Dettagli processo): configura l'ambiente e le proprietà per l'esecuzione del processo.
- Runs (Esecuzioni): visualizza le informazioni sulle precedenti esecuzioni di questo processo.
- Schedules (Piani): configura una pianificazione per l'esecuzione del processo in momenti specifici.

Salvataggio del notebook e dello script del processo

È possibile salvare il notebook e lo script del processo che si sta creando in qualsiasi momento. Basta scegliere il pulsante Save (Salva) nell'angolo in alto a destra, come se stessi utilizzando l'editor visivo o di script.

Quando scegli Save (Salva), il file del notebook viene salvato nelle posizioni predefinite:

- Per impostazione predefinita, lo script del processo viene salvato nella posizione Amazon S3 indicata nella posizione Amazon S3 indicata nella scheda Job Details (Dettagli del processo),

in Advanced properties (Proprietà avanzate), in Script path (Percorso script) della proprietà dei dettagli del processo. Gli script del processo vengono salvati in una sottocartella denominata `Scripts`.

- Per impostazione predefinita, il file del notebook (`.ipynb`) viene salvato nella posizione Amazon S3 indicata nella scheda Job Details (Dettagli del processo), in Advanced properties (Proprietà avanzate) in Script path (Percorso script) dei dettagli del processo. I file del notebook vengono salvati in una sottocartella denominata `Notebooks`.

> **Note**
>
> Quando salvi il processo, lo script contiene solo le celle di codice del notebook. Le celle Markdown e i magic non sono inclusi nello script del processo. Tuttavia, il file `.ipynb` conterrà qualsiasi markdown e magic.

Dopo aver salvato il processo, è possibile eseguirlo utilizzando lo script creato nel notebook.

Gestione delle sessioni di notebook

I notebook in AWS Glue Studio si basano sulla caratteristica di sessioni interattive di AWS Glue. Per l'utilizzo delle sessioni interattive, è previsto un costo. Per gestire i costi, puoi monitorare le sessioni create per il tuo account e configurare le impostazioni di default per tutte le sessioni.

Modifica del timeout di default per tutte le sessioni del notebook

Per impostazione predefinita, se il notebook AWS Glue Studio sottoposto a provisioning è stato avviato e non è stata eseguita alcuna cella, scadrà dopo 12 ore. Tale timeout non è configurabile e non comporta alcun costo aggiuntivo.

Dopo aver eseguito una cella, verrà avviata una sessione interattiva che scade dopo 48 ore. Questo timeout può essere configurato passando un magic `%idle_timeout` prima di eseguire una cella.

Come modificare il timeout di sessione predefinito per i notebook in AWS Glue Studio

1. Nel notebook, inserisci il magic `%idle_timeout` in una cella e specifica il valore di timeout in minuti.
2. Ad esempio: `%idle_timeout 15` cambierà il timeout di default a 15 minuti. Se la sessione non viene utilizzata entro 15 minuti, viene automaticamente interrotta.

Installazione di moduli Python aggiuntivi

Se desideri installare moduli aggiuntivi per la tua sessione usando pip, puoi farlo utilizzando `%additional_python_modules` per aggiungerli alla sessione:

```
%additional_python_modules awswrangler, s3://mybucket/mymodule.whl
```

Tutti gli argomenti di additional_python_modules vengono passati a `pip3 install -m <>`

Per visualizzare un elenco di moduli Python disponibili, consulta la pagina Using Python libraries with AWS Glue.

Modifica della configurazione di AWS Glue

Puoi usare i magic per controllare i valori di configurazione del processo AWS Glue. Se si desidera modificare un valore di configurazione del processo, è necessario utilizzare i magic corretti nel notebook. Consulta la pagina Magics supported by AWS Glue interactive sessions for Jupyter.

> **Note**
>
> La sovrascrittura delle proprietà per una sessione in esecuzione non è più disponibile. Per modificare le configurazioni della sessione, interrompere la sessione, impostare le nuove configurazioni e quindi iniziare una nuova sessione.

AWS Glue supporta diversi tipi di dipendenti. È possibile impostare il tipo di dipendente con `%worker_type`. Ad esempio: `%worker_type G.2X` . Il valore di default è G.1X.

Inoltre, è possibile specificare il numero di dipendenti con `%number_of_workers`. Ad esempio, per specificare 40 dipendenti: `%number_of_workers 40`.

Per ulteriori informazioni, consulta Definizione delle proprietà del processo

Arresto di una sessione di notebook

Per interrompere una sessione del notebook, usa il magic `%stop_session`.

Se ti allontani dal notebook nella console AWS, visualizzerai un messaggio di avviso in cui è possibile scegliere di interrompere la sessione.

Utilizzo CodeWhisperer con AWS Glue Studio notebooks

AWS Glue Studio permette di creare processi in modo interattivo in un'interfaccia notebook basata su Jupyter Notebooks. L'utilizzo CodeWhisperer migliora l'esperienza di creazione all'interno dei notebook. AWS Glue Studio

L' CodeWhisperer estensione Amazon supporta la scrittura di codice generando consigli sul codice e suggerendo miglioramenti relativi ai problemi relativi al codice.

Che cos'è Amazon CodeWhisperer?

Amazon CodeWhisperer è un servizio basato sull'apprendimento automatico che aiuta a migliorare la produttività degli sviluppatori. CodeWhisperer raggiunge questo obiettivo generando consigli sul codice basati sui commenti degli sviluppatori in linguaggio naturale e sul loro codice nell'IDE. Durante l'anteprima, Amazon CodeWhisperer è disponibile per i linguaggi Java JavaScript, Python, C# e TypeScript di programmazione. Il servizio si integra con Amazon SageMaker Studio JupyterLab, istanze di Amazon SageMaker notebook e altri ambienti di sviluppo integrati (IDE).

Per ulteriori informazioni, consulta la sezione Configurazione con. CodeWhisperer AWS Glue Studio

Stati di esecuzione dei processi AWS Glue sulla console

Puoi visualizzare lo stato di un processo di estrazione, trasformazione e caricamento (ETL) AWS Glue mentre è in esecuzione o una volta arrestato. Puoi visualizzare lo stato tramite console AWS Glue. Per ulteriori informazioni sullo stato dei processi di esecuzione, consulta the section called “Stati di esecuzione dei processi”.

Accesso al pannello di controllo di monitoraggio dei processi

Per accedere al pannello di controllo di monitoraggio dei processi, scegli il collegamento Monitoring (Monitoraggio) nel pannello di navigazione di AWS Glue.

Panoramica del pannello di controllo di monitoraggio dei processi

Il pannello di controllo di monitoraggio dei processi fornisce un riepilogo generale delle esecuzioni del processo, con i totali per i processi con lo stato di Running (In esecuzione), Canceled (Annullato), Success (Riuscito) oppure Failed (Non riuscito). I riquadri aggiuntivi forniscono il tasso di successo complessivo dell'esecuzione del processo, l'utilizzo stimato della DPU per i processi,

una suddivisione dei conteggi dello stato del processo per tipo di processo, per tipo di worker e per giorno.

I grafici nei riquadri sono interattivi. È possibile scegliere qualsiasi blocco in un grafico per eseguire un filtro che visualizzi solo quei processi nella tabella Job runs (Esecuzioni del processo) nella parte inferiore della pagina.

Per modificare l'intervallo di date delle informazioni visualizzate in questa pagina, utilizza il selettore Date range (Intervallo date). Quando si modifica l'intervallo di date, i riquadri delle informazioni vengono adattati per visualizzare i valori per il numero di giorni specificato prima della data corrente. Puoi anche usare un intervallo di date specifico scegliendo Custom (Personalizzato) dal selettore dell'intervallo di date.

Visualizzazione esecuzioni dei processi

L'elenco delle risorse Job runs (Esecuzioni dei processi) mostra i processi per l'intervallo di date specificato e i filtri.

È possibile filtrare i processi in base a criteri aggiuntivi, ad esempio lo stato, il tipo di worker, il tipo di processo e il nome del processo. Nella casella filtro nella parte superiore della tabella è possibile inserire il testo da utilizzare come filtro. Durante l'inserimento del testo, i risultati della tabella vengono aggiornati con righe contenenti testo corrispondente.

È possibile visualizzare un sottoinsieme dei processi scegliendo gli elementi dai grafici nel pannello di controllo di monitoraggio del processo. Ad esempio, se si sceglie il numero di processi in esecuzione nella finestra Job runs summary (Riepilogo delle esecuzioni), l'elenco Job runs (Esecuzioni dei processi) visualizza solo i processi che hanno attualmente lo stato `Running`. Se si sceglie una delle barre nel grafico a barre Worker type breakdown (Analisi del tipo di worker), nell'elenco Job runs (Esecuzioni dei processi) vengono mostrate solo le esecuzioni del processo con il tipo e lo stato corrispondenti.

L'elenco delle risorse Job runs (Esecuzioni dei processi) mostra i dettagli delle esecuzioni del processo. È possibile ordinare le righe nella tabella scegliendo un'intestazione di colonna. La tabella contiene le informazioni seguenti:

Proprietà	Description
Nome processo	Il nome del processo .

Proprietà	Description
Type	Il tipo di ambiente per il processo: • ETL Glue: esegue in un ambiente Apache Spark gestito da AWS Glue. • Streaming Glue: esegue in un ambiente Apache Spark ed esegue ETL sui flussi di dati. • Shell Python: esegue gli script di Python come una shell.
Ora di inizio	La data e ora in cui questa esecuzione di processo è stata avviata.
Ora di fine	La data e ora in cui questa elaborazione di processo è stata completata.
Stato di esecuzione	Lo stato attuale del processo eseguito. I valori possono essere: • `STARTING` • `RUNNING` • `STOPPING` • `STOPPED` • `SUCCEEDED` • `FAILED` • `TIMEOUT`
Tempo di esecuzione	Quantità di tempo durante la quale l'esecuzione dell'attività ha utilizzato le risorse.

Proprietà	Description
Capacità	Il numero di unità di elaborazione dati (DPU) di AWS Glue allocate per questa esecuzione. Per ulteriori informazioni sulla pianificazione della capacità, consulta Monitoraggio per la pianificazione della capacità DPU nella Guida per gli sviluppatori di AWS Glue.

Proprietà	Description
Tipo di worker	Il tipo di worker predefinito allocato quando è stato eseguito il processo. I valori possono essere `G.1X`, `G.2X`, `G.4X` o `G.8X`. • **`G.1X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria) con disco da 84 GB (circa 34 GB liberi). Consigliamo questo tipo di worker per i processi ad alto consumo di memoria. Questa è l'impostazione predefini ta per Worker type (Tipo di worker) per la versione AWS Glue 2.0 o successive. • **`G.2X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria) con disco da 128 GB (circa 77 GB liberi). Suggeriamo questo tipo di dipendente per i processi ad alto consumo di memoria e per i processi che eseguono trasformazioni machine learning. • **`G.4X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 4 DPU (16 vCPU, 64 GB di memoria) con disco da 256 GB (circa 235 GB liberi). Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i processi ETL di AWS Glue Spark

Proprietà	Description
	versione 3.0 o successiva nelle seguenti Regioni AWS: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda) ed Europa (Stoccolma). • **`G.8X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 8 DPU (32 vCPU, 128 GB di memoria) con disco da 512 GB (circa 487 GB liberi). Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i processi ETL di AWS Glue Spark versione 3.0 o successiva, nelle stesse Regioni AWS supportate per il tipo di worker `G.4X`.
Ore DPU	Numero stimato di DPU utilizzate per l'esecuzione del processo. Una DPU è una misura relativa della potenza di elaborazione. Le DPU vengono utilizzate per determinare il costo dell'esecuzione del processo. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

È possibile scegliere qualsiasi processo eseguito nell'elenco e visualizzare informazioni aggiuntive. Scegli un'esecuzione del processo, quindi esegui una delle operazioni seguenti:

- Scegli il menu Actions (Operazioni) e l'opzione View job (Visualizza processo) per visualizzare il processo nell'editor visivo.
- Scegli il menu Actions (Operazioni) e l'opzione Stop run (Interrompi esecuzione) per interrompere l'esecuzione corrente del processo.
- Scegli il pulsante View CloudWatch logs (Visualizza log CloudWatch) per visualizzare i log di esecuzione del processo.
- Scegli Visualizza dettagli per visualizzare la pagina dei dettagli dell'esecuzione.

Visualizzazione dei log di esecuzione del processo

Puoi visualizzare i log del processo in diversi modi:

- Nella pagina Monitoring (Monitoraggio), nella tabella Job runs (Esecuzioni dei processi), scegli un'esecuzione, quindi scegli View CloudWatch logs (Visualizza log CloudWatch).
- Nell'editor visivo dei processi, nella scheda Runs (Esecuzioni) per un processo, scegli i collegamenti ipertestuali per visualizzare i log:
  - Log: collega ai log dei processi di Apache Spark scritti quando la registrazione continua è abilitata per l'esecuzione di un processo. Quando si sceglie questo collegamento, si accede ai log Amazon CloudWatch nel gruppo di log `/aws-glue/jobs/logs-v2`. Per impostazione predefinita, i log escludono i messaggi di log di heartbeat inutili di driver o executor Apache Spark e Apache Hadoop YARN. Per ulteriori informazioni sulla registrazione continua, consulta Registrazione continua per processi di AWS Glue nella Guida per gli sviluppatori di AWS Glue.
  - Log di errore: collega ai log scritti in `stderr` per questa esecuzione di processo. Quando si sceglie questo collegamento, si accede ai log Amazon CloudWatch nel gruppo di log `/aws-glue/jobs/error`. Questi log possono essere utilizzati per visualizzare i dettagli su tutti gli errori riscontrati durante l'esecuzione del processo.
  - Log di output: collega ai log scritti in `stdout` per questa esecuzione del processo. Quando si sceglie questo collegamento, si accede ai log Amazon CloudWatch nel gruppo di log `/aws-glue/jobs/output`. Qui è possibile visualizzare i log per vedere tutti i dettagli sulle tabelle create in AWS Glue Data Catalog ed eventuali errori riscontrati.

Visualizzazione dei dettagli di un'esecuzione di un processo

È possibile scegliere un processo nell'elenco Job runs (Esecuzioni dei processi) nella pagina Monitoring (Monitoraggio), quindi scegliere View run details (Visualizza dettagli dell'esecuzione) per visualizzare informazioni dettagliate sull'esecuzione del processo.

Le informazioni visualizzate nella scheda dei dettagli dell'esecuzione del processo includono:

Proprietà	Description
Nome processo	Il nome del processo .
Stato di esecuzione	Lo stato attuale del processo eseguito. I valori possono essere: • `STARTING` • `RUNNING` • `STOPPING` • `STOPPED` • `SUCCEEDED` • `FAILED` • `TIMEOUT`
Versione Glue	La versione di AWS Glue utilizzata dall'esec uzione del processo.
Tentativo recente	Il numero di tentativi automatici per l'esecuzi one di questo processo.
Ora di inizio	La data e ora in cui questa esecuzione di processo è stata avviata.
Ora di fine	La data e ora in cui questa elaborazione di processo è stata completata.
Ora di inizio	La quantità di tempo impiegato per la preparazi one dell'esecuzione del processo.

Proprietà	Description
Ora di esecuzione	La quantità di tempo impiegato per l'esecuzione dello script del processo.
Nome trigger	Il nome del trigger associato al processo.
Ora ultima modifica	La data dell'ultima modifica apportata al processo.
Configurazione di sicurezza	La configurazione di sicurezza per il processo, che include la crittografia Amazon S3, la crittografia CloudWatch e le impostazioni di crittografia dei segnalibri di processo.
Timeout	Il valore della soglia di timeout per l'esecuzione del processo.
Capacità allocata	Il numero di unità di elaborazione dati (DPU) di AWS Glue allocate per questa esecuzione. Per ulteriori informazioni sulla pianificazione della capacità, consulta Monitoraggio per la pianificazione della capacità DPU nella Guida per gli sviluppatori di AWS Glue.
Capacità massima	La capacità massima disponibile per l'esecuzione del processo.
Numero di worker	Il numero di worker utilizzati per l'esecuzione del processo.

Proprietà	Description
Tipo di worker	Il tipo di worker predefiniti allocati per l'esecuzione del processo. I valori possono essere `G.1X` o `G.2X`. • **`G.1X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria, disco da 64 GB) e fornisce 1 esecutore. Consigliamo questo tipo di worker per i processi ad alto consumo di memoria. Questa è l'impostazione predefinita per Worker type (Tipo di worker) per la versione AWS Glue 2.0 o successive. • **`G.2X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria, disco da 128 GB) e fornisce 1 esecutore. Suggeriamo questo tipo di dipendente per i processi ad alto consumo di memoria e per i processi che eseguono trasformazioni machine learning.
Log	Un collegamento ai log del processo per la registrazione continua (`/aws-glue/jobs/logs-v2`).
Log di output	Un collegamento ai file di log di output del processo (`/aws-glue/jobs/output`).
Log di errore	Un collegamento ai file di log degli errori del processo (`/aws-glue/jobs/error`).

È inoltre possibile visualizzare i seguenti elementi aggiuntivi, disponibili anche quando si visualizzano le informazioni relative alle esecuzioni recenti dei processi. Per ulteriori informazioni, consulta the section called “Visualizzare le informazioni sulle esecuzioni dei processi recenti”.

- Inserimento di argomenti
- Log continui
- Parametri: puoi visualizzare le visualizzazioni dei parametri di base. Per ulteriori informazioni sui parametri inclusi, consulta the section called “Visualizzazione dei parametri di Amazon CloudWatch per l'esecuzione di un processo Spark”.
- Interfaccia utente Spark: puoi visualizzare i log di Spark relativi al processo nell'interfaccia utente di Spark. Per ulteriori informazioni sull'utilizzo dell'interfaccia utente di Spark, consulta the section called “Monitoraggio con l'interfaccia utente di Spark”. Abilita questa funzionalità seguendo la procedura in the section called “Abilitazione dell'interfaccia utente di Spark per processi”.

Visualizzazione dei parametri di Amazon CloudWatch per l'esecuzione di un processo Spark

Nella pagina dei dettagli per l'esecuzione di un processo, nella sezione Run details (Dettagli esecuzione) puoi visualizzare i parametri del processo. AWS Glue Studio invia i parametri del processo a Amazon CloudWatch per ciascuno dei processi eseguiti.

AWS Glue invia i parametri a Amazon CloudWatch ogni 30 secondi. I parametri AWS Glue rappresentano i valori delta rispetto ai valori segnalati in precedenza. Se appropriato, i pannelli di controllo dei parametri aggregano (sommano) i valori inviati ogni 30 secondi per ottenere un valore per l'intero ultimo minuto. I parametri Spark passati da AWS Glue a Amazon CloudWatch, invece, sono generalmente valori assoluti che rappresentano lo stato corrente nel momento in cui vengono segnalati.

> **Note**
>
> È necessario configurare l'account per accedere a Amazon CloudWatch.

I parametri forniscono informazioni sull'esecuzione del processo, ad esempio:

- Spostamento di dati ETL: il numero di byte letti da o scritti in Amazon S3.

- Profilo di memoria: heap utilizzata: il numero di byte di memoria utilizzati dall'heap Java Virtual Machine (JVM).
- Profilo di memoria: utilizzo heap: la frazione di memoria (scala: 0-1), mostrata come percentuale, utilizzata dall'heap JVM.
- Carico CPU: la frazione del carico di sistema della CPU utilizzata (dimensione: 0-1), indicata come percentuale.

Visualizzazione dei parametri di Amazon CloudWatch per l'esecuzione di un processo Ray

Nella pagina dei dettagli per l'esecuzione di un processo, nella sezione Run details (Dettagli esecuzione) puoi visualizzare i parametri del processo. AWS Glue Studio invia i parametri del processo a Amazon CloudWatch per ciascuno dei processi eseguiti.

AWS Glue invia i parametri a Amazon CloudWatch ogni 30 secondi. I parametri AWS Glue rappresentano i valori delta rispetto ai valori segnalati in precedenza. Se appropriato, i pannelli di controllo dei parametri aggregano (sommano) i valori inviati ogni 30 secondi per ottenere un valore per l'intero ultimo minuto. I parametri Spark passati da AWS Glue a Amazon CloudWatch, invece, sono generalmente valori assoluti che rappresentano lo stato corrente nel momento in cui vengono segnalati.

> **Note**
>
> È configurare l'account per accedere a Amazon CloudWatch, come descritto in .

Nei processi Ray, è possibile visualizzare i seguenti grafici di parametri aggregati. Con questi, è possibile creare un profilo del cluster e delle attività, nonché accedere a informazioni dettagliate su ciascun nodo. I dati di serie temporali che supportano questi grafici sono disponibili in CloudWatch per ulteriori analisi.

Profilo dell'attività: stato dell'attività

Mostra il numero di attività Ray nel sistema. A ogni ciclo di vita delle attività viene assegnata una serie temporale.

Profilo dell'attività: nome dell'attività

Mostra il numero di attività Ray nel sistema. Vengono mostrate solo le attività in sospeso e quelle attive. A ogni tipo di attività (in base al nome) viene assegnata una serie temporale distinta.

Profilo del cluster: CPU in uso

Mostra il numero di core della CPU utilizzati. A ogni nodo viene assegnata una serie temporale. I nodi sono identificati da indirizzi IP, che sono effimeri e vengono utilizzati solo per l'identificazione.

Profilo del cluster: utilizzo della memoria dell'archivio di oggetti

Mostra l'utilizzo della memoria da parte della cache degli oggetti Ray. A ogni posizione di memoria (memoria fisica, memorizzata nella cache su disco e riversata in Amazon S3) viene assegnata una serie temporale distinta. L'archivio oggetti gestisce l'archiviazione di dati su tutti i nodi del cluster. Per ulteriori informazioni, consulta la pagina Objects nella documentazione di Ray.

Profilo del cluster: conteggio dei nodi

Mostra il numero di nodi forniti per il cluster.

Dettaglio del nodo: utilizzo della CPU

Mostra l'utilizzo della CPU su ciascun nodo in percentuale. Ogni serie mostra una percentuale aggregata di utilizzo della CPU su tutti i core del nodo.

Dettaglio del nodo: utilizzo della memoria

Mostra l'utilizzo della memoria su ogni nodo in GB. Ogni serie mostra la memoria aggregata tra tutti i processi sul nodo, incluse le attività Ray e il processo di archiviazione di Plasma. Ciò non rifletterà gli oggetti archiviati su disco o riversati su Amazon S3.

Dettaglio del nodo: utilizzo del disco

Mostra l'utilizzo del disco su ogni nodo in GB.

Dettaglio del nodo: velocità di I/O del disco

Mostra l'I/O del disco su ogni nodo in KB/s.

Dettaglio del nodo: velocità di trasmissione effettiva di I/O di rete

Mostra l'I/O di rete su ogni nodo in KB/s.

Dettaglio del nodo: utilizzo della CPU da parte del componente Ray

Mostra l'utilizzo della CPU in parte dei core. A ogni componente Ray su ogni nodo viene assegnata una serie temporale.

Dettaglio del nodo: utilizzo della memoria da parte del componente Ray

Mostra l'utilizzo della memoria in GiB. A ogni componente Ray su ogni nodo viene assegnata una serie temporale.

Rileva ed elabora dati sensibili

La trasformazione Detect PII identifica le informazioni personali di identificazione (PII) nell'origine dati. È possibile scegliere l'entità PII da identificare, come si desidera che i dati vengano scansionati e cosa fare con l'entità PII identificata dalla trasformazione Detect PII.

La trasformazione Detect PII fornisce la possibilità di rilevare, mascherare o rimuovere le entità definite o che sono predefinite da AWS. Ciò consente di aumentare la conformità e ridurre la responsabilità. Ad esempio, potresti voler assicurarti che nei tuoi dati non esistano informazioni di identificazione personale che possano essere lette e mascherare i numeri di previdenza sociale con una stringa fissa (ad esempio xxx-xx-xxxx), numeri di telefono o indirizzi.

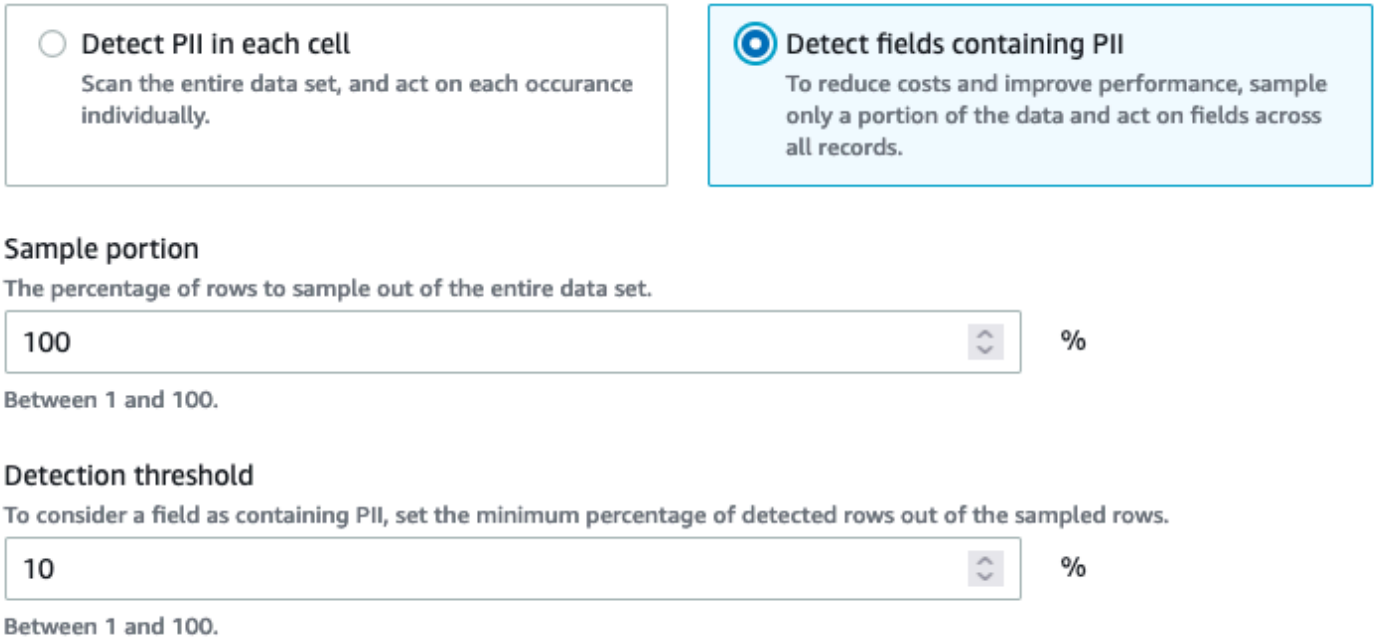
Per lavorare con dati sensibili al di fuori di AWS Glue Studio, consulta Utilizzo del rilevamento dei dati sensibili fuori da AWS Glue Studio

Argomenti

- Come scegliere il modo in cui desideri che vengano scansionati i dati
- Scelta delle entità PII da rilevare
- Specificazione del livello di distinzione di rilevamento
- Come scegliere cosa fare con i dati PII identificati
- Aggiungere sostituzioni di operazioni granulari

Come scegliere il modo in cui desideri che vengano scansionati i dati

Quando esegui la scansione del set di dati per i dati sensibili, come le informazioni di identificazione personale (PII), puoi scegliere di rilevare le PII in ogni riga o rilevare le colonne che contengono dati PII.



Quando scegli Detect PII in each cell (Rileva PII in ogni cella), stai scegliendo di scansionare tutte le righe nell'origine dati. Si tratta di una scansione completa per garantire che le entità PII siano identificate.

Quando scegli Detect fields containing PII (Rileva campi contenenti PII), stai scegliendo di scansionare un campione di righe per le entità PII. Questo è un modo per mantenere bassi costi e risorse, identificando al contempo i campi in cui si trovano le entità PII.

Quando si sceglie di rilevare i campi che contengono PII, è possibile ridurre i costi e migliorare le prestazioni campionando una parte di righe. La scelta di questa opzione ti permetterà di specificare opzioni aggiuntive:

- Sample portion (Porzione campione): consente di specificare la percentuale di righe da campionare. Ad esempio, se si immette "50", si specifica che si desidera il 50% delle righe scansionate per l'entità PII.
- Detection threshold (Soglia di rilevamento): consente di specificare la percentuale di righe che contengono l'entità PII in modo che l'intera colonna venga identificata come avente l'entità PII. Ad esempio, se inserisci "10", stai specificando che il numero dell'entità PII, US Phone, nelle righe scansionate deve essere pari o superiore al 10% per poter identificare il campo come entità PII, Numero di telefono degli Stati Uniti. Se la percentuale di righe che contengono l'entità PII è inferiore al 10%, tale campo non verrà etichettato come contenente l'entità PII, Numero di telefono degli Stati Uniti, al suo interno.

Scelta delle entità PII da rilevare

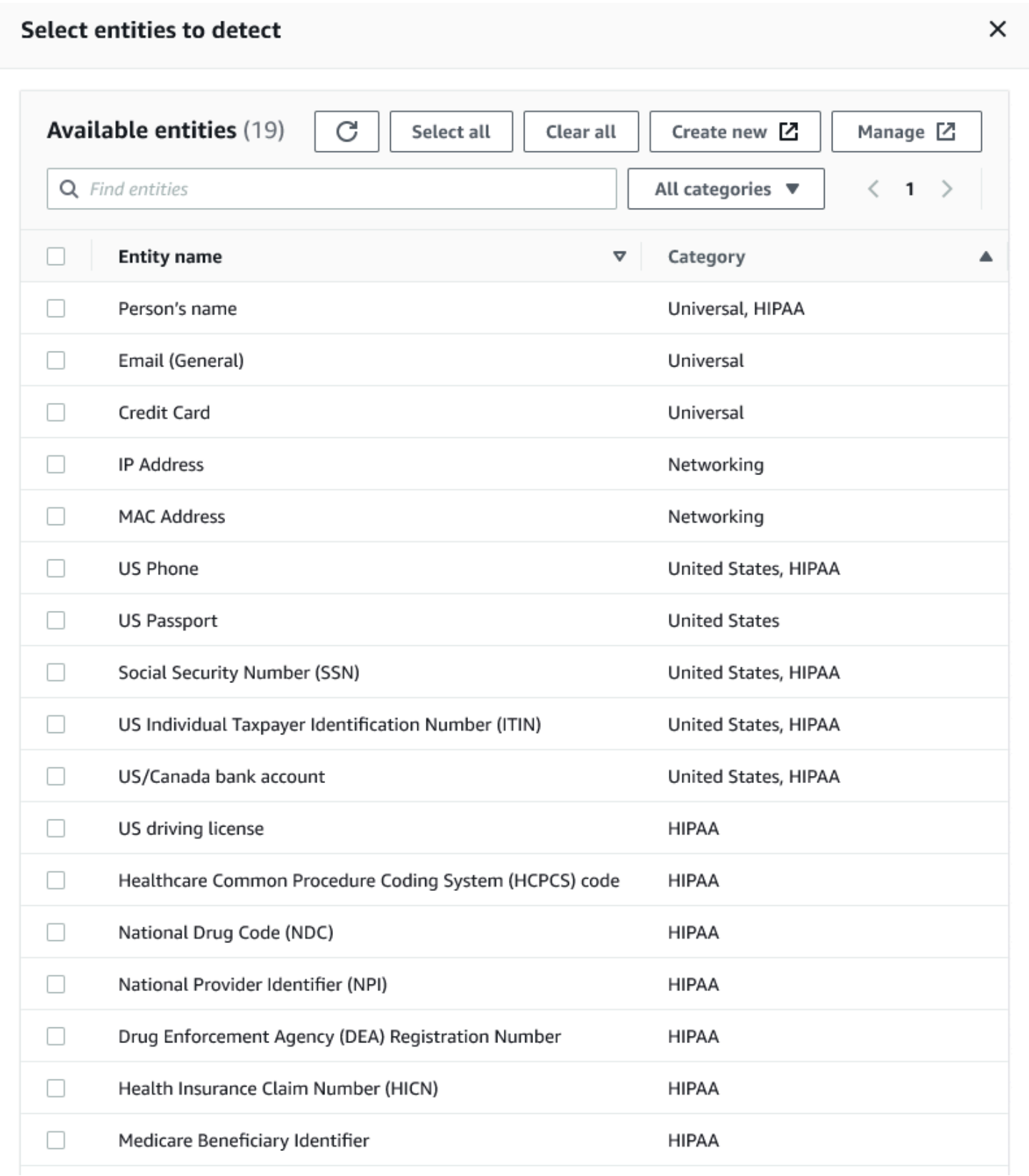
Se hai scelto Rileva PII in ogni cella puoi scegliere tra una delle tre opzioni:

- Tutti i modelli PII disponibili, incluse AWS le entità.
- Seleziona categorie: quando selezioni le categorie, i modelli PII includeranno automaticamente i modelli nelle categorie selezionate.
- Seleziona modelli specifici: verranno rilevati solo i modelli selezionati.

Per un elenco completo dei tipi di dati sensibili gestiti, consulta la pagina Managed Sensitive Data Types.

Scegli tra tutti i modelli PII disponibili

Se scegli Tutti i modelli PII disponibili, seleziona le entità predefinite da. AWS È possibile selezionare una, più di una o tutte le entità.


Select entities to detect
Available entities (19)
Select all
Clear all
Create new
Manage
Find entities
All categories
1
Entity name
Category
Person's name
Universal, HIPAA
Email (General)
Universal
Credit Card
Universal
IP Address
Networking
MAC Address
Networking
US Phone
United States, HIPAA
US Passport
United States
Social Security Number (SSN)
United States, HIPAA
US Individual Taxpayer Identification Number (ITIN)
United States, HIPAA
US/Canada bank account
United States, HIPAA
US driving license
HIPAA
Healthcare Common Procedure Coding System (HCPCS) code
HIPAA
National Drug Code (NDC)
HIPAA
National Provider Identifier (NPI)
HIPAA
Drug Enforcement Agency (DEA) Registration Number
HIPAA
Health Insurance Claim Number (HICN)
HIPAA
Medicare Beneficiary Identifier
HIPAA

Categorie di selezione

Se hai scelto Categorie di selezione come i modelli PII da rilevare, è possibile selezionare tra le opzioni del menu a discesa. Alcune entità possono appartenere a più di una categoria. Ad esempio: Nome della persona è un'entità che appartiene alle categorie Universale e HIPAA.

- Universale (esempi: e-mail, carta di credito)
- HIPAA (esempi: patente di guida statunitense, codice HCPCS [Healthcare Common Procedure Coding System])
- Rete (esempi: indirizzo IP, indirizzo MAC)
- Argentina
- Australia
- Austria
- Belgio
- Bosnia
- Bulgaria
- Canada
- Cile
- Colombia
- Croazia
- Cipro
- Cechia
- Danimarca
- Estonia
- Finlandia
- Francia
- Germania
- Grecia
- Ungheria
- Irlanda
- Corea
- Giappone

- Messico
- Paesi Bassi
- Nuova Zelanda
- Norvegia
- Portogallo
- Romania
- Singapore
- Slovacchia
- Slovenia
- Spagna
- Svezia
- Svizzera
- Turchia
- Ucraina
- Stati Uniti
- Regno Unito
- Venezuela

Seleziona modelli specifici

Se scegli Seleziona modelli specifici come modelli PII da rilevare, è possibile cercare o sfogliare da un elenco di modelli già creati o creare un nuovo modello di entità di rilevamento.

I passaggi riportati di seguito descrivono come creare un nuovo modello personalizzato per il rilevamento di dati sensibili. Creerai il modello personalizzato inserendo un nome per il modello, aggiungerai un'espressione regolare e, facoltativamente, definirai le parole contestuali.

1. Per creare un nuovo motivo, fare clic sul pulsante Creare nuovo.

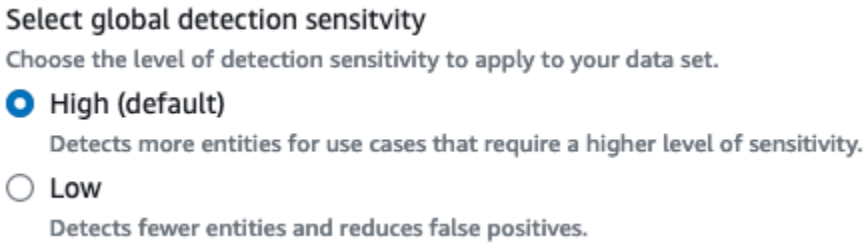
2. Nella pagina Crea entità di rilevamento, immettere il nome dell'entità e un'espressione regolare. L'espressione regolare (Regex) è quella che AWS Glue utilizzerà per abbinare le entità.
3. Fare clic su Convalida. Se la convalida ha esito positivo, verrà visualizzato un messaggio di conferma che indica che la stringa è un'espressione regolare valida. Se la convalida non ha esito positivo, verrà visualizzato un messaggio che indica che la stringa non è conforme alla formattazione corretta e ai valori letterali, operatori o costrutti dei caratteri accettati.
4. È possibile scegliere di aggiungere parole di contesto oltre all'espressione regolare. Le parole contestuali possono aumentare la probabilità di una corrispondenza. Questi possono essere utili nei casi in cui i nomi dei campi non sono descrittivi dell'entità. Ad esempio, i numeri di previdenza sociale possono essere denominati "SSN" o "SS". L'aggiunta di queste parole contestuali può aiutare a far corrispondere l'entità.
5. Fare clic su Crea per creare l'entità di rilevamento. Tutte le entità create sono visibili nella console AWS Glue Studio. Fai clic su Entità di rilevamento nel menu di navigazione a sinistra.

   È possibile modificare, eliminare o creare entità di rilevamento dalla pagina Entità di rilevamento. È inoltre possibile ricercare un modello utilizzando il campo di ricerca.

Specificazione del livello di distinzione di rilevamento

È possibile impostare il livello di distinzione quando si utilizza il rilevamento di dati sensibili.

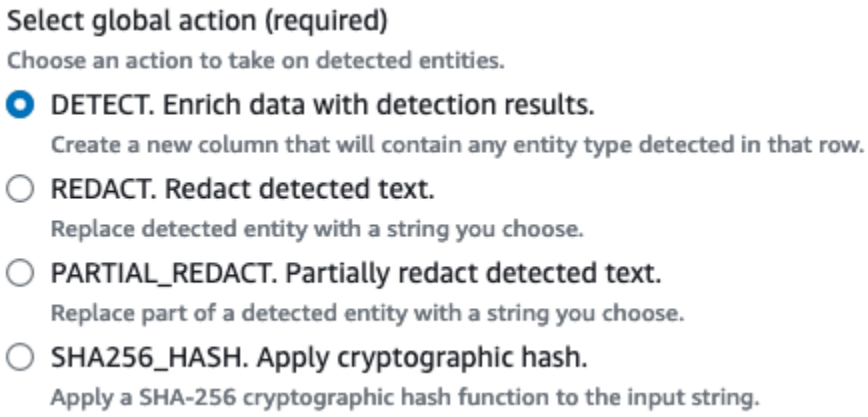
- Alto: (impostazione predefinita) rileva più entità per i casi d'uso che richiedono un livello di distinzione più elevato. Tutti i processi AWS Glue creati dopo novembre 2023 vengono automaticamente attivati per questa impostazione.
- Bassa: rileva un minor numero di entità e riduce i falsi positivi.



Come scegliere cosa fare con i dati PII identificati

Se hai deciso di rilevare le PII nell'intera origine dati, puoi selezionare l'applicazione di un'azione globale:

- Enrich data with detection results (Arricchisci i dati con i risultati di rilevamento): se scegli Detect PII in ogni cella, potrai archiviare le entità rilevate in una nuova colonna.
- Redact detected text (Rivedi il testo rilevato): è possibile sostituire il valore PII rilevato con una stringa specificata nel campo opzionale Sostituzione del testo. Se non viene specificata alcuna stringa, l'entità PII rilevata viene sostituita con "********".
- Rivedi parzialmente il testo rilevato: è possibile sostituire il valore PII rilevato con una stringa scelta. Esistono due opzioni possibili: lasciare le estremità smascherate o mascherarle fornendo un modello regex esplicito. Questa caratteristica non è disponibile in AWS Glue 2.0.
- Applica hash di crittografia: puoi passare il valore PII rilevato a una funzione hash di crittografia SHA-256 e sostituire il valore con l'output della funzione.



Differenze tra le versioni di AWS Glue 2.0 e 3.0+

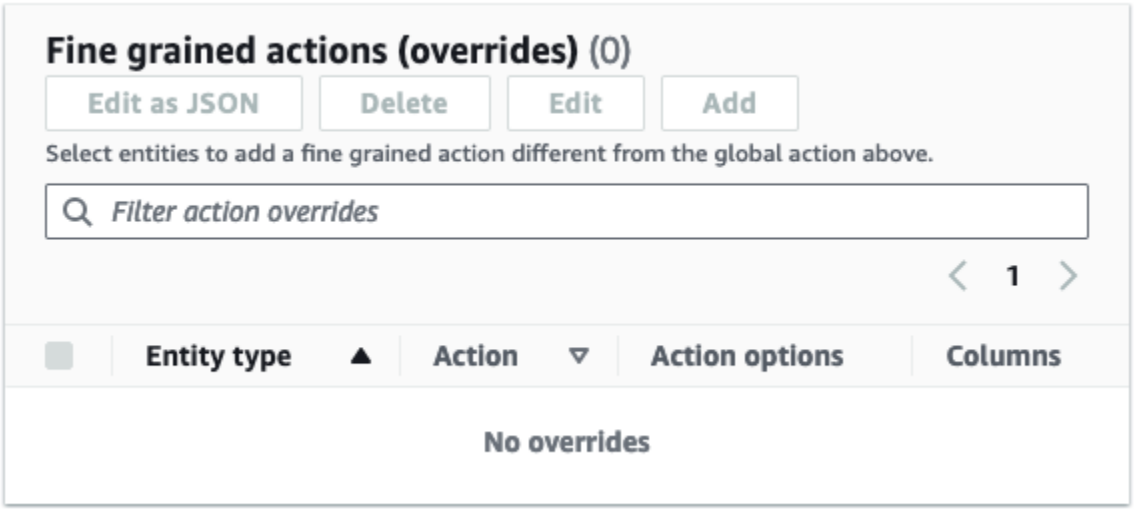
AWS Glue2.0 jobs ne restituirà una nuova DataFrame con le informazioni PII rilevate per ogni colonna in una colonna supplementare. Qualsiasi processo di redazione o hash è visibile all'interno dello script AWS Glue nella scheda visiva.

AWS GlueI lavori 3.0 e 4.0 ne restituiranno uno nuovo DataFrame con la stessa colonna supplementare. È presente una nuova chiave per "actionUsed" che può essere una tra `DETECT`, `REDACT`, `PARTIAL_REDACT` o `SHA256_HASH`. Se viene selezionata un'azione di mascheramento, DataFrame restituirà dati con dati sensibili mascherati.

Aggiungere sostituzioni di operazioni granulari

È possibile aggiungere ulteriori impostazioni di rilevamento e azione alla tabella dettagliata delle sostituzioni di azioni. Ciò consente di:

- Includere o escludere determinate colonne dal rilevamento: uno schema dedotto sull'origine dati popolerà la tabella con le colonne disponibili.
- Definire le impostazioni specifiche più granulari rispetto all'utilizzo di azioni globali: ad esempio, è possibile specificare diverse impostazioni del testo di redazione per diversi tipi di entità.
- Specificare un'azione diversa da quella globale: se si desidera applicare un'azione diversa a un tipo di dati sensibili diverso, è possibile farlo qui. Tieni presente che non è possibile utilizzare due edit-in-place azioni diverse (redazione e hashing) sulla stessa colonna, ma è sempre possibile utilizzare detect.



Gestione dei processi ETL con AWS Glue Studio

Per gestire i processi ETL, puoi utilizzare l'interfaccia visiva semplice di AWS Glue Studio. Nel pannello di navigazione, scegli Jobs (Processi) per visualizzare la pagina Jobs (Processi) In questa pagina puoi visualizzare tutti i processi creati con AWS Glue Studio o con la console AWS Glue. In questa pagina puoi visualizzare, gestire ed eseguire i processi.

In questa pagina puoi anche eseguire le seguenti operazioni:

- Avviare un'esecuzione del processo
- Pianificazione delle esecuzioni dei processi

- Gestione delle pianificazioni dei processi
- Interruzione dei processi
- Visualizzazione dei processi
- Visualizzare le informazioni sulle esecuzioni dei processi recenti
- Visualizzare lo script del processo
- Modificare le proprietà del processo
- Salvare il lavoro
- Clonazione di un processo
- Eliminazione dei processi

Avviare un'esecuzione del processo

In AWS Glue Studio, puoi eseguire i processi on demand. Un processo può essere eseguito più volte e ogni volta AWS Glue raccoglie informazioni sulle attività del processo e sulle prestazioni. Queste informazioni sono indicate come esecuzione del processo e sono identificate da un ID di esecuzione del processo.

Puoi avviare l'esecuzione di un processo in AWS Glue Studio nei seguenti modi:

- Nella pagina Jobs (Processi), scegli il processo che vuoi avviare, quindi scegli il pulsante Run job (Esecuzione del processo).
- Se stai visualizzando un processo nell'editor visivo e questo è stato salvato, puoi scegliere il pulsante Run (Esegui) per avviare l'esecuzione di un processo.

Per ulteriori informazioni sulle esecuzioni dei processi, consulta Uso di processi nella console AWS Glue nella Guida per gli sviluppatori di AWS Glue.

Pianificazione delle esecuzioni dei processi

In AWS Glue Studio, è possibile creare una pianificazione per eseguire i processi in orari specifici. Puoi specificare i vincoli, ad esempio il numero di volte in cui vengono eseguiti i processi, i giorni della settimana in cui vengono eseguiti e a che ora. Questi vincoli si basano sul comando `cron` e hanno le stesse limitazioni di `cron`. Ad esempio, se vuoi eseguire il processo il giorno 31 di ogni mese, devi ricordare che alcuni mesi non sono di 31 giorni. Per ulteriori informazioni su `cron`, vedi le espressioni Cron nella Guida per gli sviluppatori di AWS Glue.

Per eseguire i processi in base a una pianificazione

1. Crea una pianificazione del processo utilizzando uno dei seguenti metodi:

   - Nella pagina Jobs (Processi), scegli il processo per il quale creare una pianificazione, scegli Actions (Operazioni), quindi Schedule job (Pianifica processo).
   - Se stai visualizzando un processo nell'editor visivo e questo è stato salvato, puoi scegliere la scheda Schedules (pianificazioni). Quindi scegli Create Schedule (Crea pianificazione).

2. Nella pagina Schedule job run (Pianifica esecuzione del processo), inserisci le seguenti informazioni:

   - Name (Nome): inserisci un nome per il processo.
   - Frequency (Frequenza): inseriscila frequenza per la programmazione del processo. Puoi scegliere le seguenti opzioni:
     - Hourly (Orario): il processo verrà eseguito ogni ora, a partire da un minuto specifico. È possibile specificare gli attributi Minute (Minuto) dell'ora in cui il processo deve essere eseguito. Per impostazione predefinita, quando si sceglie la programmazione oraria, il processo viene eseguito all'inizio dell'ora (minuto 0).
     - Daily (Giornaliero): il processo verrà eseguito ogni giorno, a partire da un momento. È possibile specificare gli attributi Minute (Minuto) dell'ora in cui il processo deve essere eseguito e Start hour (Ora di avvio). Le ore sono specificate utilizzando un orologio di 23 ore, in cui si utilizzano i numeri da 13 a 23 per le ore pomeridiane. Il valore predefinito per i minuti e le ore è 0, il che significa che se si seleziona Daily (Giornaliero), il processo verrà eseguito per impostazione predefinita a mezzanotte.
     - Weekly (Settimanale): il processo verrà eseguito uno o più giorni della ogni settimana. Oltre alle stesse impostazioni descritte in precedenza per Daily (Giornaliero), è possibile scegliere i giorni della settimana in cui verrà eseguito il processo. È possibile scegliere uno o più giorni.
     - Monthly (Mensile): il processo verrà eseguito ogni mese in un giorno specifico. Oltre alle stesse impostazioni descritte in precedenza per Daily (Giornaliero), è possibile scegliere i giorni del mese in cui verrà eseguito il processo. Specifica il giorno come un valore numerico compreso tra 1 e 31. Se si seleziona un giorno che in un mese non esiste, ad esempio il 30 febbraio, il processo in quel mese non viene eseguito.
     - Custom (Personalizzato): inserisci un'espressione per la pianificazione del processo utilizzando la sintassi `cron`. Le espressioni cron permettono di creare pianificazioni più

complicate, ad esempio l'ultimo giorno del mese (invece di un giorno specifico del mese) o ogni terzo mese dai giorni 7 al 21.

Consulta le espressioni cron nella Guida per gli sviluppatori di AWS Glue

- Description (Descrizione): è possibile inserire una descrizione per la programmazione dei processi. Se prevedi di utilizzare la stessa pianificazione per più processi, una descrizione può rendere più facile determinare il relativo funzionamento.

3. Scegli Create schedule (Crea pianificazione) per salvare la pianificazione del processo.
4. Dopo aver creato la pianificazione, nella parte superiore della pagina della console viene visualizzato un messaggio di operazione riuscita. Puoi selezionare Job details (Dettagli del processo) in questo banner per visualizzare i dettagli. Si apre la pagina dell'editor visivo dei processi, con la scheda Schedules (Piani) selezionata.

Gestione delle pianificazioni dei processi

Dopo aver creato le pianificazioni per un processo, puoi aprirlo nell'editor visivo e scegliere la casella di controllo Schedules (Piani) per gestire le pianificazioni.

Nella scheda Schedules (Pianificazioni) dell'editor visivo, puoi eseguire le seguenti attività:

- Creare una nuova pianificazione.

  Scegli Create schedule (Crea pianificazione), quindi inserisci le informazioni per la pianificazione come descritto in the section called “Pianificazione delle esecuzioni dei processi”.

- Modificare una pianificazione esistente.

  Seleziona la pianificazione da modificare, quindi Action (Operazioni) e poi Edit schedule (Modifica pianificazione). Quando scegli di modificare una pianificazione esistente, la frequenza è personalizzata e la pianificazione viene visualizzata come espressione `cron`. Puoi modificare l'espressione `cron` oppure specificare una nuova pianificazione utilizzando l'opzione Frequency (Frequenza). Una volta terminate le modifiche, seleziona Update schedule (Aggiorna pianificazione).

- Sospendere una pianificazione attiva.

  Seleziona una pianificazione attiva, quindi Action (Operazioni) e Pause schedule (Sospendi pianificazione). La pianificazione viene disattivata immediatamente. Seleziona il pulsante di aggiornamento (ricarica) per visualizzare lo stato aggiornato della pianificazione.

- Riprendere una pianificazione sospesa.

  Seleziona una pianificazione attiva, quindi Action (Operazioni) e Resume schedule (Riprendi pianificazione). La pianificazione viene attivata immediatamente. Seleziona il pulsante di aggiornamento (ricarica) per visualizzare lo stato aggiornato della pianificazione.

- Eliminare una pianificazione.

  Seleziona la pianificazione da rimuovere, quindi Action (Operazioni) e poi Delete schedule (Elimina pianificazione). La pianificazione viene eliminata immediatamente. Seleziona il pulsante di aggiornamento (ricarica) per visualizzare l'elenco delle pianificazioni aggiornato. La pianificazione mostrerà lo stato Deleting (Eliminazione in corso) fino a quando non è completamente rimossa.

Interruzione dei processi

Puoi interrompere un processo prima che abbia completato l'esecuzione. Puoi scegliere questa opzione se sai che il processo non è configurato correttamente o se richiede troppo tempo per essere completato.

Nella pagina Monitoring (Monitoraggio), nell'elenco Job runs (Esecuzioni di processo), scegli il processo da interrompere, quindi seleziona Actions (Operazioni) e poi Stop run (Interrompi esecuzione).

Visualizzazione dei processi

Puoi visualizzare tutti i processi nella pagina Jobs (Processi). Per accedere a questa pagina, seleziona Jobs (Processi) nel pannello di navigazione.

Nella pagina Jobs (Processi) puoi visualizzare tutti i processi creati nell'account. L'elenco Your jobs (I tuoi processi) mostra il nome del processo, il tipo, lo stato dell'ultima esecuzione del processo e le date in cui è stato creato e modificato per l'ultima volta. Puoi selezionare il nome di un processo per visualizzare le relative informazioni dettagliate.

Puoi anche utilizzare il pannello di controllo Your jobs (Monitoraggio) per visualizzare tutti i processi. Puoi accedere al pannello di controllo selezionando Monitoring (Monitoraggio) nel pannello di navigazione.

Personalizzazione della visualizzazione del processo

Puoi personalizzare la modalità di visualizzazione dei processi nella sezione Your jobs (I tuoi processi) della pagina Jobs (Processi). Puoi inoltre inserire del testo nel campo di ricerca per visualizzare solo i lavori con un nome che contiene tale testo.

Selezionando l'icona delle impostazioni

nella sezione Your jobs (I tuoi processi), puoi personalizzare il modo in cui AWS Glue Studio mostra le informazioni nella tabella. Puoi scegliere di inserire a capo le righe di testo nella visualizzazione, modificare il numero di processi visualizzati nella pagina e specificare le colonne da visualizzare.

Visualizzare le informazioni sulle esecuzioni dei processi recenti

Un processo può essere eseguito più volte man mano che nuovi dati vengono aggiunti nella posizione di origine. Ogni volta che un processo viene eseguito, all'esecuzione viene assegnato un ID univoco e vengono raccolte informazioni su tale esecuzione. Puoi visualizzare queste informazioni utilizzando i seguenti metodi.

- Seleziona la scheda Runs (Esecuzioni) dell'editor visivo per visualizzare le informazioni sull'esecuzione per il processo attualmente mostrato.

  Nella scheda Runs(Esecuzioni) (la pagina Recent job runs [Esecuzioni dei processi recenti]), è presente una scheda per ogni esecuzione del processo. Le informazioni visualizzate nella scheda Runs (Esecuzioni) includono:
  - ID dell'esecuzione del processo
  - Numero di tentativi di esecuzione del processo
  - Stato dell'esecuzione del processo
  - Ora di inizio e fine dell'esecuzione del processo
  - Il runtime per l'esecuzione del processo
  - Un collegamento ai file di log del processo
  - Un collegamento ai file di log degli errori del processo
  - L'errore restituito per i processi non riusciti
- Puoi selezionare l'esecuzione di un processo per visualizzarne le informazioni aggiuntive, tra cui:
  - Inserimento di argomenti
  - Log continui

- Parametri: puoi visualizzare le visualizzazioni dei parametri di base. Per ulteriori informazioni sui parametri inclusi, consulta the section called “Visualizzazione dei parametri di Amazon CloudWatch per l'esecuzione di un processo Spark”.
- Interfaccia utente Spark: puoi visualizzare i log di Spark relativi al processo nell'interfaccia utente di Spark. Per ulteriori informazioni sull'utilizzo dell'interfaccia utente di Spark, consulta the section called “Monitoraggio con l'interfaccia utente di Spark”. Abilita questa funzionalità seguendo la procedura in the section called “Abilitazione dell'interfaccia utente di Spark per processi”.

È possibile selezionare Visualizza dettagli per visualizzare informazioni simili nella pagina dei dettagli dell'esecuzione del processo. In alternativa, è possibile accedere alla pagina dei dettagli dell'esecuzione del processo tramite la pagina Monitoraggio. Nel riquadro di navigazione, scegli Monitoring (Monitoraggio). Scorri in basso fino all'elenco Job runs (Esecuzioni processo). Scegli il processo e poi scegli View run details (Visualizza i dettagli dell'esecuzione). I contenuti sono descritti in Visualizzazione dei dettagli di un'esecuzione di un processo.

Per ulteriori informazioni sui log del processo, consulta Visualizzazione dei log di esecuzione del processo.

Visualizzare lo script del processo

Dopo aver fornito informazioni per tutti i nodi nel processo, AWS Glue Studio genera uno script utilizzato dal processo per leggere i dati dall'origine, trasformarli e scriverli nella posizione di destinazione. Salvando il processo, puoi visualizzare questo script in qualsiasi momento.

Per visualizzare lo script generato per il processo

1. Nel riquadro di navigazione seleziona Jobs (Processi).
2. Nella pagina Jobs (Processi), nell'elenco Your Jobs (I tuoi processi), scegli il nome del processo da esaminare. In alternativa, puoi selezionare un processo nell'elenco, selezionare il menu Actions (Operazioni), quindi scegliere Edit job (Modifica il processo).
3. Nella pagina dell'editor visivo, scegliere la scheda Script nella parte superiore per visualizzare lo script del processo.

   Se desideri modificare lo script del processo, consulta AWS Glue guida alla programmazione.

Modificare le proprietà del processo

I nodi nel diagramma processo definiscono le azioni eseguite dal processo, ma sono disponibili anche diverse proprietà che è possibile configurare per il processo. Queste proprietà determinano l'ambiente in cui viene eseguito il processo, le risorse utilizzate, le impostazioni di soglia, le impostazioni di sicurezza e altro ancora.

Per personalizzare l'ambiente di esecuzione dei processi

1. Nel riquadro di navigazione seleziona Jobs (Processi).
2. Nella pagina Jobs (Processi), nell'elenco Your Jobs (I tuoi processi), scegli il nome del processo da esaminare.
3. Nella pagina dell'editor visivo, scegliere la scheda Job details (Dettagli del processo) nella parte superiore del pannello di modifica del processo.
4. Modifica le proprietà del processo secondo le necessità.

   Per ulteriori informazioni sulle proprietà del processo, consulta Definizione delle proprietà del processo nella Guida per gli sviluppatori di AWS Glue.
5. Espandi la sezione Advanced properties (Proprietà avanzate) se devi specificare queste proprietà aggiuntive del processo:

   - Script filename (Nome del file di script): il nome del file che memorizza lo script del processo in Amazon S3.
   - Script path (Percorso dello script): la posizione di Amazon S3 in cui è memorizzato lo script del processo.
   - Job metrics (Parametri del processo): (non disponibile per i lavori di shell Python) attiva la creazione di parametri Amazon CloudWatch durante l'esecuzione del processo.
   - Continuous logging (Registrazione continua): (non disponibile per i lavori di shell Python) attiva la registrazione continua su CloudWatch, in modo che i log siano disponibili per la visualizzazione prima del completamento del processo.
   - Spark UI (Interfaccia utente di Spark) e Spark UI logs path (Percorso dei log dell'interfaccia utente Spark): (non disponibile per i processi di shell Python) attiva l'uso dell'interfaccia utente Spark per il monitoraggio del processo e specifica la posizione per i log dell'interfaccia utente di Spark.
   - Maximum concurrency (Simultaneità massima): imposta il numero massimo di esecuzioni simultanee consentite per il processo.

- Temporary path (Percorso temporaneo): il percorso di una directory di lavoro in Amazon S3, in cui vengono scritti i risultati intermedi temporanei quando AWS Glue esegue lo script.
- Delay notification threshold (minutes) (Soglia notifica di ritardo [minuti]): specifica una soglia di ritardo per il processo. Se il processo viene eseguito per un tempo più lungo di quello specificato dalla soglia, AWS Glue invia una notifica di ritardo per il processo a CloudWatch.
- Security configuration (Configurazione di sicurezza) e Server-side encryption (Crittografia lato server): usa questi campi per scegliere le opzioni di crittografia per il processo.
- Use Glue Data Catalog as the Hive metastore (Usa il catalogo dati di Glue come metastore Hive): scegli questa opzione per utilizzare AWS Glue Data Catalog come alternativa ad Apache Hive Metastore.
- Additional network connection (Connessione di rete aggiuntiva): per un'origine dati in un VPC, puoi specificare una connessione di tipo `Network` per assicurarti che il processo acceda ai tuoi dati tramite il VPC.
- Python library path (Percorso libreria Python), Dependent jars path (Percorso file .jar dipendenti) (non disponibili per i processi di shell Python) o Referenced files path (Percorso file di riferimento): utilizza questi campi per specificare la posizione dei file aggiuntivi utilizzati dal processo durante l'esecuzione dello script.
- Job Parameters (Parametri del processo): puoi aggiungere un insieme di coppie chiave-valore che vengono passate come parametri denominati allo script del processo. Quando in Python si chiamano le AWS Glue API, è preferibile passare i parametri in modo esplicito usando il nome. Per ulteriori informazioni sull'utilizzo dei parametri in uno script di processo, consulta Passaggio e accesso ai parametri Python in AWS Glue nella Guida per gli sviluppatori di AWS Glue.
- Tag: puoi aggiungere tag al processo per facilitarne l'organizzazione e l'individuazione.

6. Dopo aver modificato le proprietà del processo, salvalo.

Memorizza i file Spark shuffle su Amazon S3

Alcuni processi ETL richiedono la lettura e la combinazione di informazioni da più partizioni, ad esempio quando si utilizza una trasformazione di join. Questa operazione è indicata come shuffle. Durante uno shuffle, i dati vengono scritti su disco e trasferiti attraverso la rete. Con AWS Glue versione 3.0, puoi configurare Amazon S3 come posizione di storage per questi file. AWS Glue fornisce un gestore shuffle che scrive e legge file shuffle da e verso Amazon S3. La scrittura e la lettura di file shuffle da Amazon S3 è più lenta (del 5-20%) rispetto al disco locale (o Amazon EBS che è estremamente ottimizzato per Amazon EC2). Tuttavia, Amazon S3 offre una capacità

di archiviazione illimitata, pertanto non è necessario preoccuparsi errori "`No space left on device`" durante l'esecuzione del lavoro.

Per configurare il processo per l'utilizzo di Amazon S3 per i file shuffle

1. Nella pagina Jobs (Processi), nell'elenco Your Jobs (I tuoi processi), scegli il nome del processo da modificare.
2. Nella pagina dell'editor visivo, scegliere la scheda Job details (Dettagli del processo) nella parte superiore del pannello di modifica del processo.

   Scorri verso il basso fino alla sezione Job parameters (Parametri del processo).
3. Specifica le seguenti coppie chiave-valore.

   - `--write-shuffle-files-to-s3` — `true`

     Questo è il parametro principale che configura il gestore shuffle in AWS Glue per utilizzare i bucket Amazon S3 per la scrittura e la lettura di dati shuffle. Per impostazione predefinita, questo parametro ha un valore di `false`.
   - (Facoltativo) `--write-shuffle-spills-to-s3` — `true`

     Questo parametro consente di scaricare i file di riversamento nei bucket Amazon S3, che fornisce ulteriore resilienza al processo Spark in AWS Glue. Questo è necessario solo per carichi di lavoro di grandi dimensioni che riversano molti dati sul disco. Per impostazione predefinita, questo parametro ha un valore di `false`.
   - (Facoltativo) `--conf spark.shuffle.glue.s3ShuffleBucket` — `S3://<shuffle-bucket>`

     Questo parametro specifica il bucket Amazon S3 da utilizzare durante la scrittura dei file shuffle. Se non viene impostato, la posizione è la cartella `shuffle-data` nella posizione specificata per Temporary path (Percorso temporaneo) (`--TempDir`).

     > **Note**
     >
     > Verifica che la posizione del bucket di shuffle si trovi nella stessa Regione AWS in cui viene eseguito il processo.
     > Inoltre, il servizio shuffle non pulisce i file al termine dell'esecuzione del processo, pertanto è necessario configurare le policy del ciclo di vita dello storage Amazon S3

nella posizione del bucket shuffle. Per ulteriori informazioni, consulta Gestione del ciclo di vita dello storage nella Guida per l'utente di Amazon S3.

Salvare il lavoro

Finché non si salva il processo, a sinistra della finestra Save (Salva) viene visualizzato un messaggio in rosso che indica che il processo non è stato salvato.

Per salvare il processo

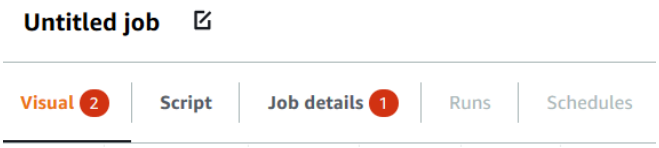
1. Fornisci tutte le informazioni richieste nelle schede Visual (Visivo) e Job details (Dettagli del processo).
2. Seleziona il pulsante Save (Salva).

   Dopo aver salvato il processo, il messaggio 'non salvato' si modifica per mostrare l'ora e la data dell'ultimo salvataggio.

Se esci da AWS Glue Studio prima di salvare il processo, all'accesso successivo ad AWS Glue Studio, verrà visualizzata una notifica. La notifica indica che esiste un processo non salvato e chiede se si desidera ripristinarlo. Se si sceglie di ripristinare il processo, è possibile continuare a modificarlo.

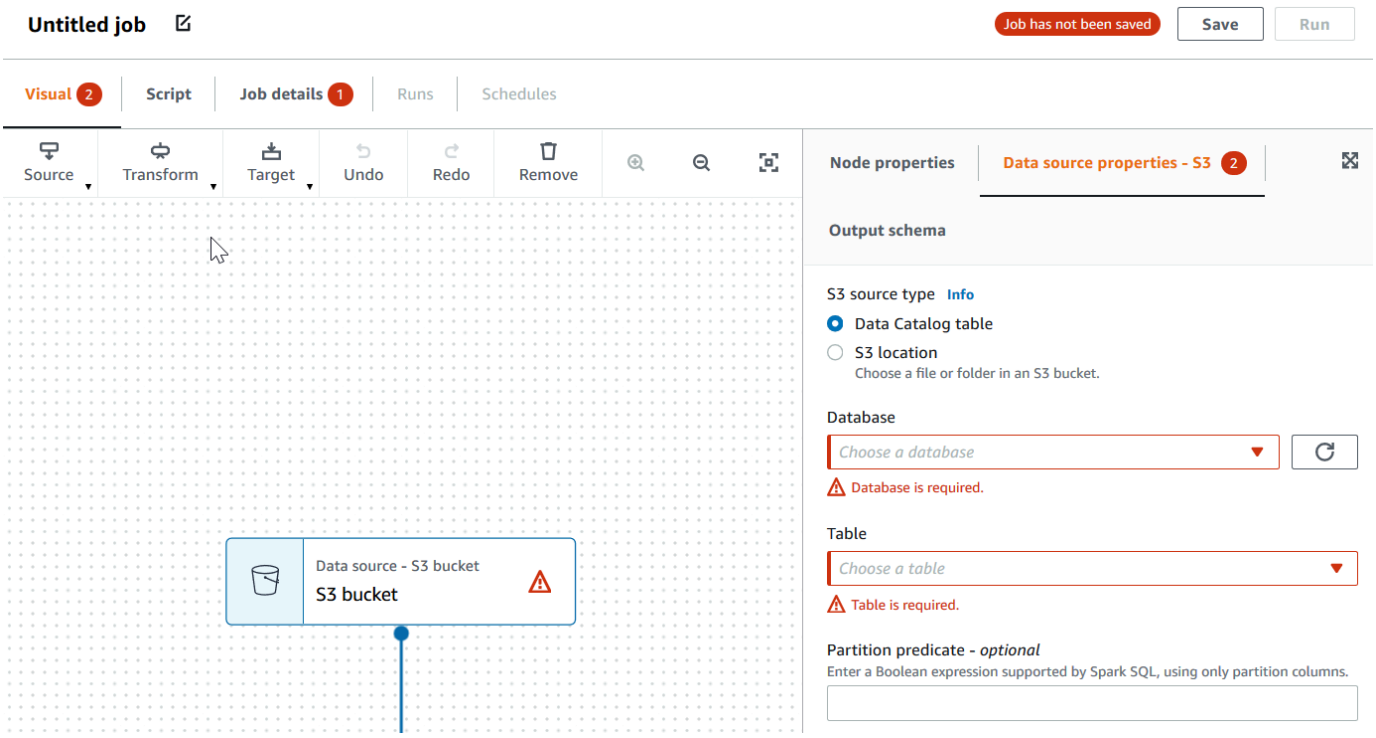
Risoluzione dei problemi relativi al salvataggio di un processo

Se scegli l'opzione Save (Salva), ma nel tuo lavoro mancano alcune informazioni richieste, nella scheda in cui mancano le informazioni viene visualizzato un messaggio in rosso. Il numero nel messaggio indica quanti campi mancanti sono stati rilevati.

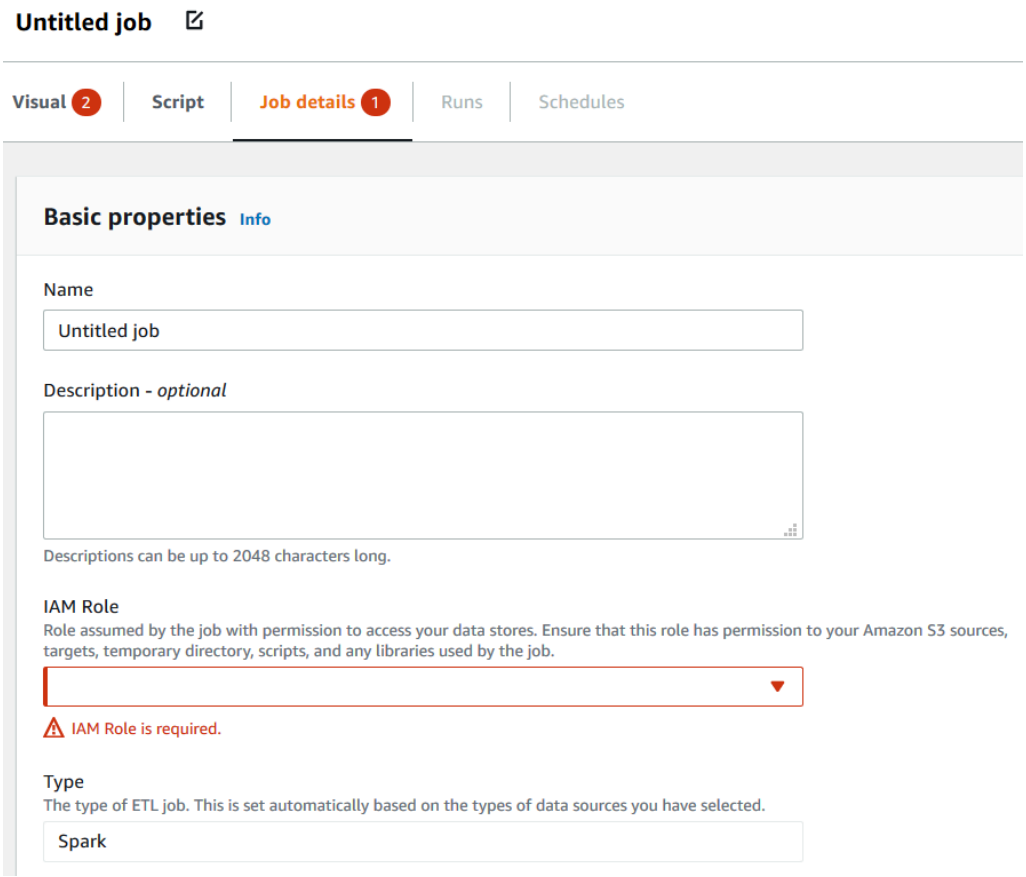


- Se un nodo nell'editor visivo non è configurato correttamente, la scheda Visual (Visivo) mostra un messaggio in rosso e il nodo con l'errore mostra un simbolo di avvertenza ⚠ .

1. Seleziona il nodo. Nel pannello dei dettagli del nodo, nella scheda in cui si trovano le informazioni mancanti o errate viene visualizzato un messaggio in rosso.
2. Scegli la scheda nel pannello dei dettagli del nodo che mostra un messaggio in rosso, quindi individua i campi interessati dal problema, che sono evidenziati. Un messaggio di errore sotto i campi fornisce ulteriori informazioni sul problema.



- Se si verifica un problema con le proprietà del processo, la scheda Job details (Dettagli del processo) mostra un messaggio in rosso. Scegli quella scheda e individua i campi interessati dal problema, che sono evidenziati. Il messaggio di errore sotto i campi fornisce ulteriori informazioni sul problema.



Clonazione di un processo

Puoi utilizzare l'operazione Clone job (Clona processo) per copiare un processo esistente in un nuovo processo.

Per creare un nuovo processo copiando un processo esistente

1. Nella pagina Jobs (Processi), nell'elenco Your Jobs (I tuoi processi), scegli il processo da duplicare.
2. Nel menu Actions (Operazioni) scegli Clone job (Clona processo).
3. Inserisci un nome per il processo. Puoi quindi salvare o modificare il processo.

Eliminazione dei processi

È possibile rimuovere i processi che non sono più necessari. È possibile eliminare uno o più processi in un'unica operazione.

Come eliminare i processi da AWS Glue Studio

1. Nella pagina Jobs (Processi), nell'elenco Your Jobs (I tuoi processi), scegli il processo da eliminare.
2. Nel menu Actions (Operazioni) seleziona Delete job (Elimina processo).
3. Conferma di voler eliminare il processo inserendo **`delete`**.

È inoltre possibile eliminare un processo salvato durante la visualizzazione della scheda Job details (Dettagli del processo) per quel lavoro nell'editor visivo.

Utilizzo dei processi in AWS Glue

Nelle sezioni seguenti vengono fornite informazioni sui processi ETL e Ray in AWS Glue.

Argomenti

Versioni AWS Glue

È possibile configurare il parametro della versione di AWS Glue quando si aggiunge o si aggiorna un processo. La versione AWS Glue determina le versioni di Apache Spark e Python supportate da AWS Glue. La versione Python indica la versione supportata per i processi di tipo Spark. La tabella seguente elenca le versioni AWS Glue disponibili, le versioni Spark e Python corrispondenti e altre modifiche di funzionalità.

Versioni AWS Glue

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
AWS Glue4.0	Versioni dell'ambiente Spark • Spark 3.3.0 • Python 3.10	AWS Glue 4.0 è la versione più recente di AWS Glue. In questa versione di AWS Glue sono presenti diverse ottimizza zioni e aggiornamenti, come: • Numerosi aggiornamenti delle funzionalità Spark da Spark 3.1 a Spark 3.3:

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
		• Diversi miglioramenti delle funzionalità se abbinato a Pandas. Per ulteriori informazioni, consulta Novità di Spark 3.3. • Ottimizzazioni aggiuntiv e sviluppate su Amazon EMR. • Aggiornamento a EMR File System (EMRFS) 2.53. • Migrazione a Log4j 2 da Log4j 1.x • Diversi aggiornamenti del modulo Python da AWS Glue 3.0, come una versione aggiornata di Boto. • Aggiornamento di diversi connettori, tra cui il connettore Amazon Redshift predefinito. Per informazi oni, consulta Appendice C: Aggiornamenti dei connettor i. • Aggiornamento di diversi driver JDBC. Per informazi oni, consulta Appendice B: aggiornamenti dei driver JDBC. • Aggiornato con un nuovo connettore Amazon Redshift e driver JDBC.

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
		• Supporto nativo per framework open data lake con Apache Hudi, Delta Lake e Apache Iceberg. • Supporto nativo per il Cloud Shuffle Storage Plugin basato su Amazon S3 (un plug-in Apache Spark) per utilizzare Amazon S3 per lo shuffling e la capacità di archiviazione elastica. Limitazioni Le limitazioni seguenti sono relative a AWS Glue 4.0: • Il machine learning e le trasformazioni di informazioni di identificazione personale (PII) di AWS Glue non sono ancora disponibili in AWS Glue 4.0. Per ulteriori informazioni sulla migrazione a AWS Glue versione 4.0, consulta Migrazione dei processi AWS Glue per Spark ad AWS Glue versione 4.0.

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
	Versioni dell'ambiente Ray • Ray 2.4.0 Python 3.9	Crea ed esegui applicazioni Python distribuite con AWS Glue for Ray. • Supporta la distribuzione dei dati Ray-2.4.0 (`ray[data]`) con Python 3.9. Per ulteriori informazioni su questa versione di Ray, vedete Ray-2.4.0 nel repository Ray. GitHub • Supporta l'installazione di librerie Python aggiuntive nell'ambiente di runtime `Ray2.4`. Per ulteriori informazioni, consulta the section called “Moduli Python aggiuntivi per i processi Ray”. • Integra log e metriche di Ray Jobs con Amazon. CloudWatch Per ulteriori informazioni, consultare the section called “Risoluzione degli errori relativi ai processi Ray” e the section called “Parametri dei processi Ray”. • Aggrega e visualizza le metriche per i lavori Ray in AWS Glue Studio, su ogni pagina di esecuzione del lavoro.

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
		• Supporta la distribuzione di file in ogni directory di lavoro del cluster, il riversamento di oggetti dall'archivio di oggetti Ray ad Amazon S3 e il controllo del numero minimo di nodi worker allocati al processo Ray. Per ulteriori informazioni, consulta the section called “Parametri dei processi Ray”. Limitazioni sui lavori Ray nella versione 4.0 AWS Glue • AWS Glue le sessioni interattive per Ray rimangono disponibili in anteprima per questa versione. • AWS Glue l'integrazione di for Ray con Amazon VPC non è attualmente disponibile. Le risorse in un VPC in non AWS saranno accessibili senza un percorso pubblico. Per ulteriori informazioni sull'util izzo AWS Glue con Amazon VPC, consulta. the section called “Endpoint VPC (AWS PrivateLink)”

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
		• AWS Glue for Ray è disponibile negli Stati Uniti orientali (Virginia settentrionale), Stati Uniti orientali (Ohio), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo) ed Europa (Irlanda).

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità

AWS Glue3.0	• Spark 3.1.1
	• Python 3.7
		Oltre all'aggiornamento del motore Spark a 3.0, questa versione di AWS Glue presenta ottimizzazioni e aggiornamenti integrati, ad esempio:
		• Creazione della libreria ETL di AWS Glue su Spark 3.0, che è una release principale per Spark.
		• I processi di streaming sono supportati su AWS Glue 3.0.
		• Include nuove ottimizza zioni del runtime di Spark AWS Glue per prestazioni e affidabilità:
		  • Elaborazione colonnare in memoria più veloce basata su Apache Arrow per la lettura dei dati CSV.
		  • Esecuzione basata su SIMD per letture vettorizz ate con dati CSV.
		  • L'aggiornamento Spark include anche ulteriori ottimizzazioni sviluppate su Amazon EMR.
		  • EMRFS aggiornato da 2.38 a 2.46, con l'abilita zione di nuove caratteri stiche e correzioni di bug

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
		per l'accesso ad Amazon S3. • Sono state aggiornat e diverse dipendenze necessarie per la nuova versione di Spark. Per informazioni, consulta Appendice A: aggiornamenti notevoli delle dipendenze. • Driver JDBC aggiornati per le nostre origini dati supportate in modo nativo. Per informazioni, consulta Appendice B: aggiornamenti dei driver JDBC. Limitazioni Le limitazioni seguenti sono relative a AWS Glue 3.0: • Le trasformazioni basate su machine learning di AWS Glue non sono ancora disponibili in AWS Glue 3.0. • Alcuni connettori Spark personalizzati non funzionan o con AWS Glue 3.0 se dipendono da Spark 2.4 e non sono compatibili con Spark 3.1.

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
		Per ulteriori informazioni sulla migrazione a AWS Glue versione 3.0, consulta Migrazione dei processi AWS Glue per Spark ad AWS Glue versione 3.0.

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
AWS Glue2.0 (obsoleto, fine del supporto)	• Spark 2.4.3 • Python 3.7	Oltre alle caratteristiche fornite in AWS Glue versione 1.0, AWS Glue versione 2.0 fornisce inoltre: • Un'infrastruttura aggiornata per l'esecuzione dei processi ETL di Apache Spark in AWS Glue con tempi di avvio ridotti. • La registrazione di default è ora in tempo reale, con flussi separati per driver ed esecutori, e contiene output ed errori. • Supporto per la specifica di moduli Python o versioni diverse aggiuntivi a livello di processo. **Note** AWS Glue versione 2.0 differisce da AWS Glue versione 1.0 per alcune dipendenze e versioni dovute a modifiche a livello di architettura. Convalida i processi AWS Glue prima di eseguire la migrazione tra le

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
		versioni principali di AWS Glue. Per ulteriori informazioni sulle funzionalità e le limitazio ni di AWS Glue versione 2.0, consulta Esecuzione di processi ETL Spark con tempi di avvio ridotti.

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
AWS Glue1.0 (obsoleta, fine del supporto)	• Spark 2.4.3 • Python 2.7 • Python 3.6	Puoi mantenere i segnalibri dei processi per i formati Parquet e ORC nei processi AWS Glue ETL (utilizzando AWS Glue versione 1.0). In precedenza, era possibile creare segnalibri solo di formati di origine Amazon S3 comuni come JSON, CSV, Apache Avro e XML nei processi AWS Glue ETL. Quando imposti le opzioni di formato per gli ingressi e le uscite ETL, puoi specificare di utilizzare il formato di lettura/scrittura Apache Avro 1.8 per supportare la lettura e la scrittura del tipo logico Avro (usando AWS Glue versione 1.0). In precedenza, era supportata solo la versione 1.7 del formato di lettura/scrittura Avro. Il tipo di connessione DynamoDB supporta un'opzione di scrittura (utilizzando AWS Glue versione 1.0). Limitazioni Le limitazioni seguenti sono relative a AWS Glue 1.0:

Versione AWS Glue	Versioni dell'ambiente di runtime supportate	Modifiche della funzionalità
		• Le versioni 0.9 e 1.0 di AWS Glue non sono disponibili nelle Regioni Asia Pacifico (Giacarta) (`ap-southeast-3`), Medio Oriente (Emirati Arabi Uniti) (`me-central-1`) o altre nuove Regioni in futuro.
AWS Glue0.9 (obsoleta, fine del supporto)	• Spark 2.2.1 • Python 2.7	I processi che sono stati creati senza specificare una versione di AWS Glue utilizzeranno AWS Glue 0.9 per impostazione predefinita. Limitazioni Le limitazioni seguenti sono relative a AWS Glue 0.9: • Le versioni 0.9 e 1.0 di AWS Glue non sono disponibili nelle Regioni Asia Pacifico (Giacarta) (`ap-southeast-3`), Medio Oriente (Emirati Arabi Uniti) (`me-central-1`) o altre nuove Regioni in futuro.

Esecuzione di processi ETL Spark con tempi di avvio ridotti

AWS Glue versione 2.0 e successive forniscono un'infrastruttura aggiornata per l'esecuzione dei processi Apache Spark ETL (estrazione, trasformazione e caricamento) in AWS Glue con tempi di avvio ridotti. Con i tempi di attesa ridotti, gli ingegneri dei dati possono essere più produttivi e

aumentare l'interattività con AWS Glue. La varianza ridotta dei tempi di inizio dei processi può aiutarti a soddisfare o superare i tuoi SLA per rendere disponibili i dati per l'analisi dei dati.

Per usare questa caratteristica con i processi ETL di AWS Glue, scegli **`2.0`** o una versione successiva per la `Glue version` durante la creazione dei processi.

Argomenti

Nuove caratteristiche supportate

In questa sezione vengono descritte le nuove funzionalità supportate con AWS Glue versione 2.0 e successive.

Supporto per la specifica di moduli Python aggiuntivi a livello di processo

AWS Glue versione 2.0 e successive consente inoltre di fornire moduli Python aggiuntivi o versioni diverse a livello di processo. Puoi utilizzare l'opzione `--additional-python-modules` con un elenco di moduli Python separati da virgole per aggiungere un nuovo modulo o modificare la versione di un modulo esistente.

Ad esempio, per aggiornare o aggiungere un nuovo modulo `scikit-learn` usa la seguente chiave-valore: `"--additional-python-modules", "scikit-learn==0.21.3"`.

Inoltre, all'interno dell'opzione `--additional-python-modules` puoi specificare un percorso Amazon S3 per un modulo ruota Python. Ad esempio:

```
--additional-python-modules s3://aws-glue-native-spark/tests/j4.2/ephem-3.7.7.1-cp37-cp37m-linux_x86_64.whl,s3://aws-glue-native-spark/tests/j4.2/fbprophet-0.6-py3-none-any.whl,scikit-learn==0.21.3
```

AWS Glue utilizza Python Package Installer (pip3) per installare i moduli aggiuntivi. È possibile passare le opzioni aggiuntive specificate da `python-modules-installer-option` a pip3 per l'installazione dei moduli. Verranno applicate eventuali incompatibilità o limitazioni da pip3.

Moduli Python già forniti in AWS Glue versione 2.0

AWS Glue versione 2.0 supporta i seguenti moduli python per impostazione predefinita:

- setuptools—45.2.0
- subprocess32—3.5.4
- ptvsd—4.3.2
- pydevd—1.9.0
- PyMySQL—0.9.3
- docutils—0.15.2
- jmespath—0.9.4
- six—1.14.0
- python_dateutil—2.8.1
- urllib3—1.25.8
- botocore—1.15.4
- s3transfer—0.3.3
- boto3—1.12.4
- certifi—2019.11.28
- chardet—3.0.4
- idna—2.9
- requests—2.23.0
- pyparsing—2.4.6
- enum34—1.1.9
- pytz—2019.3
- numpy—1.18.1
- cycler—0.10.0
- kiwisolver—1.1.0
- scipy—1.4.1
- pandas—1.0.1
- pyarrow—0.16.0
- matplotlib—3.1.3
- pyhocon—0.3.54
- mpmath—1.1.0

- sympy—1.5.1
- patsy—0.5.1
- statsmodels—0.11.1
- fsspec—0.6.2
- s3fs—0.4.0
- Cython—0.29.15
- joblib—0.14.1
- pmdarima—1.5.3
- scikit-learn—0.22.1
- tbats—1.0.9

Comportamento di registrazione

AWS Glue versione 2.0 e successive supporta diversi comportamenti di registrazione di default. Le differenze includono:

- La registrazione avviene in tempo reale.
- Esistono flussi separati per driver ed executor.
- Per ogni driver ed executor ci sono due flussi, il flusso di output e il flusso degli errori.

Flussi di driver ed executor

I flussi dei driver sono identificati dall'ID di esecuzione del processo. I flussi degli executor sono identificati da <*id esecuzione*>_<*id attività executor*> del processo. Ad esempio:

- `"logStreamName": "jr_8255308b426fff1b4e09e00e0bd5612b1b4ec848d7884cebe61ed33a31789..._g-f65f617bd31d54bd94482af755b6cdf464542..."`

Flusso di output ed errori

Il flusso di output presenta l'output standard (stdout) del codice. Il flusso degli errori presenta messaggi di log del codice/della libreria.

- Flussi di log:

- I flussi di log dei driver presentano <*jr*>, in cui <*jr*> è l'ID di esecuzione del processo.
- I flussi di log degli executor presentano <*jr*>_<*g*>, in cui <*g*> è l'ID attività per l'executor. È possibile cercare l'ID attività dell'executor nel log degli errori del driver.

I gruppi di log predefiniti per AWS Glue versione 2.0 sono i seguenti:

- `/aws-glue/jobs/logs/output` per l'output
- `/aws-glue/jobs/logs/error` per gli errori

Quando viene fornita una configurazione di sicurezza, i nomi dei gruppi di log vengono modificati in:

- `/aws-glue/jobs/<security configuration>-role/<Role Name>/output`
- `/aws-glue/jobs/<security configuration>-role/<Role Name>/error`

Nella console, il collegamento Logs (Log) punta al gruppo di log di output e il collegamento Error (Errore) punta al gruppo di log degli errori. Quando la registrazione continua è abilitata, il collegamento Logs (Log) punta al gruppo di log continuo e il collegamento Output punta al gruppo di log di output.

Regole di registrazione

> **Note**
>
> Il nome del gruppo di log di default per la registrazione continua è `/aws-glue/jobs/logs-v2`.

In AWS Glue versione 2.0 e successive, la registrazione continua ha lo stesso comportamento di AWS Glue versione 1.0:

- Gruppo di log di default: `/aws-glue/jobs/logs-v2`
- Flusso di log del driver: <*jr*>-driver
- Flusso di log dell'executor: <*jr*>-<*executor ID*>

  Il nome del gruppo di log può essere modificato impostando `--continuous-log-logGroupName`

Il nome dei flussi di log può essere preceduto impostando `--continous-log-logStreamPrefix`

Funzionalità non supportate

Le seguenti funzioni di AWS Glue non sono supportate:

- Endpoint di sviluppo
- AWS Glue versione 2.0 e successive non viene eseguito su Apache YARN, quindi le impostazioni di YARN non si applicano
- AWS Glue versione 2.0 e successive non dispone di un file di sistema distribuito Hadoop (HDFS)
- AWS Glue versione 2.0 e successive non utilizza l'allocazione dinamica, quindi i parametri ExecutorAllocationManager non sono disponibili
- Per i processi di AWS Glue versione 2.0 e successive, è possibile specificare il numero di dipendenti e il tipo di dipendente, ma non una `maxCapacity`.
- AWS Glue versione 2.0 e successive non supporta `s3n` pronto all'uso. Si consiglia di utilizzare `s3` o `s3a`. Se i processi devono usare `s3n` per qualsiasi motivo, è possibile passare il seguente argomento aggiuntivo:

```
--conf spark.hadoop.fs.s3n.impl=com.amazon.ws.emr.hadoop.fs.EmrFileSystem
```

Migrazione dei processi AWS Glue per Spark ad AWS Glue versione 3.0

In questo argomento vengono descritte le modifiche tra AWS Glue versioni 0.9, 1.0, 2.0 e 3.0 per permettere la migrazione delle applicazioni Spark e dei processi ETL ad AWS Glue 3.0.

Per usare questa caratteristica con i processi ETL di AWS Glue, scegli **`3.0`** per la `Glue version` durante la creazione dei processi.

Argomenti

- Migrazione da AWS Glue 1.0 ad AWS Glue 3.0
- Migrazione da AWS Glue 2.0 a AWS Glue 3.0
- Appendice A: aggiornamenti notevoli delle dipendenze
- Appendice B: aggiornamenti dei driver JDBC

Nuove caratteristiche supportate

In questa sezione vengono descritte le nuove caratteristiche e i vantaggi di AWS Glue versione 3.0.

- Si basa su Apache Spark 3.1.1, che presenta ottimizzazioni da Spark open source e sviluppate da AWS Glue e servizi EMR, come esecuzione adattiva delle query, lettori vettorizzati e shuffle e coalescenza delle partizioni ottimizzati.
- Driver JDBC aggiornati per tutte le fonti native Glue, inclusi MySQL, Microsoft SQL Server, Oracle, PostgreSQL, MongoDB e le librerie e le dipendenze Spark aggiornate introdotte da Spark 3.1.1.
- Accesso Amazon S3 ottimizzato con EMRFS aggiornato e committer di output ottimizzati per Amazon S3 abilitati per impostazione predefinita.
- Accesso ottimizzato al catalogo dati con indici delle partizioni, predicati push down, elenco delle partizioni e client metastore Hive aggiornato.
- Integrazione con Lake Formation per tabelle di catalogo governate con filtraggio a livello di cella e transazioni data lake.
- Esperienza dell'interfaccia utente Spark migliorata con Spark 3.1.1 con nuovi parametri di memoria dell'executor e parametri di streaming strutturati di Spark.
- Ridotta latenza di avvio, con miglioramento dei tempi complessivi di completamento del processo e dell'interattività, similmente ad AWS Glue 2.0.
- I processi Spark vengono fatturati in incrementi di 1 secondo con una durata minima di fatturazione 10 volte inferiore, da un minimo di 10 minuti a un minimo di 1 minuto, similmente ad AWS Glue 2.0.

Operazioni per eseguire la migrazione ad AWS Glue 3.0

Per i processi esistenti, modifica la `Glue version` dalla versione precedente a `Glue 3.0` nella configurazione del processo.

- Nella console, scegli `Spark 3.1, Python 3 (Glue Version 3.0) or Spark 3.1, Scala 2 (Glue Version 3.0)` in `Glue version`.

- In AWS Glue Studio, scegli `Glue 3.0 - Supports spark 3.1, Scala 2, Python 3` in `Glue version`.
- Nell'API, scegli **`3.0`** nel parametro `GlueVersion` nell'API `UpdateJob`.

Per i nuovi processi, scegli `Glue 3.0` al momento della creazione.

- Nella console, scegli `Spark 3.1, Python 3 (Glue Version 3.0) or Spark 3.1, Scala 2 (Glue Version 3.0)` in `Glue version`.
- In AWS Glue Studio, scegli `Glue 3.0 - Supports spark 3.1, Scala 2, Python 3` in `Glue version`.
- Nell'API, scegli **`3.0`** nel parametro `GlueVersion` nell'API `CreateJob`.

Per visualizzare i log eventi Spark di AWS Glue 3.0, avvia un server di cronologia Spark aggiornato per Glue 3.0 utilizzando CloudFormation o Docker.

Elenco di controllo per la migrazione

Esamina questo elenco di controllo per la migrazione.

- Il processo dipende da HDFS? In caso affermativo, prova a sostituire HDFS con S3.
  - Cerca il percorso del file system che inizia con `hdfs://` o `/` come percorso DFS nel codice di script del processo.
  - Controlla che il file system di default non sia configurato con HDFS. Se è configurato in modo esplicito, è necessario rimuovere la configurazione `fs.defaultFS`.
  - Controlla se il processo contiene dei parametri `dfs.*`. Se ne contiene, è necessario verificare che sia corretto disabilitare i parametri.
- Il processo dipende da YARN? In caso affermativo, verifica l'impatto controllando se il processo contiene i seguenti parametri. Se ne contiene, è necessario verificare che sia corretto disabilitare i parametri.
  - `spark.yarn.*`

    Ad esempio:

    ```
    spark.yarn.executor.memoryOverhead
    spark.yarn.driver.memoryOverhead
    spark.yarn.scheduler.reporterThread.maxFailures
    ```

- `yarn.*`

  Ad esempio:

  ```
  yarn.scheduler.maximum-allocation-mb
  yarn.nodemanager.resource.memory-mb
  ```

- Il processo dipende da Spark 2.2.1 o Spark 2.4.3? In caso affermativo, verifica l'impatto controllando se il processo utilizza caratteristiche modificate in Spark 3.1.1.
  - `https://spark.apache.org/docs/latest/sql-migration-guide.html#upgrading-from-spark-sql-22-to-23`

    Ad esempio la funzione `percentile_approx` o la funzione `SparkSession` con `SparkSession.builder.getOrCreate()` quando esiste un `SparkContext`.
  - `https://spark.apache.org/docs/latest/sql-migration-guide.html#upgrading-from-spark-sql-23-to-24`

    Ad esempio la funzione `array_contains` o la funzione `CURRENT_DATE`, `CURRENT_TIMESTAMP` con `spark.sql.caseSensitive=true`.
- I file jar aggiuntivi del processo sono in conflitto in Glue 3.0?
  - Da AWS Glue 0.9/1.0: i file jar aggiuntivi forniti in processi esistenti AWS Glue 0.9/1.0 potrebbero causare conflitti di classpath a causa di dipendenze nuove o aggiornate disponibili in Glue 3.0. È possibile evitare conflitti di classpath in AWS Glue 3.0 con il parametro `--user-jars-first` del processo AWS Glue oppure ombreggiando le dipendenze.
  - Da AWS Glue 2.0 è ancora possibile evitare conflitti di classpath in AWS Glue 3.0 con il parametro `--user-jars-first` del processo AWS Glue oppure ombreggiando le dipendenze.
- Il processo dipende da Scala 2.11?
  - AWS Glue 3.0 utilizza Scala 2.12, quindi è necessario ricreare le librerie con Scala 2.12 se dipendono da Scala 2.11.
- Le librerie Python esterne del processo dipendono da Python 2.7/3.6?
  - Utilizza i parametri `--additional-python-modules` invece di impostare il file egg/wheel/zip nel percorso della libreria Python.
  - Aggiorna le librerie dipendenti da Python 2.7/3.6 a Python 3.7, poiché Spark 3.1.1 ha rimosso il supporto Python 2.7.

Migrazione da AWS Glue 0.9 ad AWS Glue 3.0

Nota le seguenti modifiche durante la migrazione:

- AWS Glue 0.9 utilizza Spark 2.2.1 open source e AWS Glue 3.0 utilizza Spark 3.1.1 ottimizzato per EMR.
  - Diverse modifiche di Spark da sole potrebbero richiedere la revisione degli script per garantire che non si faccia riferimento alle caratteristiche rimosse.
  - Ad esempio, Spark 3.1.1 non abilita le FDU non tipizzate per Scala, ma Spark 2.2 le consente.
- Tutti i processi in AWS Glue 3.0 verranno eseguiti con tempi di avvio significativamente migliorati. I processi Spark verranno fatturati in incrementi di 1 secondo con una durata minima di fatturazione 10 volte inferiore poiché la latenza di avvio passerà da un massimo di 10 minuti a un massimo di 1 minuto.
- Il comportamento di registrazione è cambiato rispetto ad AWS Glue 2.0.
- Diversi aggiornamenti delle dipendenze, evidenziati in Appendice A: aggiornamenti notevoli delle dipendenze.
- Scala viene inoltre aggiornato da 2.11 a 2.12 e Scala 2.12 non è compatibile con Scala 2.11.
- Python 3.7 è anche la versione di default utilizzata per gli script Python, mentre AWS Glue 0.9 utilizzava solo Python 2.
  - Python 2.7 non è supportato con Spark 3.1.1.
  - È disponibile un nuovo meccanismo di installazione di moduli Python aggiuntivi.
- AWS Glue 3.0 non viene eseguita su Apache YARN, quindi le impostazioni di YARN non si applicano.
- AWS Glue 3.0 non dispone di un file di sistema distribuito Hadoop (HDFS).
- Eventuali file jar supplementari forniti in processi esistenti AWS Glue 0.9 potrebbero causare conflitti di dipendenze, a causa dell'aggiornamento di diverse dipendenze in 3.0 da 0.9. È possibile evitare conflitti di dipendenze in AWS Glue 3.0 con il parametro `--user-jars-first` del processo AWS Glue.
- AWS Glue 3.0 non utilizza l'allocazione dinamica, quindi i parametri ExecutorAllocationManager non sono disponibili.
- Nei processi di AWS Glue versione 3.0, è possibile specificare il numero di dipendenti e il tipo di dipendente, ma non una `maxCapacity`.
- AWS Glue 3.0 non supporta ancora le trasformazioni basate su machine learning.
- AWS Glue 3.0 non supporta ancora gli endpoint di sviluppo.

Consulta la documentazione relativa alla migrazione di Spark:

- vedi Aggiornamento da Spark SQL 2.2 a 2.3
- vedi Aggiornamento da Spark SQL 2.3 a 2.4
- vedi Aggiornamento da Spark SQL 2.4 a 3.0
- vedi Aggiornamento da Spark SQL 3.0 a 3.1
- vedi Cambiamenti nel comportamento Datetime previsti da Spark 3.0.

Migrazione da AWS Glue 1.0 ad AWS Glue 3.0

Nota le seguenti modifiche durante la migrazione:

- AWS Glue 1.0 utilizza Spark 2.4 open source e AWS Glue 3.0 utilizza Spark 3.1.1 ottimizzato per EMR.
  - Diverse modifiche di Spark da sole potrebbero richiedere la revisione degli script per garantire che non si faccia riferimento alle caratteristiche rimosse.
  - Ad esempio, Spark 3.1.1 non abilita le FDU non tipizzate per Scala, ma Spark 2.4 le consente.
- Tutti i processi in AWS Glue 3.0 verranno eseguiti con tempi di avvio significativamente migliorati. I processi Spark verranno fatturati in incrementi di 1 secondo con una durata minima di fatturazione 10 volte inferiore poiché la latenza di avvio passerà da un massimo di 10 minuti a un massimo di 1 minuto.
- Il comportamento di registrazione è cambiato rispetto ad AWS Glue 2.0.
- Diversi aggiornamenti delle dipendenze, evidenziati in
- Scala viene inoltre aggiornato da 2.11 a 2.12 e Scala 2.12 non è compatibile con Scala 2.11.
- Python 3.7 è anche la versione di default utilizzata per gli script Python, mentre AWS Glue 0.9 utilizzava solo Python 2.
  - Python 2.7 non è supportato con Spark 3.1.1.
  - È disponibile un nuovo meccanismo di installazione di moduli Python aggiuntivi.
- AWS Glue 3.0 non viene eseguita su Apache YARN, quindi le impostazioni di YARN non si applicano.
- AWS Glue 3.0 non dispone di un file di sistema distribuito Hadoop (HDFS).
- Eventuali file jar supplementari forniti in processi esistenti AWS Glue 1.0 potrebbero causare conflitti di dipendenze, a causa dell'aggiornamento di diverse dipendenze in 3.0 da 1.0. È possibile

evitare conflitti di dipendenze in AWS Glue 3.0 con il parametro `--user-jars-first` del processo AWS Glue.

- AWS Glue 3.0 non utilizza l'allocazione dinamica, quindi i parametri ExecutorAllocationManager non sono disponibili.
- Nei processi di AWS Glue versione 3.0, è possibile specificare il numero di dipendenti e il tipo di dipendente, ma non una `maxCapacity`.
- AWS Glue 3.0 non supporta ancora le trasformazioni basate su machine learning.
- AWS Glue 3.0 non supporta ancora gli endpoint di sviluppo.

Consulta la documentazione relativa alla migrazione di Spark:

- vedi Aggiornamento da Spark SQL 2.4 a 3.0
- vedi Cambiamenti nel comportamento Datetime previsti da Spark 3.0.

Migrazione da AWS Glue 2.0 a AWS Glue 3.0

Nota le seguenti modifiche durante la migrazione:

- Tutti i parametri di processo e le funzioni principali esistenti in AWS Glue 2.0 esisteranno in AWS Glue 3.0.
  - In AWS Glue 3.0, il committer ottimizzato EMRFS S3 per la scrittura dei dati Parquet in Amazon S3 è abilitato di default. Tuttavia, puoi disabilitarlo impostando `--enable-s3-parquet-optimized-committer` a `false`.
- AWS Glue 2.0 utilizza Spark 2.4 open source e AWS Glue 3.0 utilizza Spark 3.1.1 ottimizzato per EMR.
  - Diverse modifiche di Spark da sole potrebbero richiedere la revisione degli script per garantire che non si faccia riferimento alle caratteristiche rimosse.
  - Ad esempio, Spark 3.1.1 non abilita le FDU non tipizzate per Scala, ma Spark 2.4 le consente.
- AWS Glue 3.0 include anche un aggiornamento a EMRFS, driver JDBC aggiornati e include ulteriori ottimizzazioni su Spark stesso fornite da AWS Glue.
- Tutti i processi in AWS Glue 3.0 verranno eseguiti con tempi di avvio significativamente migliorati. I processi Spark verranno fatturati in incrementi di 1 secondo con una durata minima di fatturazione 10 volte inferiore poiché la latenza di avvio passerà da un massimo di 10 minuti a un massimo di 1 minuto.

- Python 2.7 non è supportato con Spark 3.1.1.
- Diversi aggiornamenti delle dipendenze, evidenziati in Appendice A: aggiornamenti notevoli delle dipendenze.
- Scala viene inoltre aggiornato da 2.11 a 2.12 e Scala 2.12 non è compatibile con Scala 2.11.
- Eventuali file jar supplementari forniti in processi esistenti AWS Glue 2.0 potrebbero causare conflitti di dipendenze, a causa dell'aggiornamento di diverse dipendenze in 3.0 da 2.0. È possibile evitare conflitti di dipendenze in AWS Glue 3.0 con il parametro `--user-jars-first` del processo AWS Glue.
- AWS Glue 3.0 ha un parallelismo di processo Spark diverso per la configurazione del driver/esecutore, rispetto a AWS Glue 2.0, migliora le prestazioni e utilizza al meglio le risorse disponibili. Sia `spark.driver.cores` che `spark.executor.cores` sono configurati in base al numero di core su AWS Glue 3.0 (4 sullo standard e sul dipendente `G.1X` e 8 sul dipendente `G.2X`). Queste configurazioni non modificano il tipo di dipendente o l'hardware per il processo AWS Glue. È possibile utilizzare queste configurazioni per calcolare il numero di partizioni o divisioni che corrispondono al parallelismo delle attività Spark nell'applicazione Spark.

  In generale, i processi avranno prestazioni simili o migliorate rispetto ad AWS Glue versione 2.0. Se i processi vengono eseguiti più lentamente, è possibile incrementare il parallelismo delle attività trasmettendo il seguente argomento del processo:

  - chiave: `--executor-cores` valore: **<***numero desiderato di attività che possono essere eseguite in parallelo***>**
  - Il valore non deve superare il doppio del numero di vCPU del tipo di worker, ossia 8 su `G.1X`, 16 su `G.2X`, 32 su `G.4X` e 64 su `G.8X`. È necessario prestare attenzione durante l'aggiornamento di questa configurazione in quanto potrebbe influire sulle prestazioni del processo; l'incremento del parallelismo, infatti, esercita pressione sulla memoria e sul disco, oltre a limitare i sistemi di origine e destinazione.
- AWS Glue 3.0 utilizza Spark 3.1, che modifica il comportamento di carico/salvataggio dei timestamp da/verso i file parquet. Per ulteriori dettagli, consulta Aggiornamento da Spark SQL 3.0 a 3.1.

  Noi suggeriamo di impostare i seguenti parametri durante la lettura/scrittura di dati del parquet che contengono colonne timestamp. L'impostazione di tali parametri può risolvere il problema di incompatibilità del calendario che si verifica durante l'aggiornamento da Spark 2 a Spark 3, sia per AWS Glue Dynamic Frame che per Spark Data Frame. Utilizzare l'opzione CORRECTED per leggere il valore datetime così com'è e l'opzione LEGACY per cambiare la base dei valori datetime in relazione alla differenza di calendario durante la lettura.

```
- Key: --conf
- Value: spark.sql.legacy.parquet.int96RebaseModeInRead=[CORRECTED|LEGACY] --
conf spark.sql.legacy.parquet.int96RebaseModeInWrite=[CORRECTED|LEGACY] --conf
 spark.sql.legacy.parquet.datetimeRebaseModeInRead=[CORRECTED|LEGACY]
```

Consulta la documentazione relativa alla migrazione di Spark:

- vedi Aggiornamento da Spark SQL 2.4 a 3.0
- vedi Cambiamenti nel comportamento Datetime previsti da Spark 3.0.

Appendice A: aggiornamenti notevoli delle dipendenze

Di seguito sono riportati gli aggiornamenti delle dipendenze:

Dipendenza	Versione in AWS Glue 0.9	Versione in AWS Glue 1.0	Versione in AWS Glue 2.0	Versione in AWS Glue 3.0
Spark	2.2.1	2.4.3	2.4.3	3.1.1-amzn-0
Hadoop	2.7.3-amzn-6	2.8.5-amzn-1	2.8.5-amzn-5	3.2.1-amzn-3
Scala	2.11	2.11	2.11	2.12
Jackson	2.7.x	2.7.x	2.7.x	2.10.x
Hive	1.2	1.2	1.2	2.3.7-amzn-4
EMRFS	2.20.0	2.30.0	2.38.0	2.46.0
Json4s	3.2.x	3.5.x	3.5.x	3.6.6
Arrow	N/D	0.10.0	0.10.0	2.0.0
Client del catalogo AWS Glue	N/D	N/D	1.10.0	3.0.0

Appendice B: aggiornamenti dei driver JDBC

Di seguito sono riportati gli aggiornamenti dei driver JDBC:

Driver	Versione del driver JDBC nelle precedenti versioni di AWS Glue	Versione del driver JDBC in AWS Glue 3.0
MySQL	5.1	8.0.23
Microsoft SQL Server	6.1.0	7.0.0
Database Oracle	11.2	21.1
PostgreSQL	42.1.0	42.2.18
MongoDB	2.0.0	4.0.0

Migrazione dei processi AWS Glue per Spark ad AWS Glue versione 4.0

In questo argomento vengono descritte le modifiche tra AWS Glue versioni 0.9, 1.0, 2.0 e 3.0 per permettere la migrazione delle applicazioni Spark e dei processi ETL a AWS Glue 4.0. Descrive inoltre le funzionalità in AWS Glue 4.0 e i vantaggi del suo utilizzo.

Per usare questa caratteristica con i processi ETL di AWS Glue, scegli **`4.0`** per la `Glue version` durante la creazione dei processi.

Argomenti

- Nuove caratteristiche supportate
- Operazioni per eseguire la migrazione ad AWS Glue 4.0
- Elenco di controllo della migrazione
- Migrazione da AWS Glue 3.0 a AWS Glue 4.0
- Migrazione da AWS Glue 2.0 a AWS Glue 4.0
- Migrazione da AWS Glue 1.0 ad AWS Glue 4.0
- Migrazione da AWS Glue 0.9 ad AWS Glue 4.0
- Migrazione di connettori e driver JDBC per AWS Glue 4.0

- Appendice A: Aggiornamenti importanti delle dipendenze
- Appendice B: aggiornamenti dei driver JDBC
- Appendice C: Aggiornamenti dei connettori

Nuove caratteristiche supportate

In questa sezione vengono descritte le nuove funzionalità e i vantaggi di AWS Glue versione 4.0.

- Si basa su Apache Spark 3.3.0, ma presenta ottimizzazioni in AWS Glue e Amazon EMR, come esecuzioni adattive delle query, lettori vettorizzati e shuffle e coalescenza delle partizioni ottimizzati.
- Driver JDBC aggiornati per tutte le origini native di AWS Glue, inclusi MySQL, Microsoft SQL Server, Oracle, PostgreSQL, MongoDB e le librerie e le dipendenze Spark aggiornate introdotte da Spark 3.3.0.
- Aggiornato con un nuovo connettore Amazon Redshift e driver JDBC.
- Accesso Amazon S3 ottimizzato con EMRFS aggiornato e committer di output ottimizzati per Amazon S3 abilitati per impostazione predefinita.
- Accesso ottimizzato al catalogo dati con indici delle partizioni, predicati pushdown, elenco delle partizioni e un client metastore Hive aggiornato.
- Integrazione con Lake Formation per tabelle di catalogo governate con filtraggio a livello di cella e transazioni data lake.
- Ridotta la latenza di avvio per migliorare i tempi complessivi di completamento dei processi e dell'interattività.
- I processi Spark vengono fatturati in incrementi di 1 secondo con una durata minima di fatturazione 10 volte inferiore, da un minimo di 10 minuti a un minimo di 1 minuto.
- Supporto nativo per framework open data lake con Apache Hudi, Delta Lake e Apache Iceberg.
- Supporto nativo per il Cloud Shuffle Storage Plugin basato su Amazon S3 (un plug-in Apache Spark) per utilizzare Amazon S3 per lo shuffling e la capacità di archiviazione elastica.

Miglioramenti principali da Spark 3.1.1 a Spark 3.3.0

Nota i seguenti miglioramenti:

- Filtraggio di runtime a livello di riga (SPARK-32268).
- Miglioramenti ANSI (SPARK-38860).

- Miglioramenti ai messaggi di errore (SPARK-38781).
- Supporto dei tipi complessi per il lettore vettorializzato Parquet (SPARK-34863).
- Supporto di metadati di file nascosti per Spark SQL (SPARK-37273).
- Fornito un profiler per le UDF di Python/Pandas (SPARK-37443).
- Introduce Trigger.availableNow per eseguire query di streaming come Trigger.Once in più batch (SPARK-36533).
- Funzionalità di pushdown Datasource V2 più complete (SPARK-38788).
- Migrazione da log4j 1 a log4j 2 (SPARK-37814).

Altre modifiche importanti

Nota le seguenti modifiche:

- Modifiche importanti
  - Elimina i riferimenti al supporto per Python 3.6 in docs e Python/docs (SPARK-36977).
  - Rimuove l'hack di tuple specificate sostituendo il pickle integrato con cloudpickle (SPARK-32079).
  - Porta la versione minima di Pandas a 1.0.5 (SPARK-37465).

Operazioni per eseguire la migrazione ad AWS Glue 4.0

Per i processi esistenti, modifica la `Glue version` dalla versione precedente a `Glue 4.0` nella configurazione del processo.

- In AWS Glue Studio, scegli `Glue 4.0 - Supports Spark 3.3, Scala 2, Python 3` in `Glue version`.
- Nell'API, scegli **`4.0`** nel parametro `GlueVersion` nell'operazione API `UpdateJob`.

Per i nuovi processi, scegli `Glue 4.0` al momento della creazione.

- Nella console, scegli `Spark 3.3, Python 3 (Glue Version 4.0) or Spark 3.3, Scala 2 (Glue Version 3.0)` in `Glue version`.
- In AWS Glue Studio, scegli `Glue 4.0 - Supports Spark 3.3, Scala 2, Python 3` in `Glue version`.
- Nell'API, scegli **`4.0`** nel parametro `GlueVersion` nell'operazione API `CreateJob`.

Per visualizzare i log di eventi Spark di AWS Glue 4.0 provenienti da AWS Glue 2.0 o versioni precedenti, avvia un server di cronologia Spark aggiornato per AWS Glue 4.0 utilizzando AWS CloudFormation o Docker.

Elenco di controllo della migrazione

Rivedi questo elenco di controllo per la migrazione:

> **Note**
>
> Per gli elementi della lista di controllo relativi a AWS Glue 3.0, consulta Elenco di controllo per la migrazione.

- Le librerie Python esterne del processo dipendono da Python 2.7/3.6?
  - Aggiorna le librerie dipendenti da Python 2.7/3.6 a Python 3.10, poiché Spark 3.3.0 ha rimosso il supporto per Python 2.7 e 3.6.

Migrazione da AWS Glue 3.0 a AWS Glue 4.0

Nota le seguenti modifiche durante la migrazione:

- Tutti i parametri dei processi e le funzionalità principali esistenti in AWS Glue 3.0 saranno presenti in AWS Glue 4.0.
- AWS Glue 3.0 utilizza Spark 3.1.1 ottimizzato per Amazon EMR e AWS Glue 4.0 utilizza Spark 3.3.0 ottimizzato per Amazon EMR.

  Diverse modifiche di Spark da sole potrebbero richiedere la revisione degli script per garantire che non si faccia riferimento alle funzionalità rimosse.
- AWS Glue 4.0 include anche un aggiornamento a EMRFS e Hadoop. Per la versione specifica, consulta Appendice A: Aggiornamenti importanti delle dipendenze.
- L'SDK AWS fornito nei processi ETL è stato aggiornato da 1.11 a 1.12.
- Tutti i processi Python utilizzeranno la versione 3.10 di Python. In precedenza, in AWS Glue 3.0 era usato Python 3.7.

  Di conseguenza, alcuni pymodule pronti all'uso da AWS Glue sono stati aggiornati.
- Log4j è stato aggiornato a Log4j2.

  - Per informazioni sul percorso di migrazione di Log4j2, consulta la documentazione di Log4j.
  - È invece necessario rinominare qualsiasi file log4j.properties personalizzato come file log4j2.properties, con le proprietà log4j2 appropriate.
- Per la migrazione di connettori specifici, consulta Migrazione di connettori e driver JDBC per AWS Glue 4.0.
- L'SDK di crittografia AWS è stato aggiornato da 1.x a 2.x. Sono interessati da questa modifica i processi AWS Glue che utilizzano configurazioni di sicurezza di AWS Glue e i processi dipendenti dalla dipendenza dell'SDK di crittografia AWS forniti in fase di runtime. Consulta le istruzioni relative alla migrazione dei processi AWS Glue.

  Puoi aggiornare in sicurezza un processo AWS Glue 2.0/3.0 a un processo AWS Glue 4.0 perché AWS Glue 2.0/3.0 contiene già la versione bridge dell'SDK di crittografia AWS.

Consulta la documentazione relativa alla migrazione di Spark:

- Aggiornamento da Spark SQL 3.1 a 3.2
- Aggiornamento da Spark SQL 3.2 a 3.3

Migrazione da AWS Glue 2.0 a AWS Glue 4.0

Nota le seguenti modifiche durante la migrazione:

> **Note**
>
> Per le fasi di migrazione relative a AWS Glue 3.0, consulta Migrazione da AWS Glue 3.0 a AWS Glue 4.0.

- Tutti i parametri di processo e le funzionalità principali esistenti in AWS Glue 2.0 saranno presenti in AWS Glue 4.0.
- Il committer ottimizzato per S3 EMRFS per la scrittura di dati Parquet in Amazon S3 è abilitato per impostazione predefinita a cominciare da AWS Glue 3.0. Tuttavia, puoi disabilitarlo impostando `--enable-s3-parquet-optimized-committer` a `false`.
- AWS Glue 2.0 utilizza Spark 2.4 open source e AWS Glue 4.0 utilizza Spark 3.3.0 ottimizzato per Amazon EMR.

  - Diverse modifiche di Spark da sole potrebbero richiedere la revisione degli script per garantire che non si faccia riferimento alle funzionalità rimosse.
  - Ad esempio, Spark 3.3.0 non abilita le UDF non tipizzate per Scala, ma Spark 2.4 le consente.
- L'SDK AWS fornito nei processi ETL è stato aggiornato da 1.11 a 1.12.
- AWS Glue 4.0 include anche un aggiornamento a EMRFS, driver JDBC aggiornati e ulteriori ottimizzazioni su Spark stesso fornite da AWS Glue.
- Scala è stato aggiornato da 2.11 a 2.12 e Scala 2.12 non è compatibile con Scala 2.11.
- Python 3.10 è la versione predefinita utilizzata per gli script Python mentre AWS Glue 2.0 utilizzava solo Python 3.7 e 2.7.
  - Python 2.7 non è supportato con Spark 3.3.0. Qualsiasi processo che richieda Python 2 nella sua configurazione avrà esito negativo, con la restituzione di IllegalArgumentException.
  - Un nuovo meccanismo per l'installazione di moduli Python aggiuntivi è disponibile a cominciare da AWS Glue 2.0.
- Diversi aggiornamenti delle dipendenze, evidenziati in Appendice A: Aggiornamenti importanti delle dipendenze.
- Eventuali file JAR supplementari forniti in processi AWS Glue 2.0 esistenti potrebbero causare conflitti di dipendenze a causa degli aggiornamenti di diverse dipendenze in 4.0 da 2.0. È possibile evitare conflitti di dipendenze in AWS Glue 4.0 con il parametro `--user-jars-first` del processo AWS Glue.
- AWS Glue 4.0 utilizza Spark 3.3. A partire da Spark 3.1, è stato apportato un cambiamento nel comportamento di caricamento/salvataggio dei timestamp da/verso i file parquet. Per ulteriori dettagli, consulta Aggiornamento da Spark SQL 3.0 a 3.1.

  Noi suggeriamo di impostare i seguenti parametri durante la lettura/scrittura di dati del parquet che contengono colonne timestamp. L'impostazione di tali parametri può risolvere il problema di incompatibilità del calendario che si verifica durante l'aggiornamento da Spark 2 a Spark 3, sia per AWS Glue Dynamic Frame che per Spark Data Frame. Utilizzare l'opzione CORRECTED per leggere il valore datetime così com'è e l'opzione LEGACY per cambiare la base dei valori datetime in relazione alla differenza di calendario durante la lettura.

```
- Key: --conf
- Value: spark.sql.legacy.parquet.int96RebaseModeInRead=[CORRECTED|LEGACY] --
conf spark.sql.legacy.parquet.int96RebaseModeInWrite=[CORRECTED|LEGACY] --conf
 spark.sql.legacy.parquet.datetimeRebaseModeInRead=[CORRECTED|LEGACY]
```

- Per la migrazione di connettori specifici, consulta Migrazione di connettori e driver JDBC per AWS Glue 4.0.
- L'SDK di crittografia AWS è stato aggiornato da 1.x a 2.x. Sono interessati da questa modifica i processi AWS Glue che utilizzano configurazioni di sicurezza di AWS Glue e i processi dipendenti dalla dipendenza dell'SDK di crittografia AWS forniti in fase di runtime. Consulta le seguenti istruzioni relative alla migrazione del processo AWS Glue:
  - Puoi aggiornare in sicurezza un processo AWS Glue 2.0 a un processo AWS Glue 4.0 perché AWS Glue 2.0 contiene già la versione bridge dell'SDK di crittografia AWS.

Consulta la documentazione relativa alla migrazione di Spark:

- Aggiornamento da Spark SQL 2.4 a 3.0
- Aggiornamento da Spark SQL 3.1 a 3.2
- Aggiornamento da Spark SQL 3.2 a 3.3
- Cambiamenti nel comportamento di Datetime previsti da Spark 3.0.

Migrazione da AWS Glue 1.0 ad AWS Glue 4.0

Nota le seguenti modifiche durante la migrazione:

- AWS Glue 1.0 utilizza Spark 2.4 open source e AWS Glue 4.0 utilizza Spark 3.3.0 ottimizzato per Amazon EMR.
  - Diverse modifiche di Spark da sole potrebbero richiedere la revisione degli script per garantire che non si faccia riferimento alle funzionalità rimosse.
  - Ad esempio, Spark 3.3.0 non abilita le UDF non tipizzate per Scala, ma Spark 2.4 le consente.
- Tutti i processi in AWS Glue 4.0 verranno eseguiti con tempi di avvio significativamente migliorati. I processi Spark verranno fatturati in incrementi di 1 secondo con una durata minima di fatturazione 10 volte inferiore poiché la latenza di avvio passerà da un massimo di 10 minuti a un massimo di 1 minuto.
- Il comportamento di registrazione è cambiato in modo significativo in AWS Glue versione 4.0, in quanto Spark 3.3.0 richiede Log4j2.
- Diversi aggiornamenti delle dipendenze, descritti nell'appendice.
- Scala è stato inoltre aggiornato da 2.11 a 2.12 e Scala 2.12 non è compatibile con Scala 2.11.

- Python 3.10 è anche la versione di predefinita utilizzata per gli script Python, mentre AWS Glue 0.9 utilizzava solo Python 2.

  Python 2.7 non è supportato con Spark 3.3.0. Qualsiasi processo che richieda Python 2 nella sua configurazione avrà esito negativo, con la restituzione di IllegalArgumentException.
- Un nuovo meccanismo per l'installazione di moduli Python aggiuntivi tramite pip è disponibile a cominciare da AWS Glue 2.0. Per ulteriori informazioni, consulta Installazione di moduli Python aggiuntivi con pip in AWS Glue 2.0+.
- AWS Glue 4.0 non viene eseguito su Apache YARN, pertanto le impostazioni di YARN non si applicano.
- AWS Glue 4.0 non dispone di un file system distribuito Hadoop (HDFS).
- Eventuali file JAR supplementari forniti in processi AWS Glue 1.0 esistenti potrebbero causare conflitti di dipendenze a causa degli aggiornamenti di diverse dipendenze in 4.0 da 1.0. Per evitare questo problema, abilitiamo automaticamente AWS Glue 4.0 con il parametro di processo AWS Glue `--user-jars-first`.
- AWS Glue 4.0 supporta il dimensionamento automatico. Pertanto, il parametro ExecutorAllocationManager sarà disponibile quando è abilitato il dimensionamento automatico.
- Nei processi di AWS Glue versione 4.0, è possibile specificare il numero di worker e il tipo di worker, ma non una `maxCapacity`.
- AWS Glue 4.0 non supporta ancora le trasformazioni basate su machine learning.
- Per la migrazione di connettori specifici, consulta Migrazione di connettori e driver JDBC per AWS Glue 4.0.
- L'SDK di crittografia AWS è stato aggiornato da 1.x a 2.x. Sono interessati da questa modifica i processi AWS Glue che utilizzano configurazioni di sicurezza di AWS Glue e i processi dipendenti dalla dipendenza dell'SDK di crittografia AWS forniti in fase di runtime. Consulta le seguenti istruzioni relative alla migrazione del processo AWS Glue.
  - Non è possibile migrare direttamente un processo AWS Glue 0.9/1.0 a un processo AWS Glue 4.0. Questo perché quando si esegue l'aggiornamento diretto alla versione 2.x o successiva e si abilitano immediatamente tutte le nuove funzionalità, l'SDK di crittografia AWS non sarà in grado di decrittare il testo criptato nelle versioni precedenti dell'SDK di crittografia AWS.
  - Per un aggiornamento sicuro, consigliamo innanzitutto di migrare a un processo AWS Glue 2.0/3.0 che contenga la versione bridge dell'SDK di crittografia AWS. Esegui il processo una volta per utilizzare la versione bridge dell'SDK di crittografia AWS.
  - Al termine, sarà possibile migrare in sicurezza il processo AWS Glue 2.0/3.0 a AWS Glue 4.0.

Consulta la documentazione relativa alla migrazione di Spark:

- Aggiornamento da Spark SQL 2.4 a 3.0
- Aggiornamento da Spark SQL 3.0 a 3.1
- Aggiornamento da Spark SQL 3.1 a 3.2
- Aggiornamento da Spark SQL 3.2 a 3.3
- Cambiamenti nel comportamento di Datetime previsti da Spark 3.0.

Migrazione da AWS Glue 0.9 ad AWS Glue 4.0

Nota le seguenti modifiche durante la migrazione:

- AWS Glue 0.9 utilizza Spark 2.2.1 open source e AWS Glue 4.0 utilizza Spark 3.3.0 ottimizzato per Amazon EMR.
  - Diverse modifiche di Spark da sole potrebbero richiedere la revisione degli script per garantire che non si faccia riferimento alle funzionalità rimosse.
  - Ad esempio, Spark 3.3.0 non abilita le UDF non tipizzate per Scala, ma Spark 2.2 le consente.
- Tutti i processi in AWS Glue 4.0 verranno eseguiti con tempi di avvio significativamente migliorati. I processi Spark verranno fatturati in incrementi di 1 secondo con una durata minima di fatturazione 10 volte inferiore poiché la latenza di avvio passerà da un massimo di 10 minuti a un massimo di 1 minuto.
- Il comportamento di registrazione è cambiato in modo significativo rispetto a AWS Glue 4.0; Spark 3.3.0 richiede Log4j2 come indicato qui (https://spark.apache.org/docs/latest/core-migration-guide.html#upgrading-from-core-32-to-33).
- Diversi aggiornamenti delle dipendenze, descritti nell'appendice.
- Scala è stato inoltre aggiornato da 2.11 a 2.12 e Scala 2.12 non è compatibile con Scala 2.11.
- Python 3.10 è anche la versione di predefinita utilizzata per gli script Python, mentre AWS Glue 0.9 utilizzava solo Python 2.
  - Python 2.7 non è supportato con Spark 3.3.0. Qualsiasi processo che richieda Python 2 nella sua configurazione avrà esito negativo, con la restituzione di IllegalArgumentException.
  - È disponibile un nuovo meccanismo di installazione di moduli Python aggiuntivi tramite pip.
- AWS Glue 4.0 non viene eseguito su Apache YARN, pertanto le impostazioni di YARN non si applicano.
- AWS Glue 4.0 non dispone di un file system distribuito Hadoop (HDFS).

- Eventuali file JAR supplementari forniti in processi AWS Glue 0.9 esistenti potrebbero causare conflitti di dipendenze a causa degli aggiornamenti di diverse dipendenze in 3.0 da 0.9. È possibile evitare conflitti di dipendenze in AWS Glue 3.0 con il parametro `--user-jars-first` del processo AWS Glue.
- AWS Glue 4.0 supporta il dimensionamento automatico. Pertanto, il parametro ExecutorAllocationManager sarà disponibile quando è abilitato il dimensionamento automatico.
- Nei processi di AWS Glue versione 4.0, è possibile specificare il numero di worker e il tipo di worker, ma non una `maxCapacity`.
- AWS Glue 4.0 non supporta ancora le trasformazioni basate su machine learning.
- Per la migrazione di connettori specifici, consulta Migrazione di connettori e driver JDBC per AWS Glue 4.0.
- L'SDK di crittografia AWS è stato aggiornato da 1.x a 2.x. Sono interessati da questa modifica i processi AWS Glue che utilizzano configurazioni di sicurezza di AWS Glue e i processi dipendenti dalla dipendenza dell'SDK di crittografia AWS forniti in fase di runtime. Consulta le seguenti istruzioni relative alla migrazione del processo AWS Glue.
  - Non è possibile migrare direttamente un processo AWS Glue 0.9/1.0 a un processo AWS Glue 4.0. Questo perché quando si esegue l'aggiornamento diretto alla versione 2.x o successiva e si abilitano immediatamente tutte le nuove funzionalità, l'SDK di crittografia AWS non sarà in grado di decrittare il testo criptato nelle versioni precedenti dell'SDK di crittografia AWS.
  - Per un aggiornamento sicuro, consigliamo innanzitutto di migrare a un processo AWS Glue 2.0/3.0 che contenga la versione bridge dell'SDK di crittografia AWS. Esegui il processo una volta per utilizzare la versione bridge dell'SDK di crittografia AWS.
  - Al termine, sarà possibile migrare in sicurezza il processo AWS Glue 2.0/3.0 a AWS Glue 4.0.

Consulta la documentazione relativa alla migrazione di Spark:

- Aggiornamento da Spark SQL 2.2 a 2.3
- Aggiornamento da Spark SQL 2.3 a 2.4
- Aggiornamento da Spark SQL 2.4 a 3.0
- Aggiornamento da Spark SQL 3.0 a 3.1
- Aggiornamento da Spark SQL 3.1 a 3.2
- Aggiornamento da Spark SQL 3.2 a 3.3
- Cambiamenti nel comportamento di Datetime previsti da Spark 3.0.

Migrazione di connettori e driver JDBC per AWS Glue 4.0

Per le versioni dei connettori JDBC e data lake che sono state aggiornate, consulta:

- Appendice B: aggiornamenti dei driver JDBC
- Appendice C: Aggiornamenti dei connettori

Hudi

- Miglioramenti al supporto Spark SQL:
  - Tramite il comando `Call Procedure`, viene aggiunto il supporto per l'aggiornamento, il downgrade, il bootstrap, la pulizia e la riparazione. In Spark SQL è possibile utilizzare la sintassi `Create/Drop/Show/Refresh Index`.
  - È stato colmato un divario legato alle prestazioni tra l'utilizzo con Spark DataSource e Spark SQL. Le scritture di Datasource in passato erano più veloci di SQL.
  - Tutti i generatori di chiavi integrati implementano operazioni API specifiche di Spark più performanti.
  - La trasformazione UDF è stata sostituita nell'operazione `insert` di massa con trasformazioni RDD per ridurre i costi di utilizzo di SerDe.
  - Spark SQL con Hudi richiede la specifica di una `primaryKey` da parte di `tblproperites` o più opzioni nell'istruzione SQL. Per le operazioni di aggiornamento ed eliminazione, è necessario anche `preCombineField`.
- Qualsiasi tabella Hudi creata prima della versione 0.10.0 senza una `primaryKey` deve essere creata nuovamente con un campo `primaryKey` a partire dalla versione 0.10.0.

PostgreSQL

- Sono state risolte diverse vulnerabilità (CVE).
- Java 8 è supportato in modo nativo.
- Se il processo utilizza array di array, ad eccezione degli array di byte, questo scenario può essere trattato come array multidimensionali.

MongoDB

- Il connettore MongoDB corrente supporta Spark versione 3.1 o versione successiva e MongoDB versione 4.0 o successiva.
- A causa dell'aggiornamento del connettore, alcuni nomi di proprietà sono cambiati. Ad esempio, il nome della proprietà URI è stato modificato in `connection.uri`. Per ulteriori informazioni sulle opzioni correnti, consulta il blog di MongoDB Spark Connector.
- L'utilizzo di MongoDB 4.0 ospitato da Amazon DocumentDB presenta alcune differenze funzionali. Per ulteriori informazioni, consulta i seguenti argomenti:
  - Differenze funzionali: Amazon DocumentDB e MongoDB
  - API, operazioni e tipi di dati di MongoDB supportati.
- L'opzione "partitioner" è limitata a `ShardedPartitioner`, `PaginateIntoPartitionsPartitioner` e `SinglePartitionPartitioner`. Non può utilizzare `SamplePartitioner` e `PaginateBySizePartitioner` predefiniti per Amazon DocumentDB perché l'operatore stage non supporta l'API MongoDB. Per ulteriori informazioni, consulta API, operazioni e tipi di dati MongoDB supportati.

Delta Lake

- Delta Lake ora supporta i viaggi nel tempo in SQL per interrogare facilmente i dati più vecchi. Con questo aggiornamento, il viaggio nel tempo è ora disponibile sia in Spark SQL che tramite l'API DataFrame. È stato aggiunto il supporto per la versione corrente di TIMESTAMP in SQL.
- Spark 3.3 introduce Trigger.AvailableNow per l'esecuzione di query di streaming come equivalente a `Trigger.Once` per le query in batch. Questo supporto è disponibile anche quando si utilizzano le tabelle Delta come fonte di streaming.
- Supporto per SHOW COLUMNS per restituire l'elenco delle colonne in una tabella.
- Supporto per DESCRIVE DETAIL nelle API DeltaTable di Scala e Python. Recupera informazioni dettagliate su una tabella Delta utilizzando l'API DeltaTable o Spark SQL.
- Supporto per la restituzione di parametri operativi dai comandi SQL Delete, Merge e Update. In precedenza questi comandi SQL restituivano un DataFrame vuoto, ora restituiscono un DataFrame con parametri utili sull'operazione eseguita.
- Ottimizza i miglioramenti in termini di prestazioni:
  - Imposta l'opzione di configurazione `spark.databricks.delta.optimize.repartition.enabled=true` in modo da

utilizzare `repartition(1)` anziché `coalesce(1)` nel comando Optimize per migliorare le prestazioni durante la compattazione di numerosi file di piccole dimensioni.

  - Prestazioni migliorate grazie a un approccio basato su code per parallelizzare i lavori di compattazione.
- Altre modifiche importanti:
  - Supporto per l'utilizzo di variabili nei comandi VACUUM e OPTIMIZE SQL.
  - Miglioramenti per CONVERT TO DELTA con tabelle di catalogo che includono:
    - Completamento automatico dello schema delle partizioni dal catalogo quando non è fornito.
    - Uso delle informazioni sulle partizioni dal catalogo per trovare i file di dati da salvare invece di eseguire una scansione completa della directory. Invece di salvare tutti i file di dati nella directory delle tabelle, verranno salvati solo i file di dati nelle directory delle partizioni attive.
  - Supporto per le letture batch di Change Data Feed (CDF) sulle tabelle abilitate alla mappatura delle colonne quando DROP COLUMN e RENAME COLUMN non sono stati utilizzati. Per ulteriori informazioni, consulta la documentazione di Delta Lake.
  - Miglioramento delle prestazioni dei comandi di aggiornamento abilitando l'eliminazione dello schema nel primo passaggio.

Apache Iceberg

- Sono stati aggiunti diversi miglioramenti delle prestazioni per la pianificazione delle scansioni e le query Spark.
- È stato aggiunto un client di catalogo REST comune che utilizza i commit basati sulle modifiche per risolvere i conflitti di commit lato del servizio.
- La sintassi `AS OF` per le query SQL relative ai viaggi temporali è supportata.
- È stato aggiunto il supporto merge-on-read (unione su lettura) per le query MERGE e UPDATE.
- È stato aggiunto il supporto per riscrivere le partizioni utilizzando l'ordine Z.
- Sono state aggiunte una specifica e un'implementazione per Puffin, un formato per statistiche di grandi dimensioni e blob di indici, come schizzi Theta o filtri bloom.
- Sono state aggiunte nuove interfacce per il consumo incrementale dei dati (scansioni di aggiunta e log delle modifiche).
- È stato aggiunto il supporto per operazioni di massa e letture a intervalli alle interfacce FileIO.
- Sono state aggiunte altre tabelle di metadati per mostrare i file di eliminazione nella struttura dei metadati.

- Il comportamento della tabella di eliminazione è cambiato. In Iceberg 0.13.1, l'esecuzione di `DROP TABLE` rimuove la tabella dal catalogo e ne elimina anche il contenuto. In Iceberg 1.0.0, `DROP TABLE` rimuove solo la tabella dal catalogo. Per eliminare il contenuto della tabella, utilizza `DROP TABLE PURGE`.
- Le letture vettorializzate in Parquet sono abilitate per impostazione predefinita in Iceberg 1.0.0. Se desideri disabilitare le letture vettorializzate, imposta `read.parquet.vectorization.enabled` su `false`.

Oracle

Le modifiche sono di lieve entità.

MySQL

Le modifiche sono di lieve entità.

Amazon Redshift

AWS Glue 4.0 presenta un nuovo connettore Amazon Redshift con un nuovo driver JDBC. Per informazioni sui miglioramenti e su come effettuare la migrazione dalle versioni precedenti di AWS Glue, consulta the section called “Connessioni Redshift”.

Appendice A: Aggiornamenti importanti delle dipendenze

Di seguito sono riportati gli aggiornamenti delle dipendenze:

Dipendenza	Versione in AWS Glue 4.0	Versione in AWS Glue 3.0	Versione in AWS Glue 2.0	Versione in AWS Glue 1.0
Spark	3.3.0-amzn-1	3.1.1-amzn-0	2.4.3	2.4.3
Hadoop	3.3.3-amzn-0	3.2.1-amzn-3	2.8.5-amzn-5	2.8.5-amzn-1
Scala	2.12	2.12	2.11	2.11
Jackson	2.13.3	2.10.x	2.7.x	2.7.x
Hive	2.3.9-amzn-2	2.3.7-amzn-4	1.2	1.2
EMRFS	2.54.0	2.46.0	2.38.0	2.30.0

Dipendenza	Versione in AWS Glue 4.0	Versione in AWS Glue 3.0	Versione in AWS Glue 2.0	Versione in AWS Glue 1.0
Json4s	3.7.0-M11	3.6.6	3.5.x	3.5.x
Arrow	7.0.0	2.0.0	0.10.0	0.10.0
Client del catalogo dati AWS Glue	3.7.0	3.0.0	1.10.0	N/D
Python	3.10	3.7	2.7 e 3.6	2.7 e 3.6
Boto	1.26	1.18	1.12	N/D

Appendice B: aggiornamenti dei driver JDBC

Di seguito sono riportati gli aggiornamenti dei driver JDBC:

Driver	Versione del driver JDBC nelle precedenti versioni di AWS Glue	Versione del driver JDBC in AWS Glue 3.0	Versione del driver JDBC in AWS Glue 4.0
MySQL	5.1	8.0.23	8.0.23
Microsoft SQL Server	6.1.0	7.0.0	9.4.0
Database Oracle	11.2	21.1	21.7
PostgreSQL	42.1.0	42.2.18	42.3.6
MongoDB	2.0.0	4.0.0	4.7.2
Amazon Redshift	redshift-jdbc41-1.2.12.1017	redshift-jdbc41-1.2.12.1017	redshift-jdbc42-2.1.0.16

Appendice C: Aggiornamenti dei connettori

Di seguito sono riportati gli aggiornamenti dei connettori:

Driver	Versione del connettore in AWS Glue 3.0	Versione del connettore in AWS Glue 4.0
MongoDB	3.0.0	10.0.4
Hudi	0.10.1	0.12.1
Delta Lake	1.0.0	2.1.0
Iceberg	0.13.1	1.0.0
DynamoDB	1.11	1.12

Migrazione di AWS Glue per Ray dall'anteprima all'ambiente di runtime Ray2.4

> **Warning**
>
> Quando si salva un processo AWS Glue per Ray (anteprima) in AWS Glue Studio, questo verrà automaticamente aggiornato al runtime `Ray2.4`. Se riscontri problemi di compatibilità con lo script, contatta il supporto.

Dovresti migrare i processi AWS Glue creati utilizzando AWS Glue per Ray (anteprima) a AWS Glue per Ray. Ciò comporterà alcune modifiche simultanee alla configurazione del processo.

- Nel campo `Runtime`, fornisci il valore di runtime `Ray2.4`. Ciò aggiornerà la versione di Ray sottostante dalla 2.0.0 alla 2.4.0.
- Alcune librerie Python incluse per valore predefinito nell'anteprima non vengono più fornite. Se il tuo processo sfrutta l'SDK AWS per pandas (awswrangler), dask, modin o pymars, dovrai includerli come librerie aggiuntive. Per ulteriori informazioni sull'inclusione di librerie Python aggiuntive, consulta la pagina the section called “Moduli Python aggiuntivi per i processi Ray”.
- Se stai utilizzando il parametro `--additional-python-modules`, i parametri utilizzati per supportare questo flusso di processo sono stati suddivisi in `--pip-install` e `--s3-py-modules`. Per ulteriori informazioni su questi parametri, consultare the section called “Moduli Python aggiuntivi per i processi Ray”.

- Se stai utilizzando il parametro `--auto-scaling-ray-min-workers`, è stato rinominato `--min-workers`.

Policy di supporto versione AWS Glue

AWS Glue è un servizio di integrazione dati serverless che semplifica l'individuazione, la preparazione e la combinazione di dati per analisi dei dati, machine learning e sviluppo di applicazioni. Un processo AWS Glue contiene la logica di business che esegue le attività di integrazione dei dati in AWS Glue. Esistono tre tipi di processi in AWS Glue: Spark (in batch e in streaming), Ray e shell Python. Quando definisci il processo, specifica la versione di AWS Glue che configura le versioni dell'ambiente di runtime Spark, Ray o Python sottostante. Ad esempio: il processo AWS Glue Spark versione 2.0 supporta Spark 2.4.3 e Python 3.7.

Policy di supporto

Occasionalmente, AWS Glue interrompe il supporto per le vecchie versioni di AWS Glue. Tuttavia, i processi in esecuzione su versioni obsolete non sono più idonei per il supporto tecnico. AWS Glue non applicherà più patch di sicurezza o altri aggiornamenti alle versioni obsolete. AWS Glue inoltre non rispetterà gli SLA quando i processi vengono eseguiti su versioni obsolete.

Quando si raggiunge la fine del supporto per la versione 2.0 o successive di AWS Glue, non sarà possibile creare processi, ma solo modificarli o eseguirli.

Per le versioni AWS Glue seguenti è stata raggiunta o pianificata la fine del supporto. La fine del supporto inizia a mezzanotte (fuso orario del Pacifico) della data specificata.

Type	Versione Glue	Fine del supporto
Spark	Spark 2.2, Scala 2 (Glue versione 0.9)	01/06/2022
Spark	Spark 2.2, Python 2 (Glue versione 0.9)	01/06/2022
Spark	Spark 2.4, Python 2 (Glue versione 1.0)	01/06/2022
Spark	Spark 2.4, Python 3 (Glue versione 1.0)	30/09/2022

Type	Versione Glue	Fine del supporto
Spark	Spark 2.4, Scala 2 (Glue versione 1.0)	30/09/2022
Spark	Glue versione 2.0	31/2024
Type	Versione di Python	Fine del supporto
Shell Python	Python 2 (Glue versione 1.0)	01/06/2022
Type	Versione notebook	Fine del supporto
Endpoint di sviluppo	Notebook Zeppelin	30/09/2022

AWS consiglia la migrazione dei processi alle versioni supportate.

Per informazioni sulla migrazione dei processi Spark all'ultima versione di AWS Glue, consulta Migrazione dei processi AWS Glue a AWS Glue 4.0.

Per eseguire la migrazione dei processi di shell Python all'ultima versione di AWS Glue:

- Nella console, scegli `Python 3 (Glue Version 4.0)`.
- Nell'UpdateJobAPI CreateJob/, imposta il `GlueVersion` parametro su e `2.0` il `PythonVersion` to `3` sotto il `Command` parametro. La `GlueVersion` configurazione non influisce sul comportamento dei job della shell Python, quindi non c'è alcun vantaggio nell'incrementare. `GlueVersion`
- Devi rendere lo script del tuo processo compatibile con Python 3.

> **Note**
>
> Tutte le regioni AWS lanciate prima del lancio della regione di Giacarta, Indonesia (ap-southeast-3) nell'agosto 2022 dispongono di un elenco di clienti autorizzati a eseguire esecuzioni di processo di AWS Glue versione 0.9/1.0. In queste regioni precedenti, è possibile creare un processo con un valore nullo e la versione predefinita sarà 0.9/1.0 a seconda della regione. Per tutte le regioni AWS avviate successivamente, è necessario impostare esplicitamente la versione AWS Glue nell'API. AWS Glue non accetta più un

parametro nullo. Se viene passato 0.9 o 1.0 nel parametro, viene visualizzato l'errore "Glue Version 0.9 (or) 1.0 is not supported".

Utilizzo dei processi Spark in AWS Glue

Fornisce informazioni sui processi ETL di AWS Glue per Spark.

Argomenti

- Parametri del processo AWS Glue
- AWS GlueSpark e lavori PySpark
- Aggiunta di processi di streaming ETL in AWS Glue
- Corrispondenza dei record con FindMatches AWS Lake Formation
- Migrare i programmi Apache Spark a AWS Glue

Parametri del processo AWS Glue

Quando si crea un processo AWS Glue, si impostano alcuni campi standard, come `Role` e `WorkerType`. È possibile fornire informazioni di configurazione aggiuntive tramite i campi `Argument` (Parametri del processo nella console). In questi campi, i processi AWS Glue possono essere forniti con gli argomenti (parametri) riportati in questo documento. Per ulteriori informazioni sull'API di processo AWS Glue, consulta la pagina the section called “Processi”.

Impostazione dei parametri del processo

È possibile configurare un processo tramite la console nella scheda Job details (Dettagli del processo), sotto l'intestazione Job parameters (Parametri del processo). Puoi inoltre configurare un processo tramite AWS CLI impostando `DefaultArguments` o `NonOverridableArguments` su un processo o `Arguments` su un'esecuzione di processo. Gli argomenti impostati nel processo verranno trasmessi ogni volta che il processo viene eseguito, mentre gli argomenti impostati durante l'esecuzione del processo verranno trasmessi solo per quella singola esecuzione.

Ad esempio, di seguito è riportata la sintassi per l'esecuzione di un processo utilizzando `--arguments` per impostare un parametro di processo.

```
$ aws glue start-job-run --job-name "CSV to CSV" --arguments='--scriptLocation="s3://
my_glue/libraries/test_lib.py"'
```

Accesso ai parametri di processo

Quando scrivi script AWS Glue, potresti voler accedere ai valori dei parametri di processo per modificare il comportamento del tuo codice. Forniamo metodi di supporto per eseguire questa operazione tramite le nostre librerie. Questi metodi riescono a risolvere i valori dei parametri di esecuzione del processo che sostituiscono i valori dei parametri del processo. Quando si risolvono i parametri impostati in più posizioni, il processo `NonOverridableArguments` sostituisce l'esecuzione del processo `Arguments`, che sostituisce il processo `DefaultArguments`.

In Python:

Nei processi Python, forniamo una funzione denominata `getResolvedParameters`. Per ulteriori informazioni, consulta the section called “getResolvedOptions”. I parametri del processo sono disponibili nella variabile `sys.argv`.

In Scala:

Nei processi Scala, forniamo un oggetto denominato `GlueArgParser`. Per ulteriori informazioni, consulta the section called “GlueArgParser”. I parametri del processo sono disponibili nella variabile `sysArgs`.

Riferimento ai parametri di processo

AWS Glue riconosce i seguenti nomi di argomenti che puoi utilizzare per configurare l'ambiente di script per i processi e le esecuzioni di processi:

`--additional-python-modules`

Un elenco delimitato da virgole che rappresenta un insieme di pacchetti Python da installare. Puoi installare pacchetti da PyPI o fornire una distribuzione personalizzata. Una voce del pacchetto PyPI sarà nel formato *`package`*==*`version`*, con il nome e la versione del pacchetto di destinazione. Una voce della distribuzione personalizzata è rappresentata dal percorso S3 della distribuzione.

Le voci utilizzano la versione di Python corrispondente al pacchetto e alla versione, in modo che l'utente non debba utilizzare due segni uguali, come ==. Per ulteriori informazioni su altri operatori che corrispondono alle versioni, consulta la pagina PEP 440.

Per inviare le opzioni di installazione del modulo a `pip3`, utilizza il parametro --python-modules-installer-option.

`--auto-scale-within-microbatch`

Il valore predefinito è false. Questo parametro può essere utilizzato solo per i processi di flussi di dati AWS Glue, che elaborano i flussi di dati in una serie di micro batch, e richiede l'abilitazione del dimensionamento automatico. Quando si imposta questo valore su "false", viene calcolata la media mobile esponenziale della durata del batch per i microbatch completati e questo valore viene confrontato con la dimensione della finestra per determinare se aumentare o ridurre il numero di esecutori. Il dimensionamento avviene solo quando viene completato un microbatch. Quando si imposta questo valore su "true", durante un microbatch, l'aumento avviene quando il numero di attività Spark rimane invariato per 30 secondi o se l'elaborazione del batch corrente è maggiore della dimensione della finestra. Il numero di esecutori diminuisce se un esecutore è rimasto inattivo per più di 60 secondi o se la media mobile esponenziale della durata del batch è bassa.

`--class`

La classe Scala che funge da punto di accesso per lo script Scala. Questo vale solo se il tuo `--job-language` è impostato su `scala`.

`--continuous-log-conversionPattern`

Specifica un modello di log di conversione personalizzato per un processo abilitato per la registrazione continua. Il modello di conversione si applica solo ai log dei driver e ai log delle esecuzioni. Non interessa la barra di avanzamento di AWS Glue.

`--continuous-log-logGroup`

Specifica un nome di gruppo di CloudWatch log Amazon personalizzato per un job abilitato alla registrazione continua.

`--continuous-log-logStreamPrefix`

Specifica un prefisso di CloudWatch log stream personalizzato per un job abilitato alla registrazione continua.

`--customer-driver-env-vars` e **`--customer-executor-env-vars`**

Questi parametri impostano le variabili di ambiente nel sistema operativo rispettivamente per ogni lavoratore (driver o esecutore). Puoi utilizzare questi parametri quando crei piattaforme e framework personalizzati su AWS Glue, per consentire ai tuoi utenti di scrivere lavori su di esso. L'attivazione di questi due flag vi consentirà di impostare diverse variabili di ambiente

rispettivamente sul driver e sull'executor senza dover inserire la stessa logica nello script di lavoro stesso.

Esempio di utilizzo

Di seguito è riportato un esempio di utilizzo di questi parametri:

```
"–customer-driver-env-vars", "CUSTOMER_KEY1=VAL1,CUSTOMER_KEY2=\"val2,val2 val2\"",
"–customer-executor-env-vars", "CUSTOMER_KEY3=VAL3,KEY4=VAL4"
```

L'impostazione di questi nell'argomento job run equivale all'esecuzione dei seguenti comandi:

Nel driver:

- esporta CUSTOMER_KEY1=VAL1
- esporta CUSTOMER_KEY2="val2, val2 val2"

Nell'esecutore:

- esporta CUSTOMER_KEY3=VAL3

Quindi, nello script di lavoro stesso, è possibile recuperare le variabili di ambiente utilizzando o. `os.environ.get("CUSTOMER_KEY1")` `System.getenv("CUSTOMER_KEY1")`

Sintassi applicata

Osserva i seguenti standard quando definisci le variabili di ambiente:

- Ogni chiave deve avere il`CUSTOMER_ prefix`.

  Ad esempio: for`"CUSTOMER_KEY3=VAL3,KEY4=VAL4"`, `KEY4=VAL4` verrà ignorato e non impostato.
- Ogni coppia di chiavi e valori deve essere delineata con una sola virgola.

  Ad esempio: `"CUSTOMER_KEY3=VAL3,CUSTOMER_KEY4=VAL4"`
- Se il «valore» contiene spazi o virgole, deve essere definito tra virgolette.

  Ad esempio: `CUSTOMER_KEY2=\"val2,val2 val2\"`

Questa sintassi modella da vicino gli standard di impostazione delle variabili di ambiente bash.

`--datalake-formats`

Supportato in AWS Glue 3.0 e versioni successive.

Specifica il framework del data lake da utilizzare. AWS Glue aggiunge i file JAR necessari per i framework specificati nella variabile `classpath`. Per ulteriori informazioni, consulta Utilizzo di framework data lake con processi ETL di AWS Glue.

Puoi specificare uno o più dei seguenti valori, separati da una virgola:

- `hudi`
- `delta`
- `iceberg`

Ad esempio, invia il seguente argomento per specificare tutti e tre i framework.

```
'--datalake-formats': 'hudi,delta,iceberg'
```

`--disable-proxy-v2`

Disattiva il proxy del AWS servizio per consentire le chiamate di servizio verso Amazon S3 e AWS Glue provenienti dallo script tramite il tuo VPC. CloudWatch Per ulteriori informazioni, consulta Configurazione di chiamate AWS affinché passino attraverso il tuo VPC. Per disabilitare il proxy del servizio, imposta il valore di questo parametro su `true`.

`--enable-auto-scaling`

Attiva la scalabilità automatica e la fatturazione per operatore quando imposti il valore su `true`.

`--enable-continuous-cloudwatch-log`

Abilita la registrazione continua in tempo reale per i processi AWS Glue. Puoi visualizzare i log dei processi Apache Spark in tempo reale in CloudWatch.

`--enable-continuous-log-filter`

Specifica un filtro standard (`true`) o nessun filtro (`false`) durante la creazione o la modifica di un processo abilitato per la registrazione continua. La scelta del filtro Standard elimina i messaggi di log di heartbeat inutili di driver/executor Apache Spark e Apache Hadoop YARN. Non scegliendo alcun filtro si ottengono tutti i messaggi di log.

`--enable-glue-datacatalog`

Consente di utilizzare il catalogo dati di AWS Glue come metastore Apache Spark Hive. Imposta questo valore su `true` per abilitare questa funzionalità.

`--enable-job-insights`

Consente di monitorare ulteriormente l'analisi degli errori con le informazioni dettagliate sull'esecuzione dei processi di AWS Glue. Per informazioni dettagliate, vedi the section called “Monitoraggio con le informazioni dell'esecuzione del processo di AWS Glue”. Per impostazione predefinita, il valore è impostato su `true` e le informazioni dettagliate sull'esecuzione dei processi sono abilitate.

Questa opzione è disponibile con AWS Glue versione 2.0 e 3.0.

`--enable-metrics`

Abilita la raccolta di parametri per la profilatura del processo per questa esecuzione. Queste metriche sono disponibili sulla AWS Glue console e sulla CloudWatch console Amazon. Il valore di questo parametro non è rilevante. Per abilitare questa funzionalità, è possibile fornire a questo parametro qualsiasi valore, ma `true` è consigliabile per motivi di chiarezza. Per disabilitare la funzionalità, rimuovi questo parametro dalla configurazione del processo.

`--enable-observability-metrics`

Abilita una serie di parametri di osservabilità per generare informazioni su ciò che accade all'interno di ogni esecuzione del processo nella pagina sul monitoraggio dell'esecuzione dei processi nella console AWS Glue e nella console Amazon CloudWatch. Per abilitare questa funzionalità, imposta il valore del parametro su "true". Per disabilitare questa funzionalità, impostalo su `false` o rimuovi questo parametro dalla configurazione del processo.

`--enable-rename-algorithm-v2`

Imposta la versione dell'algoritmo di ridenominazione EMRFS alla versione 2. Quando un processo Spark utilizza la modalità di sovrascrittura della partizione dinamica, è possibile che venga creata una partizione duplicata. Ad esempio, si può ottenere una partizione duplicata come `s3://bucket/table/location/p1=1/p1=1`. Qui, P1 è la partizione che viene sovrascritta. La versione 2 dell'algoritmo di ridenominazione risolve questo problema.

Questa opzione è disponibile solo nella versione AWS Glue 1.0.

`--enable-s3-parquet-optimized-committer`

Abilita il committer ottimizzato EMRFS S3 per la scrittura dei dati Parquet in Amazon S3. Puoi fornire la coppia parametro/valore tramite la console AWS Glue durante la creazione o l'aggiornamento di un processo AWS Glue. L'impostazione del valore su **`true`** abilita il committer. Per impostazione predefinita, il flag è attivato in AWS Glue 3.0 e disattivato in AWS Glue 2.0.

Per ulteriori informazioni, consulta Utilizzo del committer ottimizzato EMRFS S3.

--enable-spark-ui

Quando è impostato su `true`, attiva la funzionalità per utilizzare l'interfaccia utente di Spark per monitorare ed eseguire il debug dei processi ETL di AWS Glue.

--executor-cores

Numero di attività spark che possono essere eseguite in parallelo. Questa opzione è supportata su AWS Glue versione 3.0 e successive. Il valore non deve superare il doppio del numero di vCPU del tipo di worker, ossia 8 su `G.1X`, 16 su `G.2X`, 32 su `G.4X` e 64 su `G.8X`. È necessario prestare attenzione durante l'aggiornamento di questa configurazione in quanto potrebbe influire sulle prestazioni del processo; l'incremento del parallelismo, infatti, esercita pressione sulla memoria e sul disco, oltre a limitare i sistemi di origine e destinazione (ad esempio, potrebbe causare più connessioni simultanee su Amazon RDS).

--extra-files

I percorsi Amazon S3 dei file aggiuntivi, come file di configurazione, che AWS Glue copia nella directory di lavoro del tuo script prima di eseguirlo. I valori multipli devono essere percorsi completi separati dalla virgola (`,`). Solo i singoli file sono supportati, non il percorso di una directory. Questa opzione non è supportata per i tipi di processo shell Python.

--extra-jars

I percorsi Amazon S3 dei file Java `.jar` aggiuntivi che AWS Glue aggiunge alla classpath Java prima di eseguire lo script. I valori multipli devono essere percorsi completi separati dalla virgola (`,`).

--extra-py-files

I percorsi Amazon S3 dei moduli Python aggiuntivi che AWS Glue aggiunge al percorso Python prima di eseguire lo script. I valori multipli devono essere percorsi completi separati dalla virgola (`,`). Solo i singoli file sono supportati, non il percorso di una directory.

--job-bookmark-option

Controlla il comportamento di un segnalibro del processo. É possibile impostare i seguenti valori opzione.

--job-bookmark-option Valore	Descrizione
`job-bookmark-enable`	Tieni traccia dei dati elaborati in precedenza. Quando si esegue un processo, elabora i nuovi dati a partire dall'ultimo punto di controllo.
`job-bookmark-disable`	Elabora sempre l'intero set di dati. Sei responsabile della gestione dell'output dalle esecuzioni dei processi precedenti.
`job-bookmark-pause`	Elabora i dati incrementali dall'ultima esecuzione riuscita o i dati nell'intervallo identificato dalle seguenti opzioni secondarie, senza aggiornare lo stato dell'ultimo segnalibro. Sei responsabile della gestione dell'output dalle esecuzioni dei processi precedenti. Le due opzioni secondarie sono le seguenti: • **`job-bookmark-from`** `<from-value>` è l'ID di esecuzione che rappresenta tutto l'input che è stato elaborato fino all'ultima esecuzione riuscita prima, incluso l'ID di esecuzione specificato. L'input corrispondente viene ignorato. • **`job-bookmark-to`** `<to-value>` è l'ID di esecuzione che rappresenta tutto l'input che è stato elaborato fino all'ultima esecuzione riuscita prima, incluso l'ID di esecuzione specificato. L'input corrispondente escluso l'input identificato da `<from-value>` viene elaborato dal processo. Qualsiasi input successivo a questo è escluso anche dall'elaborazione. Lo stato dei segnalibri di processo non viene aggiornato quando viene specificato questo set di opzioni. Le opzioni secondarie sono facoltative. Tuttavia, se vengono utilizzate, devono essere fornite entrambe le opzioni secondarie.

Ad esempio, per abilitare un segnalibro di processo, passa l'argomento seguente.

```
'--job-bookmark-option': 'job-bookmark-enable'
```

`--job-language`

Il linguaggio di programmazione script. Questo valore deve essere `scala` o `python`. Se questo parametro non è presente, il valore predefinito è `python`.

`--python-modules-installer-option`

Una stringa di testo semplice che definisce le opzioni da inviare a `pip3` quando si installano i moduli con --additional-python-modules. Fornisci le opzioni come faresti nella riga di comando, separate da spazi e precedute da trattini. Per ulteriori informazioni sull'utilizzo, consulta the section called “Installazione di moduli Python aggiuntivi in AWS Glue 2.0 con pip”.

> **Note**
>
> Questa opzione non è supportata per i processi di AWS Glue quando si utilizza Python 3.9.

`--scriptLocation`

La posizione di Amazon Simple Storage Service (Amazon S3) in cui si trova lo script ETL (nel formato `s3://path/to/my/script.py`). Questo parametro sovrascrive un percorso script impostato nell'oggetto `JobCommand`.

`--spark-event-logs-path`

Specifica un percorso Amazon S3. Quando si utilizza la funzionalità di monitoraggio dell'interfaccia utente di Spark, AWS Glue scarica i log degli eventi Spark su questo percorso Amazon S3 ogni 30 secondi in un bucket utilizzabile come directory temporanea per l'archiviazione di eventi dell'interfaccia utente Spark.

`--TempDir`

Specifica un percorso Amazon S3 a un bucket utilizzabile come directory temporanea per il processo.

Ad esempio, per impostare una directory temporanea, passa l'argomento seguente.

```
'--TempDir': 's3-path-to-directory'
```

> **Note**
>
> AWS Glue crea un bucket temporaneo per i processi, se il bucket non esiste già una regione. Questo bucket potrebbe consentire l'accesso pubblico. Puoi modificare il bucket in Amazon S3 per impostare il blocco dell'accesso pubblico oppure eliminare il bucket in un secondo momento dopo che tutti i processi in quella regione sono stati completati.

`--use-postgres-driver`

Impostando questo valore su `true`, assegna la priorità al driver JDBC Postgres nella variabile classpath per evitare un conflitto con il driver JDBC Amazon Redshift. Questa opzione è disponibile solo in AWS Glue versione 2.0.

`--user-jars-first`

Impostando questo valore su `true`, dà la priorità ai file JAR aggiuntivi del cliente nella variabile classpath. Questa opzione è disponibile solo nella versione AWS Glue 2.0 o successive.

`--conf`

Controlla i parametri di configurazione di Spark. È per casi d'uso avanzati.

`--encryption-type`

Parametro legacy. Il comportamento corrispondente deve essere configurato utilizzando le configurazioni di sicurezza. Per ulteriori informazioni sulle configurazioni di sicurezza, consulta la pagina the section called “Crittografia dei dati scritti da AWS Glue”.

AWS Glue utilizza internamente i seguenti argomenti, che non dovresti mai usare:

- `--debug`: interno a AWS Glue. Non impostare.
- `--mode`: interno a AWS Glue. Non impostare.
- `--JOB_NAME`: interno a AWS Glue. Non impostare.
- `--endpoint`: interno a AWS Glue. Non impostare.

AWS Glue supporta l'operazione di bootstrap di un ambiente con il modulo `site` di Python utilizzando `sitecustomize` per eseguire personalizzazioni specifiche del sito. L'operazione di bootstrap delle proprie funzioni di inizializzazione è consigliata solo per casi d'uso avanzati ed è supportata su base best effort dalla versione 4.0 di AWS Glue.

Il prefisso della variabile di ambiente, `GLUE_CUSTOMER`, è riservato all'uso da parte dei clienti.

AWS GlueSpark e lavori PySpark

Le seguenti sezioni forniscono informazioni su AWS Glue Spark e PySpark i lavori.

Argomenti

- Aggiunta di lavori Spark e PySpark in AWS Glue
- Monitoraggio dei dati elaborati mediante segnalibri di processo
- Plug-in shuffle di AWS Glue Spark con Amazon S3
- Monitoraggio dei processi Spark AWS Glue

Aggiunta di lavori Spark e PySpark in AWS Glue

Nelle sezioni seguenti vengono fornite informazioni sull'aggiunta di lavori Spark e PySpark in AWS Glue.

Argomenti

- Aggiunta di processi in AWS Glue
- Modifica degli script Spark nella console AWS Glue
- Processi (legacy)

Aggiunta di processi in AWS Glue

Un processo AWS Glue incapsula uno script che si connette ai dati di origine, lo elabora e quindi lo scrive nella destinazione dati. Di solito un processo esegue script di estrazione, trasformazione e caricamento (ETL). I processi possono anche eseguire script Python generici (processi shell Python). AWS Glue I trigger possono avviare processi in base a una pianificazione, un evento o su richiesta. È possibile monitorare le esecuzioni dei processi per comprendere i parametri di runtime come esito positivo, durata e ora di inizio.

È possibile utilizzare gli script generati da AWS Glue oppure è possibile fornire i propri. Con uno schema di origine e una posizione o uno schema di destinazione, il generatore di AWS Glue codice può creare automaticamente uno script Apache Spark API (PySpark). Puoi usare questo script come punto di partenza e modificarlo per soddisfare gli obiettivi.

AWS Glue può scrivere file di output in diversi formati di dati, tra cui JSON, CSV, ORC (Optimized Row Columnar), Apache Parquet e Apache Avro. Per alcuni formati di dati, possono essere scritti formati comuni di compressione.

Esistono tre tipi di lavori in AWS Glue: Spark, Streaming ETL e Python shell.

- Un job Spark viene eseguito in un ambiente Apache Spark gestito da. AWS Glue Elabora i dati in batch.
- Un processo ETL di streaming è simile a un processo Spark, ad eccezione del fatto che esegue ETL sui flussi di dati. Esso utilizza il framework Apache Spark Structured Streaming. Alcune caratteristiche dei processi Spark non sono disponibili per i processi ETL in streaming.
- Un processo shell Python: esegue script di Python come shell e supporta una versione di Python a seconda della versione di AWS Glue che si sta utilizzando. Puoi utilizzare questi processi per pianificare ed eseguire attività che non richiedono un ambiente Apache Spark.

Definire le proprietà di processo per i processi Spark

Quando definisci un processo nella console AWS Glue, devi fornire i valori delle proprietà per controllare l'ambiente di runtime AWS Glue.

L'elenco seguente descrive le proprietà di un processo Spark. Per le proprietà di un processo shell di Python, consulta Definire le proprietà del processo per i processi shell di Python. Per le proprietà di un processo ETL di streaming, vedere the section called “Definizione delle proprietà di processo per un processo di streaming ETL”.

Le proprietà sono elencate nell'ordine in cui vengono visualizzate nel processo di creazione guidata Add job (Aggiungi processo) nella console AWS Glue.

Nome

Stringa UTF-8 con un massimo di 255 caratteri.

Descrizione

Fornisci una descrizione opzionale di un massimo di 2048 caratteri.

Ruolo IAM

Specifica il ruolo IAM; utilizzato per definire l'autorizzazione alle risorse utilizzate per eseguire il processo e accedere agli archivi dati. Per ulteriori informazioni sulle autorizzazioni per l'esecuzione di processi in AWS Glue, consulta Gestione delle identità e degli accessi per AWS Glue.

Type

Il tipo di processo ETL. Viene impostato automaticamente in base al tipo di fonti di dati selezionate.

- Spark esegue uno script ETL di Apache Spark con il comando job. `glueetl`
- Spark Streaming esegue uno script ETL di streaming Apache Spark con il comando job. `gluestreaming` Per ulteriori informazioni, consulta the section called “Aggiunta di processi di streaming ETL”.
- Python shell esegue uno script Python con il comando job. `pythonshell` Per ulteriori informazioni, consulta Processi shell di Python in AWS Glue.

Versione AWS Glue

La versione di AWS Glue determina le versioni di Apache Spark e Python disponibili per il processo, come specificato nella tabella seguente.

Versione AWS Glue	Versioni Spark e Python supportate
4.0	• Spark 3.3.0 • Python 3.10
3.0	• Spark 3.1.1 • Python 3.7
2.0	• Spark 2.4.3 • Python 3.7
1	• Spark 2.4.3 • Python 2.7 • Python 3.6
0.9	• Spark 2.2.1

Versione AWS Glue	Versioni Spark e Python supportate
	• Python 2.7

Tipo di worker

Sono disponibili i seguenti tipi di worker:

Le risorse disponibili per AWS Glue i worker sono misurate in DPU. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria.

- **`G.1X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria) con disco da 84 GB (circa 34 GB liberi). Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
- **`G.2X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria) con disco da 128 GB (circa 77 GB liberi). Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
- **`G.4X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 4 DPU (16 vCPU, 64 GB di memoria) con disco da 256 GB (circa 235 GB liberi). Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di lavoratore è disponibile solo per i job Spark ETL AWS Glue versione 3.0 o successiva AWS nelle seguenti regioni: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda) ed Europa (Stoccolma).
- **`G.8X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 8 DPU (32 vCPU, 128 GB di memoria) con disco da 512 GB (circa 487 GB liberi). Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i job Spark ETL AWS Glue versione 3.0 o successiva, nelle stesse AWS regioni supportate per il tipo di `G.4X` lavoratore.

- **`G.025X`**: quando si sceglie questo tipo, si fornisce anche un valore per Number of workers (Numero di worker). Ogni worker esegue la mappatura su 0,25 DPU (2 vCPU, 4 GB di memoria) con disco da 84 GB (circa 34 GB liberi). Consigliamo questo tipo di worker per i processi di streaming a basso volume. Questo tipo di worker è disponibile solo per AWS Glue processi di streaming versione 3.0.

Viene addebitata una tariffa oraria calcolata in base al numero di DPU utilizzate per eseguire i processi ETL. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

Per la versione 1.0 AWS Glue o processi precedenti, quando si configura un processo utilizzando la console e si specifica un Worker type (Tipo di worker) Standard, viene impostata la Maximum capacity (Capacità massima) e il Number of workers (Numero di worker) diventa il valore di Maximum capacity (Capacità massima) - 1. Se utilizzi AWS Command Line Interface (AWS CLI) o AWS SDK, puoi specificare il parametro Capacità massima oppure puoi specificare sia il tipo di lavoratore che il numero di lavoratori.

Per processi AWS Glue versione 2.0 o successiva, non è possibile specificare una Capacità massima. È invece necessario specificare un Worker type (Tipo di worker) e il Number of workers (Numero di worker).

Lingua

Il codice nello script ETL definisce la logica del processo. Lo script può essere codificato in Python o Scala. Puoi scegliere se lo script che esegue il processo è generato da AWS Glue o fornito da te. Puoi fornire il nome e la posizione dello script in Amazon Simple Storage Service (Amazon S3). Conferma che non esiste un file con lo stesso nome della directory di script nel percorso. Per ulteriori informazioni sull'uso degli script, consulta AWS Glue guida alla programmazione.

Numero richiesto di lavoratori

Per la maggior parte dei tipi di worker è necessario specificare il numero di worker allocati quando il processo viene eseguito.

Segnalibro di processo

Specifica in che modo AWS Glue elabora informazioni sullo stato quando viene eseguito il processo. Puoi ricordare di aver già elaborato i dati, aggiornato le informazioni sullo stato o ignorato le informazioni sullo stato. Per ulteriori informazioni, consulta the section called “Monitoraggio dei dati elaborati mediante segnalibri di processo”.

Esecuzione flessibile

Quando configuri un lavoro utilizzando AWS Studio o l'API, puoi specificare una classe di esecuzione del lavoro standard o flessibile. I tuoi processo possono avere diversi gradi di priorità e sensibilità temporale. La classe di esecuzione standard è ideale per carichi di lavoro sensibili al tempo che richiedono un avvio rapido dei processi e risorse dedicate.

La classe di esecuzione flessibile è adatta per processi non urgenti come i processi di pre-produzione, test e caricamenti di dati una tantum. Le esecuzioni dei processi flessibili sono supportate per i processi che utilizzano AWS Glue versione 3.0 o successive e i tipi di worker `G.1X` o `G.2X`.

Le esecuzioni dei processi flessibili vengono fatturate in base al numero di worker che vengono eseguiti alla volta. Il numero di worker può essere aggiunto o rimosso per un'esecuzione di lavoro flessibile in esecuzione. Invece di fatturare come semplice calcolo di `Max Capacity * Execution Time`, ogni worker contribuirà per il tempo che è stato eseguito durante l'esecuzione del processo. La fattura è la somma di (`Number of DPUs per worker * time each worker ran`).

Per ulteriori informazioni, consulta il pannello di aiuto in AWS Studio oppure Processi eEsecuzioni di processi.

Numero di tentativi

Specifica il numero di volte, da 0 a 10, che il processo deve essere riavviato da AWS Glue se non va a buon fine. I processi che raggiungono il limite di timeout non vengono riavviati.

Timeout dei processi

Imposta il tempo di esecuzione massimo in minuti. Il valore predefinito è 2.880 minuti (48 ore) per i processi batch. Quando il tempo di esecuzione del processo supera questo limite, lo stato del processo cambia in `TIMEOUT`.

Per i processi di streaming eseguiti a tempo indeterminato, lascia vuoto il valore, che è il valore predefinito per i processi di streaming.

> **Le migliori pratiche per le interruzioni lavorative**
>
> I lavori vengono fatturati in base al tempo di esecuzione. Per evitare addebiti imprevisti, configura i valori di timeout appropriati per il tempo di esecuzione previsto del lavoro.

Proprietà avanzate

Nome del file dello script

Un nome di script univoco per il tuo lavoro. Non può essere denominato Untitled job.

Percorso dello script

La posizione dello script in Amazon S3. Il percorso deve essere nel formato `s3://bucket/prefix/path/`. Deve terminare con una barra (/) e non includere alcun file.

Parametri del processo

Attiva o disattiva la creazione di CloudWatch metriche Amazon durante l'esecuzione di questo processo. Per visualizzare i dati di profiling, è necessario abilitare questa opzione. Per ulteriori informazioni su come attivare e visualizzare i parametri, consulta Monitoraggio e debug dei processi.

Metriche di osservabilità del lavoro

Attiva la creazione di CloudWatch metriche di osservabilità aggiuntive durante l'esecuzione di questo lavoro. Per ulteriori informazioni, consulta the section called “Monitoraggio con parametri AWS Glue di osservabilità”.

Registrazione continua

Attiva la registrazione continua su Amazon CloudWatch. Se questa opzione non è abilitata, i registri sono disponibili solo dopo il completamento del processo. Per ulteriori informazioni, consulta the section called “Registrazione continua dei processi AWS Glue”.

Interfaccia utente di Spark

Attiva l'uso dell'interfaccia utente di Spark per monitorare questo processo. Per ulteriori informazioni, consulta Abilitazione dell'interfaccia utente Web di Apache Spark per processi AWS Glue.

Percorso dei registri dell'interfaccia utente di Spark

Il percorso per scrivere i log quando l'interfaccia utente Spark è abilitata.

Configurazione di registrazione e monitoraggio dell'interfaccia utente Spark

Selezionare una delle seguenti opzioni:

- Standard: scrive i log usando l'ID del AWS Glue job run come nome del file. Attiva il monitoraggio dell'interfaccia utente Spark nella console. AWS Glue

- Legacy: scrivi i log usando 'spark-application- {timestamp} 'come nome del file. Non attivare il monitoraggio dell'interfaccia utente Spark.
- Standard e legacy: scrivi i log sia nelle posizioni standard che in quelle precedenti. Attiva il monitoraggio dell'interfaccia utente Spark nella AWS Glue console.

Simultaneità massima

Imposta il numero massimo di esecuzioni simultanee consentite per il processo. Il valore di default è 1. Viene restituito un errore al raggiungimento della soglia. Il valore massimo che è possibile specificare è controllato da un limite di servizio. Ad esempio, se un'esecuzione di un processo precedente non è ancora terminata quando una nuova istanza viene avviata, è possibile restituire un errore per evitare che due istanze dello stesso processo vengano eseguite simultaneamente.

Percorso temporaneo

Fornisci il percorso di una directory di lavoro in Amazon S3, in cui vengono scritti i risultati intermedi temporanei quando AWS Glue esegue lo script. Conferma che non esiste un file con lo stesso nome della directory temporanea nel percorso. Questa directory viene usata quando AWS Glue legge e scrive in Amazon Redshift e da alcune trasformazioni AWS Glue.

> **Note**
>
> AWS Glue crea un bucket temporaneo per i processi, se il bucket non esiste già una regione. Questo bucket potrebbe consentire l'accesso pubblico. Puoi modificare il bucket in Amazon S3 per impostare il blocco dell'accesso pubblico oppure eliminare il bucket in un secondo momento dopo che tutti i processi in quella regione sono stati completati.

Soglia notifica di ritardo (minuti)

Imposta la soglia (in minuti) prima di inviare una notifica di ritardo. Puoi impostare questa soglia per inviare notifiche quando l'esecuzione di un processo `RUNNING`, `STARTING` o `STOPPING` impiega di più rispetto alla quantità di minuti attesa.

Configurazione di sicurezza

Scegliere una configurazione di sicurezza dall'elenco. Una configurazione di sicurezza specifica come vengono crittografati i dati del target Amazon S3: nessuna crittografia,

crittografia lato server con chiavi gestite da AWS KMS (SSE-KMS) o chiavi di crittografia gestite da Amazon S3 (SSE-S3).

Crittografia lato server

Se selezioni questa opzione, quando il processo ETL scrive in Amazon S3, i dati vengono crittografati quando sono inattivi tramite crittografia SSE-S3. Vengono crittografati sia i dati di destinazione Amazon S3 sia tutti gli altri dati scritti in una directory temporanea Amazon S3. Questa opzione viene passata come parametro del processo. Per ulteriori informazioni, consulta Protezione dei dati mediante la crittografia lato server con chiavi di crittografia gestite da Amazon S3 (SSE-S3) nella Guida per l'utente di Amazon Simple Storage Service.

> **Important**
>
> Questa opzione viene ignorata se viene specificata una configurazione di protezione.

Opzione per l'uso del catalogo dati di Glue come metastore Hive

Selezionare per utilizzare il catalogo dati di AWS Glue come metastore Hive. Il ruolo IAM utilizzato per il processo deve disporre dell'autorizzazione `glue:CreateDatabase`. Viene creato un database chiamato "default" nel catalogo dati, nel caso non fosse già presente.

Connessioni

Scegli una configurazione VPC per accedere alle fonti di dati Amazon S3 situate nel tuo cloud privato virtuale (VPC). Puoi creare e gestire una connessione di rete in. AWS Glue Per ulteriori informazioni, consulta Connessione ai dati.

Libraries (Librerie)

Percorso della libreria Python, percorso JARs dipendente e percorso dei file di riferimento

Specificare queste opzioni se lo script le richiede. Puoi definire percorsi separati da virgole Amazon S3 per queste opzioni quando definisci il processo. Puoi sostituire tali percorsi quando esegui il processo. Per ulteriori informazioni, consulta Fornire i propri script personalizzati.

Parametri del processo

Un insieme di coppie chiave-valore che vengono passate come parametri denominati allo script. Si tratta di valori predefiniti che vengono utilizzati quando lo script viene eseguito, ma è possibile ignorarli nei trigger o quando si esegue il processo. È necessario prefissare il nome

della chiave con `--`; ad esempio: `--myKey`. I parametri del lavoro vengono passati come mappa quando si utilizza. AWS Command Line Interface

Per ulteriori esempi, vedere i parametri Python in Passaggio di parametri Python in AWS Glue e accesso ai parametri.

Tag

Il tag si applica al processo tramite una Tag key (Chiave tag) e un Tag value (Valore tag) facoltativo. Una volta create, le chiavi di tag sono di sola lettura. Usa i tag su alcune risorse per facilitarne l'organizzazione e l'individuazione. Per ulteriori informazioni, consulta AWS tag in AWS Glue.

Restrizioni per i processi che accedono alle tabelle gestite da Lake Formation

Tieni presente le seguenti note e restrizioni quando crei lavori che leggono o scrivono su tabelle gestite da AWS Lake Formation:

- Le seguenti caratteristiche non sono supportate nei processi che accedono alle tabelle con filtri a livello di cella:
  - Segnalibri di processo ed esecuzione limitata
  - Predicati pushdown
  - Predicati delle partizioni dei cataloghi lato server
  - enableUpdateCatalog

Modifica degli script Spark nella console AWS Glue

Uno script contiene il codice che estrae dati dalle origini, li trasforma e li carica nelle destinazioni. AWS Glue esegue uno script quando avvia un processo.

Gli script ETL AWS Glue possono essere codificati in Python o Scala. Gli script Python utilizzano un linguaggio che è un'estensione del dialetto PySpark Python per i lavori di estrazione, trasformazione e caricamento (ETL). Lo script contiene costrutti estesi per gestire le trasformazioni ETL. Quando si genera automaticamente la logica del codice sorgente per un processo, viene creato lo script. Puoi modificare questo script oppure puoi fornire il tuo script per elaborare il lavoro ETL.

Per informazioni sulla definizione e sulla modifica di script in AWS Glue, consulta la pagina AWS Glue guida alla programmazione.

Librerie o file aggiuntivi

Se lo script richiede librerie o file aggiuntivi, puoi specificarli come segue:

Python library path (Percorso libreria Python)

Percorsi Amazon Simple Storage Service (Amazon S3) separati da virgole per le librerie Python richieste dallo script.

> **Note**
>
> Possono essere utilizzate solo le librerie pure Python. Le librerie che si basano sulle estensioni C, come la libreria di analisi dati Python pandas, non sono ancora supportate.

Dependent jars path (Percorso file .jar dipendente)

Percorsi Amazon S3 separati da virgole dei file JAR richiesti dallo script.

> **Note**
>
> Al momento possono essere utilizzate solo le librerie pure Java o Scala (2.11).

Percorso dei file di riferimento

Percorsi Amazon S3 separati da virgole di file aggiuntivi (ad esempio i file di configurazione) richiesti dallo script.

Processi (legacy)

Uno script contiene il codice che estrae, trasforma e carica il lavoro (ETL). Puoi fornire uno script personalizzato oppure AWS Glue può generare uno script seguendo le tue indicazioni. Per informazioni su come creare gli script, consulta Fornire i propri script personalizzati.

Puoi modificare uno script nella console AWS Glue. Quando modifichi uno script, puoi aggiungere origini, destinazioni e trasformazioni.

Per modificare uno script

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/. Selezionare Processi scheda.
2. Scegli un processo nell'elenco, quindi Action (Operazione), Edit script (Modifica script) per aprire l'editor di script.

   Puoi inoltre accedere all'editor di script dalla pagina dei dettagli del processo. Scegli la scheda Script, quindi scegli Edit script (Modifica script).

Editor di script

L'editor di script AWS Glue permette di inserire, modificare ed eliminare origini, destinazioni e trasformazioni nello script. L'editor di script visualizza sia lo script sia un diagramma per aiutarti a visualizzare il flusso di dati.

Per creare un diagramma per lo script, scegli Generate diagram (Genera diagramma). AWS Glue usa righe di annotazione nello script che iniziano con # # per visualizzare il diagramma. Per rappresentare correttamente lo script nel diagramma, è necessario mantenere sincronizzati i parametri nelle annotazioni e i parametri nel codice Apache Spark.

L'editor di script ti consente di aggiungere modelli di codice ovunque il cursore sia posizionato nello script. Nella parte superiore dell'editor, sceglie tra le seguenti opzioni:

- Per aggiungere una tabella di origine allo script, scegli Source (Origine).
- Per aggiungere una tabella di destinazione allo script, scegli Target (Destinazione).
- Per aggiungere una posizione di destinazione allo script, scegli Target location (Posizione di destinazione).
- Per aggiungere una trasformazione allo script, scegli Transform (Trasformazione). Per informazioni sulle funzioni richiamate nel tuo script, consulta Programmazione di script ETL AWS Glue in PySpark.
- Per aggiungere una trasformazione Spigot allo script, scegli Spigot.

Nel codice inserito, modifica i `parameters` nelle annotazioni e nel codice Apache Spark. Ad esempio, se aggiungi una trasformazione Spigot, verifica che `path` sia sostituito sia nella riga di annotazione `@args` sia nella riga di codice `output`.

La scheda Logs (Log) mostra i log che sono associati al tuo processo durante l'esecuzione. Vengono visualizzate le 1.000 righe più recenti.

La scheda Schema mostra lo schema delle origini e delle destinazioni selezionate, quando disponibili nel catalogo dati.

Monitoraggio dei dati elaborati mediante segnalibri di processo

AWS Glue monitora i dati già elaborati durante una precedente esecuzione di un processo ETL conservando le informazioni sullo stato dall'esecuzione del processo. Questa informazione sullo stato persistente è chiamato segnalibro di processo. I segnalibri del processo aiutano AWS Glue a mantenere le informazioni sullo stato e prevenire la rielaborazione dei dati precedenti. Con i segnalibri del processo, è possibile elaborare nuovi dati quando vengono rieseguiti su un intervallo pianificato. Un segnalibro di un processo è costituito dagli stati dei vari elementi dei processi, come le origini, le trasformazioni e le destinazioni. Ad esempio, il processo ETL potrebbe leggere nuove partizioni in un file Amazon S3. AWS Glue tiene traccia delle partizioni che sono state elaborate correttamente dal processo al fine di evitare elaborazioni ripetute o dati duplicati nel datastore di destinazione del processo.

I segnalibri di processo sono implementati per le origini dati JDBC, la trasformazione Relationalize e alcune origini Amazon Simple Storage Service (Amazon S3). La tabella seguente elenca i formati di origine Amazon S3 supportati da AWS Glue per i segnalibri del processo.

Versione AWS Glue	Formati Amazon S3 di origine
Versione 0.9	JSON, CSV, Apache Avro, XML
Versione 1.0 e successive.	JSON, CSV, Apache Avro, XML, Parquet, ORC

Per ulteriori informazioni sulle versioni AWS Glue, consulta Definire le proprietà di processo per i processi Spark.

La funzionalità dei segnalibri di processo offre opzioni aggiuntive quando si accede tramite script AWS Glue. Quando si sfoglia lo script generato, è possibile visualizzare contesti di trasformazione correlati a questa funzionalità. Per ulteriori informazioni, consulta the section called “Utilizzo di segnalibri di processo”.

Argomenti

Utilizzo di segnalibri di processo in AWS Glue

L'opzione di segnalibro di processo viene passata come parametro all'avvio del processo. La tabella seguente descrive le opzioni per impostare i segnalibri di processo nella console AWS Glue.

Segnalibro di processo	Descrizione
Attiva	Provoca l'aggiornamento dello stato del processo dopo un'esecuzione per tenere traccia dei dati elaborati in precedenza. Se il processo ha un'origine con un supporto segnalibro del processo, questo tiene traccia dei dati elaborati e, quando un processo viene eseguito, elabora i nuovi dati a partire dall'ultimo punto di controllo.
Disabilita	I segnalibri del processo non vengono utilizzati e il processo elabora sempre l'intero set di dati. Sei responsabile della gestione dell'outp ut dalle esecuzioni dei processi precedenti. Questa è l'impostazione predefinita.
Metti in pausa	Elabora i dati incrementali dall'ultima esecuzione riuscita o i dati nell'inte rvallo identificato dalle seguenti opzioni secondarie, senza aggiornare lo stato dell'ultimo segnalibro. Sei responsabile della gestione dell'outp ut dalle esecuzioni dei processi precedenti. Le due opzioni secondarie sono: • job-bookmark-from <from-value> è l'ID di esecuzione che rappresen ta tutto l'input che è stato elaborato fino all'ultima esecuzione riuscita prima, incluso l'ID di esecuzione specificato. L'input corrispondente viene ignorato. • job-bookmark-to <to-value> è l'ID di esecuzione che rappresenta tutto l'input che è stato elaborato fino all'ultima esecuzione riuscita prima, incluso l'ID di esecuzione specificato. L'input corrispondente escluso l'input identificato da <from-value> viene elaborato dal processo. Qualsiasi input successivo a questo è escluso anche dall'elaborazione.

Segnalibro di processo	Descrizione
	Lo stato dei segnalibri di processo non viene aggiornato quando viene specificato questo set di opzioni. Le opzioni secondarie sono facoltative, tuttavia se utilizzate, entrambe le opzioni secondarie devono essere specificate.

Per informazioni dettagliate sui parametri passati a un processo nella riga di comando e, in particolare, sui segnalibri di lavoro, consulta Parametri del processo AWS Glue.

Per le origini di input Amazon S3, i segnalibri del processo AWS Glue controllano l'ora dell'ultima modifica degli oggetti per individuare gli oggetti da rielaborare. Se i dati dell'origine di input sono stati modificati dall'ultima esecuzione del processo, i file vengono rielaborati alla nuova esecuzione del processo.

Per le origini JDBC, si applicano le seguenti regole:

- Per ogni tabella, AWS Glue utilizza una o più colonne come chiavi di segnalibro per determinare i dati nuovi ed elaborati. Le chiavi di segnalibro si combinano per formare una singola chiave composta.
- Per impostazione predefinita, AWS Glue utilizza la chiave primaria come chiave di segnalibro, a condizione che sia in ordine sequenziale (senza spazi vuoti) crescente o decrescente.
- Nello script AWS Glue è possibile specificare le colonne da utilizzare come chiavi di segnalibro. Per ulteriori informazioni sull'utilizzo dei segnalibri di processo negli script AWS Glue, consulta la pagina the section called “Utilizzo di segnalibri di processo”.
- AWS Glue non supporta l'utilizzo di colonne che fanno distinzione tra maiuscole e minuscole come chiavi di segnalibro del processo.

È possibile ripristinare i segnalibri di processo per i processi ETL di AWS Glue Spark a qualsiasi esecuzione precedente del processo. È possibile supportare meglio gli scenari di recupero dei dati ripristinando i segnalibri di lavoro a qualsiasi esecuzione di lavoro precedente, con conseguente esecuzione del lavoro di rielaborazione dei dati solo relativi all'esecuzione del lavoro con segnalibro.

Se hai intenzione di rielaborare tutti i dati utilizzando lo stesso processo, reimposta il segnalibro del processo. Per reimpostare lo stato del segnalibro del processo, utilizza la console AWS Glue,

l'operazione API ResetJobBookmark azione (Python: reset_job_bookmark) o la AWS CLI. Ad esempio, inserisci il seguente comando utilizzando AWS CLI:

```
aws glue reset-job-bookmark --job-name my-job-name
```

Quando si riavvolge o si reimposta un segnalibro, AWS Glue non pulisce i file di destinazione, perché potrebbero esserci più destinazioni e le destinazioni non sono monitorate con i segnalibri di processo. Solo i file di origine vengono tracciati con i segnalibri di processo. È possibile creare diverse destinazioni dell'output durante il riavvolgimento e la rielaborazione dei file di origine per evitare la duplicazione dei dati nell'output.

AWS Glue tiene traccia dei segnalibri dei processi per ogni processo. All'eliminazione del processo, seguirà l'eliminazione del segnalibro di processo.

In alcuni casi, potresti aver abilitato i segnalibri del processo AWS Glue, ma il processo ETL rielabora i dati già elaborati in una esecuzione precedente. Per ulteriori informazioni sulla risoluzione delle cause comuni di questo errore, consulta Risoluzione degli errori in AWS Glue per Spark.

Dettagli operativi della funzione dei segnalibri di processo

Questa sezione descrive ulteriori dettagli operativi utilizzando i segnalibri del processo.

I segnalibri del processo archiviano gli stati per un processo. Ogni istanza dello stato viene contrassegnata da un nome processo e da un numero di versione. Quando uno script richiama `job.init`, ne recupera lo stato e ottiene sempre la versione più recente. Uno stato contiene più elementi dello stato, specifici per ogni origine, trasformazione e istanza sink nello script. Gli elementi dello stato sono identificati da un contesto di trasformazione collegato all'elemento corrispondente (origine, trasformazione o sink) nello script. Gli elementi dello stato vengono salvati in modo atomico quando `job.commit` viene richiamato dallo script dell'utente. Lo script ottiene dagli argomenti il nome del processo e l'opzione di controllo per i segnalibri del processo.

Gli elementi dello stato nel segnalibro del processo sono origine, trasformazione oppure dati specifici del sink. Ad esempio, supponiamo che si desideri leggere i dati incrementali da una posizione Amazon S3 costantemente scritta da un processo upstream o da un processo. In questo caso, lo script deve determinare quanto è stato elaborato fino a ora. L'implementazione del segnalibro del processo per l'origine Amazon S3 salva le informazioni in modo che, quando il processo viene eseguito nuovamente, è possibile filtrare solo i nuovi oggetti utilizzando le informazioni salvate e

ricalcolare lo stato per l'esecuzione successiva del processo. Un timestamp viene utilizzato per filtrare i nuovi file.

In aggiunta agli elementi dello stato, i segnalibri del processo dispongono di un numero di esecuzione, un numero di tentativo e un numero di versione. Il numero di esecuzione monitora l'esecuzione del processo e il numero di tentativo registra i tentativi di esecuzione di un processo. Il numero di esecuzione di un processo è un numero che aumenta in maniera monotona e che subisce incrementi a ogni esecuzione riuscita. Il numero di tentativi monitora i tentativi per ogni esecuzione e viene incrementato solo in caso di un'esecuzione dopo un tentativo non riuscito. Il numero di versione aumenta in maniera monotona e monitora gli aggiornamenti a un segnalibro del processo.

Nel database di servizio AWS Glue, gli stati dei segnalibri per tutte le trasformazioni sono memorizzati insieme come coppie chiave-valore:

```
{
  "job_name" : ...,
  "run_id": ...,
  "run_number": ..,
  "attempt_number": ...
  "states": {
    "transformation_ctx1" : {
      bookmark_state1
    },
    "transformation_ctx2" : {
      bookmark_state2
    }
  }
}
```

Best practice

Di seguito sono indicate le best practice per l'utilizzo dei segnalibri di processo.

- Non modificare la proprietà dell'origine dati con il segnalibro abilitato. Ad esempio, esiste un datasource0 che punta a un percorso di input di Amazon S3 A e il lavoro è stato letto da un'origine che è in esecuzione per diversi round con il segnalibro abilitato. Se modifichi il percorso di input di datasource0 nel percorso Amazon S3 B senza modificare il `transformation_ctx`, il processo AWS Glue utilizzerà il vecchio stato dei segnalibri memorizzati. Questo si tradurrà in file mancanti o saltati nel percorso di input B, in quanto AWS Glue presuppone che quei file siano stati elaborati nelle esecuzioni precedenti.

- Utilizzare una tabella di catalogo con segnalibri per una migliore gestione delle partizioni. I segnalibri funzionano sia per le origini dati del Catalogo dati che per le opzioni. Tuttavia, è difficile rimuovere/aggiungere nuove partizioni con l'approccio dalle opzioni. L'utilizzo di una tabella di catalogo con crawler può fornire una migliore automazione per tracciare le partizioni appena aggiunte e ti offre la flessibilità necessaria per selezionare partizioni particolari con Predicato pushdown.
- Utilizzo dell'elenco di file AWS Glue Amazon S3 per set di dati di grandi dimensioni. Un segnalibro elenca tutti i file sotto ogni partizione di input e esegue il filering, quindi se ci sono troppi file sotto una singola partizione il segnalibro può essere eseguito nel driver OOM. Utilizzo dell'elenco di file AWS Glue Amazon S3 per evitare di elencare tutti i file in memoria contemporaneamente.

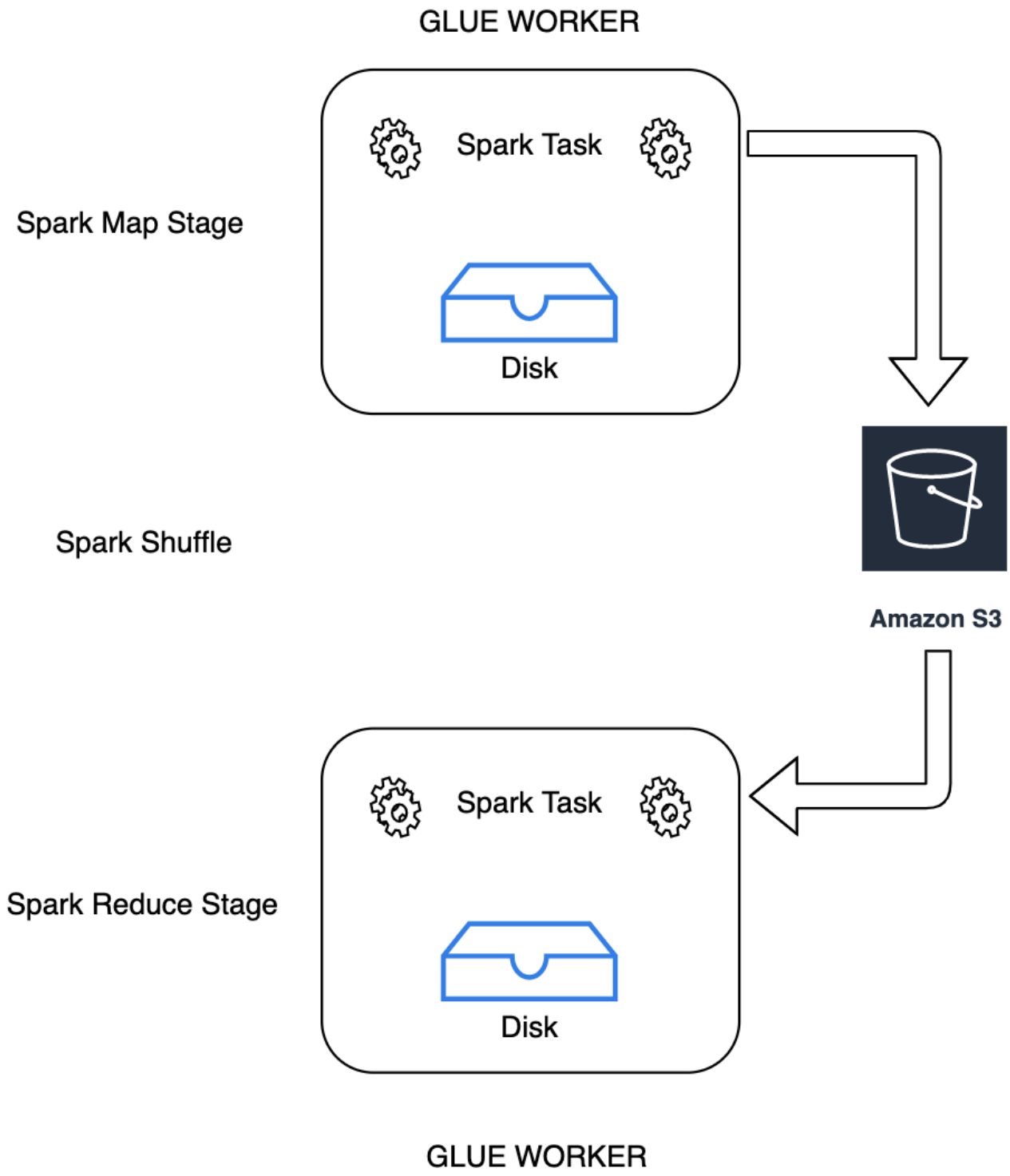
Plug-in shuffle di AWS Glue Spark con Amazon S3

Lo shuffle rappresenta un passaggio importante in un processo Spark quando i dati vengono riorganizzati tra le partizioni. È necessario perché trasformazioni estese, come `join`, `groupByKey`, `reduceByKey` e `repartition`, hanno bisogno di informazioni da altre partizioni per completare l'elaborazione. Spark raccoglie i dati richiesti da ciascuna partizione e li combina in una nuova partizione. Durante uno shuffle, i dati vengono scritti su disco e trasferiti attraverso la rete. Di conseguenza, l'operazione di shuffle è legata alla capacità del disco locale. Spark genera un errore `No space left on device` o `MetadataFetchFailedException` quando sull'executor non è rimasto sufficiente spazio su disco e non vi è un ripristino.

> **Note**
>
> Il plug-in di shuffle di AWS Glue Spark con Amazon S3 è supportato solo per i processi ETL diAWS Glue.

Soluzione

Con AWS Glue, è ora possibile usare Amazon S3 per archiviare i dati shuffle Spark. Amazon S3 è un servizio di archiviazione di oggetti che offre scalabilità, disponibilità dei dati, sicurezza e prestazioni tra le migliori del settore. Questa soluzione disaggrega calcolo e storage per i processi Spark e offre elasticità completa e storage per lo shuffle a basso costo, consentendo di eseguire in modo affidabile i carichi di lavoro con shuffle intensivo.



Stiamo introducendo un nuovo plug-in di archiviazione cloud shuffle per Apache Spark per utilizzare Amazon S3. Puoi attivare lo shuffle Amazon S3 per eseguire i processi AWS Glue in modo affidabile senza errori se sono legati alla capacità del disco locale per operazioni di shuffle di grandi dimensioni.

In alcuni casi, lo shuffle su Amazon S3 è leggermente più lento del disco locale (o EBS) se si dispone di un numero elevato di piccole partizioni o file shuffle scritti su Amazon S3.

Prerequisiti per l'utilizzo del plug-in Cloud Shuffle Storage

Per utilizzare il plug-in Cloud Shuffle Storage con i processi ETL di AWS Glue, sono richieste le seguenti informazioni:

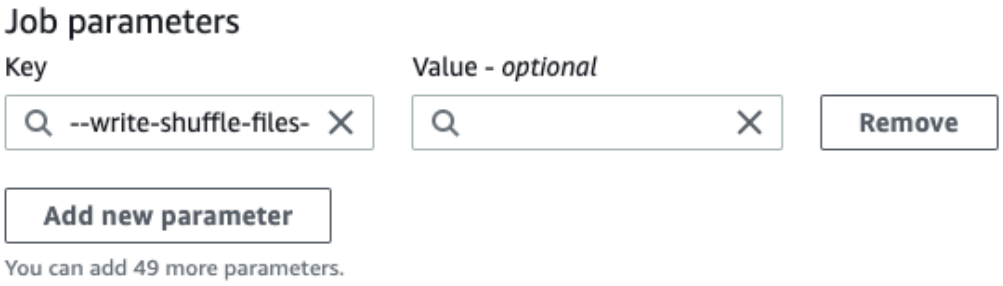
- Un bucket Amazon S3 situato nella stessa regione in cui viene eseguito il processo, per archiviare i dati intermedi e i dati riversati. Il prefisso Amazon S3 dell'archiviazione con shuffle può essere specificato con `--conf spark.shuffle.glue.s3ShuffleBucket=s3://`*`shuffle-bucket`*`/`*`prefix`*`/`, come nell'esempio seguente:

```
--conf spark.shuffle.glue.s3ShuffleBucket=s3://glue-shuffle-123456789-us-east-1/glue-shuffle-data/
```

- Imposta le policy del ciclo di vita dell'archiviazione di Amazon S3 sul prefisso (come `glue-shuffle-data`), poiché lo shuffle manager non pulisce i file al termine del processo. Lo shuffle intermedio e i dati riversati devono essere eliminati al termine di un processo. Gli utenti possono impostare policy del ciclo di vita breve sul prefisso. Le istruzioni per configurare il ciclo di vita per Amazon S3 sono disponibili nella sezione Setting lifecycle configuration on a bucket nella Guida per l'utente di Amazon Simple Storage Service.

Utilizzo dello shuffle manager di AWS Glue Spark dalla console AWS

Per impostare lo shuffle manager di AWS Glue Spark utilizzando la console AWS Glue o AWS Glue Studio durante la configurazione di un processo: scegli il parametro --write-shuffle-files-to-s3 per attivare lo shuffle su Amazon S3 per il processo.



Utilizzo del plug-in shuffle di AWS Glue Spark

I seguenti parametri di processo attivano e regolano AWS Glue shuffle manager. Questi parametri sono flag, quindi i valori forniti non vengono considerati.

- `--write-shuffle-files-to-s3`: il flag principale, che abilita lo shuffle manager di AWS Glue Spark all'utilizzo dei bucket Amazon S3 per scrivere e leggere i dati shuffle. Quando il flag non è specificato, lo shuffle manager non viene utilizzato.
- `--write-shuffle-spills-to-s3`: (supportato solo in AWS Glue versione 2.0). Un flag facoltativo che consente di scaricare i file riversati nei bucket Amazon S3, fornendo ulteriore resilienza al processo Spark. Questo è necessario solo per carichi di lavoro di grandi dimensioni che riversano molti dati sul disco. Quando il flag non è specificato, non viene scritto alcun file riversato intermedio.
- `--conf spark.shuffle.glue.s3ShuffleBucket=s3://<shuffle-bucket>` — Un altro flag opzionale che specifica il bucket Amazon S3 in cui vengono scritti i file shuffle. Per impostazione predefinita, `--TempDir`/shuffle-data. AWS Glue versione 3.0 e successive supporta la scrittura di file shuffle su più bucket specificando i bucket con un delimitatore di virgola, come in `--conf spark.shuffle.glue.s3ShuffleBucket=s3://`*`shuffle-bucket-1`*`/`*`prefix`*`,s3://`*`shuffle-bucket-2`*`/`*`prefix`*`/`. L'utilizzo di più bucket migliora le prestazioni.

Per abilitare la crittografia a riposo per i dati shuffle, fornisci le impostazioni di configurazione della sicurezza. Per ulteriori informazioni sulle configurazioni di sicurezza, consulta the section called “Configurazione della crittografia”. AWS Glue supporta tutte le altre configurazioni relative ai dati shuffle fornite da Spark.

File binari software per il plug-in di archiviazione cloud shuffle

Puoi anche scaricare i file binari software del plug-in di archiviazione cloud shuffle per Apache Spark con la licenza Apache 2.0 ed eseguirli in qualsiasi ambiente Spark. Il nuovo plug-in include il supporto predefinito per Amazon S3 e può essere facilmente configurato per utilizzare anche altre forme di archiviazione cloud come Google Cloud Storage e Microsoft Azure Blob Storage. Per ulteriori informazioni, consulta Plug-in di archiviazione cloud shuffle per Apache Spark.

Note e limitazioni

Di seguito sono riportate note o limitazioni per AWS Glue shuffle manager:

- Lo shuffle manager di AWS Glue non elimina automaticamente i file di dati shuffle (temporanei) archiviati nel bucket Amazon S3 dopo il completamento di un processo. Per garantire la protezione dei dati, segui le istruzioni riportate nella sezione Prerequisiti per l'utilizzo del plug-in Cloud Shuffle Storage prima di abilitare il plug-in Cloud Shuffle Storage.
- È possibile usare questa funzionalità se i dati sono asimmetrici.

Plug-in di archiviazione cloud shuffle per Apache Spark

Il plug-in di archiviazione cloud shuffle è un plug-in Apache Spark compatibile con l'API `ShuffleDataIO` che consente di archiviare dati shuffle su sistemi di archiviazione cloud (come Amazon S3). Consente di integrare o sostituire la capacità di archiviazione locale su disco per operazioni shuffle di grandi dimensioni, normalmente innescate da trasformazioni come `join`, `reduceByKey`, `groupByKey` e `repartition` nelle applicazioni Spark, riducendo così i guasti più comuni o la dislocazione prezzo/prestazioni dei processi e delle pipeline di analisi dei dati serverless.

AWS Glue

Le versioni 3.0 e 4.0 di AWS Glue includono il plug-in preinstallato e pronto per abilitare lo shuffling su Amazon S3 senza passaggi aggiuntivi. Per ulteriori informazioni, consulta Plug-in shuffle di AWS Glue Spark con Amazon S3 per abilitare la funzionalità per le tue applicazioni Spark.

Altri ambienti Spark

Il plug-in richiede che in altri ambienti Spark siano impostate le seguenti configurazioni Spark:

- `--conf spark.shuffle.sort.io.plugin.class=com.amazonaws.spark.shuffle.io.cloud.Choppe` indica a Spark di utilizzare questo plug-in per Shuffle IO.
- `--conf spark.shuffle.storage.path=s3://`*`bucket-name`*`/`*`shuffle-file-dir`*: il percorso in cui verranno archiviati i file shuffle.

> **Note**
>
> Il plug-in sovrascrive una classe principale di Spark. Di conseguenza, il jar del plug-in deve essere caricato prima dei jar di Spark. Ciò è possibile utilizzando `userClassPathFirst` in ambienti YARN on-premise se il plug-in viene utilizzato all'esterno di AWS Glue.

Creazione di bundle per il plug-in con le applicazioni Spark

È possibile raggruppare il plug-in con le applicazioni Spark e le distribuzioni Spark (versioni 3.1 e successive) aggiungendo la dipendenza del plug-in nel file Mavenpom.xml mentre si sviluppano le applicazioni Spark in locale. Per ulteriori informazioni sulle versioni del plug-in e di Spark, consulta Versioni del plug-in.

```
<repositories>
   ...
    <repository>
        <id>aws-glue-etl-artifacts</id>
        <url>https://aws-glue-etl-artifacts.s3.amazonaws.com/release/ </url>
    </repository>
</repositories>
...
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>chopper-plugin</artifactId>
    <version>3.1-amzn-LATEST</version>
</dependency>
```

In alternativa, è possibile scaricare i binari direttamente dagli artefatti di AWS Glue Maven e includerli nell'applicazione Spark come riportato di seguito.

```
#!/bin/bash
sudo wget -v https://aws-glue-etl-artifacts.s3.amazonaws.com/release/com/amazonaws/
chopper-plugin/3.1-amzn-LATEST/chopper-plugin-3.1-amzn-LATEST.jar -P /usr/lib/spark/
jars/
```

Esempio spark-submit

```
spark-submit --deploy-mode cluster \
--conf spark.shuffle.storage.s3.path=s3://<ShuffleBucket>/<shuffle-dir> \
--conf spark.driver.extraClassPath=<Path to plugin jar> \
--conf spark.executor.extraClassPath=<Path to plugin jar> \
--class <your test class name> s3://<ShuffleBucket>/<Your application jar> \
```

Configurazioni facoltative

Questi sono i valori delle configurazioni facoltative che controllano il comportamento dello shuffle di Amazon S3.

- `spark.shuffle.storage.s3.enableServerSideEncryption`: abilita/disabilita S3 SSE per i file shuffle e spill. Il valore predefinito è `true`.
- `spark.shuffle.storage.s3.serverSideEncryption.algorithm`: l'algoritmo SSE da utilizzare. Il valore predefinito è `AES256`.
- `spark.shuffle.storage.s3.serverSideEncryption.kms.key`: l'ARN della chiave KMS quando è abilitato SSE aws:kms.

Oltre a queste configurazioni, potrebbe essere necessario impostarne altre come `spark.hadoop.fs.s3.enableServerSideEncryption` e configurazioni aggiuntive specifiche dell'ambiente per garantire l'applicazione della crittografia appropriata per il caso d'uso.

Versioni del plug-in

Questo plug-in è supportato per le versioni Spark associate a ogni versione di AWS Glue. La tabella seguente mostra la versione di AWS Glue, la versione di Spark e la versione del plug-in associata alla posizione di Amazon S3 per il file binario del software del plug-in.

Versione AWS Glue	Versione di Spark	Versione del plug-in	Posizione di Amazon S3.
3.0	3.1	3.1-amzn-LATEST	s3://aws-glue-etl-artifacts/release/com/amazonaws/chopper-plugin/3.1-amzn-0/chopper-plugin-3.1-amzn-LATEST.jar
4.0	3.3	3.3-amzn-LATEST	s3://aws-glue-etl-artifacts/release/com/amazonaws/chopper-plugin/3.3-amzn-0/chopper-plugin-3.3-amzn-LATEST.jar

Licenza

Il software del plug-in è concesso in licenza ai sensi della licenza Apache-2.0.

Monitoraggio dei processi Spark AWS Glue

Argomenti

Parametri di Spark disponibili in AWS Glue Studio

La scheda Metrics (Parametri) mostra i parametri raccolti quando un processo viene eseguito ed è attivata la profilatura. Nei processi Spark vengono visualizzati i grafici seguenti:

- Spostamento di dati ETL
- Profilo di memoria: driver ed executor

Scegli View additional metrics (Visualizza parametri aggiuntivi) per visualizzare i grafici relativi agli elementi seguenti:

- Spostamento di dati ETL
- Profilo di memoria: driver ed executor
- Distribuzione casuale dei dati tra executor
- Carico CPU: driver ed executor
- Esecuzione del processo: executor attivi, fasi completate e numero massimo di executor necessari

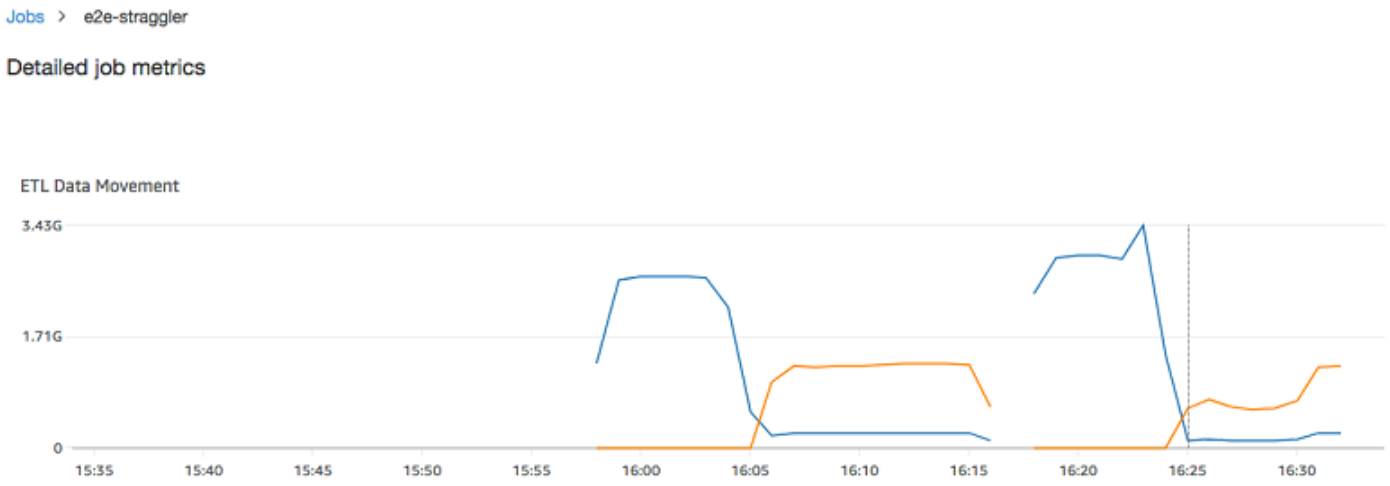
I dati per questi grafici vengono inviati ai parametri CloudWatch se per il processo è abilitata la raccolta dei parametri. Per ulteriori informazioni su come abilitare i parametri e interpretare i grafici, consulta Monitoraggio e debug dei processi.

Example Grafico relativo allo spostamento di dati ETL

Il grafico relativo allo spostamento di dati ETL mostra i parametri seguenti:

- Numero di byte letti da Amazon S3 da tutti gli executor: `glue.ALL.s3.filesystem.read_bytes`

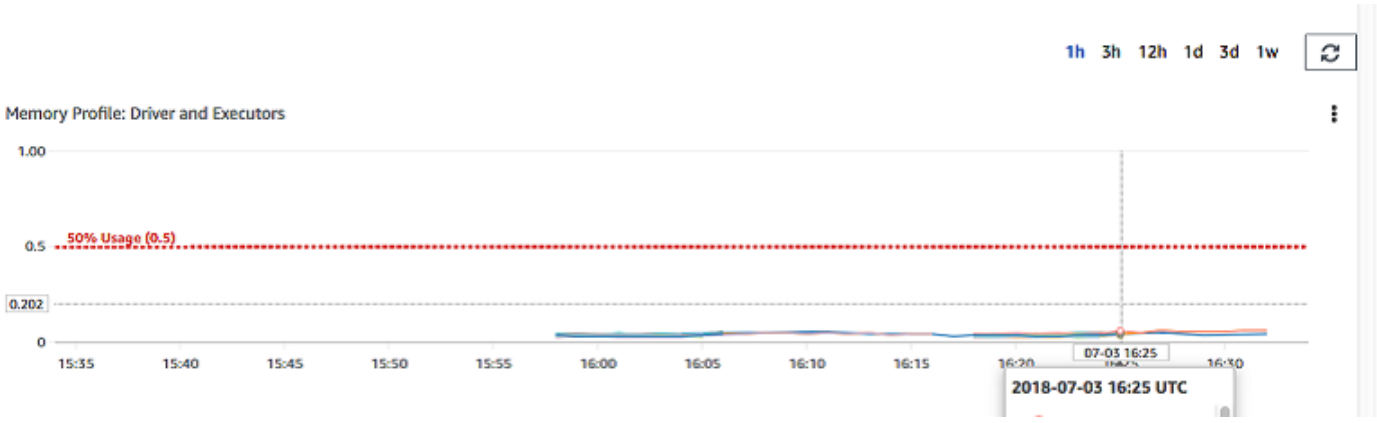
- Numero di byte scritti in Amazon S3 da tutti gli executor: `glue.ALL.s3.filesystem.write_bytes`



Example Grafico relativo al profilo di memoria

Il grafico relativo al profilo di memoria mostra i parametri seguenti:

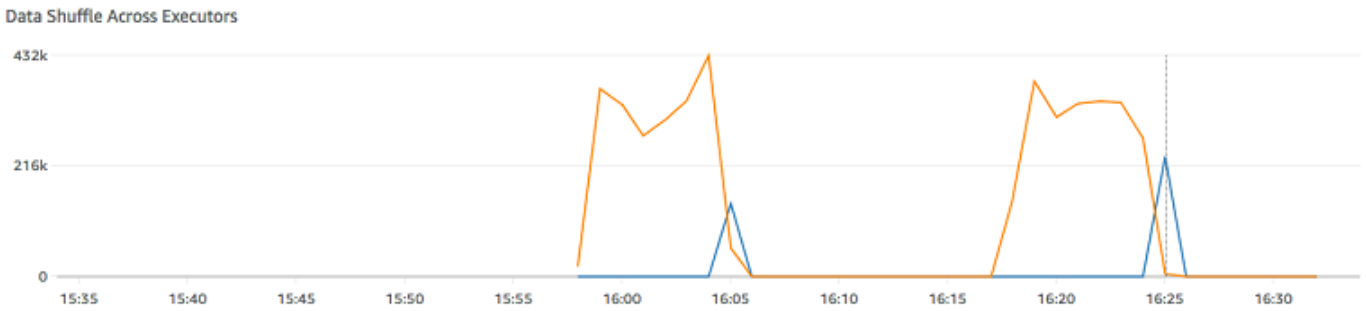
- Frazione di memoria usata dall'heap JVM per questo driver (dimensione: 0-1) dal driver, da un executor identificato da executorId o da tutti gli executor—
  - `glue.driver.jvm.heap.usage`
  - `glue.executorId.jvm.heap.usage`
  - `glue.ALL.jvm.heap.usage`



Example Grafico relativo alla distribuzione casuale dei dati tra executor

Il grafico relativo alla distribuzione casuale dei dati tra executor mostra i parametri seguenti:

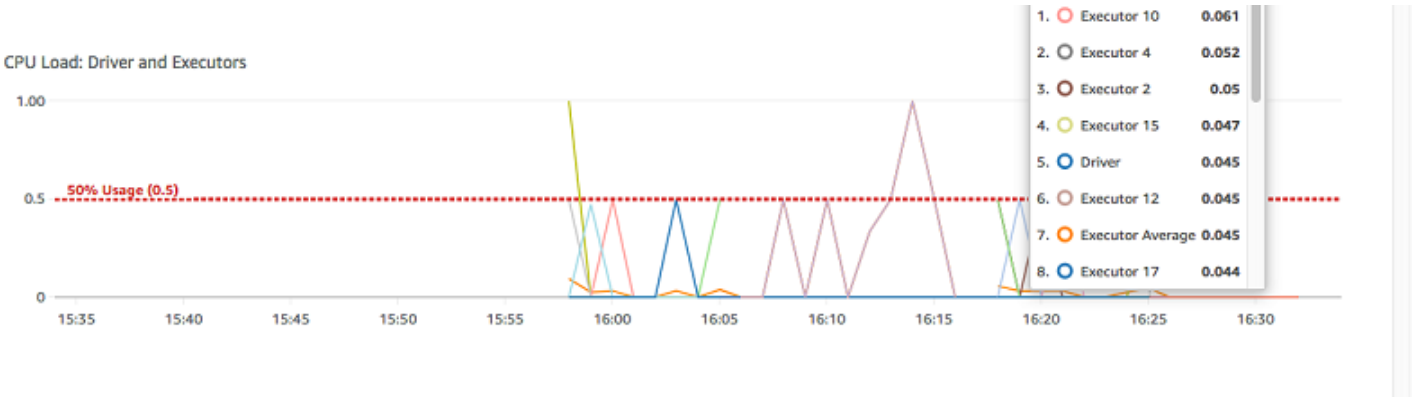
- Numero di byte letti da tutti gli executor per distribuire i dati in modo casuale: `glue.driver.aggregate.shuffleLocalBytesRead`
- Numero di byte scritti da tutti gli executor per distribuire i dati in modo casuale: `glue.driver.aggregate.shuffleBytesWritten`



Example Grafico relativo al carico CPU

Il grafico relativo al carico CPU mostra i parametri seguenti:

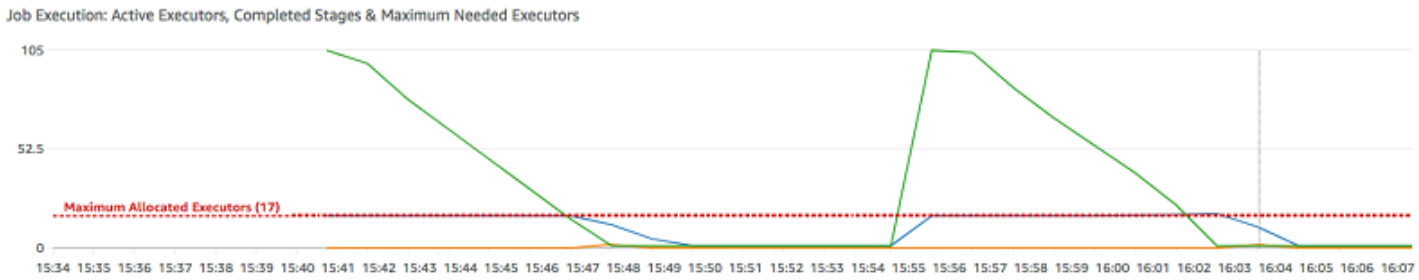
- Frazione del carico di sistema della CPU usata (dimensione: 0-1) dal driver, da un executor identificato da executorId o da tutti gli executor:
  - `glue.driver.system.cpuSystemLoad`
  - `glue.executorId.system.cpuSystemLoad`
  - `glue.ALL.system.cpuSystemLoad`



Example Grafico relativo all'esecuzione del processo

Il grafico relativo all'esecuzione del processo mostra i parametri seguenti:

- Numero di executor attivamente in esecuzione: `glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`
- Numero di fasi completate: `glue.aggregate.numCompletedStages`
- Numero massimo di executor necessari: `glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors`



Monitoraggio dei processi tramite l'interfaccia utente Web di Apache Spark

Puoi utilizzare l'interfaccia utente Web di Apache Spark per monitorare ed eseguire il debug dei processi ETL AWS Glue in esecuzione sul sistema di processi AWS Glue e anche delle applicazioni Spark in esecuzione sugli endpoint di sviluppo AWS Glue. L'interfaccia utente di Spark consente di controllare quanto segue per ogni processo:

- Tempistica eventi di ogni fase Spark
- Un grafo aciclico orientato (DAG) del processo
- Piani fisici e logici per le query SparkSQL
- Le variabili ambientali Spark sottostanti per ogni processo

Per ulteriori informazioni sull'utilizzo dell'interfaccia utente Web di Spark, consulta l'interfaccia utente Web nella documentazione di Spark. Per indicazioni su come interpretare i risultati dell'interfaccia utente di Spark per migliorare le prestazioni del tuo processo, consulta Best practice per l'ottimizzazione delle prestazioni per i processi di AWS Glue per Apache Spark nel Prontuario AWS.

Puoi vedere l'interfaccia utente di Spark nella console AWS Glue per i processi più recenti. È disponibile quando un processo AWS Glue viene eseguito su AWS Glue versione 3.0 o successive con log generati nel formato Standard (anziché legacy), che è l'impostazione predefinita per i processi più recenti. Per ulteriori informazioni su come trovare l'interfaccia utente di Spark nella console, consulta the section called “Visualizzare le informazioni sulle esecuzioni dei processi recenti”. Per le altre versioni di AWS Glue, dovrai predisporre il tuo server di cronologia. Per ulteriori informazioni, consulta the section called “Avvio del server della cronologia di Spark”.

Puoi abilitare l'interfaccia utente di Spark utilizzando la console AWS Glue o AWS Command Line Interface (AWS CLI). Quando abiliti l'interfaccia utente di Spark, i processi ETL AWS Glue e le applicazioni Spark su endpoint di sviluppo AWS Glue possono eseguire il backup dei log degli eventi Spark in un percorso specificato in Amazon Simple Storage Service (Amazon S3). Puoi utilizzare i log degli eventi sottoposti a backup in Amazon S3 con l'interfaccia utente di Spark sia in tempo reale, ovvero durante l'esecuzione del processo, sia al termine dello stesso. Se i log rimangono in Amazon S3, l'interfaccia utente di Spark nella console AWS Glue è in grado di visualizzarli.

Per utilizzare l'interfaccia utente di Spark nella console AWS Glue, il ruolo della console deve disporre dell'autorizzazione `glue:UseGlueStudio`. Per ulteriori informazioni su questa autorizzazione, consulta the section called “Creazione di criteri IAM personalizzati per AWS Glue Studio”.

Limitazioni:

- L'interfaccia utente di Spark nella console AWS Glue non è disponibile per le esecuzioni di processi avvenute prima del 20 novembre 2023, poiché sono nel formato log precedente.
- L'interfaccia utente di Spark nella console AWS Glue non supporta i log in sequenza, come quelli generati per impostazione predefinita nei processi di streaming.

  Puoi disattivare i log in sequenza per un processo di streaming inserendo una configurazione aggiuntiva. Tieni presente che la manutenzione di file di log molto grandi può essere costosa.

  Per disattivare i log in sequenza, fornisci la seguente configurazione:

```
'--spark-ui-event-logs-path': 'true',
'--conf': 'spark.eventLog.rolling.enabled=false'
```

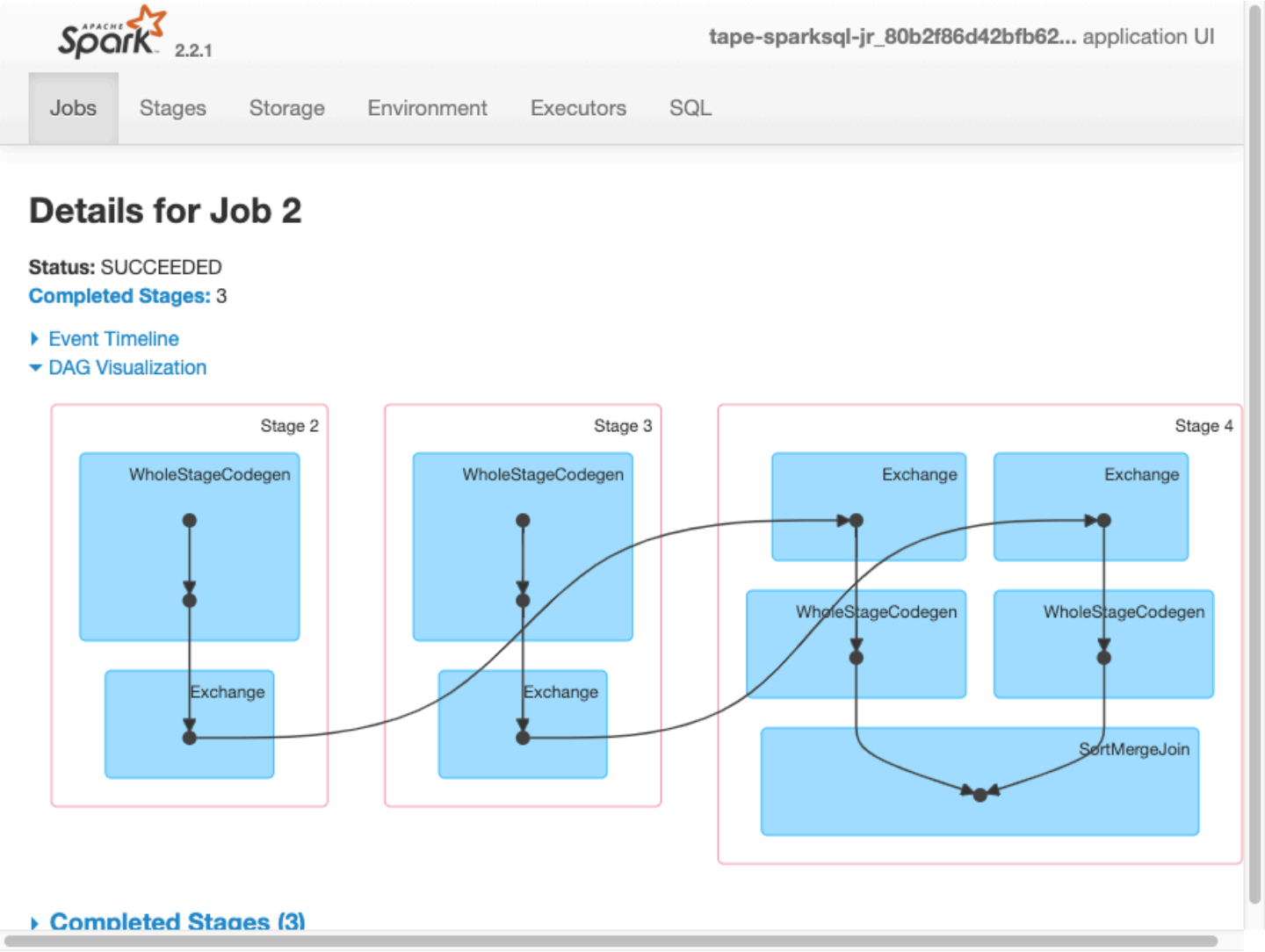
Esempio: interfaccia utente Web di Apache Spark

Questo esempio illustra come utilizzare l'interfaccia utente di Spark per comprendere le prestazioni del processo. Gli screenshot mostrano l'interfaccia utente Web di Spark fornita da un server della cronologia Spark autogestito. L'interfaccia utente di Spark nella console AWS Glue offre visualizzazioni simili. Per ulteriori informazioni sull'utilizzo dell'interfaccia utente Web di Spark, consulta l'interfaccia utente Web nella documentazione di Spark.

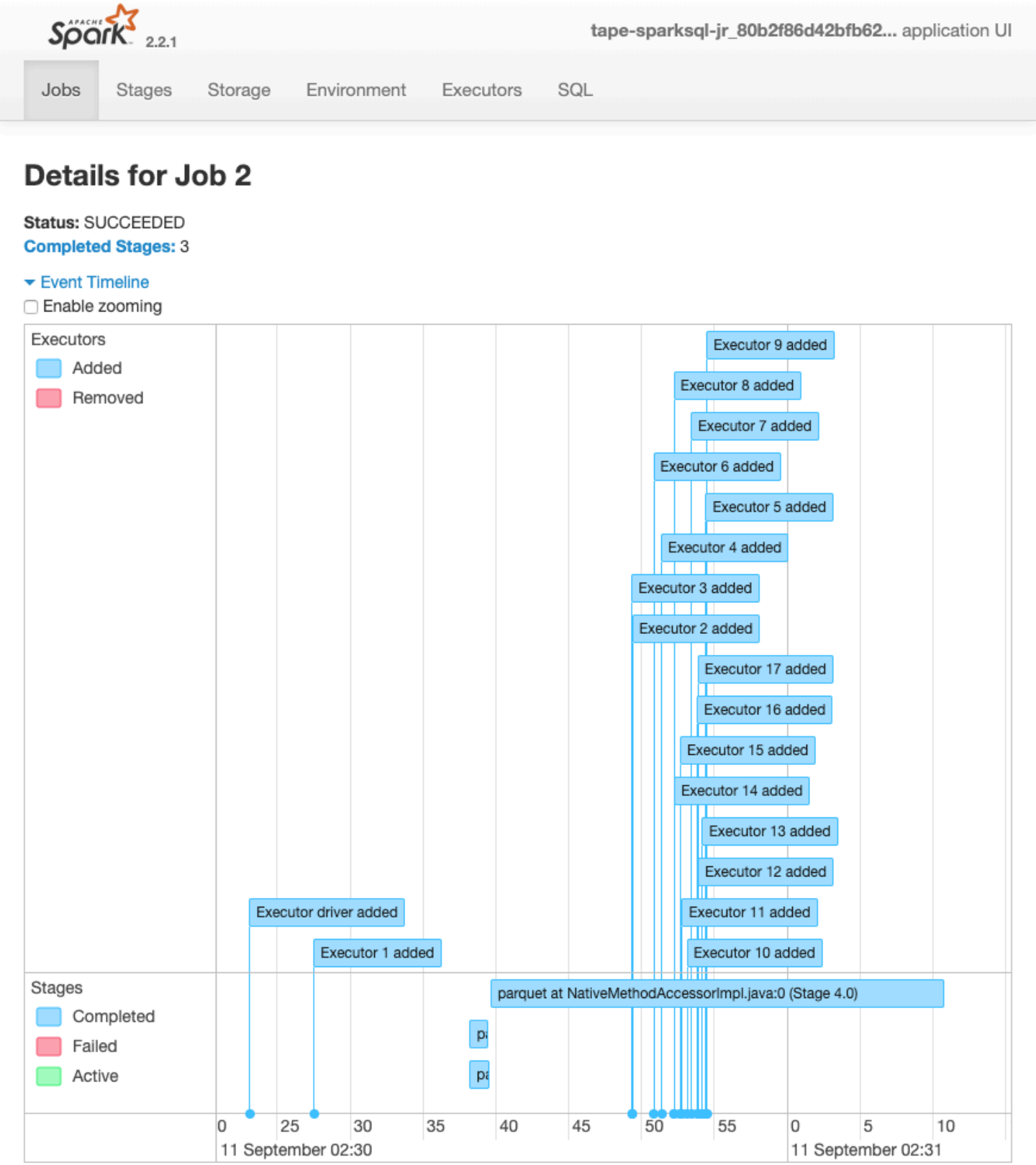
Di seguito è riportato un esempio di un'applicazione Spark che legge da due origini dati, esegue una trasformazione join e la scrive in Amazon S3 nel formato Parquet.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.functions import count, when, expr, col, sum, isnull
from pyspark.sql.functions import countDistinct
from awsglue.dynamicframe import DynamicFrame

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'])

df_persons = spark.read.json("s3://awsglue-datasets/examples/us-legislators/all/
persons.json")
df_memberships = spark.read.json("s3://awsglue-datasets/examples/us-legislators/all/
memberships.json")

df_joined = df_persons.join(df_memberships, df_persons.id == df_memberships.person_id,
 'fullouter')
df_joined.write.parquet("s3://aws-glue-demo-sparkui/output/")

job.commit()
```

La seguente visualizzazione DAG mostra le diverse fasi in questo processo Spark.



La seguente tempistica eventi per un processo mostra l'avvio, l'esecuzione e l'arresto di diversi executor Spark.


Spark 2.2.1
tape-sparksql-jr_80b2f86d42bfb62... application UI
Jobs
Stages
Storage
Environment
Executors
SQL
Details for Job 2
Status: SUCCEEDED
Completed Stages: 3
Event Timeline
Enable zooming
Executors
Added
Removed
Executor 9 added
Executor 8 added
Executor 7 added
Executor 6 added
Executor 5 added
Executor 4 added
Executor 3 added
Executor 2 added
Executor 17 added
Executor 16 added
Executor 15 added
Executor 14 added
Executor 13 added
Executor 12 added
Executor 11 added
Executor 10 added
Executor driver added
Executor 1 added
Stages
Completed
Failed
Active
parquet at NativeMethodAccessorImpl.java:0 (Stage 4.0)
0
25
30
35
40
45
50
55
0
5
10
11 September 02:30
11 September 02:31
DAG Visualization
Completed Stages (3)

La schermata seguente mostra i dettagli dei piani di query SparkSQL:

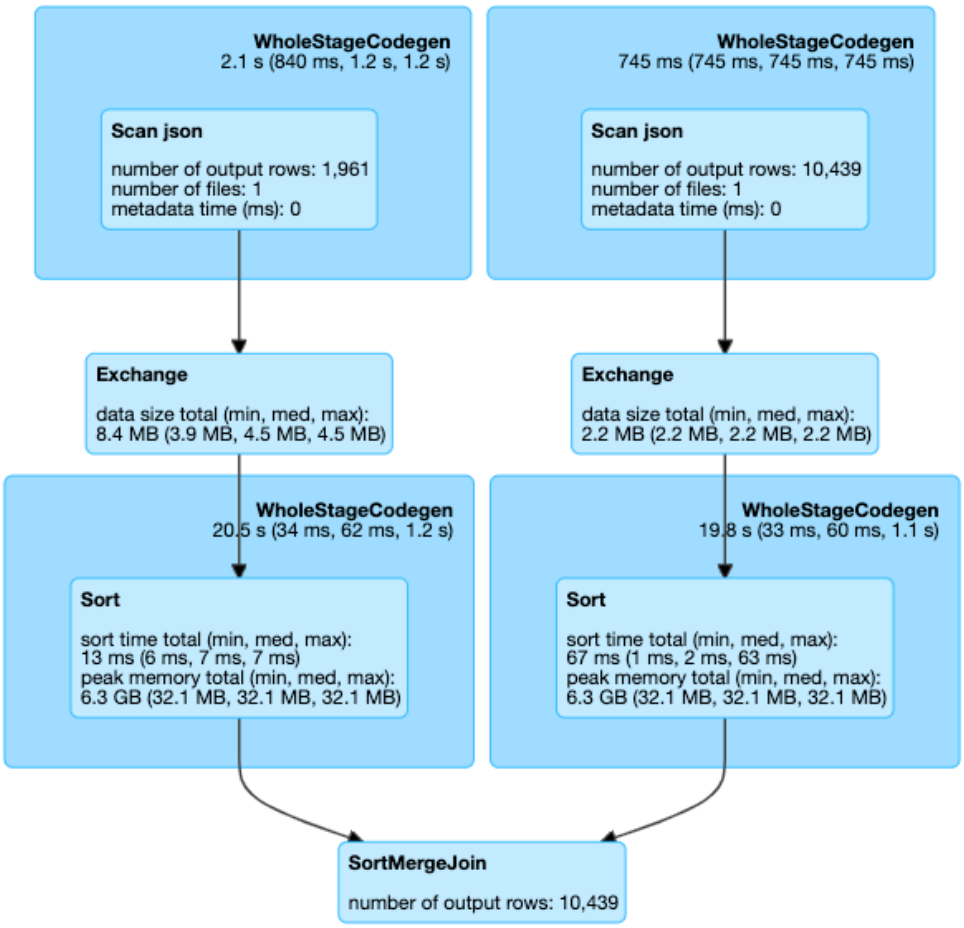
- Piano logico esaminato
- Piano logico analizzato
- Piano logico ottimizzato
- Piano fisico per l'esecuzione

Spark 2.2.1 tape-sparksql-jr_80b2f86d42bfb62... application UI

Jobs Stages Storage Environment Executors SQL

Details for Query 0

Submitted Time: 2019/09/11 02:30:37
Duration: 34 s
Succeeded Jobs: 2



▾ Details

```
== Parsed Logical Plan ==
Join FullOuter, (id#14 = person_id#50)
:-
Relation[birth_date#8,contact_details#9,death_date#10,family_name#11,gender#12,given_name#13,id#14,identifiers#15
,image#16,images#17,links#18,name#19,other_names#20,sort_name#21] json
+-
Relation[area_id#45,end_date#46,legislative_period_id#47,on_behalf_of_id#48,organization_id#49,person_id#50,role#
51,start_date#52] json

== Analyzed Logical Plan ==
birth_date: string, contact_details: array<struct<type:string,value:string>>, death_date: string, family_name:
string, gender: string, given_name: string, id: string, identifiers:
array<struct<identifier:string,scheme:string>>, image: string, images: array<struct<url:string>>, links:
array<struct<note:string,url:string>>, name: string, other_names:
array<struct<lang:string,name:string,note:string>>, sort_name: string, area_id: string, end_date: string,
legislative_period_id: string, on_behalf_of_id: string, organization_id: string, person_id: string, role: string,
start_date: string
Join FullOuter, (id#14 = person_id#50)
```

Argomenti

Abilitazione dell'interfaccia utente Web di Apache Spark per processi AWS Glue

Puoi utilizzare l'interfaccia utente Web di Apache Spark per monitorare ed eseguire il debug dei processi ETL AWS Glue in esecuzione sul sistema di processi AWS Glue. Puoi configurare l'interfaccia utente di Spark tramite la console AWS Glue o AWS Command Line Interface (AWS CLI).

Ogni 30 secondi, AWS Glue esegue il backup dei log degli eventi Spark nel percorso Amazon S3 specificato.

Argomenti

Configurazione dell'interfaccia utente di Spark (console)

Segui queste fasi per configurare l'interfaccia utente di Spark mediante la AWS Management Console. Quando si crea un processo AWS Glue, l'interfaccia utente di Spark è abilitata per impostazione predefinita.

Per attivare l'interfaccia utente di Spark durante la creazione o la modifica di un processo

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione scegliere Jobs (Processi).
3. Scegli Aggiungi processo o selezionane uno esistente.
4. In Dettagli processo, apri le Proprietà avanzate.
5. Nella scheda Interfaccia utente Spark, scegli Scrivi i log dell'interfaccia utente di Spark su Amazon S3.
6. Specifica un percorso Amazon S3 per archiviare i log di eventi Spark per il processo. Tieni presente che, se utilizzi una configurazione di sicurezza nel processo, la crittografia verrà

applicata anche al file di log dell'interfaccia utente di Spark. Per ulteriori informazioni, consulta Crittografia dei dati scritti da AWS Glue.

7. Nella sezione Configurazione della registrazione e del monitoraggio dell'interfaccia utente di Spark:

   - Seleziona Standard se stai generando i log da visualizzare nella console AWS Glue.
   - Seleziona Legacy se stai generando i log da visualizzare su un server della cronologia di Spark.
   - Puoi anche decidere di generarli entrambi.

Configurazione dell'interfaccia utente di Spark (AWS CLI)

Per generare i log da visualizzare con l'interfaccia utente di Spark nella console AWS Glue, utilizza la AWS CLI per trasferire i seguenti parametri di processo ai processi AWS Glue. Per ulteriori informazioni, consulta the section called “Parametri del processo”.

```
'--enable-spark-ui': 'true',
'--spark-event-logs-path': 's3://s3-event-log-path'
```

Per distribuire i log nelle rispettive posizioni precedenti, imposta il parametro `--enable-spark-ui-legacy-path` su `"true"`. Se non desideri generare log in entrambi i formati, rimuovi il parametro `--enable-spark-ui`.

Configurazione dell'interfaccia utente di Spark per sessioni che utilizzano notebook

> **Warning**
>
> Le sessioni interattive AWS Glue attualmente non supportano l'interfaccia utente di Spark nella console. Configura un server della cronologia di Spark.

Se usi i notebook AWS Glue, imposta la configurazione dell'interfaccia utente di Spark prima di iniziare la sessione. A tale scopo, utilizza il magic per celle `%%configure`:

```
%%configure { “--enable-spark-ui”: “true”, “--spark-event-logs-path”: “s3://path” }
```

Avvio del server della cronologia di Spark

Puoi utilizzare un server della cronologia Spark per visualizzare i log di Spark sull'infrastruttura. Puoi osservare le stesse visualizzazioni nella console AWS Glue per le esecuzioni dei processi AWS Glue su versioni AWS Glue 4.0 o successive con log generati nel formato Standard (anziché legacy). Per ulteriori informazioni, consulta the section called “Monitoraggio con l'interfaccia utente di Spark”.

Puoi avviare il server della cronologia di Spark utilizzando un modello AWS CloudFormation che ospita il server su un'istanza EC2 o avviarlo localmente utilizzando Docker.

Argomenti

Avvio del server della cronologia di Spark e visualizzazione dell'interfaccia utente di Spark tramite AWS CloudFormation

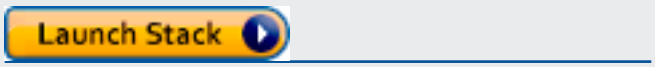
Puoi utilizzare un modello AWS CloudFormation per avviare il server della cronologia di Apache Spark e visualizzare l'interfaccia utente Web di Spark. Questi modelli sono esempi che è necessario modificare per soddisfare i requisiti.

Per avviare il server della cronologia di Spark e visualizzare l'interfaccia utente di Spark utilizzando AWS CloudFormation

1. Scegli uno dei pulsanti Launch Stack (Avvia stack) nella tabella seguente. Questo avvia lo stack nella console AWS CloudFormation.

Regione	Avvia
Stati Uniti orientali (Ohio)	Launch Stack
Stati Uniti orientali (Virginia settentrionale)	Launch Stack
Stati Uniti occidentali (California settentrionale)	Launch Stack

Regione	Avvia
Stati Uniti occidentali (Oregon)	Launch Stack
Africa (Città del Capo)	Launch Stack
Asia Pacific (Hong Kong)	Launch Stack
Asia Pacific (Mumbai)	Launch Stack
Asia Pacific (Osaka)	Launch Stack
Asia Pacific (Seul)	Launch Stack
Asia Pacifico (Singapore)	Launch Stack
Asia Pacifico (Sydney)	Launch Stack
Asia Pacifico (Tokyo)	Launch Stack
Canada (Centrale)	Launch Stack
Europa (Francoforte)	Launch Stack
Europa (Irlanda)	Launch Stack
Europa (Londra)	Launch Stack
Europa (Milano)	Launch Stack
Europa (Parigi)	Launch Stack

Regione	Avvia
Europa (Stoccolma)	
Medio Oriente (Bahrein)	
Sud America (San Paolo)	

2. Nella pagina Specify template (Specifica modello), scegli Next (Avanti).
3. Nella pagina Specify stack details (Specifica dettagli stack), immetti Stack name (Nome stack). Inserisci informazioni aggiuntive sotto Parameters (Parametri).

   a. Configurazione dell'interfaccia utente di Spark

      Inserisci le informazioni che seguono:

      - Intervallo di indirizzi IP: l'intervallo di indirizzi IP che può essere utilizzato per visualizzare l'interfaccia utente di Spark. Se desideri limitare l'accesso da un intervallo di indirizzi IP specifico, devi utilizzare un valore personalizzato.
      - Porta del server di cronologia: la porta per l'interfaccia utente di Spark. Puoi usare il valore predefinito.
      - Directory del log di eventi: scegli la posizione in cui sono archiviati i log di eventi Spark dagli endpoint di sviluppo o dal processo AWS Glue. È necessario utilizzare **`s3a://`** per lo schema di percorso dei log di eventi.
      - Posizione del pacchetto Spark: puoi usare il valore di default.
      - Percorso keystore: percorso keystore SSL/TLS per HTTPS. Se desideri utilizzare un file keystore personalizzato, puoi specificare il percorso S3 `s3://path_to_your_keystore_file` qui. Se lasci questo parametro vuoto, viene generato e utilizzato un keystore basato su certificato autofirmato.
      - Keystore password (Password keystore): inserisci una password del keystore SSL/TLS per HTTPS.

   b. Configurazione di un'istanza EC2

      Inserisci le informazioni che seguono:

- Tipo di istanza: il tipo di istanza Amazon EC2 che ospita il server della cronologia di Spark. Poiché questo modello avvia un'istanza Amazon EC2 nel tuo account, il costo di Amazon EC2 verrà addebitato separatamente nel tuo account.
- ID AMI più recente: l'ID AMI di Amazon Linux 2 per l'istanza del server della cronologia di Spark. Puoi usare il valore predefinito.
- ID VPC: l'ID del cloud privato virtuale (VPC) per l'istanza del server della cronologia di Spark. Puoi utilizzare uno qualsiasi dei VPC disponibili nell'account. L'uso di un VPC predefinito con una lista di controllo degli accessi di rete predefinita non è consigliato. Per ulteriori informazioni, consulta VPC predefinito e sottoreti predefinite e Creazione di un VPC nella Guida per l'utente di Amazon VPC.
- ID sottorete: l'ID dell'istanza del server della cronologia di Spark. Puoi utilizzare una qualsiasi delle sottoreti nel VPC. Devi essere in grado di raggiungere la rete dal client alla sottorete. Se desideri accedere tramite Internet, devi utilizzare una sottorete pubblica che dispone dell'Internet gateway nella tabella di routing.

c. Seleziona Successivo.

4. Sulla pagina Configure stack options (Impostazione di opzioni della pila), per utilizzare le credenziali utente correnti per determinare in che modo CloudFormation può creare, modificare o eliminare risorse nella pila, scegli Next (Successivo). Inoltre, nella sezione Autorizzazioni, è possibile specificare un ruolo da utilizzare al posto delle autorizzazioni utente correnti, dopodiché occorre scegliere Successivo.
5. Nella pagina Review (Revisione), rivedi il modello.

   Scegli I acknowledge that AWS CloudFormation might create IAM resources (Acconsento alla creazione di risorse IAM da parte di AWS CloudFormation), quindi seleziona Create stack (Crea stack).
6. Attendi la creazione dello stack.
7. Apri la scheda Outputs (Output).

   a. Copia l'URL di SparkUiPublicUrl se utilizzi una sottorete pubblica.

   b. Copia l'URL di SparkUiPrivateUrl se utilizzi una sottorete privata.
8. Apri un browser Web e incolla l'URL. Ciò consente di accedere al server tramite HTTPS sulla porta specificata. Il tuo browser potrebbe non riconoscere il certificato del server. Se ciò accade, ignora la protezione e procedi comunque.

Avvio del server della cronologia Spark e visualizzazione dell'interfaccia utente di Spark mediante Docker

Se preferisci l'accesso locale (non mantenere un'istanza EC2 per il server della cronologia Apache Spark), puoi anche utilizzare Docker per avviare il server della cronologia Apache Spark e visualizzare l'interfaccia utente di Spark in locale. Questo Dockerfile è un esempio che devi modificare per soddisfare i tuoi requisiti.

Prerequisiti

Per informazioni su come installare Docker sul laptop, vedi la community di Docker Engine.

Per avviare il server della cronologia Spark e visualizzare l'interfaccia utente di Spark in locale utilizzando Docker

1. Scarica i file da GitHub.

   Scarica il Dockerfile e `pom.xml` dagli esempi di codice di AWS Glue.
2. Stabilisci se desideri utilizzare le credenziali utente o le credenziali dell'utente federato per accedere a AWS.

   - Per utilizzare le credenziali utente correnti per l'accesso ad AWS, ottieni i valori per cui utilizzare `AWS_ACCESS_KEY_ID` e `AWS_SECRET_ACCESS_KEY` nel comando `docker run`. Per ulteriori informazioni, consulta Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente di IAM.
   - Per utilizzare gli utenti federati SAML 2.0 per accedere ad AWS, ottieni i valori per `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` e `AWS_SESSION_TOKEN`. Per ulteriori informazioni, consulta la sezione relativa alla richiesta di credenziali di sicurezza provvisorie
3. Determina la posizione della directory del log di eventi da utilizzare nel comando `docker run`.
4. Crea l'immagine Docker utilizzando i file nella directory locale, utilizzando il nome `glue/sparkui` e il tag `latest`.

   ```
   $ docker build -t glue/sparkui:latest .
   ```
5. Crea e avvia il container docker.

   Nei comandi seguenti, utilizza i valori ottenuti in precedenza nei passaggi 2 e 3.

   a. Per creare il container docker utilizzando le credenziali utente, utilizza un comando simile al seguente

```
docker run -itd -e SPARK_HISTORY_OPTS="$SPARK_HISTORY_OPTS -
Dspark.history.fs.logDirectory=s3a://path_to_eventlog
 -Dspark.hadoop.fs.s3a.access.key=AWS_ACCESS_KEY_ID -
Dspark.hadoop.fs.s3a.secret.key=AWS_SECRET_ACCESS_KEY"
 -p 18080:18080 glue/sparkui:latest "/opt/spark/bin/spark-class
 org.apache.spark.deploy.history.HistoryServer"
```

b. Per creare il container docker utilizzando credenziali temporanee, utilizzare `org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider` come provider e fornisci i valori delle credenziali ottenuti nel passaggio 2. Per ulteriori informazioni, consult aUtilizzo delle credenziali di sessione con TemporaryAWS CredentialsProvider nella documentazione Hadoop: integrazione con Amazon Web Services.

```
docker run -itd -e SPARK_HISTORY_OPTS="$SPARK_HISTORY_OPTS -
Dspark.history.fs.logDirectory=s3a://path_to_eventlog
 -Dspark.hadoop.fs.s3a.access.key=AWS_ACCESS_KEY_ID -
Dspark.hadoop.fs.s3a.secret.key=AWS_SECRET_ACCESS_KEY
 -Dspark.hadoop.fs.s3a.session.token=AWS_SESSION_TOKEN
 -
Dspark.hadoop.fs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.TemporaryAWSCred
 -p 18080:18080 glue/sparkui:latest "/opt/spark/bin/spark-class
 org.apache.spark.deploy.history.HistoryServer"
```

> **Note**
>
> Questi parametri di configurazione provengono dal Modulo Hadoop-AWS. Potrebbe essere necessario aggiungere una configurazione specifica in base al proprio caso d'uso. Ad esempio: gli utenti in regioni isolate dovranno configurare il `spark.hadoop.fs.s3a.endpoint`.

6. Apri `http://localhost:18080` nel browser per visualizzare l'interfaccia utente di Spark in locale.

Monitoraggio con le informazioni dell'esecuzione del processo di AWS Glue

Le informazioni dell'esecuzione del processo AWS Glue sono una funzionalità di AWS Glue che semplifica il debug e l'ottimizzazione dei lavori per i processi AWS Glue. AWS Glue fornisce

l'Interfaccia utente di Spark e i Log e parametri di CloudWatch per monitorare i processi AWS Glue. Con questa funzione, ottieni queste informazioni sull'esecuzione del processo AWS Glue:

- Numero di riga dello script del processo AWS Glue che ha riscontrato un errore.
- Azione Spark eseguita per l'ultima volta nel piano di query di Spark poco prima dell'errore del processo.
- Eventi di eccezione Spark correlati all'errore riscontrato in un flusso di log in ordine cronologico.
- Analisi della causa principale e azione consigliata (come l'ottimizzazione dello script) per risolvere il problema.
- Eventi Spark comuni (messaggi log relativi a un'azione Spark) con un'azione consigliata che affronta la causa principale.

Tutte queste informazioni sono disponibili utilizzando due nuovi flussi di log nei log di CloudWatch per il processo AWS Glue.

Requisiti

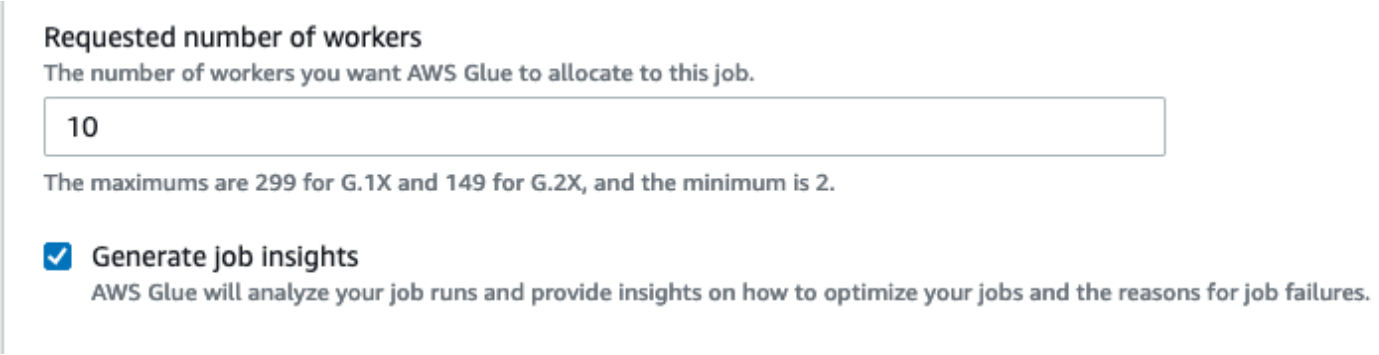
La funzione di informazioni dell'esecuzione del processo AWS Glue è disponibile per versione 2.0 AWS Glue e versione 3.0 AWS Glue. Puoi seguire la Guida alla migrazione per i processi esistenti per aggiornarli rispetto alle versioni più vecchie AWS Glue.

Abilitazione di informazioni dell'esecuzione del processo per un processo ETL AWS Glue

È possibile abilitare le informazioni dell'esecuzione del processo tramite AWS Glue Studio o la CLI.

AWS Glue Studio

Quando si crea un processo tramite AWS Glue Studio, si possono abilitare o disabilitare le informazioni dell'esecuzione del processo nella scheda Dettagli del processo. Controlla che la casella Generate job insights (Genera informazioni sul processo) sia selezionata (impostazione predefinita).



Riga di comando

Se si crea un processo tramite CLI, è possibile avviare un processo con un singolo nuovo parametro del processo: `--enable-job-insights = true`.

Per impostazione predefinita, i flussi di log di informazioni dell'esecuzione del processo vengono creati nello stesso gruppo di log predefinito utilizzato da Registrazione continua AWS Glue, cioè `/aws-glue/jobs/logs-v2/`. È possibile impostare il nome del gruppo di log personalizzato, i filtri di log e le configurazioni del gruppo di log utilizzando lo stesso set di argomenti per la registrazione continua. Per ulteriori informazioni, consulta l'articolo relativo all'Abilitazione della registrazione continua di processi AWS Glue.

Accesso ai flussi di log di informazioni dell'esecuzione del processo in CloudWatch

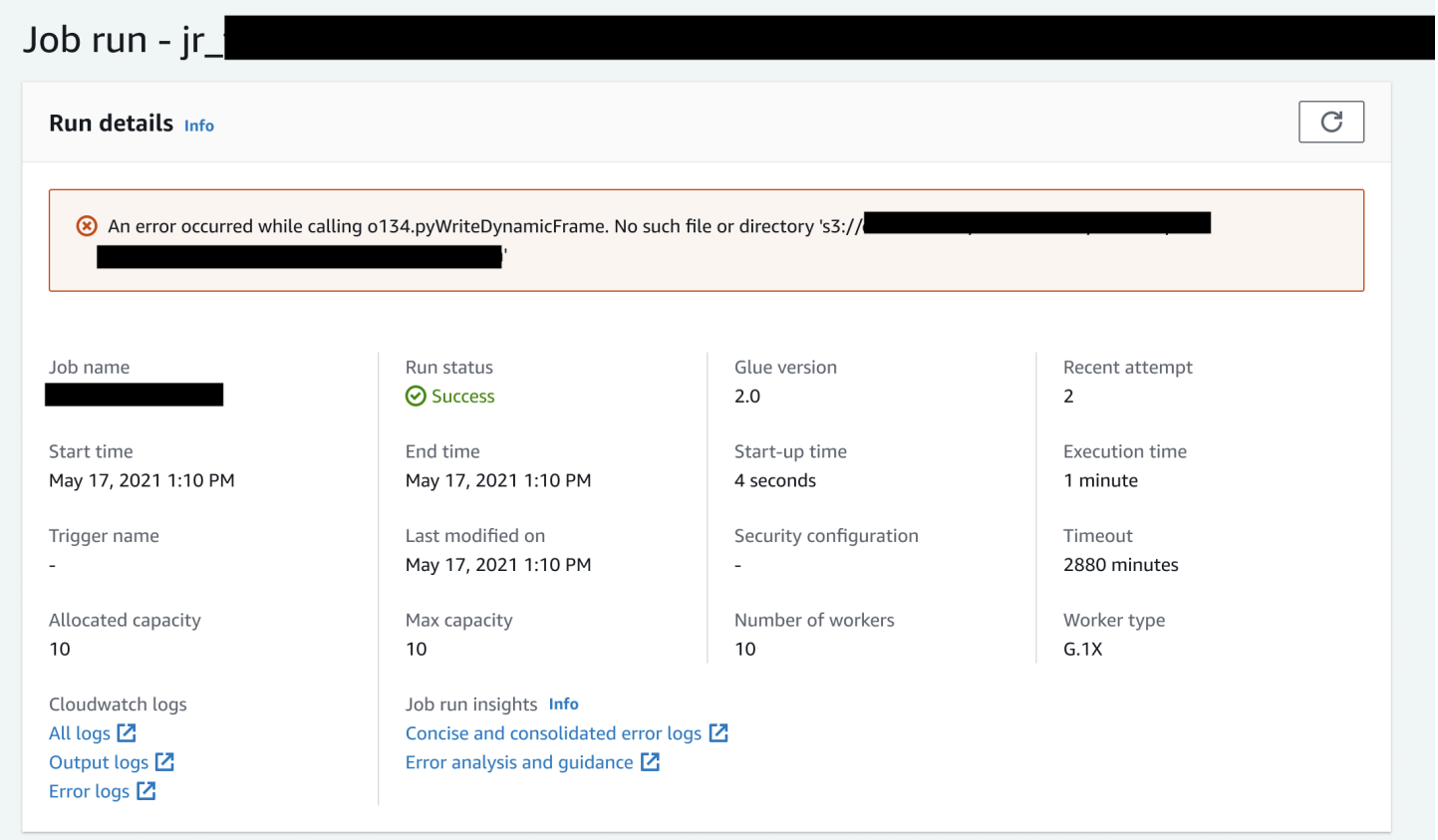
Con la funzione di informazioni dell'esecuzione del processo abilitata, potrebbero esserci due flussi di log creati quando un processo non riesce. Quando un processo termina correttamente, nessuno dei flussi viene generato.

1. Flusso di log analisi delle eccezioni: `<job-run-id>-job-insights-rca-driver`. Questo flusso fornisce quanto segue:
   - Numero di riga dello script del processo AWS Glue che ha causato un errore.
   - Azione Spark eseguita per ultima nel piano di query Spark (DAG).
   - Eventi brevi in ordine cronologico dal driver e dagli esecutori Spark correlati all'eccezione. Se necessario, è possibile trovare dettagli come i messaggi di errore completi, l'attività Spark non riuscita e il relativo ID esecutori che consentono di concentrarsi sul flusso di log dell'esecutore specifico per un'indagine più approfondita.
2. Flusso di informazioni basato su regole:
   - Analisi della causa principale e consigli su come correggere gli errori (ad esempio l'utilizzo di un parametro di lavoro specifico per ottimizzare le prestazioni).
   - Eventi Spark rilevanti come base per l'analisi della causa principale e un'azione consigliata.

> **Note**
>
> Il primo flusso esiste solo se sono disponibili eventi di eccezione Spark per un'esecuzione del processo non riuscita e il secondo stream esisterà solo se sono disponibili informazioni per l'esecuzione del processo non riuscita. Ad esempio, se il processo termina correttamente,

nessuno dei flussi verrà generato. Se il processo fallisce ma non esiste una regola definita dal servizio che può corrispondere allo scenario di errore, verrà generato solo il primo flusso.

Se il processo viene creato da AWS Glue Studio, i collegamenti ai flussi di cui sopra sono disponibili anche nella scheda dei dettagli esecuzione del processo (Job run insights, informazioni dell'esecuzione del processo) come "Log degli errori concisi e consolidati" e "Analisi e guida degli errori".



Esempio per informazioni dell'esecuzione del processo AWS Glue

In questa sezione, presentiamo un esempio di come la funzione Informazioni dell'esecuzione del processo può essere d'aiuto nella risoluzione di un problema nel processo non riuscito. In questo esempio, un utente ha dimenticato di importare il modulo richiesto (tensorflow) in un processo AWS Glue per analizzare e costruire un modello di machine learning sui propri dati.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
```

```
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.types import *
from pyspark.sql.functions import udf,col

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

data_set_1 = [1, 2, 3, 4]
data_set_2 = [5, 6, 7, 8]

scoresDf = spark.createDataFrame(data_set_1, IntegerType())

def data_multiplier_func(factor, data_vector):
    import tensorflow as tf
    with tf.compat.v1.Session() as sess:
        x1 = tf.constant(factor)
        x2 = tf.constant(data_vector)
        result = tf.multiply(x1, x2)
        return sess.run(result).tolist()

data_multiplier_udf = udf(lambda x:data_multiplier_func(x, data_set_2),
 ArrayType(IntegerType(),False))
factoredDf = scoresDf.withColumn("final_value", data_multiplier_udf(col("value")))
print(factoredDf.collect())
```

Senza la funzione di Informazioni dell'esecuzione del processo, data la non riuscita del processo, viene visualizzato solo questo messaggio generato da Spark:

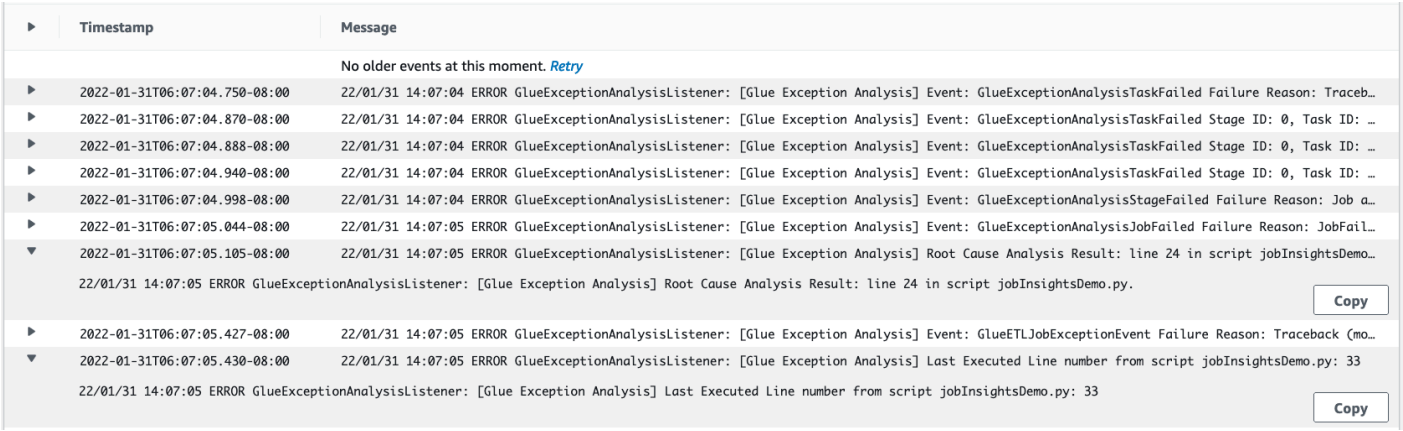
`An error occurred while calling o111.collectToPython. Traceback (most recent call last):`

Il messaggio è ambiguo e limita l'esperienza di debug. In questo caso, questa funzione fornisce informazioni aggiuntive in due flussi di log di CloudWatch:

1. Il flusso di log `job-insights-rca-driver`:
   - Eventi eccezioni: questo flusso di log fornisce gli eventi di eccezione Spark relativi all'errore raccolto dal driver Spark e dai diversi lavoratori distribuiti. Questi eventi aiutano nella

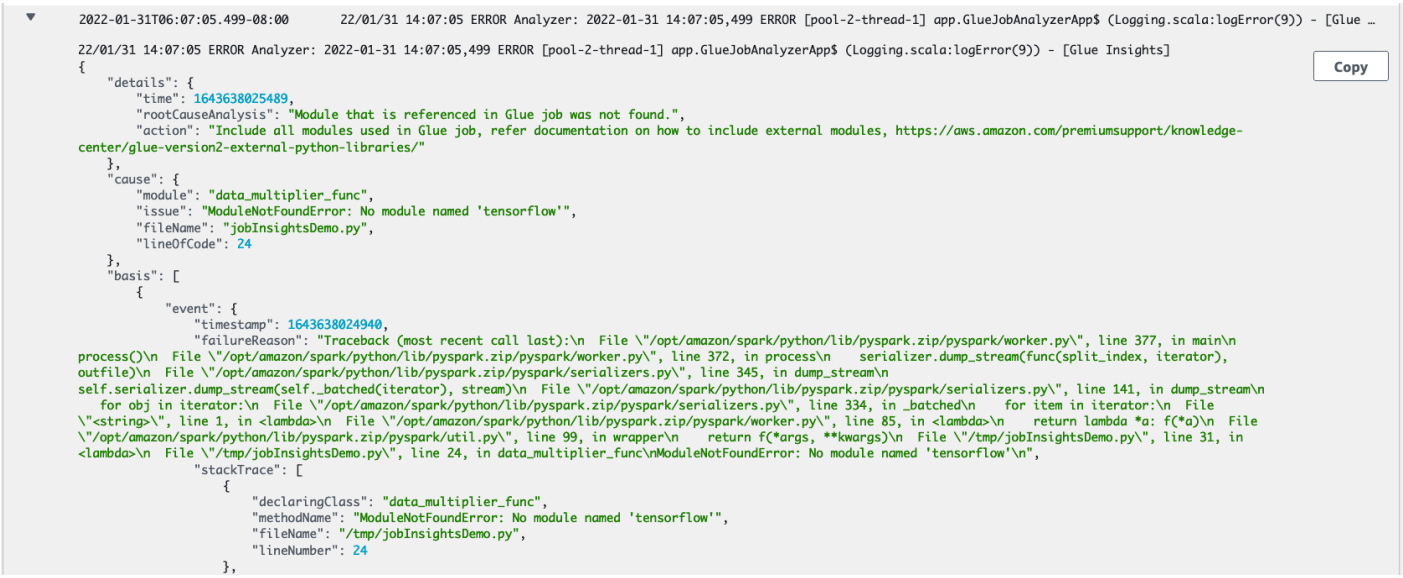
comprensione della propagazione in ordine cronologico dell'eccezione mentre il codice difettoso viene eseguito su attività, esecutori e fasi di Spark distribuiti tra tutti i dipendenti AWS Glue.

- Numeri riga: questo flusso di log identifica la riga 21, che ha eseguito la chiamata per importare il modulo Python mancante che ha causato l'errore; identifica anche la riga 24, la chiamata all'azione Spark `collect()`, come ultima riga eseguita nello script.



2. Il flusso di log `job-insights-rule-driver`:

- Causa principale e raccomandazione: oltre al numero di riga e all'ultimo numero di riga eseguito per l'errore nello script, questo flusso di log mostra l'analisi della causa principale e la raccomandazione per seguire il doc AWS Glue e impostare i parametri di processo necessari per utilizzare un modulo Python aggiuntivo nel processo AWS Glue.
- Evento base: questo flusso di log mostra anche l'evento di eccezione Spark che è stato valutato con la regola definita dal servizio per dedurre la causa principale e fornire una raccomandazione.



Monitoraggio con Amazon CloudWatch

È possibile monitorare AWS Glue usando Amazon CloudWatch, che raccoglie i dati non elaborati da AWS Glue e li elabora trasformandoli in parametri leggibili quasi in tempo reale. Queste statistiche vengono registrate per un periodo di due settimane, per permettere l'accesso alle informazioni storiche e offrire una prospettiva migliore sulle prestazioni del servizio o dell'applicazione Web. Per impostazione predefinita, i dati dei parametri AWS Glue vengono inviati automaticamente a CloudWatch. Per ulteriori informazioni, consulta Che cos'è Amazon CloudWatch? nella Guida per l'utente di Amazon CloudWatch e Parametri di AWS Glue.

Registrazione continua

AWS Glue supporta anche la registrazione continua in tempo reale per i processi AWS Glue. Quando è abilitata la registrazione continua per un processo, puoi visualizzare i log in tempo reale nella console AWS Glue o nel pannello di controllo della console CloudWatch. Per ulteriori informazioni, consulta Registrazione continua dei processi AWS Glue.

Parametri di osservabilità

Quando i parametri di osservabilità del processo sono abilitati, vengono generati parametri Amazon CloudWatch aggiuntivi quando il processo viene eseguito. Utilizza i parametri AWS Glue di osservabilità per generare approfondimenti su ciò che accade all'interno di AWS Glue e migliorare la classificazione e l'analisi dei problemi.

Argomenti

- Monitoraggio di AWS Glue con i parametri di Amazon CloudWatch
- Configurazione degli allarmi Amazon CloudWatch e dei profili dei processi AWS Glue
- Registrazione continua dei processi AWS Glue
- Monitoraggio con parametri AWS Glue di osservabilità

Monitoraggio di AWS Glue con i parametri di Amazon CloudWatch

Puoi creare il profilo delle attività AWS Glue e monitorarle utilizzando il profiler dei processi AWS Glue. Raccoglie i dati non elaborati dai processi AWS Glue e li elabora in parametri leggibili quasi in tempo reale archiviati in Amazon CloudWatch. Queste statistiche vengono conservate e aggregate in CloudWatch, per permettere l'accesso alle informazioni di cronologia per ottenere una panoramica migliore delle prestazioni dell'applicazione.

> **Note**
>
> Potrebbero essere applicati costi aggiuntivi quando si abilitano i parametri del processo e vengono creati i parametri personalizzati di CloudWatch. Per ulteriori informazioni, consulta Prezzi di Amazon CloudWatch .

Panoramica dei parametri AWS Glue

Quando interagisci con AWS Glue, i parametri vengono inviati a CloudWatch. Puoi visualizzare questi parametri nella console AWS Glue (metodo preferenziale), nel pannello di controllo della console CloudWatch o in AWS Command Line Interface (AWS CLI).

Per visualizzare i parametri usando il pannello di controllo della console AWS Glue

Puoi visualizzare grafici dettagliati o di riepilogo dei parametri per un processo oppure grafici dettagliati per un'esecuzione di un processo.

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione, scegli Monitoraggio dell'esecuzione del processo.
3. In Esecuzioni del processo, scegli Operazioni per interrompere un processo attualmente in esecuzione, visualizzare un processo o riavvolgerne il segnalibro.
4. Seleziona un processo, quindi scegli Visualizza dettagli di esecuzione per visualizzare informazioni aggiuntive sull'esecuzione del processo.

Per visualizzare i parametri utilizzando il pannello di controllo della console CloudWatch

I parametri vengono raggruppati prima in base allo spazio dei nomi del servizio e successivamente in base alle diverse combinazioni di dimensioni all'interno di ogni spazio dei nomi.

1. Aprire la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Nel riquadro di navigazione, selezionare Parametri.
3. Selezionare lo spazio dei nomi Glue.

Per visualizzare i parametri usando AWS CLI

- Al prompt dei comandi utilizza il comando seguente.

```
aws cloudwatch list-metrics --namespace Glue
```

AWS Glue invia i parametri a CloudWatch ogni 30 secondi e i pannelli di controllo dei parametri di CloudWatch sono configurati per visualizzare i parametri ogni minuto. I parametri AWS Glue rappresentano i valori delta rispetto ai valori segnalati in precedenza. Se appropriato, i pannelli di controllo dei parametri aggregano (sommano) i valori inviati ogni 30 secondi per ottenere un valore per l'intero ultimo minuto.

Comportamento dei parametri di AWS Glue per i processi Spark

I parametri di AWS Glue vengono abilitati al momento dell'inizializzazione di un `GlueContext` in uno script e vengono in genere aggiornati solo al termine di un'attività di Apache Spark. Rappresentano i valori aggregati per tutte le attività di Spark completate fino al momento attuale.

Tuttavia, i parametri Spark passati da AWS Glue a CloudWatch sono generalmente valori assoluti che rappresentano lo stato corrente nel momento in cui vengono segnalati. AWS Glue invia i parametri a CloudWatch ogni 30 secondi e i pannelli di controllo dei parametri mostrano in genere la media tra i punti dati ricevuti nell'ultimo minuto.

I nomi dei parametri AWS Glue sono tutti preceduti da uno dei seguenti tipi di prefisso:

- `glue.driver.`: i parametri i cui nomi iniziano con questo prefisso rappresentano parametri AWS Glue aggregati da tutti gli executor nel driver Spark oppure parametri Spark corrispondenti al driver Spark.
- `glue.`executorId`.`: executorId è il numero di un executor Spark specifico. Corrisponde agli executor elencati nei log.
- `glue.ALL.`: i parametri i cui nomi iniziano con questo prefisso aggregano i valori di tutti gli executor Spark.

Parametri di AWS Glue

AWS Glue fornisce i profili e invia i seguenti parametri a CloudWatch ogni 30 secondi e il pannello di controllo dei parametri di AWS Glue li segnala una volta al minuto:

Parametro	Descrizione
`glue.driver.aggregate.bytesRead`	Il numero di byte letti da tutte le origini dati da tutti i processi Spark completati in esecuzione in tutti gli executor. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: byte Può essere utilizzato per monitorare: • I byte letti. • L'avanzamento del processo. • L'origine dati JDBC. • Problemi relativi ai segnalibri di processo. • Varianza tra esecuzioni di processo. Questo parametro può essere utilizzato come il parametro `glue.ALL.s3.filesystem.read_bytes` , con la differenza che questo viene aggiornato alla fine di un processo Spark e acquisisce anche origini dati non S3.
`glue.driver.aggregate.elapsedTime`	Il tempo di ETL trascorso in millisecondi (non include i tempi di bootstrap del processo).

Parametro	Descrizione
	Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: millisecondi Può essere utilizzato per determinare il tempo medio di esecuzione di un processo. Alcuni modi per utilizzare i dati: • Impostare allarmi per i ritardi. • Misurare la varianza tra esecuzioni di processo.

Parametro	Descrizione
`glue.driver.aggregate.numCompletedStages`	Il numero di fasi completate nel processo. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: numero Può essere utilizzato per monitorare: • L'avanzamento del processo. • La sequenza temporale per fase dell'esecuzione del processo, se correlata ad altri parametri. Alcuni modi per utilizzare i dati: • Identificare fasi impegnative nell'esecuzione di un processo. • Impostare gli allarmi per picchi correlati (fasi impegnative) tra le esecuzioni dei lavori.

Parametro	Descrizione
`glue.driver.aggregate.numCompletedTasks`	Il numero di attività completate nel processo. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: numero Può essere utilizzato per monitorare: • L'avanzamento del processo. • Parallelismo all'interno di una fase.

Parametro	Descrizione
`glue.driver.aggregate.numFailedTasks`	Il numero di processi non riusciti. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: numero Può essere utilizzato per monitorare: • Anomalie dei dati che causano la non riuscita delle attività del processo. • Anomalie del cluster che causano la non riuscita delle attività del processo. • Anomalie dello script che causano la non riuscita delle attività del processo. I dati possono essere utilizzati per impostare allarmi per errori maggiori che potrebbero suggerire anomalie nei dati, nel cluster o negli script.

Parametro	Descrizione
`glue.driver.aggregate.numKilledTasks`	Il numero di attività interrotte. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: numero Può essere utilizzato per monitorare: • Anomalie nella differenza dei dati che provocano eccezioni (OOM) che interrompono le attività. • Anomalie dello script che provocano eccezioni (OOM) che interrompono le attività. Alcuni modi per utilizzare i dati: • Impostare allarmi per errori maggiori che potrebber o suggerire anomalie nei dati. • Impostare allarmi per errori maggiori che potrebber o suggerire anomalie nel cluster. • Impostare allarmi per errori maggiori che potrebber o suggerire anomalie nello script.

Parametro	Descrizione
`glue.driver.aggregate.recordsRead`	Il numero di record letti da tutte le origini dati da tutti i processi Spark completati in esecuzione in tutti gli executor. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: numero Può essere utilizzato per monitorare: • Record letti. • L'avanzamento del processo. • L'origine dati JDBC. • Problemi relativi ai segnalibri di processo. • Differenza nelle esecuzioni dei processi nei giorni. Questo parametro può essere utilizzato come il parametro `glue.ALL.s3.filesystem.read_bytes` , con la differenza che questo viene aggiornato alla fine di un processo Spark.

Parametro	Descrizione
`glue.driver.aggregate.shuffleBytesWritten`	Numero di byte scritti da tutti gli executor per distribuire i dati in modo casuale dal report precedente (aggregati in base al pannello di controllo dei parametri di AWS Glue come il numero di byte scritti a questo scopo nel minuto precedente). Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: byte Può essere utilizzato per monitorare: la distribuzione casuale dei dati nei processi (join di grandi dimensioni, groupBy, repartition, coalesce). Alcuni modi per utilizzare i dati: • Ripartizionare o decomprimere file di input di grandi dimensioni prima di ulteriori elaborazioni. • Ripartizionare i dati in modo più uniforme per evitare i tasti di scelta rapida. • Prefiltrare i dati prima delle operazioni di join o groupBy.

Parametro	Descrizione
`glue.driver.aggregate.shuffleLocalBytesRead`	Numero di byte letti da tutti gli executor per distribuire i dati in modo casuale dal report precedente (aggregati in base al pannello di controllo dei parametri di AWS Glue come il numero di byte letti a questo scopo nel minuto precedente). Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue, viene utilizzata una statistica SUM (Somma) per l'aggregazione. Unità: byte Può essere utilizzato per monitorare: la distribuzione casuale dei dati nei processi (join di grandi dimensioni, groupBy, repartition, coalesce). Alcuni modi per utilizzare i dati: • Ripartizionare o decomprimere file di input di grandi dimensioni prima di ulteriori elaborazioni. • Ripartizionare i dati in modo più uniforme utilizzando i tasti di scelta rapida. • Prefiltrare i dati prima delle operazioni di join o groupBy.

Parametro	Descrizione
`glue.driver.BlockManager.disk.diskSpaceUsed_MB`	Numero di megabyte di spazio su disco utilizzati in tutti gli executor. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(valutazione). Statistiche valide: Average (Media). Si tratta di un parametro Spark, riportato come valore assoluto. Unità: megabyte Può essere utilizzato per monitorare: • Spazio su disco utilizzato per i blocchi che rappresentano partizioni RDD memorizzate nella cache. • Spazio su disco utilizzato per i blocchi che rappresentano uscite shuffle intermedie. • Spazio su disco utilizzato per i blocchi che rappresentano le trasmissioni. Alcuni modi per utilizzare i dati: • Identificare gli errori dei processi dovuti a un maggiore utilizzo del disco. • Identificare partizioni di grandi dimensioni con conseguente riversamento o distribuzione casuale. • Aumentare la capacità DPU sottoposta a provisioning per risolvere questi problemi.

Parametro	Descrizione
`glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`	Numero di executor di processo attivi. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(valutazione). Statistiche valide: Average (Media). Si tratta di un parametro Spark, riportato come valore assoluto. Unità: numero Può essere utilizzato per monitorare: • L'attività dei processi. • Calo degli executor (con solo pochi executor attivi) • Parallelismo attuale a livello di executor. Alcuni modi per utilizzare i dati: • Ripartizionare o decomprimere i file di input di grandi dimensioni in anticipo se il cluster è sottoutilizzato. • Identificare i ritardi di esecuzione delle fasi o dei processi dovuti a scenari di rallentamento. • • Confrontare con numberMaxNeededExecutors per comprendere il backlog per il provisioning di più DPU.

Parametro	Descrizione
`glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors`	Il numero massimo di executor di processo (attivi e in sospeso) necessari per soddisfare il carico corrente. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(valutazione). Statistiche valide: Maximum (Massimo). Si tratta di un parametro Spark, riportato come valore assoluto. Unità: numero Può essere utilizzato per monitorare: • L'attività dei processi. • Parallelismo attuale a livello di executor e backlog delle attività in sospeso non ancora pianificate per la non disponibilità degli executor, a causa della capacità DPU o di executor interrotti/non riusciti. Alcuni modi per utilizzare i dati: • Identificare sospensione/backlog della coda di pianificazione. • Identificare i ritardi di esecuzione delle fasi o dei processi dovuti a scenari di rallentamento. • Confrontare con numberAllExecutors per comprendere il backlog per il provisioning di più DPU. • Aumentare la capacità DPU sottoposta a provision ing per correggere il backlog dell'executor in sospeso.

Parametro	Descrizione

`glue.driver.jvm.heap.usage`
 	
`glue.executorId.jvm.heap.usage`
 	
`glue.ALL.jvm.heap.usage`
	Frazione di memoria usata dall'heap JVM per questo driver (dimensione: 0-1) per driver, executor identific ato da executorId o TUTTI gli executor.
	
	Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(valutazione).
	
	Statistiche valide: Average (Media). Si tratta di un parametro Spark, riportato come valore assoluto.
	
	Unità: percentuale
	
	Può essere utilizzato per monitorare:
	
	• Condizioni di memoria insufficiente (OOM) del driver utilizzando `glue.driver.jvm.he ap.usage` .
	• Condizioni di memoria insufficiente (OOM) dell'executor utilizzando `glue.ALL.jvm.heap. usage` .
	
	Alcuni modi per utilizzare i dati:
	
	• Identificare gli ID e le fasi dell'executor che consumano memoria.
	• Identificare gli ID e le fasi dell'executor in calo.
	• Identificare una condizione di memoria esaurita (OOM) del driver.
	• Identificare una condizione di memoria insufficiente (OOM) dell'executor e ottenere l'ID dell'executor corrispondente, in modo da avere una traccia di stack dal log dell'executor.

Parametro	Descrizione
	• Identificare i file o le partizioni con potenziali differenze dei dati che causano ritardi o condizioni di memoria insufficiente (OOM).

Parametro	Descrizione

`glue.driver.jvm.heap.used`
 	
`glue.executorId.jvm.heap.used`
 	
`glue.ALL.jvm.heap.used`
	Il numero di byte di memoria utilizzati dall'heap JVM per il driver, l'executor identificato da executorId o TUTTI gli executor.
	
	Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(valutazione).
	
	Statistiche valide: Average (Media). Si tratta di un parametro Spark, riportato come valore assoluto.
	
	Unità: byte
	
	Può essere utilizzato per monitorare:
	
	• Condizioni di memoria insufficiente (OOM) del driver.
	• Condizioni di memoria insufficiente (OOM) dell'executor.
	
	Alcuni modi per utilizzare i dati:
	
	• Identificare gli ID e le fasi dell'executor che consumano memoria.
	• Identificare gli ID e le fasi dell'executor in calo.
	• Identificare una condizione di memoria esaurita (OOM) del driver.
	• Identificare una condizione di memoria insufficiente (OOM) dell'executor e ottenere l'ID dell'executor corrispondente, in modo da avere una traccia di stack dal log dell'executor.
	• Identificare i file o le partizioni con potenziali differenze dei dati che causano ritardi o condizioni di memoria insufficiente (OOM).

Parametro	Descrizione

`glue.driver.s3.filesystem.read_bytes`
 	
`glue.executorId.s3.filesystem.read_bytes`
 	
`glue.ALL.s3.filesystem.read_bytes`
	Numero di byte letti da Amazon S3 dal driver, un executor identificato da executorId o TUTTI gli executor dal report precedente (aggregati in base al pannello di controllo dei parametri di AWS Glue come il numero di byte letti nel minuto precedente).
	
	Dimensioni valide: `JobName`, `JobRunId` e `Type` (valutazione).
	
	Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue viene utilizzata una statistica SUM (Somma) per l'aggregazione. L'area sotto la curva nel pannello di controllo dei parametri di AWS Glue può essere utilizzata per confrontare visivamente i byte letti da due diverse esecuzioni di processi.
	
	Unità: byte.
	
	Può essere utilizzato per monitorare:
	
	• Spostamento di dati ETL.
	• L'avanzamento del processo.
	• Problemi relativi ai segnalibri di processo (dati elaborati, rielaborati e saltati).
	• Confronto tra letture e velocità di importazione da origini dati esterne.
	• Varianza tra esecuzioni di processo.
	
	I dati risultanti possono essere utilizzati per:
	
	• Pianificazione della capacità DPU.

Parametro	Descrizione
	• Impostare gli allarmi per picchi o riduzioni di grandi dimensioni nei dati letti per le esecuzioni le fasi dei processi.

Parametro	Descrizione

`glue.driver.s3.filesystem.write_bytes`
 	
`glue.executorId.s3.filesystem.write_bytes`
 	
`glue.ALL.s3.filesystem.write_bytes`
	Numero di byte scritti da Amazon S3 dal driver, un executor identificato da executorId o TUTTI gli executor dal report precedente (aggregati in base al pannello di controllo dei parametri di AWS Glue come il numero di byte scritti nel minuto precedente).
	
	Dimensioni valide: `JobName`, `JobRunId` e `Type` (valutazione).
	
	Statistiche valide: SUM (Somma). Questo parametro è un valore delta rispetto all'ultimo valore riportato, quindi nel pannello di controllo dei parametri di AWS Glue viene utilizzata una statistica SUM (Somma) per l'aggregazione. L'area sotto la curva nel pannello di controllo dei parametri di AWS Glue può essere utilizzata per confrontare visivamente i byte scritti da due diverse esecuzioni di processi.
	
	Unità: byte
	
	Può essere utilizzato per monitorare:
	
	• Spostamento di dati ETL.
	• L'avanzamento del processo.
	• Problemi relativi ai segnalibri di processo (dati elaborati, rielaborati e saltati).
	• Confronto tra letture e velocità di importazione da origini dati esterne.
	• Varianza tra esecuzioni di processo.
	
	Alcuni modi per utilizzare i dati:
	
	• Pianificazione della capacità DPU.

Parametro	Descrizione
	• Impostare gli allarmi per picchi o riduzioni di grandi dimensioni nei dati letti per le esecuzioni le fasi dei processi.
`glue.driver.streaming.numRecords`	Numero di record ricevuti in un micro-batch. Questo parametro è disponibile solo per processi di streaming AWS Glue con AWS Glue versione 2.0 e successive. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Valid Statistics: Sum (Somma), Maximum (Massimo) , Minimum (Minimo), Average (Media), Percentile (Percentuale) Unità: numero Può essere utilizzato per monitorare: • Record letti. • L'avanzamento del processo.

Parametro	Descrizione
`glue.driver.streaming.batchProcessingTimeInMs`	Il tempo necessario per elaborare i batch in millisecondi. Questo parametro è disponibile solo per processi di streaming AWS Glue con AWS Glue versione 2.0 e successive. Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(conteggio). Valid Statistics: Sum (Somma), Maximum (Massimo), Minimum (Minimo), Average (Media), Percentile (Percentuale) Unità: numero Può essere utilizzato per monitorare: • L'avanzamento del processo. • Prestazioni dello script.

Parametro	Descrizione

`glue.driver.system.cpuSystemLoad`
 	
`glue.executorId.system.cpuSystemLoad`
 	
`glue.ALL.system.cpuSystemLoad`
	Frazione del carico di sistema della CPU usata (dimensione: 0-1) dal driver, da un executor identificato da executorId o da tutti gli executor.
	
	Dimensioni valide: `JobName` (il nome del processo AWS Glue), `JobRunId` (l'ID JobRun o `ALL`) e `Type`(valutazione).
	
	Statistiche valide: Average (Media). Questo parametro è riportato come valore assoluto.
	
	Unità: percentuale
	
	Può essere utilizzato per monitorare:
	
	• Carico della CPU del driver.
	• Carico della CPU dell'executor.
	• Rilevamento di executor o fasi associati alla CPU o all'IO in un processo.
	
	Alcuni modi per utilizzare i dati:
	
	• Pianificazione della capacità DPU insieme alle metriche IO (Byte letti/casuali, parallelismo dell'attività) e al numero massimo di parametri di executor necessari.
	• Identificare il rapporto CPU/IO. Questo consente il ripartizionamento e l'aumento della capacità di provisioning per i processi a esecuzione prolungata con set di dati suddivisibili con minore utilizzo della CPU.

Dimensioni dei parametri di AWS Glue

I parametri di AWS Glue utilizzano lo spazio dei nomi AWS Glue e forniscono i parametri per le seguenti dimensioni:

Dimensione	Descrizione
`JobName`	Questa dimensione filtra i parametri di tutte le esecuzioni di processo di uno specifico processo AWS Glue.
`JobRunId`	Questa dimensione filtra i parametri di uno specifico processo AWS Glue eseguito da un ID JobRun o `ALL`.
`Type`	Questa dimensione filtra i parametri in base a `count` (numero aggregato) o `gauge` (valore in un determina to momento).

Per ulteriori informazioni, consulta la Guida per l'utente di Amazon CloudWatch.

Configurazione degli allarmi Amazon CloudWatch e dei profili dei processi AWS Glue

I parametri di AWS Glue sono disponibili anche in Amazon CloudWatch. È possibile configurare allarmi per qualsiasi parametro AWS Glue per i processi pianificati.

Di seguito sono illustrati alcuni scenari comuni per l'impostazione degli allarmi:

- Processi che stanno per esaurire la memoria: è possibile impostare un allarme quando l'utilizzo della memoria supera la media normale per il driver o un executor per un processo AWS Glue.
- Calo degli executor: è possibile impostare un allarme quando il numero di executor scende sotto una determinata soglia per un intervallo di tempo lungo in un processo AWS Glue.
- Backlog di dati o nuova elaborazione: è possibile confrontare i parametri dei singoli processi in un flusso di lavoro usando un'espressione matematica di CloudWatch. È quindi possibile attivare un allarme in base al valore dell'espressione risultante (ad esempio il rapporto tra byte scritti da un processo e byte letti dal processo seguente).

Per istruzioni dettagliate su come configurare gli allarmi, consulta Creazione o modifica di un allarme CloudWatch nella Guida per l'utente di Amazon CloudWatch Events.

Per scenari di monitoraggio e debug tramite CloudWatch, consulta Monitoraggio e debug dei processi.

Registrazione continua dei processi AWS Glue

AWS Glue fornisce la registrazione continua in tempo reale dei processi AWS Glue. Puoi visualizzare i log dei job di Apache Spark in tempo reale su Amazon CloudWatch, inclusi i log dei driver, i log degli esecutori e una barra di avanzamento dei job di Apache Spark. La visualizzazione di log in tempo reale fornisce una migliore prospettiva sul processo in esecuzione.

Quando avvii un AWS Glue job, invia le informazioni di registrazione in tempo reale a CloudWatch (ogni 5 secondi e prima di ogni chiusura dell'executor) dopo l'avvio dell'applicazione Spark. Puoi visualizzare i log sulla console o sulla dashboard della AWS Glue console. CloudWatch

La registrazione continua include le seguenti caratteristiche:

- Registrazione continua
- Un logger di script personalizzato per registrare messaggi specifici di applicazioni
- La barra di avanzamento della console per monitorare lo stato di esecuzione del processo AWS Glue corrente

Per informazioni su come è supportata la registrazione continua in AWS Glue versione 2.0, consulta Esecuzione di processi ETL Spark con tempi di avvio ridotti.

Puoi limitare l'accesso ai gruppi o ai flussi di CloudWatch log per consentire ai ruoli IAM di leggere i log. Per maggiori dettagli sulla limitazione dell'accesso, consulta Using Identity-based policy (IAM policies) for Logs nella documentazione. CloudWatch CloudWatch

> **Note**
>
> È possibile che vengano addebitati costi aggiuntivi quando si abilita la registrazione continua e vengono creati eventi di registro aggiuntivi. CloudWatch Per ulteriori informazioni, consulta i CloudWatch prezzi di Amazon.

Argomenti

- Abilitazione della registrazione continua di processi AWS Glue
- Visualizzazione della registrazione continua dei processi AWS Glue

Abilitazione della registrazione continua di processi AWS Glue

È possibile abilitare la registrazione continua utilizzando la AWS Glue console o tramite AWS Command Line Interface (AWS CLI).

È possibile abilitare la registrazione continua quando si crea un nuovo lavoro, si modifica un lavoro esistente o si abilita tramite. AWS CLI

È inoltre possibile specificare opzioni di configurazione personalizzate come il nome del gruppo di Amazon CloudWatch log, il prefisso del flusso di CloudWatch registro prima dell'ID di esecuzione del AWS Glue processo, l'ID driver/executor e il modello di conversione dei log per i messaggi di log. Queste configurazioni consentono di impostare log aggregati in gruppi di CloudWatch log personalizzati con diverse politiche di scadenza e di analizzarli ulteriormente con prefissi e modelli di conversione personalizzati per i flussi di log.

Argomenti

- Utilizzando il AWS Management Console
- Registrazione di messaggi specifici di applicazioni tramite logger di script personalizzato
- Abilitazione della barra di avanzamento per visualizzare l'avanzamento del processo
- Configurazione di sicurezza con la registrazione continua.

Utilizzando il AWS Management Console

Segui questi passaggi per utilizzare la console per abilitare la registrazione continua durante la creazione o la modifica di un processo AWS Glue.

Per creare un nuovo processo AWS Glue con la registrazione continua

1. Accedere AWS Management Console e aprire la AWS Glue console all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione, scegli ETL jobs.
3. Scegli Visual ETL.
4. Nella scheda Dettagli del lavoro, espandi la sezione Proprietà avanzate.

5. In Registrazione continua seleziona Abilita accessi. CloudWatch

Per abilitare la registrazione continua di un processo AWS Glue esistente

1. Apri la AWS Glue console all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione scegliere Jobs (Processi).
3. Scegliere un processo esistente dall'elenco Jobs (Processi).
4. Scegliere Action (Operazione), Edit job (Modifica processo).
5. Nella scheda Dettagli del lavoro, espandi la sezione Proprietà avanzate.
6. In Registrazione continua seleziona Abilita accessi. CloudWatch

Utilizzando il AWS CLI

Per abilitare la registrazione continua, trasferisci i parametri del processo a un processo AWS Glue. Passa i seguenti parametri di lavoro speciali in modo simile agli altri parametri di AWS Glue lavoro. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.

```
'--enable-continuous-cloudwatch-log': 'true'
```

Puoi specificare un nome di gruppo di CloudWatch log Amazon personalizzato. Se non specificato, il nome predefinito del gruppo di log è `/aws-glue/jobs/logs-v2/`.

```
'--continuous-log-logGroup': 'custom_log_group_name'
```

Puoi specificare un prefisso Amazon CloudWatch Log Stream personalizzato. Se non specificato, il prefisso del flusso di log predefinito è l'ID di esecuzione del processo.

```
'--continuous-log-logStreamPrefix': 'custom_log_stream_prefix'
```

È possibile specificare un modello di conversione di registrazione continua personalizzato. Se non specificato, il modello di conversione predefinito è `%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n`. Tieni presente che il modello di conversione si applica solo ai log dei driver e ai log delle esecuzioni. Non interessa la barra di avanzamento di AWS Glue.

```
'--continuous-log-conversionPattern': 'custom_log_conversion_pattern'
```

Registrazione di messaggi specifici di applicazioni tramite logger di script personalizzato

Puoi utilizzare il logger AWS Glue per registrare nello script qualsiasi messaggio specifico di applicazioni inviato in tempo reale al flusso di log di driver.

Il seguente esempio mostra uno script Python.

```
from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)
logger = glueContext.get_logger()
logger.info("info message")
logger.warn("warn message")
logger.error("error message")
```

Il seguente esempio mostra uno script Scala.

```
import com.amazonaws.services.glue.log.GlueLogger

object GlueApp {
  def main(sysArgs: Array[String]) {
    val logger = new GlueLogger
    logger.info("info message")
    logger.warn("warn message")
    logger.error("error message")
  }
}
```

Abilitazione della barra di avanzamento per visualizzare l'avanzamento del processo

AWS Glue fornire una barra di avanzamento del processo in tempo reale sotto il flusso di log `JOB_RUN_ID-progress-bar` per controllare lo stato dell'esecuzione del processo AWS Glue. Al momento, supporta solo i processi che inizializzano `glueContext`. Se esegui un semplice processo Spark senza inizializzare `glueContext`, la barra di avanzamento AWS Glue non viene visualizzata.

La barra di avanzamento mostra il seguente aggiornamento dell'avanzamento ogni 5 secondi.

```
Stage Number (Stage Name): > (numCompletedTasks + numActiveTasks) /
 totalNumOfTasksInThisStage]
```

Configurazione di sicurezza con la registrazione continua.

Se è abilitata una configurazione di sicurezza per CloudWatch i log, AWS Glue creerà un gruppo di log denominato come segue per i log continui:

```
<Log-Group-Name>-<Security-Configuration-Name>
```

I gruppi di log predefiniti e personalizzati saranno i seguenti:

- Il gruppo di log continuo di default sarà `/aws-glue/jobs/logs-v2-`*`<Security-Configuration-Name>`*
- Il gruppo di log continuo di default sarà *`<custom-log-group-name>`*`-`*`<Security-Configuration-Name>`*

È necessario aggiungere le autorizzazioni `logs:AssociateKmsKey` al ruolo IAM, se si abilita una configurazione di sicurezza con Logs. CloudWatch Se tale autorizzazione non è inclusa, la registrazione continua verrà disabilitata. Inoltre, per configurare la crittografia per CloudWatch i log, segui le istruzioni in Encrypt Log Data in CloudWatch Logs Using nella Amazon CloudWatch Logs AWS Key Management Service User Guide.

Per ulteriori informazioni sulla creazione delle configurazioni di sicurezza, consulta Uso di configurazioni di sicurezza nella console AWS Glue.

Visualizzazione della registrazione continua dei processi AWS Glue

Puoi visualizzare i log in tempo reale utilizzando la console AWS Glue o la console Amazon CloudWatch.

Per visualizzare i log in tempo reale tramite il pannello di controllo della console AWS Glue

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione scegliere Jobs (Processi).
3. Aggiungere o avviare un processo esistente. Scegliere Action (Operazione), Run job (Esegui processo).

   Quando avvii l'esecuzione di un processo, puoi accedere a una pagina che contiene informazioni sul processo in esecuzione:

- La scheda Logs (Log) mostra i precedenti log dell'applicazione aggregati.
- La scheda Continuous logging (Registrazione continua) mostra una barra di avanzamento in tempo reale quando il processo è in esecuzione con `glueContext` inizializzato.
- La scheda Continuous logging (Registrazione continua) contiene inoltre Driver logs (Log driver), che acquisisce log di driver Apache Spark in tempo reale e log di applicazioni dallo script registrato tramite il logger di applicazione AWS Glue quando il processo è in esecuzione.

4. Per processi precedenti, è anche possibile visualizzare i log in tempo reale nella vista Job History (Cronologia processi) scegliendo Logs (Log). Questa operazione porta alla console CloudWatch che mostra tutti i flussi di log di driver Spark, executor e barra di avanzamento l'esecuzione di quel processo.

Per visualizzare i log in tempo reale tramite il pannello di controllo della console CloudWatch

1. Aprire la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Nel riquadro di navigazione selezionare Log.
3. Scegliere il gruppo di log /aws-glue/jobs/logs-v2/.
4. Nella casella Filter (Filtro), incollare l'ID dell'esecuzione del processo.

   È possibile visualizzare i log di driver, i log di executor e la barra di avanzamento (se si utilizza Standard filter (Filtro standard)).

Monitoraggio con parametri AWS Glue di osservabilità

> **Note**
>
> I parametri di osservabilità AWS Glue sono disponibili in AWS Glue 4.0 e versioni successive.

Utilizza i parametri AWS Glue di osservabilità per generare approfondimenti su ciò che accade all'interno di AWS Glue per i processi di Apache Spark e migliorare la classificazione e l'analisi dei problemi. I parametri di osservabilità vengono visualizzati tramite i pannelli di controllo Amazon CloudWatch e possono essere utilizzati per aiutare a eseguire l'analisi delle cause principali degli errori e diagnosticare i rallentamenti delle prestazioni. È possibile ridurre il tempo impiegato per il debug dei problemi su larga scala così da poterti concentrare sulla risoluzione dei problemi in modo più rapido ed efficace.

AWS GlueL'osservabilità fornisce Amazon CloudWatch metriche classificate nei seguenti quattro gruppi:

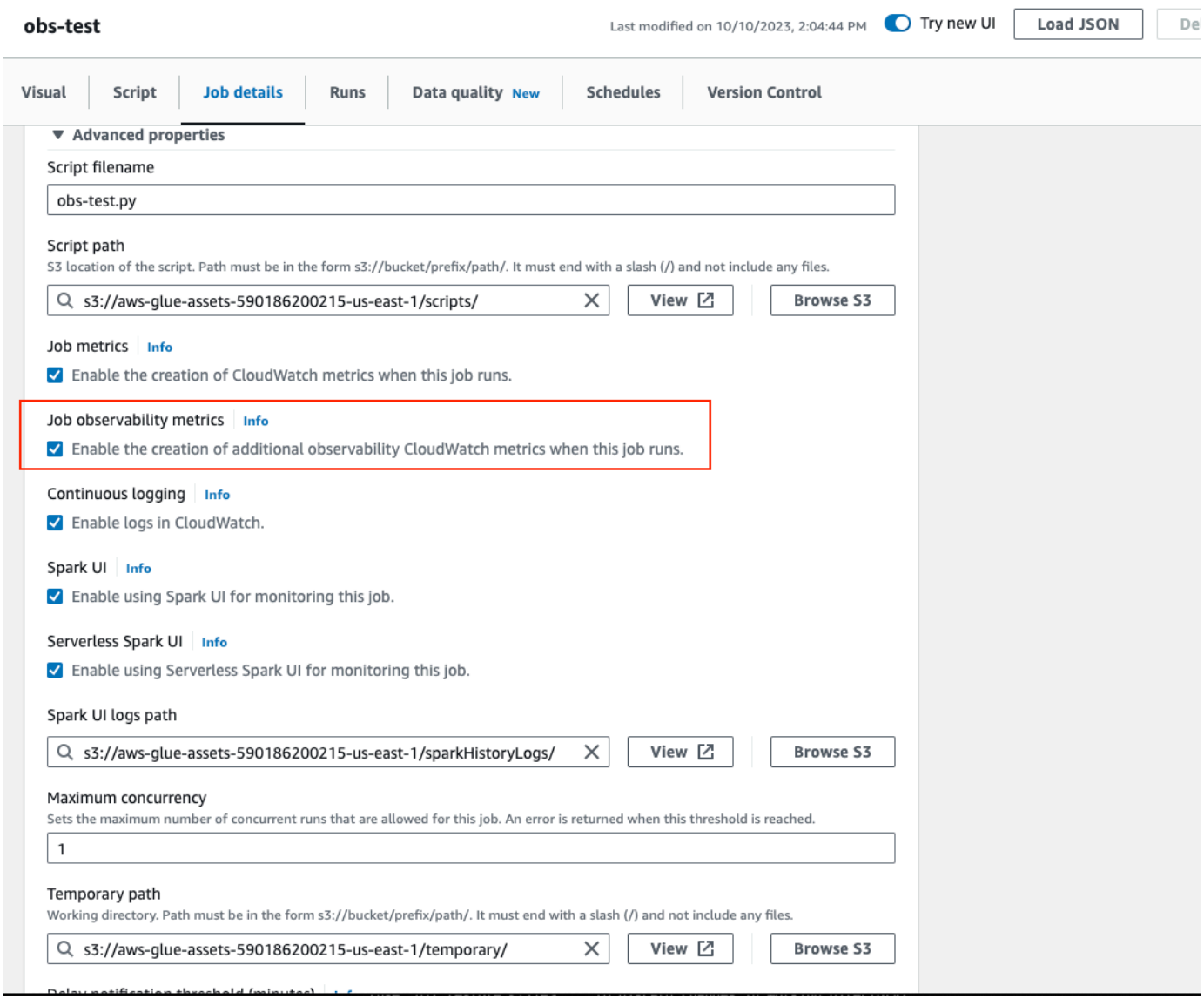
- Affidabilità (ad esempio, classi di errori): identifica facilmente i motivi di errore più comuni in un determinato intervallo di tempo che potresti voler risolvere.
- Prestazioni (ad esempio, asimmetria): individua un ostacolo prestazionale e applica tecniche di ottimizzazione. Ad esempio, quando riscontri un peggioramento delle prestazioni a causa dell'asimmetria del processo, potresti voler abilitare l'esecuzione delle query adattive Spark e ottimizzare la soglia di unione skew.
- Velocità di trasmissione effettiva (ossia, velocità effettiva per sorgente/sink): monitora le tendenze delle letture e scritture dei dati. Puoi anche configurare Amazon CloudWatch allarmi per anomalie.
- Utilizzo delle risorse (ad esempio, personale, utilizzo della memoria e del disco): individuazione efficiente dei processi con un basso utilizzo della capacità. Potresti voler abilitare il dimensionamento automatico AWS Glue per questi processi.

Guida introduttiva ai parametri AWS Glue di osservabilità

> **Note**
>
> I nuovi parametri sono abilitati per impostazione predefinita nella console AWS Glue Studio.

Per configurare i parametri di osservabilità in AWS Glue Studio:

1. Accedi alla console AWS Glue e scegli processi ETL dal menu della console.
2. Scegli un processo facendo clic sul suo nome nella sezione I tuoi processi.
3. Seleziona la scheda Job details (Dettagli del processo).
4. Scorri verso il basso e scegli Proprietà avanzate, quindi Parametri di osservabilità del processo.



Per abilitare le metriche di AWS Glue osservabilità utilizzando: AWS CLI

- Aggiungi alla mappa `--default-arguments` il seguente valore-chiave nel file JSON di input:

```
--enable-observability-metrics, true
```

Utilizzo dell'osservabilità AWS Glue

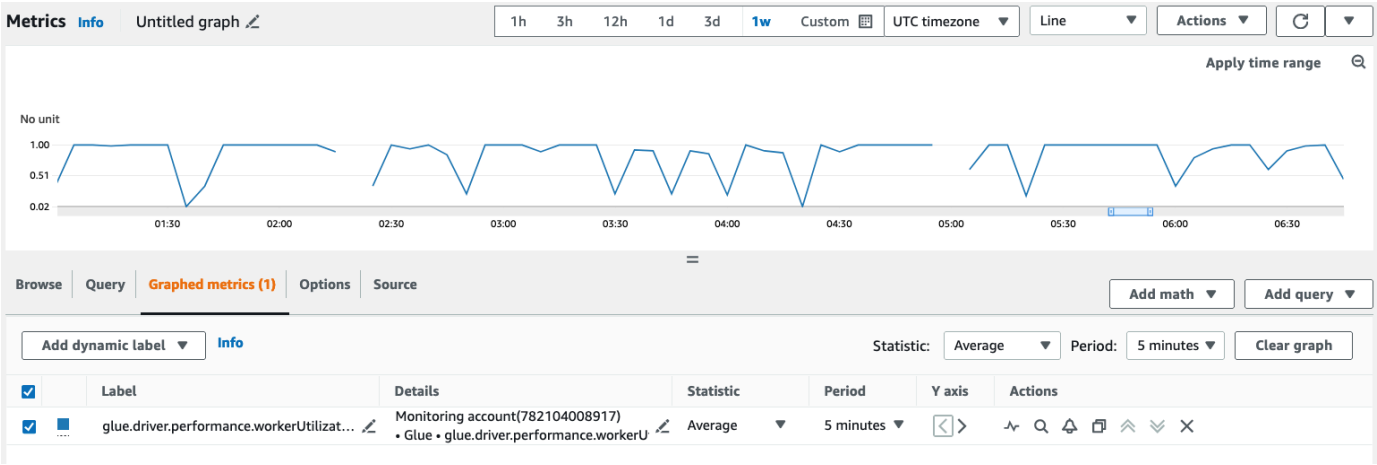
Poiché le metriche AWS Glue di osservabilità vengono fornite tramite Amazon CloudWatch, puoi utilizzare la Amazon CloudWatch console, l'SDK o l'API per interrogare i AWS CLI datapoint delle

metriche di osservabilità. Consulta Utilizzo dell'osservabilità Glue per monitorare l'utilizzo delle risorse per ridurre i costi per un caso d'uso di esempio su quando utilizzare i parametri AWS Glue di osservabilità.

Utilizzo dell'osservabilità nella console AWS GlueAmazon CloudWatch

Per interrogare e visualizzare le metriche nella console: Amazon CloudWatch

1. Apri la Amazon CloudWatch console e scegli Tutte le metriche.
2. In Spazi dei nomi personalizzati, seleziona AWS Glue.
3. Scegli Parametri di osservabilità del processo, Parametri di osservabilità per origine oppure Parametri di osservabilità per Sink.
4. Cerca il nome specifico del parametro, il nome del processo, l'ID di esecuzione del processo e selezionali.
5. Nella scheda Parametri nel grafico, configura la statistica, il periodo e altre opzioni che preferisci.



Per interrogare una metrica di osservabilità utilizzando: AWS CLI

1. Crea un file JSON di definizione dei parametri e sostituisci `your-Glue-job-name` e `your-Glue-job-run-id` con quelli pertinenti.

```
$ cat multiplequeries.json
[
    {
        "Id": "avgWorkerUtil_0",
        "MetricStat": {
            "Metric": {
                "Namespace": "Glue",
```

```
                "MetricName": "glue.driver.workerUtilization",
                "Dimensions": [
                    {
                        "Name": "JobName",
                        "Value": "<your-Glue-job-name-A>"
                    },
                    {
                        "Name": "JobRunId",
                        "Value": "<your-Glue-job-run-id-A>"
                    },
                    {
                        "Name": "Type",
                        "Value": "gauge"
                    },
                    {
                        "Name": "ObservabilityGroup",
                        "Value": "resource_utilization"
                    }
                ]
            },
            "Period": 1800,
            "Stat": "Minimum",
            "Unit": "None"
        }
    },
    {
        "Id": "avgWorkerUtil_1",
        "MetricStat": {
            "Metric": {
                "Namespace": "Glue",
                "MetricName": "glue.driver.workerUtilization",
                "Dimensions": [
                    {
                        "Name": "JobName",
                        "Value": "<your-Glue-job-name-B>"
                    },
                    {
                        "Name": "JobRunId",
                        "Value": "<your-Glue-job-run-id-B>"
                    },
                    {
                        "Name": "Type",
                        "Value": "gauge"
                    },
```

```
                    {
                        "Name": "ObservabilityGroup",
                        "Value": "resource_utilization"
                    }
                ]
            },
            "Period": 1800,
            "Stat": "Minimum",
            "Unit": "None"
        }
    }
]
```

2. Eseguire il comando `get-metric-data`:

```
$ aws cloudwatch get-metric-data --metric-data-queries file: //multiplequeries.json
 \
     --start-time '2023-10-28T18: 20' \
     --end-time '2023-10-28T19: 10'  \
     --region us-east-1
{
    "MetricDataResults": [
        {
            "Id": "avgWorkerUtil_0",
            "Label": "<your-label-for-A>",
            "Timestamps": [
                "2023-10-28T18:20:00+00:00"
            ],
            "Values": [
                0.06718750000000001
            ],
            "StatusCode": "Complete"
        },
        {
            "Id": "avgWorkerUtil_1",
            "Label": "<your-label-for-B>",
            "Timestamps": [
                "2023-10-28T18:50:00+00:00"
            ],
            "Values": [
                0.5959183673469387
            ],
```

```
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Parametri di osservabilità

AWS GlueL'osservabilità profila e invia le seguenti metriche Amazon CloudWatch ogni 30 secondi e alcune di queste metriche possono essere visibili nella pagina AWS Glue Studio Job Runs Monitoring.

Parametro	Descrizione	Categoria
glue.driver.skewness.stage	Categoria parametro: job_performance L'asimmetria di esecuzion e delle fasi di Spark: questo parametro rileva l'asimmetria di esecuzione, che potrebbe essere causata dall'asim metria dei dati di input o da una trasformazione (ad es. join asimmetrico). I valori di questo parametro rientrano nell'intervallo [0, infinito], dove 0 indica il rapporto tra il tempo di esecuzione massimo e quello medio delle attività. Tra tutte le attività nella fase, è inferiore a un determinato fattore di asimmetria della stessa. Il fattore predefinito di asimmetria della fase è `5` e può essere sovrascri tto tramite la configurazione	job_performance

Parametro	Descrizione	Categoria
	spark: spark.metrics.conf .driver.source.glue.jobPerf ormance.skewnessFactor Un valore di asimmetria della fase pari a 1 significa che il rapporto è il doppio del fattore di asimmetria della fase. Il valore dell'asimmetria della fase viene aggiornato ogni 30 secondi per riflettere l'asimmet ria corrente. Il valore alla fine della fase riflette l'asimmetria della fase finale. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabi lityGroup (job_performance) Statistiche valide: media, massimo, minimo, percentuale Unità: numero	

Parametro	Descrizione	Categoria
glue.driver.skewness.job	Categoria parametro: job_performance L'asimmetria del processo corrisponde alla media ponderata dell'asimmetria delle fasi del processo. La media ponderata dà un peso maggiore alle fasi che richiedono più tempo per essere eseguite. In questo modo si evita il caso limite in cui una fase molto asimmetri ca viene eseguita per un periodo molto breve rispetto ad altre fasi (quindi la sua asimmetria non è significativa per le prestazioni complessi ve del processo e non vale la pena cercare di correggerla). Questo parametro viene aggiornato al completam ento di ogni fase, perciò l'ultimo valore riflette l'effettiva asimmetria complessiva del processo. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabi lityGroup (job_performance) Statistiche valide: media, massimo, minimo, percentuale	job_performance

Parametro	Descrizione	Categoria
	Unità: numero	
glue.succeed.ALL	Categoria parametro: errore Numero totale di processi eseguiti con successo, per completare il quadro delle categorie di errori Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (count) e Observabi lityGroup (error) Statistiche valide: SOMMA Unità: numero	error
glue.error.ALL	Categoria parametro: errore Numero totale di errori di esecuzione del processo, per completare il quadro delle categorie di errori Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (count) e Observabi lityGroup (error) Statistiche valide: SOMMA Unità: numero	error

Parametro	Descrizione	Categoria
glue.error.[error category]	Categoria parametro: errore Questo insieme di parametri viene aggiornato solo se l'esecuzione di un processo fallisce. La categorizzazione degli errori facilita la classific azione e il debug. Quando l'esecuzione di un processo fallisce, la causa dell'erro re viene classificata e il parametro della categoria di errore corrispondente viene impostato su 1. Ciò consente di eseguire l'analisi degli errori nel corso tempo, nonché quella relativa a tutti i processi, per identificare le categorie di errore più comuni e risolverle. AWS Glue include 28 categorie di errore, tra cui OUT_OF_ME MORY (driver ed executor) , AUTORIZZAZIONE, SINTASSI e LIMITAZIONE (DELLA LARGHEZZA DI BANDA DELLA RETE). Le categorie di errore includono anche COMPILAZIONE, AVVIO e TIMEOUT. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL),	error

Parametro	Descrizione	Categoria
	Type (count) e Observabi lityGroup (error) Statistiche valide: SOMMA Unità: numero	
glue.driver.workerUtilization	Categoria parametro: resource_utilization La percentuale dei worker allocati che vengono effettiva mente utilizzati. Se non va bene, può essere utile il dimensionamento automatico. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabi lityGroup (resource_utilization) Statistiche valide: media, massimo, minimo, percentuale Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.memory.heap.[available | used]	Categoria parametro: resource_utilization La memoria heap del driver disponibile/utilizzata durante l'esecuzione del processo. Ciò è utile per comprendere le tendenze di utilizzo della memoria, soprattutto nel tempo, il che può contribuire a evitare potenziali errori e a eseguirne il debug. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: byte	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.memory.heap.used.percentage	Categoria parametro: resource_utilization La memoria heap del driver utilizzata (%) durante l'esecuzione del processo. Ciò è utile per comprendere le tendenze di utilizzo della memoria, soprattutto nel tempo, il che può contribuire a evitare potenziali errori e a eseguirne il debug. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.memory.non-heap.[available | used]	Categoria parametro: resource_utilization La memoria non heap del driver disponibile/utilizzata durante l'esecuzione del processo. Ciò è utile per comprendere le tendenze di utilizzo della memoria, soprattutto nel tempo, il che può contribuire a evitare potenziali errori e a eseguirne il debug. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabilità Group (resource_utilization) Statistiche valide: media Unità: byte	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.memory.non-heap .used.percentage	Categoria parametro: resource_utilization La memoria non heap del driver utilizzata (%) durante l'esecuzione del processo. Ciò è utile per comprendere le tendenze di utilizzo della memoria, soprattutto nel tempo, il che può contribuire a evitare potenziali errori e a eseguirne il debug. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabi lityGroup (resource_utilization) Statistiche valide: media Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.memory.total.[available | used]	Categoria parametro: resource_utilization La memoria totale del driver disponibile/utilizzata durante l'esecuzione del processo. Ciò è utile per comprendere le tendenze di utilizzo della memoria, soprattutto nel tempo, il che può contribuire a evitare potenziali errori e a eseguirne il debug. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: byte	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.memory.total.used.percentage	Categoria parametro: resource_utilization La memoria totale del driver utilizzata (%) durante l'esecuzione del processo. Ciò è utile per comprendere le tendenze di utilizzo della memoria, soprattutto nel tempo, il che può contribuire a evitare potenziali errori e a eseguirne il debug. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.ALL.memory.heap.[available | used]	Categoria parametro: resource_utilization La memoria heap degli executor disponibile/utilizzata. ALL significa tutti gli executor. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: byte	resource_utilization
glue.ALL.memory.heap.used.percentage	Categoria parametro: resource_utilization La memoria heap degli executor utilizzata (%). ALL significa tutti gli executor. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.ALL.memory.non-heap.[available | used]	Categoria parametro: resource_utilization La memoria non heap degli executor disponibile/utilizzata. ALL significa tutti gli executor. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: byte	resource_utilization
glue.ALL.memory.non-heap.used.percentage	Categoria parametro: resource_utilization La memoria non heap degli executor utilizzata (%). ALL significa tutti gli executor. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.ALL.memory.total.[available | used]	Categoria parametro: resource_utilization La memoria totale degli executor disponibile/utilizzata. ALL significa tutti gli executor. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: byte	resource_utilization
glue.ALL.memory.total.used.percentage	Categoria parametro: resource_utilization La memoria totale degli executor utilizzata (%). ALL significa tutti gli executor. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e ObservabilityGroup (resource_utilization) Statistiche valide: media Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.disk.[available_GB | used_GB]	Categoria parametro: resource_utilization Lo spazio su disco del driver disponibile/utilizzato durante l'esecuzione del processo. Ciò è utile per comprende re le tendenze di utilizzo del disco, soprattutto nel tempo, il che può contribuire a evitare potenziali errori e a eseguire il debug di quelli relativi alla presenza di spazio non sufficiente sul disco. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabi lityGroup (resource_utilization) Statistiche valide: media Unità: gigabyte	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.disk.used.percentage]	Categoria parametro: resource_utilization Lo spazio su disco del driver disponibile/utilizzato durante l'esecuzione del processo. Ciò è utile per comprende re le tendenze di utilizzo del disco, soprattutto nel tempo, il che può contribuire a evitare potenziali errori e a eseguire il debug di quelli relativi alla presenza di spazio non sufficiente sul disco. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabi lityGroup (resource_utilization) Statistiche valide: media Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.ALL.disk.[available_GB | used_GB]	Categoria parametro: resource_utilization Lo spazio su disco degli executor disponibile/utilizzato. ALL significa tutti gli executor. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabi lityGroup (resource_utilization) Statistiche valide: media Unità: gigabyte	resource_utilization
glue.ALL.disk.used.percenta ge	Categoria parametro: resource_utilization Lo spazio su disco degli executor disponibile/utilizzato/ utilizzato (%). ALL significa tutti gli executor. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge) e Observabi lityGroup (resource_utilization) Statistiche valide: media Unità: percentuale	resource_utilization

Parametro	Descrizione	Categoria
glue.driver.bytesRead	Categoria parametro: velocità di trasmissione effettiva Il numero di byte letti per ogni origine di input in questa esecuzione del processo e per TUTTE le origini. È possibile così comprendere il volume dei dati e le relative variazioni nel tempo, il che consente di risolvere problemi come l'asimmetria dei dati. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge), (resource_utilization) e Source ObservabilityGroup (posizione dei dati di origine) Statistiche valide: media Unità: byte	velocità di trasmissione effettiva

Parametro	Descrizione	Categoria
glue.driver.[recordsRead | filesRead]	Categoria parametro: velocità di trasmissione effettiva Il numero di record/file letti per ogni origine di input in questa esecuzione del processo e per TUTTE le origini. È possibile così comprendere il volume dei dati e le relative variazioni nel tempo, il che consente di risolvere problemi come l'asimmetria dei dati. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge), (resource_utilization) e Source ObservabilityGroup (posizione dei dati di origine) Statistiche valide: media Unità: numero	velocità di trasmissione effettiva

Parametro	Descrizione	Categoria
glue.driver.partitionsRead	Categoria parametro: velocità di trasmissione effettiva Il numero di partizioni lette per ogni origine di input di Amazon S3 in questa esecuzione del processo e per TUTTE le origini. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge), (resource_utilization) e Source ObservabilityGroup (posizione dei dati di origine) Statistiche valide: media Unità: numero	velocità di trasmissione effettiva

Parametro	Descrizione	Categoria
glue.driver.bytesWrittten	Categoria parametro: velocità di trasmissione effettiva Il numero di byte scritti per ogni sink di output in questa esecuzione del processo e per TUTTI i sink. È possibile così comprendere il volume dei dati e il modo in cui evolve nel tempo, il che consente di risolvere problemi come l'asimmetria dell'elaborazione. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge), (resource _utilization) e Sink Observabi lityGroup (posizione dei dati del sink) Statistiche valide: media Unità: byte	velocità di trasmissione effettiva

Parametro	Descrizione	Categoria
glue.driver.[recordsWritten | filesWritten]	Categoria parametro: velocità di trasmissione effettiva Il numero di record/file scritti per ogni sink di output in questa esecuzione del processo e per TUTTI i sink. È possibile così comprende re il volume dei dati e il modo in cui evolve nel tempo, il che consente di risolvere problemi come l'asimmetria dell'elab orazione. Dimensioni valide: JobName (il nome del AWS Glue Job), JobRunId (JobRun ID. o ALL), Type (gauge), (resource _utilization) e Sink Observabi lityGroup (posizione dei dati del sink) Statistiche valide: media Unità: numero	velocità di trasmissione effettiva

Categorie di errore

Categorie di errore	Descrizione
COMPILATION_ERROR	Gli errori si verificano durante la compilazione del codice Scala.

Categorie di errore	Descrizione
CONNECTION_ERROR	Gli errori si verificano durante la connessione a un servizio/host remoto/servizio di database, ecc.
DISK_NO_SPACE_ERROR	Gli errori si verificano quando non c'è più spazio nel disco sul driver/executor.
OUT_OF_MEMORY_ERROR	Gli errori si verificano quando non c'è più spazio nella memoria sul driver/executor.
IMPORT_ERROR	Gli errori si verificano durante l'importazione delle dipendenze.
INVALID_ARGUMENT_ERROR	Gli errori sorgono quando gli argomenti di input non sono validi/illegali.
PERMISSION_ERROR	Gli errori si verificano in mancanza di autorizza zioni per il servizio, per i dati, ecc.
RESOURCE_NOT_FOUND_ERROR	Gli errori si verificano quando i dati, la posizione, ecc. non esistono.
QUERY_ERROR	Gli errori derivano dall'esecuzione delle query di Spark SQL.
SYNTAX_ERROR	Gli errori si verificano quando nello script è presente un errore di sintassi.
THROTTLING_ERROR	Gli errori si verificano quando si supera la limitazione della concorrenza del servizio o il limite della quota di servizio.
DATA_LAKE_FRAMEWORK_ERROR	Gli errori derivano da framework data lake supportati nativamente da AWS Glue, come Hudi, Iceberg, ecc.

Categorie di errore	Descrizione
UNSUPPORTED_OPERATION_ERROR	Gli errori si verificano quando si eseguono operazioni non supportate.
RESOURCES_ALREADY_EXISTS_ERROR	Gli errori si verificano quando una risorsa da creare o aggiungere esiste già.
GLUE_INTERNAL_SERVICE_ERROR	Gli errori si verificano quando c'è un problema interno al servizio AWS Glue.
GLUE_OPERATION_TIMEOUT_ERROR	Gli errori si verificano quando un'operazione AWS Glue è in timeout.
GLUE_VALIDATION_ERROR	Gli errori si verificano quando un valore richiesto non può essere convalidato per un processo AWS Glue.
GLUE_JOB_BOOKMARK_VERSION_MISMATCH_ERROR	Gli errori si verificano quando uno stesso processo è in esecuzione su uno stesso bucket di origine e scrive contemporaneamente nella stessa destinazione o in una destinazione diversa (simultaneità >1)
LAUNCH_ERROR	Gli errori si verificano durante la fase di avvio del processo AWS Glue.
DYNAMODB_ERROR	Gli errori generici derivano dal servizio. Amazon DynamoDB
GLUE_ERROR	Gli errori generici derivano dal servizio AWS Glue.
LAKEFORMATION_ERROR	Gli errori generici derivano dal AWS Lake Formation servizio.
REDSHIFT_ERROR	Gli errori generici derivano dal Amazon Redshift servizio.

Categorie di errore	Descrizione
S3_ERROR	Gli errori generici derivano dal servizio Amazon S3.
SYSTEM_EXIT_ERROR	Errore generico di uscita dal sistema.
TIMEOUT_ERROR	Gli errori generici si verificano quando il processo fallisce per timeout dell'operazione.
UNCLASSIFIED_SPARK_ERROR	Gli errori generici derivano da Spark.
UNCLASSIFIED_ERROR	Categoria di errore predefinita.

Limitazioni

> **Note**
>
> `glueContext` deve essere inizializzato per poter pubblicare i parametri.

Nella dimensione di origine, il valore corrisponde al percorso o al nome della tabella Amazon S3, a seconda del tipo di origine. Inoltre, se l'origine è JDBC e viene utilizzata l'opzione di query, la stringa di query viene impostata nella dimensione di origine. Se il valore supera i 500 caratteri, viene ridotto per rispettare questo limite. Di seguito sono riportate le limitazioni del valore:

- I caratteri non ASCII verranno rimossi.
- Se il nome dell'origine non contiene alcun carattere ASCII, verrà convertito in <non-ASCII input>.

Limitazioni e considerazioni relative ai parametri della velocità di trasmissione effettiva

- DataFrame e DataFrame based DynamicFrame (ad esempio JDBC, lettura da parquet su Amazon S3) sono supportati, mentre quelli DynamicFrame basati su RDD (ad esempio la lettura di csv, json su Amazon S3, ecc.) non sono supportati. Tecnicamente, tutte le letture e le scritture visibili sull'interfaccia utente di Spark sono supportate.
- Il parametro `recordsRead` viene emesso se l'origine dati è una tabella di catalogo e il formato è JSON, CSV, testo o Iceberg.

- I parametri `glue.driver.throughput.recordsWritten`, `glue.driver.throughput.bytesWritten` e `glue.driver.throughput.filesWritten` non sono disponibili nelle tabelle JDBC e Iceberg.
- I parametri potrebbero subire ritardi. Se il lavoro termina in circa un minuto, è possibile che in Metrics non sia presente alcuna metrica di throughput. Amazon CloudWatch

Monitoraggio e debug dei processi

È possibile raccogliere i parametri relativi ai processi AWS Glue e visualizzarli nelle console AWS Glue e Amazon CloudWatch per identificare e risolvere i problemi. Per la profilatura dei processi AWS Glue sono necessarie le fasi seguenti:

1. Abilitare i parametri:

   a. Abilitare l'opzione Job metrics (Parametri processo) nella definizione del processo. È possibile abilitare la profilatura nella console AWS Glue o come parametro per il processo. Per ulteriori informazioni, consultare Definire le proprietà di processo per i processi Spark o Parametri del processo AWS Glue.

   b. Abilita l'opzione Parametri AWS Glue di osservabilità nella definizione del processo. È possibile abilitare l'osservabilità nella console AWS Glue o come parametro per il processo. Per ulteriori informazioni, consulta Monitoraggio con parametri AWS Glue di osservabilità.

2. Verificare che lo script del processo inizializzi un oggetto `GlueContext`. Il frammento di script seguente inizializza ad esempio un oggetto `GlueContext` e mostra dove viene inserito il codice profilato nello script. Questo formato generale viene usato negli scenari di debug seguenti.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
import time

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
```

```
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

...
...
code-to-profile
...
...


job.commit()
```

3. Esegui il processo.
4. Visualizzare i parametri:

   a. Visualizza i parametri nella console AWS Glue e identifica quelli anomali per il driver o per un executor.

   b. Controlla i parametri di osservabilità nella pagina del monitoraggio dell'esecuzione dei processi, in quella dei dettagli sull'esecuzione dei processi o su Amazon CloudWatch. Per ulteriori informazioni, consulta Monitoraggio con parametri AWS Glue di osservabilità.
5. Risalire alla causa principale usando il parametro identificato.
6. Facoltativamente, confermare la causa principale usando il flusso di log del driver o dell'executor del processo identificato.

Casi d'uso per i parametri AWS Glue di osservabilità

- Debug di eccezioni di memoria esaurita (OOM) e anomalie dei processi
- Debug di fasi impegnative e attività in ritardo
- Monitoraggio dell'avanzamento di processi multipli
- Monitoraggio per la pianificazione della capacità DPU
- Utilizzo dell'osservabilità AWS Glue per monitorare l'utilizzo delle risorse per ridurre i costi

Debug di eccezioni di memoria esaurita (OOM) e anomalie dei processi

È possibile eseguire il debug di eccezioni di memoria esaurita e anomalie dei processi in AWS Glue. Nelle sezioni seguenti vengono descritti gli scenari per il debug di eccezioni di memoria esaurita del driver Apache Spark o di un executor Spark.

- Debug dell'eccezione di memoria esaurita (OOM) di un driver
- Debug di un'eccezione di memoria esaurita (OOM) dell'executor

Debug dell'eccezione di memoria esaurita (OOM) di un driver

In questo scenario un processo Spark sta leggendo un numero elevato di piccoli file da Amazon Simple Storage Service (Amazon S3). I file vengono convertiti nel formato Apache Parquet e quindi scritti in Amazon S3. Il driver Spark sta per esaurire la memoria. I dati Amazon S3 di input hanno più di 1 milione di file in partizioni Amazon S3 diverse.

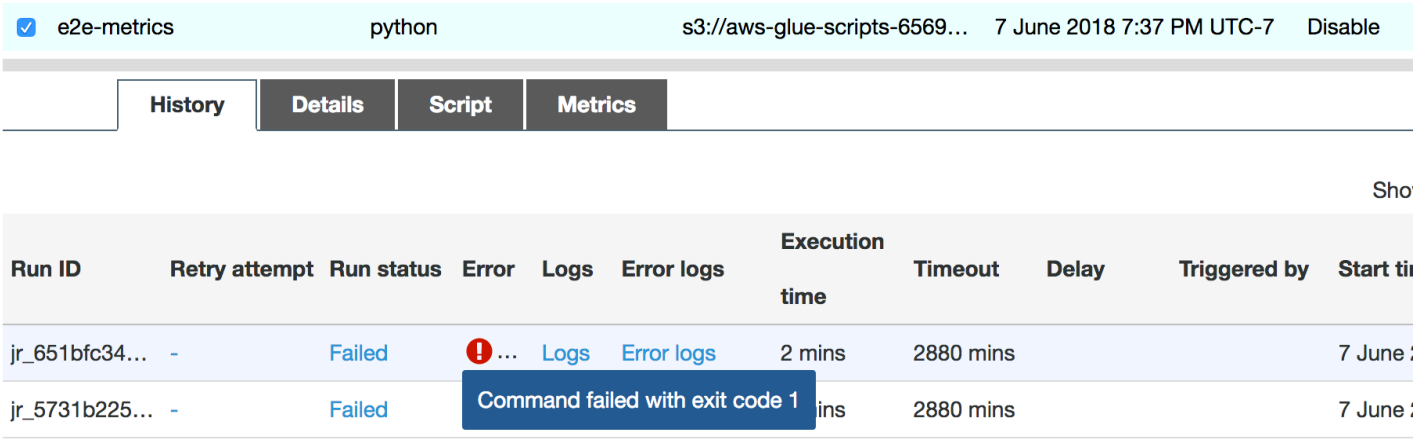
Il codice profilato è il seguente:

```
data = spark.read.format("json").option("inferSchema", False).load("s3://input_path")
data.write.format("parquet").save(output_path)
```

Visualizzazione dei parametri profilati nella console AWS Glue

Il grafico seguente mostra l'utilizzo di memoria sotto forma di percentuale per il driver e gli executor. Il grafico dell'utilizzo viene tracciato usando punti dati che rappresentano la media dei valori segnalati nell'ultimo minuto. Nel profilo di memoria del processo è possibile vedere che la memoria del driver supera la soglia di sicurezza del 50% di utilizzo in modo rapido. L'utilizzo di memoria medio di tutti gli executor rimane invece inferiore al 4%. Ciò indica chiaramente un'anomalia nell'esecuzione del driver nel processo Spark.

L'esecuzione del processo ha esito negativo in breve tempo e viene visualizzato l'errore seguente nella scheda History (Cronologia) della console AWS Glue: Command Failed with Exit Code 1 (Comando non riuscito con codice di uscita 1). Questa stringa di errore indica che il processo non è riuscito a causa di un errore di sistema, che in questo caso è l'esaurimento di memoria del driver.



Nella console fai clic sul collegamento Error logs (Log di errore) nella scheda History (Cronologia) per verificare l'esaurimento della memoria del driver in CloudWatch Logs. Cerca "**`Error`**" nel log di errori del processo per verificare che la mancata riuscita del processo sia effettivamente dovuta a un'eccezione di memoria esaurita:

```
# java.lang.OutOfMemoryError: Java heap space
```

```
# -XX:OnOutOfMemoryError="kill -9 %p"
# Executing /bin/sh -c "kill -9 12039"...
```

Nella scheda History (Cronologia) per il processo scegli Logs (Log). Puoi trovare la seguente traccia dell'esecuzione del driver in CloudWatch Logs all'inizio del processo. Il driver Spark cerca di elencare tutti i file in tutte le directory, crea una oggetto `InMemoryFileIndex` e avvia un'attività per ogni file. Di conseguenza, il driver Spark deve gestire una quantità elevata di stato in memoria per tenere traccia di tutte le attività. Viene memorizzato nella cache l'elenco completo di un numero elevato di file per l'indice in memoria, provocando l'esaurimento della memoria del driver.

Correzione dell'elaborazione di più file mediante il raggruppamento

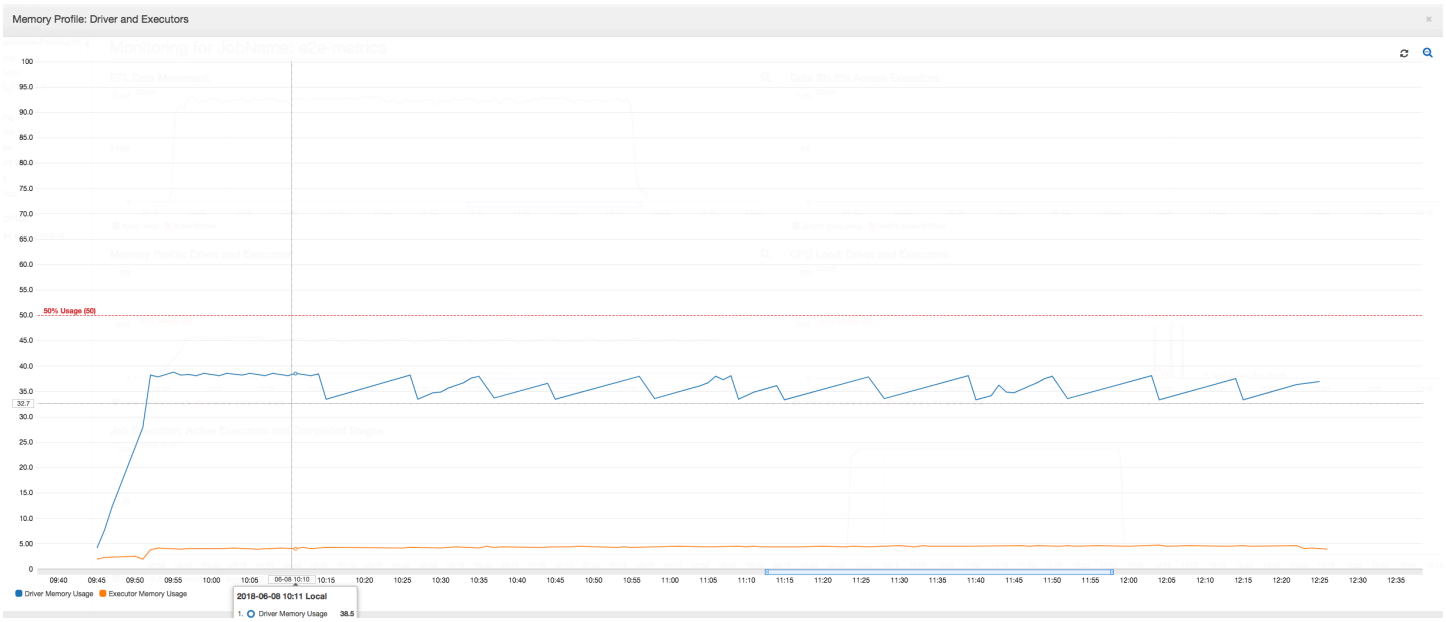
È possibile correggere l'elaborazione di più file usando la caratteristica di raggruppamento in AWS Glue. Il raggruppamento viene abilitato automaticamente quando si usano frame dinamici e quando il set di dati di input include un numero elevato di file (più di 50.000). Il raggruppamento permette di unire più file in un gruppo e permette a un'attività di elaborare l'intero gruppo invece di un singolo file. Di conseguenza, il driver Spark archivia una quantità significativamente inferiore di stato in memoria per tenere traccia di un numero minore di attività. Per ulteriori informazioni sull'abilitazione manuale del raggruppamento per un set di dati, consulta Lettura di file di input in gruppi di grandi dimensioni.

Per controllare il profilo di memoria del processo AWS Glue, profila il codice seguente con il raggruppamento abilitato:

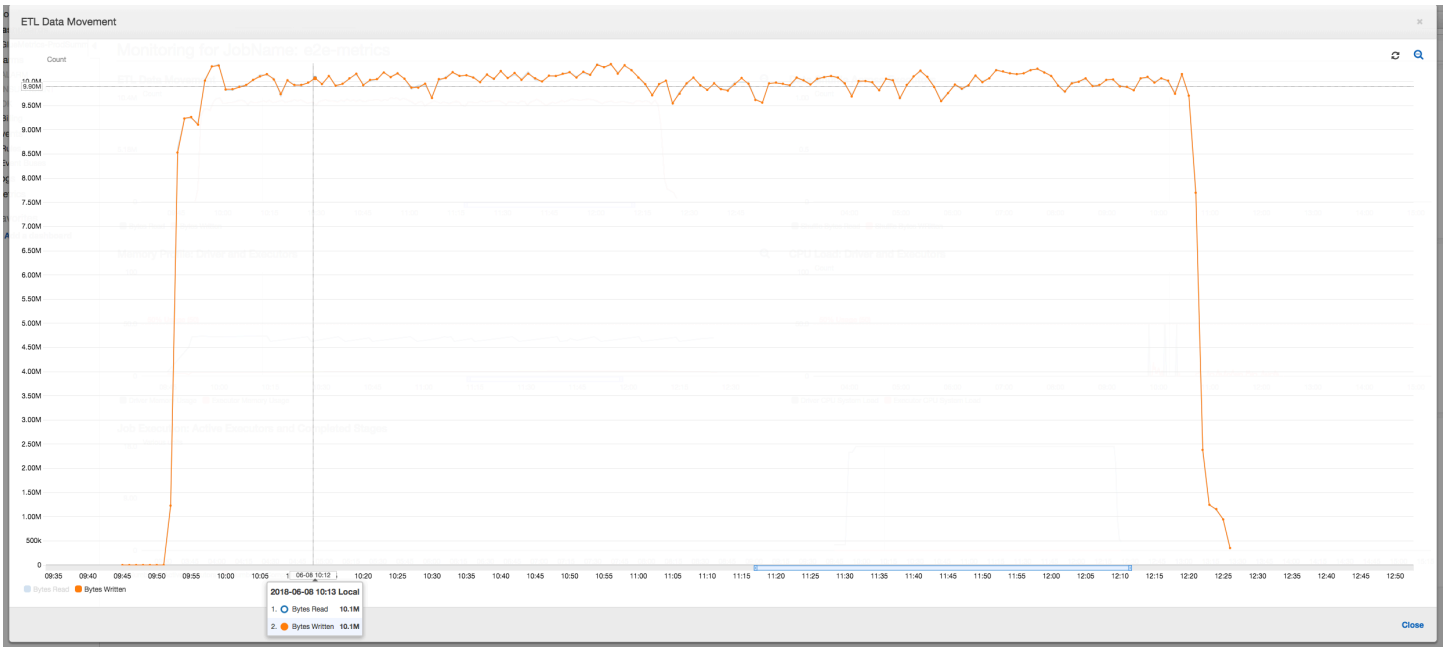
```
df = glueContext.create_dynamic_frame_from_options("s3", {'paths': ["s3://input_path"],
 "recurse":True, 'groupFiles': 'inPartition'}, format="json")
datasink = glueContext.write_dynamic_frame.from_options(frame = df, connection_type
 = "s3", connection_options = {"path": output_path}, format = "parquet",
 transformation_ctx = "datasink")
```

È possibile monitorare il profilo di memoria e lo spostamento di dati ETL nel profilo del processo AWS Glue.

Il driver viene eseguito sotto la soglia del 50% di utilizzo di memoria per l'intera durata del processo AWS Glue. Gli executor trasmettono i dati da Amazon S3, li elaborano e li scrivono in Amazon S3. Di conseguenza, usano meno del 5% di memoria in qualsiasi momento.



Il profilo di spostamento dei dati seguente mostra il numero totale di byte Amazon S3 che vengono letti e scritti nell'ultimo minuto da tutti gli executor man mano che il processo avanza. In entrambi i casi viene seguito un modello simile mentre i dati vengono trasmessi tra tutti gli executor. Il processo completa l'elaborazione di tutto il milione di file in meno di tre ore.



Debug di un'eccezione di memoria esaurita (OOM) dell'executor

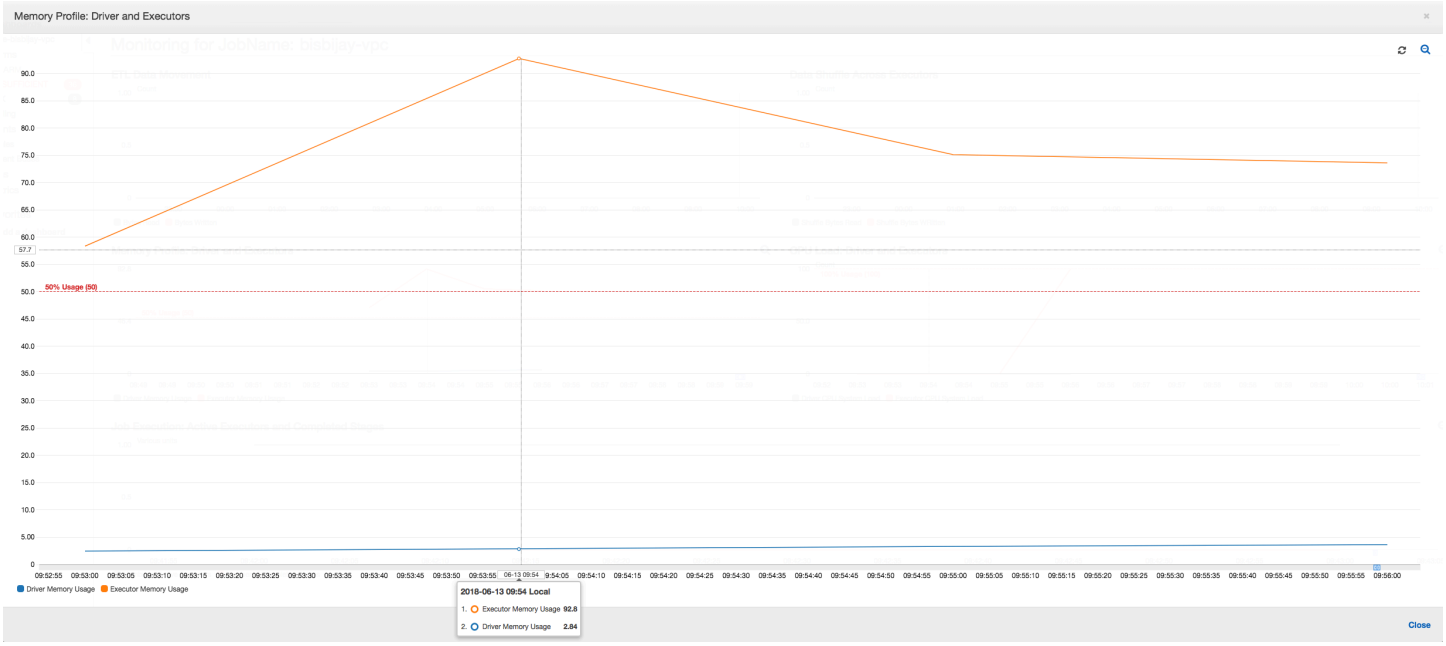
In questo scenario è possibile imparare a eseguire il debug delle eccezioni di memoria esaurita che potrebbero verificarsi negli executor Apache Spark. Il codice seguente usa il lettore Spark MySQL per leggere una tabella di grandi dimensioni con circa 34 milioni di righe in un dataframe Spark. Scrive

quindi i dati in Amazon S3 in formato Parquet. È possibile fornire le proprietà di connessione e usare le configurazioni Spark predefinite per leggere la tabella.

```
val connectionProperties = new Properties()
connectionProperties.put("user", user)
connectionProperties.put("password", password)
connectionProperties.put("Driver", "com.mysql.jdbc.Driver")
val sparkSession = glueContext.sparkSession
val dfSpark = sparkSession.read.jdbc(url, tableName, connectionProperties)
dfSpark.write.format("parquet").save(output_path)
```

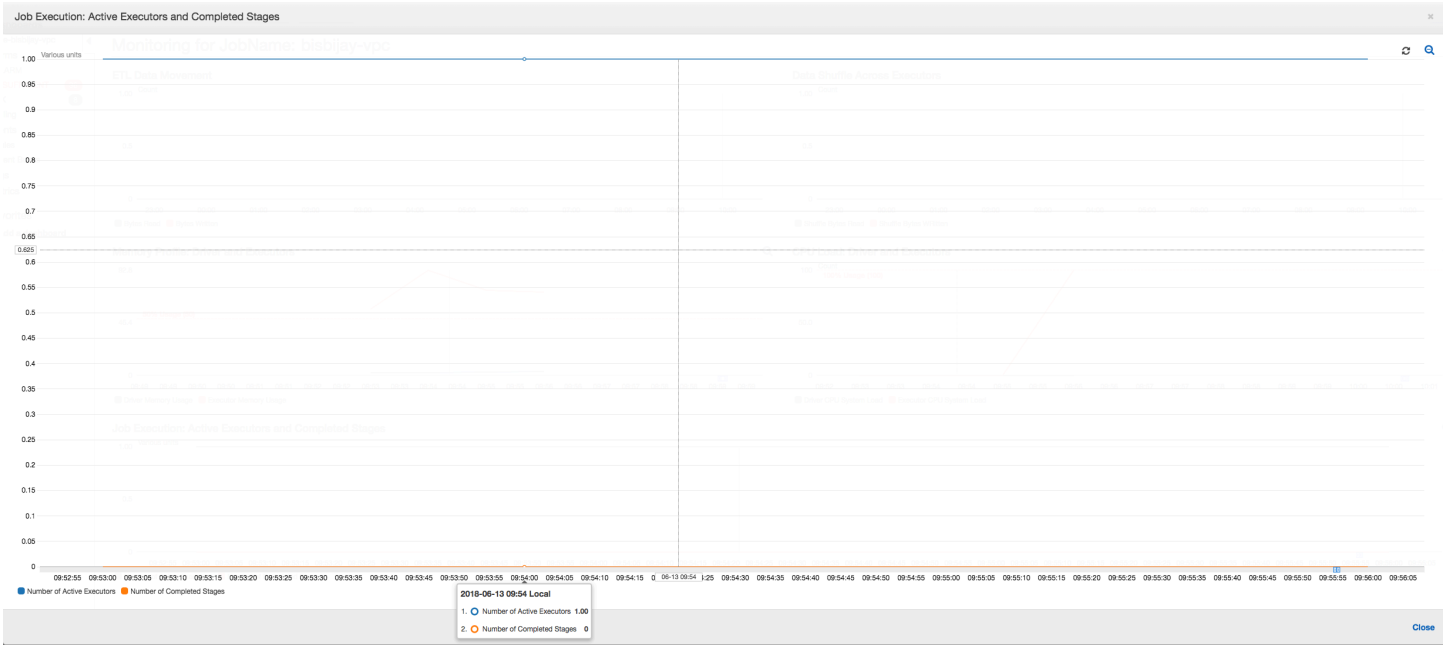
Visualizzazione dei parametri profilati nella console AWS Glue

Se la pendenza del grafico di utilizzo della memoria è positiva e supera il 50% e se il processo non riesce prima che venga emesso il parametro successivo, l'esaurimento della memoria può probabilmente essere la causa. Il grafico seguente mostra che entro un minuto di esecuzione, l'utilizzo di memoria medio in tutti gli executor sale rapidamente sopra il 50%. L'utilizzo raggiunge il 92% e il container che esegue l'executor viene interrotto da Apache Hadoop YARN.

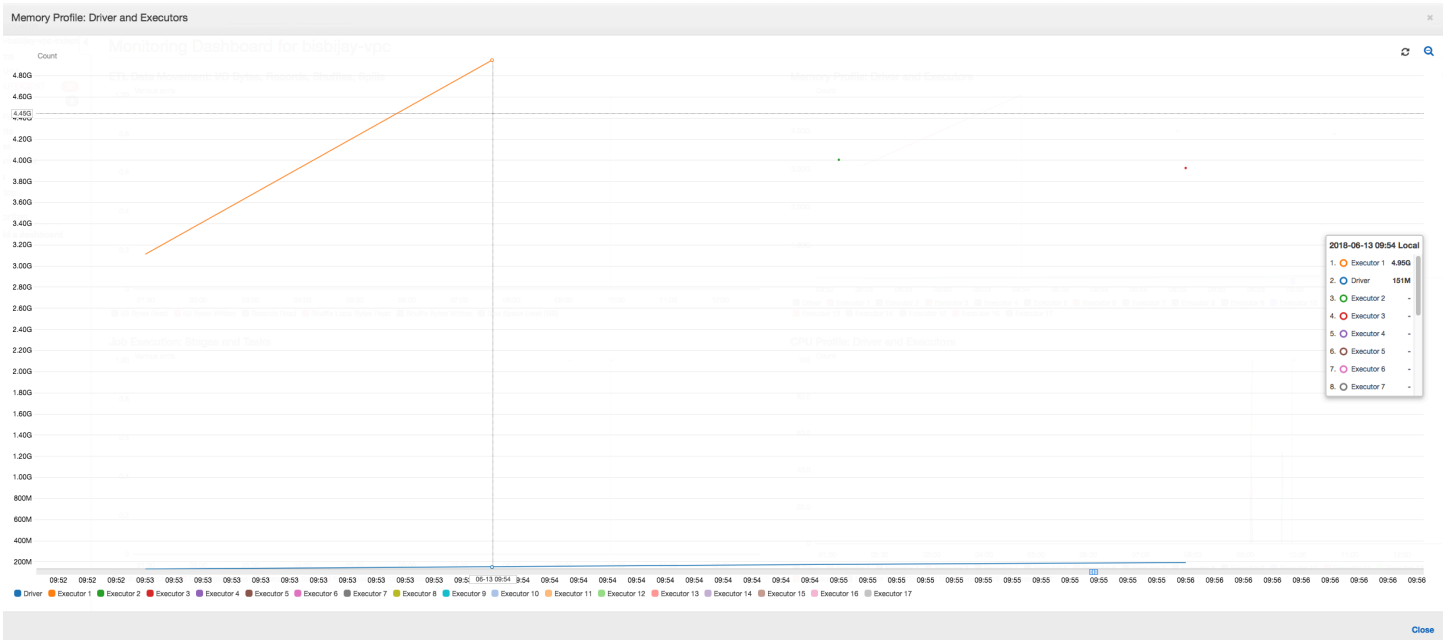


Come mostra il grafico seguente, c'è sempre un singolo executor in esecuzione fino a quando il processo non ha esito negativo. Ciò avviene perché viene avviato un nuovo executor per sostituire quello interrotto. Le operazioni di lettura dell'origine dati JDBC non sono parallelizzate per impostazione predefinita perché ciò richiederebbe il partizionamento della tabella in una

colonna e l'apertura di più connessioni. Di conseguenza, un solo executor legge la tabella completa sequenzialmente.



Come mostra il grafico seguente, Spark cerca di avviare una nuova attività quattro volte prima che il processo abbia esito negativo. È possibile visualizzare il profilo di memoria di tre executor. Ogni executor consuma rapidamente tutta la relativa memoria. Il quarto executor esaurisce la memoria e il processo ha esito negativo. Di conseguenza, il relativo parametro non viene segnalato immediatamente.



È possibile verificare dalla stringa di errore nella console AWS Glue che il processo ha avuto esito negativo a causa di eccezioni di memoria esaurita, come illustrato nell'immagine seguente.

Log di output del processo: per confermare ulteriormente l'eccezione di memoria esaurita dell'executor, puoi esaminare CloudWatch Logs. Eseguendo la ricerca di **`Error`**, puoi trovare quattro executor interrotti circa nella stessa finestra temporale, come indicato nel pannello di controllo dei parametri. Gli executor sono stati terminati tutti da YARN a causa del superamento dei limiti di memoria.

Executor 1

```
18/06/13 16:54:29 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 ERROR YarnClusterScheduler: Lost executor 1 on
 ip-10-1-2-175.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 WARN TaskSetManager: Lost task 0.0 in stage 0.0 (TID 0,
 ip-10-1-2-175.ec2.internal, executor 1): ExecutorLostFailure (executor 1
 exited caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

Executor 2

```
18/06/13 16:55:35 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 ERROR YarnClusterScheduler: Lost executor 2 on
 ip-10-1-2-16.ec2.internal: Container killed by YARN for exceeding
```

```
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 WARN TaskSetManager: Lost task 0.1 in stage 0.0 (TID 1,
 ip-10-1-2-16.ec2.internal, executor 2): ExecutorLostFailure (executor 2 exited
 caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

Executor 3

```
18/06/13 16:56:37 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 ERROR YarnClusterScheduler: Lost executor 3 on
 ip-10-1-2-189.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 WARN TaskSetManager: Lost task 0.2 in stage 0.0 (TID 2,
 ip-10-1-2-189.ec2.internal, executor 3): ExecutorLostFailure (executor 3
 exited caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

Executor 4

```
18/06/13 16:57:18 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 ERROR YarnClusterScheduler: Lost executor 4 on
 ip-10-1-2-96.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 WARN TaskSetManager: Lost task 0.3 in stage 0.0 (TID 3,
 ip-10-1-2-96.ec2.internal, executor 4): ExecutorLostFailure (executor 4 exited
 caused by one of the running tasks) Reason: Container killed by YARN for
```

```
exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
spark.yarn.executor.memoryOverhead.
```

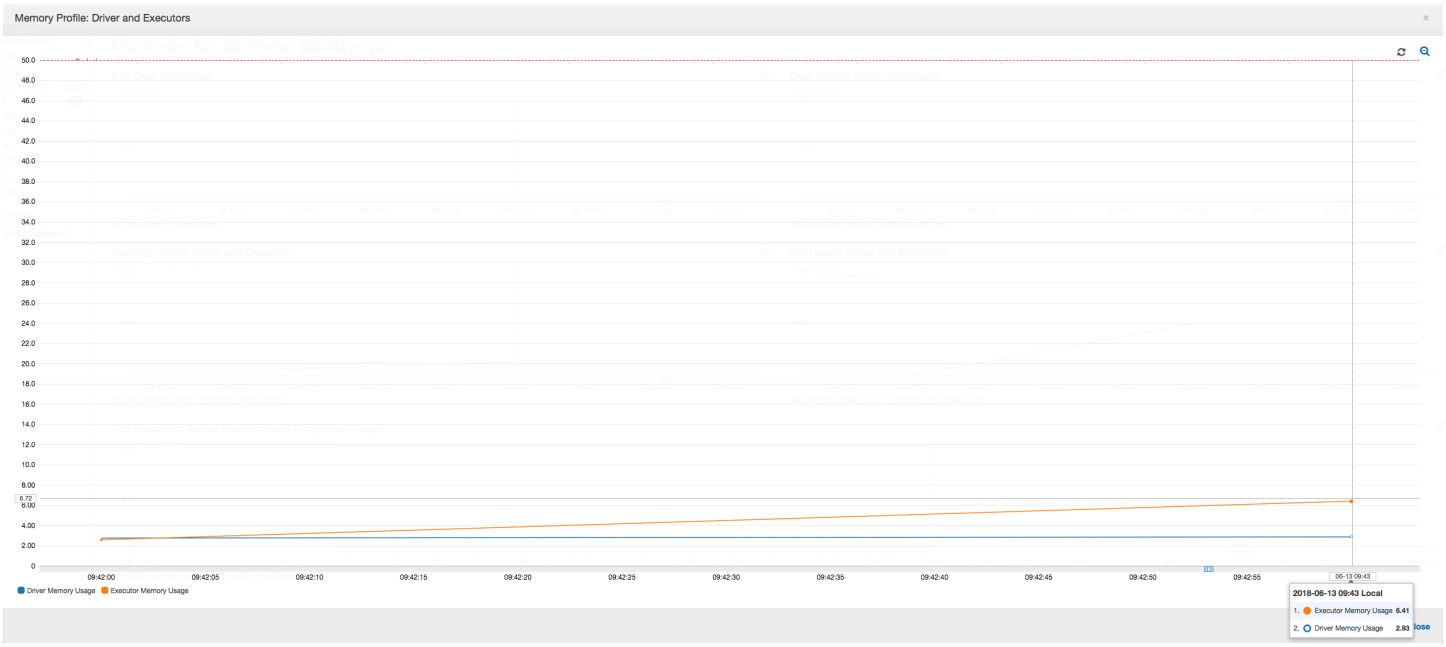
Correzione dell'impostazione relativa alle dimensioni di recupero tramite frame dinamici AWS Glue

L'executor ha esaurito la memoria durante la lettura della tabella JDBC perché la configurazione predefinita per le dimensioni di recupero JDBC Spark è zero. Ciò significa che il driver JDBC nell'executor Spark cerca di recuperare i 34 milioni di righe del database contemporaneamente e di eseguirne la memorizzazione nella cache, anche se il flusso di Spark avviene una riga per volta. Con Spark, è possibile evitare questo scenario impostando il parametro relativo alle dimensioni di recupero su un valore predefinito diverso da zero.

È inoltre possibile risolvere questo errore usando frame dinamici AWS Glue. Per impostazione predefinita, i frame dinamici utilizzano una dimensione di recupero di 1.000 righe che in genere è un valore sufficiente. Di conseguenza, l'executor non usa più del 7% della memoria totale. Il processo AWS Glue termina in meno di due minuti con solo un singolo executor. Sebbene l'uso dei frame dinamici AWS Glue sia l'approccio consigliato, è anche possibile impostare la dimensione di recupero utilizzando la proprietà `fetchsize` di Apache Spark. Consulta la guida a Spark SQL, DataFrames e set di dati.

```
val (url, database, tableName) = {
 ("jdbc_url", "db_name", "table_name")
 }
val source = glueContext.getSource(format, sourceJson)
val df = source.getDynamicFrame
glueContext.write_dynamic_frame.from_options(frame = df, connection_type = "s3",
 connection_options = {"path": output_path}, format = "parquet", transformation_ctx =
 "datasink")
```

Parametri profilati normali: la memoria dell'executor con frame dinamici AWS Glue non supera mai la soglia di sicurezza, come illustrato nell'immagine seguente. Il flusso passa nelle righe dal database e vengono memorizzate nella cache solo 1.000 righe nel driver JDBC in un determinato momento. Non si è verificata un'eccezione di memoria esaurita.



Debug di fasi impegnative e attività in ritardo

È possibile usare la profilatura dei processi AWS Glue per identificare le fasi impegnative e le attività in ritardo nei processi ETL (Extract, Transform and Load, estrazione, trasformazione e caricamento). Un'attività in ritardo richiede molto più tempo rispetto al resto delle attività in una fase di un processo AWS Glue. Di conseguenza, il completamento della fase richiede più tempo e aumenta il tempo totale di esecuzione del processo.

Unione di file di input di piccole dimensioni in file di output di dimensioni maggiori

Un'attività in ritardo può verificarsi in caso di distribuzione non uniforme del lavoro tra attività diverse oppure in caso di un'asimmetria dei dati a causa della quale un'attività elabora più dati.

È possibile profilare il codice seguente, che rappresenta un modello comune in Apache Spark, per unire un numero elevato di piccoli file in file di output di dimensioni maggiori. Per questo esempio, il set di dati di input è costituito da 32 GB di file compressi Gzip JSON. Il set di dati di output include circa 190 GB di file JSON non compressi.

Il codice profilato è il seguente:

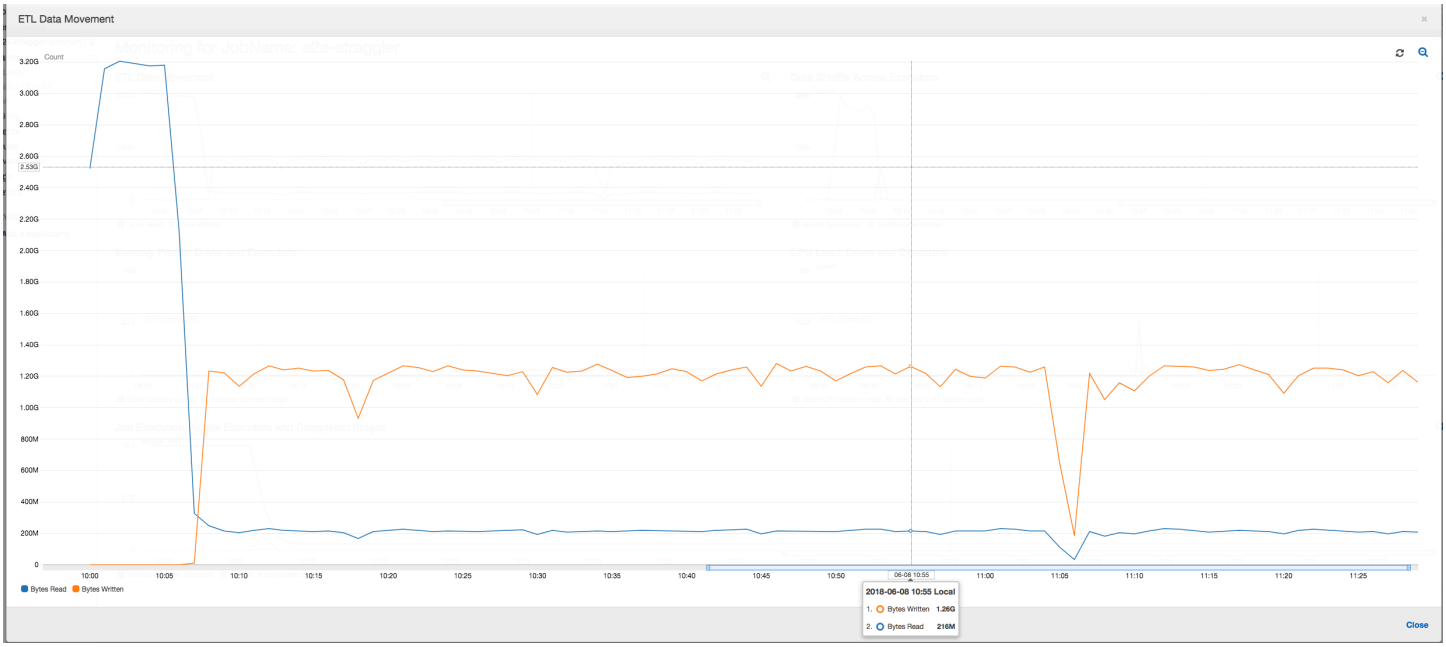
```
datasource0 = spark.read.format("json").load("s3://input_path")
df = datasource0.coalesce(1)
df.write.format("json").save(output_path)
```

Visualizzazione dei parametri profilati nella console AWS Glue

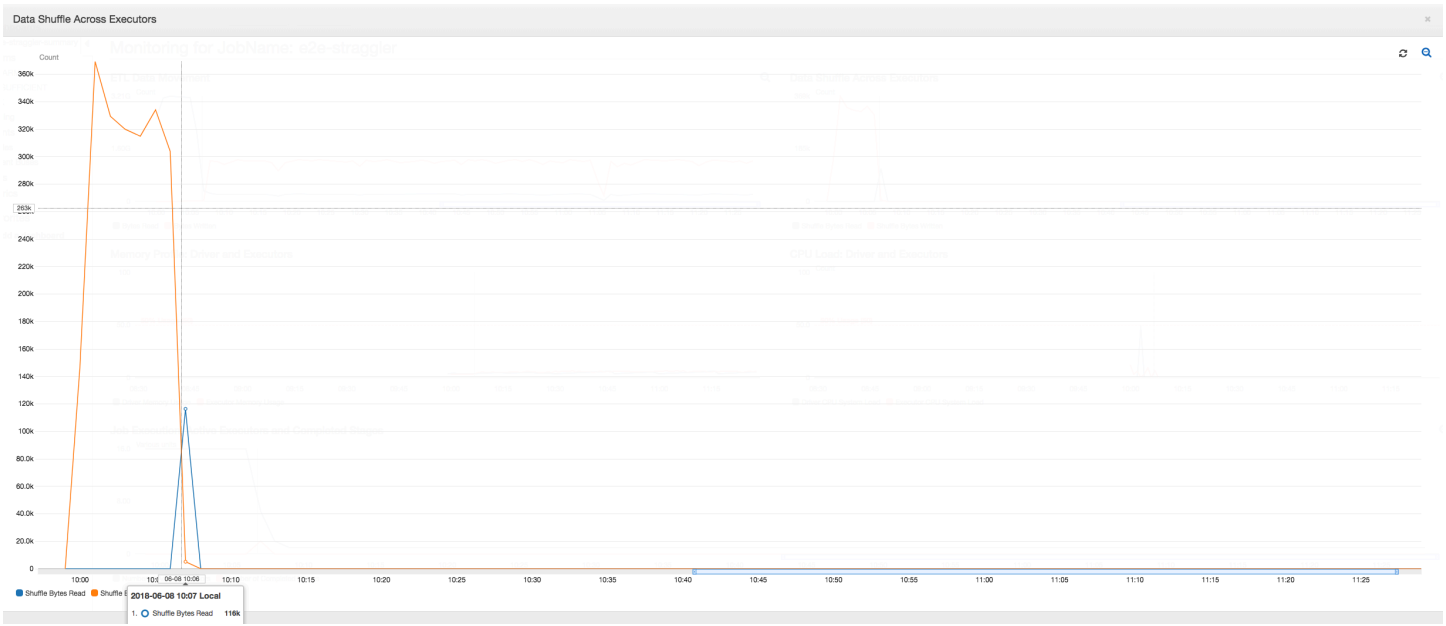
È possibile profilare il processo per esaminare quattro diversi set di parametri:

- Spostamento di dati ETL
- Distribuzione casuale dei dati tra executor
- Esecuzione del processo
- Profilo di memoria

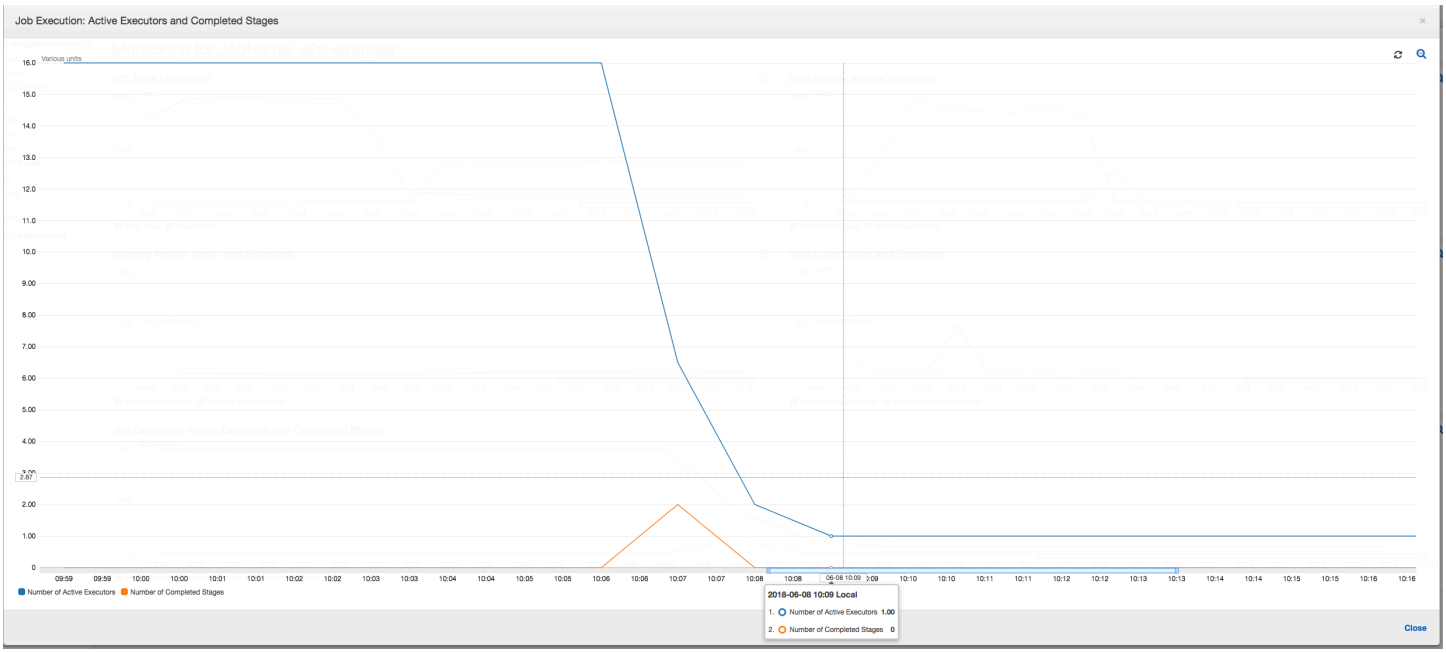
Spostamento di dati ETL: nel profilo ETL Data Movement (Spostamento di dati ETL) i byte vengono letti abbastanza rapidamente da tutti gli executor nella prima fase, che viene completata entro i primi sei minuti. Tuttavia, il tempo di esecuzione totale del processo è di circa un'ora, principalmente a causa delle operazioni di scrittura dei dati.



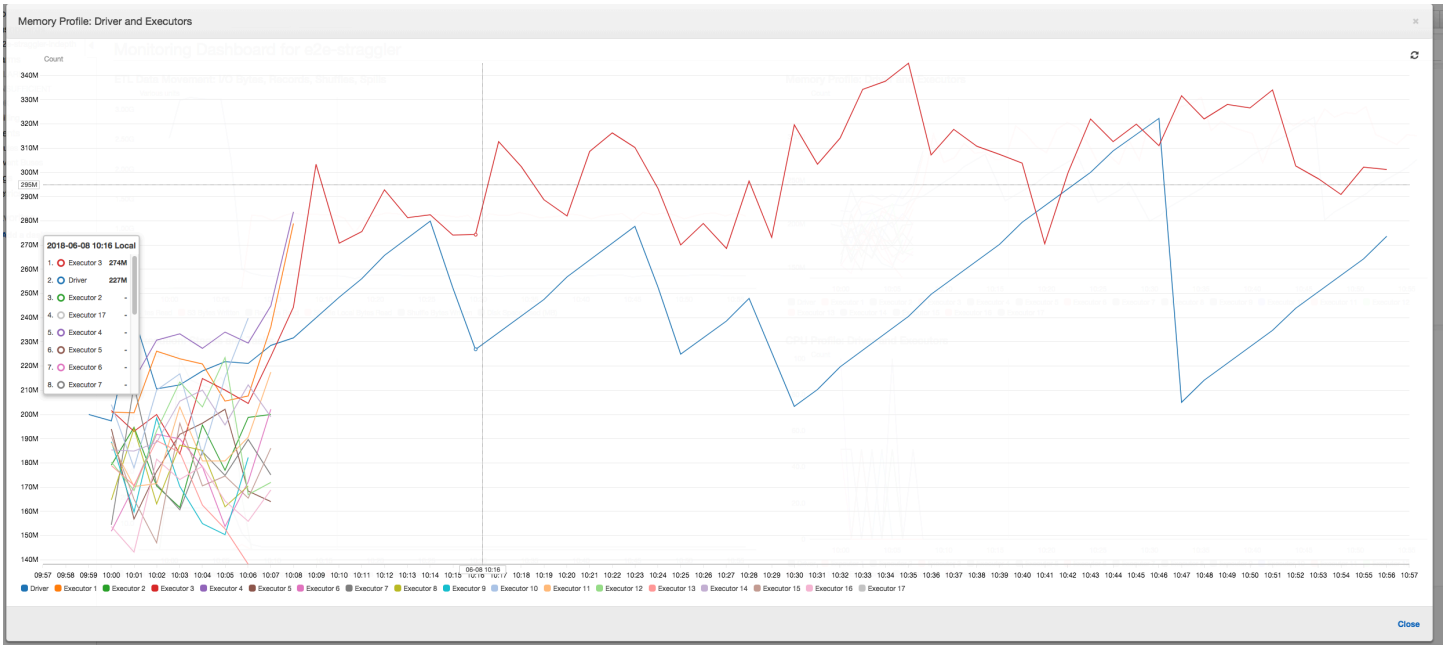
Distribuzione casuale dei dati tra executor: il numero di byte letti e scritti durante la distribuzione casuale mostra un picco prima del completamento della fase 2, come indicato dai parametri Job Execution (Esecuzione processo) e Data Shuffle (Distribuzione casuale dei dati). Dopo che i dati sono stati distribuiti in modo causale da tutti gli executor, le operazioni di lettura e scrittura procedono solo dall'executor numero 3.



Esecuzione del processo: come illustrato nel grafico seguente, tutti gli altri executor sono inattivi e vengono infine rilasciati entro le 10:09. A questo punto, il numero totale di executor scende a uno solo. Ciò mostra chiaramente che l'executor numero 3 è costituito dall'attività in ritardo che sta impiegando il tempo di esecuzione più lungo e che contribuisce in misura maggiore al tempo di esecuzione del processo.



Profilo di memoria: dopo le prime due fasi, solo l'executor numero 3 consuma attivamente memoria per elaborare i dati. Gli altri executor sono semplicemente inattivi o sono stati rilasciati poco dopo il completamento delle prime due fasi.



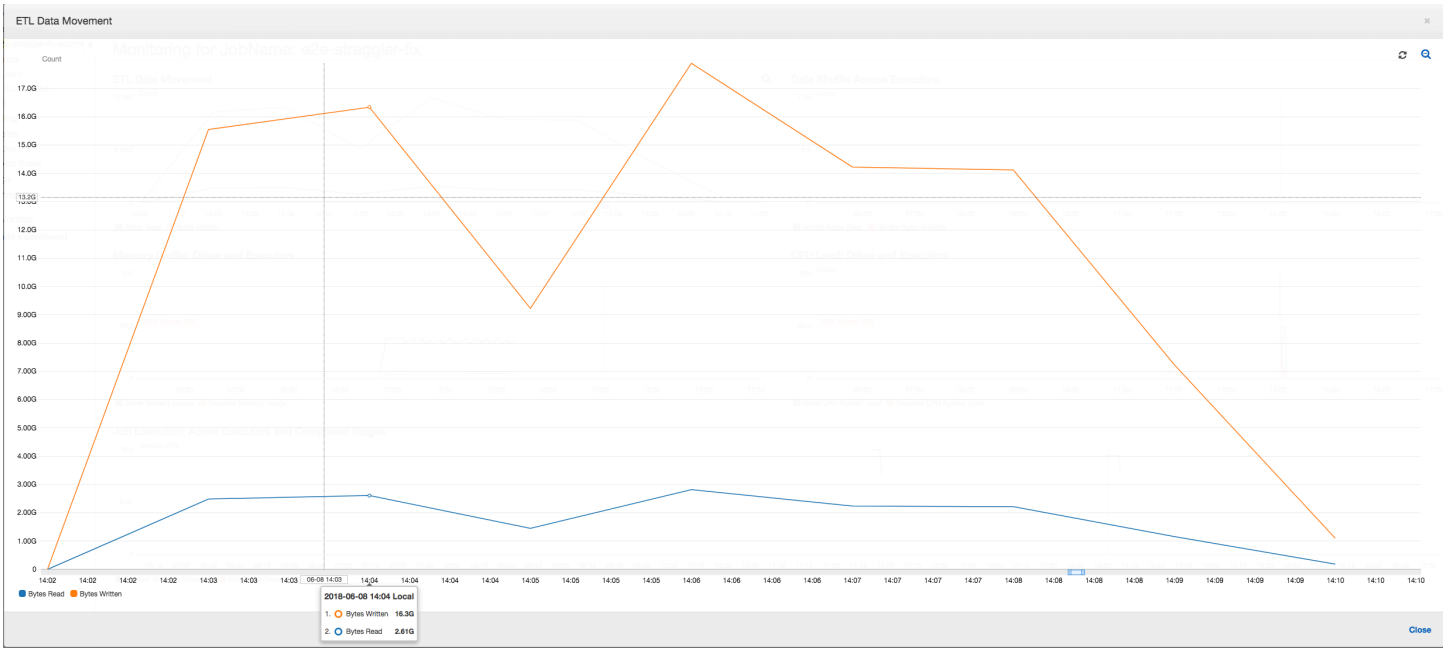
Correzione del calo degli executor tramite il raggruppamento

È possibile evitare il calo degli executor usando la caratteristica di raggruppamento in AWS Glue. Usa il raggruppamento per distribuire i dati in modo uniforme tra tutti gli executor e unire i file in file di dimensioni maggiori usando tutti gli executor disponibili nel cluster. Per ulteriori informazioni, consulta Lettura di file di input in gruppi di grandi dimensioni.

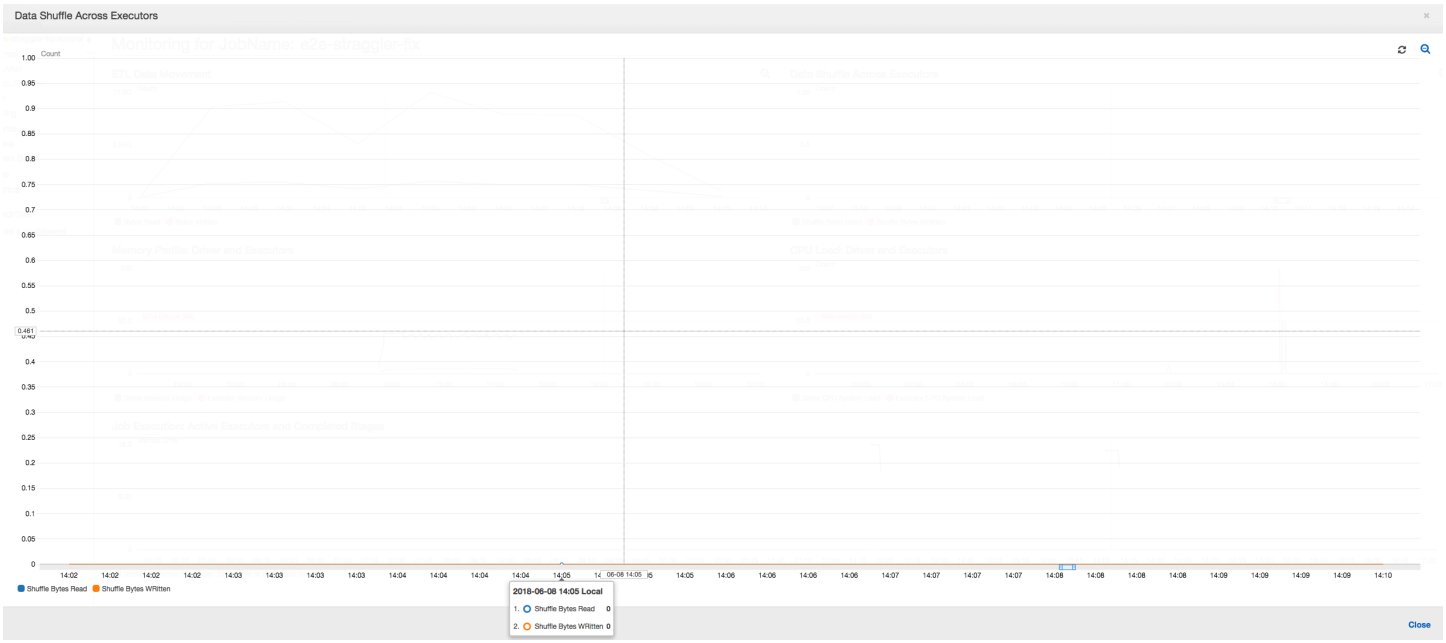
Per controllare gli spostamenti di dati ETL nel processo AWS Glue, profila il codice seguente con il raggruppamento abilitato:

```
df = glueContext.create_dynamic_frame_from_options("s3", {'paths': ["s3://input_path"],
 "recurse":True, 'groupFiles': 'inPartition'}, format="json")
datasink = glueContext.write_dynamic_frame.from_options(frame = df, connection_type =
 "s3", connection_options = {"path": output_path}, format = "json", transformation_ctx
 = "datasink4")
```

Spostamento di dati ETL: le operazioni di scrittura dei dati vengono ora trasmesse in parallelo alle operazioni di lettura dei dati per tutto il tempo di esecuzione del processo. Di conseguenza, il processo viene completato entro otto minuti, molto più rapidamente che in precedenza.

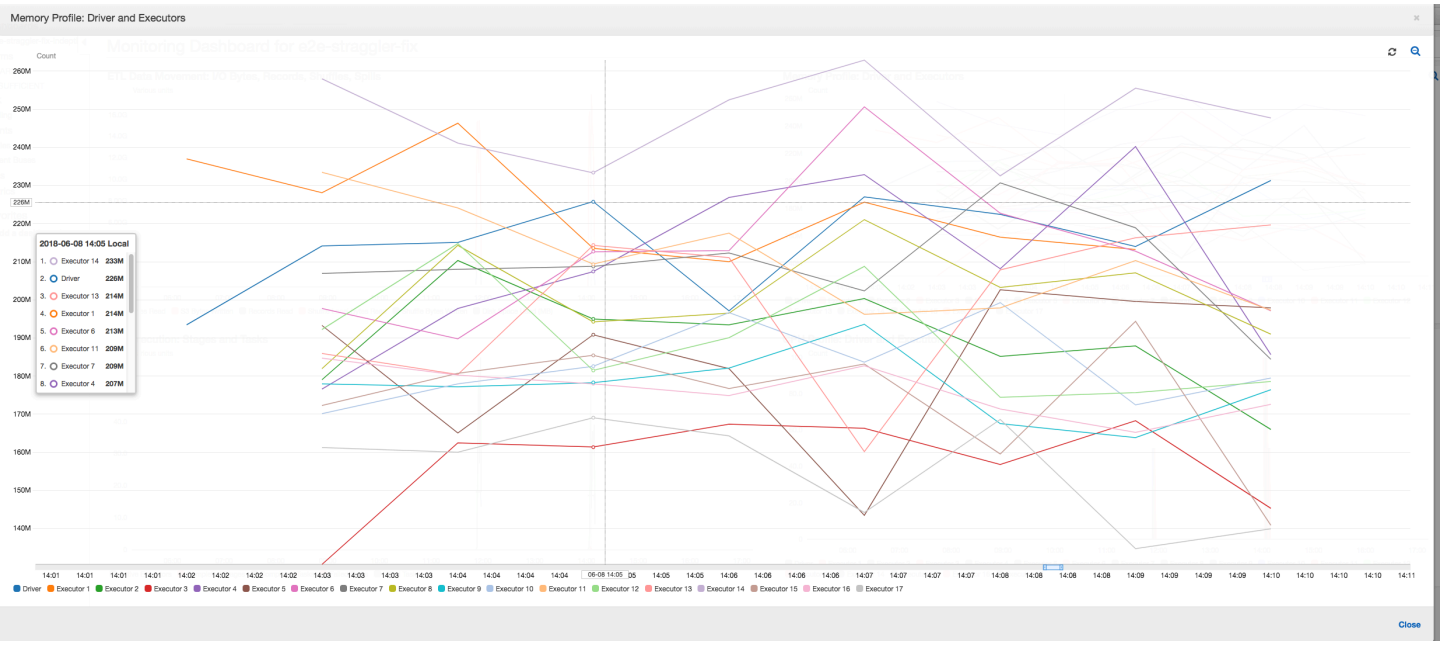


Distribuzione casuale dei dati tra executor: poiché i file di input vengono uniti durante le operazioni di lettura usando la caratteristica di raggruppamento, non vengono eseguite costose attività di distribuzione casuale dei dati dopo le operazioni di lettura dei dati.



Esecuzione del processo: i parametri relativi all'esecuzione del processo mostrano che il numero totale di executor attivi in esecuzione e che elaborano i dati rimane relativamente costante. Non vi sono singole attività in ritardo nel processo. Tutti gli executor sono attivi e non vengono rilasciati fino al completamento del processo. Poiché non vi è alcuna operazione intermedia di distribuzione casuale dei dati tra gli executor, il processo è costituito da un'unica fase.

Profilo di memoria: i parametri mostrano il consumo di memoria attivo tra tutti gli executor, riconfermando la presenza di attività in tutti gli executor. Poiché i dati vengono trasmessi e scritti in parallelo, l'utilizzo totale di memoria di tutti gli executor è piuttosto uniforme e ben al di sotto della soglia di sicurezza per tutti gli executor.



Monitoraggio dell'avanzamento di processi multipli

Puoi profilare più processi AWS Glue contemporaneamente e monitorare il flusso di dati tra loro. Si tratta di un modello di flusso di lavoro comune e richiede il monitoraggio per l'avanzamento del

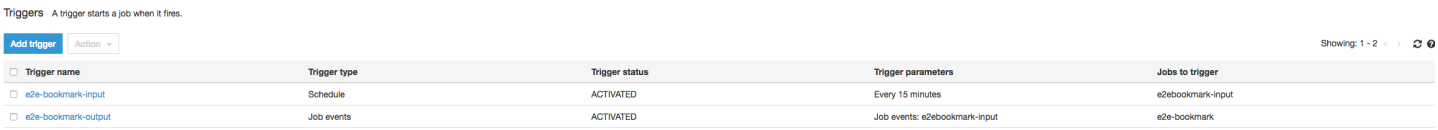
processo individuale, il backlog dell'elaborazione dei dati, la rielaborazione dei dati e i segnalibri dei processi.

Argomenti

- Codice profilato
- Visualizzazione dei parametri profilati nella console AWS Glue
- Correzione dell'elaborazione dei file

Codice profilato

In questo flusso di lavoro, ci sono due processi: un processo di input e uno di output. Il processo di input è pianificato per l'esecuzione ogni 30 minuti usando un trigger periodico. Il processo di output è pianificato per l'esecuzione dopo ogni esecuzione del processo di input. Questi processi pianificati sono controllati usando trigger dei processi.



Processo di input: questo processo legge i dati da una posizione Amazon Simple Storage Service (Amazon S3), li trasforma tramite `ApplyMapping` e li scrive in una posizione Amazon S3 di staging. Il codice seguente è il codice profilato per il processo di input:

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": ["s3://input_path"],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [map_spec])
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": staging_path, "compression":
 "gzip"}, format = "json")
```

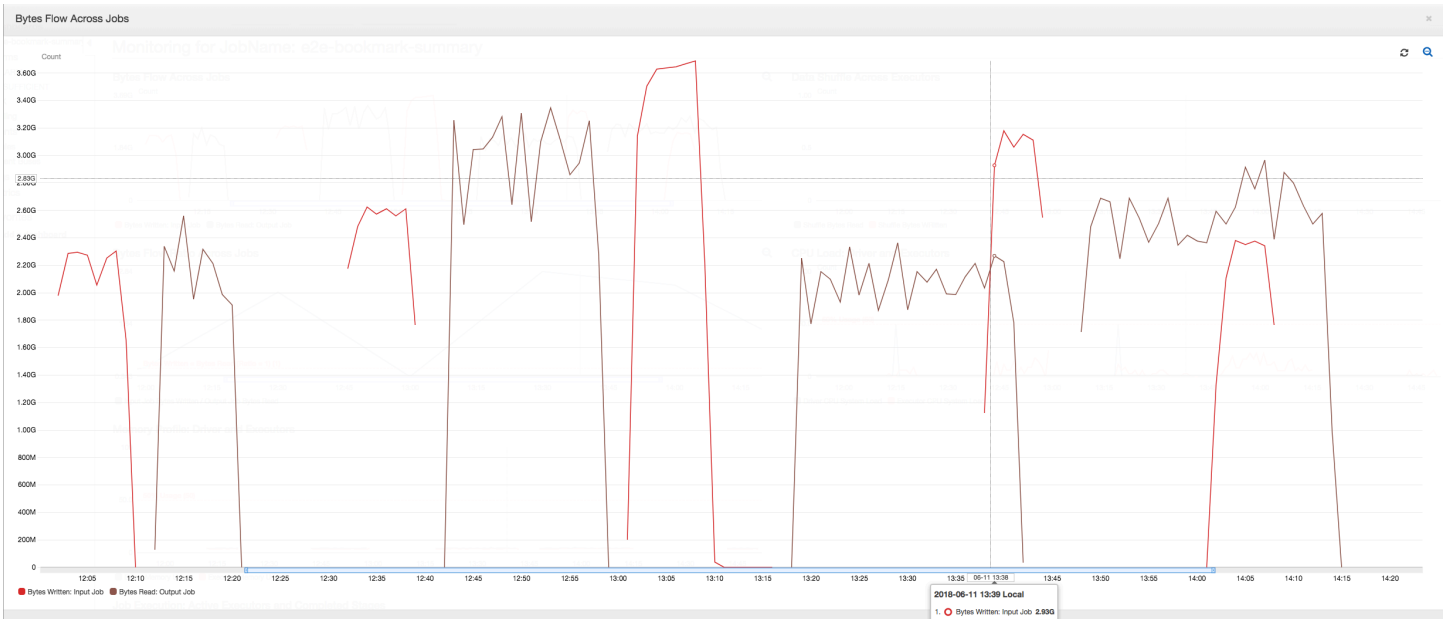
Processo di output: questo processo legge l'output del processo di input dalla posizione di staging in Amazon S3, lo trasforma nuovamente e lo scrive in una destinazione:

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [staging_path],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [map_spec])
```

```
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": output_path}, format = "json")
```

Visualizzazione dei parametri profilati nella console AWS Glue

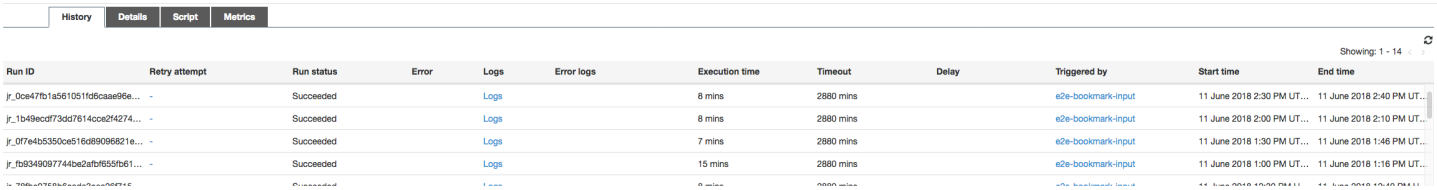
Il pannello di controllo seguente sovrappone il parametro Amazon S3 dei byte scritti dal processo di input al parametro Amazon S3 dei byte letti nella stessa sequenza temporale del processo di output. La sequenza temporale mostra diverse esecuzioni dei processi di input e di output. Il processo di input (mostrato in rosso) inizia ogni 30 minuti. Il processo di output (mostrato in marrone) viene avviato al completamento del processo di input, con una simultaneità massima di 1.



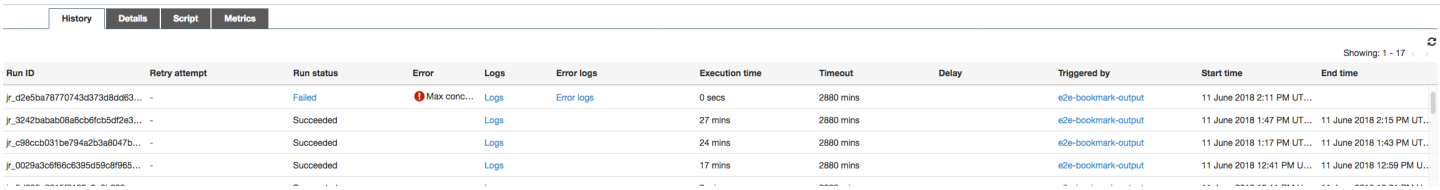
In questo esempio, i segnalibri dei processi non sono abilitati. Non vengono usati contesti di trasformazione per abilitare i segnalibri dei processi nel codice dello script.

Cronologia dei processi: i processi di input e di output hanno più esecuzioni, come illustrato nella scheda History (Cronologia), a partire dalle 12:00.

Il processo di input nella console AWS Glue ha il seguente aspetto:

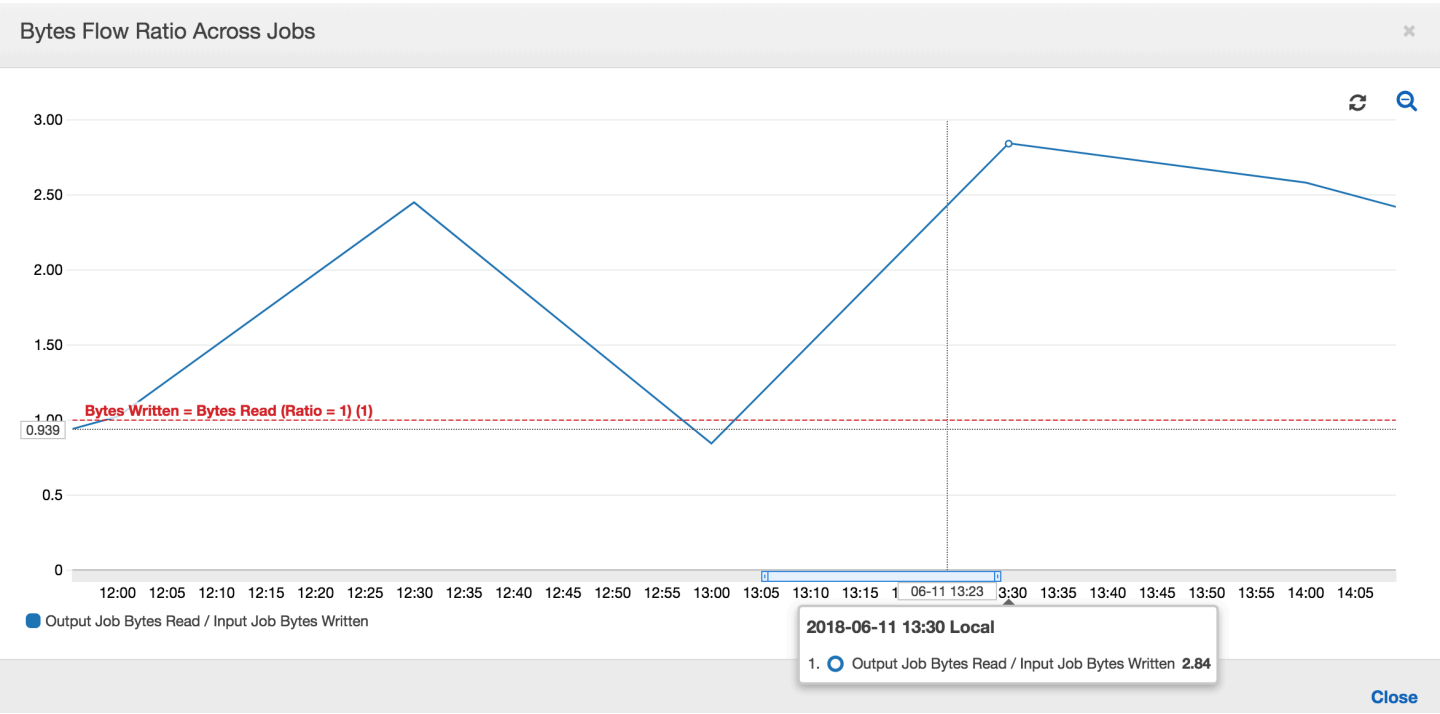


L'immagine seguente mostra il processo di output:



Prime esecuzioni dei processi: come illustrato nel grafico seguente dei byte di dati letti e scritti, le prime esecuzioni dei processi di input e di output tra le 12:00 e 12:30 mostrano pressappoco la stessa area sotto le curve. Tali aree rappresentano i byte Amazon S3 scritti dal processo di input e i byte Amazon S3 letti dal processo di output. Questi dati vengono inoltre confermati dal rapporto di byte Amazon S3 scritti (sommati nell'arco di 30 minuti, la frequenza del trigger dei processi per il processo di input). Il punto dati del rapporto dell'esecuzione del processo di input iniziato alle 12:00 è 1.

Il grafico seguente mostra il rapporto del flusso di dati in tutte le esecuzioni dei processi:



Seconde esecuzioni dei processi: nella seconda esecuzione del processo, c'è una chiara differenza tra il numero di byte letti dal processo di output rispetto al numero di byte scritti dal processo di input. Confronta l'area sotto la curva tra le due esecuzioni del processo di output o confronta le aree nella seconda esecuzione dei processi di input e di output. Il rapporto tra i byte letti e scritti mostra che il processo di output ha letto 2,5 volte i dati scritti dal processo di input nel secondo intervallo di 30 minuti dalle 12:30 alle 13:00. Ciò è dovuto al fatto che il processo di output ha rielaborato l'output

della prima esecuzione del processo di input perché i segnalibri non erano abilitati. Un rapporto superiore a 1 mostra che c'è un ulteriore backlog di dati che è stato elaborato dal processo di output.

Terze esecuzioni dei processi: il processo di input è abbastanza coerente in termini di numero di byte scritti (vedi l'area sotto le curve rosse). Tuttavia, la terza esecuzione del processo di input è durata più tempo del previsto (vedi la lunga coda della curva rossa). Di conseguenza, la terza esecuzione del processo di output è iniziata tardi. La terza esecuzione del processo ha elaborato solo una parte dei dati accumulati nella posizione di staging nei rimanenti 30 minuti tra le 13:00 e le 13:30. Il rapporto del flusso di byte mostra che ha elaborato solo un valore pari a 0,83 dei dati scritti dalla terza esecuzione del processo di input (vedi il rapporto alle 13:00).

Sovrapposizione dei processi di input e di output: la quarta esecuzione del processo di input è iniziata alle 13:30 in base alla pianificazione, prima del completamento della terza esecuzione del processo di output. C'è una sovrapposizione parziale tra queste due esecuzioni del processo. Tuttavia, la terza esecuzione del processo di output acquisisce solo i file elencati nella posizione di staging di Amazon S3 al momento dell'avvio, intorno alle 13:17. Ciò corrisponde a tutti i dati di output della prima esecuzione del processo di input. Il rapporto effettivo alle 13:30 è di circa 2,75. La terza esecuzione del processo di output ha elaborato circa 2,75 volte la quantità di dati scritti dalla quarta esecuzione del processo di input dalle 13:30 alle 14:00.

Come mostrano queste immagini, il processo di output sta rielaborando i dati dalla posizione di staging di tutte le esecuzioni precedenti del processo di input. Di conseguenza, la quarta esecuzione del processo di output è la più lunga e si sovrappone all'intera quinta esecuzione del processo di input.

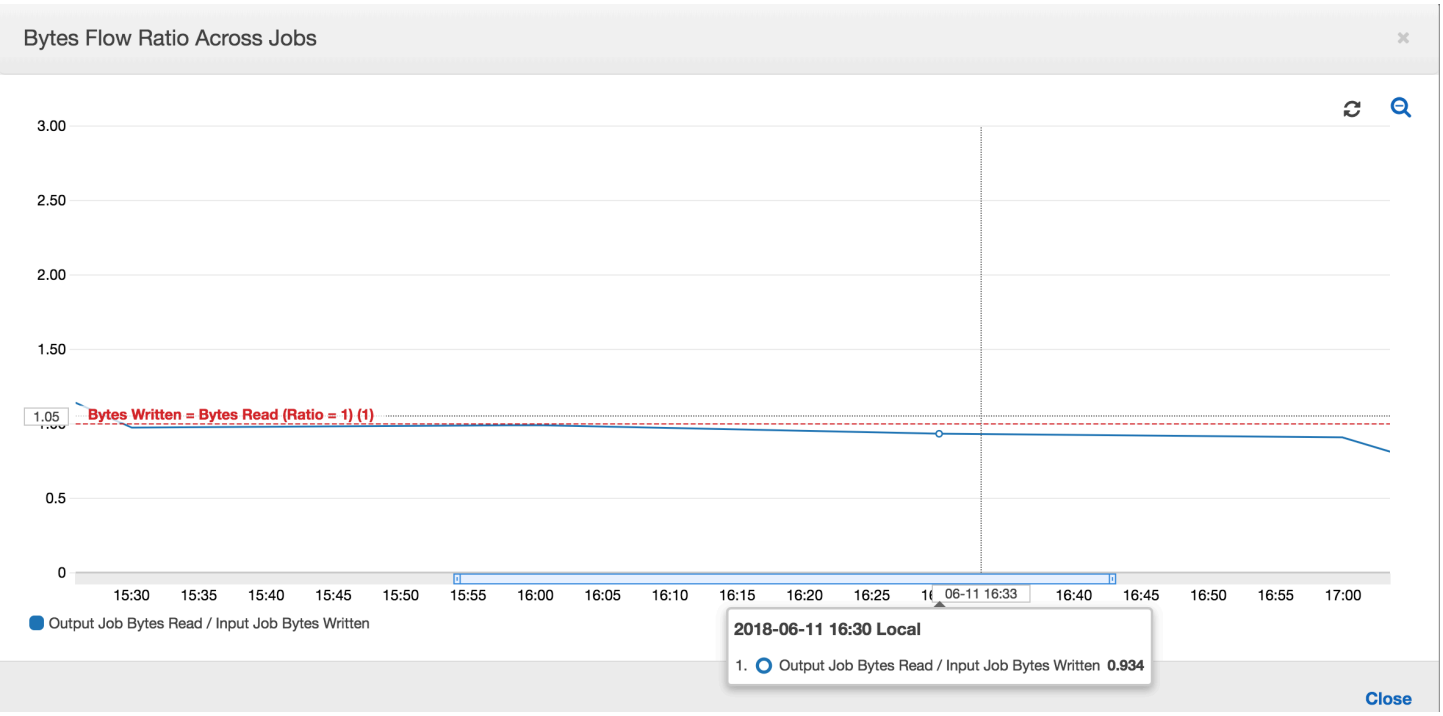
Correzione dell'elaborazione dei file

È necessario accertarsi che i processi di output elaborino solo i file che non sono stati elaborati da esecuzioni precedenti del processo di output. A tale scopo, abilita i segnalibri dei processi e imposta il contesto di trasformazione nel processo di output, come segue:

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [staging_path],
 "useS3ListImplementation":True,"recurse":True}, format="json", transformation_ctx =
 "bookmark_ctx")
```

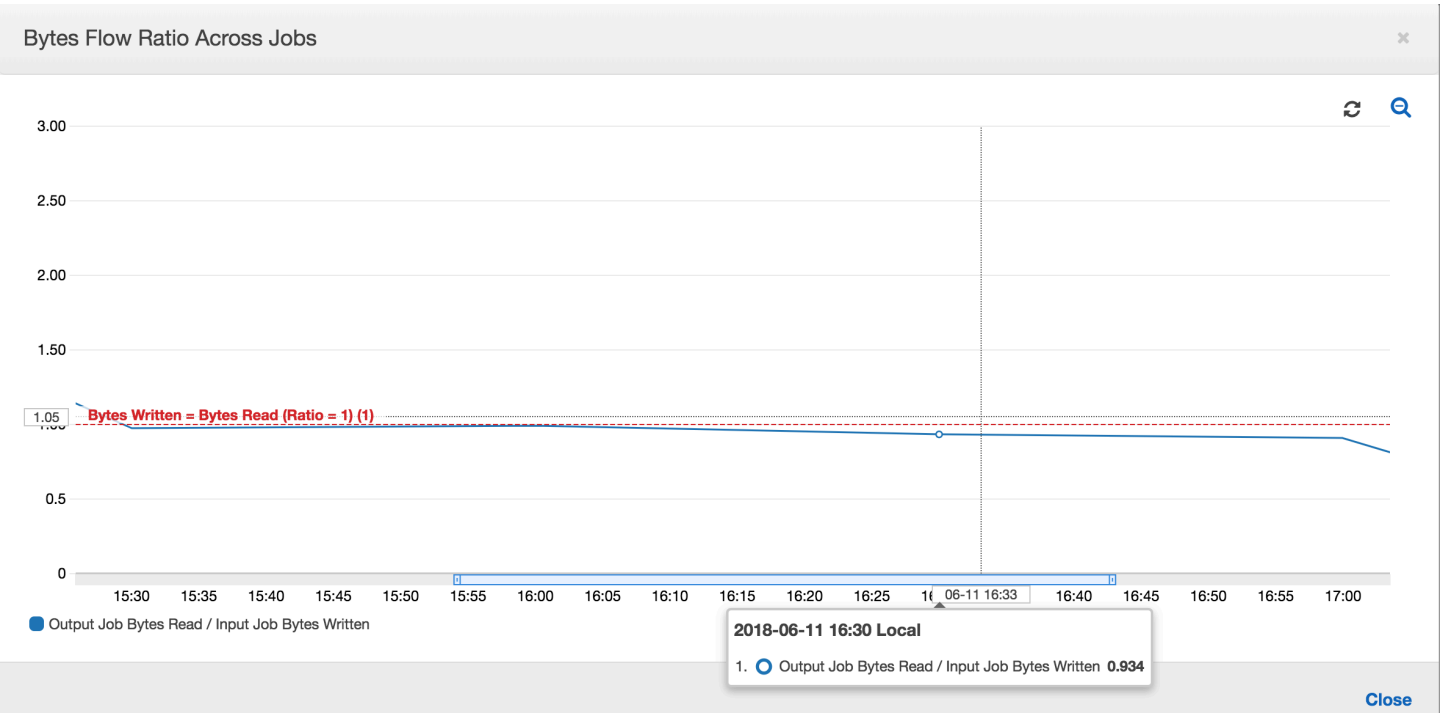
Con i segnalibri dei processi abilitati, il processo di output non rielabora i dati nella posizione di staging di tutte le precedenti esecuzioni del processo di input. Nell'immagine seguente, che mostra i dati letti e scritti, l'area sotto la curva marrone è abbastanza coerente e simile alle curve rosse.

I rapporti dei flussi di byte rimangono abbastanza vicino a 1 perché non ci sono dati aggiuntivi elaborati.



Un'esecuzione del processo di output viene avviata e acquisisce i file nella posizione di staging prima che la successiva esecuzione del processo di input inizi a inserire ulteriori dati nella posizione

di staging. Fino a quando ciò avviene, vengono elaborati solo i file acquisiti dall'esecuzione del processo di input precedente e il rapporto rimane vicino a 1.



Supponiamo che il processo di input richieda più tempo del previsto e, di conseguenza, il processo di output acquisisca i file nella posizione di staging da due esecuzioni del processo di input. Il rapporto è quindi superiore a 1 per l'esecuzione del processo di output. Tuttavia, le esecuzioni successive del processo di output non elaborano file già elaborati dalle esecuzioni precedenti del processo di output.

Monitoraggio per la pianificazione della capacità DPU

Puoi usare i parametri dei processi in AWS Glue per stimare il numero di unità di elaborazione dei dati (DPU) che è possibile usare per dimensionare un processo AWS Glue.

> **Note**
>
> Questa pagina è applicabile solo a AWS Glue versioni 0.9 e 1.0. Versioni successive di AWS Glue contengono caratteristiche di risparmio che introducono ulteriori considerazioni durante la pianificazione della capacità.

Argomenti

- Codice profilato
- Visualizzazione dei parametri profilati nella console AWS Glue

- Determinazione della capacità DPU ottimale

Codice profilato

Lo script seguente legge una partizione Amazon Simple Storage Service (Amazon S3) contenente 428 file JSON GZIP. Lo script applica una mappatura per modificare i nomi dei campi, li converte e li scrive in Amazon S3 in un formato Apache Parquet. Puoi effettuare il provisioning di 10 DPU secondo le impostazioni di default ed eseguire questo processo.

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [input_path],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [(map_spec])
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": output_path}, format =
 "parquet")
```

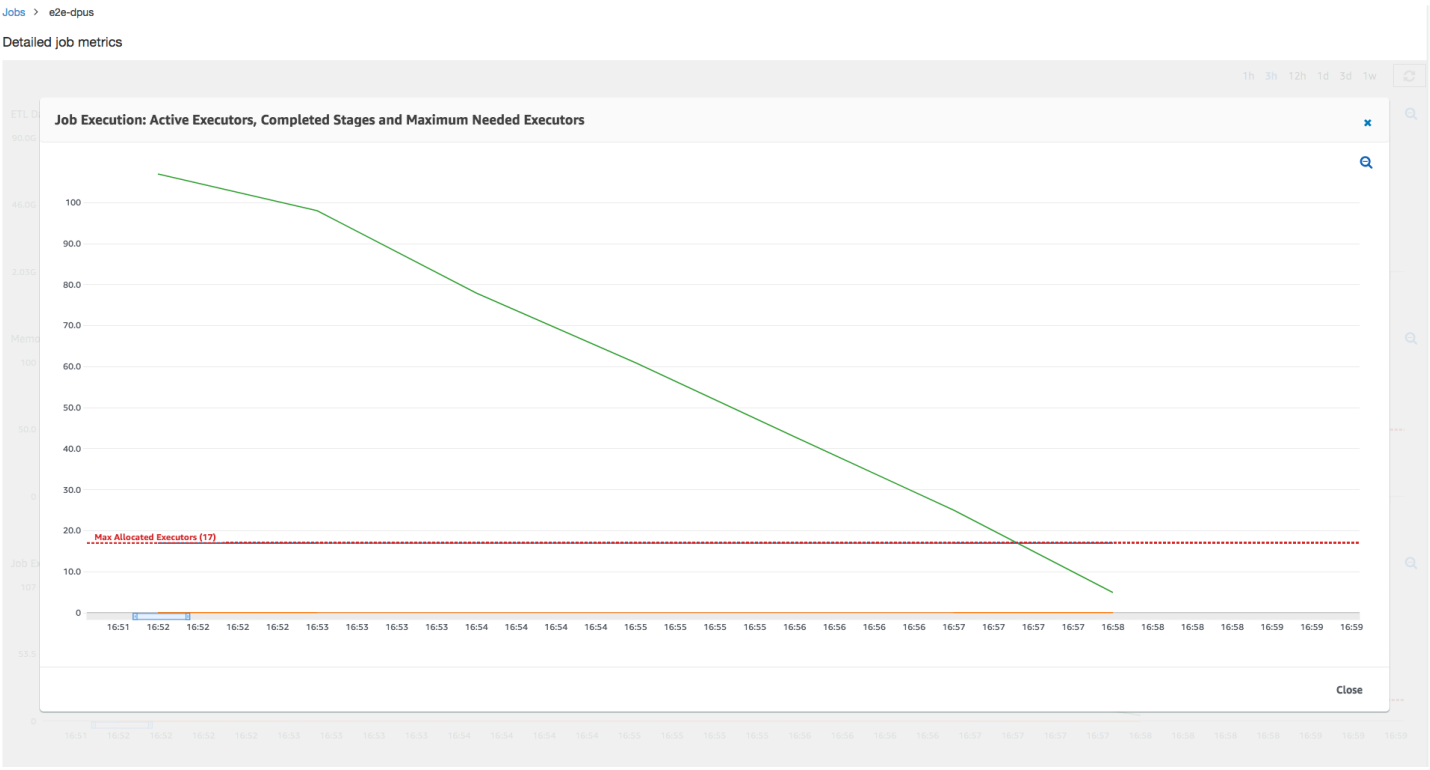
Visualizzazione dei parametri profilati nella console AWS Glue

Esecuzione processo 1: in questa esecuzione del processo esaminiamo come individuare se mancano DPU nel cluster. La funzionalità di esecuzione del processo in AWS Glue mostra il numero totale di esecutori attivi, il numero di fasi completate e il numero massimo di esecutori necessari.

Il numero massimo di executor necessari viene calcolato aggiungendo il numero totale di attività in esecuzione e attività in sospeso e dividendo per le attività per executor. Questo risultato è una misura del numero totale di executor necessari per soddisfare il carico corrente.

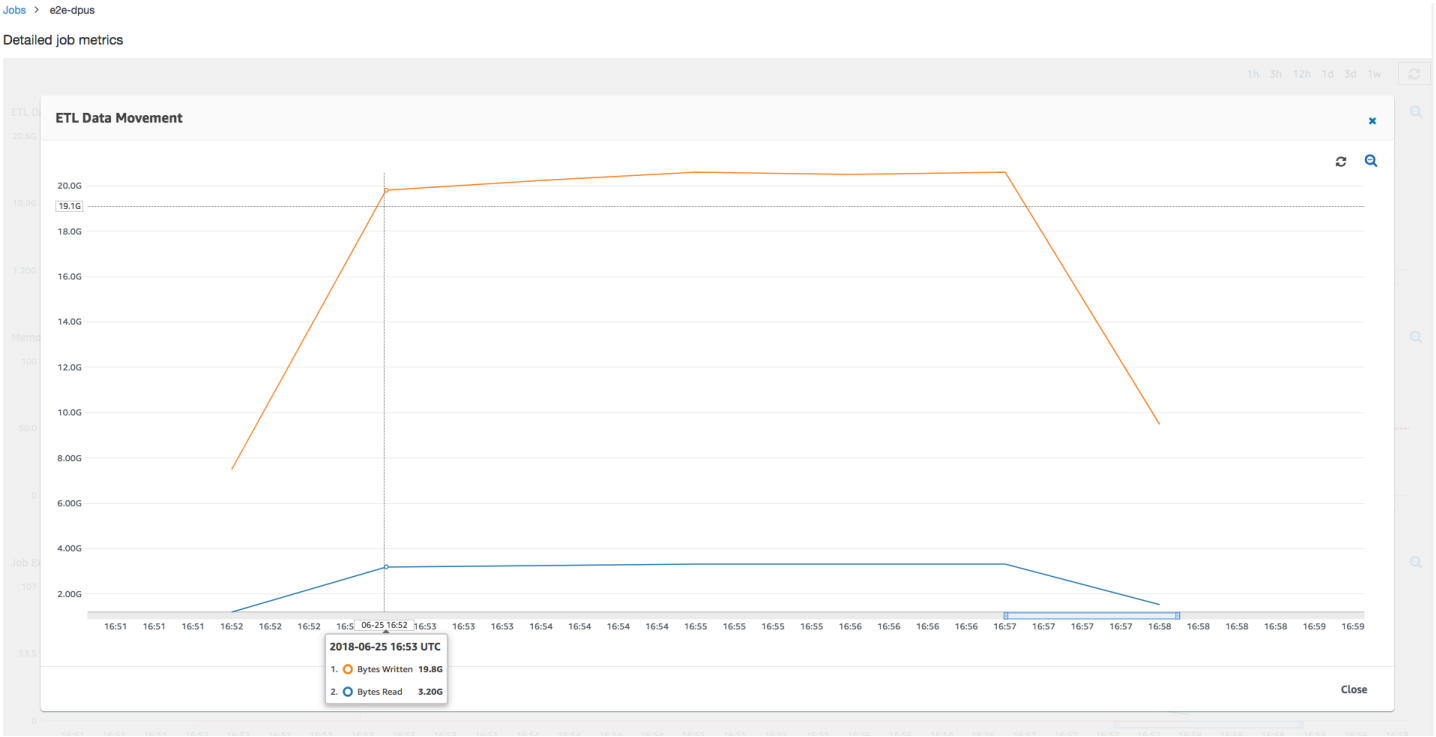
Al contrario, il numero di executor attivi misura quanti executor stanno eseguendo attivamente attività Apache Spark. Con l'avanzamento del processo, il numero massimo di executor necessari può cambiare e in genere diminuisce verso la fine del processo, in quanto le attività in coda si riducono.

La linea rossa orizzontale nel grafico seguente mostra il numero massimo di executor allocati, che dipende dal numero di unità DPU allocate al processo. In questo caso, puoi allocare 10 DPU per l'esecuzione del processo. Una DPU è riservata alla gestione. Nove DPU eseguono due executor ciascuna e un executor è riservato per il driver Spark. Il driver Spark viene eseguito all'interno dell'applicazione principale. Pertanto, il numero massimo di executor allocati è 2*9 - 1 = 17 executor.



Come mostra il grafico, il numero massimo di executor parte da 107 all'inizio del processo, mentre il numero di executor attivi rimane 17. Ciò corrisponde al numero massimo di executor allocati con 10 DPU. Il rapporto tra il numero massimo di executor necessari e il numero massimo di executor allocati (aggiungendo 1 a entrambi per il driver Spark) indica il fattore di provisioning in difetto: 108/18 = 6x. È possibile effettuare il provisioning di 6 (con il rapporto di provisioning) * 9 (capacità DPU corrente - 1) +1 DPU = 55 DPU per dimensionare il processo per eseguirlo con il massimo parallelismo e terminare più velocemente.

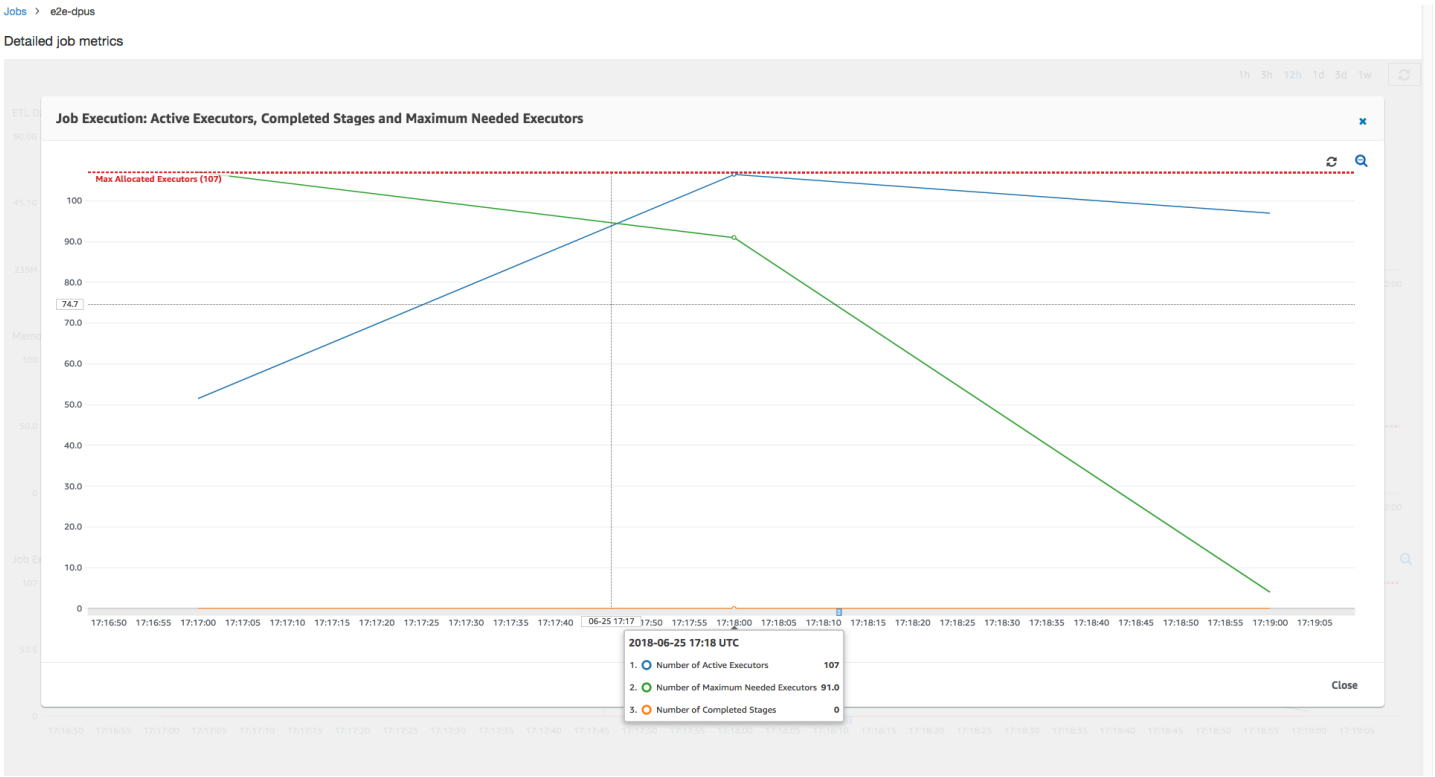
La console AWS Glue mostra i parametri dettagliati dei processi come una linea statica che rappresenta il numero massimo originale di esecutori allocati. La console calcola il numero massimo di esecutori allocati dalla definizione del processo per i parametri. Al contrario, per i parametri dettagliati di esecuzione dei processi, la console calcola il numero massimo di esecutori allocati dalla configurazione di esecuzione del processo, in particolare le DPU allocate per tale esecuzione. Per visualizzare i parametri dell'esecuzione di un singolo processo, seleziona l'esecuzione del processo e scegli View run metrics (Visualizza parametri di esecuzione).



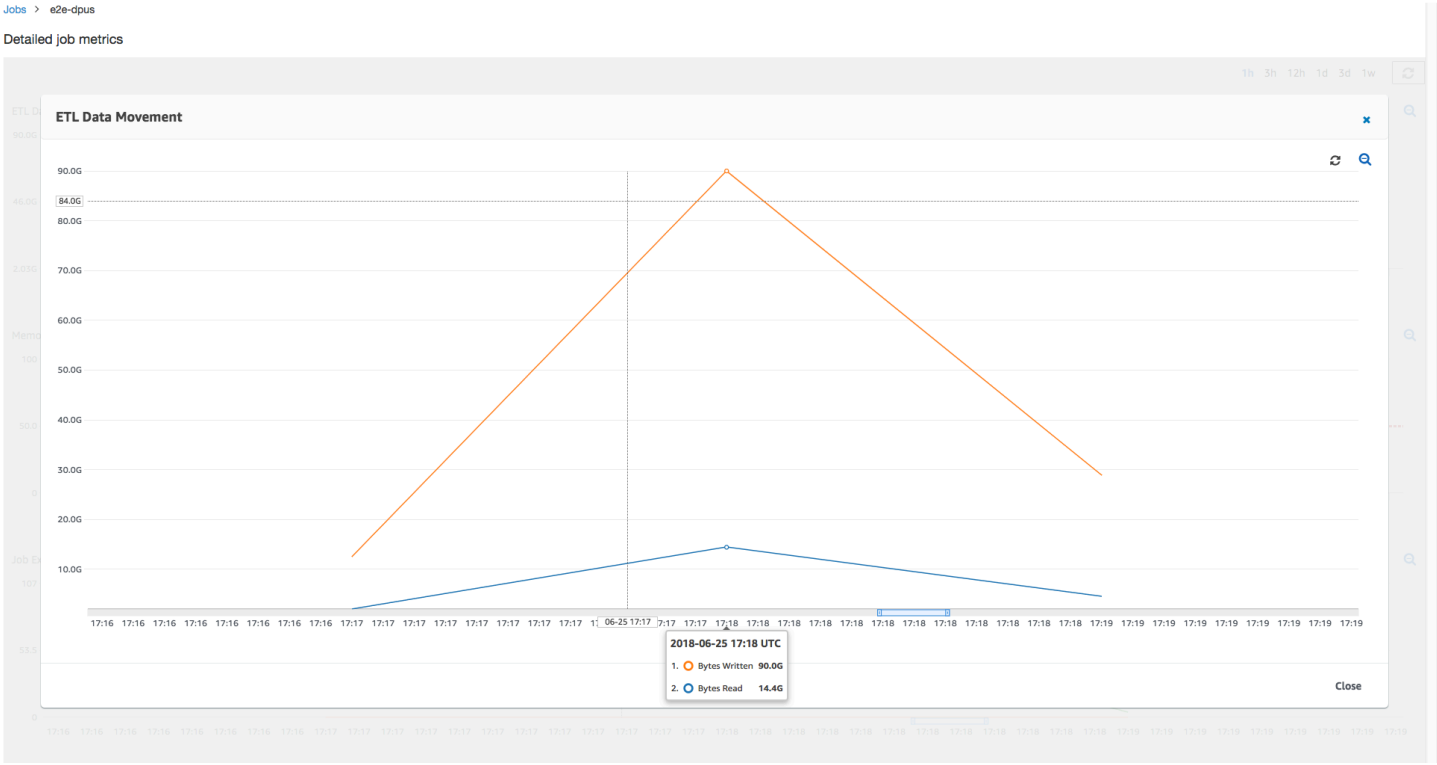
Osservando i byte Amazon S3 letti e scritti, si nota che il processo impiega tutti e sei i minuti per lo streaming in ingresso dei dati da Amazon S3 e la scrittura in uscita in parallelo. Tutti i core nelle DPU allocate leggono e scrivono in Amazon S3. Il numero massimo di executor necessari (107), corrisponde anche al numero di file nel percorso di input Amazon S3 path428. Ogni executor può avviare quattro attività Spark per elaborare quattro file di input (JSON GZIP).

Determinazione della capacità DPU ottimale

In base ai risultati dell'esecuzione del processo precedente, puoi aumentare il numero totale di DPU allocate a 55 ed esaminare le prestazioni del processo. Il processo viene completato in meno di tre minuti, ossia in metà del tempo richiesto in precedenza. Il dimensionamento del processo non è lineare in questo caso, perché si tratta di un processo a esecuzione breve. I processi con attività di lunga durata o con un numero elevato di attività (un numero elevato di executor necessari massimi) traggono vantaggio da un aumento delle prestazioni con un aumento delle DPU il più lineare possibile.



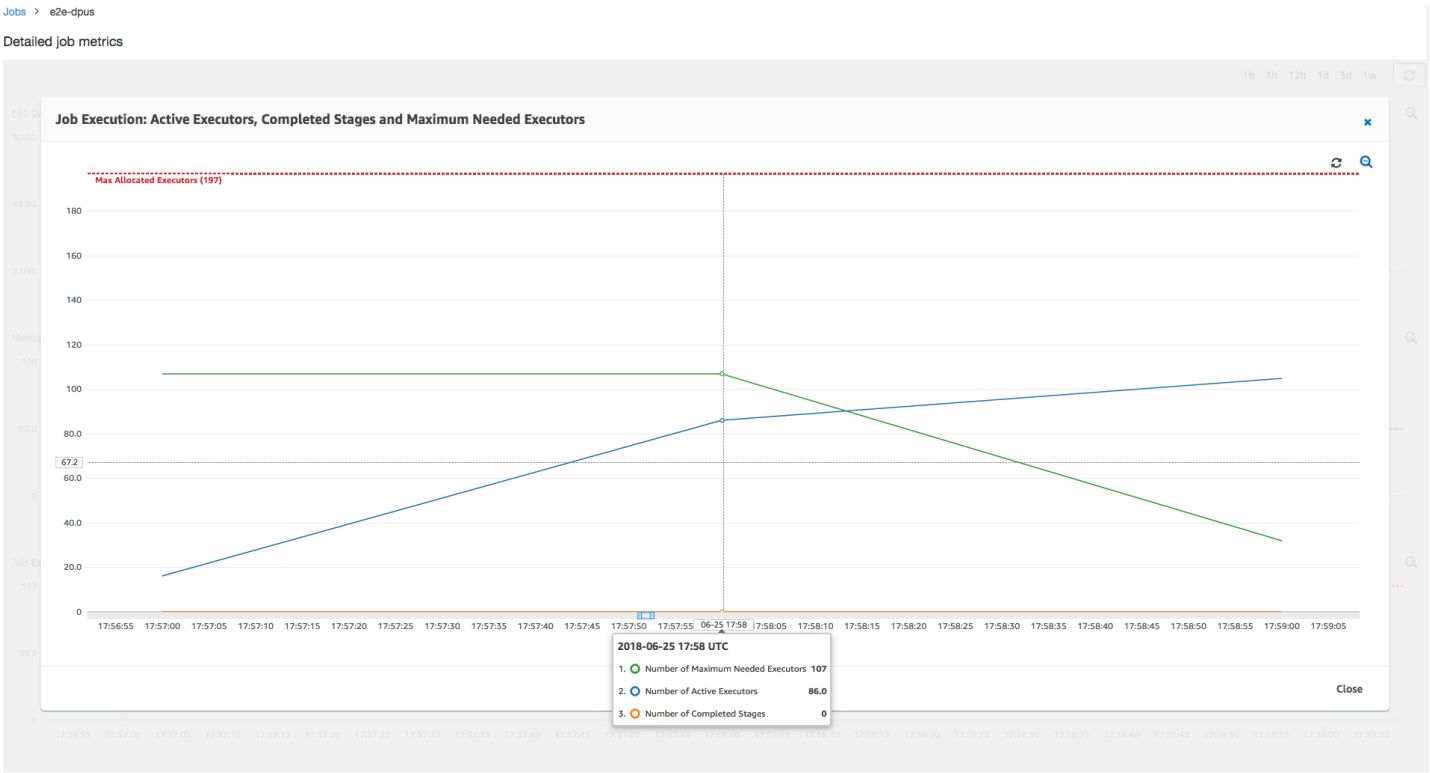
Come mostra l'immagine seguente, il numero totale di executor attivi raggiunge il numero massimo di executor allocati (107). Analogamente, il numero massimo di executor necessari non supera mai il numero massimo di executor allocati. Il numero massimo di executor necessari viene calcolato dai conteggi di attività in esecuzione e in attesa, quindi potrebbe essere inferiore al numero di executor attivi. Questo perché non ci possono essere executor che sono parzialmente o completamente inattivi per un breve periodo di tempo e non sono ancora stati rimossi.



Questa esecuzione del processo usa una quantità di executor sei volte maggiore per leggere e scrivere da Amazon S3 in parallelo. Di conseguenza, questa esecuzione del processo usa più larghezza di banda Amazon S3 per le operazioni di scrittura e lettura e termina più velocemente.

Identificazione delle DPU per le quali è stato effettuato un provisioning in eccesso

Puoi quindi determinare se il dimensionamento del processo con l'aumento a 100 DPU (99 * 2 = 198 executor) permette ulteriore scalabilità. Come mostra il grafico seguente, il processo richiede ancora tre minuti per giungere al termine. Analogamente, il processo non viene dimensionato oltre i 107 executor (configurazione di 55 DPU) e i rimanenti 91 executor sono in eccesso e non vengono usati. Questo mostra che l'aumento del numero di DPU non sempre migliora le prestazioni, come dimostrato dal numero massimo di executor necessari.



Confronto tra differenze di tempo

Le tre esecuzioni del processo illustrate nella tabella seguente riepilogano i tempi di esecuzione del processo per 10 DPU, 55 DPU e 100 DPU. Puoi individuare la capacità DPU per migliorare il tempo di esecuzione del processo usando le stime definite monitorando la prima esecuzione del processo.

Job ID	Numero di DPU	Ora di esecuzione
jr_c894524c8ef5048a4d9...	10	6 min.
jr_1a466cf2575e7ffe6856...	55	3 min.
jr_34fa1ed4c6aa9ff0a814...	100	3 min.

Aggiunta di processi di streaming ETL in AWS Glue

È possibile creare operazioni in streaming di estrazione, trasformazione e caricamento (ETL) che vengono eseguite continuamente, consumano dati da origini di streaming come Amazon Kinesis Data Streams, Apache Kafka e Amazon Managed Streaming for Apache Kafka (Amazon MSK). I processi puliscono e trasformano i dati, quindi caricano i risultati in data lake Amazon S3 o datastore JDBC.

Inoltre, è possibile produrre dati per i flussi di dati Amazon Kinesis. Questa funzionalità è disponibile solo per la scrittura di AWS Glue script. Per ulteriori informazioni, consulta the section called "Connessioni Kinesis".

Per impostazione predefinita, AWS Glue elabora e scrive i dati in finestre di 100 secondi. Ciò consente di elaborare i dati in modo efficiente e di eseguire aggregazioni su dati che arrivano più tardi del previsto. È possibile modificare questa dimensione della finestra per aumentare la tempestività o la precisione dell'aggregazione. AWS Glue I processi di streaming utilizzano i checkpoint anziché i segnalibri di lavoro per tenere traccia dei dati letti.

> **Note**
>
> AWS Glue fatture orarie per lo streaming delle operazioni ETL mentre sono in esecuzione.

Questo video illustra le problematiche relative ai costi dello streaming ETL e le funzionalità di riduzione dei costi di. AWS Glue

La creazione di un processo di streaming ETL prevede i seguenti passaggi:

1. Per un'origine di streaming Apache Kafka, creare una connessione AWS Glue alla sorgente Kafka o al cluster Amazon MSK.
2. Creare manualmente un catalogo dati per l'origine di streaming.
3. Creare un processo ETL per l'origine dati di streaming. Definire le proprietà del processo specifiche dello streaming e fornire uno script personalizzato o, facoltativamente, modificare lo script generato.

Per ulteriori informazioni, consulta Streaming ETL in AWS Glue.

Quando si crea un processo ETL di streaming per Amazon Kinesis Data Streams, non è necessario creare una connessione AWS Glue. Tuttavia, se è presente una connessione collegata al flusso di lavoro ETL AWS Glue con Kinesis Data Streams come origine, quindi è necessario un endpoint cloud privato virtuale (VPC) a Kinesis. Per ulteriori informazioni, consulta Creazione di un endpoint dell'interfaccia nella Guida per l'utente di Amazon VPC. Quando si specifica un flusso Amazon Kinesis Data Streams in un altro account, è necessario impostare i ruoli e le politiche per consentire l'accesso multi-account. Per ulteriori informazioni, consulta Esempio: lettura da un flusso Kinesis in un account diverso.

I processi ETL in streaming di AWS Glue possono rilevare automaticamente i dati compressi, decomprimere in modo trasparente i dati in streaming, eseguire le consuete trasformazioni sulla sorgente di ingresso e caricare nell'archivio di output.

AWS Glue supporta la decompressione automatica per i seguenti tipi di compressione, dato il formato di input:

Tipo di compressione	File Avro	Dato Avro	JSON	CSV	Grok
BZIP2	Sì	Sì	Sì	Sì	Sì
GZIP	No	Sì	Sì	Sì	Sì
SNAPPY	Sì (Snappy raw)	Sì (Snappy framed)	Sì (Snappy framed)	Sì (Snappy framed)	Sì (Snappy framed)
XZ	Sì	Sì	Sì	Sì	Sì
ZSTD	Sì	No	No	No	No
DEFLATE	Sì	Sì	Sì	Sì	Sì

Argomenti

Creazione di una connessione AWS Glue per un flusso di dati Apache Kafka

Per leggere da un flusso Apache Kafka, è necessario creare una connessione AWS Glue.

Come creare una connessione AWS Glue per un'origine Kafka (Console)

1. Apri la console all'indirizzo https://console.aws.amazon.com/glue/ AWS Glue .
2. Nel riquadro di navigazione, in Data catalog (Catalogo dati), seleziona Connections (Connessioni).
3. Scegliere Aggiungi connessione e, nella pagina Imposta proprietà della connessione, immettere un nome per la connessione.

   > **Note**
   >
   > Per ulteriori informazioni sulla specifica delle proprietà della connessione, consulta Proprietà della connessione di AWS Glue.

4. Per Tipo di connessione, scegli Kafka.
5. Per gli URL dei server di bootstrap Kafka, immettere il numero host e porta per i broker bootstrap per il cluster Amazon MSK o il cluster Apache Kafka. Utilizza solo endpoint Transport Layer Security (TLS) per stabilire la connessione iniziale al cluster Kafka. Gli endpoint in testo normale non sono supportati.

   Di seguito è riportato un elenco di esempio di coppie di nomi di host e numeri di porta per un cluster Amazon MSK.

   ```
   myserver1.kafka.us-east-1.amazonaws.com:9094,myserver2.kafka.us-
   east-1.amazonaws.com:9094,
   myserver3.kafka.us-east-1.amazonaws.com:9094
   ```

   Per ulteriori informazioni su come ottenere le informazioni del broker bootstrap, consulta Ottenere i broker bootstrap per un cluster Amazon MSK in Amazon Managed Streaming for Apache Kafka: Guida per gli sviluppatori.
6. Se si desidera una connessione sicura all'origine dati Kafka, seleziona Require SSL connection (Connessione SSL necessaria), e per Kafka private CA certificate location (Posizione del certificato emesso da una CA Kafka privata), inserisci un percorso Amazon S3 valido per un certificato SSL personalizzato.

   Per una connessione SSL a Kafka autogestito, il certificato personalizzato è obbligatorio. P Amazon MSK è facoltativo.

Per ulteriori informazioni su come specificare un certificato personalizzato per Kafka, consulta the section called “Proprietà della connessione SSL”.

7. Usa AWS Glue Studio o la AWS CLI per specificare un metodo di autenticazione del client Kafka. Per accedere, AWS Glue Studio seleziona AWS Gluedal menu ETL nel riquadro di navigazione a sinistra.

   Per ulteriori informazioni sui metodi di autenticazione client Kafka, consulta AWS Glue Proprietà di connessione Kafka per l'autenticazione client .
8. Opzionalmente, inserisci una descrizione, quindi scegli Next (Successivo).
9. Per un cluster Amazon MSK, specifica il cloud privato virtuale (VPC), la sottorete e il gruppo di sicurezza. Le informazioni VPC sono opzionali per Kafka autogestito.
10. Scegli Next (Successivo) per esaminare tutte le proprietà della connessione, quindi scegli Finish (Termina).

Per ulteriori informazioni sulle connessioni AWS Glue, consulta Connessione ai dati.

AWS Glue Proprietà di connessione Kafka per l'autenticazione client

Autenticazione SASL/GSSAPI (Kerberos)

La scelta di questo metodo di autenticazione consentirà di specificare le proprietà Kerberos.

Keytab Kerberos

Scegliere la posizione del file keytab. Un keytab memorizza le chiavi a lungo termine per uno o più principali. Per ulteriori informazioni, consulta la Documentazione di MIT Kerberos: keytab.

File Kerberos krb5.conf

Scegliere il file krb5.conf. Contiene l'area di autenticazione predefinita (una rete logica, simile a un dominio, che definisce un gruppo di sistemi sotto lo stesso KDC) e la posizione del server KDC. Per ulteriori informazioni, consulta la Documentazione di MIT Kerberos: krb5.conf.

Principale Kerberos e nome del servizio Kerberos

Immettere il nome del principale e il nome del servizio Kerberos. Per ulteriori informazioni, consulta Documentazione MIT Kerberos: principale Kerberos.

Autenticazione SASL/SCRAM-SHA-512

Scegliere questo metodo di autenticazione consentirà di specificare le credenziali di autenticazione.

AWS Secrets Manager

Cercare il token nella casella Cerca digitando il nome o l'ARN.

Nome utente e password del provider direttamente

Cercare il token nella casella Cerca digitando il nome o l'ARN.

Autenticazione client SSL

Scegliere questo metodo di autenticazione consente di selezionare la posizione del keystore client Kafka navigando su Amazon S3. Facoltativamente, è possibile inserire la password del keystore del client Kafka e la password della chiave del client Kafka.

Autenticazione IAM

Questo metodo di autenticazione non richiede specifiche aggiuntive ed è applicabile solo quando la sorgente di streaming è MSK Kafka.

autenticazione SASL/PLAIN

La scelta di questo metodo di autenticazione consente di specificare le credenziali di autenticazione.

Creazione di un catalogo dati per un'origine di streaming

Una tabella del catalogo dati che specifica le proprietà del flusso dei dati di origine, incluso lo schema dei dati, può essere creata manualmente per una sorgente di streaming. Questa tabella viene utilizzata come origine dati per il processo di streaming ETL.

Se non si conosce lo schema dei dati nel flusso dei dati di origine, è possibile creare la tabella senza uno schema. Quindi, quando si crea il processo ETL di streaming, è possibile attivare la AWS Gluefunzione di rilevamento dello schema. AWS Glue determina lo schema dai dati di streaming.

Usa la AWS Glueconsole, il AWS Command Line Interface (AWS CLI) o l'AWS GlueAPI per creare la tabella. Per informazioni sulla creazione manuale di una tabella con la console AWS Glue, vedere the section called “Tabelle AWS Glue”.

> **Note**
>
> Non puoi usare la AWS Lake Formation console per creare la tabella; devi usare la AWS Glue console.

Considera inoltre le seguenti informazioni per le origini di streaming in formato Avro o per i dati di log a cui è possibile applicare i pattern Grok.

- the section called “Note e restrizioni per le origini di streaming Avro”
- the section called “Applicazione di pattern Grok alle origini di streaming”

Argomenti

- Origine dati Kinesis
- Origine dati Kafka
- Origine della tabella di AWS Glue Schema Registry

Origine dati Kinesis

Durante la creazione della tabella, impostare le seguenti proprietà di streaming ETL (console).

Tipo di origine

Kinesis

Per una fonte Kinesis nello stesso account:

Regione

La AWS regione in cui risiede il servizio Amazon Kinesis Data Streams. Il nome della regione e del flusso Kinesis sono tradotti insieme in un flusso ARN.

Esempio: https://kinesis.us-east-1.amazonaws.com

Nome del flusso Kinesis

Nome del flusso come descritto in Creazione di un flusso nella Guida per gli sviluppatori Amazon Kinesis Data Streams.

Per un'origine Kinesis in un altro account, fai riferimento a questo esempio per configurare i ruoli e i criteri per consentire l'accesso a più account. Configura queste impostazioni:

Flusso ARN

L'ARN del flusso dei dati Kinesis con il quale il consumatore è registrato. Per ulteriori informazioni, consulta Amazon Resource Names (ARN) e AWS Service Namespaces nel. Riferimenti generali di AWS

ARN del ruolo assunto

L'Amazon Resource Name (ARN) del ruolo assegnato al ruolo da assumere.

Nome sessione (facoltativo)

Un identificatore della sessione del ruolo assunto.

Utilizza il nome della sessione del ruolo per identificare in modo univoco una sessione quando lo stesso ruolo viene assunto da diverse entità principali o per motivi diversi. In scenari multi-account, l'account proprietario del ruolo può vedere il nome della sessione del ruolo e può registrarlo. Il nome della sessione del ruolo viene utilizzato anche nell'ARN dell'entità ruolo assunto. Ciò significa che le successive richieste API tra più account che utilizzano le credenziali di sicurezza temporanee esporranno il nome della sessione del ruolo all'account esterno nei relativi log. AWS CloudTrail

Per impostare le proprietà di streaming ETL per Amazon Kinesis Data Streams (API AWS Glue o AWS CLI)

- Per impostare le proprietà di streaming ETL per un'origine Kinesis nello stesso account, specifica i parametri `streamName` e `endpointUrl` nella struttura `StorageDescriptor` dell'operazione API `CreateTable` o del comando CLI `create_table`.

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamName": "sample-stream",
  "endpointUrl": "https://kinesis.us-east-1.amazonaws.com"
 }
 ...
}
```

In alternativa, specifica il `streamARN`.

Example

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamARN": "arn:aws:kinesis:us-east-1:123456789:stream/sample-stream"
 }
 ...
}
```

- Per impostare le proprietà di streaming ETL per un'origine Kinesis nello stesso account, specifica i parametri `streamARN`, `awsSTSRoleARN` e `awsSTSSessionName` (facoltativo) nella struttura `StorageDescriptor` dell'operazione API `CreateTable` o del comando CLI `create_table`.

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamARN": "arn:aws:kinesis:us-east-1:123456789:stream/sample-stream",
  "awsSTSRoleARN": "arn:aws:iam::123456789:role/sample-assume-role-arn",
  "awsSTSSessionName": "optional-session"
 }
 ...
}
```

Origine dati Kafka

Durante la creazione della tabella, impostare le seguenti proprietà di streaming ETL (console).

Tipo di origine

Kafka

Per una fonte Kafka:

Nome argomento

Nome argomento specificato in Kafka.

Connessione

Connessione AWS Glue che fa riferimento a un'origine Kafka, come descritto in the section called “Creazione di una connessione per un flusso di dati Kafka”.

Origine della tabella di AWS Glue Schema Registry

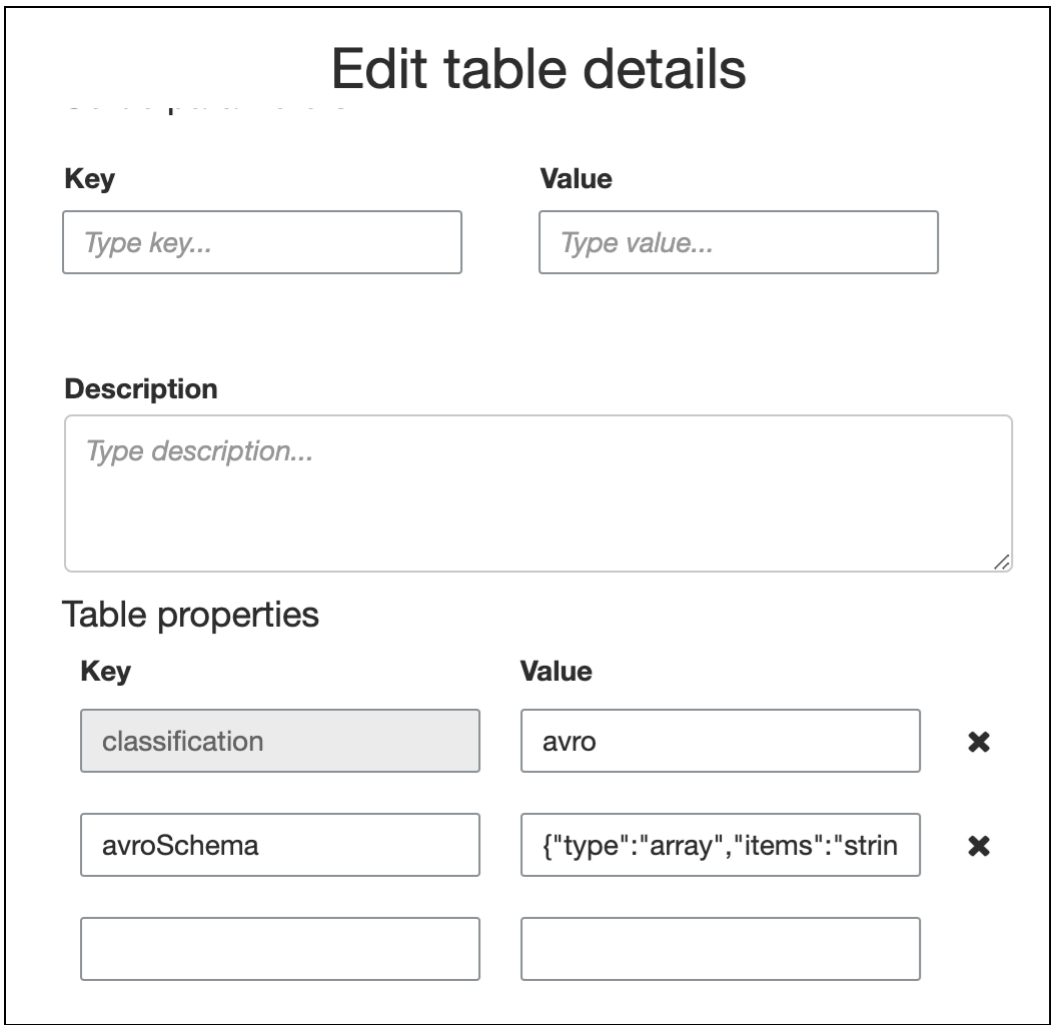
Per utilizzare AWS Glue Schema Registry per i processi di streaming, segui le istruzioni all'indirizzo Caso d'uso: AWS Glue Data Catalog su come creare o aggiornare una tabella Schema Registry.

Attualmente, lo streaming AWS Glue supporta solo il formato Glue Schema Registry Avro con inferenza dello schema impostata su `false`.

Note e restrizioni per le origini di streaming Avro

Le seguenti note e restrizioni si applicano alle origini di streaming nel formato Avro:

- Quando il rilevamento dello schema è attivato, lo schema Avro deve essere incluso nel payload. Quando disattivato, il payload deve contenere solo dati.
- Alcuni tipi di dati Avro non sono supportati nei frame dinamici. Non è possibile specificare questi tipi di dati quando si definisce lo schema con la pagina Define a schema (Definire uno schema) nella procedura guidata per la creazione della tabella nella console AWS Glue. Durante il rilevamento dello schema, i tipi non supportati nello schema Avro vengono convertiti in tipi supportati come segue:
  - `EnumType => StringType`
  - `FixedType => BinaryType`
  - `UnionType => StructType`
- Se si definisce lo schema della tabella utilizzando la pagina Define a schema (Definire uno schema) nella console, il tipo di elemento root implicito per lo schema è `record`. Se si desidera un tipo di elemento root diverso da `record`, ad esempio `array` o `map`, non è possibile specificare lo schema utilizzando la pagina Define a schema (Definire uno schema). È invece necessario saltare quella pagina e specificare lo schema come proprietà di tabella o all'interno dello script ETL.
  - Per specificare lo schema nelle proprietà della tabella, completa la procedura guidata per la creazione della tabella, modifica i dettagli della tabella e aggiungi una nuova coppia chiave-valore in Table properties (Proprietà della tabella). Utilizza la chiave `avroSchema`, e inserisci un oggetto JSON dello schema per il valore, come mostrato nello screenshot seguente.



- Per specificare lo schema nello script ETL, modifica l'istruzione di assegnazione `datasource0` e aggiungi la chiave `avroSchema` all'argomento `additional_options`, come mostrato nei seguenti esempi Python e Scala.

  Python

  ```
  SCHEMA_STRING = '{"type":"array","items":"string"}'
  datasource0 = glueContext.create_data_frame.from_catalog(database =
   "database", table_name = "table_name", transformation_ctx = "datasource0",
   additional_options = {"startingPosition": "TRIM_HORIZON", "inferSchema":
   "false", "avroSchema": SCHEMA_STRING})
  ```

  Scala

  ```
  val SCHEMA_STRING = """{"type":"array","items":"string"}"""
  val datasource0 = glueContext.getCatalogSource(database = "database", tableName
   = "table_name", redshiftTmpDir = "", transformationContext = "datasource0",
  ```

```
additionalOptions = JsonOptions(s"""{"startingPosition": "TRIM_HORIZON",
"inferSchema": "false", "avroSchema":"$SCHEMA_STRING"}""")).getDataFrame()
```

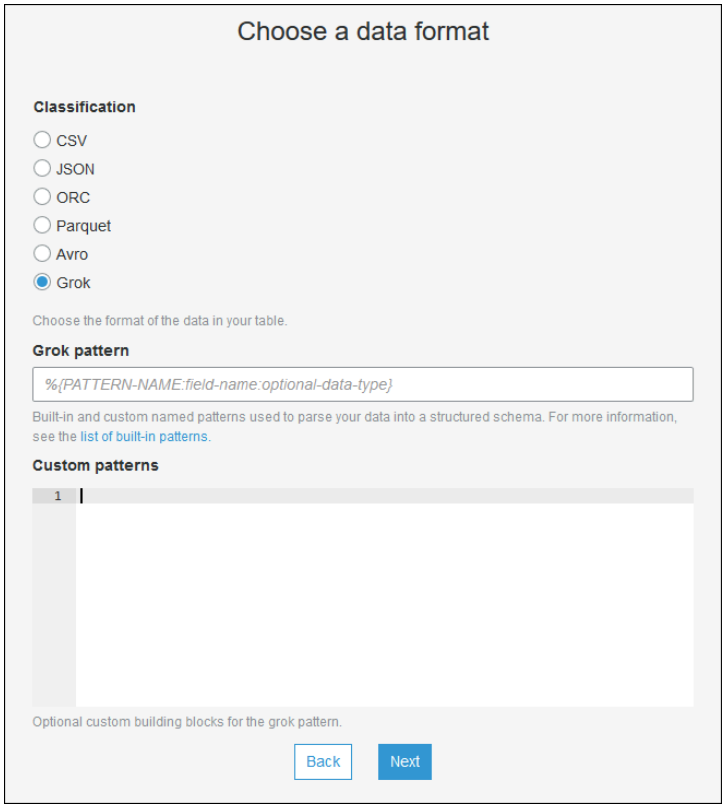
Applicazione di pattern Grok alle origini di streaming

È possibile creare un processo di streaming ETL per un'origine dati dei log e utilizzare i pattern Grok per convertire i registri in dati strutturati. Il processo ETL elabora quindi i dati come origine dati strutturata. È possibile specificare i pattern Grok da applicare quando si crea la tabella Catalogo dati per l'origine di streaming.

Per informazioni sui pattern Grok e sui valori delle stringhe di pattern personalizzati, consulta Scrittura di classificatori personalizzati grok.

Aggiungere pattern Grok alla tabella Catalogo dati (console)

- Utilizza la procedura guidata per la creazione della tabella e crea la tabella con i parametri specificati in the section called “Creazione di un catalogo dati per un'origine di streaming”. Specifica il formato dei dati come Grok, compila il pattern Grok e, facoltativamente, aggiungi pattern personalizzati in Custom patterns (optional) (Modelli personalizzati [facoltativo]).



  Premi Invio dopo ogni pattern personalizzato.

Aggiungere pattern Grok alla tabella Catalogo dati (API AWS Glue o AWS CLI)

- Aggiungi il parametro `GrokPattern` e, facoltativamente, il parametro `CustomPatterns` al processo API `CreateTable` o al comando CLI `create_table`.

```
 "Parameters": {
...
    "grokPattern": "string",
    "grokCustomPatterns": "string",
...
},
```

Esprimi `grokCustomPatterns` come stringa e usa "\n" come separatore tra i pattern.

Di seguito è riportato un esempio di specifica di questi parametri.

Example

```
"parameters": {
...
    "grokPattern": "%{USERNAME:username} %{DIGIT:digit:int}",
    "grokCustomPatterns": "digit \d",
...
}
```

Definizione delle proprietà di processo per un processo di streaming ETL

Quando definisci un processo di streaming ETL nella console AWS Glue, fornisci le seguenti proprietà specifiche dei flussi. Per le descrizioni di proprietà aggiuntive, consulta Definire le proprietà di processo per i processi Spark.

Ruolo IAM

Specificate il ruolo AWS Identity and Access Management (IAM) utilizzato per l'autorizzazione alle risorse utilizzate per eseguire il job, accedere alle sorgenti di streaming e accedere agli archivi dati di destinazione.

Per accedere ad Amazon Kinesis Data Streams, `AmazonKinesisFullAccess` AWS collega la policy gestita al ruolo o allega una policy IAM simile che consenta un accesso più dettagliato.

Per i criteri di esempio, consulta Controllo dell'accesso alle risorse Amazon Kinesis Data Streams tramite IAM.

Per ulteriori informazioni sulle autorizzazioni per l'esecuzione di processi in AWS Glue, consulta Gestione delle identità e degli accessi per AWS Glue.

Type

Scegli Spark streaming.

Versione AWS Glue

La versione di AWS Glue determina le versioni di Apache Spark e Python che sono disponibili per il processo. Scegli una selezione che specifica la versione di Python o Scala disponibile per il processo. AWS Glue La versione 2.0 con supporto Python 3 è la versione predefinita per lo streaming delle operazioni ETL.

Timeout dei processi

È possibile inserire una durata in minuti. Il valore predefinito è vuoto, il che significa che il processo potrebbe essere eseguito all'infinito.

Origine dati

Rimuovi la tabella creata in the section called “Creazione di un catalogo dati per un'origine di streaming”.

Destinazione dati

Esegui una di queste operazioni:

- Scegliere Crea tabelle nell'oggetto dati e specificare le seguenti proprietà dell'oggetto dati.

  Datastore

  Seleziona Amazon S3 o JDBC.

  Formato

  Scegli un formato qualsiasi. Tutti sono supportati per lo streaming.

- Scegli Use tables in the data catalog and update your data target (Usa tabelle nel catalogo dati e aggiorna la destinazione dati) e scegli una tabella per un data store JDBC.

Definizione dello schema di output

Esegui una di queste operazioni:

- Scegli Automatically detect schema of each record (Rileva automaticamente lo schema di ciascun record) per attivare il rilevamento dello schema. AWS Gluedetermina lo schema dai dati di streaming.
- Scegli Specify output schema for all records (Specificare lo schema di output per tutti i record) per utilizzare la trasformazione Apply Mapping (Applica mapping) per definire lo schema di output.

Script

È possibile fornire uno script personalizzato o modificare lo script generato per eseguire le operazioni supportate dal motore di Apache Spark Structured Streaming. Per informazioni sulle operazioni disponibili, consulta Operations on streaming/Datasets. DataFrames

Streaming di note e restrizioni ETL

Tieni presente le seguenti note e restrizioni:

- La decompressione automatica per i processi ETL in streaming di AWS Glue è disponibile solo per i tipi di compressione supportati. Tieni presente quanto segue:
  - Snappy framed si riferisce al formato di framing ufficiale di Snappy.
  - Deflate è supportato in Glue versione 3.0, non Glue versione 2.0.
- Quando si utilizza il rilevamento dello schema, non è possibile eseguire join dei dati di streaming.
- I processi ETL di streaming di AWS Glue non supportano il tipo di dati Union per il registro degli schemi AWS Glue con il formato Avro.
- Lo script ETL può utilizzare le trasformazioni integrate di AWS Glue e le trasformazioni native di Apache Spark Structured Streaming. Per ulteriori informazioni, consulta Operazioni sullo streaming DataFrames /Datasets sul sito Web di Apache Spark oppure. AWS Glue PySpark trasforma il riferimento
- Le operazioni ETL di streaming AWS Glue utilizzano i checkpoint per tenere traccia dei dati letti. Pertanto, un processo arrestato e riavviato riprende da dove era stato interrotto nello stream. Se si desidera rielaborare i dati, è possibile eliminare la cartella di checkpoint a cui si fa riferimento nello script.
- I segnalibri delle operazioni non sono supportati.
- Per utilizzare la funzionalità di fan-out avanzato di Flusso di dati Kinesis, consulta la pagina the section called “Utilizzo del fan-out avanzato nei processi di flussi di dati Kinesis”.

- Se si utilizza una tabella di catalogo dati creata da AWS Glue Schema Registry, quando diventa disponibile una nuova versione dello schema, per rifletterlo, è necessario effettuare le seguenti operazioni:
  1. Arrestare i processi associati alla tabella.
  2. Aggiornare lo schema per la tabella catalogo dati.
  3. Riavviare i processi associati alla tabella.

Corrispondenza dei record con FindMatches AWS Lake Formation

> **Note**
>
> La Corrispondenza dei record non è attualmente disponibile nelle seguenti regioni della console AWS Glue: Medio Oriente (Emirati Arabi Uniti), Europa (Spagna) (`eu-south-2`) ed Europa (Zurigo) (`eu-central-2`).

AWS Lake Formation fornisce caratteristiche di machine learning per creare trasformazioni personalizzate per ripulire i dati. Attualmente è disponibile una trasformazione denominata FindMatches. La trasformazione FindMatches consente di identificare record duplicati o corrispondenti nel set di dati, anche quando i record non dispongono di un identificatore univoco comune e nessun campo corrisponde esattamente. Ciò non richiede la scrittura di codice o la conoscenza del funzionamento del machine learning. FindMatches può essere utile in molti problemi diversi, ad esempio:

- Matching Customers (Corrispondenza di clienti): collegamento di record dei clienti tra diversi database, anche quando molti campi non corrispondono esattamente tra i database (ad es. diversa ortografia dei nomi, differenze di indirizzo, dati mancanti o imprecisi e così via).
- Matching Products (Corrispondenza di prodotti): abbinamento dei prodotti nel catalogo rispetto ad altre origini, ad esempio catalogo di prodotti rispetto al catalogo di un concorrente, in cui le voci sono strutturate in modo diverso.
- Improving Fraud Detection (Miglioramento del rilevamento delle frodi): identificazione di account dei clienti duplicati, determinazione di quando un nuovo account creato è (o potrebbe essere) una corrispondenza per un utente fraudolento noto.
- Other Matching Problems (Altri problemi di corrispondenza): abbinare indirizzi, film, elenchi di parti e così via. In generale, se un essere umano può esaminare le righe del database e determinare

che corrispondono, c'è una buona possibilità che la trasformazione FindMatches possa essere utile.

È possibile creare queste trasformazioni al momento della creazione di un processo. La trasformazione creata si basa su uno schema del datastore di origine e su dati di esempio del set di dati di origine etichettato (questo processo viene denominato "insegnamento" di una trasformazione). I record etichettati devono essere presenti nel set di dati di origine. In questo processo viene generato un file etichettato e quindi ricaricato nel modo appreso dalla trasformazione. Dopo aver istruito la trasformazione, è possibile chiamarla dal processo AWS Glue basato su Spark (PySpark o Scala Spark) e utilizzarla in altri script con un datastore di origine compatibile.

Dopo la creazione della trasformazione, questa viene memorizzata all'interno di AWS Glue. Nella console di AWS Glue è possibile gestire le trasformazioni create. Nel riquadro di navigazione in Integrazione dei dati ed ETL, Strumenti di classificazione dei dati > Corrispondenza dei record, puoi modificare e continuare a istruire la trasformazione del machine learning. Per ulteriori informazioni sulla gestione delle trasformazioni nella console, consultare Operare con le trasformazioni basate su machine learning nella console di AWS Glue.

> **Note**
>
> AWS Glue FindMatches versione 2.0 utilizza il bucket Amazon S3 `aws-glue-temp-<accountID>-<region>` per archiviare i file temporanei mentre la trasformazione elabora i dati. Puoi eliminare questi dati dopo aver completato l'esecuzione, manualmente o impostando una regola del ciclo di vita di Amazon S3.

Tipi di trasformazioni basate su machine learning

È possibile creare trasformazioni di machine learning per pulire i dati. Puoi chiamare queste trasformazioni dallo script ETL. I tuoi dati vengono trasformati in una struttura dati chiamata DynamicFrame, che è un'estensione di un `DataFrame` SQL Apache Spark. `DynamicFrame` contiene i tuoi dati e il suo schema di riferimento per elaborare i dati.

Sono disponibili i seguenti tipi di trasformazioni basate su machine learning:

Rilevamento delle corrispondenze

Individua i record duplicati nei dati di origine. È possibile addestrare questa trasformazione basata su machine learning etichettando dei set di dati di esempio e indicando tra quali righe

sono presenti delle corrispondenze. La trasformazione basata su machine learning apprende quali righe debbano essere abbinate man mano che vengono offerti dati di esempio etichettati. A seconda della configurazione della trasformazione, l'output è uno dei seguenti:

- Una copia della tabella di input con una colonna `match_id` aggiuntiva compilata con i valori che indicano insiemi di record corrispondenti. La colonna `match_id` è un identificatore arbitrario. Tutti i record con lo stesso `match_id` sono stati identificati come tra loro corrispondenti. I record con `match_id` diversi non corrispondono.
- Una copia della tabella di input con le righe duplicate rimosse. Se vengono rilevati molteplici duplicati, viene mantenuto il record con la chiave primaria minore.

Trova corrispondenze incrementali

La trasformazione Find matches può anche essere configurata per trovare le corrispondenze tra i frame esistenti e incrementali e restituire come output una colonna contenente un ID univoco per gruppo di corrispondenza.

Per ulteriori informazioni, consulta la pagina: Trovare corrispondenze incrementali

Utilizzo della trasformazione FindMatches

È possibile utilizzare la trasformazione `FindMatches` per individuare i record duplicati nei dati di origine. Viene generato o fornito un file di etichettatura che possa aiutare nell'addestramento della trasformazione.

> **Note**
>
> Attualmente, le trasformazioni `FindMatches` che usano una chiave di crittografia personalizzata non sono supportate nelle seguenti Regioni:
>
> - Asia Pacifico (Osaka): `ap-northeast-3`

Per iniziare a utilizzare la trasformazione FindMatches, è possibile completare la procedura seguente. Per un esempio più avanzato e dettagliato, consulta l'articolo del Blog sui big data di AWS: Harmonize data using AWS Glue and AWS Lake Formation FindMatches ML to build a customer 360 view.

Nozioni di base sull'utilizzo della trasformazione con rilevamento delle corrispondenze

Seguire questi passaggi per iniziare a usare la trasformazione `FindMatches`:

1. Creare una tabella in AWS Glue Data Catalog per ospitare i dati di origine da ripulire. Per informazioni su come creare un crawler, consultare Uso di crawler nella console AWS Glue.

   Se i dati di origine sono contenuti in un file di testo, ad esempio un file di valori separati da virgola (CSV), tenere conto delle seguenti considerazioni:

   - Mantenere il file CSV contenente i record di input e i file di etichettatura in cartelle separate. In caso contrario, il crawler di AWS Glue potrebbe considerarli come più componenti della stessa tabella e creare tabelle nel catalogo dati in modo non corretto.
   - A meno che il file CSV includa solo caratteri ASCII, assicurarsi che per la codifica dei file CSV venga utilizzato UTF-8 senza BOM (Byte Order Mark). Microsoft Excel spesso aggiunge un BOM all'inizio dei file CSV UTF-8. Per rimuoverlo, aprire il file CSV con un editor di testo e salvare nuovamente il file in formato UTF-8 senza BOM.

2. Nella console di AWS Glue, creare un processo e scegliere il tipo di trasformazione Find matches (Rilevamento delle corrispondenze)

   > **Important**
   >
   > La tabella dell'origine dati selezionata per il processo può contenere fino a un massimo di 100 colonne.

3. Indica a AWS Glue di generare un file di etichettatura scegliendo Generate labeling file (Genera file di etichettatura). AWS Glue utilizza il primo passaggio per raggruppare record simili per ciascun `labeling_set_id` in modo da poter rivedere tali raggruppamenti. Etichetta corrispondenze nella colonna `label`.

   - Se già disponi di un file di etichettatura, ossia di un esempio di record che indicano righe corrispondenti, carica il file su Amazon Simple Storage Service (Amazon S3). Per informazioni sul formato del file di etichettatura, consultare Formato del file di etichettatura. Continuare con la fase 4.

4. Scaricare il file di etichettatura ed etichettare il file come descritto nella sezione Etichettatura.

5. Caricare il file di etichettatura corretto. AWS Glue esegue delle attività per addestrare la trasformazione al riconoscimento delle corrispondenze.

   Nella pagina di elenco delle Machine learning transforms (Trasformazioni basate su machine learning), scegliere la scheda History (Cronologia). Questa pagina indica quando AWS Glue esegue le seguenti attività:

   - Import labels (Importa le etichette)

- Export labels (Esporta le etichette)
- Generate labels (Genera le etichette)
- Estimate quality (Valuta la qualità)

6. Per creare una migliore trasformazione, è possibile scaricare, etichettare e caricare il file etichettato in modo iterativo. Nell'esecuzione iniziale, molti record potrebbero essere rilevati come non corrispondenti. Ma man mano che prosegue l'addestramento tramite la verifica del file di etichettatura, AWS Glue avanza nell'apprendimento.
7. Valutare e ottimizzare la trasformazione tramite la valutazione delle prestazioni e dei risultati della ricerca delle corrispondenze. Per ulteriori informazioni, consulta Ottimizzazione delle trasformazioni basate su machine learning in AWS Glue.

Etichettatura

Quando `FindMatches` genera un file di etichettatura, i record vengono selezionati dalla tabella di origine. Sulla base del training precedente, `FindMatches` identifica i record più importanti da cui apprendere.

L'atto di etichettatura consiste nella modifica di un file di etichettatura (ad esempio, un foglio di calcolo come Microsoft Excel) e l'aggiunta di identificatori, o etichette, nella colonna `label` che identifica i record con o senza corrispondenze. È importante avere una chiara e coerente definizione di corrispondenza nei dati di origine. `FindMatches` apprende sulla base dei record designati come corrispondenti (o meno) e utilizza le decisioni dell'utente per ricavare le informazioni necessarie all'individuazione dei record duplicati.

Quando il file di etichettatura viene generato da `FindMatches`, vengono generati circa 100 record. Questi 100 record sono in genere suddivisi in 10 set di etichettatura, dove ogni set di etichettatura è identificato da un unico `labeling_set_id` generato da `FindMatches`. Ogni set di etichettatura deve essere considerato come un'attività di etichettatura separata indipendente dagli altri set di etichettatura. Il tuo compito consiste nell'identificare i record corrispondenti e non corrispondenti all'interno di ciascun set di etichette.

Suggerimenti per la modifica dei file di etichettatura in un foglio di calcolo.

Quando si modifica il file di etichettatura in un foglio di calcolo, considerare i seguenti aspetti:

- Il file potrebbe non aprirsi con le colonne dei campi completamente espanse. Per visualizzare i contenuti di tali celle, potrebbe essere necessario espandere le colonne `labeling_set_id` e `label`.

- Se la colonna chiave primaria è un numero, ad esempio un tipo di dato `long`, il foglio di calcolo potrebbe interpretarlo come un numero e modificarne il valore. Questo valore chiave deve essere trattato come un testo. Per risolvere il problema, formattare tutte le celle nella colonna chiave primaria come Text data (Formato testo).

Formato del file di etichettatura

Il file di etichettatura generato da AWS Glue per insegnare la trasformazione `FindMatches` utilizza il seguente formato. Se si genera il proprio file per AWS Glue, anch'esso deve seguire questo formato:

- Si tratta di un file di valori separati da virgola (CSV).
- Deve essere codificato in `UTF-8`. Se il file è stato modificato con Microsoft Windows, potrebbe essere codificato con `cp1252`.
- Affinché possa essere passato a AWS Glue è necessario che si trovi su Amazon S3.
- Utilizza un numero modesto di righe per ogni attività di etichettatura. Sono consigliate 10-20 righe per attività, anche se 2-30 righe per attività sono accettabili. Le attività superiori a 50 righe non sono consigliate e potrebbero causare risultati scadenti o errori di sistema.
- Se si dispone di dati già etichettati costituiti da coppie di record etichettati come "corrispondenza" o "nessuna corrispondenza", questo va bene. Queste coppie etichettate possono essere rappresentate come set di etichettatura di dimensione 2. In questo caso etichettare entrambi i record con, ad esempio, una lettera "A" se corrispondono, ma etichettare uno come "A" e uno come "B" se non corrispondono.

> **Note**
>
> Poiché possiede delle colonne aggiuntive, il file di etichettatura presenta uno schema diverso da quello di un file che contiene i dati di origine. Posizionare il file di etichettatura in una cartella diversa da quella di qualsiasi altro file CSV di input della trasformazione in modo che il crawler di AWS Glue non lo prenda in considerazione al momento della creazione delle tabelle nel catalogo dati. In caso contrario, le tabelle create dal crawler di AWS Glue potrebbero non rappresentare correttamente i dati.

- Le prime due colonne (`labeling_set_id`, `label`) sono richieste obbligatoriamente da AWS Glue. Le colonne rimanenti devono corrispondere allo schema dei dati che devono essere elaborati.

- Per ogni `labeling_set_id`, è necessario identificare tutti i record corrispondenti utilizzando la stessa etichetta. Un'etichetta è una stringa univoca posizionata nella colonna `label`. Consigliamo di usare etichette contenenti caratteri semplici, ad esempio A, B, C e così via. Le etichette considerano in modo differente le maiuscole dalle minuscole e vengono inserite nella colonna `label`.
- Le righe che contengono lo stesso `labeling_set_id` e la stessa etichetta si intendono etichettate come corrispondenza.
- Le righe che contengono lo stesso `labeling_set_id` e un'etichetta diversa si intendono etichettate come non una corrispondenza
- Le righe che contengono un `labeling_set_id` diverso non trasmettono alcuna informazione a favore o contro la corrispondenza.

  Di seguito è riportato un esempio di etichettatura dei dati:

labeling_set_id	etichetta	first_name	last_name	Compleanno
ABC123	A	John	Doe	04/01/1980
ABC123	B	Jane	Smith	04/03/1980
ABC123	A	Johnny	Doe	04/01/1980
ABC123	A	Jon	Doe	04/01/1980
DEF345	A	Richard	Jones	12/11/1992
DEF345	A	Rich	Jones	11/12/1992
DEF345	B	Sara	Jones	12/11/1992
DEF345	C	Richie	Jones Junior.	05/06/2017
DEF345	B	Sara	Jones-Walker	12/11/1992
GHI678	A	Roberto	Miller	1/3/1999
GHI678	A	Bob	Miller	1/3/1999
XYZABC	A	Guglielmo	Robinson	2/5/2001

labeling_set_id	etichetta	first_name	last_name	Compleanno
XYZABC	B	Andrea	Robinson	2/5/1971

- Nell'esempio precedente identifichiamo John/Johnny/Jon Doe come una corrispondenza e insegniamo al sistema che questi record non corrispondono a Jane Smith. Separatamente, insegniamo al sistema che Richard e Rich Jones sono la stessa persona, ma che questi dischi non corrispondono a Sarah Jones/Jones-Walker e Richie Jones Jr.
- Come si può vedere, l'ambito delle etichette è limitato al `labeling_set_id`. Quindi le etichette non attraversano i limiti imposti dal `labeling_set_id`. Ad esempio, un'etichetta "A" nel `labeling_set_id` 1 non ha alcuna relazione con l'etichetta "A" nel `labeling_set_id` 2.
- Se un record non ha alcuna corrispondenza all'interno di un set di etichette, assegnargli un'etichetta univoca. Ad esempio, Jane Smith non corrisponde ad alcun record nel set di etichettatura ABC123, quindi è l'unico record in quel set di etichettatura con l'etichetta di B.
- Il set di etichettatura "GHI678" mostra che un set di etichettatura può essere costituito da solo due record che hanno la stessa etichetta per mostrare che corrispondono. Allo stesso modo, "XYZABC" mostra due record con etichette diverse per mostrare che non corrispondono.
- Si noti che a volte un set di etichette non può contenere corrispondenze (ovvero, si attribuisce a ogni record nel set di etichette un'etichetta diversa) o un set di etichette potrebbe essere "uguale" (ad essi è stata assegnata la stessa etichetta). Questo va bene fintanto che i set di etichettatura contengono collettivamente esempi di record "uguali" o "non uguali" secondo i propri criteri.

> **Important**
>
> Confermare che il ruolo IAM passato a AWS Glue abbia accesso al bucket Amazon S3 che contiene il file di etichettatura. Per convenzione, le policy di AWS Glue concedono le autorizzazioni sui bucket o sulle cartelle di Amazon S3 i cui nomi contengono il prefisso aws-glue-. Se i file di etichettatura si trovano in percorsi diversi, aggiungere al ruolo IAM l'autorizzazione di accesso a tale posizione.

Ottimizzazione delle trasformazioni basate su machine learning in AWS Glue

È possibile ottimizzare le trasformazioni basate su machine learning in AWS Glue al fine di migliorare i risultati delle operazioni di pulizia dei dati affinché soddisfino gli obiettivi desiderati. Per migliorare

la trasformazione, è possibile addestrarla generando un set di dati da etichettare, aggiungendo le etichette e quindi ripetendo questi passaggi diverse volte fino a ottenere i risultati desiderati. È inoltre possibile applicare l'ottimizzazione modificando alcuni parametri del sistema di machine learning.

Per ulteriori informazioni sulle trasformazioni basate su machine learning, consultare Corrispondenza dei record con FindMatches AWS Lake Formation.

Argomenti

- Misurazioni del machine learning
- Scelta tra precisione e recupero
- Scelta tra accuratezza e costo
- Stima della qualità delle corrispondenze utilizzando i punteggi di confidenza delle corrispondenze
- Addestramento della trasformazione di rilevamento delle corrispondenze

Misurazioni del machine learning

Per comprendere le misurazioni che vengono utilizzate per ottimizzare una trasformazione basata su machine learning, è necessario avere familiarità con la seguente terminologia:

Vero positivo (True positive, TP)

: Una corrispondenza nei dati correttamente individuata dalla trasformazione, denominata anche colpo a segno.

Vero negativo (True negative , TN)

: Una mancata corrispondenza nei dati correttamente esclusa dalla trasformazione.

Falso positivo (False positive, FP)

: Una mancata corrispondenza nei dati che la trasformazione ha erroneamente classificato come una corrispondenza, denominata anche falso allarme.

Falso negativo (False negative, FN)

: Una corrispondenza nei dati non rilevata dalla trasformazione, denominata anche colpo mancato.

Per ulteriori informazioni sulla terminologia utilizzata nel campo del machine learning, consultare la voce Matrice di confusione su Wikipedia.

Per ottimizzare le trasformazioni basate su machine learning, è possibile modificare il valore delle seguenti misurazioni nella sezione Advanced properties (Proprietà avanzate) della trasformazione.

- Precision (Precisione) misura la capacità della trasformazione di individuare veri positivi sul numero totale di record che identifica come positivi (veri positivi e falsi positivi). Per ulteriori informazioni, consulta la voce Precisione e recupero su Wikipedia.
- Recall (Recupero) misura la capacità della trasformazione di individuare i veri positivi rispetto al totale dei record che compongono i dati di origine. Per ulteriori informazioni, consulta la voce Precisione e recupero su Wikipedia.
- Accuracy (Accuratezza) misura la capacità della trasformazione di individuare i veri positivi e i veri negativi. L'incremento dell'accuratezza implica maggiori risorse di elaborazione e costi superiori. Tuttavia permette di raggiungere anche un livello maggiore di recupero. Per ulteriori informazioni, consultare la voce Accuratezza e precisione su Wikipedia.
- Cost (Costo) misura la quantità di risorse di elaborazione (e quindi di denaro) consumate per l'esecuzione della trasformazione.

Scelta tra precisione e recupero

Ogni trasformazione `FindMatches` contiene un parametro `precision-recall`. È possibile utilizzare questo parametro per specificare uno dei seguenti requisiti:

- Se la preoccupazione maggiore riguarda la possibilità che la trasformazione indichi la corrispondenza tra due record quando in effetti tale corrispondenza non esiste, allora è opportuno enfatizzare l'aspetto della precisione.
- Se la preoccupazione maggiore riguarda la mancata rilevazione di record tra i quali esiste in effetti una corrispondenza, allora è opportuno enfatizzare l'aspetto del recupero.

È possibile regolare questo compromesso all'interno della console di AWS Glue o utilizzando le operazioni API di machine learning di AWS Glue.

Quando favorire la precisione

È opportuno favorire la precisione se la preoccupazione maggiore riguarda il rischio che `FindMatches` stabilisca una relazione tra due record quando in effetti tale corrispondenza non esiste. Per favorire la precisione, scegliere un valore più alto per il compromesso tra precisione e recupero. Con un valore più alto, la trasformazione `FindMatches` richiede un numero maggiore di elementi a sostegno per stabilire se una coppia di record deve essere è legata da una

corrispondenza. Si incrementa la predisposizione della trasformazione a supporre che tra i record non esista una corrispondenza.

Ad esempio, si supponga di utilizzare `FindMatches` per rilevare gli elementi duplicati in un catalogo di video e di assegnare al parametro precisione-recupero della trasformazione un valore elevato. Se la trasformazione rileva erroneamente che Star Wars: Una nuova speranza è la stessa cosa di Star Wars: L'impero colpisce ancora, a un cliente che desidera Una nuova speranza potrebbe essere mostrato L'impero colpisce ancora. Si tratterebbe di un'esperienza utente scadente.

Tuttavia, se la trasformazione non riesce a rilevare che Star Wars: Una nuova speranza e Star Wars: Episodio IV - Una nuova speranza sono lo stesso elemento, il cliente potrebbe essere confuso all'inizio ma potrebbe alla fine riconoscere i due elementi come lo stesso film. Sarebbe un errore, ma non così grave come nel caso precedente.

Quando favorire il recupero

È opportuno favorire il recupero se la preoccupazione maggiore riguarda il rischio che i risultati della trasformazione `FindMatches` possano non riuscire a rilevare una coppia di record tra i quali esiste un effetti una corrispondenza. Per favorire il recupero, scegliere un valore più basso per il compromesso tra precisione e recupero. Con un valore più basso, la trasformazione `FindMatches` richiede un numero minore di elementi a sostegno per decidere che una coppia di record è legata da una corrispondenza. Si incrementa la predisposizione della trasformazione a supporre che tra i record esista una corrispondenza.

Ad esempio, questa potrebbe essere una priorità per un'azienda che si occupa di sicurezza. Si supponga di confrontare l'elenco dei clienti con uno di noti frodatori e che sia importante determinare se un cliente è un frodatore. Si sta utilizzando `FindMatches` per trovare le corrispondenze tra l'elenco dei frodatori e quello dei clienti. Ogni volta che `FindMatches` rileva una corrispondenza tra i due elenchi, a un revisore umano viene assegnato il compito di verificare che la persona sia, in effetti, un frodatore. L'azienda potrebbe scegliere di favorire il recupero rispetto alla precisione. In altre parole, è preferibile che i verificatori debbano esaminare manualmente e rigettare alcuni casi in cui il cliente non è un frodatore piuttosto che fallire nell'identificazione di un cliente che si trova, in effetti, nell'elenco dei frodatori.

Come favorire sia la precisione che il recupero

Il modo migliore per migliorare la precisione e il recupero è etichettare una maggiore quantità di dati. Etichettando una maggiore quantità di dati, migliora la precisione globale della trasformazione `FindMatches`, con conseguenti miglioramenti sia della precisione che del recupero. Tuttavia, anche

nel caso della trasformazione più accurata possibile, esiste sempre un'area grigia dove è necessario sperimentare se favorire precisione o recupero oppure scegliere un valore intermedio.

Scelta tra accuratezza e costo

Ogni trasformazione `FindMatches` contiene un parametro `accuracy-cost`. È possibile utilizzare questo parametro per specificare uno dei seguenti requisiti:

- Se la preoccupazione maggiore riguarda la possibilità che la trasformazione riveli con precisione la corrispondenza tra due record, allora è opportuno enfatizzare l'aspetto dell'accuratezza.
- Se la preoccupazione maggiore riguarda il costo o la velocità di esecuzione della trasformazione, allora è opportuno enfatizzare l'aspetto della riduzione del costo.

È possibile regolare questo compromesso all'interno della console di AWS Glue o utilizzando le operazioni API di machine learning di AWS Glue.

Quando favorire l'accuratezza

È opportuno favorire l'accuratezza se la preoccupazione maggiore riguarda il rischio che i risultati della trasformazione `find matches` non includano le corrispondenze. Per favorire l'accuratezza, scegliere un valore più alto per il compromesso tra accuratezza e costo. Con un valore più elevato, la trasformazione `FindMatches` richiede più tempo per approfondire la ricerca sui record tra i quali esiste una corrispondenza. Si noti che questo parametro non rende meno probabile la possibilità di indicare erroneamente corrispondenti due record tra i quali non esiste nessuna corrispondenza. Si incrementa la predisposizione della trasformazione a dedicare un tempo maggiore alla ricerca delle corrispondenze.

Quando favorire il costo

È opportuno favorire il costo se la preoccupazione maggiore riguarda il costo di esecuzione della trasformazione `find matches` rispetto al numero di corrispondenze rilevate. Per favorire il costo, scegliere un valore più basso per il compromesso tra accuratezza e costo. Con un valore più basso, la trasformazione `FindMatches` richiede una minore quantità di risorse per l'esecuzione. Si incrementa la predisposizione della trasformazione alla ricerca di un numero minore di corrispondenze. Utilizzare questa impostazione se, pur favorendo la ricerca di costi inferiori, i risultati sono comunque accettabili.

Come favorire sia l'accuratezza che il costo

Per esaminare un numero maggiore di coppie di record al fine di determinare la presenza di eventuali corrispondenze, serve un tempo di elaborazione maggiore. Se si desidera ridurre i costi senza ridurre la qualità, è possibile seguire la procedura illustrata qui di seguito:

- Eliminare i record dell'origine dati per i quali la presenza di una corrispondenza non è di interesse.
- Eliminare le colonne dell'origine dati che si è certi non siano utili ai fini della determinazione della presenza o meno di una corrispondenza. Un buon metodo per stabilirle quali siano è eliminare le colonne che non sembrano influenzare la propria valutazione sul fatto che un insieme di record rappresentino "la stessa cosa".

Stima della qualità delle corrispondenze utilizzando i punteggi di confidenza delle corrispondenze

I punteggi di confidenza delle corrispondenze forniscono una stima della qualità delle corrispondenze trovate da FindMatches per distinguere tra registri corrispondenti in cui il modello di machine learning è altamente sicuro, incerto o improbabile. Un punteggio di confidenza delle corrispondenze sarà compreso tra 0 e 1, dove il punteggio più alto significa una somiglianza più elevata. L'esame dei punteggi di confidenza delle corrispondenze consente di distinguere tra cluster di corrispondenze in cui il sistema è altamente sicuro (che potresti decidere di unire), cluster su cui il sistema è incerto (che potresti decidere di far esaminare da un essere umano) e cluster che il sistema ritiene improbabile (che potresti decidere di rifiutare).

Potresti dover modificare i tuoi dati di formazione in situazioni in cui vedi un punteggio di confidenza elevato, ma determinare che non ci sono corrispondenze, o dove vedi un punteggio basso determinare che ci sono, di fatto, corrispondenze.

I punteggi di corrispondenza sono particolarmente utili quando esistono set di dati industriali di grandi dimensioni, dove è impossibile riesaminare ogni decisione FindMatches.

I punteggi di confidenza delle corrispondenze sono disponibili in AWS Glue versione 2.0 o successive.

Generazione di punteggi di confidenza delle corrispondenze

È possibile generare punteggi di confidenza delle corrispondenze impostando il valore booleano di `computeMatchConfidenceScores` su Vero quando si chiama `FindMatches` o lAPI `FindIncrementalMatches`.

AWS Glue aggiunge una nuova `column match_confidence_score` all'output.

Esempi di punteggio di corrispondenza

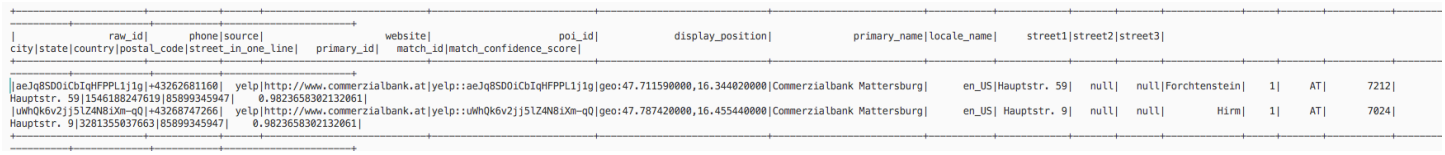
Considera, ad esempio, le corrispondenze di registri seguenti:

Punteggio >= 0,9

Riepilogo dei registri corrispondenti:

```
  primary_id  |   match_id  | match_confidence_score

3281355037663    85899345947   0.9823658302132061
1546188247619    85899345947   0.9823658302132061
```

Informazioni:



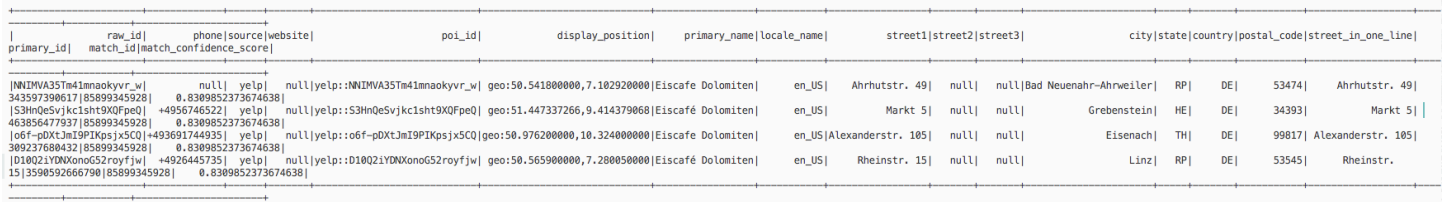
Da questo esempio, possiamo vedere che due registri sono molto simili e condividono `display_position`, `primary_name` `estreet name`.

Punteggio >= 0,8 e punteggio < 0,9

Riepilogo dei registri corrispondenti:

```
  primary_id  |   match_id  | match_confidence_score

309237680432     85899345928   0.8309852373674638
3590592666790    85899345928   0.8309852373674638
343597390617     85899345928   0.8309852373674638
249108124906     85899345928   0.8309852373674638
463856477937     85899345928   0.8309852373674638
```

Informazioni:



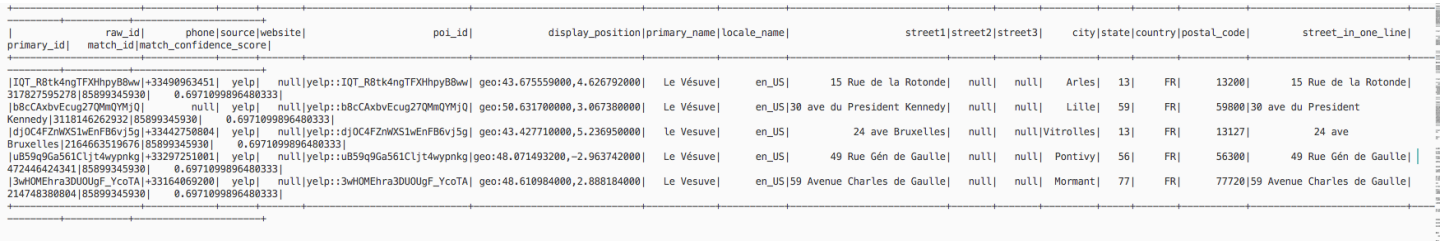
Da questo esempio, possiamo vedere che questi registri condividono gli stessi `primary_name` `ecountry`.

Punteggio >= 0,6 e punteggio < 0,7

Riepilogo dei registri corrispondenti:

```
  primary_id  |   match_id   | match_confidence_score

2164663519676    85899345930   0.6971099896480333
 317827595278    85899345930   0.6971099896480333
 472446424341    85899345930   0.6971099896480333
3118146262932    85899345930   0.6971099896480333
 214748380804    85899345930   0.6971099896480333
```

Informazioni:



Da questo esempio, possiamo vedere che questi registri condividono solo lo stesso `primary_name`.

Per ulteriori informazioni, consulta:

- Fase 5: aggiunta ed esecuzione di un processo con la trasformazione basata su machine learning
- PySpark: Classe FindMatches
- PySpark: Classe FindIncrementalMatches
- Scala: Classe FindMatches
- Scala: Classe FindIncrementalMatches

Addestramento della trasformazione di rilevamento delle corrispondenze

Ogni trasformazione `FindMatches` deve essere addestrata rispetto a ciò che deve essere considerato una corrispondenza e ciò che non deve essere considerato tale. Una trasformazione viene addestrata aggiungendo delle etichette a un file e caricando tali scelte su AWS Glue.

È possibile orchestrare questa operazione di etichettatura all'interno della console di AWS Glue o utilizzando le operazioni API di machine learning di AWS Glue.

Quante volte è necessario eseguire l'operazione di etichettatura? Quante etichette sono necessarie?

Le risposte a queste domande dipendono generalmente dall'utilizzatore. È necessario valutare se `FindMatches` offre il livello di accuratezza di cui si necessita e se si ritiene che un'etichettatura aggiuntiva possa valere la pena. Il modo migliore per deciderlo è analizzare i parametri "Precision", "Recall" e "Area sotto la curva di precisione e recall" che è possibile generare selezionando Estimate quality (Valuta la qualità) all'interno della console di AWS Glue. Dopo aver etichettato ulteriori insiemi di attività, ricalcolare questi parametri e verificare il loro eventuale miglioramento. Se, dopo l'etichettatura di alcuni insiemi di attività, non si percepisce un miglioramento del parametro di interesse, la qualità della trasformazione potrebbe aver raggiunto uno stato stazionario.

Perché servono le etichette sia per gli eventi veri positivi che per quelli veri negativi?

La trasformazione `FindMatches` ha bisogno di esempi sia positivi che negativi per comprendere cosa intende l'utente per corrispondenza. Se si stanno etichettando dei dati di addestramento generati da `FindMatches` (ad esempio, utilizzando l'opzione I do not have labels (Non dispongo di etichette)), `FindMatches` prova a generare un set di "id di insiemi di etichette". All'interno di ciascuna attività, si assegna la stessa "etichetta" ad alcuni record e diverse "etichette" ad altri record. In altre parole, le attività generalmente non prevedono solo la presenza di elementi tutti uguali o tutti diversi (anche se è normale che una specifica attività comprenda elementi "tutti uguali" o "tutti diversi").

Se si sta addestrando la trasformazione `FindMatches` utilizzando l'opzione Upload labels from S3 (Caricamento delle etichette da S3), provare a includere sia esempi di record corrispondenti e che di record non corrispondenti. È accettabile averne di un solo tipo. Queste etichette consentono di creare una trasformazione `FindMatches` più accurata, ma è comunque necessario etichettare alcuni record generati utilizzando l'opzione Generate labeling file (Genera file di etichettatura).

Come posso fare in modo che la trasformazione rilevi le corrispondenze esattamente come è stata addestrata a fare?

La trasformazione `FindMatches` esegue un processo di apprendimento a partire dalle etichette fornite, perciò potrebbe generare coppie di record che non rispettano tali etichette. Per costringere la trasformazione `FindMatches` a rispettare le etichette, selezionare Enforce Provided Labels (Imponi l'utilizzo delle etichette fornite) in Find Matches Parameter (Parametro di rilevamento delle corrispondenze).

Quali tecniche è possibile utilizzare quando una trasformazione basata su ML identifica come corrispondenti degli elementi che non lo sono?

È possibile utilizzare le seguenti tecniche:

- Incrementare il valore di `precisionRecallTradeoff`. Questa operazione porterà all'individuazione di un numero minore di corrispondenze ma, al raggiungimento di un valore sufficientemente elevato, dovrebbe anche suddividere un cluster di grandi dimensioni in componenti più piccole.
- Selezionare le righe di output corrispondenti ai risultati errati e riformattarle sotto forma di insieme di dati per l'etichettatura (rimuovendo la colonna `match_id` e aggiungendo le colonne `labeling_set_id` e `label`). Se necessario, spezzettarle (suddividerle) in più insiemi di dati per l'etichettatura al fine di assicurare che l'addetto all'etichettatura possa concentrarsi su ogni set di dati durante il processo di etichettatura. Quindi, etichettare correttamente i set corrispondenti e caricare il file di etichettatura accodandolo alle etichette esistenti. Queste informazioni potrebbero addestrare la trasformazione a sufficienza su cosa cercare per comprendere lo schema.
- (Avanzato) Infine, controllare i dati per verificare la presenza di uno schema che il sistema non sta rilevando. Pre-elaborare tali dati utilizzando le funzioni standard di AWS Glue per normalizzarli. Evidenziare gli elementi dai quali si desidera che l'algoritmo tragga insegnamenti separando i dati che l'utente ritiene importanti per la loro diversità nelle rispettive colonne. Oppure creare colonne combinate a partire dalle colonne i cui dati sono da ritenersi correlati.

Operare con le trasformazioni basate su machine learning nella console di AWS Glue

È possibile utilizzare AWS Glue per creare trasformazioni basate su machine learning personalizzate che possono essere utilizzate per ripulire i dati. È possibile creare queste trasformazioni al momento della creazione di un processo nella console di AWS Glue.

Per informazioni su come creare una trasformazione basata su machine learning, consultare Corrispondenza dei record con FindMatches AWS Lake Formation.

Argomenti

Proprietà della trasformazione

Per visualizzare una trasformazione basata su machine learning esistente, accedere alla AWS Management Console e aprire la console di AWS Glue all'indirizzo https://console.aws.amazon.com/glue/. Nel pannello di navigazione sotto Integrazione dati ed ETL, scegli Strumenti di classificazione dei dati > Corrispondenza dei record.

Le proprietà di ogni trasformazione:

Nome trasformazione

Il nome univoco che assegnato alla trasformazione al momento della creazione.

ID

Un identificatore unico della trasformazione.

Numero delle etichette

Il numero di etichette nel file di etichettatura fornito per l'addestramento della trasformazione.

Stato

Indica se la trasformazione è Ready (Pronta) o Needs training (Ha bisogno di addestramento). Per eseguire correttamente una trasformazione basata su machine learning in un processo, questa deve trovarsi nello stato Ready (Pronta).

Creato

La data di creazione della trasformazione.

Modificato

La data dell'ultimo aggiornamento della trasformazione.

Descrizione

La descrizione fornita per la trasformazione, se ne è stata fornita una.

Versione AWS Glue

La versione di AWS Glue utilizzata.

ID esecuzione

Il nome univoco che assegnato alla trasformazione al momento della creazione.

Tipo di attività

Il tipo di trasformazione basata su machine learning; ad esempio, Find matching records (Rilevamento record corrispondenti).

Stato

Indica lo stato dell'esecuzione dell'attività. Gli stati possibili comprendono:

- Avvio in corso
- In esecuzione
- In arresto
- Arrestato
- Riuscito
- Non riuscito
- Timeout

Errore

Se lo stato è Non riuscito, viene visualizzato un messaggio di errore che descrive il motivo dell'errore.

Aggiunta e modifica della trasformazione basata su machine learning

Nella console AWS Glue è possibile visualizzare, eliminare, impostare e addestrare o ottimizzare una trasformazione. Selezionare la casella di controllo accanto alla trasformazione nell'elenco, scegliere Action (Operazione) e quindi scegliere l'operazione che si desidera eseguire.

Creazione di una nuova trasformazione ML

Per aggiungere una nuova trasformazione di machine learning, scegli Crea trasformazione. Segui le istruzioni nella procedura guidata Aggiungi crawler. Per ulteriori informazioni, consulta Corrispondenza dei record con FindMatches AWS Lake Formation.

Fase 1. Imposta le proprietà della trasformazione.

1. Inserisci il nome e la descrizione (facoltativo).
2. Facoltativamente, imposta la configurazione di sicurezza. Per informazioni, consulta Utilizzo della crittografia dati con le trasformazioni basate su machine learning.

3. Facoltativamente, configura le impostazioni di esecuzione delle attività. Le impostazioni di esecuzione delle attività consentono di personalizzare la modalità di esecuzione dell'attività. Seleziona il tipo di e il numero di worker, il timeout dell'attività (in minuti), il numero di nuovi tentativi e la versione di AWS Glue.
4. Facoltativamente, imposta i tag. I tag sono etichette che possono essere assegnate a una risorsa AWS. Ciascun tag è formato da una chiave e da un valore facoltativo. È possibile utilizzare i tag per cercare e filtrare la risorsa o monitorare i costi AWS.

Fase 2. Scegli la tabella e la chiave primaria.

1. Scegli il database e la tabella di Catalogo AWS Glue.
2. Scegli una chiave primaria dalla tabella selezionata. La colonna della chiave primaria contiene in genere un identificatore univoco per ogni record nell'origine dati.

Fase 3. Seleziona le opzioni di ottimizzazione.

1. Per Richiamo o precisione, scegli il valore di regolazione per ottimizzare la trasformazione in modo da favorire il richiamo o la precisione. Per impostazione predefinita, è selezionata l'opzione Bilanciato, ma puoi scegliere di favorire il richiamo o la precisione; puoi anche scegliere l'opzione Personalizzato e inserire un valore compreso tra 0,0 e 1,0 (inclusi).
2. Per Costo o precisione inferiore, scegli il valore di regolazione per favorire un costo o una precisione inferiori oppure scegli Personalizzato e inserisci un valore compreso tra 0,0 e 1,0 (inclusi).
3. Per Forza corrispondenza, scegli Forza l'output a corrispondere alle etichette se desideri addestrare la trasformazione ML forzando l'output a corrispondere alle etichette utilizzate.

Fase 4. Revisione e creazione.

1. Esamina le opzioni per i passaggi da 1 a 3.
2. Scegli Modifica per qualsiasi passaggio che desideri modificare. Scegli Crea trasformazione per completare la procedura guidata di creazione della trasformazione.

Utilizzo della crittografia dati con le trasformazioni basate su machine learning

Quando si aggiunge una trasformazione basata su machine learning a AWS Glue, è possibile specificare facoltativamente una configurazione di sicurezza associata all'origine dati o alla

destinazione dati. Se il bucket Amazon S3 utilizzato per memorizzare i dati è crittografato con una configurazione di sicurezza, specifica la stessa configurazione di sicurezza durante la creazione della trasformazione.

È inoltre possibile scegliere di utilizzare la crittografia lato server con AWS KMS (SSE-KMS) per crittografare il modello e le etichette per impedire a persone non autorizzate di ispezionarlo. Se si sceglie questa opzione, viene richiesto di scegliere AWS KMS key per nome, oppure puoi scegliere Enter a key ARN (Inserisci l'ARN della chiave). Se si sceglie di inserire l'ARN per la chiave KMS, viene visualizzato un secondo campo in cui è possibile inserire l'ARN della chiave KMS.

> **Note**
>
> Attualmente, le trasformazioni ML che usano una chiave di crittografia personalizzata non sono supportate nelle seguenti Regioni:
>
> - Asia Pacifico (Osaka): `ap-northeast-3`

Visualizzazione dei dettagli della trasformazione

Visualizzazione delle proprietà della trasformazione

La pagina Proprietà della trasformazione include gli attributi della trasformazione. Mostra i dettagli relativi alla definizione della trasformazione, tra cui i seguenti:

- Transform name (Nome della trasformazione) mostra il nome della trasformazione.
- Tipo elenca il tipo della trasformazione.
- Stato indica se la trasformazione è pronta per essere utilizzata in uno script o un processo.
- Force output to match labels (Forza l'output affinché corrisponda alle etichette) mostra se la trasformazione esegue una forzatura affinché l'output corrisponda alle etichette indicate dall'utente.
- Versione Spark è correlato alla versione di AWS Glue che hai scelto nelle Proprietà esecuzione processo all'aggiunta della trasformazione. AWS Glue 1.0 e Spark 2.4 sono consigliati per la maggior parte dei clienti. Per ulteriori informazioni, consulta Versioni di AWS Glue.

Schede Cronologia, Stima qualità e Tag

I dettagli includono le informazioni definite al momento della creazione della trasformazione. Per visualizzare i dettagli di una trasformazione, selezionare la trasformazione nell'elenco delle Machine

learning transforms (Trasformazioni basate su machine learning) e rivedere le informazioni contenute nelle seguenti schede:

- Cronologia
- Stima della qualità
- Tag

Cronologia

La scheda History (Cronologia) mostra la cronologia delle esecuzioni della trasformazione. Per addestrare una trasformazione, vengono eseguiti diversi tipi di attività. Per ogni attività, i parametri di esecuzione includono:

- Run ID (ID esecuzione) è un identificatore creato da AWS Glue per ogni esecuzione di questo processo.
- Task type (Tipo di attività) mostra il tipo di attività eseguita.
- Status (Stato) mostra la corretta conclusione di ogni esecuzione posizionando quella più recente in cima all'elenco.
- Errore mostra i dettagli di un messaggio di errore se l'esecuzione non riesce.
- Start time (Orario inizio) mostra la data e l'ora (ora locale) in cui è stato avviato il processo.
- Orario fine mostra la data e l'ora (ora locale) in cui il processo è finito.
- Log collega ai log scritti in `stdout` per questa esecuzione di processo.

  Il link Logs ti porta ad Amazon CloudWatch Logs. Qui è possibile visualizzare i dettagli sulle tabelle create in AWS Glue Data Catalog ed eventuali errori riscontrati. Puoi gestire il periodo di conservazione dei log sulla CloudWatch console. Il periodo di conservazione dei log di default è `Never Expire`. Per ulteriori informazioni su come modificare il periodo di conservazione, consulta Change Log Data Retention in CloudWatch Logs nella Amazon CloudWatch Logs User Guide.
- File di etichettatura mostra un link ad Amazon S3 che permette di raggiungere un file di etichettatura generato.

Stima della qualità

La scheda Estimate quality (Stima qualità) mostra i parametri utilizzati per misurare la qualità della trasformazione. Le stime vengono calcolate confrontando le previsioni di corrispondenza delle

trasformazioni utilizzando un sottoinsieme di dati etichettati rispetto alle etichette fornite. Queste stime sono approssimative. Da questa scheda è possibile richiamare l'esecuzione dell'attività Estimate quality (Stima qualità).

La scheda Estimate quality (Stima qualità) mostra i parametri dell'ultima esecuzione Estimate quality (Stima qualità), incluse le seguenti proprietà:

- Area under the Precision-Recall curve (Area sotto la curva precisione-recupero) è un singolo numero che stima il limite superiore della qualità complessiva della trasformazione. È indipendente dalla scelta del parametro precisione-recupero. Valori più elevati indicano che si dispone di un compromesso precisione-recupero migliore.
- Precision (Precisione) stima la frequenza di correttezza della trasformazione quando prevede una corrispondenza.
- Recall upper limit (Limite superiore recupero) stima quanto spesso la trasformazione prevede una corrispondenza in caso di effettiva presenza.
- F1 stima l'accuratezza della trasformazione con un valore tra 0 e 1, dove 1 è la migliore precisione. Per ulteriori informazioni, consulta la voce F1 score su Wikipedia.
- La tabella Column importance (Importanza colonna) mostra i nomi delle colonne e il punteggio di importanza per ogni colonna. L'importanza delle colonne consente di comprendere il modo in cui queste contribuiscono al modello, identificando quali colonne nei record vengono maggiormente utilizzate per la corrispondenza. Questi dati possono richiedere di aggiungere o modificare il set di etichette per aumentare o diminuire l'importanza delle colonne.

  La colonna Importance (Importanza) fornisce un punteggio numerico per ogni colonna, come decimale non maggiore di 1,0.

Per ulteriori informazioni su come comprendere le stime della qualità rispetto alla vera qualità, consultare Stime sulla qualità rispetto alla qualità end-to-end (vera).

Per ulteriori informazioni sull'ottimizzazione della trasformazione, consultare Ottimizzazione delle trasformazioni basate su machine learning in AWS Glue.

Stime sulla qualità rispetto alla qualità end-to-end (vera)

AWS Glue stima la qualità della trasformazione passando al modello addestrato tramite machine learning interno un certo numero di coppie di record per i quali sono state fornite delle etichette corrispondenti ma che il modello non ha mai visto in precedenza. Queste stime di qualità sono una

funzione della qualità del modello addestrato tramite machine learning (che dipende dal numero di record etichettati per "addestrare" la trasformazione). Il richiamo end-to-end, o vero, (che non viene calcolato automaticamente da`ML transform`) è influenzato anche dal meccanismo di `ML transform` filtraggio che propone un'ampia varietà di possibili corrispondenze al modello di apprendimento automatico.

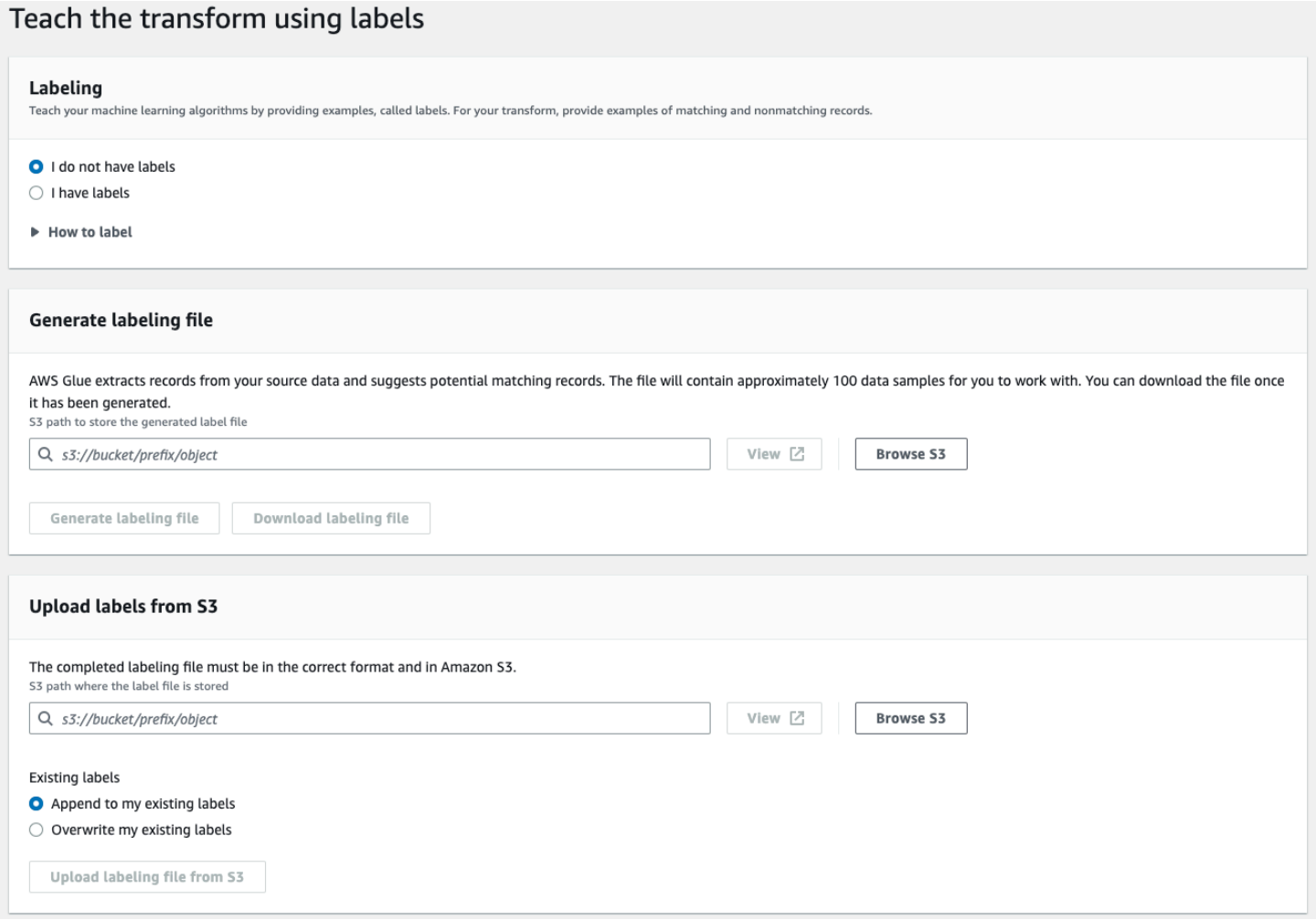
È possibile ottimizzare tale metodo di filtraggio principalmente utilizzando il cursore Costo o accuratezza inferiore. Spostando il cursore verso Accuratezza per favorire questo aspetto, il sistema esegue una ricerca più vasta e approfondita delle coppie di record che potrebbero rappresentare delle corrispondenze. Più coppie di record vengono inserite nel modello di apprendimento automatico e il tuo richiamo effettivo si avvicina alla metrica `ML transform` di end-to-end richiamo stimata. Di conseguenza, le variazioni nella end-to-end qualità delle partite dovute a variazioni del rapporto costo/precisione delle partite in genere non si riflettono nella stima della qualità.

Tag

I tag sono etichette che possono essere assegnate a una risorsa AWS. Ciascun tag è formato da una chiave e da un valore facoltativo. È possibile utilizzare i tag per cercare e filtrare la risorsa o monitorare i costi AWS.

Insegnamento delle trasformazioni utilizzando le etichette

È possibile insegnare la trasformazione ML tramite le etichette (esempi) scegliendo Insegna la trasformazione dalla pagina dei dettagli della trasformazione ML. Quando addestri l'algoritmo di machine learning fornendo esempi (chiamati etichette), puoi scegliere etichette esistenti da utilizzare o creare un file di etichettatura.



- Etichettatura: se hai delle etichette, scegli Ho delle etichette. Se non disponi di etichette, puoi comunque proseguire con il passaggio successivo per generare un file di etichettatura.
- Genera un file di etichettatura: AWS Glue estrae i record dai dati di origine e suggerisce potenziali record corrispondenti. Scegli il bucket Amazon S3 per archiviare il file di etichette generato. Scegli Genera file di etichettatura per avviare il processo. Al termine, scegli Scarica il file di etichettatura. Il file scaricato avrà una colonna per le etichette in cui potrai inserire le etichette.
- Carica etichette da Amazon S3: scegli il file di etichettatura completo dal bucket Amazon S3 in cui è archiviato il file di etichette. Quindi, scegli se aggiungere le etichette alle etichette esistenti o sovrascriverle. Scegli Carica file di etichettatura da Amazon S3.

Tutorial: creazione di una trasformazione basata su machine learning con AWS Glue

Questa esercitazione guida l'utente nelle operazioni necessarie per creare e gestire una trasformazione basata su machine learning (ML) utilizzando AWS Glue. Prima di seguire questa

esercitazione, è necessario sapere come utilizzare la console di AWS Glue per aggiungere crawler e processi e modificare script. È inoltre necessario avere familiarità con la ricerca e il download di file tramite la console Amazon Simple Storage Service (Amazon S3).

In questo esempio, viene creata una trasformazione `FindMatches` per identificare i record corrispondenti, insegnando ad essa come identificare i record con corrispondenze e quelli senza, e la si utilizza in un processo AWS Glue. Il processo AWS Glue genera un nuovo file Amazon S3 con una colonna aggiuntiva denominata `match_id`.

I dati di origine utilizzati da questa esercitazione cono contenuti in un file denominato `dblp_acm_records.csv`. Questo file è una versione modificata derivante da pubblicazioni accademiche (DBLP e ACM) disponibili presso la fonte originale set di dati DBLP ACM . Il file `dblp_acm_records.csv` è un file di valori separati da virgole (CSV) in formato UTF-8 senza BOM (Byte Order Mark).

Un secondo file, `dblp_acm_labels.csv`, è un esempio di file di etichettatura che contiene sia i record con corrispondenze che quelli senza utilizzato per addestrare la trasformazione come parte dell'esercitazione.

Argomenti

Fase 1: crawling dei dati di origine

In primo luogo, esegui il crawling del file CSV di origine su Amazon S3 per creare una tabella di metadati corrispondente nel catalogo dati.

> **Important**
>
> Per ottenere dal crawler la creazione di una tabella per il solo file CSV, archivia il file CSV dei dati di origine in una cartella Amazon S3 diversa da quella degli altri file.

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione, selezionare Crawlers (Crawler), Add crawler (Aggiungi Crawler).
3. Seguire la procedura guidata per creare ed eseguire un crawler denominato `demo-crawl-dblp-acm` con output indirizzato verso il database `demo-db-dblp-acm`. Se questo non esiste già, durante l'esecuzione della procedura guidata è necessario creare il database `demo-db-dblp-acm`. Scegli un percorso di inclusione su Amazon S3 per i dati di esempio nella regione AWS corrente. Ad esempio, per `us-east-1`, il percorso di inclusione per i dati di origine su Amazon S3 è `s3://ml-transforms-public-datasets-us-east-1/dblp-acm/records/dblp_acm_records.csv`.

   Se conclude l'attività con successo, il crawler crea la tabella `dblp_acm_records_csv` con le seguenti colonne: id, title, authors, venue, year e source.

Fase 2: aggiunta di una trasformazione basata su machine learning

A questo punto, aggiungere una trasformazione basata su machine learning basata sullo schema dei dati della tabella di origine creata dal crawler e denominata `demo-crawl-dblp-acm`.

1. Nella console AWS Glue, nel riquadro di navigazione in Integrazione dati e ETL, scegli Strumenti di classificazione dei dati > Corrispondenza dei record, quindi Aggiungi trasformazione. Quindi segui la procedura guidata per creare una trasformazione `Find matches` con le seguenti proprietà.

   a. Alla voce Transform name (Nome trasformazione), immettere **`demo-xform-dblp-acm`**. Questo è il nome della trasformazione che viene utilizzato per trovare le corrispondenze nei dati di origine.

   b. Alla voce IAM role (Ruolo IAM), scegli un ruolo IAM che disponga delle autorizzazioni per accedere ai dati di origine su Amazon S3, ai file di etichettatura e alle operazioni API di AWS Glue. Per ulteriori informazioni, consulta Creazione di un ruolo IAM per AWS Glue nella Guida per sviluppatori di AWS Glue.

   c. Alla voce Data source (Origine dati), scegliere la tabella di database denominata dblp_acm_records_csv nel database demo-db-dblp-acm.

   d. Alla voce Primary key (Chiave primaria), scegliere la colonna chiave primaria della tabella, id.

2. Nella procedura guidata, scegliere Finish (Fine) e tornare all'elenco delle ML transforms (Trasformazioni basate su ML).

Fase 3: addestramento della trasformazione basata su machine learning

A questo punto è necessario addestrare la trasformazione basata su machine learning utilizzando il file di etichettatura di esempio del tutorial.

Non è possibile utilizzare una trasformazione basata su machine learning in un processo di estrazione, trasformazione e caricamento (ETL) finché il suo stato non è Ready for use (Pronta per l'uso). Affinché la trasformazione sia pronta, è necessario addestrarla a identificare i record con corrispondenze e quelli senza fornendo esempi di record con corrispondenza e di record senza corrispondenza. Per addestrare la trasformazione, è possibile scegliere Generate a label file (Genera un file di etichettatura), aggiungere le etichette e quindi selezionare Upload label file (Carica un file di etichettatura). In questa esercitazione è possibile utilizzare il file di etichettatura di esempio denominato `dblp_acm_labels.csv`. Per ulteriori informazioni sul processo di etichettatura, consultare Etichettatura.

1. Nella console AWS Glue, seleziona Corrispondenza dei record nel riquadro di navigazione.
2. Scegliere la trasformazione `demo-xform-dblp-acm` e quindi scegliere Action (Operazione), Teach (Addestra). Seguire la procedura guidata per addestrare la trasformazione `Find matches`.
3. Nella pagina delle proprietà della trasformazione, scegliere I have labels (Dispongo delle etichette). Scegliere un percorso su Amazon S3 nella regione AWS corrente per il file di etichettatura di esempio. Ad esempio, nel caso di `us-east-1`, caricare il file di etichettatura fornito dal percorso su Amazon S3 `s3://ml-transforms-public-datasets-us-east-1/dblp-acm/labels/dblp_acm_labels.csv` con l'opzione di overwrite (sovrascrivere) le etichette esistenti. Il file di etichettatura deve essere situato su Amazon S3 nella stessa regione in cui si trova la console di AWS Glue.

   Quando si carica un file di etichettatura, AWS Glue avvia un'attività per aggiungere o sovrascrivere le etichette utilizzate per addestrare la trasformazione sull'elaborazione dell'origine dati.
4. Nella pagina finale della procedura guidata scegliere Finish (Fine) e tornare all'elenco delle ML transforms (Trasformazioni basate su ML).

Fase 4: stima della qualità della trasformazione basata su machine learning

Successivamente, è possibile stimare la qualità della propria trasformazione basata su machine learning. La qualità varia in base al numero di etichettature eseguite. Per ulteriori informazioni sulla stima della qualità, consultare Stima della qualità.

1. Nella console AWS Glue, nel riquadro di navigazione in Integrazione dati e ETL, scegli Strumenti di classificazione dei dati > Corrispondenza dei record.
2. Scegliere la trasformazione `demo-xform-dblp-acm` e scegliere la scheda Estimate quality (Stima della qualità). Questa scheda visualizza l'attuale stima di della qualità, se disponibile, per la trasformazione.
3. Scegliere Estimate quality (Stima della qualità) per avviare un'attività di stima della qualità della trasformazione. La precisione della stima della qualità si poggia sull'etichettatura dei dati di origine.
4. Passare alla scheda History (Cronologia). In questo riquadro sono elencate le esecuzioni di attività per ogni trasformazione, inclusa l'attività di Estimate quality (Stima della qualità). Per ulteriori informazioni sull'esecuzione, scegliere Logs (Log). Verificare che, al termine dell'operazione, lo stato di esecuzione sia Succeeded (Completata correttamente).

Fase 5: aggiunta ed esecuzione di un processo con la trasformazione basata su machine learning

In questo passaggio si utilizza la trasformazione basata su machine learning per aggiungere ed eseguire un processo in AWS Glue. Quando la trasformazione `demo-xform-dblp-acm` è Ready for use (Pronta per l'uso) è possibile utilizzarla in un processo ETL.

1. Nel riquadro di navigazione della console di AWS Glue, scegliere Jobs (Processi).
2. Scegliere Add job (Aggiungi processo) e seguire la procedura guidata per creare un processo ETL Spark con uno script generato. Per le proprietà della trasformazione scegliere i seguenti valori:

    a. In Name (Nome) scegliere il processo di esempio presente in questa esercitazione, demo-etl-dblp-acm.

    b. Alla voce IAM role (Ruolo IAM), scegli un ruolo IAM che disponga delle autorizzazioni per accedere ai dati di origine su Amazon S3, ai file di etichettatura dei dati e alle operazioni API di AWS Glue. Per ulteriori informazioni, consulta Creazione di un ruolo IAM per AWS Glue nella Guida per sviluppatori di AWS Glue.

c. Alla voce ETL language (Linguaggio ETL) scegli Scala. Questo è il linguaggio di programmazione dello script ETL.

d. Come Script file name (Nome del file di script), scegli demo-etl-dblp-acm. Questo è il nome del file dello script Scala (uguale al nome del processo).

e. Come Data source (Origine dati), scegliere dblp_acm_records_csv. L'origine dati scelta deve corrispondere allo schema dell'origine dati della trasformazione basata su machine learning.

f. Alla voce Transform type (Tipo di trasformazione), scegliere Find matching records (Individuazione record corrispondenti) per creare un processo che utilizza una trasformazione basata su machine learning.

g. Annullare la selezione di Remove duplicate records (Rimuovi record duplicati). Si sceglie di non rimuovere i record duplicati perché i record di output dispongono di un campo aggiuntivo `match_id` accodato.

h. Alla voce Transform (Trasformazione), scegliere demo-xform-dblp-acm, la trasformazione basata su machine learning utilizzata del processo.

i. Alla voce Create tables in your data target (Crea tabelle nella destinazioni dati), scegliere di creare tabelle con le seguenti proprietà:

- Tipo di memorizzazione dei dati: **`Amazon S3`**
- Formato: **`CSV`**
- Tipo di compressione: **`None`**
- Percorso di destinazione: il percorso Amazon S3 in cui viene scritto l'output del processo (nella regione AWS attuale della console)

3. Scegliere Save job and edit script (Salva processo e modifica script) per visualizzare la pagina dell'editor dello script.

4. Modificare lo script per aggiungere un'istruzione che faccia sì che l'output del processo sia scritto sul Target path (Percorso di destinazione) in un file a singola partizione. Aggiungere questa istruzione immediatamente dopo l'istruzione che esegue la trasformazione `FindMatches`. Le istruzioni sono simili alle seguenti.

```
val single_partition = findmatches1.repartition(1)
```

È necessario modificare l'istruzione `.writeDynamicFrame(findmatches1)` per scrivere l'output come `.writeDynamicFrame(single_partion)`.

5. Dopo aver modificato lo script, scegliere Save (Salva). Lo script modificato è simile al codice riportato qui di seguito, ma personalizzato in base al proprio tuo ambiente.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.ml.FindMatches
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    // @type: DataSource
    // @args: [database = "demo-db-dblp-acm", table_name = "dblp_acm_records_csv",
 transformation_ctx = "datasource0"]
    // @return: datasource0
    // @inputs: []
    val datasource0 = glueContext.getCatalogSource(database = "demo-db-dblp-acm",
 tableName = "dblp_acm_records_csv", redshiftTmpDir = "", transformationContext =
 "datasource0").getDynamicFrame()
    // @type: FindMatches
    // @args: [transformId = "tfm-123456789012", emitFusion = false,
 survivorComparisonField = "<primary_id>", transformation_ctx = "findmatches1"]
    // @return: findmatches1
    // @inputs: [frame = datasource0]
    val findmatches1 = FindMatches.apply(frame = datasource0, transformId
 = "tfm-123456789012", transformationContext = "findmatches1",
 computeMatchConfidenceScores = true)


    // Repartition the previous DynamicFrame into a single partition.
    val single_partition = findmatches1.repartition(1)


    // @type: DataSink
```

```
    // @args: [connection_type = "s3", connection_options = {"path": "s3://aws-
glue-ml-transforms-data/sal"}, format = "csv", transformation_ctx = "datasink2"]
    // @return: datasink2
    // @inputs: [frame = findmatches1]
    val datasink2 = glueContext.getSinkWithFormat(connectionType =
 "s3", options = JsonOptions("""{"path": "s3://aws-glue-ml-transforms-
data/sal"}"""), transformationContext = "datasink2", format =
 "csv").writeDynamicFrame(single_partition)
    Job.commit()
  }
}
```

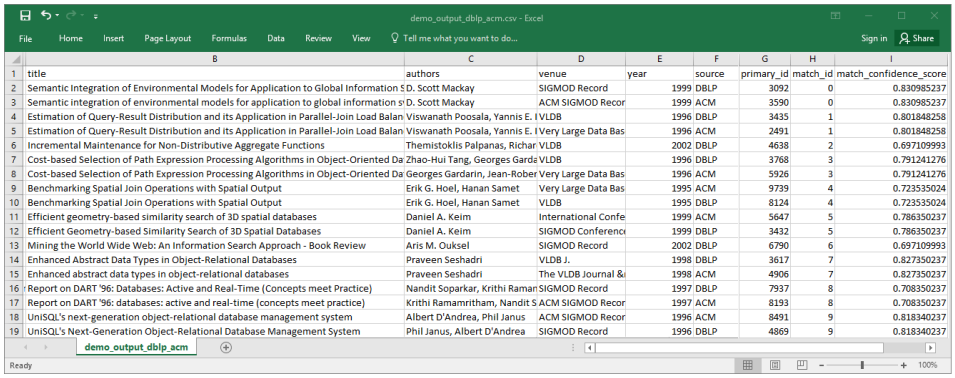
6. Scegliere Run job (Esegui processo) per avviare l'esecuzione del processo. Controllare lo stato del processo nell'elenco dei processi. Al termine del processo, nella finestra ML transform (Trasformazione ML), History (Cronologia), è disponibile una nuova riga Run ID (ID esecuzione) aggiunta di tipo ETL job (Processo ETL).
7. Passare alla scheda Jobs (Processi), History (Cronologia). In questo riquadro vengono elencate le esecuzioni dei processi. Per ulteriori informazioni sull'esecuzione, scegliere Logs (Log). Verificare che, al termine dell'operazione, lo stato di esecuzione sia Succeeded (Completata correttamente).

Fase 6: verifica dei dati di output da Amazon S3

In questa fase si verifica l'output dell'esecuzione del processo nel bucket Amazon S3 scelto al momento dell'aggiunta del processo. È possibile scaricare il file di output sulla propria macchina locale e verificare che i record corrispondenti siano stati identificati.

1. Apri la console Amazon S3 su https://console.aws.amazon.com/s3/.
2. Scaricare il file di output di destinazione del processo `demo-etl-dblp-acm`. Aprire il file in un foglio di calcolo (per aprire il file correttamente, potrebbe essere necessario aggiungere al file l'estensione `.csv`).

   L'immagine seguente mostra un estratto dell'output in Microsoft Excel.



demo_output_dblp_acm.csv - Excel

	B	C	D	E	F	G	H	I
1	title	authors	venue	year	source	primary_id	match_id	match_confidence_score
2	Semantic Integration of Environmental Models for Application to Global Information S	D. Scott Mackay	SIGMOD Record	1999	DBLP	3092	0	0.830985237
3	Semantic integration of environmental models for application to global information s	D. Scott Mackay	ACM SIGMOD Recor	1999	ACM	3590	0	0.830985237
4	Estimation of Query-Result Distribution and its Application in Parallel-Join Load Balan	Viswanath Poosala, Yannis E. I	VLDB	1996	DBLP	3435	1	0.801848258
5	Estimation of Query-Result Distribution and its Application in Parallel-Join Load Balan	Viswanath Poosala, Yannis E. I	Very Large Data Bas	1996	ACM	2491	1	0.801848258
6	Incremental Maintenance for Non-Distributive Aggregate Functions	Themistoklis Palpanas, Richar	VLDB	2002	DBLP	4638	2	0.697109993
7	Cost-based Selection of Path Expression Processing Algorithms in Object-Oriented Da	Zhao-Hui Tang, Georges Garda	VLDB	1996	DBLP	3768	3	0.791241276
8	Cost-based Selection of Path Expression Processing Algorithms in Object-Oriented Da	Georges Gardarin, Jean-Rober	Very Large Data Bas	1996	ACM	5926	3	0.791241276
9	Benchmarking Spatial Join Operations with Spatial Output	Erik G. Hoel, Hanan Samet	Very Large Data Bas	1995	ACM	9739	4	0.723535024
10	Benchmarking Spatial Join Operations with Spatial Output	Erik G. Hoel, Hanan Samet	VLDB	1995	DBLP	8124	4	0.723535024
11	Efficient geometry-based similarity search of 3D spatial databases	Daniel A. Keim	International Confe	1999	ACM	5647	5	0.786350237
12	Efficient Geometry-based Similarity Search of 3D Spatial Databases	Daniel A. Keim	SIGMOD Conferenc	1999	DBLP	3432	5	0.786350237
13	Mining the World Wide Web: An Information Search Approach - Book Review	Aris M. Ouksel	SIGMOD Record	2002	DBLP	6790	6	0.697109993
14	Enhanced Abstract Data Types in Object-Relational Databases	Praveen Seshadri	VLDB J.	1998	DBLP	3617	7	0.827350237
15	Enhanced abstract data types in object-relational databases	Praveen Seshadri	The VLDB Journal &	1998	ACM	4906	7	0.827350237
16	Report on DART '96: Databases: Active and Real-Time (Concepts meet Practice)	Nandit Soparkar, Krithi Raman	SIGMOD Record	1997	DBLP	7937	8	0.708350237
17	Report on DART '96: databases: active and real-time (concepts meet practice)	Krithi Ramamritham, Nandit S	ACM SIGMOD Recor	1997	ACM	8193	8	0.708350237
18	UniSQL's next-generation object-relational database management system	Albert D'Andrea, Phil Janus	ACM SIGMOD Recor	1996	ACM	8491	9	0.818340237
19	UniSQL's Next-Generation Object-Relational Database Management System	Phil Janus, Albert D'Andrea	SIGMOD Record	1996	DBLP	4869	9	0.818340237

demo_output_dblp_acm

Ready 100%

L'origine e la destinazione dei dati contano entrambe 4.911 record. Tuttavia, la trasformazione `Find matches` aggiunge un'altra colonna denominata `match_id` per identificare i record corrispondenti nell'output. Le righe con gli stessi `match_id` sono considerate record corrispondenti. La `match_confidence_score` è un numero compreso tra 0 e 1 che fornisce una stima della qualità delle corrispondenze trovate da `Find matches`.

3. Ordinare il file di output per `match_id` al fine di visualizzare facilmente i record corrispondenti. Confrontare i valori nelle altre colonne per confermare i risultati della trasformazione `Find matches`. Se i risultati non sono soddisfacenti, è possibile continuare ad addestrare la trasformazione aggiungendo ulteriori etichette.

   È anche possibile ordinare i file per un altro campo, ad esempio `title`, per vedere se record con titoli simili presentano lo stesso `match_id`.

Trovare corrispondenze incrementali

La caratteristica FindMatches permette di identificare registri duplicati o corrispondenti nel set di dati, anche quando i registri non dispongono di un identificatore univoco comune e nessun campo corrisponde esattamente. La versione iniziale di Trova corrispondenze trasforma i registri corrispondenti identificati all'interno di un singolo set di dati. Quando si aggiungono nuovi dati al set, avrai già dovuto unirli con il set di dati pulito esistente e rieseguire la corrispondenza con il set di dati unito completo.

La funzione di corrispondenza incrementale semplifica la corrispondenza con i registri incrementali rispetto ai set di dati corrispondenti esistenti. Supponiamo che desideri abbinare i dati dei potenziali clienti con i set di dati esistenti dei clienti. La funzionalità di corrispondenza incrementale offre la flessibilità necessaria per abbinare centinaia di migliaia di nuovi prospect con un database esistente di prospect e potenziali clienti combinando i risultati in un unico database o tabella. Corrispondendo

solo tra i set di dati nuovi ed esistenti, l'ottimizzazione delle corrispondenze incrementali di ricerca riduce i tempi di calcolo, riducendo anche i costi.

L'uso della corrispondenza incrementale è simile a Trova corrispondenze come descritto in Tutorial: creazione di una trasformazione basata su machine learning con AWS Glue. Questo argomento identifica solo le differenze con la corrispondenza incrementale.

Per ulteriori informazioni, leggi il post del blog su Corrispondenza incrementale dei dati.

Esecuzione di un processo di corrispondenza incrementale

Per la seguente procedura, supponiamo quanto segue:

- Hai eseguito il crawling del set di dati esistente nella tabella first_records. Il set di dati first_records deve essere un set di dati corrispondente o l'output del processo corrispondente.
- Hai creato e addestrato una trasformazione Find matches (Trova corrispondenze) con AWS Glue Glue versione 2.0. Questa è l'unica versione di AWS Glue che supporti le corrispondenze incrementali.
- Il linguaggio ETL è Scala. Si noti che anche Python è supportato.
- Il modello già generato viene chiamato `demo-xform`.

1. Esegui la scansione del set di dati incrementale nella tabella second_records.
2. Nel riquadro di navigazione della console di AWS Glue, scegliere Jobs (Processi).
3. Scegliere Add job (Aggiungi processo) e seguire la procedura guidata per creare un processo ETL Spark con uno script generato. Per le proprietà della trasformazione scegliere i seguenti valori:

   a. Per Name (Nome), scegli demo-etl.

   b. Alla voce IAM role (Ruolo IAM), scegli un ruolo IAM che disponga delle autorizzazioni per accedere ai dati di origine su Amazon S3, ai file di etichettatura dei dati e alle operazioni API di AWS Glue.

   c. Alla voce ETL language (Linguaggio ETL) scegli Scala.

   d. Come Script file name (Nome del file di script), scegli demo-etl. Questo è il nome del file dello script Scala.

   e. Per Data source (Origine dati), scegli first_records. L'origine dati scelta deve corrispondere allo schema dell'origine dati della trasformazione basata su machine learning.

f. Alla voce Transform type (Tipo di trasformazione), scegliere Find matching records (Individuazione record corrispondenti) per creare un processo che utilizza una trasformazione basata su machine learning.

g. Seleziona l'opzione di corrispondenza incrementale e per Data source (Origine dati) seleziona la tabella denominata second_records.

h. Alla voce Transform (Trasformazione), scegli demo-xform, la trasformazione basata su machine learning utilizzata del processo.

i. Scegli Create tables in your data target (Crea tabelle nella tua destinazione di dati) o Use tables in the catalogo dati and update your data target (Usa tabelle nel catalogo dati e aggiorna la destinazione dati).

4. Scegliere Save job and edit script (Salva processo e modifica script) per visualizzare la pagina dell'editor dello script.

5. Scegliere Run job (Esegui processo) per avviare l'esecuzione del processo.

Utilizzo di FindMatches in un processo visivo

Per utilizzare la trasformazione FindMatches in AWS Glue Studio, puoi utilizzare il nodo Trasformazione personalizzata che richiama l'API FindMatches. Per ulteriori informazioni su come utilizzare una trasformazione personalizzata, consulta la pagina Creating a custom transformation

> **Note**
>
> Attualmente, l'API FindMatches funziona solo con `Glue 2.0`. Per eseguire un processo con la trasformazione personalizzata che richiama l'API FindMatches, assicurati che la versione di AWS Glue sia `Glue 2.0` nella scheda Dettagli del processo. Se la versione di AWS Glue non è `Glue 2.0`, il processo avrà esito negativo in fase di esecuzione con il seguente messaggio di errore: "cannot import name 'FindMatches' from 'awsglueml.transforms'".

Prerequisiti

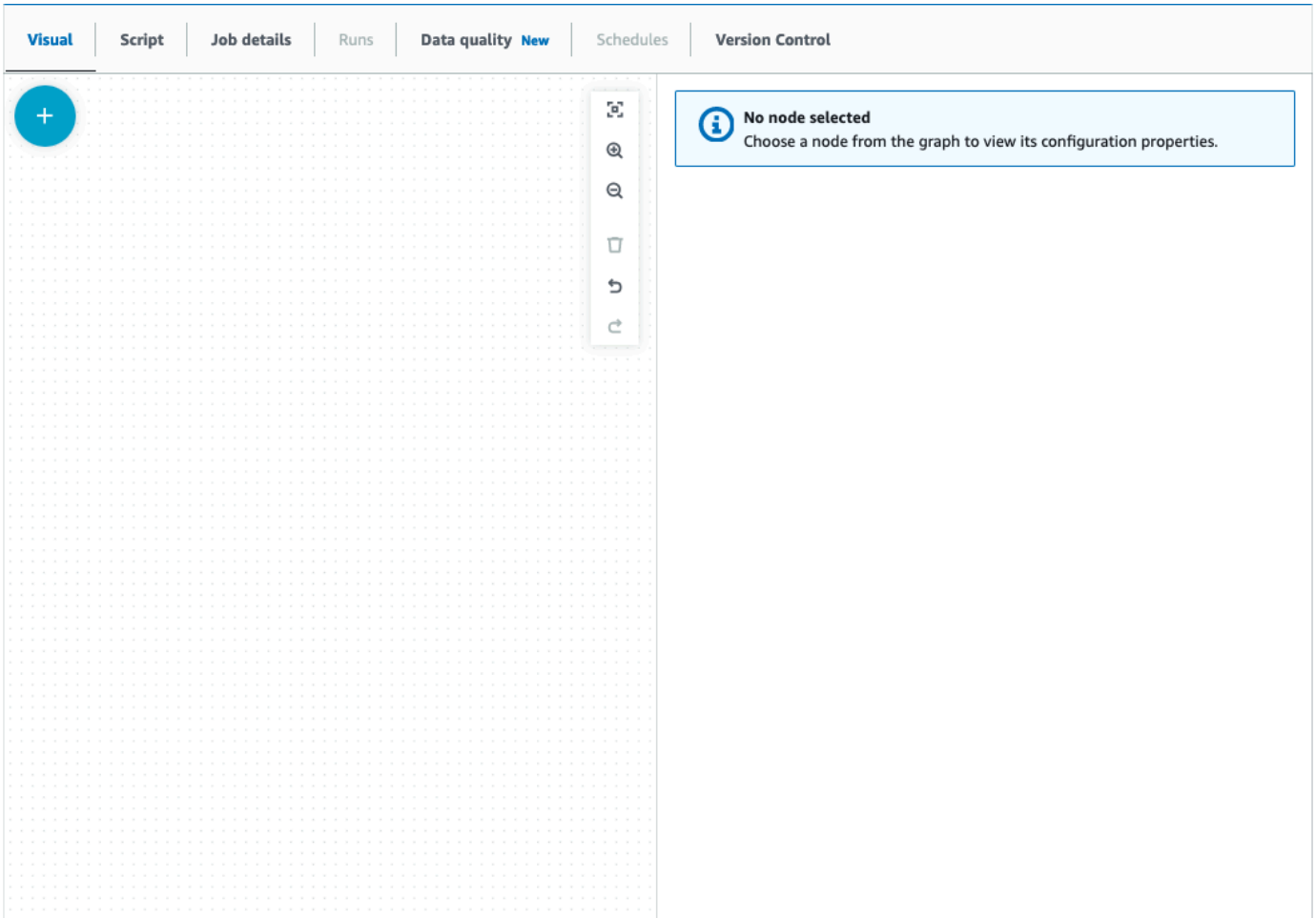
- Per utilizzare la trasformazione Trova corrispondenze, apri la console AWS Glue Studio all'indirizzo https://console.aws.amazon.com/gluestudio/.
- Crea una trasformazione basata su machine learning. Una volta creata, viene generato un transformId. Avrai bisogno di questo ID per i passaggi successivi. Per ulteriori informazioni su come

creare una trasformazione basata su machine learning, consulta la pagina Adding and editing machine learning transforms.

Aggiunta di una trasformazione FindMatches

Per aggiungere una trasformazione FindMatches:

1. Nell'editor dei processi AWS Glue Studio, apri il pannello Risorse facendo clic sul simbolo della croce nell'angolo in alto a sinistra del grafico visivo del processo e scegli un'origine dati selezionando la scheda Dati. Questa è l'origine dati nella quale verificare la presenza di corrispondenze.

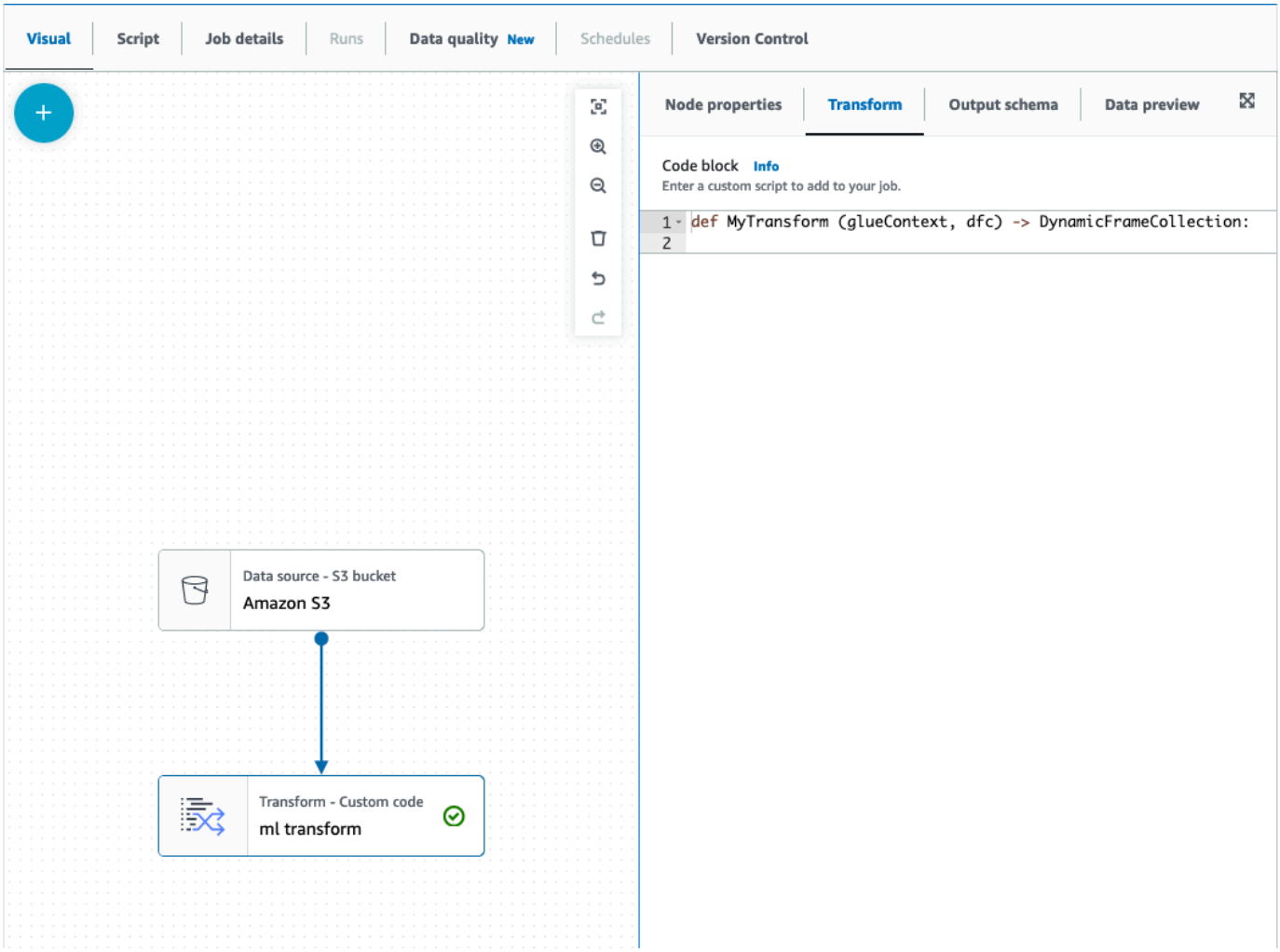


2. Scegli il nodo dell'origine dati, quindi apri il pannello Risorse facendo clic sul simbolo della croce nell'angolo in alto a sinistra del grafico visivo del processo e cerca "trasformazione personalizzata". Scegli il nodo Trasformazione personalizzata per aggiungerlo al grafico. La Trasformazione personalizzata è collegata al nodo dell'origine dati. In caso contrario, puoi fare

clic sul nodo Trasformazione personalizzata e scegliere la scheda Proprietà del nodo, quindi, in Padri del nodo, scegli l'origine dati.

3. Fai clic sul nodo Trasformazione personalizzata nel grafico visivo, quindi scegli la scheda Proprietà del nodo e assegna un nome alla trasformazione personalizzata. Ti consigliamo di rinominare la trasformazione assegnandole un nome facilmente identificabile nel grafico visivo.

4. Scegli la scheda Trasforma, dove puoi modificare il blocco di codice. Qui è possibile aggiungere il codice per richiamare l'API FindMatches.



Il blocco di codice contiene codice precompilato per aiutarti a iniziare. Sovrascrivi il codice precompilato con il modello seguente. Il modello ha un segnaposto per transformId, che puoi sostituire con il tuo valore.

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    dynf = dfc.select(list(dfc.keys())[0])
    from awsglueml.transforms import FindMatches
```

```
    findmatches = FindMatches.apply(frame = dynf, transformId = "<your id>")
    return(DynamicFrameCollection({"FindMatches": findmatches}, glueContext))
```

5. Fai clic sul nodo Trasformazione personalizzata nel grafico visivo, quindi apri il pannello Risorse facendo clic sul simbolo della croce nell'angolo in alto a sinistra del grafico visivo del processo e cerca "Seleziona dalla raccolta". Non è necessario modificare la selezione predefinita poiché nella raccolta è presente un solo DynamicFrame.
6. È possibile continuare ad aggiungere trasformazioni o archiviare il risultato, che ora è arricchito con le colonne aggiuntive delle corrispondenze trovate. Se vuoi fare riferimento a quelle nuove colonne nelle trasformazioni a valle, devi aggiungerle allo schema di output della trasformazione. Il modo più semplice per farlo è scegliere la scheda Anteprima dati e quindi, nella scheda Schema, scegliere "Utilizza schema di anteprima dati".
7. Per personalizzare FindMatches, puoi aggiungere parametri aggiuntivi da trasmettere al metodo "apply". Consulta la pagina FindMatches class.

Aggiunta di una trasformazione FindMatches in modo incrementale

Nel caso delle corrispondenze incrementali, il processo è lo stesso dell'aggiunta di una trasformazione FindMatches, ma presenta le seguenti differenze:

- Per la trasformazione personalizzata sono necessari due nodi padri anziché un solo nodo.
- Il primo nodo padre dovrebbe essere il set di dati.
- Il secondo nodo padre dovrebbe essere il set di dati incrementale.

Sostituisci il valore `transformId` con il tuo `transformId` nel blocco di codice del modello:

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    dfs = list(dfc.values())
    dynf = dfs[0]
    inc_dynf = dfs[1]
    from awsglueml.transforms import FindIncrementalMatches
    findmatches = FindIncrementalMatches.apply(existingFrame = dynf, incrementalFrame
 = inc_dynf,
                                    transformId = "<your id>")
    return(DynamicFrameCollection({"FindMatches": findmatches}, glueContext))
```

- Per i parametri opzionali, consulta la pagina FindIncrementalMatches class.

Migrare i programmi Apache Spark a AWS Glue

Apache Spark è una piattaforma open source per carichi di lavoro di calcolo distribuiti eseguiti su set di dati di grandi dimensioni. AWS Glue sfrutta le capacità di Spark per fornire un'esperienza ottimizzata per ETL. È possibile migrare i programmi Spark a AWS Glue per usufruire dei vantaggi delle nostre funzionalità. AWS Glue fornisce gli stessi miglioramenti delle prestazioni che ci si aspetterebbe da Apache Spark su Amazon EMR.

Esegui codice Spark

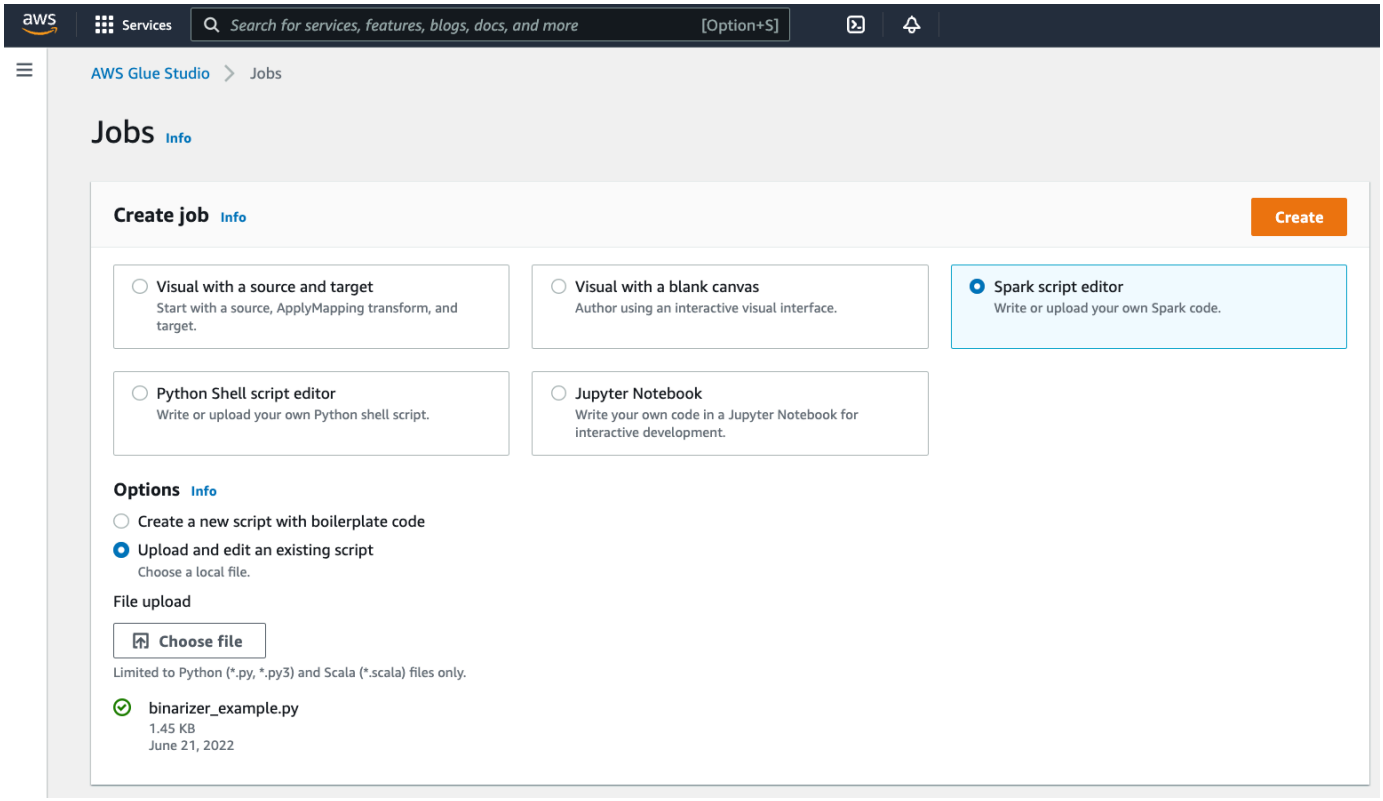
Il codice nativo di Spark può essere eseguito in ambiente AWS Glue così com'è. Gli script sono spesso sviluppati modificando iterativamente un pezzo di codice, un flusso di lavoro adatto per una sessione interattiva. Tuttavia, il codice esistente è più adatto per l'esecuzione in un processo AWS Glue, che consente di pianificare e ottenere costantemente registri e parametri per ogni esecuzione di script. Puoi caricare e modificare uno script esistente tramite la console.

1. Acquisisci la fonte del tuo script. Per questo esempio, si utilizzerà uno script di esempio dal repository Apache Spark. Esempio di Binarizer
2. Nella console AWS Glue, espandi il riquadro di navigazione a sinistra e seleziona ETL >Jobs (processi)

   Nel panello Create job (crea processo), seleziona Spark script editor (editor di script di Spark). Apparirà una sezione Options (opzioni). Sotto Options (opzioni) , seleziona Upload and edit an existing script (carica e modifica uno script esistente).

   Apparirà una sezione File upload (caricamento file). Sotto a File upload (caricamento file), fai clic su Choose file (seleziona file). Apparirà il tuo selettore di file di sistema. Accedi al percorso in cui hai effettuato il salvataggio di `binarizer_example.py`, selezionalo e conferma della selezione.

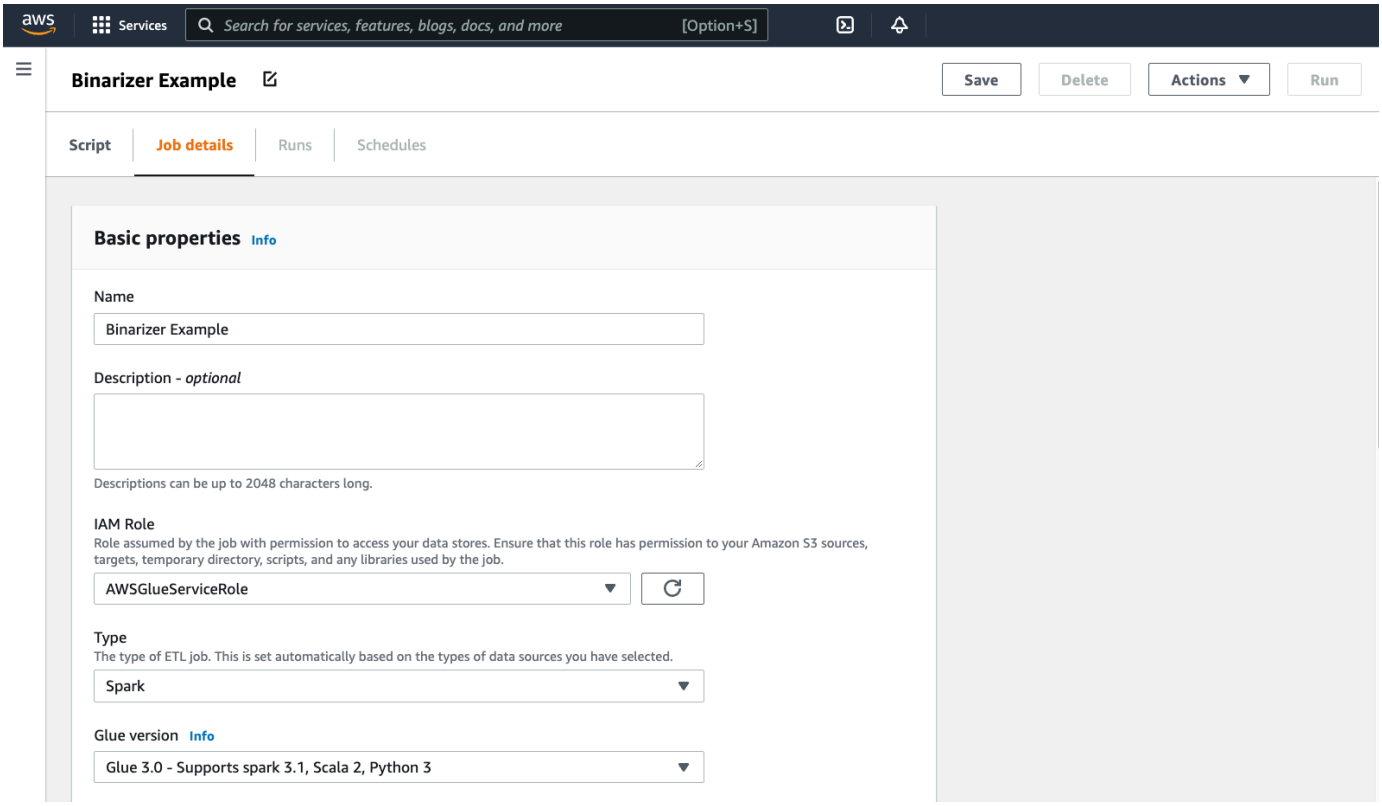
   Verrà visualizzato un pulsante Create (crea) nell'intestazione del pannello Create job (crea processo). Fai clic sul pulsante.



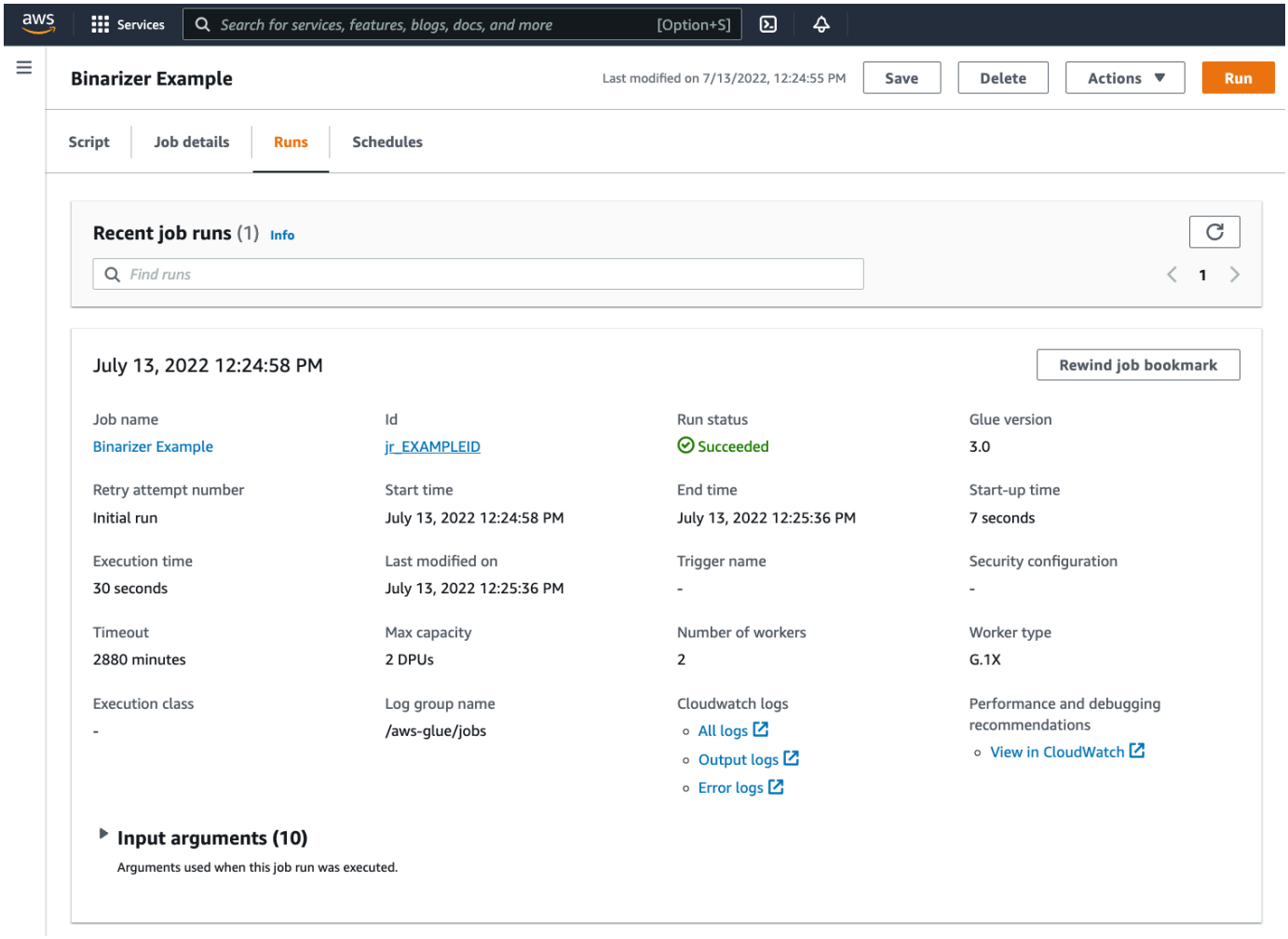
3. Il tuo browser accederà all'editor di script. Nell'intestazione, fai clic sul sulla scheda Job details (dettagli processo). Imposta il Nome e il Ruolo IAM. Per indicazioni sui ruoli AWS Glue IAM, consulta la pagina the section called “Impostazione delle autorizzazioni IAM”.

   Facoltativo: imposta Requested number of workers (numero di worker richiesti) su 2 e Number of retries (numero di tentativi) su 1. Queste opzioni sono utili quando si eseguono lavori di produzione, ma la loro riduzione semplificherà la tua esperienza durante il test di una funzionalità.

   Nella barra del titolo, fai clic su Save (salva), quindi Run (esegui)

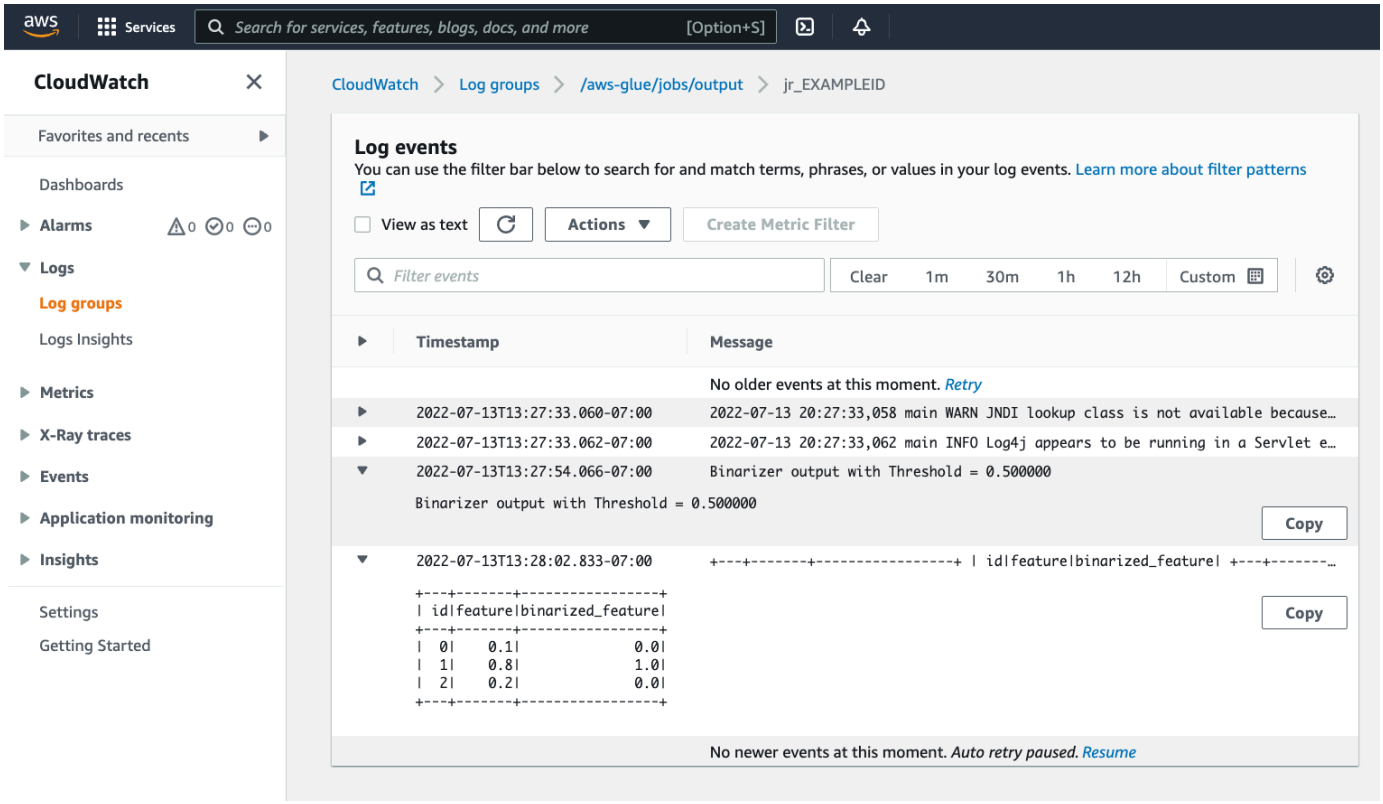


4. Passa alla scheda Runs (esecuzioni). Vedrai un pannello corrispondente all'esecuzione del processo. Attendi alcuni minuti e la pagina dovrebbe essere aggiornata automaticamente per mostrare Succeeded (elaborazione riuscita) sotto a Run status (stato dell'esecuzione).



5. Dovrai esaminare l'output per confermare che lo script Spark è stato eseguito come previsto. Questo script di esempio di Apache Spark dovrebbe scrivere una stringa nel flusso di output. Puoi trovarlo navigando su Output logs (registri di output) sotto a Cloudwatch logs(registri di Cloudwatch) nel pannello dell'esecuzione del processo riuscito. Ricorda l'id di esecuzione del processo, un id generato sotto l'etichetta Id che inizia con `jr_`.

   Si aprirà la console di CloudWatch, impostata per visualizzare i contenuti del gruppo di log AWS Glue di default `/aws-glue/jobs/output`, filtrato in base al contenuto dei flussi di log per l'id di esecuzione del processo. Ogni worker avrà generato un flusso di log, mostrato in righe sotto Log streams (Flussi di log). Un worker dovrebbe aver eseguito il codice richiesto. Sarà necessario aprire tutti i flussi di log per identificare il worker corretto. Una volta trovato il worker giusto, dovresti vedere l'output dello script, come mostrato nell'immagine seguente:



Procedure comuni necessarie per la migrazione dei programmi Spark

Convalida il supporto della versione di Spark

La versione di AWS Glue determina le versioni di Apache Spark e Python disponibili per il processo AWS Glue. Puoi trovare le nostre versioni AWS Glue e cosa supportano qui the section called “Versioni AWS Glue”. Potrebbe essere necessario aggiornare il programma Spark per renderlo compatibile con una versione più recente di Spark per accedere a determinate caratteristiche di AWS Glue.

Includi librerie di terze parti

Molti programmi Spark esistenti avranno dipendenze, sia da artefatti privati che pubblici. AWS Glue supporta le dipendenze in stile JAR per processi Scala così come le dipendenze Wheel e pure-Python per i processi Python.

Python : per informazioni sulle dipendenze Python, consulta the section called “Librerie Python”

Le dipendenze Python comuni sono fornite nell'ambiente AWS Glue, incluso la libreria Pandas. Le dipendenze sono incluse in AWS Glue versione 2.0+. Per ulteriori informazioni sui moduli disponibili,

consulta the section called “Moduli Python già forniti in AWS Glue”. Se è necessario fornire un processo con una versione diversa da quella di una dipendenza inclusa per impostazione predefinita, è possibile utilizzare `--additional-python-modules`. Per informazioni su questi argomenti, consulta the section called “Parametri del processo”.

È possibile fornire dipendenze Python aggiuntive con l’argomento del processo `--extra-py-files`. Se stai eseguendo la migrazione di un processo da un programma Spark, questo parametro rappresenta una buona opzione perché è funzionalmente equivalente al flag `--py-files` in PySpark ed è soggetto alle stesse limitazioni. Per ulteriori informazioni sul parametro `--extra-py-files`, consulta the section called “Inclusione di file Python con funzionalità native PySpark ”.

Per i nuovi processi, è possibile gestire le dipendenze Python con l’argomento del processo `--additional-python-modules`. L'utilizzo di questo argomento consente un'esperienza di gestione delle dipendenze più approfondita. Questo parametro supporta le dipendenze dello stile Wheel, incluse quelle con associazioni di codice native compatibili con Amazon Linux 2.

Scala

È possibile fornire dipendenze Scala aggiuntive con l’argomento del processo `--extra-jars`. Le dipendenze devono essere ospitate in Amazon S3 e il valore dell'argomento deve essere un elenco delimitato da virgole di percorsi Amazon S3 senza spazi. Potresti trovare più facile gestire la configurazione riaggregando le dipendenze prima di ospitarle e configurarle. AWS Glue Le dipendenze JAR contengono bytecode Java, che può essere generato da qualsiasi linguaggio JVM. È possibile utilizzare altri linguaggi JVM, come Java, per scrivere dipendenze personalizzate.

Gestisci le credenziali dell'origine dei dati.

I programmi Spark esistenti possono avere una configurazione complessa o personalizzata per estrarre dati dalle loro origini dati. I flussi di autenticazione delle origini dati comuni sono supportati da connessioni AWS Glue. Per ulteriori informazioni sulle connessioni AWS Glue, consulta Connessione ai dati.

Le connessioni AWS Glue facilitano la connessione del processo a una varietà di tipi di archivi di dati in due modi principali: attraverso chiamate di metodo alle nostre librerie e configurando la Connessione di rete aggiuntiva nella console AWS. Puoi anche chiamare l’SDK AWS dall'interno del processo per recuperare informazioni da una connessione.

Chiamate di metodo – Le connessioni AWS Glue sono strettamente integrate con il catalogo dati AWS Glue, un servizio che consente di gestire le informazioni sui set di dati e sui metodi disponibili per interagire con le connessioni AWS Glue che li riflettono. Se si dispone di una configurazione di

autenticazione esistente che si desidera riutilizzare, per le connessioni JDBC, è possibile accedere alla configurazione della connessione AWS Glue tramite il metodo `extract_jdbc_conf` su `GlueContext`. Per ulteriori informazioni, consulta the section called “extract_jdbc_conf”.

Configurazione della console - I processi AWS Glue utilizzano le connessioni AWS Glue associate per configurare le connessioni alle sottoreti Amazon VPC. Se gestisci i materiali di sicurezza, potrebbe rendersi necessario fornire una connessione di rete aggiuntiva di tipo `NETWORK` nella console AWS per configurare il routing. Per ulteriori informazioni sull'API di connessione AWS Glue, consulta the section called “Connessioni”.

Se i tuoi programmi Spark hanno un flusso di autenticazione personalizzato o non comune, potresti dover gestire i materiali di sicurezza in modo pratico. Se le connessioni AWS Glue non sembrano adatte, puoi ospitare in modo sicuro i materiali di sicurezza in Secrets Manager e accedervi tramite boto3 o SDK AWS, forniti nel processo.

Configurazione di Apache Spark

Le migrazioni complesse spesso alterano la configurazione di Spark per adattarsi ai loro carichi di lavoro. Le versioni moderne di Apache Spark consentono di impostare la configurazione del runtime con `SparkSession`.AWS Glue I processi 3.0+ vengono forniti a `SparkSession`, che può essere modificato per impostare la configurazione del runtime. Configurazione di Apache Spark. Regolare Spark è un'operazione complessa e AWS Glue non garantisce il supporto per l'impostazione di tutte le configurazioni necessarie. Se la migrazione richiede una configurazione sostanziale a livello Spark, contatta il supporto.

Impostazione della configurazione personalizzata

I programmi Spark migrati possono essere progettati per accettare una configurazione personalizzata. AWS Glue consente di impostare la configurazione a livello di processo ed esecuzione del processo, tramite gli argomenti. Per informazioni sugli argomenti di processo, consulta the section called “Parametri del processo”. Puoi accedere agli argomenti del processo nel contesto tramite le nostre librerie. AWS Glue fornisce una funzione utility per fornire una vista coerente tra gli argomenti impostati sul processo e gli argomenti impostati nell'esecuzione del processo. Consulta the section called “getResolvedOptions” in Python e the section called “GlueArgParser” in Scala.

Migrazione del codice Java

Come spiegato in the section called “Librerie di terze parti”, le dipendenze possono contenere classi generate da linguaggi JVM, come Java o Scala. Le dipendenze possono includere un metodo `main`.

Puoi utilizzare un metodo `main` in una dipendenza come punto di ingresso per un processo Scala AWS Glue. Questo permette di scrivere il proprio metodo `main` in Java o riutilizzare un metodo `main` elaborato in pacchetti secondo gli standard della tua libreria.

Per utilizzare un metodo `main` da una dipendenza, esegui le seguenti operazioni: cancella il contenuto del riquadro di modifica fornendo il valore predefinito dell'oggetto `GlueApp`. Fornisci il nome completo di una classe in una dipendenza come argomento di processo con la chiave`--class`. A questo punto dovresti essere in grado di attivare un'esecuzione del processo.

Non è possibile configurare l'ordine o la struttura degli argomenti che AWS Glue passa al metodo `main`. Se il codice esistente deve leggere la configurazione impostata in AWS Glue, questo probabilmente causerà incompatibilità con il codice precedente. Se utilizzi `getResolvedOptions`, non avrai nemmeno un luogo ideale per chiamare questo metodo. Prendi in considerazione la possibilità di invocare la tua dipendenza direttamente da un metodo principale generato da AWS Glue. Il seguente ETL AWS Glue mostra un esempio di ciò.

```
import com.amazonaws.services.glue.util.GlueArgParser

object GlueApp {
  def main(sysArgs: Array[String]) {
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)

    // Invoke static method from JAR. Pass some sample arguments as a String[], one
 defined inline and one taken from the job arguments, using getResolvedOptions
    com.mycompany.myproject.MyClass.myStaticPublicMethod(Array("string parameter1",
 args("JOB_NAME")))

    // Alternatively, invoke a non-static public method.
    (new com.mycompany.myproject.MyClass).someMethod()
  }
}
```

Utilizzo dei processi Ray in AWS Glue

Questa sezione fornisce informazioni sull'utilizzo di AWS Glue per i processi Ray. Per ulteriori informazioni sulla scrittura di script AWS Glue per Ray, consulta la sezione the section called “AWS Glue per Ray”.

Argomenti

- Ambienti di runtime Ray supportati
- Contabilità per i worker nei processi Ray
- Utilizzo dei parametri di processo nei processi Ray
- Monitoraggio dei processi di Ray con i parametri

Nozioni di base su AWS Glue per Ray

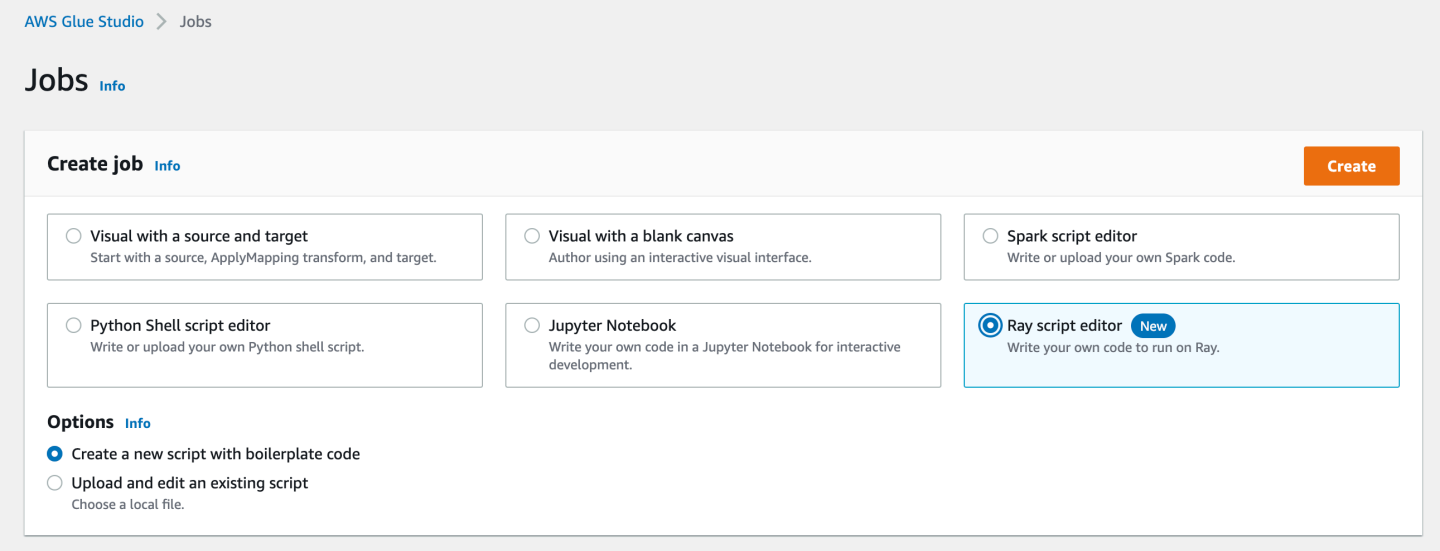
Per lavorare con AWS Glue per Ray, utilizzi le stesse funzionalità, i processi e le sessioni interattive di AWS Glue che useresti con AWS Glue per Spark. I processi AWS Glue sono progettati per eseguire lo stesso script a cadenza ricorrente, mentre le sessioni interattive sono progettate per consentire di eseguire frammenti di codice in sequenza sulle stesse risorse in provisioning.

AWS Glue ETL e Ray di sono diversi alla base, quindi nello script si avrà accesso a strumenti, funzionalità e configurazioni differenti. In quanto nuovo framework di calcolo gestito da AWS Glue, Ray ha un'architettura diversa e utilizza un vocabolario diverso per descrivere ciò che fa. Per ulteriori informazioni, consulta Whitepaper sull'architettura nella documentazione di Ray.

> **Note**
>
> AWS Glue per Ray è disponibile nelle Regioni Stati Uniti orientali (Virginia settentrionale), Stati Uniti orientali (Ohio), Stati Uniti occidentali (Oregon), Asia Pacifico (Tokyo) ed Europa (Irlanda).

Processi Ray nella console AWS Glue Studio

Nella pagina Processi della console AWS Glue Studio, puoi selezionare una nuova opzione quando crei un processo in AWS Glue Studio: Editor di script Ray. Scegli questa opzione per creare un processo Ray nella console. Per ulteriori informazioni sui processi e sul relativo utilizzo, consulta la pagina Creazione di processi ETL visivi con AWS Glue Studio.



Processi Ray nella AWS CLI e nell'SDK

I processi Ray nella AWS CLI utilizzano le stesse operazioni e parametri dell'SDK degli altri processi. AWS Glue per Ray introduce nuovi valori per determinati parametri. Per ulteriori informazioni sull'API Processi, consulta la pagina the section called “Processi”.

Ambienti di runtime Ray supportati

Nei processi Spark, `GlueVersion` determina le versioni di Apache Spark e Python disponibili in un processo AWS Glue per Spark. La versione di Python indica la versione supportata per i processi di tipo Spark. Questo non è il modo in cui sono configurati gli ambienti di runtime Ray.

Per i processi Ray, è necessario impostare `GlueVersion` su `4.0` o superiore. Tuttavia, le versioni di Ray, Python e le librerie aggiuntive disponibili nel processo Ray sono determinate dal campo `Runtime` nella definizione del processo.

L'ambiente di runtime `Ray2.4` sarà disponibile per un minimo di 6 mesi dopo il rilascio. Di pari passo con la rapida evoluzione di Ray, potrai incorporare aggiornamenti e miglioramenti di Ray nelle future versioni dell'ambiente di runtime.

Valori validi: `Ray2.4`

Valore di runtime	Versioni di Ray e Python
`Ray2.4` (per AWS Glue 4.0 e versioni successive)	Ray 2.4.0

Valore di runtime	Versioni di Ray e Python
	Python 3.9

Informazioni aggiuntive

- Per le note di rilascio che accompagnano le versioni di AWS Glue su Ray, consulta la pagina the section called “Versioni AWS Glue”.
- Per le librerie Python disponibili in un ambiente di runtime, consulta la pagina the section called “Moduli disponibili con i processi Ray”.

Contabilità per i worker nei processi Ray

AWS Glue esegue i processi Ray su nuovi tipi di worker EC2 basati su Graviton, disponibili solo per i processi Ray. Per fornire in modo appropriato questi worker per i carichi di lavoro per cui Ray è progettato, forniamo un rapporto diverso tra risorse di calcolo e risorse di memoria rispetto alla maggior parte dei worker. Per tenere conto di queste risorse, utilizziamo l'unità di elaborazione dati ottimizzata per la memoria (M-DPU) anziché l'unità di elaborazione dati standard (DPU).

- Una M-DPU corrisponde a 4 vCPU e 32 GB di memoria.
- Una DPU corrisponde a 4 vCPU e 16 GB di memoria. Le DPU vengono utilizzate per rendicontare le risorse disponibili in AWS Glue con i processi Spark e i worker corrispondenti.

I processi Ray attualmente hanno accesso a un tipo di worker, `Z.2X`. Il worker `Z.2X` esegue la mappatura su 2 M-DPU (8 vCPU, 64 GB di memoria) e dispone di 128 GB di spazio su disco. Una macchina `Z.2X` fornisce 8 worker Ray (uno per vCPU).

Il numero di M-DPU che è possibile utilizzare contemporaneamente in un account è soggetto a una quota di servizio. Per ulteriori informazioni sui limiti degli account AWS Glue, consulta Endpoint e quote di AWS Glue.

Nella definizione del processo si specifica il numero di nodi worker disponibili per un processo Ray con `--number-of-workers (NumberOfWorkers)`. Per ulteriori informazioni sui valori di Ray nell'API Processi, consulta la pagina the section called “Processi”.

È possibile specificare ulteriormente un numero minimo di worker che un processo Ray deve allocare con il parametro di processo `--min-workers`. Per ulteriori informazioni sui parametri di processo, consulta the section called “Riferimento”.

Utilizzo dei parametri di processo nei processi Ray

L'impostazione degli argomenti per i processi AWS Glue Ray è analoga a quella per i processi AWS Glue per Spark. Per ulteriori informazioni sull'API AWS Glue, consulta the section called “Processi”. È possibile configurare i processi AWS Glue Ray con i vari argomenti che sono riportati in questa sezione. È anche possibile fornire i propri argomenti.

È possibile configurare un processo tramite la console, nella scheda Job details (Dettagli del processo), sotto l'intestazione Job Parameters (Parametri del processo). Puoi inoltre configurare un processo tramite la AWS CLI impostando `DefaultArguments` su un processo o `Arguments` sull'esecuzione di un processo. Gli argomenti e i parametri dei processi predefiniti resteranno gli stessi nel processo anche dopo più esecuzioni.

Ad esempio, la seguente è la sintassi per l'esecuzione di un processo utilizzando `--arguments` per impostare un parametro speciale.

```
$ aws glue start-job-run --job-name "CSV to CSV" --arguments='--scriptLocation="s3://
my_glue/libraries/test_lib.py",--test-environment="true"'
```

Dopo aver impostato gli argomenti, è possibile accedere ai parametri di processo dall'interno del processo Ray tramite le variabili di ambiente. Questo ti consente di configurare il processo per ogni esecuzione. Il nome della variabile di ambiente sarà il nome dell'argomento del processo senza il prefisso `--`.

Ad esempio, nell'esempio precedente, i nomi delle variabili sarebbero `scriptLocation` e `test-environment`. Pertanto, l'argomento dovrebbe essere recuperato tramite i metodi disponibili nella libreria standard: `test_environment = os.environ.get('test-environment')`. Per ulteriori informazioni sull'accesso alle variabili di ambiente con Python, consulta la sezione OS module nella documentazione di Python.

Configurazione delle modalità di generazione dei log da parte dei processi Ray

Per impostazione predefinita, i processi Ray generano log e parametri che vengono inviati a CloudWatch e Amazon S3. È possibile utilizzare il parametro `--logging_configuration` per

modificare la modalità di generazione dei log; attualmente è possibile utilizzarlo per impedire ai processi Ray di generare vari tipi di log. Questo parametro accetta un oggetto JSON, le cui chiavi corrispondono ai log/comportamenti che desideri modificare. Supporta le seguenti chiavi:

- `CLOUDWATCH_METRICS`: configura delle serie di parametri di CloudWatch che possono essere utilizzate per visualizzare l'integrità del processo. Per ulteriori informazioni sui parametri, consulta the section called “Parametri dei processi Ray”.
- `CLOUDWATCH_LOGS`: configura i log di CloudWatch che forniscono dettagli a livello di applicazione Ray sullo stato di esecuzione del processo. Per ulteriori informazioni sui log, consulta the section called “Risoluzione degli errori relativi ai processi Ray”.
- `S3`: configura ciò che AWS Glue scrive in Amazon S3, principalmente informazioni simili ai log di CloudWatch ma sotto forma di file anziché flussi di log.

Per disabilitare un comportamento di registrazione di Ray, fornisci il valore `{\"IS_ENABLED\": \"False\"}`. Ad esempio, per disabilitare i parametri di CloudWatch e i log di CloudWatch, fornisci la seguente configurazione:

```
"--logging_configuration": "{\"CLOUDWATCH_METRICS\": {\"IS_ENABLED\": \"False\"},
 \"CLOUDWATCH_LOGS\": {\"IS_ENABLED\": \"False\"}}"
```

Riferimento

I processi Ray riconoscono i seguenti nomi di argomenti che possono essere utilizzati per configurare l'ambiente di script per i processi Ray e le esecuzioni di processo:

- `--logging_configuration`: viene utilizzato per interrompere la generazione di vari log creati dai processi Ray. Questi log vengono generati per impostazione predefinita su tutti i processi Ray. Formato: oggetto JSON con escape di stringhe. Per ulteriori informazioni, consulta the section called “Configurazione delle modalità di generazione dei log da parte dei processi Ray”.
- `--min-workers`: il numero minimo di nodi worker allocati a un processo Ray. Un nodo worker può eseguire più repliche, una per CPU virtuale. Formato: numero intero. Minimo: 0 Massimo: valore specificato in `--number-of-workers (NumberOfWorkers)` nella definizione di processo. Per ulteriori informazioni su come allocare adeguatamente i nodi worker, consulta la pagina the section called “Contabilità per i worker nei processi Ray”.
- `--object_spilling_config`: AWS Glue per Ray supporta l'utilizzo di Amazon S3 per estendere lo spazio disponibile per l'archivio di oggetti di Ray. Per abilitare questo comportamento, è possibile fornire a Ray un oggetto di configurazione JSON per il riversamento di oggetti con

questo parametro. Per ulteriori informazioni sulla configurazione del riversamento di oggetti in Ray, consulta la pagina Object Spilling nella documentazione di Ray. Formato: oggetto JSON.

AWS Glue per Ray supporta il riversamento simultaneo solo su disco o su Amazon S3. È possibile fornire più punti di riversamento, purché rispettino questa limitazione. In caso di riversamento su Amazon S3, sarà necessario aggiungere al processo anche le autorizzazioni IAM per questo bucket.

Quando si fornisce un oggetto JSON come configurazione con la CLI, è necessario fornirlo come stringa, specificando l'oggetto JSON con escape di stringa. Ad esempio, un valore di stringa per il riversamento su un percorso Amazon S3 apparirebbe come: `"{\"type\": \"smart_open\", \"params\": {\"uri\":\"s3path\"}}"`. In AWS Glue Studio, fornisci questo parametro come oggetto JSON senza formattazione aggiuntiva.

- `--object_store_memory_head`: la memoria allocata all'archivio di oggetti Plasma sul nodo principale di Ray. Questa istanza esegue i servizi di gestione dei cluster e le repliche dei worker. Il valore rappresenta una percentuale di memoria libera sull'istanza dopo un avvio a caldo. Questo parametro viene utilizzato per ottimizzare i carichi di lavoro che richiedono un uso intensivo della memoria: i valori predefiniti sono accettabili per la maggior parte dei casi d'uso. Formato: numero intero positivo. Minimo: 1. Massimo: 100

  Per ulteriori informazioni su Plasma, consulta L'archivio oggetti in memoria di Plasma nella documentazione di Ray.

- `--object_store_memory_worker`: la memoria allocata all'archivio di oggetti Plasma sui nodi worker di Ray. Queste istanze eseguono solo repliche worker. Il valore rappresenta una percentuale di memoria libera sull'istanza dopo un avvio a caldo. Questo parametro viene utilizzato per ottimizzare i carichi di lavoro che richiedono un uso intensivo della memoria: i valori predefiniti sono accettabili per la maggior parte dei casi d'uso. Formato: numero intero positivo. Minimo: 1. Massimo: 100

  Per ulteriori informazioni su Plasma, consulta L'archivio oggetti in memoria di Plasma nella documentazione di Ray.

- `--pip-install`: un set di pacchetti Python da installare. È possibile installare pacchetti da PyPI utilizzando questo argomento. Formato: elenco delimitato da virgole.

  Una voce del pacchetto PyPI sarà nel formato `package==version`, con il nome e la versione di PyPI del pacchetto di destinazione. Le voci usano la corrispondenza della versione Python per abbinare il pacchetto e la versione, come ==, non il singolo uguale a =. Esistono altri operatori di

corrispondenza delle versioni. Per ulteriori informazioni, consulta PEP 440 sul sito Web di Python. È inoltre possibile fornire moduli personalizzati con `--s3-py-modules`.

- `--s3-py-modules`: un set di percorsi Amazon S3 che ospitano le distribuzioni di moduli Python. Formato: elenco delimitato da virgole.

  Puoi utilizzarlo per distribuire i tuoi moduli al tuo processo di Ray. I moduli possono essere forniti anche da PyPI con `--pip-install`. A differenza di AWS Glue ETL, i moduli personalizzati non vengono impostati tramite pip, ma vengono trasmessi a Ray per la distribuzione. Per ulteriori informazioni, consulta the section called “Moduli Python aggiuntivi per i processi Ray”.
- `--working-dir`: un percorso verso un file .zip ospitato in Amazon S3 che contiene file da distribuire a tutti i nodi che eseguono il processo Ray. Formato: stringa. Per ulteriori informazioni, consulta the section called “Fornitura di file al processo Ray”.

Monitoraggio dei processi di Ray con i parametri

È possibile monitorare i processi Ray utilizzando AWS Glue Studio e Amazon CloudWatch. CloudWatch raccoglie ed elabora i parametri non elaborati da AWS Glue con Ray, rendendoli disponibili per l'analisi. Questi parametri vengono visualizzati nella console AWS Glue Studio, in modo da poter monitorare il processo durante l'esecuzione.

Per una panoramica generale su come monitorare AWS Glue, consulta the section called “Utilizzo di parametri di CloudWatch”. Per una panoramica generale su come utilizzare i parametri di CloudWatch pubblicati da AWS Glue, consulta the section called “Monitoraggio con CloudWatch”.

Monitoraggio dei processi Ray nella console AWS Glue

Nella pagina dei dettagli dell'esecuzione di un processo, nella sezione Dettagli esecuzione, è possibile visualizzare grafici aggregati incorporati che mostrano i parametri del processo disponibili. AWS Glue Studio invia i parametri del processo a CloudWatch per ciascuna esecuzione del processo. Con questi, è possibile creare un profilo del cluster e delle attività, nonché accedere a informazioni dettagliate su ciascun nodo.

Per ulteriori informazioni sui grafici di parametri disponibili, consulta the section called “Visualizzazione dei parametri di Amazon CloudWatch per l'esecuzione di un processo Ray”.

Panoramica dei parametri dei processi Ray in CloudWatch

Pubblichiamo i parametri di Ray quando in CloudWatch è abilitato il monitoraggio dettagliato. I parametri sono pubblicati nello spazio dei nomi di `Glue/Ray` CloudWatch.

- Parametri dell'istanza

  Pubblichiamo i parametri sull'utilizzo della CPU, della memoria e del disco delle istanze assegnate a un processo. Questi parametri sono identificati da funzionalità quali `ExecutorId`, `ExecutorType` e `host`. Questi parametri sono un sottoinsieme dei parametri standard degli agenti Linux CloudWatch. Puoi trovare le informazioni relative ai nomi e alle funzionalità dei parametri nella documentazione di CloudWatch. Per ulteriori informazioni, consulta Parametri raccolti dall'agente CloudWatch.
- Parametri del cluster Ray

  Inoltriamo i parametri dai processi Ray che eseguono lo script a questo spazio dei nomi, quindi ti trasmettiamo quelli più rilevanti per te. I parametri disponibili potrebbero differire in base alla versione di Ray. Per ulteriori informazioni sulla versione di Ray utilizzata dal processo, consulta the section called “Versioni AWS Glue”.

  Ray raccoglie i parametri a livello di istanza. Inoltre, fornisce parametri per le attività e il cluster. Per ulteriori informazioni sulla strategia dei parametri di base di Ray, consulta la pagina Metrics nella documentazione di Ray.

> **Note**
>
> Non pubblichiamo i parametri Ray nello spazio dei nomi `Glue/Job Metrics/`; questo viene utilizzato solo per i processi AWS Glue ETL.

Processi shell di Python in AWS Glue

Puoi utilizzare un processo shell di Python per eseguire gli script di Python come una shell in AWS Glue. Con un processo della shell Python puoi eseguire script compatibili con Python 2.7, Python 3.6 o Python 3.9.

Non è possibile utilizzare i segnalibri nei processi di shell di Python.

L'output del gruppo Amazon CloudWatch Logs per i job della shell Python è. `/aws-glue/python-jobs/output` Per gli errori, consulta il gruppo di log `/aws-glue/python-jobs/error`.

Argomenti

- Definire le proprietà del processo per i processi shell di Python

- Librerie supportate dai processi shell di Python
- Limitazioni
- Fornire la propria libreria Python

Definire le proprietà del processo per i processi shell di Python

In queste sezioni sono descritte le proprietà dei processi di definizione in AWS Glue Studio oppure tramite AWSCLI.

AWS Glue Studio

Quando definisci il processo della shell Python in AWS Glue Studio, fornisci alcune delle seguenti proprietà:

Ruolo IAM

Specifica il ruolo AWS Identity and Access Management (IAM) necessario per l'autorizzazione alle risorse usate per eseguire il processo e accedere agli archivi dati. Per ulteriori informazioni sulle autorizzazioni per l'esecuzione di processi in AWS Glue, consulta Gestione delle identità e degli accessi per AWS Glue.

Type

Scegli Python shell (Shell di Python) per eseguire uno script Python con il comando di processo denominato `pythonshell`.

Versione di Python

Scegli la versione di Python. La versione predefinita è Python 3.9. Le versioni valide sono Python 3.6 e Python 3.9.

Carica librerie di analisi comuni (scelta consigliata)

Scegli questa opzione per includere le librerie comuni per Python 3.9 nella shell Python.

Se le tue librerie sono personalizzate o sono in conflitto con quelle preinstallate, puoi scegliere di non installare librerie comuni. Tuttavia, oltre alle librerie comuni puoi installare librerie aggiuntive.

Quando selezioni questa opzione, l'opzione `library-set` è impostata su `analytics`. Quando deselezioni questa opzione, l'opzione `library-set` è impostata su `none`.

Nome file e percorso dello script

Il codice nello script definisce la logica procedurale del processo. Puoi fornire il nome e la posizione dello script in Amazon Simple Storage Service (Amazon S3). Conferma che non esiste un file con lo stesso nome della directory di script nel percorso. Per ulteriori informazioni sull'uso degli script, consulta AWS Glue guida alla programmazione.

Script

Il codice nello script definisce la logica procedurale del processo. Puoi codificare lo script in Python 3.6 o Python 3.9. Puoi modificare uno script in AWS Glue Studio.

Unità di elaborazione dati (DPU)

Il numero massimo di unità di elaborazione dei dati AWS Glue (DPU) che può essere allocato quando viene eseguito il processo. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta Prezzi di AWS Glue.

Puoi impostare il valore su 0,0625 o 1. Il valore predefinito è 0.0625. In entrambi i casi, il disco locale per l'istanza sarà di 20 GB.

CLI

Puoi anche creare un processo Python shell (Shell Python) utilizzando la AWS CLI, come nell'esempio seguente.

```
aws glue create-job --name python-job-cli --role Glue_DefaultRole
    --command '{"Name" :  "pythonshell", "PythonVersion": "3.9", "ScriptLocation" :
"s3://DOC-EXAMPLE-BUCKET/scriptname.py"}'
    --max-capacity 0.0625
```

Processi creati con l'impostazione predefinita di AWS CLI in Python 3. Le versioni valide di Python sono 3 (corrispondenti a 3.6) e 3.9. Per specificare Python 3.6, aggiungi questa tupla al parametro `--command`: `"PythonVersion":"3"`

Per specificare Python 3.9, aggiungi questa tupla al parametro `--command`: `"PythonVersion":"3.9"`

Per impostare la capacità massima utilizzata da un processo shell di Python, fornire il parametro `--max-capacity`. Per i processi di shell di Python non è possibile utilizzare il parametro `--allocated-capacity`.

Librerie supportate dai processi shell di Python

Nella shell Python con Python 3.9 puoi scegliere il set di librerie per utilizzare set di librerie preconfezionati per le tue esigenze. Puoi utilizzare l'opzione `library-set` per scegliere il set di librerie. I valori validi sono `analytics` e `none`.

L'ambiente per l'esecuzione di un processo shell di Python supporta le seguenti librerie:

Versione di Python	Python 3.6	Python 3.9	
Set di librerie	N/D	analytics	nessuno
avro		1.11.0	
awscli	116,242	1,23,5	1,23,5
awswrangler		2,15,1	
botocore	1,1,232	1,23,5	1,23,5
boto3	1,9203	1,22,5	
elasticsearch		8.2.0	
numpy	1,16,2	1.22.3	
pandas	0,24,2	1.4.2	
psycopg2		2,9,3	
pyathena		2.5.3	
PyGreSQL	5.0.6		
PyMySQL		1.0.2	
pyodbc		4.0.32	

Versione di Python	Python 3.6	Python 3.9	
pyorc		0,6,0	
redshift-connector		2.0,907	
richieste	2.22.0	2,27,1	
scikit-learn	0,20,3	1.0.2	
scipy	1.2.1	1.8.0	
SQLAlchemy		1,4,36	
s3fs		2022,3,0	

Puoi utilizzare la libreria NumPy in un processo shell di Python per il calcolo scientifico. Per ulteriori informazioni, vedere. NumPy L'esempio seguente mostra uno NumPy script che può essere usato in un job della shell Python. Lo script visualizza "Hello world" e i risultati di numerosi calcoli matematici.

```
import numpy as np
print("Hello world")

a = np.array([20,30,40,50])
print(a)

b = np.arange( 4 )

print(b)

c = a-b

print(c)

d = b**2

print(d)
```

Limitazioni

Tieni presente le seguenti limitazioni dei processi shell Python:

- Non puoi impacchettare alcuna libreria Python come `.egg` file in Python 3.9+. Utilizza invece `.whl`.
- Per via di una limitazione sulle copie temporanee dei dati di S3, l'opzione `--extra-files` non può essere utilizzata.

Fornire la propria libreria Python

Utilizzo di PIP

La shell Python che utilizza Python 3.9 consente di fornire moduli Python aggiuntivi o versioni diverse a livello di processo. Puoi utilizzare l'opzione `--additional-python-modules` con un elenco di moduli Python separati da virgole per aggiungere un nuovo modulo o modificare la versione di un modulo esistente. Non è possibile fornire moduli Python personalizzati ospitati su Amazon S3 con questo parametro quando si utilizzano processi di shell Python.

Ad esempio, per aggiornare o aggiungere un nuovo modulo `scikit-learn` usa la seguente coppia di chiave-valore: `"--additional-python-modules", "scikit-learn==0.21.3"`.

AWS Glue utilizza Python Package Installer (pip3) per installare i moduli aggiuntivi. Puoi passare opzioni pip3 aggiuntive all'interno del valore di `--additional-python-modules`. Ad esempio, `"scikit-learn==0.21.3 -i https://pypi.python.org/simple/"`. Si applicano eventuali incompatibilità o limitazioni da pip3.

> **Note**
>
> Per evitare incompatibilità in futuro, si consiglia di utilizzare le librerie create per Python 3.9.

Utilizzo di un file Egg o Whl

È possibile che uno o più pacchetti di librerie Python siano disponibili come un file `.whl` o `.egg`. In questo caso, puoi specificarli nel tuo processo utilizzando AWS Command Line Interface (AWS CLI) sotto il flag `"--extra-py-files"`, come mostrato nell'esempio seguente.

```
aws glue create-job --name python-redshift-test-cli --role role --command '{"Name" :
 "pythonshell", "ScriptLocation" : "s3://MyBucket/python/library/redshift_test.py"}'
```

```
    --connections Connections=connection-name --default-arguments '{"--extra-py-
files" : ["s3://DOC-EXAMPLE-BUCKET/EGG-FILE", "s3://DOC-EXAMPLE-BUCKET/WHEEL-FILE"]}'
```

In caso di dubbi su come creare un file `.egg` o `.whl` da una libreria Python, utilizza la procedura seguente. Questo esempio si applica su sistemi macOS, Linux e Windows Subsystem for Linux (WSL).

Per creare un file .egg o .whl Python

1. Crea un cluster Amazon Redshift in un cloud privato virtuale (VPC, Virtual Private Cloud) e aggiungi alcuni dati a una tabella.
2. Crea una AWS Glue connessione per la combinazione VPC- SecurityGroup -Subnet utilizzata per creare il cluster. Verifica che la connessione funzioni.
3. Crea una directory denominata `redshift_example` e crea un file denominato `setup.py`. Incollare il codice seguente in `setup.py`.

   ```
   from setuptools import setup

   setup(
       name="redshift_module",
       version="0.1",
       packages=['redshift_module']
   )
   ```
4. Nella directory `redshift_example` crea una directory `redshift_module`. Nella directory `redshift_module` crea i file `__init__.py` e `pygresql_redshift_common.py`.
5. Lascia il file `__init__.py` vuoto. Incolla il codice seguente in `pygresql_redshift_common.py`. Sostituisci *`port`*, *`db_name`*, *`user`* e *`password_for_user`* con i dettagli specifici del cluster Amazon Redshift. Sostituisci *`table_name`* con il nome della tabella in Amazon Redshift.

   ```
   import pg


   def get_connection(host):
       rs_conn_string = "host=%s port=%s dbname=%s user=%s password=%s" % (
           host, port, db_name, user, password_for_user)

       rs_conn = pg.connect(dbname=rs_conn_string)
       rs_conn.query("set statement_timeout = 1200000")
   ```

```
    return rs_conn


def query(con):
    statement = "Select * from table_name;"
    res = con.query(statement)
    return res
```

6. Se non sei ancora in tale directory, passa alla directory `redshift_example`.
7. Esegui una di queste operazioni:

   - Per creare un file `.egg`, esegui il comando seguente.

     ```
     python setup.py bdist_egg
     ```

   - Per creare un file `.whl`, esegui il comando seguente.

     ```
     python setup.py bdist_wheel
     ```

8. Installa le dipendenze necessarie per il comando precedente.
9. Il comando crea un file nella directory `dist`:

   - Se hai creato un file egg, viene denominato `redshift_module-0.1-py2.7.egg`.
   - Se hai creato un file wheel, viene denominato `redshift_module-0.1-py2.7-none-any.whl`.

   Carica questo file in Amazon S3.

   In questo esempio, il percorso del file caricato è *s3://DOC-EXAMPLE-BUCKET/EGG-FILE* o *s3://DOC-EXAMPLE-BUCKET/WHEEL-FILE*.

10. Crea un file Python da utilizzare come script per il processo AWS Glue e aggiungi al file il codice seguente.

    ```
    from redshift_module import pygresql_redshift_common as rs_common

    con1 = rs_common.get_connection(redshift_endpoint)
    res = rs_common.query(con1)

    print "Rows in the table cities are: "
    ```

```
print res
```

11. Carica il file precedente in Amazon S3. In questo esempio, il percorso del file caricato è *s3://DOC-EXAMPLE-BUCKET/scriptname.py*.
12. Crea un processo shell di Python utilizzando questo script. Nella console AWS Glue, nella pagina Job properties (Proprietà processo), specifica il percorso al file `.egg/.whl` nella casella Python library path (Percorso libreria Python). Se sono presenti più file `.egg/.whl` e file Python, occorre fornire un elenco separato da virgole in questa casella.

    Quando si modificano o si rinominano i file `.egg`, i nomi dei file devono utilizzare i nomi predefiniti generati dal comando "python setup.py bdist_egg" o devono rispettare le convenzioni di denominazione del modulo Python. Per ulteriori informazioni, vedere la Guida di stile per il codice Python.

    Utilizzando AWS CLI, crea un processo con un comando, come nell'esempio seguente.

    ```
    aws glue create-job --name python-redshift-test-cli --role Role --command
     '{"Name" :  "pythonshell", "ScriptLocation" : "s3://DOC-EXAMPLE-BUCKET/
    scriptname.py"}'
         --connections Connections="connection-name" --default-arguments '{"--extra-
    py-files" : ["s3://DOC-EXAMPLE-BUCKET/EGG-FILE", "s3://DOC-EXAMPLE-BUCKET/WHEEL-
    FILE"]}'
    ```

    Quando il processo è in esecuzione, lo script visualizza le righe create nella tabella *table_name* nel cluster Amazon Redshift.

Monitoraggio di AWS Glue

Il monitoraggio è importante per mantenere l'affidabilità, la disponibilità e le prestazioni di AWS Glue e delle altre soluzioni AWS. AWS fornisce strumenti di monitoraggio che puoi usare per controllare AWS Glue, segnalare eventuali problemi e intervenire automaticamente quando appropriato:

Per controllare AWS Glue e segnalare l'eventuale presenza di problemi, puoi usare gli strumenti di monitoraggio automatici seguenti:

- Amazon CloudWatch Events eroga un flusso quasi in tempo reale di eventi di sistema che descrivono le modifiche alle risorse AWS. CloudWatch Events consente il calcolo automatizzato

basato sugli eventi. Puoi scrivere le regole che controllano determinati eventi e attivano operazioni automatiche in altri servizi AWS quando questi eventi si verificano. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon CloudWatch Events.

- Amazon CloudWatch Logs consente di monitorare, archiviare e accedere ai file di log dalle istanze Amazon EC2, AWS CloudTrail e da altre origini. CloudWatch Logs è in grado di monitorare le informazioni nei file di log e notificare quando vengono raggiunte determinate soglie. Puoi inoltre archiviare i dati del log in storage estremamente durevole. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon CloudWatch Logs.
- AWS CloudTrail acquisisce le chiamate API e gli eventi correlati effettuati da o per conto del tuo account AWS e fornisce i file di log a un bucket Amazon S3 specificato. Puoi identificare quali utenti e account chiamano AWS, l'indirizzo IP di origine da cui vengono effettuate le chiamate e quando sono avvenute le chiamate. Per ulteriori informazioni, consulta la Guida per l'utente di AWS CloudTrail.

Inoltre, hai accesso agli approfondimenti seguenti nella console AWS Glue per aiutarti a eseguire il debug e per i processi di profilo:

- Processi Spark: puoi osservare una visualizzazione delle serie di parametri di CloudWatch selezionati, inoltre i processi più recenti hanno accesso all'interfaccia utente di Spark. Per ulteriori informazioni, consulta the section called “Monitoraggio dei processi Spark”.
- Processi Ray: puoi osservare una visualizzazione delle serie di parametri di CloudWatch selezionati. Per ulteriori informazioni, consulta the section called “Parametri dei processi Ray”.

Argomenti

AWS tag in AWS Glue

Per facilitare la gestione delle risorse AWS Glue puoi facoltativamente assegnare i tag ad alcuni tipi di risorse AWS Glue. Un tag è un'etichetta che si assegna a una AWS risorsa. Ogni tag è composto da una chiave e da un valore opzionale, entrambi personalizzabili. Puoi usare i tag in AWS Glue per organizzare e identificare le risorse. I tag possono essere utilizzati per creare

report di contabilità dei costi e limitare l'accesso alle risorse. Se lo utilizzi AWS Identity and Access Management, puoi controllare quali utenti del tuo AWS account sono autorizzati a creare, modificare o eliminare i tag. Oltre alle autorizzazioni per chiamare le API relative ai tag, è necessaria anche l'autorizzazione `glue:GetConnection`, per chiamare le API di applicazione di tag sulle connessioni, e l'autorizzazione `glue:GetDatabase`, per chiamare le API di applicazione di tag sui database. Per ulteriori informazioni, consulta ABAC con AWS Glue.

In AWS Glue è possibile applicare tag alle seguenti risorse:

- Connessione
- Database
- Crawler
- Sessione interattiva
- Endpoint di sviluppo
- Processo
- Trigger
- Flusso di lavoro
- Piano
- Trasformazione basata su machine learning
- Set di regole sulla qualità dei dati
- Schemi di flussi di dati
- Registri degli schemi di flussi di dati

> **Note**
>
> Come best practice, per consentire il tagging di queste risorse AWS Glue, includi sempre l'operazione `glue:TagResource` nelle policy.

Tieni in considerazione le informazioni seguenti quando usi i tag con AWS Glue.

- Il numero massimo di tag supportati per entità è 50.
- In AWS Glue, i tag vengono specificati come elenco di coppie chiave-valore nel formato `{"string": "string" ...}`

- Quando crei un tag su un oggetto, la chiave di tag è obbligatoria e il valore di tag è facoltativo.
- La chiave di tag e il valore di tag fanno distinzione tra maiuscole e minuscole.
- La chiave di tag e il valore di tag non devono contenere il prefisso aws. Non sono consentite operazioni su questi tag.
- La lunghezza massima delle chiavi di tag è 128 caratteri Unicode in UTF-8. La chiave di tag non deve essere vuota o nulla.
- Il valore massimo dei tag è 256 caratteri Unicode in UTF-8. Il valore di tag può essere vuoto o nullo.

Supporto per AWS Glue l'etichettatura delle connessioni

È possibile limitare le autorizzazioni alle operazioni `CreateConnection`, `UpdateConnection`, `GetConnection` e `DeleteConnection` basate sull'assegnazione di tag alla risorsa. Ciò consente di implementare il controllo degli accessi con privilegi minimi sui AWS Glue lavori con origini dati JDBC che devono recuperare le informazioni di connessione JDBC dal Data Catalog.

Esempio di utilizzo

Crea una AWS Glue connessione con il tag ["connection-category», «dev-test"].

Specifica la condizione del tag per l'operazione `GetConnection` nella policy IAM.

```
{
   "Effect": "Allow",
   "Action": [
     "glue:GetConnection"
     ],
   "Resource": "*",
   "Condition": {
     "ForAnyValue:StringEquals": {
       "aws:ResourceTag/tagKey": "dev-test"
     }
   }
 }
```

Esempi

I seguenti esempi creano un processo con tag assegnati.

AWS CLI

```
aws glue create-job --name job-test-tags --role MyJobRole --command
 Name=glueetl,ScriptLocation=S3://aws-glue-scripts//prod-job1
--tags key1=value1,key2=value2
```

AWS CloudFormation JSON

```
{
  "Description": "AWS Glue Job Test Tags",
  "Resources": {
    "MyJobRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "glue.amazonaws.com"
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
        "Policies": [
          {
            "PolicyName": "root",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
                }
              ]
            }
          }
```

```
        ]
      }
    },
    "MyJob": {
      "Type": "AWS::Glue::Job",
      "Properties": {
        "Command": {
          "Name": "glueetl",
          "ScriptLocation": "s3://aws-glue-scripts//prod-job1"
        },
        "DefaultArguments": {
          "--job-bookmark-option": "job-bookmark-enable"
        },
        "ExecutionProperty": {
          "MaxConcurrentRuns": 2
        },
        "MaxRetries": 0,
        "Name": "cf-job1",
        "Role": {
           "Ref": "MyJobRole",
           "Tags": {
                "key1": "value1",
                "key2": "value2"
           }
        }
      }
    }
  }
}
```

AWS CloudFormation YAML

```
Description: AWS Glue Job Test Tags
Resources:
  MyJobRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
```

```
                - glue.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: "/"
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action: "*"
                Resource: "*"
  MyJob:
    Type: AWS::Glue::Job
    Properties:
      Command:
        Name: glueetl
        ScriptLocation: s3://aws-glue-scripts//prod-job1
      DefaultArguments:
        "--job-bookmark-option": job-bookmark-enable
      ExecutionProperty:
        MaxConcurrentRuns: 2
      MaxRetries: 0
      Name: cf-job1
      Role:
        Ref: MyJobRole
        Tags:
           key1: value1
           key2: value2
```

Per ulteriori informazioni, consulta Strategie di tagging diAWS.

Per informazioni su come controllare l'accesso tramite i tag, consulta ABAC con AWS Glue.

Automazione di AWS Glue con CloudWatch Events

Puoi usare Amazon CloudWatch Events per automatizzare i servizi AWS e rispondere automaticamente a eventi di sistema, come i problemi relativi alla disponibilità delle applicazioni o le modifiche delle risorse. Gli eventi dei servizi AWS vengono recapitati a CloudWatch Events quasi in tempo reale. 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. Le azioni che possono essere attivate automaticamente includono le seguenti:

- Richiamo di una funzione AWS Lambda
- Richiamo del comando di esecuzione di Amazon EC2
- Inoltro dell'evento a Amazon Kinesis Data Streams
- Attivazione di una macchina a stati AWS Step Functions
- Notifica di un argomento Amazon SNS o di una coda Amazon SQS

Alcuni esempi di utilizzo di CloudWatch Events con AWS Glue includono i seguenti:

- Attivazione di una funzione Lambda in caso di esito positivo di un processo ETL
- Notifica di un argomento Amazon SNS quando un processo ETL ha esito negativo

I seguenti eventi CloudWatch Events sono generati da AWS Glue.

- Gli eventi per `"detail-type":"Glue Job State Change"` vengono generati per `SUCCEEDED`, `FAILED`, `TIMEOUT` e `STOPPED`.
- Eventi per `"detail-type":"Glue Job Run Status"` vengono generati per l'esecuzione dei processi `RUNNING`, `STARTING`e `STOPPING` quando superano la soglia di notifica di ritardo del processo. È necessario impostare la proprietà della soglia di notifica del ritardo del processo per ricevere questi eventi.

  Quando viene superata la soglia di notifica del ritardo del processo, viene generato un solo evento per ciascuno stato di esecuzione del processo.
- Eventi per `"detail-type":"Glue Crawler State Change"` vengono generati per `Started`, `Succeeded` e `Failed`.
- Gli eventi per `"detail-type":"Glue Data Catalog Database State Change"` vengono generati per `CreateDatabase`, `DeleteDatabase`, `CreateTable`, `DeleteTable` e `BatchDeleteTable`. Ad esempio, se una tabella viene creata o eliminata, viene inviata una notifica a CloudWatch Events. Si noti che non è possibile scrivere un programma che dipende dall'ordine o dall'esistenza di eventi di notifica, poiché potrebbero essere fuori sequenza o mancanti. Gli eventi vengono emessi nel miglior modo possibile. Nei dettagli della notifica:
  - `typeOfChange` contiene il nome dell'operazione API.
  - `databaseName` contiene il nome del database interessato.
  - `changedTables` contiene fino a 100 nomi di tabelle interessate per ogni notifica. Quando i nomi di tabella sono lunghi, potrebbero essere create più notifiche.

- Gli eventi per `"detail-type":"Glue Data Catalog Table State Change"` vengono generati per `UpdateTable`, `CreatePartition`, `BatchCreatePartition`, `UpdatePartition`, `DeletePartition`, `BatchUpdatePartition` e `BatchDeletePartition`. Ad esempio, se una tabella o una partizione viene aggiornata, viene inviata una notifica a CloudWatch Events. Si noti che non è possibile scrivere un programma che dipende dall'ordine o dall'esistenza di eventi di notifica, poiché potrebbero essere fuori sequenza o mancanti. Gli eventi vengono emessi nel miglior modo possibile. Nei dettagli della notifica:
  - `typeOfChange` contiene il nome dell'operazione API.
  - `databaseName` contiene il nome del database contenente le risorse interessate.
  - `tableName` contiene il nome della tabella interessata.
  - `changedPartitions` specifica fino a 100 partizioni interessate in una notifica. Quando i nomi di partizione sono lunghi, potrebbero essere create più notifiche.

    Ad esempio, se ci sono due chiavi di partizione, `Year` e `Month`, `"2018,01", "2018,02"` modifica la partizione dove `"Year=2018" and "Month=01"` e la partizione dove `"Year=2018" and "Month=02"`.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Glue Data Catalog Table State Change",
    "source":"aws.glue",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":["arn:aws:glue:us-west-2:123456789012:database/default/foo"],
    "detail":{
        "changedPartitions": [
            "2018,01",
            "2018,02"
        ],
        "databaseName": "default",
        "tableName": "foo",
        "typeOfChange": "BatchCreatePartition"
        }
}
```

Per ulteriori informazioni, consulta la Guida per l'utente di Amazon CloudWatch Events. Per eventi specifici di AWS Glue, consulta Eventi AWS Glue.

Monitoraggio delle risorse di AWS Glue

AWS Glue prevede dei limiti di servizio per proteggere i clienti da forniture eccessive e impreviste e da operazioni dannose volte ad aumentare le spese. I limiti proteggono anche il servizio. Accedendo alla console AWS Service Quotas, i clienti possono visualizzare i limiti attuali delle risorse e richiederne un aumento, se necessario.

AWS Glue consente di visualizzare l'utilizzo delle risorse del servizio come percentuale in Amazon CloudWatch e di configurare allarmi CloudWatch dedicati per monitorare l'utilizzo. Amazon CloudWatch fornisce il monitoraggio delle risorse AWS e delle applicazioni dei clienti in esecuzione sull'infrastruttura Amazon. I parametri sono gratuiti. Sono supportati i parametri seguenti:

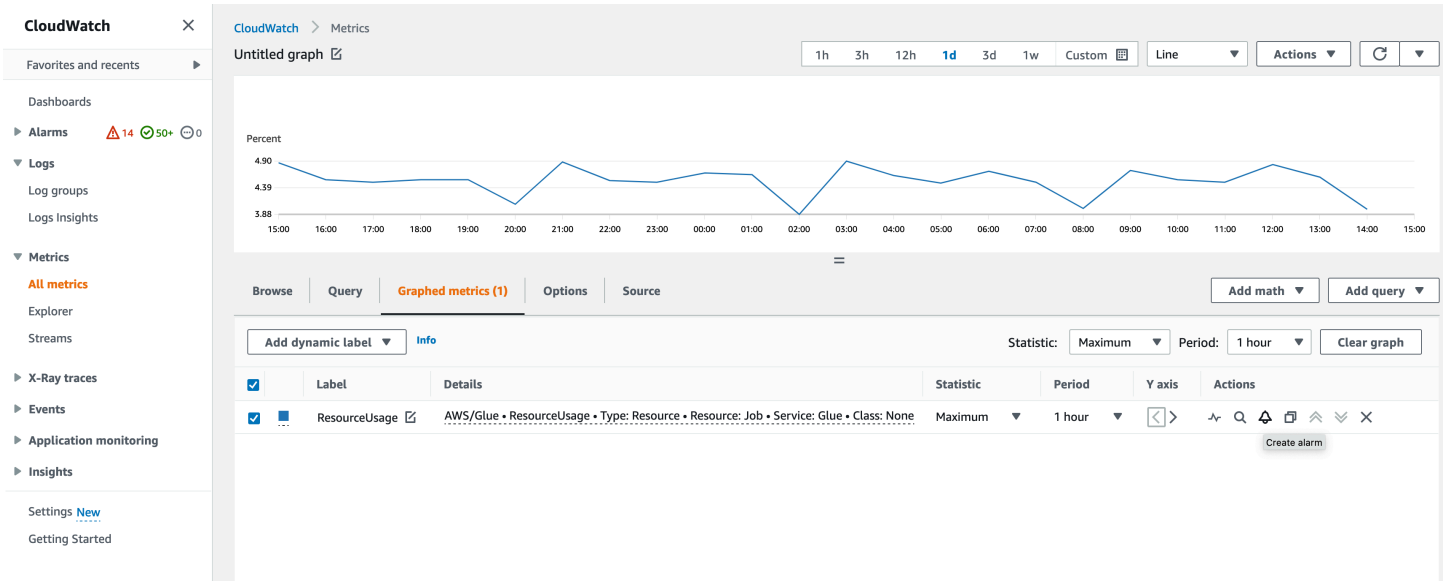
- Numero di flussi di lavoro per account
- Numero di trigger per account
- Numero di processi per account
- Numero di esecuzioni processo simultanee per account
- Numero di schemi per account
- Numero di sessioni interattive per account

Configurazione e utilizzo dei parametri delle risorse

Per utilizzare questa funzionalità, puoi accedere alla console Amazon CloudWatch per visualizzare i parametri e configurare gli allarmi. I parametri si trovano nello spazio dei nomi AWS/Glue e rappresentano una percentuale del conteggio effettivo dell'utilizzo delle risorse diviso per la quota di risorse. I parametri di CloudWatch vengono forniti ai tuoi account senza alcun costo. Ad esempio, se hai creato 10 flussi di lavoro e la tua quota di servizio ti consente di avere un massimo di 200 flussi di lavoro, l'utilizzo è 10/200 = 5% e nel grafico vedrai un punto dati di 5 come percentuale. Per maggiore specificità:

```
Namespace: AWS/Glue
Metric name: ResourceUsage
Type: Resource
Resource: Workflow (or Trigger, Job, JobRun, Blueprint, InteractiveSession)
Service: Glue
```

```
Class: None
```



Per creare un allarme per un parametro nella console CloudWatch:

1. Una volta individuato il parametro, vai a Parametri definiti.
2. Fai clic su Crea allarme in Operazioni.
3. Configura l'allarme secondo necessità.

Emettiamo dei parametri ogni volta che l'utilizzo delle risorse cambia, ad esempio in caso di aumento o diminuzione. Tuttavia, se l'utilizzo delle risorse non cambia, emettiamo i parametri ogni ora, in modo da avere un grafico CloudWatch continuo. Per evitare la perdita di punti dati, sconsigliamo di configurare un periodo inferiore a 1 ora.

È possibile configurare gli allarmi anche tramite AWS CloudFormation, come nel seguente esempio. In questo esempio, quando l'utilizzo delle risorse del flusso di lavoro raggiunge l'80%, viene attivato un allarme per inviare un messaggio all'argomento SNS esistente, a cui è possibile abbonarsi per ricevere notifiche.

```
{
 "Type": "AWS::CloudWatch::Alarm",
 "Properties": {
  "AlarmName": "WorkflowUsageAlarm",
  "ActionsEnabled": true,
  "OKActions": [],
  "AlarmActions": [
```

```
    "arn:aws:sns:af-south-1:085425700061:Default_CloudWatch_Alarms_Topic"
   ],
   "InsufficientDataActions": [],
   "MetricName": "ResourceUsage",
   "Namespace": "AWS/Glue",
   "Statistic": "Maximum",
   "Dimensions": [{
     "Name": "Type",
     "Value": "Resource"
    },
    {
     "Name": "Resource",
     "Value": "Workflow"
    },
    {
     "Name": "Service",
     "Value": "Glue"
    },
    {
     "Name": "Class",
     "Value": "None"
    }
   ],
   "Period": 3600,
   "EvaluationPeriods": 1,
   "DatapointsToAlarm": 1,
   "Threshold": 80,
   "ComparisonOperator": "GreaterThanThreshold",
   "TreatMissingData": "notBreaching"
  }
}
```

Registrazione delle chiamate API AWS Glue con AWS CloudTrail

AWS Glue è integrato con AWS CloudTrail, un servizio che offre un record delle operazioni eseguite da un utente, un ruolo o un servizio AWS in AWS Glue. CloudTrail acquisisce tutte le chiamate API AWS Glue come eventi. Le chiamate acquisite includono le chiamate dalla console di AWS Glue e le chiamate di codice alle operazioni delle API AWS Glue. Se si crea un trail, è possibile abilitare la distribuzione continua di eventi CloudTrail in un bucket Amazon S3, inclusi gli eventi per AWS Glue. Se non si configura un trail, è comunque possibile visualizzare gli eventi più recenti nella console di CloudTrail in Event history (Cronologia eventi). Le informazioni raccolte da CloudTrail consentono

di determinare la richiesta effettuata ad AWS Glue, l'indirizzo IP da cui è partita la richiesta, l'autore della richiesta, il momento in cui è stata eseguita e altri dettagli.

Per ulteriori informazioni su CloudTrail, consultare la AWS CloudTrailGuida per l'utente di .

Informazioni su AWS Glue in CloudTrail

CloudTrail è abilitato sull'account AWS al momento della sua creazione. Quando si verifica un'attività in AWS Glue, tale attività viene registrata in un evento CloudTrail insieme ad altri eventi di servizio AWS nella Cronologia eventi. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, consulta Visualizzazione di eventi nella cronologia degli eventi di CloudTrail.

Per una registrazione continua degli eventi nell'account AWS che includa gli eventi per AWS Glue, creare un trail. Un percorso abilita la distribuzione da parte di CloudTrail dei file di log in un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le Regioni nella partizione AWS e distribuisce i file di log nel bucket Amazon S3 specificato. Inoltre, è possibile configurare altri servizi AWS per analizzare con maggiore dettaglio e usare i dati evento raccolti nei log CloudTrail. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Creazione di un percorso per il tuo account AWS
- Servizi e integrazioni CloudTrail supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di log CloudTrail da più regioni e Ricezione di file di log CloudTrail da più account

Tutte le operazioni di AWS Glue vengono registrate da CloudTrail e sono documentate in API AWS Glue. Ad esempio, le chiamate alle operazioni `CreateDatabase`, `CreateTable` e `CreateScript` generano voci nei file di log di 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 le credenziali dell'utente IAM o root.
- 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 servizio AWS.

Per ulteriori informazioni, consulta Elemento userIdentity di CloudTrail.

Tuttavia, CloudTrail non registra tutte le informazioni relative alle chiamate. Ad esempio, non registra determinate informazioni sensibili, ad esempio gli elementi `ConnectionProperties` utilizzati nelle richieste di connessione e registra un valore `null` al posto delle risposte restituite dalle seguenti API:

```
BatchGetPartition       GetCrawlers          GetJobs          GetTable
CreateScript            GetCrawlerMetrics    GetJobRun        GetTables
GetCatalogImportStatus  GetDatabase          GetJobRuns       GetTableVersions
GetClassifier           GetDatabases         GetMapping       GetTrigger
GetClassifiers          GetDataflowGraph     GetObjects       GetTriggers
GetConnection           GetDevEndpoint       GetPartition     GetUserDefinedFunction
GetConnections          GetDevEndpoints      GetPartitions    GetUserDefinedFunctions
GetCrawler              GetJob               GetPlan
```

Comprensione delle voci dei file di log di AWS Glue

Un trail è una configurazione che consente la distribuzione di eventi come i file di log in un bucket Amazon S3 specificato. I file di log di CloudTrail possono contenere una o più voci di log. Un evento rappresenta una singola richiesta da un'origine e include informazioni sull'operazione richiesta, sulla data e sull'ora dell'operazione, sui parametri richiesti e così via. I file di log CloudTrail non sono una traccia di pila ordinata delle chiamate API pubbliche e di conseguenza non devono apparire in base a un ordine specifico.

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `DeleteCrawler`.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/johndoe",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "johndoe"
  },
  "eventTime": "2017-10-11T22:29:49Z",
  "eventSource": "glue.amazonaws.com",
  "eventName": "DeleteCrawler",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "72.21.198.64",
  "userAgent": "aws-cli/1.11.148 Python/3.6.1 Darwin/16.7.0 botocore/1.7.6",
```

```
  "requestParameters": {
    "name": "tes-alpha"
  },
  "responseElements": null,
  "requestID": "b16f4050-aed3-11e7-b0b3-75564a46954f",
  "eventID": "e73dd117-cfd1-47d1-9e2f-d1271cad838c",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

L'esempio seguente mostra una voce di log di CloudTrail che illustra un'operazione `CreateConnection`.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/johndoe",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "johndoe"
  },
  "eventTime": "2017-10-13T00:19:19Z",
  "eventSource": "glue.amazonaws.com",
  "eventName": "CreateConnection",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "72.21.198.66",
  "userAgent": "aws-cli/1.11.148 Python/3.6.1 Darwin/16.7.0 botocore/1.7.6",
  "requestParameters": {
    "connectionInput": {
      "name": "test-connection-alpha",
      "connectionType": "JDBC",
      "physicalConnectionRequirements": {
        "subnetId": "subnet-323232",
        "availabilityZone": "us-east-1a",
        "securityGroupIdList": [
          "sg-12121212"
        ]
      }
    }
  },
  "responseElements": null,
```

```
  "requestID": "27136ebc-afac-11e7-a7d6-ab217e5c3f19",
  "eventID": "e8b3baeb-c511-4597-880f-c16210c60a4a",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Stati di esecuzione dei processi AWS Glue

Puoi visualizzare lo stato di un processo di estrazione, trasformazione e caricamento (ETL) AWS Glue mentre è in esecuzione o una volta arrestato. Puoi visualizzare lo stato utilizzando la AWS Glue console, il AWS Command Line Interface (AWS CLI) o l'GetJobRunazione nell'AWS GlueAPI.

I possibili stati di esecuzione dei processi sono `STARTING`, `RUNNING`, `STOPPING`, `STOPPED`, `SUCCEEDED`, `FAILED`, `ERROR`, `WAITING` e `TIMEOUT`.

La tabella seguente elenca gli stati che indicano l'interruzione anormale del processo.

Stato di esecuzione dei processi	Descrizione
`FAILED`	Il processo ha superato il numero massimo di esecuzioni simultanee consentite o si è interrott o con un codice di uscita sconosciuto.
`ERROR`	Un flusso di lavoro, un trigger pianificato o un trigger di evento ha tentato di eseguire un processo eliminato.
`TIMEOUT`	Il tempo di esecuzione del processo ha superato il valore di timeout specificato.

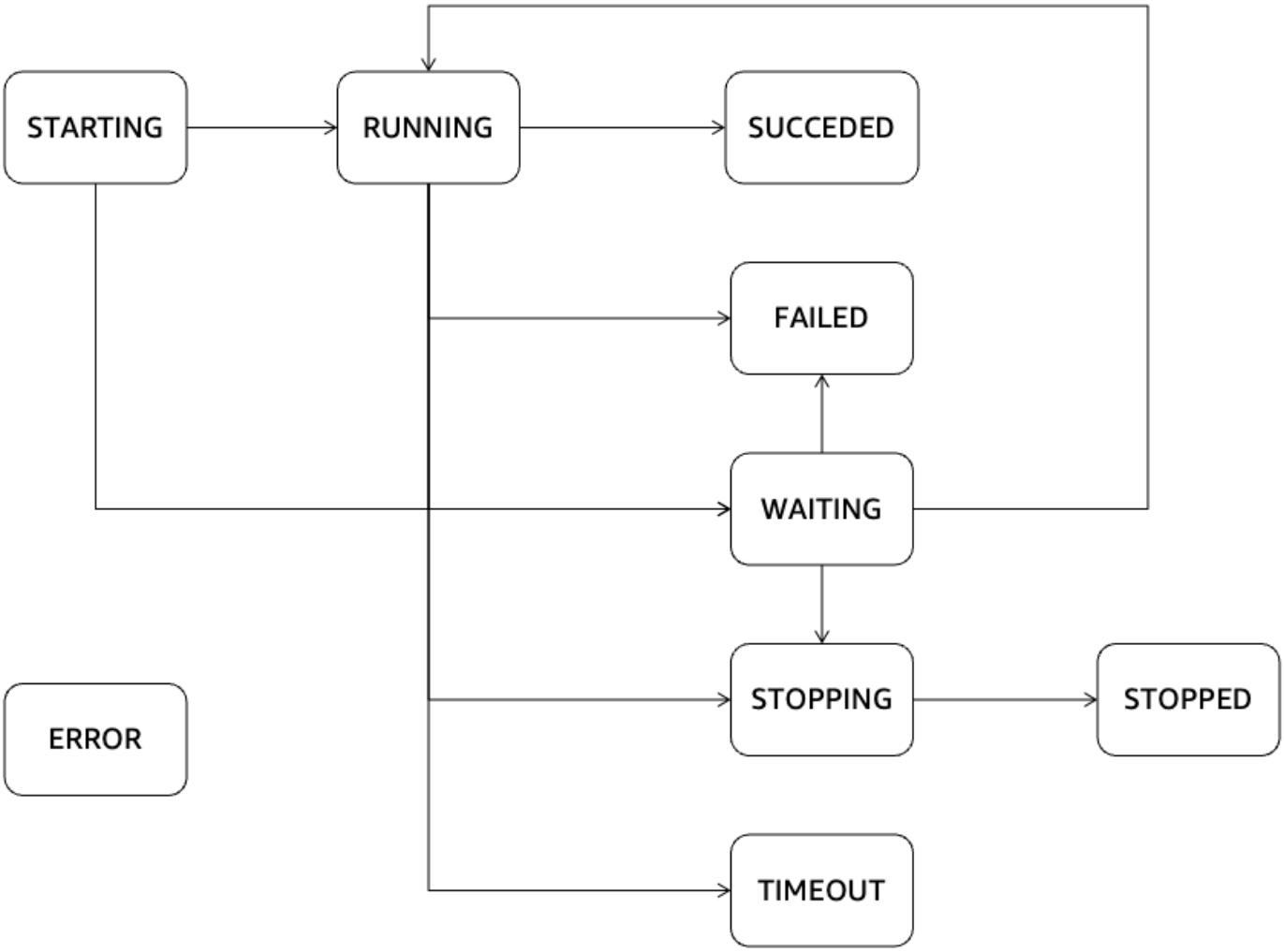
Lo stato `WAITING` indica che l'esecuzione di un processo è in attesa di risorse. La tabella seguente descrive il comportamento di attesa per diverse classi di processi.

Tipo di processo	Comportamento
Processi Spark (standard)	I processi che non sono stati configurati per riprovare in base alla configurazione `maxRetries` potrebbero passare allo stato IN

Tipo di processo	Comportamento
	ATTESA. Una nuova esecuzione del processo sarà nello stato WAITING (IN ATTESA) se il servizio non è in grado di acquisire risorse sufficienti per avviare l'esecuzione. Questo può verificarsi a causa delle Service Quotas per l'account o dei limiti di capacità nella regione. Si può verificare uno dei seguenti casi di errore: • Numero massimo di esecuzioni simultanee dei processi per account superato • Il numero massimo di esecuzioni simultane e per processo è stato superato (include la quota di servizio a livello di account e il limite specificato per il processo con `MaxConcurrentRuns`) • È stato superato il limite massimo di calcolo simultaneo (utilizzo della DPU) • Risorsa non disponibile Per ulteriori informazioni sulle Service Quotas AWS Glue, consulta Endpoint e quote di AWS. Il tempo di attesa da parte di AWS Glue per le risorse può variare in base alle circostanze. Un processo può passare da uno stato non terminale all'altro nel tentativo di acquisire risorse. Alla fine, il processo passerà a `FAILED` se non sarà in grado di acquisire risorse. AWS Glue riproverà per un massimo di 15 minuti o 10 tentativi, a seconda dell'evento che si verifica per primo.

Tipo di processo	Comportamento
Processi Spark (Flex)	Una nuova esecuzione del processo sarà nello stato WAITING (IN ATTESA) se il servizio non è in grado di acquisire risorse sufficienti per avviare l'esecuzione, il che ritarda l'avvio dell'esecuzione. L'esecuzione sarà in stato WAITING (IN ATTESA) per un massimo di 20 minuti (timeout controllato dal servizio). Dopo 15 minuti, il servizio proverà a eseguire un avvio forzato e, a seconda della capacità disponibile, l'esecuzione potrebbe iniziare o non riuscire con un messaggio di errore appropriato.
Processi shell di Python	Stesso comportamento dei processo standard che utilizza Spark.

Il seguente diagramma di stato delinea le transizioni di stato previste durante il ciclo di vita di un job Glue. AWS Queste informazioni sono applicabili a tutti i tipi di job.


STARTING
RUNNING
SUCCEDED
FAILED
WAITING
STOPPING
STOPPED
ERROR
TIMEOUT

AWS Glue Streaming

AWS Glue Streaming, un componente di AWS Glue, consente di gestire con efficienza i dati in streaming quasi in tempo reale, permettendo di svolgere attività cruciali come l'importazione e l'elaborazione dei dati nonché il machine learning. Utilizzando il framework Apache Spark Streaming, AWS Glue Streaming fornisce un servizio serverless in grado di gestire lo streaming di dati su larga scala. AWS Glue offre varie ottimizzazioni oltre ad Apache Spark, come l'infrastruttura serverless, il dimensionamento automatico, lo sviluppo visivo dei processi, i notebook ad accensione istantanea per i processi di streaming e altri miglioramenti delle prestazioni.

Casi d'uso per lo streaming

Alcuni casi d'uso comuni per AWS Glue Streaming includono:

Elaborazione dei dati quasi in tempo reale: AWS Glue Streaming consente alle organizzazioni di elaborare i dati di streaming quasi in tempo reale, consentendo di ricavare approfondimenti e prendere decisioni tempestive sulla base delle informazioni più recenti.

Rilevamento delle frodi: è possibile utilizzare AWS Glue Streaming per l'analisi in tempo reale dei dati di streaming, traendone informazioni utili per rilevare attività fraudolente, come frodi con carte di credito, intrusioni nella rete o truffe online. Elaborando e analizzando continuamente i dati in entrata, è possibile identificare rapidamente anomalie o sequenze sospette.

Analisi dei social media: AWS Glue Streaming può elaborare in tempo reale i dati sui social media, come tweet, post o commenti, consentendo alle organizzazioni di monitorare le tendenze, l'analisi del sentiment e gestire la reputazione del marchio in tempo reale.

Analisi dell'Internet delle cose (IoT): AWS Glue Streaming è adatto per gestire e analizzare flussi di dati ad alta velocità generati da dispositivi IoT, sensori e macchinari connessi. Consente il monitoraggio in tempo reale, il rilevamento delle anomalie, la manutenzione predittiva e altri casi d'uso di analisi IoT.

Analisi di clickstream: AWS Glue Streaming può elaborare e analizzare in tempo reale i dati di clickstream provenienti da siti web o applicazioni per dispositivi mobili. In tal modo, le aziende possono ottenere approfondimenti sul comportamento degli utenti, personalizzare le loro esperienze e ottimizzare le campagne di marketing sulla base di dati di clickstream in tempo reale.

Monitoraggio e analisi dei log: AWS Glue Streaming è in grado di elaborare e analizzare continuamente i dati di log da server, applicazioni o dispositivi di rete in tempo reale. Ciò contribuisce a rilevare le anomalie, risolvere i problemi e monitorare lo stato e le prestazioni del sistema.

Sistemi di raccomandazione: AWS Glue Streaming può elaborare i dati sulle attività degli utenti in tempo reale e aggiornare i modelli di raccomandazione in modo dinamico. Ciò consente di fornire consigli personalizzati e in tempo reale in base al comportamento e alle preferenze degli utenti.

Questi sono alcuni esempi della vasta gamma di casi d'uso in cui è possibile applicare AWS Glue Streaming. La sua integrazione con l'ecosistema e i servizi gestiti AWS lo rendono una scelta conveniente per l'elaborazione e l'analisi dei flussi in tempo reale nel cloud.

Quali sono i vantaggi dell'utilizzo di AWS Glue Streaming?

I vantaggi dell'utilizzo di AWS Glue Streaming sono i seguenti:

- Serverless: AWS Glue Streaming è serverless, il che elimina la necessità di gestire l'infrastruttura. Ciò riduce il sovraccarico operativo e consente agli utenti di concentrarsi sulle attività di elaborazione e analisi dei dati anziché sulla gestione dell'infrastruttura.
- Dimensionamento automatico: AWS Glue Streaming offre funzionalità di dimensionamento automatico, regolando dinamicamente la capacità di elaborazione in base al carico di lavoro. È scalabile automaticamente in orizzontale o in verticale per gestire le fluttuazioni del volume di dati, garantendo livelli ottimali di prestazioni e utilizzo delle risorse.
- Sviluppo visivo: lo sviluppo di processi in streaming può essere complesso. AWS Glue Streaming affronta questa sfida offrendo AWS Glue Studio, uno strumento di creazione visiva. AWS Glue Studio semplifica il processo di creazione di flussi di lavoro di streaming e consente agli sviluppatori di progettare e gestire visivamente le applicazioni di streaming, riducendo la curva di apprendimento e aumentando la produttività.
- Convenienza: in quanto servizio serverless, AWS Glue Streaming offre efficienza in termini di costi eliminando la necessità di fornire e mantenere l'infrastruttura. Agli utenti vengono fatturate le risorse utilizzate durante l'esecuzione dei processi di streaming, contribuendo all'ottimizzazione dei costi e a un dimensionamento in base all'utilizzo effettivo.
- Gestione di carichi di lavoro complessi: AWS Glue Streaming è progettato per gestire carichi di lavoro di streaming complessi. Può elaborare e analizzare grandi volumi di dati in tempo reale, supportare trasformazioni avanzate e integrarsi con altri servizi AWS, abilitando pipeline di dati di streaming e flussi di lavoro di analisi sofisticati.

- Nessun vincolo: AWS Glue Streaming offre flessibilità e non richiede il vincolo a un fornitore. Gli utenti possono utilizzare AWS Glue Streaming come parte di un più ampio ecosistema AWS, integrandolo senza problemi con altri servizi AWS. Ciò consente una facile integrazione con le origini dati, le applicazioni e i servizi esistenti senza vincolare a una tecnologia o piattaforma specifica.

Quando è indicato utilizzare AWS Glue Streaming?

I casi d'uso dello streaming includono numerose opzioni. Consigliamo di utilizzare AWS Glue Streaming negli scenari riportati di seguito.

1. Se stai già utilizzando AWS Glue Spark per l'elaborazione di batch, AWS Glue Streaming è la scelta ideale per te. Fornisce una transizione ottimale alla creazione di processi di streaming senza la necessità di imparare un nuovo linguaggio o framework. Sfruttando le conoscenze e l'infrastruttura esistenti, AWS Glue Streaming semplifica il percorso di sviluppo del processo e consente di estendere facilmente le capacità di elaborazione dei dati a scenari di streaming in tempo reale.
2. Se hai bisogno di un servizio o di un prodotto unificato per gestire carichi di lavoro in batch, in streaming e basati sugli eventi, AWS Glue Streaming è la soluzione ideale per te. Con AWS Glue Streaming, puoi consolidare le tue esigenze di elaborazione dei dati in un unico framework, eliminando la complessità legate alla gestione di più sistemi. Ciò consente di sviluppare e mantenere flussi di lavoro di dati diversi in modo efficiente, garantendo al contempo la coerenza e la compatibilità tra diversi tipi di carichi di lavoro.
3. AWS Glue Streaming è ideale per scenari che prevedono volumi di flussi di dati estremamente grandi e trasformazioni complesse, come unioni di flussi o database relazionali. È in grado di elaborare e analizzare in modo efficiente enormi flussi di dati, consentendoti di affrontare con facilità carichi di lavoro impegnativi. Dall'importazione dei dati ad alta velocità a complesse manipolazioni dei dati, la scalabilità e le funzionalità di elaborazione avanzate di AWS Glue Streaming garantiscono prestazioni ottimali e risultati accurati.
4. Se preferisci un approccio visivo alla creazione di processi di streaming, AWS Glue offre AWS Glue Studio, con cui puoi progettare e gestire visivamente le tue applicazioni di streaming, semplificando il processo di sviluppo. Questa interfaccia intuitiva consente agli sviluppatori di creare, configurare e monitorare i flussi di lavoro di streaming utilizzando un'interfaccia visiva, riducendo la curva di apprendimento e aumentando la produttività.
5. AWS Glue Streaming è una scelta eccellente per i casi d'uso quasi in tempo reale soggetti ad accordi sul livello di servizio (SLA) rigorosi superiori a 10 secondi.

6. Se stai creando un data lake transazionale utilizzando Apache Iceberg, Apache Hudi o Delta Lake, AWS Glue Streaming fornisce il supporto nativo per questi formati di tabelle aperte. Questa perfetta integrazione consente di elaborare i dati in streaming direttamente da questi data lake transazionali, garantendo la coerenza, l'integrità e la compatibilità dei dati.
7. Quando è necessario importare dati di streaming per una varietà di destinazioni di dati: AWS Glue Streaming fornisce destinazioni native a una varietà di destinazioni di dati come Amazon Redshift, Amazon RDS, Amazon Aurora, Oracle, SQL Server e altre.

Origini dati supportate

AWS Glue Streaming supporta le seguenti origini dati:

- Amazon Kinesis
- Amazon MSK (Streaming gestito per Apache Kafka)
- Apache Kafka gestito dal cliente

Destinazioni di dati supportate

AWS Glue Streaming supporta una varietà di destinazioni di dati, come ad esempio:

- Destinazioni di dati supportate da Catalogo dati AWS Glue
- Amazon S3
- Amazon Redshift
- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server
- Snowflake
- Qualsiasi database che possa essere collegato tramite JDBC
- Apache Iceberg, Delta e Apache Hudi
- Connettori Marketplace AWS Glue

Tutorial: creazione del primo carico di lavoro di streaming utilizzando AWS Glue Studio

In questo tutorial imparerai a creare un processo di streaming utilizzando AWS Glue Studio. AWS Glue Studio è un'interfaccia visiva per creare processi di AWS Glue.

È possibile creare processi in streaming di estrazione, trasformazione e caricamento (ETL) che vengono eseguiti continuamente e utilizzano dati da origini di streaming in Flusso di dati Amazon Kinesis, Apache Kafka e Streaming gestito da Amazon per Apache Kafka (Amazon MSK).

Prerequisiti

Per seguire questo tutorial è necessario un utente munito delle autorizzazioni della console AWS per utilizzare AWS Glue, Amazon Kinesis, Amazon S3, Amazon Athena, AWS CloudFormation, AWS Lambda e Amazon Cognito.

Utilizzo dei dati in streaming da Amazon Kinesis

Argomenti

Generazione di dati fittizi con Kinesis Data Generator

È possibile generare sinteticamente dati di esempio in formato JSON utilizzando Kinesis Data Generator (KDG). Puoi trovare le istruzioni complete e i dettagli nella documentazione dello strumento.

1. Per iniziare, fai clic su

   per eseguire un modello AWS CloudFormation nel tuo ambiente AWS.

   > **Note**
   >
   > Potresti riscontrare un errore nel modello CloudFormation perché alcune risorse, come l'utente Amazon Cognito per Kinesis Data Generator, esistono già nel tuo account AWS.

> Ciò potrebbe essere dovuto al fatto che l'hai già configurato in un altro tutorial o da un post di un blog. Per risolvere questo problema, puoi provare a utilizzare il modello in un nuovo account AWS oppure in un'altra regione AWS. Queste opzioni consentono di eseguire il tutorial senza entrare in conflitto con le risorse esistenti.

Il modello fornisce un flusso di dati Kinesis e un account Kinesis Data Generator. Crea anche un bucket Amazon S3 per contenere i dati e un ruolo di servizio Glue con l'autorizzazione richiesta per questo tutorial.

2. Immetti un Nome utente e una Password che KDG utilizzerà per l'autenticazione. Prendi nota del nome utente e della password per utilizzarli in seguito.
3. Seleziona Avanti fino all'ultimo passaggio. Esprimi il consenso alla creazione di risorse IAM. Verifica la presenza di eventuali errori nella parte superiore dello schermo, ad esempio la password che non soddisfa i requisiti minimi, e implementa il modello.
4. Vai alla scheda Output dello stack. Una volta implementato, il modello mostrerà la proprietà generata KinesisDataGeneratorUrl. Fai clic su quell'URL.
5. Inserisci il Nome utente e la Password di cui hai preso nota.
6. Seleziona la regione che stai utilizzando e seleziona il flusso Kinesis `GlueStreamTest-{AWS::AccountId}`.
7. Immetti il seguente modello:

```
{
    "ventilatorid": {{random.number(100)}},
    "eventtime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
    "serialnumber": "{{random.uuid}}",
    "pressurecontrol": {{random.number(
        {
            "min":5,
            "max":30
        }
    )}},
    "o2stats": {{random.number(
        {
            "min":92,
            "max":98
        }
    )}},
    "minutevolume": {{random.number(
```

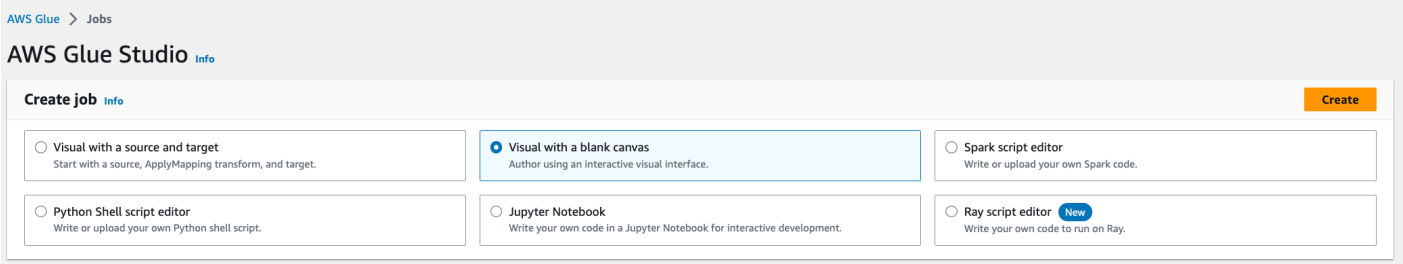
```
        {
            "min":5,
            "max":8
        }
    )}},
    "manufacturer": "{{random.arrayElement(
        ["3M", "GE","Vyaire", "Getinge"]
    )}}"
}
```

Ora puoi visualizzare i dati fittizi con Modello di prova e importare i dati fittizi in Kinesis con Invia dati.

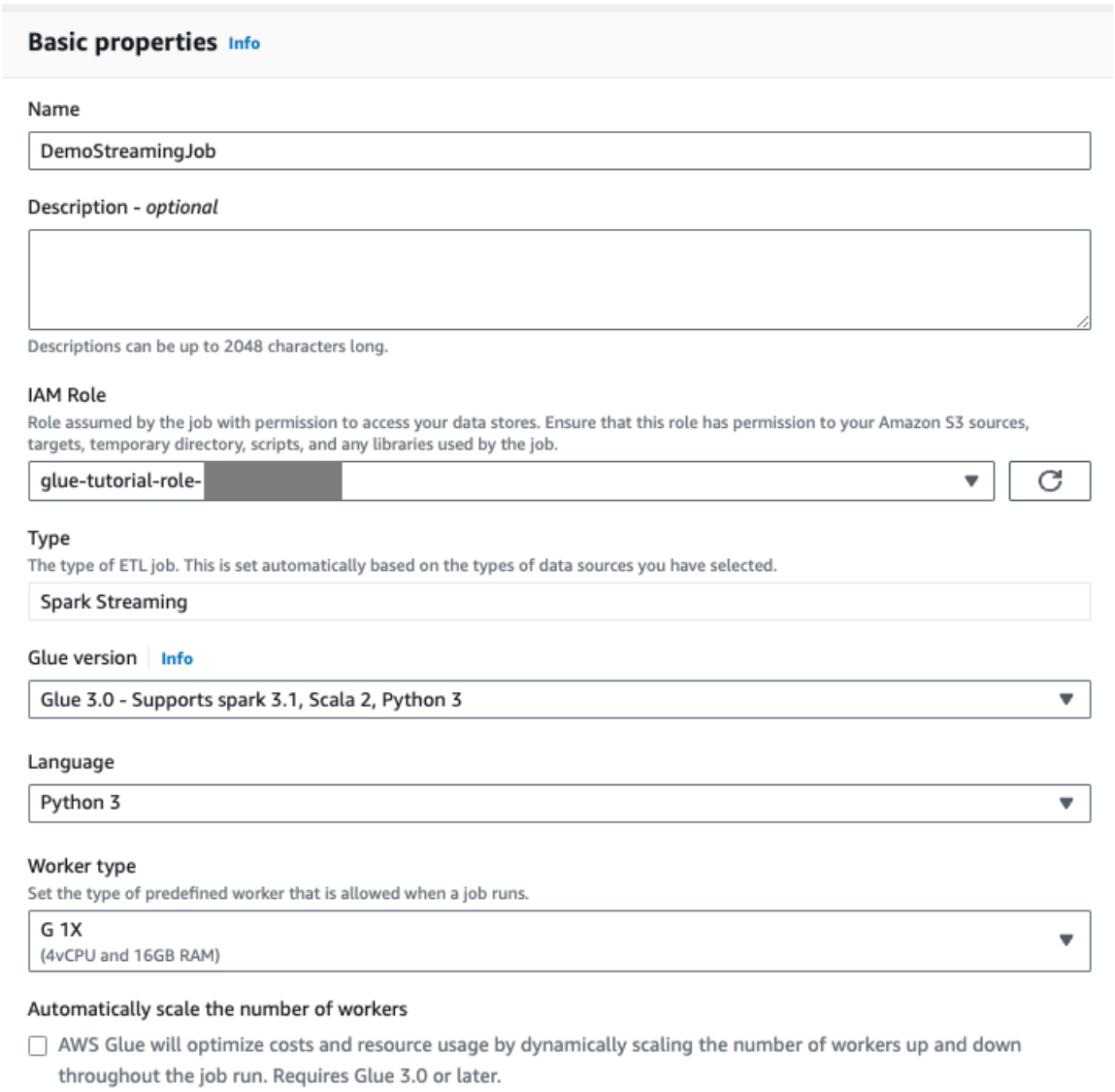
8. Fai clic su Invia dati e genera 5-10.000 record su Kinesis.

Creazione di un processo di streaming di AWS Glue con AWS Glue Studio

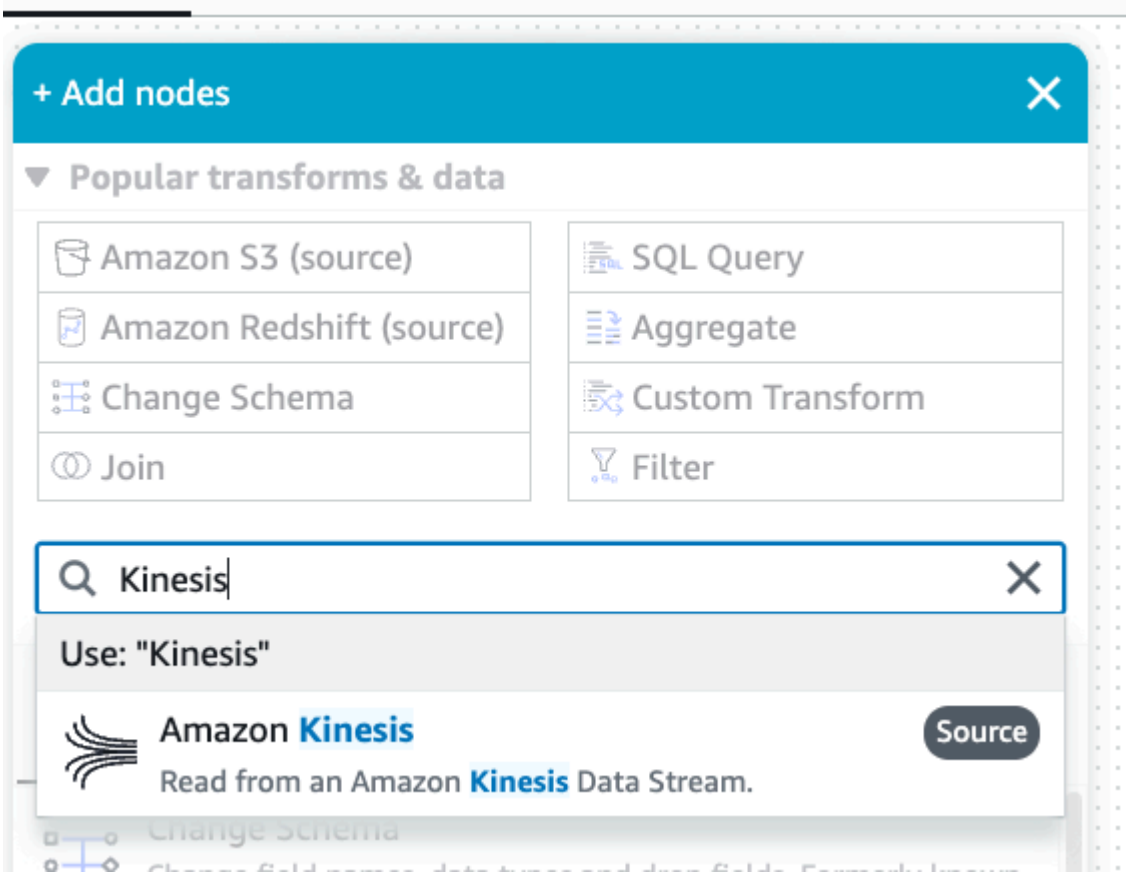
1. Passa alla console AWS Glue nella stessa regione.
2. Seleziona Processi ETL nella barra di navigazione a sinistra in Integrazione dati ed ETL.
3. Crea un processo di AWS Glue tramite Visivo con canvas vuoto.



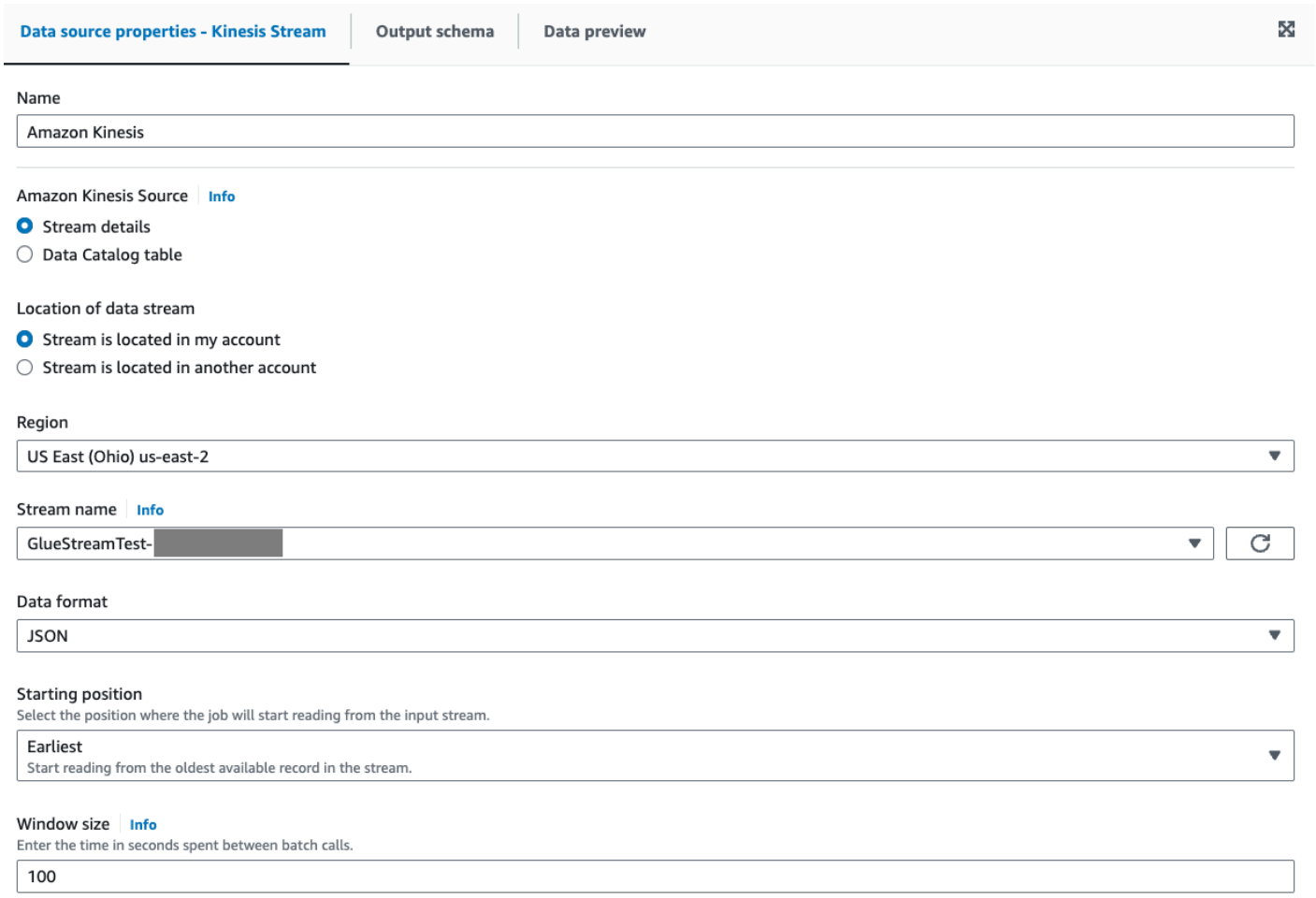
4. Passa alla scheda Dettagli del processo.
5. Per il nome del processo di AWS Glue, immetti `DemoStreamingJob`.
6. Per Ruolo IAM, seleziona il ruolo fornito dal modello CloudFormation, `glue-tutorial-role-${AWS::AccountId}`.
7. Per Versione Glue, seleziona Glue 3.0. Mantieni tutte le altre opzioni predefinite.



8. Vai alla scheda Visivo.
9. Fai clic sull'icona del segno più. Immetti Kinesis nella barra di ricerca. Seleziona l'origine dati Amazon Kinesis.



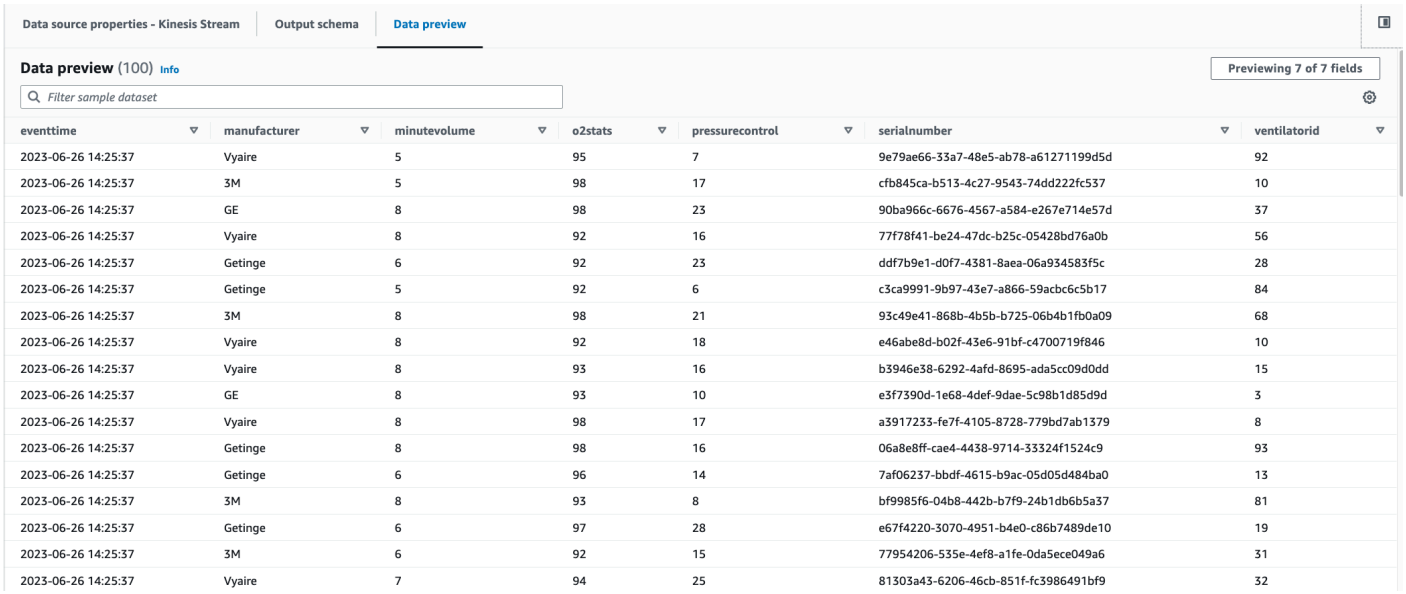
10. Seleziona Dettagli del flusso per Origine Amazon Kinesis nella scheda Proprietà dell'origine dati - Flusso Kinesis.
11. Seleziona Il flusso si trova nel mio account per Posizione del flusso di dati.
12. Seleziona la regione che stai utilizzando.
13. Seleziona il flusso `GlueStreamTest-{AWS::AccountId}`.
14. Mantieni tutte le altre impostazioni predefinite.



15. Vai alla scheda Anteprima dei dati.
16. Fai clic su Avvia sessione di anteprima dei dati, che visualizza in anteprima i dati fittizi generati da KDG. Scegli il ruolo di servizio Glue che hai creato in precedenza per il processo di AWS Glue Streaming.

    Occorrono 30-60 secondi prima che i dati di anteprima vengano visualizzati. Se compare Nessun dato da visualizzare, fai clic sull'icona a forma di ingranaggio e imposta il Numero di righe in base al quale campionare su `100`.

    Puoi visualizzare i dati di esempio come segue:



Data source properties - Kinesis Stream | Output schema | Data preview

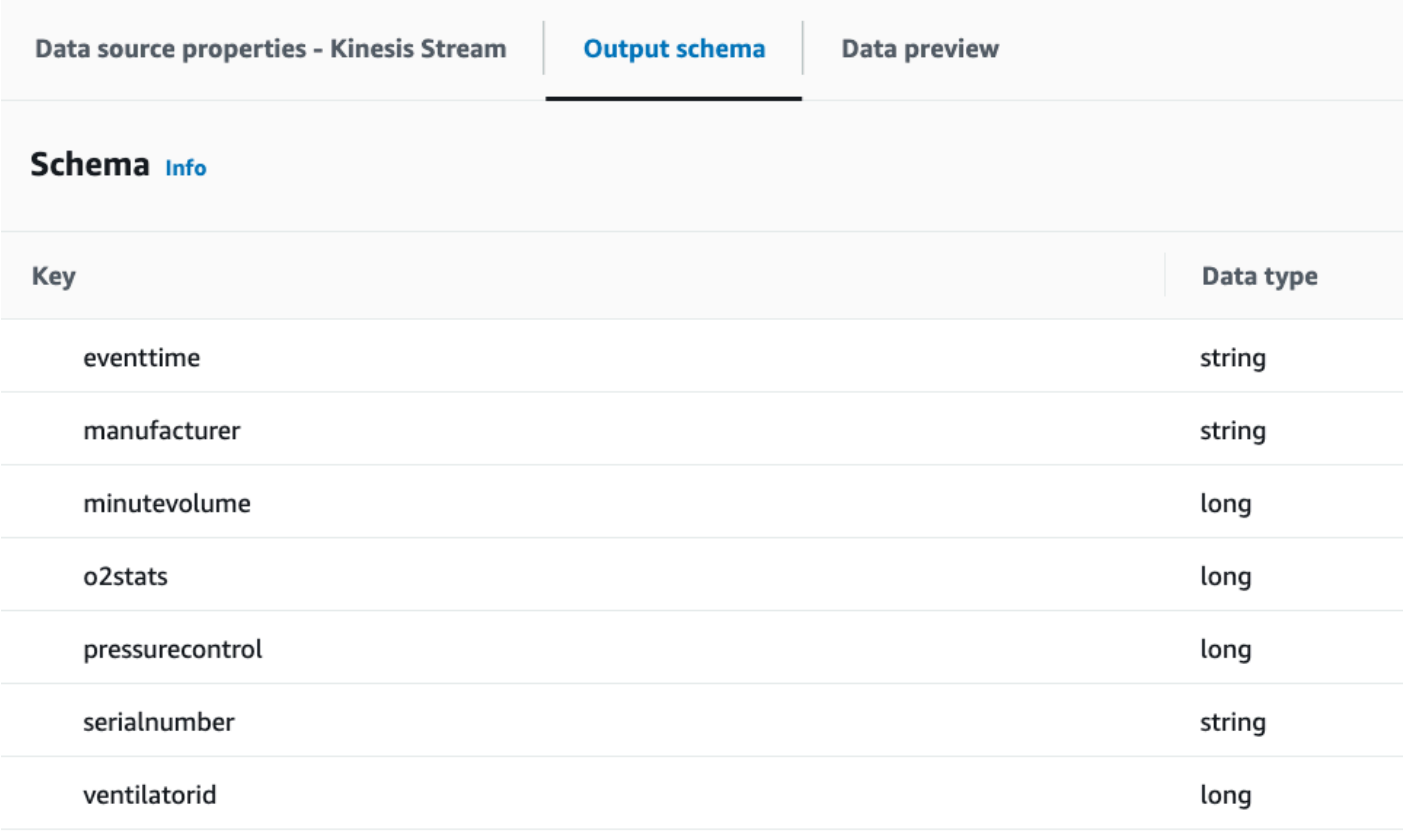
Data preview (100) Info

Previewing 7 of 7 fields

Filter sample dataset

eventtime	manufacturer	minutevolume	o2stats	pressurecontrol	serialnumber	ventilatorid
2023-06-26 14:25:37	Vyaire	5	95	7	9e79ae66-33a7-48e5-ab78-a61271199d5d	92
2023-06-26 14:25:37	3M	5	98	17	cfb845ca-b513-4c27-9543-74dd222fc537	10
2023-06-26 14:25:37	GE	8	98	23	90ba966c-6676-4567-a584-e267e714e57d	37
2023-06-26 14:25:37	Vyaire	8	92	16	77f78f41-be24-47dc-b25c-05428bd76a0b	56
2023-06-26 14:25:37	Getinge	6	92	23	ddf7b9e1-d0f7-4381-8aea-06a934583f5c	28
2023-06-26 14:25:37	Getinge	5	92	6	c3ca9991-9b97-43e7-a866-59acbc6c5b17	84
2023-06-26 14:25:37	3M	8	98	21	93c49e41-868b-4b5b-b725-06b4b1fb0a09	68
2023-06-26 14:25:37	Vyaire	8	92	18	e46abe8d-b02f-43e6-91bf-c4700719f846	10
2023-06-26 14:25:37	Vyaire	8	93	16	b3946e38-6292-4afd-8695-ada5cc09d0dd	15
2023-06-26 14:25:37	GE	8	93	10	e3f7390d-1e68-4def-9dae-5c98b1d85d9d	3
2023-06-26 14:25:37	Vyaire	8	98	17	a3917233-fe7f-4105-8728-779bd7ab1379	8
2023-06-26 14:25:37	Getinge	8	98	16	06a8e8ff-cae4-4438-9714-33324f1524c9	93
2023-06-26 14:25:37	Getinge	6	96	14	7af06237-bbdf-4615-b9ac-05d05d484ba0	13
2023-06-26 14:25:37	3M	8	93	8	bf9985f6-04b8-442b-b7f9-24b1db6b5a37	81
2023-06-26 14:25:37	Getinge	6	97	28	e67f4220-3070-4951-b4e0-c86b7489de10	19
2023-06-26 14:25:37	3M	6	92	15	77954206-535e-4ef8-a1fe-0da5ece049a6	31
2023-06-26 14:25:37	Vyaire	7	94	25	81303a43-6206-46cb-851f-fc3986491bf9	32

È inoltre possibile visualizzare lo schema dedotto nella scheda Schema di output.



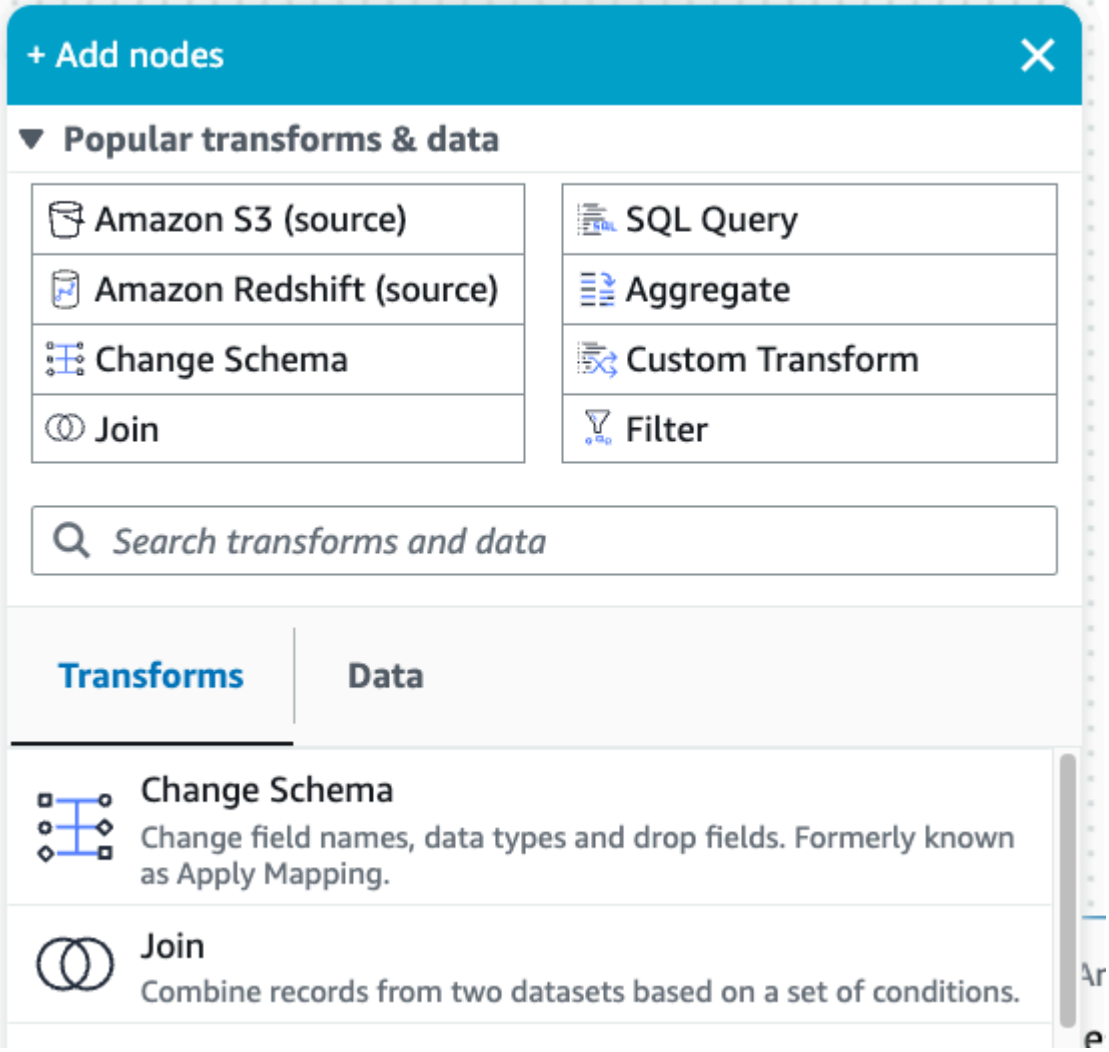
Data source properties - Kinesis Stream | Output schema | Data preview

Schema Info

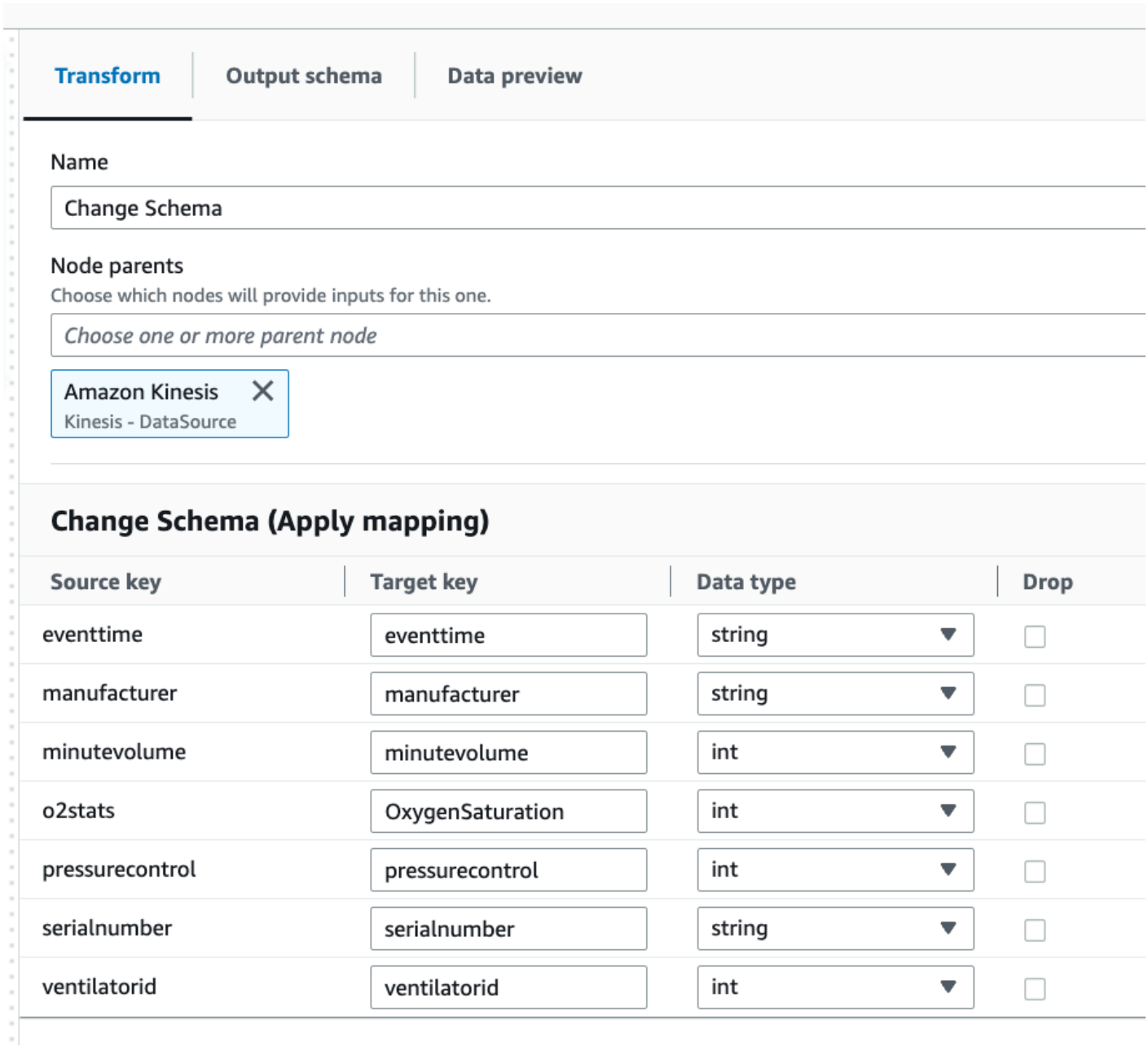
Key	Data type
eventtime	string
manufacturer	string
minutevolume	long
o2stats	long
pressurecontrol	long
serialnumber	string
ventilatorid	long

Esecuzione di una trasformazione e archiviazione del risultato della trasformazione in Amazon S3

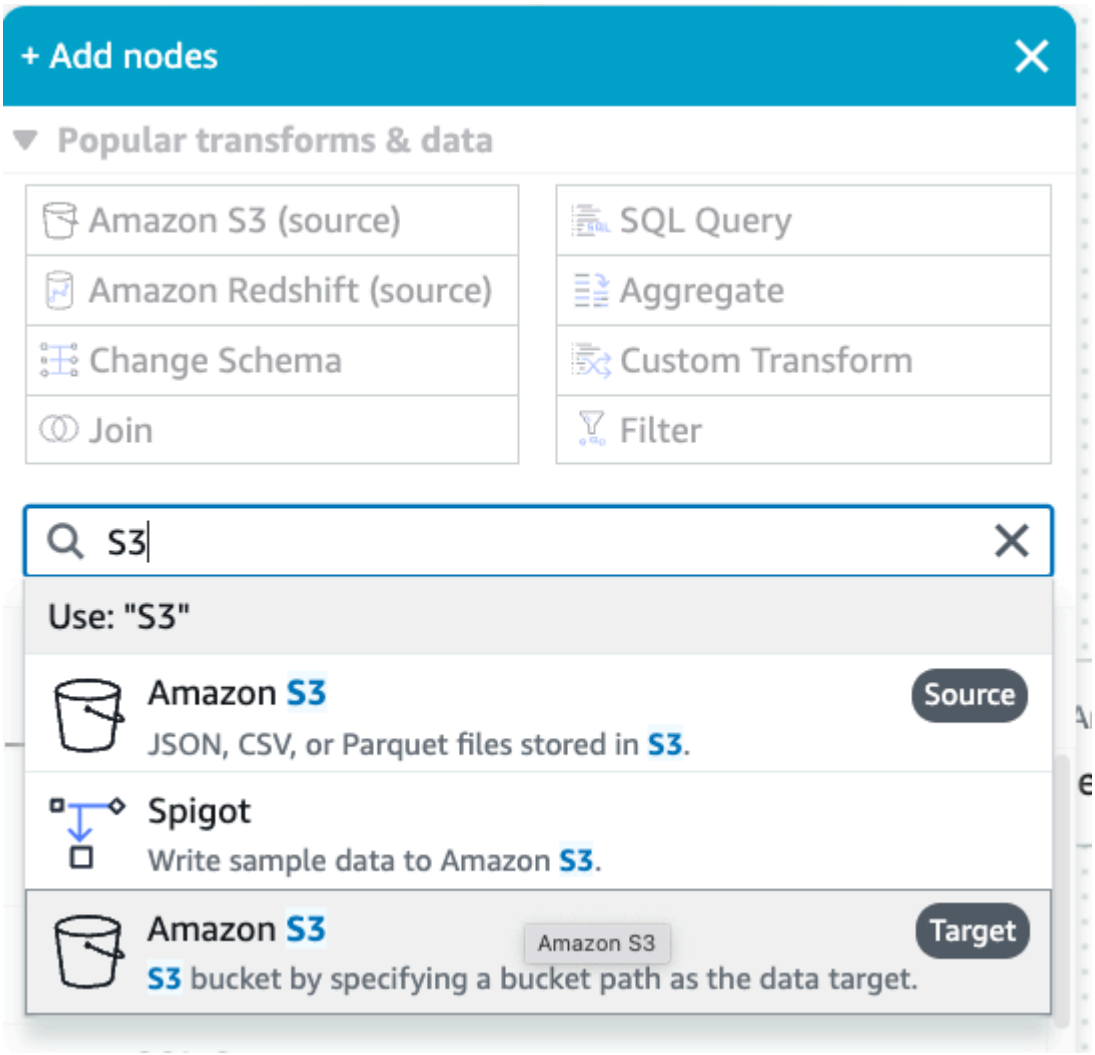
1. Con il nodo di origine selezionato, fai clic sull'icona del segno più in alto a sinistra per aggiungere un passaggio Trasformazioni.
2. Seleziona il passaggio Modifica schema.



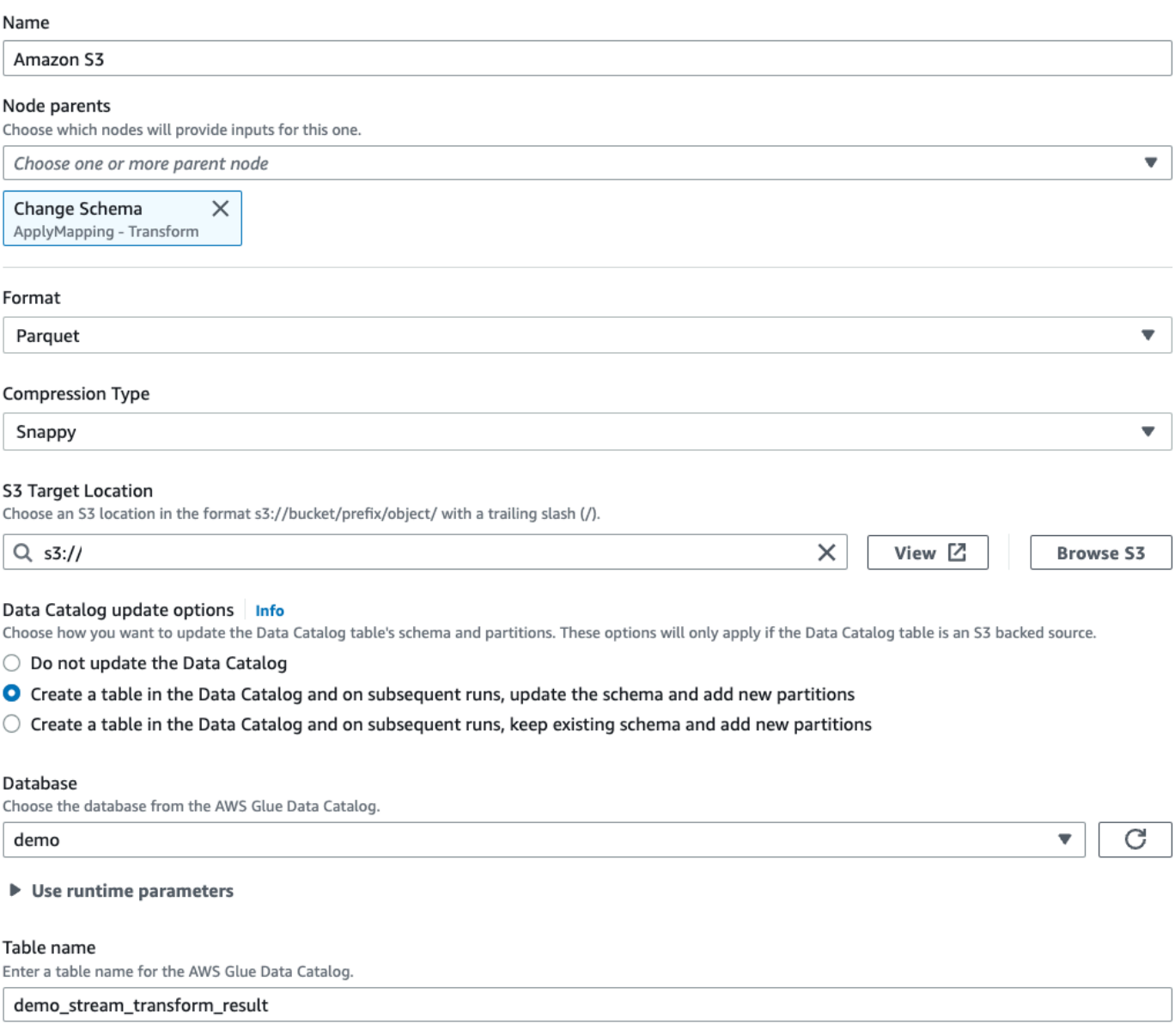
3. In questo passaggio è possibile rinominare i campi e convertire il tipo di dati dei campi. Rinomina la colonna `o2stats` in `OxygenSaturation` e converti tutti i tipi di dati `long` in `int`.



4. Fai clic sull'icona del segno più per aggiungere una destinazione Amazon S3. Immetti S3 nella casella di ricerca e seleziona la fase di trasformazione di Amazon S3 - Destinazione.

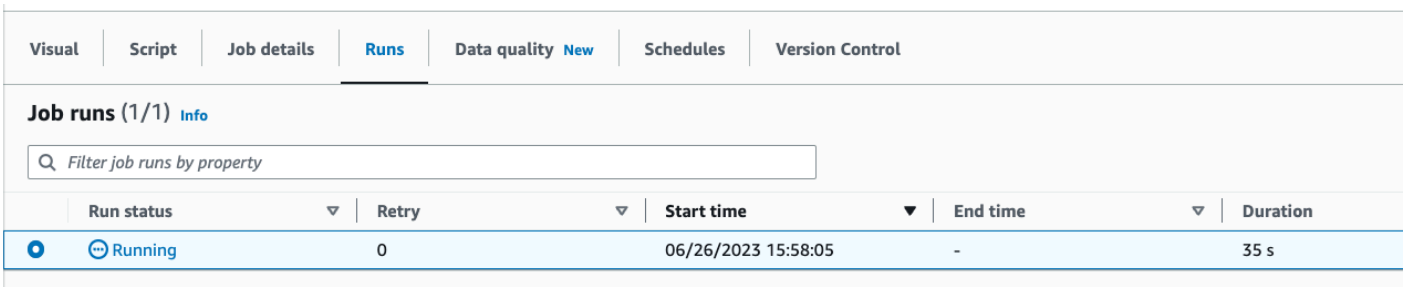


5. Seleziona Parquet come formato del file di destinazione.
6. Seleziona Snappy come tipo di compressione.
7. Inserisci una Posizione di destinazione S3 creata dal modello CloudFormation, `streaming-tutorial-s3-target-{AWS::AccountId}`.
8. Seleziona Crea una tabella nel Catalogo dati e, nelle esecuzioni successive, aggiorna lo schema e aggiungi nuove partizioni.
9. Inserisci il nome del Database e della Tabella di destinazione per archiviare lo schema della tabella di destinazione Amazon S3.

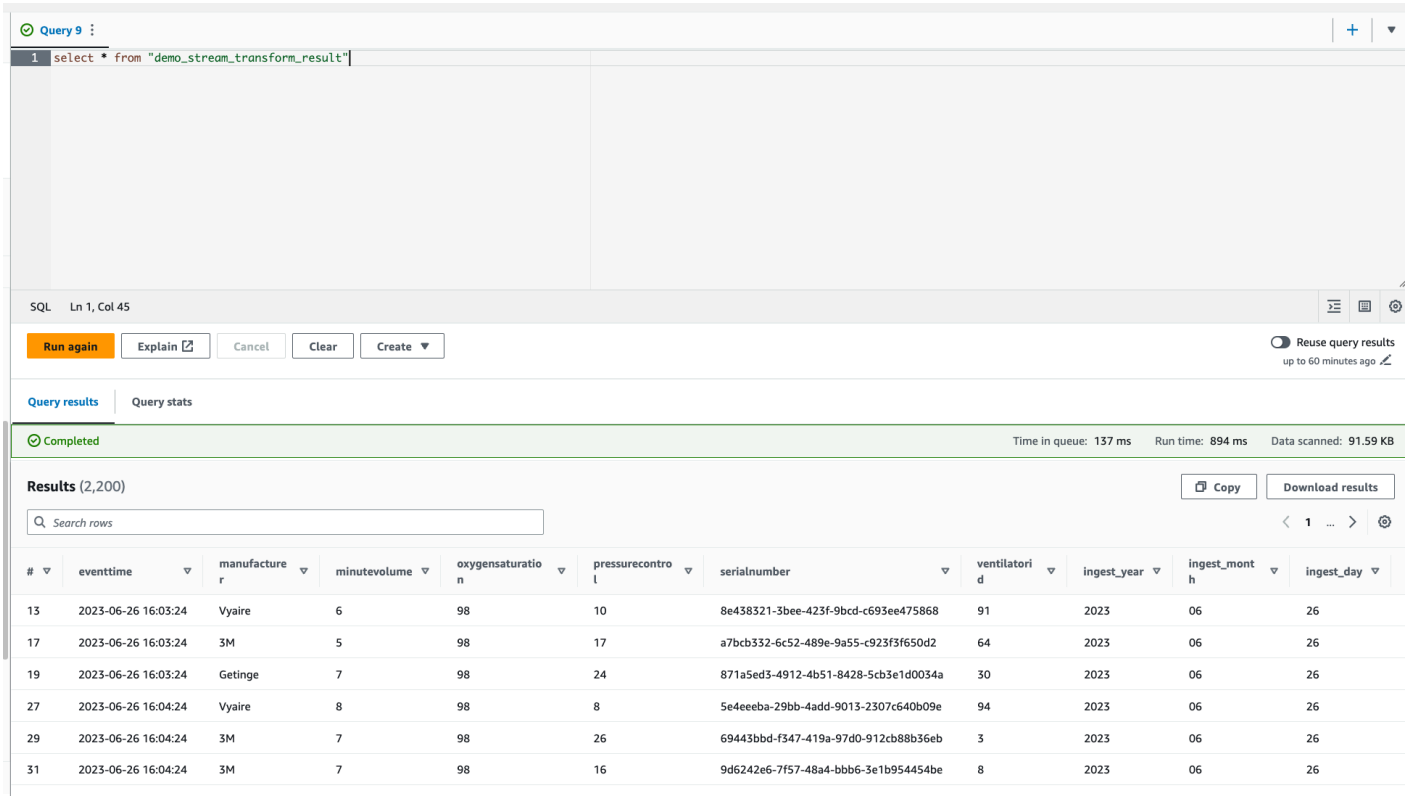


10. Fai clic sulla scheda Script per visualizzare il codice generato.
11. Fai clic su Salva in alto a destra per salvare il codice ETL, quindi fai clic su Esegui per avviare il processo di streaming di AWS Glue.

    Puoi trovare lo Stato di esecuzione nella scheda Esecuzioni. Lascia che il processo venga eseguito per 3-5 minuti, quindi interrompilo.



12.Verifica la nuova tabella creata in Amazon Athena.



Tutorial: creazione del primo carico di lavoro di streaming utilizzando i notebook AWS Glue Studio

Questo tutorial spiega come sfruttare i notebook AWS Glue Studio per creare e ottimizzare in modo interattivo i processi di ETL per un'elaborazione dei dati quasi in tempo reale. Questa guida, indicata tanto per chi è ai primi passi con AWS Glue quanto per chi desideri migliorare le proprie competenze, illustra ogni fase del processo, consentendo di sfruttare tutto il potenziale dei notebook con le sessioni interattive di AWS Glue.

Con AWS Glue Streaming, è possibile creare processi in streaming di estrazione, trasformazione e caricamento (ETL) che vengono eseguiti continuamente e utilizzano dati da origini di streaming in Flusso di dati Amazon Kinesis, Apache Kafka e Streaming gestito da Amazon per Apache Kafka (Amazon MSK).

Prerequisiti

Per seguire questo tutorial è necessario un utente munito delle autorizzazioni della console AWS per utilizzare AWS Glue, Amazon Kinesis, Amazon S3, Amazon Athena, AWS CloudFormation, AWS Lambda e Amazon Cognito.

Utilizzo dei dati in streaming da Amazon Kinesis

Argomenti

Generazione di dati fittizi con Kinesis Data Generator

> **Note**
>
> Se hai già completato i passaggi del precedente Tutorial: creazione del primo carico di lavoro di streaming utilizzando AWS Glue Studio e hai già installato Kinesis Data Generator sull'account, puoi saltare i passaggi da 1 a 8 riportati di seguito e andare direttamente alla sezione Creazione di un processo di streaming di AWS Glue con AWS Glue Studio.

È possibile generare sinteticamente dati di esempio in formato JSON utilizzando Kinesis Data Generator (KDG). Puoi trovare le istruzioni complete e i dettagli nella documentazione dello strumento.

1. Per iniziare, fai clic su

   per eseguire un modello AWS CloudFormation nel tuo ambiente AWS.

> **Note**
>
> Potresti riscontrare un errore nel modello CloudFormation perché alcune risorse, come l'utente Amazon Cognito per Kinesis Data Generator, esistono già nel tuo account AWS. Ciò potrebbe essere dovuto al fatto che l'hai già configurato in un altro tutorial o da un post di un blog. Per risolvere questo problema, puoi provare a utilizzare il modello in un nuovo account AWS oppure in un'altra regione AWS. Queste opzioni consentono di eseguire il tutorial senza entrare in conflitto con le risorse esistenti.

Il modello fornisce un flusso di dati Kinesis e un account Kinesis Data Generator.

2. Immetti un Nome utente e una Password che KDG utilizzerà per l'autenticazione. Prendi nota del nome utente e della password per utilizzarli in seguito.
3. Seleziona Avanti fino all'ultimo passaggio. Esprimi il consenso alla creazione di risorse IAM. Verifica la presenza di eventuali errori nella parte superiore dello schermo, ad esempio la password che non soddisfa i requisiti minimi, e implementa il modello.
4. Vai alla scheda Output dello stack. Una volta implementato, il modello mostrerà la proprietà generata KinesisDataGeneratorUrl. Fai clic su quell'URL.
5. Inserisci il Nome utente e la Password di cui hai preso nota.
6. Seleziona la regione che stai utilizzando e seleziona il flusso Kinesis `GlueStreamTest-{AWS::AccountId}`.
7. Immetti il seguente modello:

```
{
    "ventilatorid": {{random.number(100)}},
    "eventtime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
    "serialnumber": "{{random.uuid}}",
    "pressurecontrol": {{random.number(
        {
            "min":5,
            "max":30
        }
    )}},
    "o2stats": {{random.number(
        {
            "min":92,
            "max":98
```

```
        }
    )}},
    "minutevolume": {{random.number(
        {
            "min":5,
            "max":8
        }
    )}},
    "manufacturer": "{{random.arrayElement(
        ["3M", "GE","Vyaire", "Getinge"]
    )}}"
}
```

Ora puoi visualizzare i dati fittizi con Modello di prova e importare i dati fittizi in Kinesis con Invia dati.

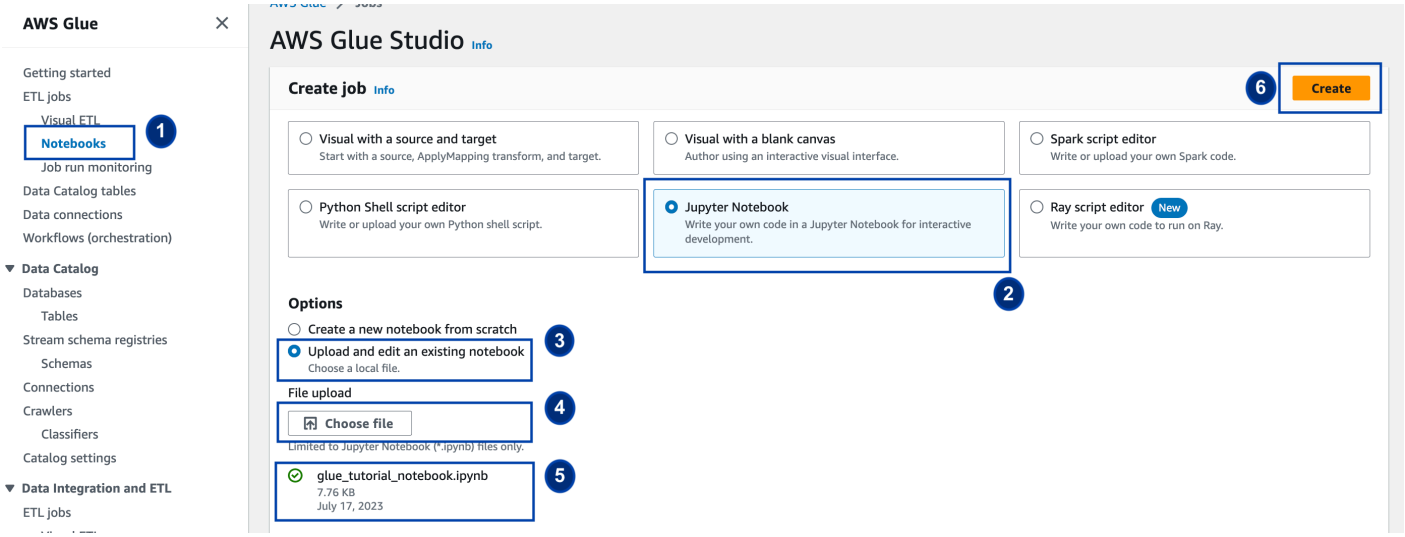
8. Fai clic su Invia dati e genera 5-10.000 record su Kinesis.

Creazione di un processo di streaming di AWS Glue con AWS Glue Studio

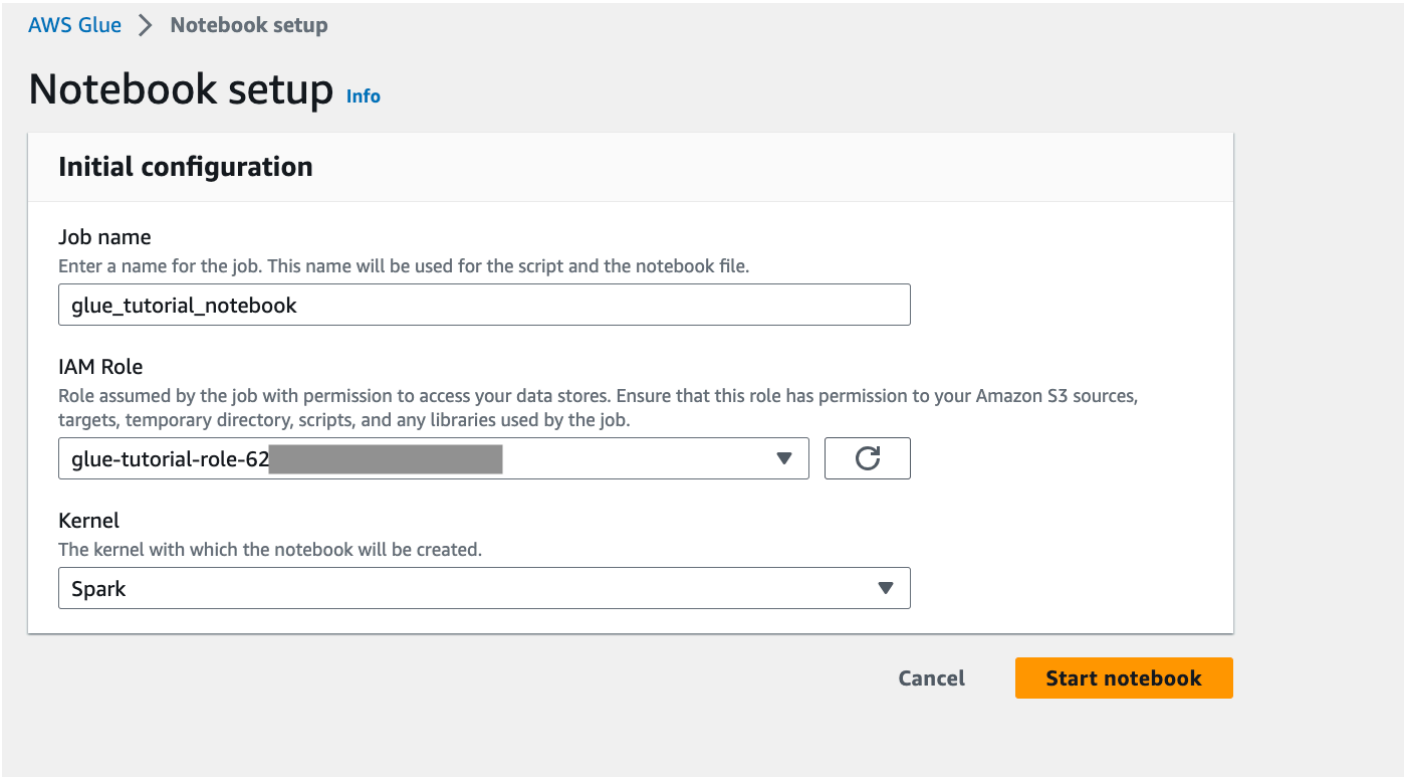
AWS Glue Studio è un'interfaccia visiva che semplifica il processo di progettazione, orchestrazione e monitoraggio delle pipeline di integrazione dei dati. Consente agli utenti di creare pipeline di trasformazione dei dati senza scrivere codice esteso. Oltre all'esperienza di creazione visiva dei processi, AWS Glue Studio include anche un notebook Jupyter supportato dalle sessioni interattive di AWS Glue che utilizzerai nel resto di questo tutorial.

Configurazione del processo delle sessioni interattive di AWS Glue Streaming

1. Scarica il file del notebook fornito e salvalo in una directory locale
2. Apri la console AWS Glue e nel riquadro sinistro fai clic su Notebook > Notebook Jupyter > Carica e modifica un notebook esistente. Carica il notebook dal passaggio precedente e fai clic su Crea.



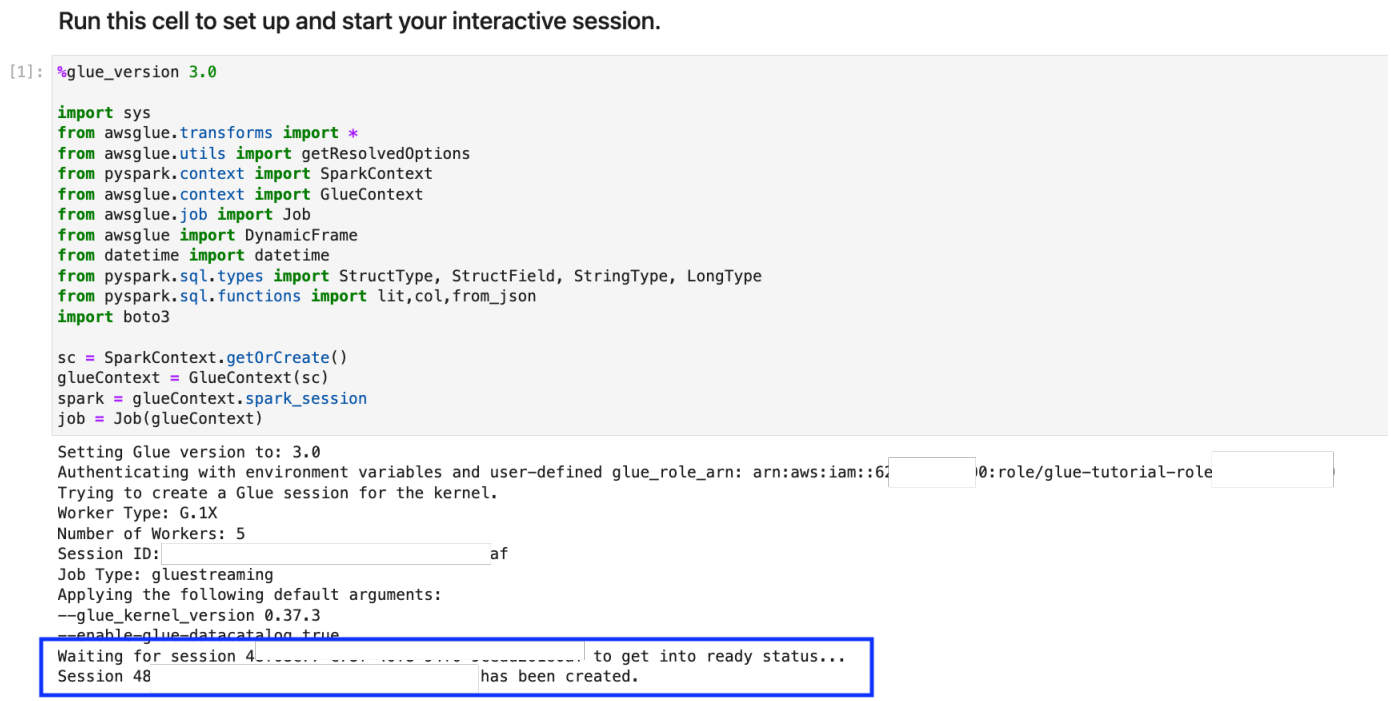
3. Fornisci un nome e un ruolo per il processo e seleziona il kernel Spark predefinito. Quindi fai clic su Avvia notebook. Per Ruolo IAM, seleziona il ruolo fornito dal modello CloudFormation. Puoi visualizzarlo nella scheda Output di CloudFormation.



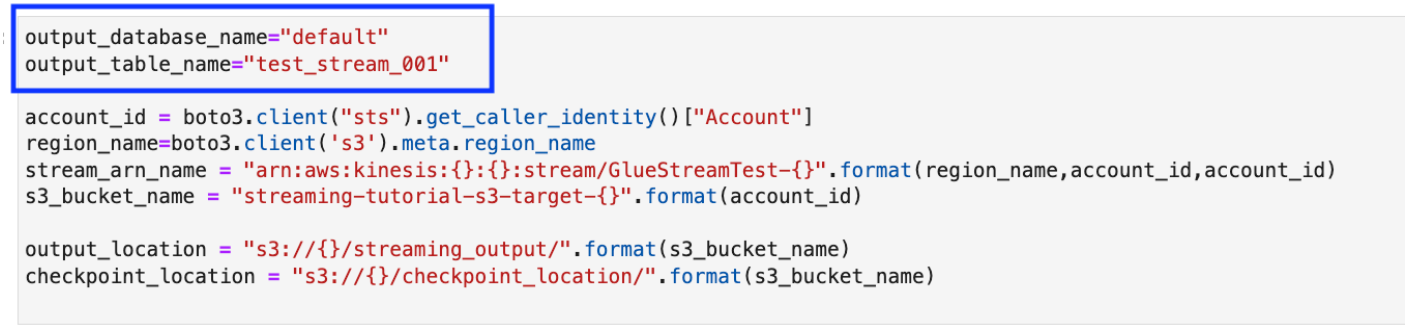
Il notebook contiene tutte le istruzioni necessarie per continuare il tutorial. Puoi eseguire le istruzioni sul notebook o seguire questo tutorial per continuare con lo sviluppo del processo.

Esecuzione delle celle del notebook

1. (Facoltativo) La prima cella di codice, `%help`, elenca tutte le funzioni magic disponibili per il notebook. Per ora puoi saltare questa cella, ma se desideri puoi esplorarla.
2. Inizia con il blocco di codice successivo, `%streaming`. Questa funzione magic imposta il tipo di processo sullo streaming, che consente di sviluppare, eseguire il debug e implementare un processo ETL in streaming di AWS Glue.
3. Esegui la cella successiva per creare una sessione interattiva di AWS Glue. La cella di output contiene un messaggio che conferma la creazione della sessione.



4. La cella successiva definisce le variabili. Sostituisci i valori con quelli appropriati per il tuo processo ed esegui la cella. Per esempio:



5. Poiché i dati vengono già trasmessi in streaming a Flussi di dati Kinesis, la cella successiva utilizzerà i risultati del flusso. Esegui la cella successiva. Poiché non ci sono istruzioni di stampa, non è previsto alcun output per questa cella.
6. Nella cella seguente, esplori il flusso in entrata prelevando un set di esempio e stampandone lo schema e i dati effettivi. Per esempio:

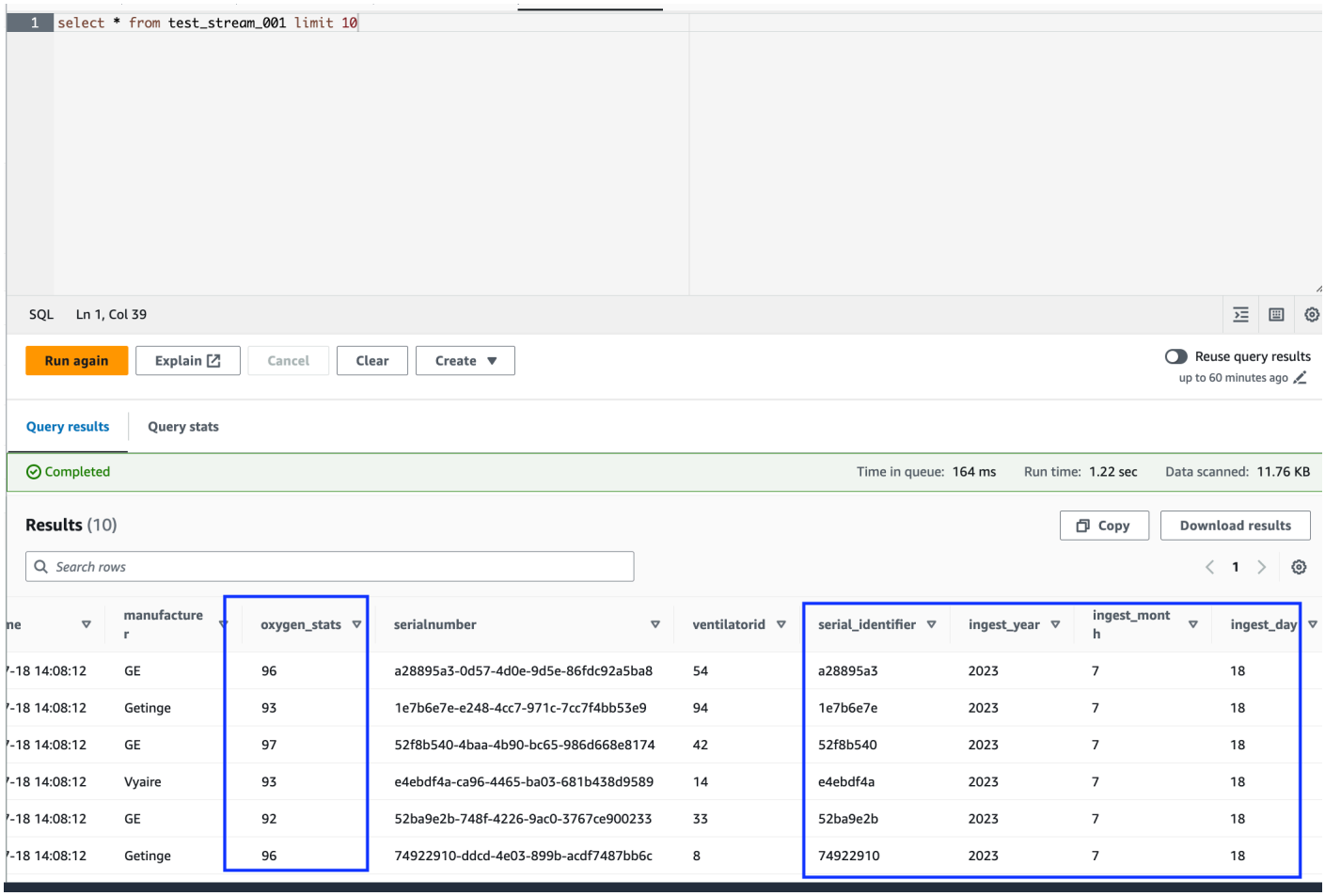
Sample and print the incoming records

the sampling is for debugging purpose. You may comment off the entire code cell below, before deploying the actual code

```
[4]: options = {
        "pollingTimeInMs": "20000",
        "windowSize": "5 seconds"
     }
     sampled_dynamic_frame = glueContext.getSampleStreamingDynamicFrame(data_frame, options, None)

     count_of_sampled_records = sampled_dynamic_frame.count()

     print(count_of_sampled_records)

     sampled_dynamic_frame.printSchema()

     sampled_dynamic_frame.toDF().show(10,False)
```

```
100
root
|-- eventtime: string
|-- manufacturer: string
|-- minutevolume: long
|-- o2stats: long
|-- pressurecontrol: long
|-- serialnumber: string
|-- ventilatorid: long

+-------------------+------------+------------+-------+---------------+------------------------------------+------------+
|eventtime          |manufacturer|minutevolume|o2stats|pressurecontrol|serialnumber                        |ventilatorid|
+-------------------+------------+------------+-------+---------------+------------------------------------+------------+
|2023-07-18 10:20:11|3M          |6           |92     |24             |a3e860ba-24b9-41c4-bc10-91c6b35e1406|6           |
|2023-07-18 10:20:11|Vyaire      |6           |95     |6              |96101dca-3e88-457f-b390-e3291df48a81|26          |
|2023-07-18 10:20:12|Getinge     |8           |96     |24             |18f3d448-1dee-4c80-835b-1a0daa818915|22          |
|2023-07-18 10:20:12|Getinge     |7           |98     |30             |25f425cd-b978-4953-9a03-4d607a639364|91          |
|2023-07-18 10:20:12|GE          |5           |93     |25             |2cd7cdc2-f5f5-4ff2-ae32-45e5a8922d53|93          |
```

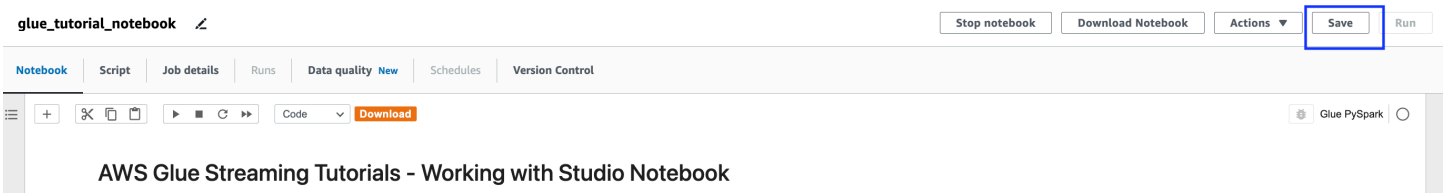
7. Successivamente, definisci la logica di trasformazione dei dati effettiva. La cella è costituita dal metodo `processBatch` che viene attivato durante ogni microbatch. Esegui la cella. Ad alto livello, eseguiamo le operazioni seguenti per il flusso in entrata:
   a. Seleziona un sottoinsieme delle colonne di input.
   b. Rinomina una colonna (o2stats in oxygen_stats).
   c. Ricava nuove colonne (serial_identifier, ingest_year, ingest_month e ingest_day).
   d. Archivia i risultati in un bucket Amazon S3 e crea anche una tabella del catalogo di AWS Glue partizionata.
8. Nell'ultima cella, il batch di processo si attiva ogni 10 secondi. Esegui la cella e attendi circa 30 secondi affinché compili il bucket Amazon S3 e la tabella del catalogo AWS Glue.
9. Infine, esplora i dati archiviati utilizzando l'editor di query di Amazon Athena. Puoi visualizzare la colonna rinominata e le nuove partizioni.



Il notebook contiene tutte le istruzioni necessarie per continuare il tutorial. Puoi eseguire le istruzioni sul notebook o seguire questo tutorial per continuare con lo sviluppo del processo.

Salvataggio ed esecuzione del processo AWS Glue

Una volta completato lo sviluppo e il test dell'applicazione utilizzando il notebook delle sessioni interattive, fai clic su Salva nella parte superiore dell'interfaccia del notebook. Una volta salvata l'applicazione, puoi anche eseguirla come processo.



Eliminazione

Per evitare addebiti aggiuntivi sul tuo account, interrompi il processo di streaming che hai avviato seguendo le istruzioni. Puoi farlo arrestando il notebook, operazione che termina la sessione. Svuota il bucket Amazon S3 ed elimina lo stack AWS CloudFormation di cui hai effettuato il provisioning in precedenza.

Conclusioni

In questo tutorial, abbiamo dimostrato come eseguire le operazioni seguenti tramite il notebook AWS Glue Studio:

- Creazione di un processo di ETL in streaming utilizzando i notebook
- Visualizzazione in anteprima dei flussi di dati in entrata
- Codifica e risoluzione dei problemi senza dover pubblicare processi di AWS Glue
- Revisione di ogni porzione del codice in uso, rimozione di eventuali errori di debug e stampa delle istruzioni o delle celle del notebook
- Pubblicazione del codice come processo di AWS Glue

L'obiettivo di questo tutorial è fornirti un'esperienza di utilizzo pratica di AWS Glue Streaming e delle sessioni interattive. Ti invitiamo a utilizzarlo come riferimento per i tuoi casi d'uso specifici di AWS Glue Streaming. Per ulteriori informazioni, consulta Nozioni di base sulle sessioni interattive AWS Glue.

Concetti relativi ad AWS Glue Streaming

Nelle sezioni seguenti vengono fornite informazioni sui concetti relativi ad AWS Glue Streaming.

Argomenti

Anatomia di un processo di streaming di AWS Glue

I processi di streaming di AWS Glue si basano sul paradigma dello streaming Spark e sfruttano lo streaming strutturato del framework Spark. I processi di streaming effettuano costantemente il polling dall'origine dati di streaming a un intervallo di tempo specifico per recuperare i record sotto forma di microbatch. Le sezioni seguenti esaminano le diverse parti di un processo di streaming di AWS Glue.

forEachBatch

Il metodo `forEachBatch` è il punto di ingresso dell'esecuzione di un processo di streaming di AWS Glue. I processi di streaming di AWS Glue utilizzano il metodo `forEachBatch` per eseguire il polling dei dati. Esso funziona in modo iterativo, rimanendo attivo durante il ciclo di vita del processo di streaming, interrogando regolarmente l'origine di streaming alla ricerca di nuovi dati ed elaborando i dati più recenti in microbatch.

```
glueContext.forEachBatch(
    frame=dataFrame_AmazonKinesis_node1696872487972,
    batch_function=processBatch,
    options={
```

```
        "windowSize": "100 seconds",
        "checkpointLocation": args["TempDir"] + "/" + args["JOB_NAME"] + "/
checkpoint/",
    },
)
```

Configura la proprietà `frame` di `forEachBatch` per specificare un'origine di streaming. In questo esempio, il nodo di origine creato nel canvas vuoto durante la creazione del processo viene popolato con il DataFrame predefinito del processo. Imposta la proprietà `batch_function` come `function` che decidi di richiamare per ogni operazione di microbatch. Per gestire la trasformazione in batch sui dati in entrata è necessario definire una funzione.

Origine

Nella prima fase della funzione `processBatch`, il programma verifica il conteggio dei record del DataFrame che hai definito come proprietà frame di `forEachBatch`. Il programma aggiunge un timestamp di importazione a un DataFrame non vuoto. La clausola `data_frame.count()>0` determina se l'ultimo microbatch non è vuoto ed è pronto per un'ulteriore elaborazione.

```
def processBatch(data_frame, batchId):
    if data_frame.count() >0:
        AmazonKinesis_node1696872487972 = DynamicFrame.fromDF(
            glueContext.add_ingestion_time_columns(data_frame, "hour"),
            glueContext,
            "from_data_frame",
        )
```

Mapping

La sezione successiva del programma consiste nell'applicare la mappatura. Il metodo `Mapping.apply` su un DataFrame Spark consente di definire una regola di trasformazione per gli elementi di dati. In genere è possibile rinominare, modificare il tipo di dati o applicare una funzione personalizzata alla colonna di dati di origine e mapparli alle colonne di destinazione.

```
    #Script generated for node ChangeSchema
    ChangeSchema_node16986872679326 = ApplyMapping.apply(
        frame =  AmazonKinesis_node1696872487972,
```

```
        mappings = [
            ("eventtime", "string", "eventtime", "string"),
            ("manufacturer", "string", "manufacturer", "string"),
            ("minutevolume", "long", "minutevolume", "int"),
            ("o2stats", "long", "OxygenSaturation", "int"),
            ("pressurecontrol", "long", "pressurecontrol", "int"),
            ("serialnumber", "string", "serialnumber", "string"),
            ("ventilatorid", "long", "ventilatorid", "long"),
            ("ingest_year", "string", "ingest_year", "string"),
            ("ingest_month", "string", "ingest_month", "string"),
            ("ingest_day", "string", "ingest_day", "string"),
            ("ingest_hour", "string", "ingest_hour", "string"),
        ],
        transformation_ctx="ChangeSchema_node16986872679326",
    )
        )
```

Sink

In questa sezione, il set di dati in entrata dall'origine di streaming viene archiviato in una posizione di destinazione. In questo esempio scriveremo i dati in una posizione Amazon S3. I dettagli della proprietà `AmazonS3_node_path` sono precompilati in base alle impostazioni utilizzate durante la creazione del processo dal canvas. È possibile impostare `updateBehavior` in base al proprio caso d'uso e decidere di non aggiornare la tabella del Catalogo dati, creare il Catalogo dati e aggiornare il relativo schema nelle esecuzioni successive oppure creare una tabella di catalogo e non aggiornare la definizione dello schema nelle esecuzioni successive.

La proprietà `partitionKeys` definisce l'opzione della partizione di archiviazione. Il comportamento predefinito consiste nel partizionare i dati in base al valore `ingestion_time_columns` fornito nella sezione di origine. La proprietà `compression` consente di impostare l'algoritmo di compressione da applicare durante la scrittura della destinazione. È possibile impostare la tecnica di compressione su Snappy, LZO o GZIP. La proprietà `enableUpdateCatalog` controlla se la tabella del catalogo AWS Glue deve essere aggiornata. Le opzioni disponibili per questa proprietà sono `True` o `False`.

```
    #Script generated for node Amazon S3
    AmazonS3_node1696872743449 = glueContext.getSink(
        path =  AmazonS3_node1696872743449_path,
        connection_type = "s3",
        updateBehavior = "UPDATE_IN_DATABASE",
```

```
        partitionKeys = ["ingest_year", "ingest_month", "ingest_day", "ingest_hour"],
        compression = "snappy",
        enableUpdateCatalog = True,
        transformation_ctx = "AmazonS3_node1696872743449",
    )
```

Sink del Catalogo AWS Glue

Questa sezione del processo controlla il comportamento di aggiornamento della tabella del Catalogo AWS Glue. Imposta le proprietà `catalogDatabase` e `catalogTableName` in base al nome del database del Catalogo AWS Glue e al nome della tabella associata al processo AWS Glue che stai progettando. È possibile definire il formato di file dei dati di destinazione tramite la proprietà `setFormat`. Per questo esempio, i dati verranno archiviati in formato Parquet.

Una volta configurato ed eseguito il processo di streaming di AWS Glue che fa riferimento a questo tutorial, i dati di streaming prodotti in Amazon Kinesis Data Streams verranno archiviati nella posizione di Amazon S3 in formato Parquet con compressione Snappy. Una volta eseguito correttamente il processo di streaming, potrai interrogare i dati tramite Amazon Athena.

```
    AmazonS3_node1696872743449 = setCatalogInfo(
        catalogDatabase =  "demo", catalogTableName = "demo_stream_transform_result"
    )
    AmazonS3_node1696872743449.setFormat("glueparquet")
    AmazonS3_node1696872743449.writeFormat("ChangeSchema_node16986872679326")
    )
```

Connessioni Kafka

Indica una connessione a un cluster Kafka o a un cluster Amazon Managed Streaming for Apache Kafka.

Puoi utilizzare i metodi seguenti sotto l'oggetto `GlueContext` per utilizzare i registri da un'origine di streaming Kafka:

- `getCatalogSource`

- `getSource`
- `getSourceWithFormat`
- `createDataFrameFromOptions`

Se utilizzi `getCatalogSource`, il processo ha le informazioni sul nome della tabella e del database del catalogo dati e può utilizzarle per ottenere alcuni parametri di base per la lettura dal flusso Apache Kafka. Se utilizzi `getSource`, `getSourceWithFormat` o `createDataFrameFromOptions`, è necessario specificare esplicitamente questi parametri:

Puoi specificare queste opzioni utilizzando `connectionOptions` con `getSource` o `createDataFrameFromOptions`, `options` con `getSourceWithFormat` o `additionalOptions` con `getCatalogSource`.

Per osservazioni e restrizioni sui processi ETL dei flussi di dati, consulta la pagina the section called "Streaming di note e restrizioni ETL".

Configurazione di Kafka

Non sussistono prerequisiti AWS per la connessione ai flussi di Kafka disponibili su Internet.

È possibile creare una connessione AWS Glue Kafka per gestire le proprie credenziali di connessione. Per ulteriori informazioni, consulta the section called "Creazione di una connessione per un flusso di dati Kafka". Nella configurazione del processo AWS Glue, fornisci *`connectionName`* come Connessione di rete aggiuntiva, quindi, nella chiamata al metodo, fornisci *`connectionName`* al parametro `connectionName`.

In alcuni casi, è necessario configurare ulteriori prerequisiti:

- Se utilizzi Streaming gestito da Amazon per Apache Kafka con l'autenticazione IAM, avrai bisogno di una configurazione appropriata di IAM.
- Se utilizzi Streaming gestito da Amazon per Apache Kafka con un Amazon VPC, avrai bisogno di una configurazione appropriata di Amazon VPC. Dovrai creare una connessione ad AWS Glue che fornisca informazioni sulla connessione ad Amazon VPC. È necessario che la configurazione del processo includa la connessione AWS Glue come connessione di rete aggiuntiva.

Per ulteriori informazioni sui prerequisiti dei processi ETL dei flussi di dati, consulta la pagina the section called "Aggiunta di processi di streaming ETL".

Esempio: lettura di flussi da Kafka

Usato in combinazione con the section called “forEachBatch”.

Esempio per l'origine di streaming Kafka:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

Indicazioni di riferimento alle opzioni di connessione a Kafka

Utilizzare le seguenti opzioni di connessione con `"connectionType": "kafka"`:

- `"bootstrap.servers"` (Obbligatorio) un elenco di URL del server bootstrap, ad esempio, come `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Questa opzione deve essere specificata nella chiamata API o definita nei metadati della tabella in catalogo dati.
- `"security.protocol"` (Obbligatorio) Il protocollo utilizzato per comunicare con i broker. I valori possibili sono `"SSL"` o `"PLAINTEXT"`.
- `"topicName"`: (obbligatorio) un elenco separato da virgole di argomenti a cui iscriversi. Devi specificare solo uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.
- `"assign"`: (obbligatorio) una stringa JSON che specifica il `TopicPartitions` specifico da utilizzare. Devi specificare solo uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.

  Esempio: '{"topicA":[0,1],"topicB":[2,4]}'
- `"subscribePattern"`: (Obbligatorio) una stringa regex Java che identifichi l'elenco degli argomenti a cui effettuare la sottoscrizione. Devi specificare solo uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.

  Esempio: 'topic.*'
- `"classification"` (obbligatorio): il formato di file utilizzato dai dati nel record. Obbligatorio, a meno che non sia fornito tramite Catalogo dati.

- `"delimiter"` (facoltativo): il separatore di valori utilizzato quando `classification` è CSV. Il valore predefinito è `","`.
- `"startingOffsets"`: (Facoltativo) la posizione di partenza nell'argomento Kafka da cui leggere i dati. I valori possibili sono `"earliest"` o `"latest"`. Il valore predefinito è `"latest"`.
- `"startingTimestamp"`: (facoltativo, supportato solo per AWS Glue versione 4.0 o successiva) il timestamp del record nell'argomento Kafka da cui leggere i dati. Il valore possibile è una stringa timestamp in formato UTC nel modello `yyyy-mm-ddTHH:MM:SSZ`, dove `Z` rappresenta un offset del fuso orario UTC con un segno +/- (ad esempio: "2023-04-04T08:00:00-04:00").

  Nota: nell'elenco delle opzioni di connessione dello script di flussi di dati di AWS Glue può essere presente solo un valore tra "startingOffsets" o "startingTimestamp"; l'inclusione di entrambe queste proprietà comporterà un errore del processo.
- `"endingOffsets"`: (Facoltativo) il punto di fine di una query batch. I valori possibili sono `"latest"` o una stringa JSON che specifica un offset finale per ogni `TopicPartition`.

  Per la stringa JSON, il formato è `{"topicA":{"0":23,"1":-1},"topicB":{"0":-1}}`. Il valore `-1` come offset rappresenta `"latest"`.
- `"pollTimeoutMs"`: (Facoltativo) il timeout in millisecondi per il polling dei dati da Kafka negli executor del processo Spark. Il valore predefinito è `512`.
- `"numRetries"`: (Facoltativo) i numero di tentativi prima di non riuscire a recuperare gli offset Kafka. Il valore predefinito è `3`.
- `"retryIntervalMs"`: (Facoltativo) il tempo di attesa in millisecondi prima di riprovare a recuperare gli offset Kafka. Il valore predefinito è `10`.
- `"maxOffsetsPerTrigger"`: (Facoltativo) il limite di velocità sul numero massimo di offset elaborati per intervallo di trigger. Il numero totale di offset specificato viene suddiviso proporzionalmente tra `topicPartitions` di diversi volumi. Il valore di default è null, il che significa che il consumer legge tutti gli offset fino all'ultimo offset noto.
- `"minPartitions"`: (Facoltativo) il numero minimo desiderato di partizioni da leggere da Kafka. Il valore di default è null, il che significa che il numero di partizioni Spark è uguale al numero di partizioni Kafka.
- `"includeHeaders"`: (Facoltativo) indica se includere le intestazioni Kafka. Quando l'opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "glue_streaming_kafka_headers" con tipo `Array[Struct(key: String, value: String)]`. Il valore di default è "false". Questa opzione è disponibile in AWS Glue versione 3.0 o successive.

- `"schema"`: (obbligatorio quando inferSchema è impostato su false) lo schema da utilizzare per elaborare il payload. Se la classificazione è `avro`, lo schema fornito dovrà essere nel formato dello schema Avro. Se la classificazione è `avro`, lo schema fornito dovrà essere nel formato dello schema DDL.

  Di seguito sono riportati alcuni esempi di schema.

  Example in DDL schema format

  ```
  'column1' INT, 'column2' STRING , 'column3' FLOAT
  ```

  Example in Avro schema format

  ```
  {
  "type":"array",
  "items":
  {
  "type":"record",
  "name":"test",
  "fields":
  [
    {
      "name":"_id",
      "type":"string"
    },
    {
      "name":"index",
      "type":
      [
        "int",
        "string",
        "float"
      ]
    }
  ]
  }
  }
  ```

- `"inferSchema"`: (facoltativo) il valore di default è "false". Se impostato su "true", lo schema verrà rilevato in fase di runtime dal payload all'interno di `foreachbatch`.
- `"avroSchema"`: (obsoleto) parametro utilizzato per specificare uno schema di dati Avro quando viene utilizzato il formato Avro. Questo parametro è obsoleto. Utilizzo del parametro `schema`.

- `"addRecordTimestamp"`: (Facoltativo) Quando questa opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "__src_timestamp" che indica l'ora in cui il record corrispondente è stato ricevuto dall'argomento. Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.
- `"emitConsumerLagMetrics"`: (Facoltativo) Quando questa opzione è impostata su "true", per ogni batch, emetterà i parametri relativi alla durata tra il record più vecchio ricevuto dall'argomento e l'ora in cui arriva in AWS Glue a CloudWatch. Il nome del parametro è "glue.driver.streaming.maxConsumerLaginMs". Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.

Connessioni Kinesis

Puoi leggere e scrivere su flussi di dati Amazon Kinesis utilizzando le informazioni archiviate in una tabella catalogo dati o fornendo informazioni per accedere direttamente al flusso di dati. Puoi leggere le informazioni da Kinesis in Spark DataFrame, quindi convertirle in un Glue. AWS DynamicFrame Puoi DynamicFrames scrivere su Kinesis in formato JSON. Se accedi direttamente al flusso di dati, utilizza queste opzioni per fornire le informazioni su come accedere al flusso di dati.

Se utilizzi `getCatalogSource` o `create_data_frame_from_catalog` per consumare i registri da una sorgente di streaming Kinesis, il processo avrà le informazioni sul database catalogo dati e sul nome della tabella, e potrà usarle per ottenere alcuni parametri di base per la lettura dalla sorgente di streaming Kinesis. Se utilizzi `getSource`, `getSourceWithFormat`, `createDataFrameFromOptions` o `create_data_frame_from_options`, dovrai specificare questi parametri di base utilizzando le opzioni di connessione descritte qui.

È possibile specificare le opzioni di connessione per Kinesis utilizzando i seguenti argomenti per i metodi specificati nella classe `GlueContext`.

- Scala
  - `connectionOptions`: utilizza con `getSource`, `createDataFrameFromOptions` e `getSink`
  - `additionalOptions`: utilizza con `getCatalogSource`, `getCatalogSink`
  - `options`: utilizza con `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: utilizza con `create_data_frame_from_options`, `write_dynamic_frame_from_options`

- `additional_options`: utilizza con `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
- `options`: utilizza con `getSource`, `getSink`

Per osservazioni e restrizioni sui processi ETL dei flussi di dati, consulta la pagina the section called “Streaming di note e restrizioni ETL”.

Configurazione di Kinesis

Per connetterti a un flusso di dati Kinesis in un job AWS Glue Spark, avrai bisogno di alcuni prerequisiti:

- In caso di lettura, il job AWS Glue deve disporre delle autorizzazioni IAM di livello di accesso Read per il flusso di dati Kinesis.
- In fase di scrittura, il job AWS Glue deve disporre delle autorizzazioni IAM di livello di accesso Write per il flusso di dati Kinesis.

In alcuni casi, è necessario configurare ulteriori prerequisiti:

- Se il tuo job AWS Glue è configurato con connessioni di rete aggiuntive (in genere per connettersi ad altri set di dati) e una di queste connessioni offre opzioni di rete Amazon VPC, questo indirizzerà il tuo lavoro a comunicare tramite Amazon VPC. In questo caso, per comunicare tramite Amazon VPC dovrai configurare anche il flusso di dati Kinesis. È possibile farlo creando un endpoint VPC di interfaccia tra l'Amazon VPC e il flusso di dati Kinesis. Per ulteriori informazioni, consulta la pagina Using Amazon Kinesis Data Streams with Interface VPC Endpoints.
- Quando si specifica un flusso di dati Amazon Kinesis in un altro account, è necessario impostare i ruoli e le policy per consentire l'accesso multi-account. Per ulteriori informazioni, consulta Esempio: lettura da un flusso Kinesis in un account diverso.

Per ulteriori informazioni sui prerequisiti dei processi ETL dei flussi di dati, consulta la pagina the section called “Aggiunta di processi di streaming ETL”.

Lettura da Kinesis

Esempio: lettura da flussi Kinesis

Usato in combinazione con the section called “forEachBatch”.

Esempio per l'origine di streaming Amazon Kinesis:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Scrittura su Kinesis

Esempio: scrittura su flussi Kinesis

Usato in combinazione con the section called “forEachBatch”. Il tuo DynamicFrame verrà scritto nello stream in formato JSON. Se il processo non riesce a scrivere dopo diversi tentativi, riporterà un errore. Per impostazione predefinita, ogni DynamicFrame record viene inviato allo stream Kinesis singolarmente. È possibile configurare questo comportamento utilizzando `aggregationEnabled` e i parametri associati.

Esempio di scrittura su Amazon Kinesis da un processo di streaming:

Python

```
glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite
    connection_type="kinesis",
    connection_options={
        "partitionKey": "part1",
        "streamARN": "arn:aws:kinesis:us-east-1:111122223333:stream/streamName",
    }
)
```

Scala

```
glueContext.getSinkWithFormat(
                connectionType="kinesis",
                options=JsonOptions("""{
                    "streamARN": "arn:aws:kinesis:us-
east-1:111122223333:stream/streamName",
```

```
                    "partitionKey": "part1"
                }"""),
            )
            .writeDynamicFrame(frameToWrite)
```

Parametri di connessione Kinesis

Indica le opzioni di connessione ad Amazon Kinesis Data Streams.

Utilizza le seguenti opzioni di connessione per le origini dati in streaming Kinesis:

- `"streamARN"`: (obbligatorio) utilizzato per la lettura/scrittura. L'ARN del flusso di dati di Kinesis.
- `"classification"`: (obbligatorio per la lettura) utilizzato per la lettura. Il formato di file utilizzato dai dati nel record. Obbligatorio, a meno che non sia fornito tramite Catalogo dati.
- `"streamName"`: (facoltativo) utilizzato per la lettura. Il nome di un flusso di dati Kinesis da cui leggere. Usato con `endpointUrl`.
- `"endpointUrl"`: (facoltativo) utilizzato per la lettura. Predefinito: "https://kinesis.us-east-1.amazonaws.com". L' AWS endpoint del flusso Kinesis. Non è necessario modificarlo a meno che non ci si stia connettendo a una regione speciale.
- `"partitionKey"`: (facoltativo) utilizzato per la scrittura. La chiave di partizione di Kinesis utilizzata per la produzione dei record.
- `"delimiter"`: (facoltativo) utilizzato per la lettura. Il separatore di valori utilizzato quando `classification` è CSV. Il valore predefinito è `","`.
- `"startingPosition"`: (facoltativo) utilizzato per la lettura. La posizione di partenza nel flusso dei dati Kinesis da cui leggere i dati. I valori possibili sono `"latest"`, `"trim_horizon"`, `"earliest"` o una stringa di timestamp in formato UTC con il modello `yyyy-mm-ddTHH:MM:SSZ`, dove `Z` rappresenta uno scostamento del fuso orario UTC con un +/- (ad esempio: "2023-04-04T08:00:00-04:00"). Il valore predefinito è `"latest"`. Nota: la stringa Timestamp in formato UTC per `"startingPosition"` è supportata solo per AWS Glue versione 4.0 o successiva.
- `"failOnDataLoss"`: (facoltativo) non è possibile eseguire il processo se una partizione attiva è mancante o scaduta. Il valore predefinito è `"false"`.
- `"awsSTSRoleARN"`: (facoltativo) utilizzato per la lettura/scrittura. L'Amazon Resource Name (ARN) del ruolo da assumere utilizzando AWS Security Token Service (AWS STS). Questo ruolo deve disporre delle autorizzazioni per descrivere o leggere le operazioni dei registri per il flusso di

dati Kinesis. Quando si accede a un flusso di dati in un altro account, è necessario utilizzare questo parametro. Usato in combinazione con `"awsSTSSessionName"`.

- `"awsSTSSessionName"`: (facoltativo) utilizzato per la lettura/scrittura. Un identificatore della sessione che assume il ruolo usando AWS STS. Quando si accede a un flusso di dati in un altro account, è necessario utilizzare questo parametro. Usato in combinazione con `"awsSTSRoleARN"`.
- `"awsSTSEndpoint"`: (Facoltativo) L' AWS STS endpoint da utilizzare quando ci si connette a Kinesis con un ruolo presunto. Ciò consente di utilizzare l' AWS STS endpoint regionale in un VPC, cosa non possibile con l'endpoint globale predefinito.
- `"maxFetchTimeInMs"`: (facoltativo) utilizzato per la lettura. Il tempo massimo impiegato nell'esecutore del processo per recuperare un registro dal flusso dei dati Kinesis per partizione, specificato in millisecondi (ms). Il valore predefinito è `1000`.
- `"maxFetchRecordsPerShard"`: (facoltativo) utilizzato per la lettura. Il numero massimo di record da recuperare per shard nel flusso di dati Kinesis per microbatch. Nota: il client può superare questo limite se il job di streaming ha già letto record aggiuntivi da Kinesis (nella stessa chiamata get-records). Se `maxFetchRecordsPerShard` deve essere rigoroso, deve essere un multiplo di. `maxRecordPerRead` Il valore predefinito è `100000`.
- `"maxRecordPerRead"`: (facoltativo) utilizzato per la lettura. Il numero massimo di record da recuperare nel flusso di dati Kinesis in ciascuna operazione `getRecords`. Il valore predefinito è `10000`.
- `"addIdleTimeBetweenReads"`: (facoltativo) utilizzato per la lettura. Aggiunge un ritardo tra due operazioni consecutive `getRecords`. Il valore predefinito è `"False"`. Questa opzione è configurabile solo per Glue versione 2.0 e successive.
- `"idleTimeBetweenReadsInMs"`: (facoltativo) utilizzato per la lettura. Il ritardo minimo tra due operazioni consecutive `getRecords`, specificato in ms. Il valore predefinito è `1000`. Questa opzione è configurabile solo per Glue versione 2.0 e successive.
- `"describeShardInterval"`: (facoltativo) utilizzato per la lettura. L'intervallo di tempo minimo tra due chiamate API `ListShards` affinché lo script consideri il resharding. Per ulteriori informazioni, consulta Strategie per il resharding nella Guida per gli sviluppatori di Amazon Kinesis Data Streams. Il valore predefinito è `1s`.
- `"numRetries"`: (facoltativo) utilizzato per la lettura. Il numero massimo di tentativi per le richieste API Kinesis Data Streams. Il valore predefinito è `3`.

- `"retryIntervalMs"`: (facoltativo) utilizzato per la lettura. Il periodo di raffreddamento (specificato in ms) prima di riprovare la chiamata API Kinesis Data Streams. Il valore predefinito è `1000`.
- `"maxRetryIntervalMs"`: (facoltativo) utilizzato per la lettura. Il periodo di raffreddamento (specificato in ms) tra due tentativi di chiamata API Kinesis Data Streams. Il valore predefinito è `10000`.
- `"avoidEmptyBatches"`: (facoltativo) utilizzato per la lettura. Impedisce la creazione di un processo microbatch vuoto controllando la presenza di dati non letti nel flusso dei dati Kinesis prima che il batch venga avviato. Il valore predefinito è `"False"`.
- `"schema"`: (obbligatorio quando inferSchema è impostato su falso) utilizzato per la lettura. Lo schema da utilizzare per elaborare il payload. Se la classificazione è `avro`, lo schema fornito dovrà essere nel formato dello schema Avro. Se la classificazione è `avro`, lo schema fornito dovrà essere nel formato dello schema DDL.

  Di seguito sono riportati alcuni esempi di schema.

  Example in DDL schema format

  ```
  `column1` INT, `column2` STRING , `column3` FLOAT
  ```

  Example in Avro schema format

  ```
  {
    "type":"array",
    "items":
    {
      "type":"record",
      "name":"test",
      "fields":
      [
        {
          "name":"_id",
          "type":"string"
        },
        {
          "name":"index",
          "type":
          [
            "int",
            "string",
  ```

```
          "float"
        ]
      }
    ]
  }
}
```

- `"inferSchema"`: (facoltativo) utilizzato per la lettura. Il valore predefinito è "false". Se impostato su "true", lo schema verrà rilevato in fase di runtime dal payload all'interno di `foreachbatch`.
- `"avroSchema"`: (obsoleto) utilizzato per la lettura. Parametro utilizzato per specificare uno schema di dati Avro quando viene utilizzato il formato Avro. Questo parametro è obsoleto. Utilizzo del parametro `schema`.
- `"addRecordTimestamp"`: (facoltativo) utilizzato per la lettura. Quando questa opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "__src_timestamp" che indica l'ora in cui il record corrispondente è stato ricevuto dal flusso. Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.
- `"emitConsumerLagMetrics"`: (facoltativo) utilizzato per la lettura. Quando l'opzione è impostata su «true», per ogni batch emetterà le metriche relative alla durata compresa tra il record più vecchio ricevuto dallo stream e il momento in AWS Glue cui arriva. CloudWatch Il nome della metrica è «glue.driver.streaming. maxConsumerLagInMs». Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.
- `"fanoutConsumerARN"`: (facoltativo) utilizzato per la lettura. L'ARN di un consumatore di un flusso Kinesis per il flusso specificato in `streamARN`. Utilizzato per abilitare la modalità di fan-out avanzato per la connessione Kinesis. Per ulteriori informazioni sull'utilizzo di un flusso Kinesis con fan-out avanzato, consulta la pagina the section called “Utilizzo del fan-out avanzato nei processi di flussi di dati Kinesis”.
- `"recordMaxBufferedTime"`: (facoltativo) utilizzato per la scrittura. Predefinito: 1000 (ms). Tempo massimo di memorizzazione nel buffer di un record in attesa di essere scritto.
- `"aggregationEnabled"`: (facoltativo) utilizzato per la scrittura. Default: true (VERO). Speciifica se i record devono essere aggregati prima di inviarli a Kinesis.
- `"aggregationMaxSize"`: (facoltativo) utilizzato per la scrittura. Impostazione predefinita: 51200 (byte). Se un record è superiore a questo limite, ignorerà l'aggregatore. Ricorda che Kinesis impone un limite di 50 KB alla dimensione del record. Se imposti questo valore oltre i 50 KB, i record di grandi dimensioni verranno rifiutati da Kinesis.
- `"aggregationMaxCount"`: (facoltativo) utilizzato per la scrittura. Predefinito: 4294967295. Numero massimo di voci da inserire in un record aggregato.

- `"producerRateLimit"`: (facoltativo) utilizzato per la scrittura. Predefinito: 150 (%). Limita la velocità di trasmissione effettiva per partizione inviata da un singolo produttore (ad esempio, il tuo processo), come percentuale del limite di backend.
- `"collectionMaxCount"`: (facoltativo) utilizzato per la scrittura. Predefinito: 500. Numero massimo di articoli da imballare in una PutRecords richiesta.
- `"collectionMaxSize"`: (facoltativo) utilizzato per la scrittura. Impostazione predefinita: 5242880 (byte). Quantità massima di dati da inviare con una PutRecords richiesta.

Opzioni di AWS Glue Streaming

Indica una connessione a un cluster Kafka o a un cluster Amazon Managed Streaming for Apache Kafka.

Puoi utilizzare i metodi seguenti sotto l'oggetto `GlueContext` per utilizzare i registri da un'origine di streaming Kafka:

- `getCatalogSource`
- `getSource`
- `getSourceWithFormat`
- `createDataFrameFromOptions`

Se utilizzi `getCatalogSource`, il processo ha le informazioni sul nome della tabella e del database del catalogo dati e può utilizzarle per ottenere alcuni parametri di base per la lettura dal flusso Apache Kafka. Se utilizzi `getSource`, `getSourceWithFormat` o `createDataFrameFromOptions`, è necessario specificare esplicitamente questi parametri:

Puoi specificare queste opzioni utilizzando `connectionOptions` con `getSource` o `createDataFrameFromOptions`, `options` con `getSourceWithFormat` o `additionalOptions` con `getCatalogSource`.

Per osservazioni e restrizioni sui processi ETL dei flussi di dati, consulta la pagina the section called “Streaming di note e restrizioni ETL”.

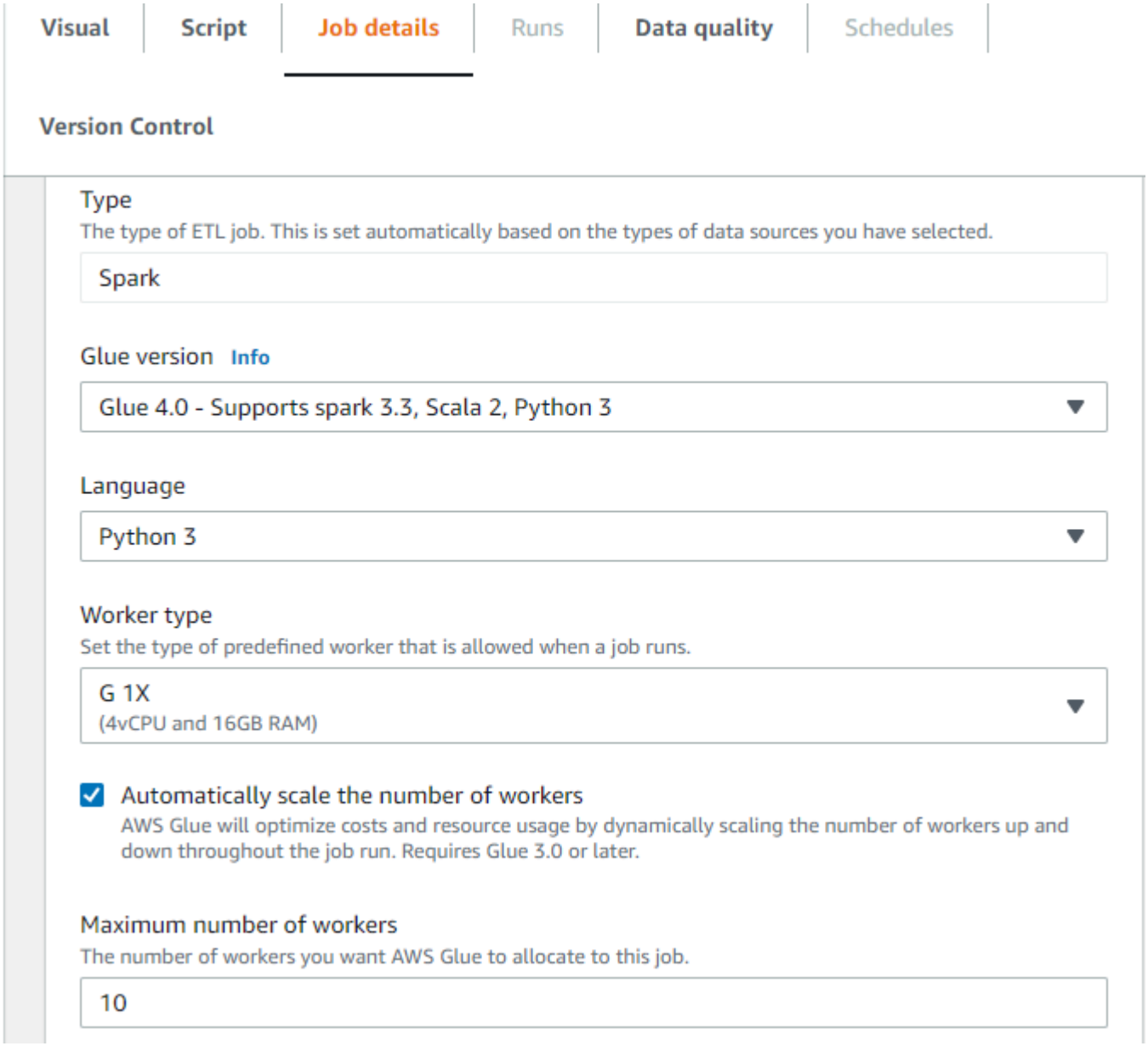
Dimensionamento automatico di AWS Glue Streaming

Nelle sezioni seguenti vengono fornite informazioni sul dimensionamento automatico dello streaming di AWS Glue

Abilitazione dell'Auto Scaling in AWS Glue Studio

Sulla scheda Job details (Dettagli processo) in AWS Glue Studio, scegli il tipo Spark o Spark Streaming e Glue version (Versione di Glue) come **`Glue 3.0`** o **`Glue 4.0`**. In seguito, una casella di controllo verrà visualizzata sotto Worker type (tipo di worker).

- Seleziona l'opzione Dimensiona automaticamente il numero di worker.
- Imposta la proprietà Numero massimo di dipendenti per definire il numero massimo di dipendenti che possono essere ceduti all'esecuzione del processo.



Abilitazione di Auto Scaling con il CLI o SDK di AWS

Per abilitare Auto Scaling dal CLI di AWS per eseguire il processo, eseguire`start-job-run` con la configurazione seguente:

```
{
    "JobName": "<your job name>",
    "Arguments": {
        "--enable-auto-scaling": "true"
    },
    "WorkerType": "G.2X", // G.1X and G.2X are allowed for Auto Scaling Jobs
    "NumberOfWorkers": 20, // represents Maximum number of workers
    ...other job run configurations...
}
```

Una volta terminata l'esecuzione del processo ETL, puoi anche chiamare`get-job-run` per verificare l'effettivo utilizzo delle risorse del processo eseguito in secondi DPU. Nota: il nuovo campo DPUSeconds verrà visualizzato solo per i processi batch su AWS Glue 3.0 o versioni successive abilitati con Dimensionamento automatico. Questo campo non è supportato per i processi di streaming.

```
$ aws glue get-job-run --job-name your-job-name --run-id jr_xx --endpoint https://
glue.us-east-1.amazonaws.com --region us-east-1
{
    "JobRun": {
        ...
        "GlueVersion": "3.0",
        "DPUSeconds": 386.0
    }
}
```

È inoltre possibile configurare le esecuzioni dei processi con Auto Scaling utilizzando l'SDKAWS Glue con la stessa configurazione.

Come funziona

Dimensionamento tra microbatch

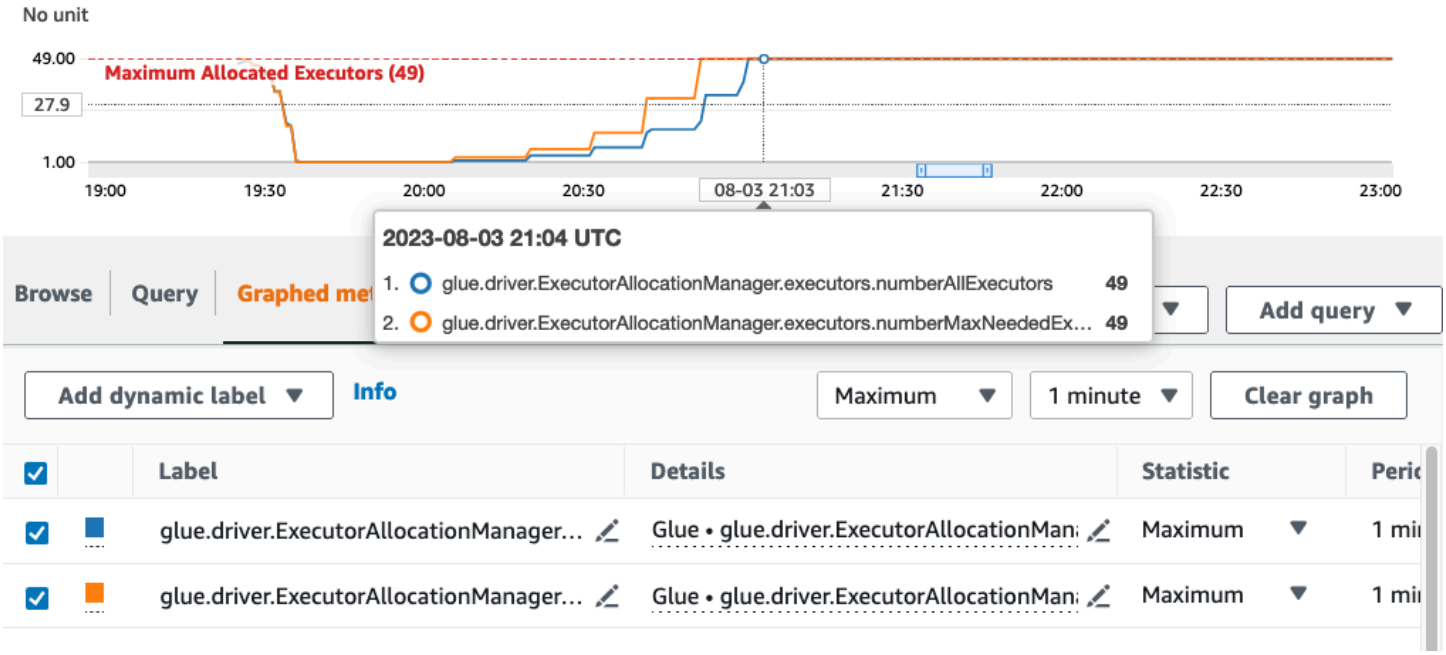
L'esempio seguente viene utilizzato per descrivere come funziona il dimensionamento automatico.

- Hai un processo AWS Glue che inizia con 50 DPU.

- Il dimensionamento automatico è abilitato.

In questo esempio, AWS Glue esamina la metrica batchProcessingTime InMs ««per alcuni micro batch e determina se i lavori vengono completati entro le dimensioni della finestra stabilite. Se i tuoi processi vengono completati prima e a seconda della tempistica con cui vengono completati, AWS Glue potrebbe ridursi. Questa metrica, tracciata con» numberAllExecutors «, può essere monitorata Amazon CloudWatch per vedere come funziona la scalabilità automatica.

Il numero di esecutori aumenta o diminuisce in modo esponenziale solo dopo il completamento di ogni microbatch. Come si può vedere dal log di monitoraggio di Amazon CloudWatch, AWS Glue analizza il numero di esecutori necessari (linea arancione) e dimensiona gli esecutori (linea blu) in modo che corrispondano automaticamente a tale numero.

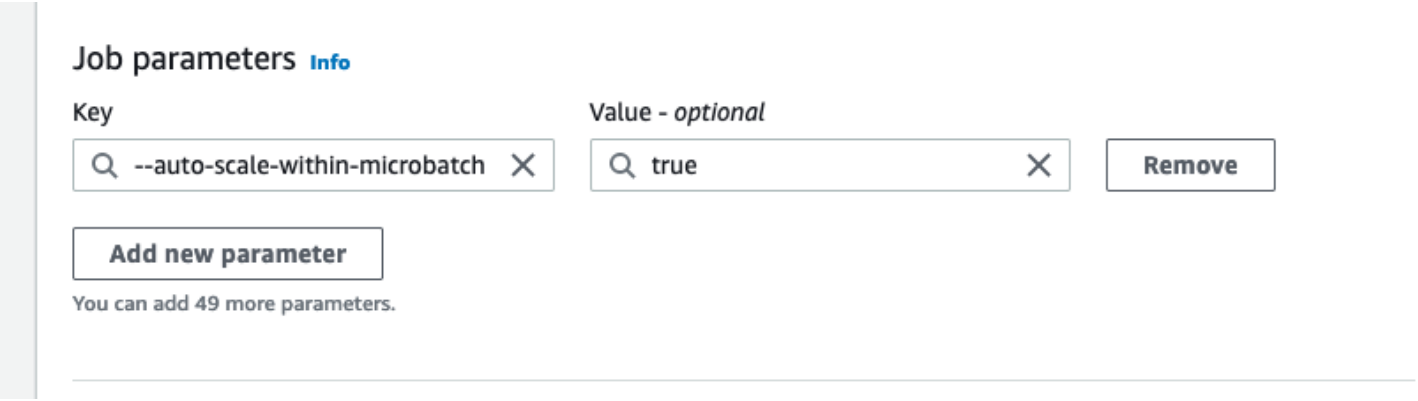


Dopo che AWS Glue riduce il numero di esecutori e rileva che il volume di dati aumenta, con conseguente aumento del tempo di elaborazione dei microbatch, AWS Glue aumenta fino a 50 DPU, che è il limite massimo specificato.

Dimensionamento all'interno di un microbatch

Nell'esempio precedente, il sistema monitora alcuni microbatch completati per decidere se eseguire un aumento o una riduzione. Finestre più lunghe richiedono la scalabilità automatica per rispondere più rapidamente all'interno del microbatch, anziché attendere alcuni microbatch. In questi casi, è possibile utilizzare la configurazione aggiuntiva `--auto-scaIe-within-microbatch` per `true`.

È possibile aggiungerla alle proprietà del processo AWS Glue in AWS Glue Studio come illustrato di seguito.



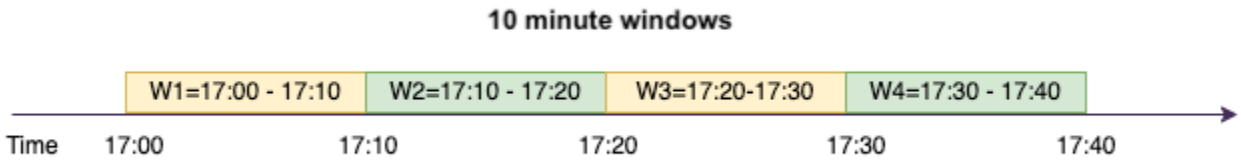
Concetti avanzati relativi allo streaming in AWS Glue

Nelle odierne applicazioni basate sui dati, l'importanza dei dati diminuisce nel tempo e il loro valore predittivo si trasforma nella possibilità di reagire. Di conseguenza, i clienti vogliono elaborare i dati in tempo reale per prendere decisioni più rapide. Quando si gestiscono feed di dati in tempo reale, ad esempio dai sensori IoT, i dati possono arrivare non ordinati o subire ritardi nell'elaborazione dovuti alla latenza della rete e ad altri errori legati all'origine durante l'importazione. Come parte della piattaforma AWS Glue, AWS Glue Streaming si basa su queste funzionalità per fornire capacità di ETL in streaming scalabili e serverless. Inoltre, si basa sullo streaming strutturato di Apache Spark, che consente agli utenti di elaborare i dati in tempo reale.

In questo argomento, esploreremo i concetti e le funzionalità di streaming avanzati di AWS Glue Streaming.

Considerazioni di carattere temporale relative all'elaborazione dei flussi

Esistono quattro nozioni di tempo relative all'elaborazione dei flussi:



- Ora dell'evento: l'ora in cui si è verificato l'evento. Nella maggior parte dei casi, questo campo è incorporato nei dati degli eventi stessi all'origine.

- E vent-time-window — L'intervallo di tempo tra due momenti dell'evento. Come mostrato nel diagramma precedente, W1 è compreso tra le 17:00 e le 17:10. event-time-window Ciascuno event-time-window è un raggruppamento di più eventi.
- Tempo di attivazione: il tempo di attivazione controlla la frequenza con cui si verificano l'elaborazione dei dati e l'aggiornamento dei risultati. Si tratta dell'ora in cui è iniziata l'elaborazione del microbatch.
- Ora di importazione: l'ora in cui i dati del flusso sono stati importati nel servizio di streaming. Se l'ora dell'evento non è incorporata nell'evento stesso, in alcuni casi può essere utilizzata per la creazione di finestre.

Raggruppamenti in finestre

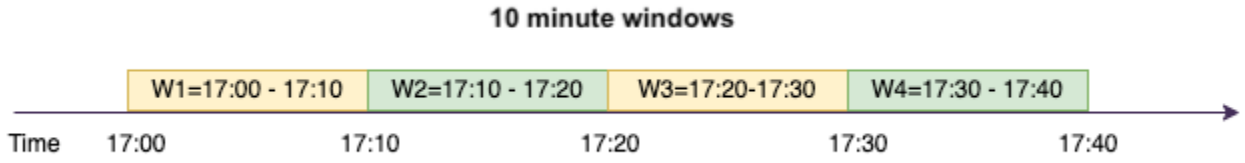
Il windowing è una tecnica che consente di raggruppare e aggregare più eventi in base a. event-time-window Esploreremo i vantaggi del windowing e le possibilità di utilizzarlo nei seguenti esempi.

A seconda del caso d'uso aziendale, Spark supporta tre tipi di finestre temporali.

- Tumbling window: una serie di dimensioni fisse non sovrapposte su cui aggregare. event-time-windows
- Finestra scorrevole: come le finestre a cascata ha dimensioni fisse, ma a differenza di esse può sovrapporsi o scorrere, a condizione che la durata dello scorrimento sia inferiore alla durata della finestra stessa.
- Finestra di sessione: inizia con un evento relativo ai dati di input e continua a espandersi fintantoché riceve input entro un intervallo di tempo o un periodo di inattività. Una finestra di sessione può avere una lunghezza fissa o dinamica a seconda degli input.

Finestra a cascata

La finestra tumbling è una serie di dimensioni fisse non sovrapposte su cui si aggregano. event-time-windows Cerchiamo di capirlo con un esempio tratto dalla realtà.



La società ABC Auto vuole lanciare una campagna di marketing per un nuovo marchio di auto sportive. Vuole scegliere la città con il maggior numero di appassionati di auto sportive. Per raggiungere questo obiettivo, pubblica sul suo sito web un breve annuncio pubblicitario di 15 secondi di presentazione dell'auto. Tutti i "clic" e la "città" corrispondente vengono registrati e trasmessi a Amazon Kinesis Data Streams. Vogliamo contare il numero di clic in una finestra di 10 minuti e raggrupparlo per città per vedere quale città registra la domanda maggiore. Di seguito è riportato l'output dell'aggregazione.

ora_inizio_finestra	ora_fine_finestra	città	clic_totali
2023-07-10 17:00:00	2023-07-10 17:10:00	Dallas	75
2023-07-10 17:00:00	2023-07-10 17:10:00	Chicago	10
2023-07-10 17:20:00	2023-07-10 17:30:00	Dallas	20
2023-07-10 17:20:00	2023-07-10 17:30:00	Chicago	50

Come spiegato sopra, questi event-time-windows sono diversi dagli intervalli di tempo di attivazione. Ad esempio, anche se l'intervallo di attivazione è ogni minuto, i risultati di output mostreranno solo finestre di aggregazione di 10 minuti non sovrapposte. Per l'ottimizzazione, è preferibile che l'intervallo di attivazione sia allineato con. event-time-window

Nella tabella precedente, nella finestra 17:00-17:10 Dallas ha registrato 75 clic mentre Chicago ha registrato 10 clic. Inoltre, per nessuna città sono presenti dati per la finestra 17:10-17:20, quindi questa finestra viene omessa.

Ora puoi eseguire ulteriori analisi su questi dati nell'applicazione di analisi a valle per determinare la città più indicata per la conduzione della campagna di marketing.

Utilizzo delle finestre a cascata in AWS Glue

1. Crea un file Amazon Kinesis Data Streams DataFrame e leggi da esso. Esempio:

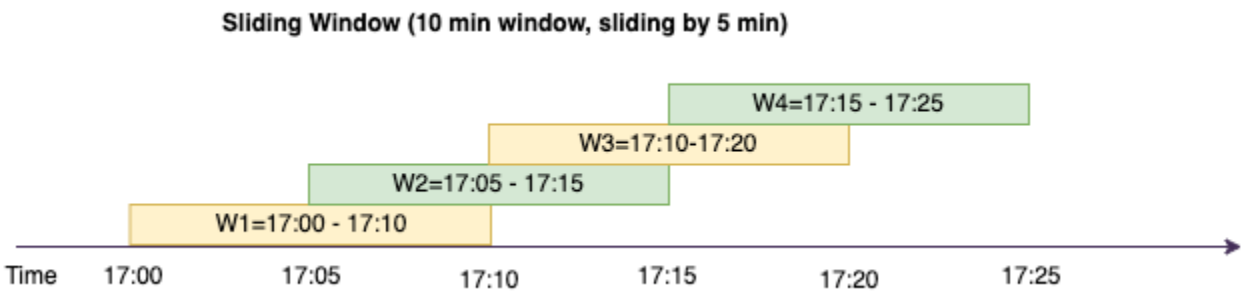
```
parsed_df = kinesis_raw_df \
                .selectExpr('CAST(data AS STRING)') \
                .select(from_json("data", ticker_schema).alias("data")) \
                .select('data.event_time','data.ticker','data.trade','data.volume',
 'data.price')
```

2. Elabora i dati in una finestra a cascata. Nell'esempio seguente, i dati vengono raggruppati in base al campo di input "ora_evento" in finestre a cascata di 10 minuti e l'output viene scritto in un data lake Amazon S3.

```
grouped_df = parsed_df \
                .groupBy(window("event_time", "10 minutes"), "city") \
                .agg(sum("clicks").alias("total_clicks"))

summary_df = grouped_df \
                .withColumn("window_start_time", col("window.start")) \
                .withColumn("window_end_time", col("window.end")) \
                .withColumn("year", year("window_start_time")) \
                .withColumn("month", month("window_start_time")) \
                .withColumn("day", dayofmonth("window_start_time")) \
                .withColumn("hour", hour("window_start_time")) \
                .withColumn("minute", minute("window_start_time")) \
                .drop("window")

write_result = summary_df \
                .writeStream \
                .format("parquet") \
                .trigger(processingTime="10 seconds") \
                .option("checkpointLocation", "s3a://bucket-stock-stream/stock-
stream-catalog-job/checkpoint/") \
                .option("path", "s3a://bucket-stock-stream/stock-stream-catalog-
job/summary_output/") \
                .partitionBy("year", "month", "day") \
                .start()
```

Finestra scorrevole

Come le finestre a cascata, le finestre scorrevoli hanno dimensioni fisse, ma a differenza di esse possono sovrapporsi o scorrere, a condizione che la durata dello scorrimento sia inferiore alla durata della finestra stessa. In virtù della natura dello scorrimento, uno stesso input può essere associato a più finestre.



Per comprendere meglio, prendiamo in considerazione l'esempio di una banca che desidera rilevare potenziali frodi relative alle carte di credito. Un'applicazione di streaming potrebbe monitorare un flusso continuo delle transazioni con carta di credito. Queste transazioni potrebbero essere aggregate in finestre della durata di 10 minuti e ogni 5 minuti la finestra scorrerebbe in avanti, eliminando i 5 minuti di dati più vecchi e aggiungendo gli ultimi 5 minuti di dati più recenti. All'interno di ciascuna finestra, le transazioni potrebbero essere raggruppate per paese, verificando la presenza di schemi sospetti, ad esempio una transazione negli Stati Uniti seguita immediatamente da un'altra in Australia. Per semplicità, tali transazioni vengono classificate come frodi quando l'importo totale delle transazioni è superiore a 100 USD. Se viene rilevato uno schema di questo tipo, viene segnalata una frode potenziale e la carta potrebbe essere bloccata.

Il sistema di elaborazione delle carte di credito sta inviando a Kinesis una serie di transazioni con i dati relativi agli ID delle carte di credito e al paese. Un processo di AWS Glue esegue l'analisi e produce il seguente output aggregato.

ora_inizio_finestra	ora_fine_finestra	ultime_quattro_cifre_carta	country	importo_totale
2023-07-10 17:00:00	2023-07-10 17:10:00	6544	US	85
2023-07-10 17:00:00	2023-07-10 17:10:00	6544	Australia	10
2023-07-10 17:05:45	2023-07-10 17:15:45	6544	US	50
2023-07-10 17:10:45	2023-07-10 17:20:45	6544	US	50

ora_inizio_finestra	ora_fine_finestra	ultime_quattro_cifre_carta	country	importo_totale
2023-07-10 17:10:45	2023-07-10 17:20:45	6544	Australia	150

In base all'aggregazione di cui sopra, si può osservare come la finestra di 10 minuti scorra ogni 5 minuti, sommata per importo della transazione. L'anomalia viene rilevata nella finestra 17:10-17:20 dove compare un valore anomalo, ossia una transazione del valore di 150 USD in Australia. AWS Glue è in grado di rilevare questa anomalia e inviare un evento di allarme con la chiave incriminata a un argomento SNS utilizzando boto3. Approfondisci il Lambda, leggi i dati da questo argomento e agisci.

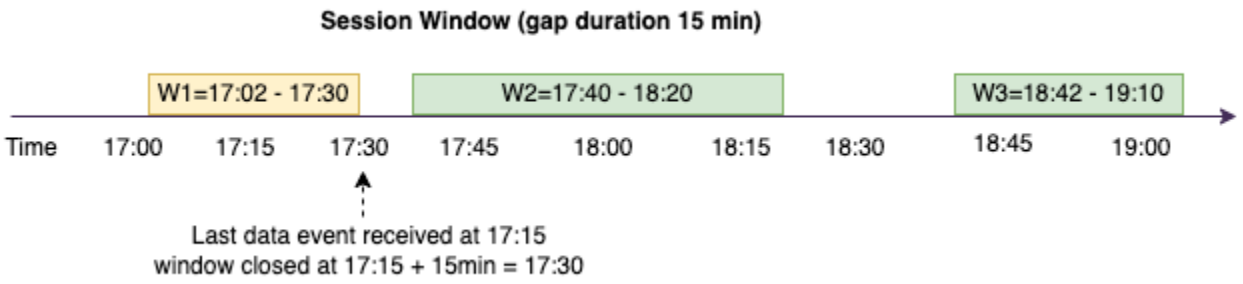
Elaborazione dei dati in una finestra scorrevole

Per implementare la finestra scorrevole vengono utilizzate la clausola `group-by` e la funzione finestra, come mostrato di seguito.

```
grouped_df = parsed_df \
            .groupBy(window(col("event_time"), "10 minute", "5 min"), "country",
 "card_last_four") \
            .agg(sum("tx_amount").alias("total_amount"))
```

Finestra di sessione

A differenza delle due finestre precedenti, che hanno una dimensione fissa, la finestra di sessione può avere una lunghezza fissa o dinamica a seconda degli input. Una finestra di sessione inizia con un evento di dati di input e continua a espandersi finché riceve input entro un intervallo di tempo o un periodo di inattività.



Facciamo un esempio. L'hotel ABC vuole scoprire qual è il periodo più trafficato della settimana e proporre agli ospiti offerte più allettanti. Non appena un ospite effettua il check-in, viene avviata una finestra di sessione e Spark mantiene uno stato con relativa aggregazione. event-time-window Ogni volta che un ospite effettua il check-in, un evento viene generato e inviato ad Amazon Kinesis Data Streams. L'hotel decide che se non ci sono check-in per un periodo di 15 minuti, può essere chiuso. event-time-window Il prossimo event-time-window ricomincerà quando ci sarà un nuovo check-in. L'output è simile al seguente.

ora_inizio_finestra	ora_fine_finestra	città	checkin_totali
2023-07-10 17:02:00	2023-07-10 17:30:00	Dallas	50
2023-07-10 17:02:00	2023-07-10 17:30:00	Chicago	25
2023-07-10 17:40:00	2023-07-10 18:20:00	Dallas	75
2023-07-10 18:50:45	2023-07-10 19:15:45	Dallas	20

Il primo check-in è avvenuto all'ora_evento=17:02. L'aggregazione avrà inizio alle 17:02. event-time-window L'aggregazione continuerà fintantoché verranno ricevuti eventi nell'arco di 15 minuti. Nell'esempio precedente, l'ultimo evento è stato ricevuto alle 17:15, poi per i successivi 15 minuti non si sono verificati eventi. Di conseguenza, Spark lo ha chiuso alle 17:15 +15min = event-time-window 17:30 e lo ha impostato come 17:02 - 17:30. È iniziato un nuovo evento alle 17:47 quando ha ricevuto un nuovo event-time-window evento relativo ai dati del check-in.

Elaborazione dei dati in una finestra di sessione

Per implementare la finestra scorrevole vengono utilizzate la clausola `group-by` e la funzione finestra.

```
grouped_df = parsed_df \
            .groupBy(session_window(col("event_time"), "10 minute"), "city") \
            .agg(count("check_in").alias("total_checkins"))
```

Modalità di output

La modalità di output è la modalità in cui i risultati della tabella illimitata vengono scritti nel sink esterno. Sono disponibili tre modalità. Nell'esempio seguente si contano le occorrenze di una parola mentre le righe di dati vengono trasmesse ed elaborate in ogni microbatch.

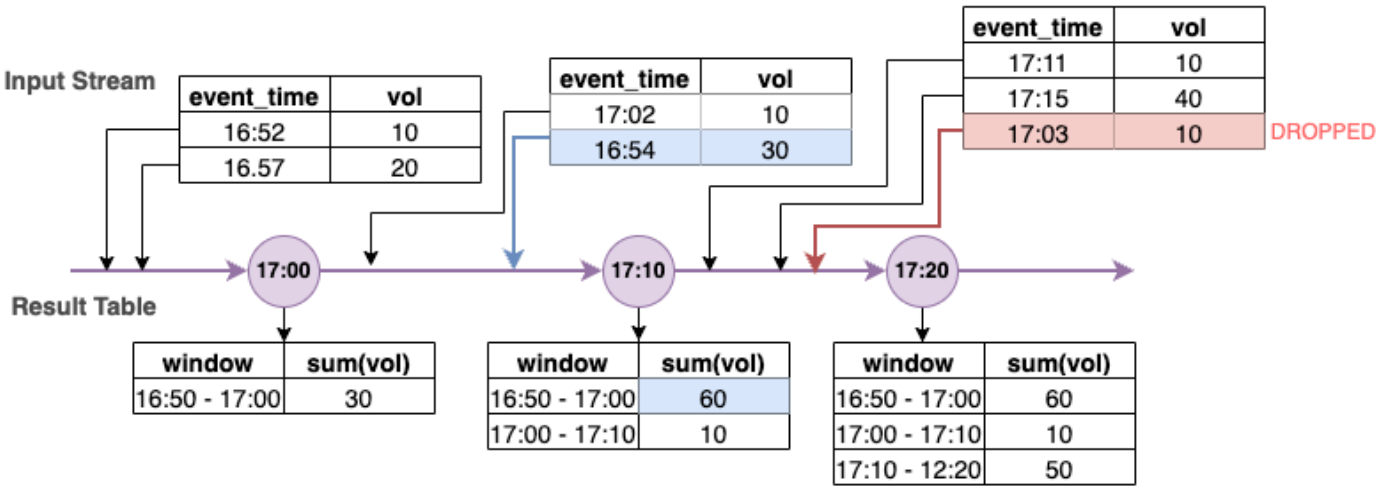
- Modalità completa: l'intera tabella dei risultati verrà scritta nel sink dopo ogni elaborazione in microbatch, anche se il conteggio delle parole non è stato aggiornato nella versione corrente event-time-window.
- Modalità di aggiunta: questa è la modalità predefinita, in cui solo le nuove parole e/o righe aggiunte alla tabella dei risultati dall'ultima attivazione vengono scritte nel sink. Questa modalità è utile per lo streaming stateless per query come map, flatMap, filter, ecc.
- Modalità di aggiornamento: nel sink vengono scritte solo le parole e/o le righe che sono state aggiornate o aggiunte nella tabella dei risultati dall'ultima attivazione.

> **Note**
>
> La modalità di output "aggiornamento" non è supportata per le finestre di sessione.

Gestione di dati in ritardo e filigrane

Quando si lavora con dati in tempo reale, potrebbero verificarsi ritardi nell'arrivo dei dati a causa della latenza della rete e di guasti a monte e abbiamo bisogno di un meccanismo per eseguire nuovamente l'aggregazione dei dati persi. event-time-window Tuttavia, a tale scopo, è necessario mantenere lo stato. Allo stesso tempo, per limitare le dimensioni dello stato, è necessario rimuovere i dati più vecchi. La versione 2.1 di Spark ha aggiunto il supporto per una funzionalità chiamata "watermarking", ossia "applicazione della filigrana", che mantiene lo stato e consente all'utente di specificare la soglia per i dati in ritardo.

Facendo riferimento all'esempio sul simbolo azionario riportato sopra, poniamo che i dati in ritardo non possano superare la soglia dei 10 minuti. Per semplificare, supponiamo di utilizzare le finestre a cascata, il simbolo AMZ e l'opzione ACQUISTA.



Nel diagramma precedente, calcoliamo il volume totale su una finestra a cascata di 10 minuti. L'attivazione è impostata alle ore 17:00, 17:10 e 17:20. Sopra la freccia della linea temporale si trova il flusso di dati di input, mentre sotto si trova la tabella dei risultati illimitata.

Nella prima finestra a cascata di 10 minuti i dati sono stati aggregati in base a ora_evento e il volume_totale calcolato è stato 30. Nel secondo event-time-window, spark ha ricevuto il primo evento di dati con event_time= 17:02. Poiché questo è il valore massimo di ora_evento visto finora da Spark, la soglia della filigrana viene riportata indietro di 10 minuti (ossia, ora_evento_filigrana=16:52). Qualsiasi evento di dati con un valore di ora_evento successivo alle 16:52 verrà preso in considerazione per l'aggregazione entro i limiti temporali, mentre gli eventi di dati precedenti verranno eliminati. Ciò consente a Spark di mantenere uno stato intermedio per altri 10 minuti per accogliere i dati in ritardo. Intorno alle 17:08, Spark ha ricevuto un evento con un valore di ora_evento=16:54 che rientrava nella soglia. Quindi spark ha ricalcolato «16:50 - 17:00 « event-time-window e il volume totale è stato aggiornato da 30 a 60.

Tuttavia, all'ora di attivazione 17:20, quando Spark ha ricevuto un evento con ora_evento=17:15, ha impostato ora_evento_filigrana=17:05. Pertanto, l'evento di dati in ritardo con il valore di ora_evento=17:03 è stato considerato successivo alla soglia di tolleranza e quindi ignorato.

```
Watermark Boundary = Max(Event Time) - Watermark Threshold
```

Utilizzo delle filigrane in AWS Glue

Spark non emette né scrive i dati nel sink esterno finché non viene superato il limite della filigrana. Per implementare una filigrana in AWS Glue, consulta l'esempio seguente.

```
grouped_df = parsed_df \
                .withWatermark("event_time", "10 minutes") \
                .groupBy(window("event_time", "5 minutes"), "ticker") \
                .agg(sum("volume").alias("total_volume"))
```

Monitoraggio dei processi di streaming di AWS Glue

Il monitoraggio dei processi di streaming è una parte fondamentale della creazione delle pipeline di ETL. Oltre a utilizzare l'interfaccia utente Spark, puoi anche utilizzare Amazon CloudWatch per monitorare le metriche. Di seguito è riportato un elenco di parametri di streaming emessi dal framework AWS Glue. Per un elenco completo di tutte le AWS Glue metriche, consulta Monitoraggio AWS Glue tramite i CloudWatch parametri Amazon.

AWS Glue utilizza un framework di streaming strutturato per elaborare gli eventi di input. Puoi utilizzare l'API Spark direttamente nel tuo codice o sfruttare il valore `ForEachBatch` fornito da `GlueContext`, che pubblica questi parametri. Per comprendere questi parametri, dobbiamo prima capire `windowSize`.

windowSize: `windowSize` è l'intervallo di microbatch fornito. Se specifichi una dimensione della finestra di 60 secondi, il processo di streaming di AWS Glue aspetterà 60 secondi (o più, se il batch precedente non è stato completato entro tale lasso di tempo) prima di leggere i dati in un batch dall'origine di streaming e applicare le trasformazioni fornite in `ForEachBatch`. Questo valore viene chiamato anche intervallo di attivazione.

Esaminiamo i parametri in modo più dettagliato per comprendere le caratteristiche di integrità e prestazioni.

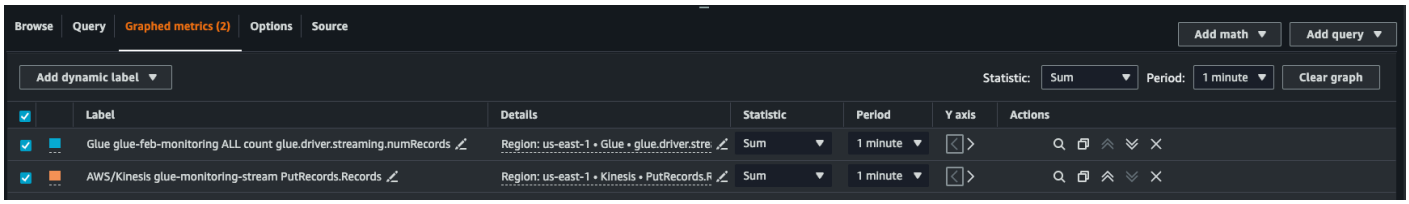
> **Note**
>
> Questi parametri vengono emessi ogni 30 secondi. Se il valore `windowSize` fornito è inferiore a 30 secondi, i parametri riportati sono un'aggregazione. Ad esempio, supponiamo che il valore `windowSize` fornito sia di 10 secondi e che si stiano elaborando costantemente 20 record per microbatch. In questo scenario, il valore del parametro emesso per numRecords sarebbe 60.
> Un parametro non viene emesso se per esso non sono disponibili dati. Inoltre, nel caso del parametro del ritardo del consumatore, è necessario abilitare la funzione per ottenere i parametri associati.

Visualizzazione dei parametri

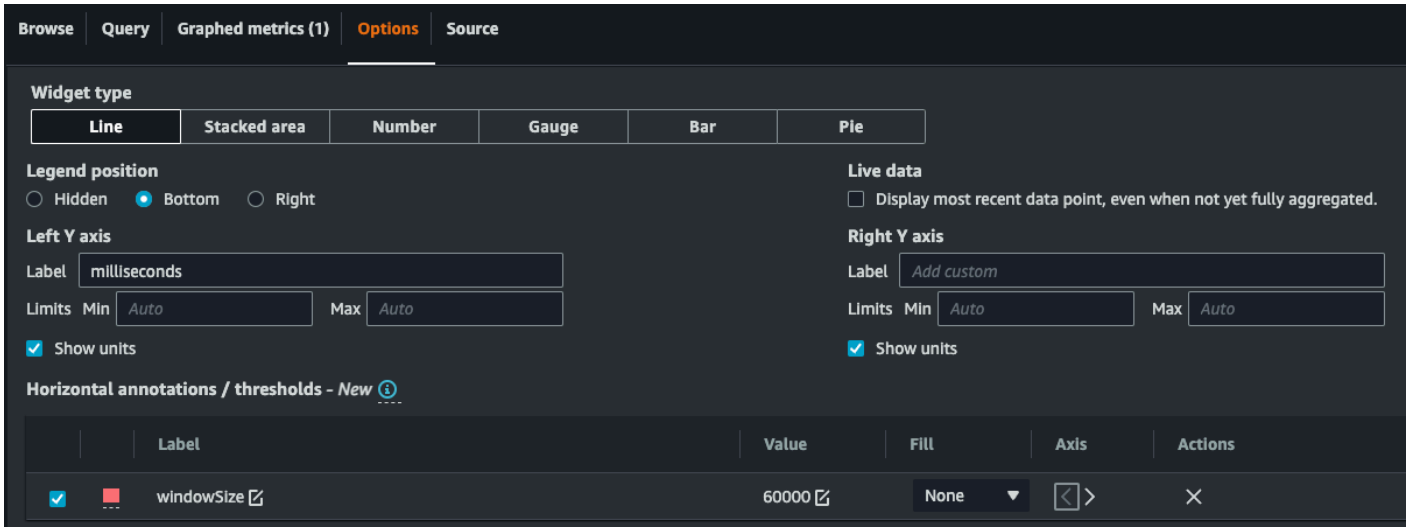
Per tracciare i parametri visivamente:

1. Vai a Metrics nella CloudWatch console Amazon, quindi seleziona la scheda Sfoglia. In "Spazi dei nomi personalizzati", scegli Glue.

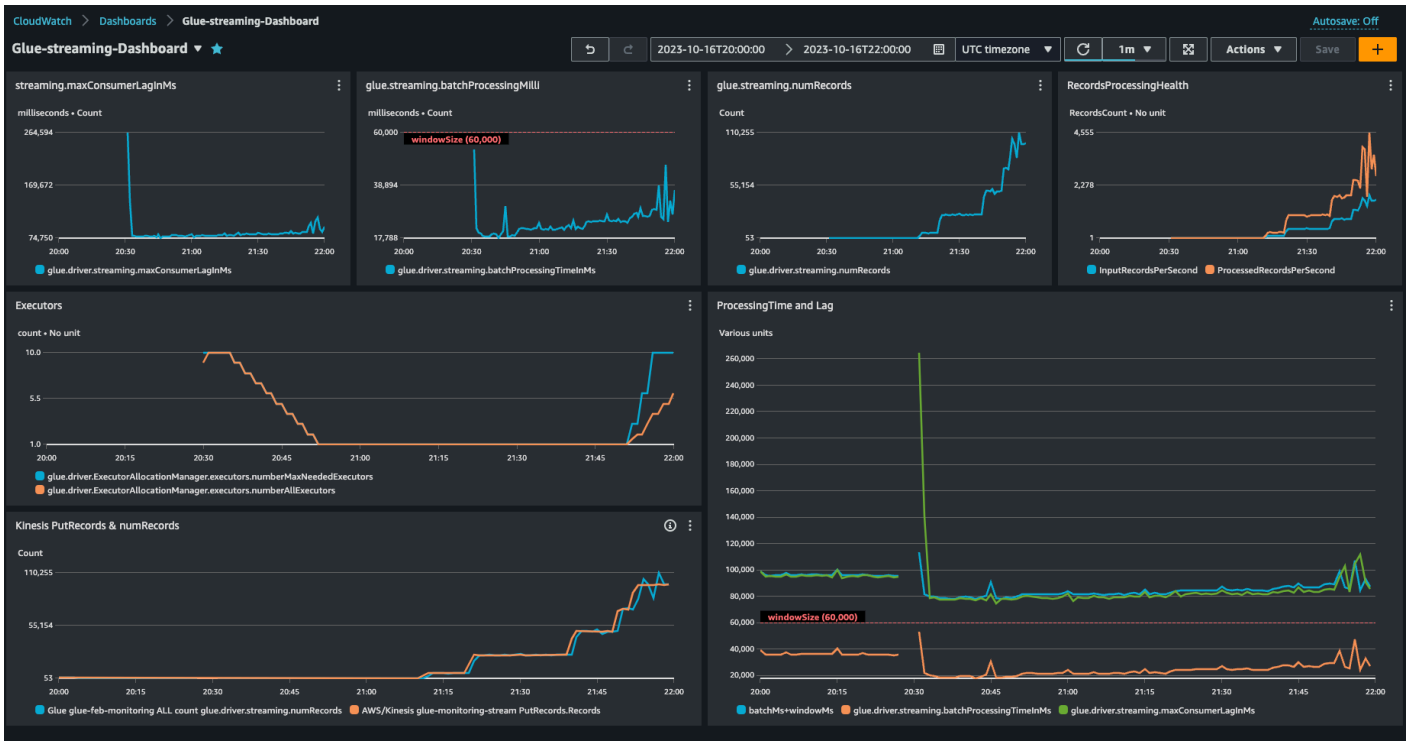
2. Scegli Parametri processo per visualizzare i parametri di tutti i processi.
3. Filtra le metriche in base a JobName = glue-feb-monitoring e poi a =ALL. JobRunId Puoi fare clic sul segno "+", come mostrato nella figura seguente, per aggiungerlo al filtro di ricerca.
4. Seleziona la casella di controllo relativa ai parametri che ti interessano. Nella figura seguente abbiamo selezionato `numberAllExecutors` e `numberMaxNeededExecutors`.



5. Dopo aver selezionato questi parametri, puoi andare alla scheda Parametri definiti e applicare le tue statistiche.
6. Poiché i parametri vengono emesse ogni minuto, puoi applicare la "media" su un minuto per `batchProcessingTimeInMs` e `maxConsumerLagInMs`. Per `numRecords`, puoi applicare la "somma" per ogni minuto.
7. È possibile aggiungere un'annotazione `windowSize` orizzontale al grafico tramite la scheda Opzioni.



8. Dopo aver selezionato i parametri, crea un pannello di controllo e aggiungilo. Di seguito è mostrato un pannello di controllo di esempio.

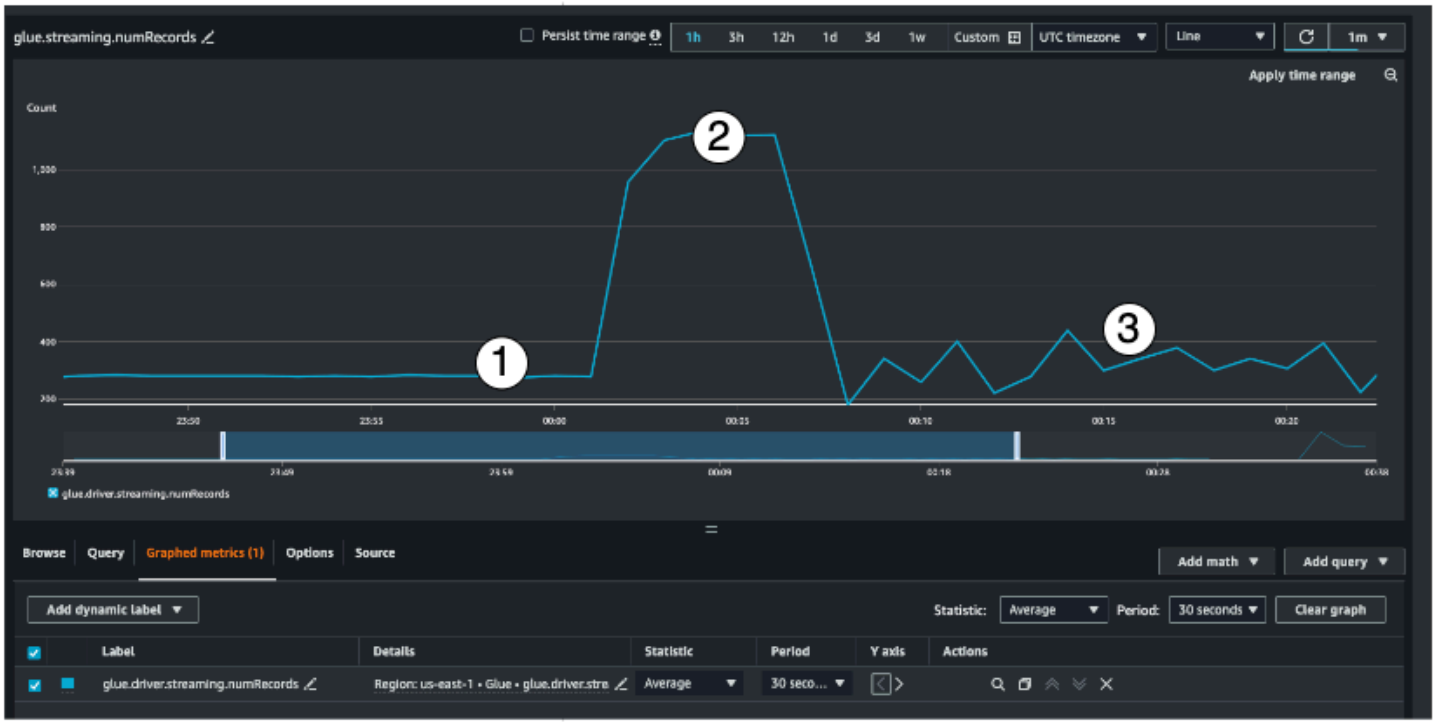


Approfondimento sui parametri

Questa sezione descrive ciascuno dei parametri e il modo in cui sono correlati tra loro.

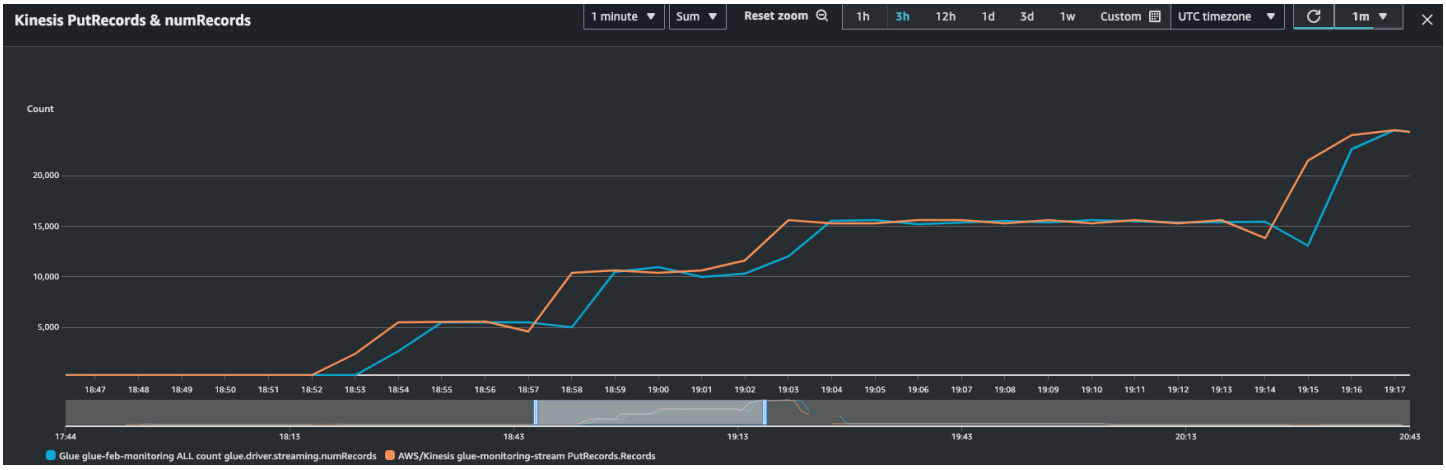
Numero di record (parametro: streaming.numRecords)

Questo parametro indica quanti record sono in fase di elaborazione.



Questo parametro di streaming consente di visualizzare il numero di record in fase di elaborazione in una finestra. Oltre a specificare il numero di record in fase di elaborazione, agevola la comprensione del comportamento del traffico di input.

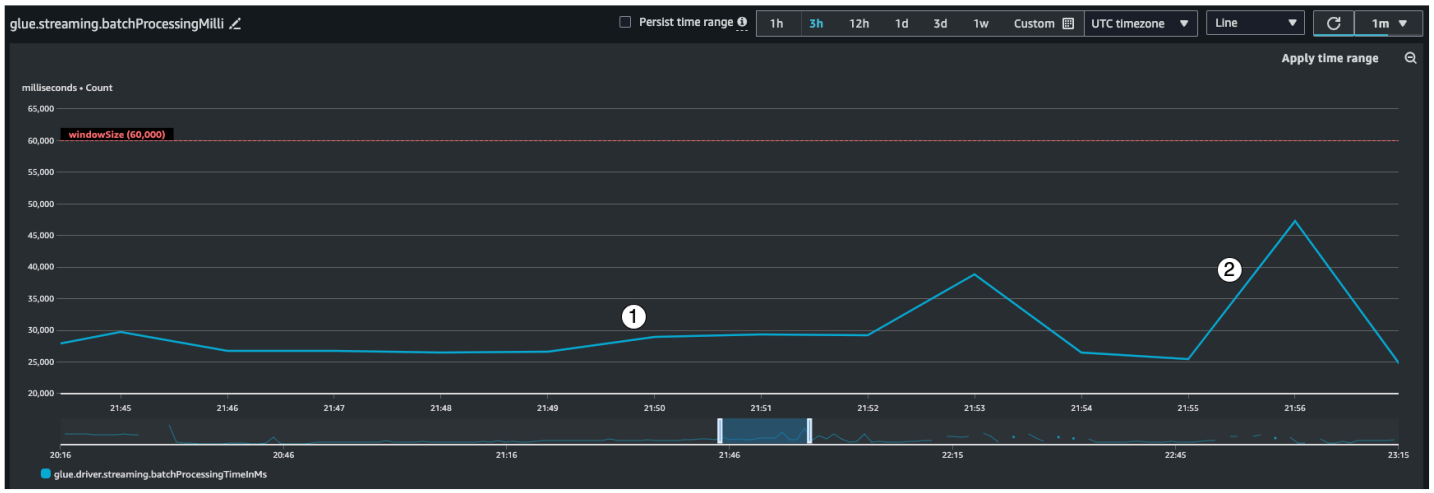
- L'indicatore n. 1 mostra un esempio di traffico stabile senza picchi. In genere si tratta di applicazioni come i sensori IoT che raccolgono dati a intervalli regolari e li inviano all'origine di streaming.
- L'indicatore n. 2 mostra un esempio di improvviso aumento del traffico su un carico altrimenti stabile. In un'applicazione clickstream, ciò può accadere in concomitanza con un evento di marketing come il Black Friday, quando si verifica un aumento esponenziale del numero di clic.
- L'indicatore n. 3 mostra un esempio di traffico imprevedibile. Il traffico imprevedibile non significa che ci sia un problema. ma è una caratteristica intrinseca dei dati di input. Tornando all'esempio del sensore IoT, immaginiamo centinaia di sensori che inviano eventi di cambiamenti meteorologici all'origine di streaming. Poiché i cambiamenti meteorologici non sono prevedibili, non lo sono nemmeno i dati. Comprendere l'andamento del traffico è fondamentale per dimensionare gli esecutori. Se l'input presenta molti picchi, potresti prendere in considerazione l'utilizzo del dimensionamento automatico, di cui parleremo più avanti.



Puoi combinare questa metrica con la metrica PutRecords Kinesis per assicurarti che il numero di eventi da inserire e il numero di record letti siano quasi gli stessi. Questo è particolarmente utile quando si cerca di comprendere il ritardo. All'aumentare della frequenza di importazione, aumenta anche il numero di `numRecords` letti da AWS Glue.

Tempo di elaborazione in batch (metrica: streaming). batchProcessingTimeInMs)

Il parametro del tempo di elaborazione del batch consente di determinare se il provisioning del cluster è insufficiente o eccessivo.

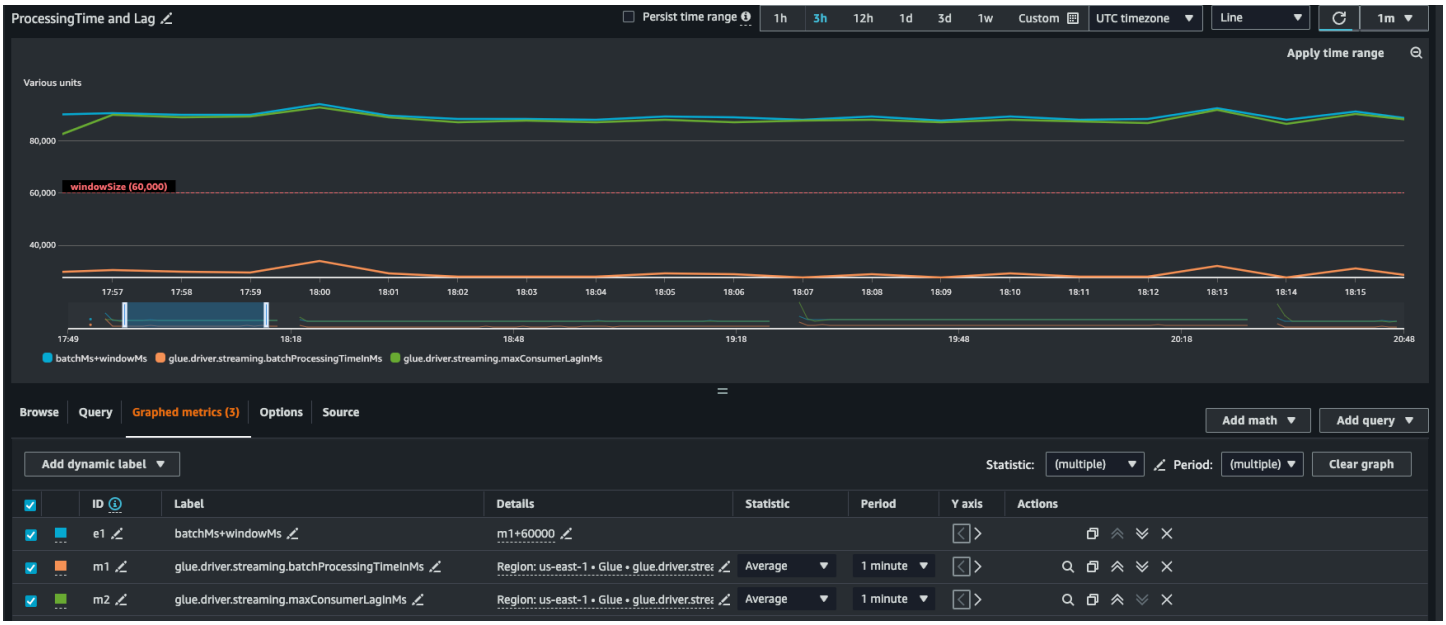


Questo parametro indica il numero di millisecondi necessari per elaborare ogni microbatch di record. L'obiettivo principale in questo caso è monitorare questo periodo per assicurarsi che sia inferiore all'intervallo `windowSize`. È accettabile che `batchProcessingTimeInMs` salga temporaneamente, purché torni alla normalità nell'intervallo di finestra successivo. L'indicatore n. 1 mostra un tempo più o meno stabile richiesto per elaborare il processo. Tuttavia, se il numero di record di input aumenta, il tempo necessario per elaborare il processo aumenta di conseguenza,

come segnalato dall'indicatore n. 2. Se `numRecords` non aumenta ma il tempo di elaborazione sì, è necessario esaminare più a fondo l'elaborazione del processo sugli esecutori. È buona norma impostare una soglia e un allarme per assicurarsi che `batchProcessingTimeInMs` non superi il 120% per più di 10 minuti. Per ulteriori informazioni sull'impostazione degli allarmi, consulta Using Amazon CloudWatch alarms.

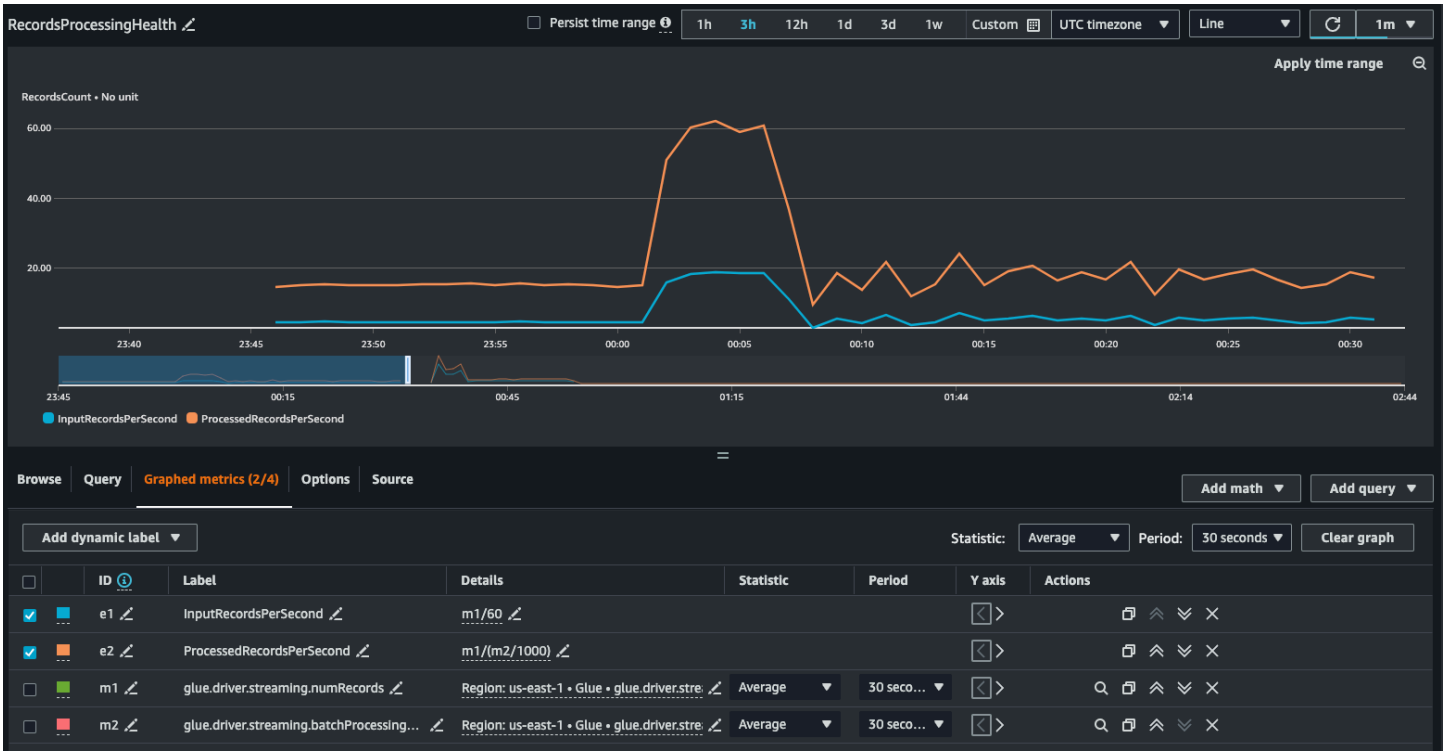
Ritardo dei consumatori (metrica: streaming). maxConsumerLagInMs)

Il parametro del ritardo dei consumatori aiuta a comprendere se sussiste un ritardo nell'elaborazione degli eventi. Se il ritardo è troppo elevato, potresti non rispettare lo SLA di elaborazione sottoscritto dalla tua azienda, anche se disponi di un windowSize corretto. È necessario abilitare esplicitamente questi parametri utilizzando l'opzione di connessione `emitConsumerLagMetrics`. Per ulteriori informazioni, vedere KinesisStreamingSourceOptions.



Parametri derivati

Per ottenere informazioni più approfondite, puoi creare metriche derivate per saperne di più sui tuoi lavori di streaming in Amazon CloudWatch.



Puoi creare un grafico con parametri derivati per decidere se è necessario utilizzare più DPU. Sebbene il dimensionamento automatico ti aiuti a farlo automaticamente, puoi utilizzare parametri derivati per stabilire se il dimensionamento automatico funziona in modo efficace.

- InputRecordsPerSecondindica la frequenza con cui vengono ricevuti i record di input. È derivato come segue: numero di record di input (glue.driver.streaming.numRecords)/. WindowSize
- ProcessingRecordsPerSecondindica la velocità con cui vengono elaborati i record. È derivato come segue: numero di record di input (glue.driver.streaming.numRecords)/. batchProcessingTime InMs

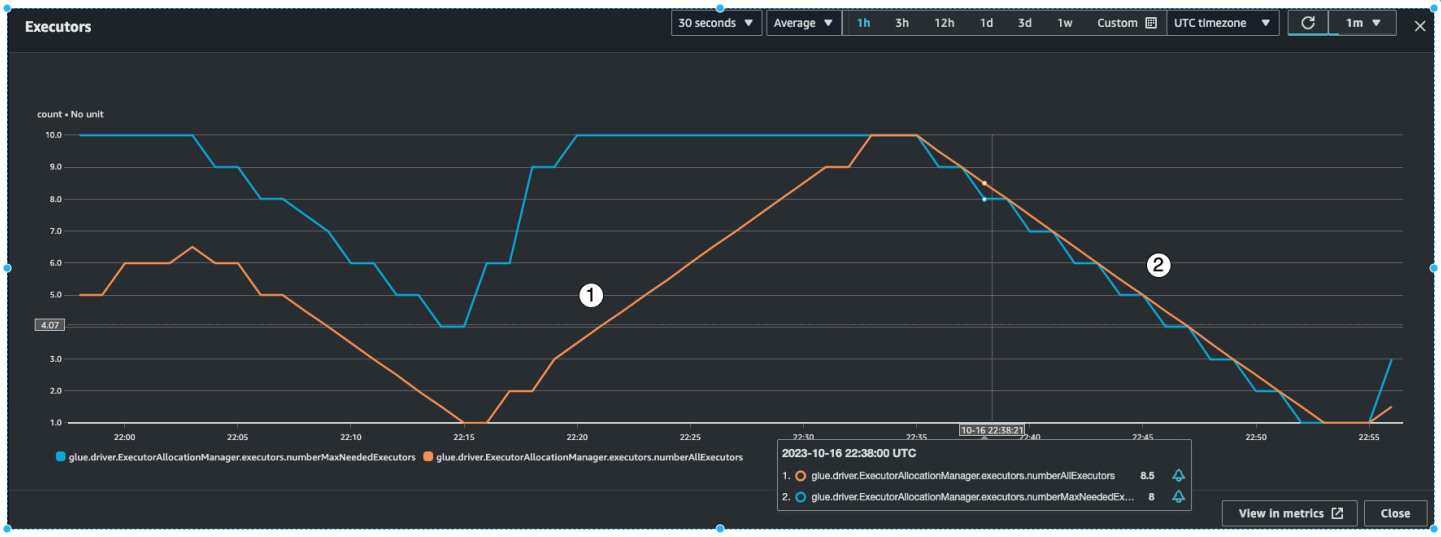
Se la velocità di input è superiore a quella di elaborazione, potrebbe essere necessario incrementare la capacità per elaborare il processo oppure aumentare il parallelismo.

Parametri di dimensionamento automatico

Se il traffico in entrata ha molti picchi, dovresti valutare l'abilitazione del dimensionamento automatico e specificare il numero massimo di worker. In tal caso ottieni due parametri aggiuntivi, `numberAllExecutors` e `numberMaxNeededExecutors`.

- numberAllExecutorsè il numero di esecutori di lavori che eseguono attivamente
- numberMaxNeededExecutor è il numero massimo di job executor (in esecuzione attiva e in sospeso) necessari per soddisfare il carico corrente.

Questi due parametri ti aiuteranno a capire se il dimensionamento automatico funziona correttamente.



AWS Glue monitorerà il parametro `batchProcessingTimeInMs` su alcuni microbatch e compirà una delle due operazioni seguenti. Aumenterà gli esecutori, se `batchProcessingTimeInMs` è più vicino a `windowSize`, oppure ridurrà gli esecutori, se `batchProcessingTimeInMs` è relativamente più basso di `windowSize`. Inoltre, utilizzerà un algoritmo per dimensionare gradualmente gli esecutori.

- L'indicatore n. 1 mostra come gli esecutori attivi sono aumentati fino a raggiungere il numero massimo di esecutori necessari per elaborare il carico.
- L'indicatore n. 2 mostra che gli esecutori attivi sono diminuiti rispetto a quando `batchProcessingTimeInMs` era basso.

È possibile utilizzare questi parametri per monitorare l'attuale parallelismo a livello di esecutore e regolare di conseguenza il numero massimo di worker nella configurazione di dimensionamento automatico.

Come ottenere le prestazioni migliori

Spark cercherà di creare per ogni shard un'attività da cui leggere nel flusso Amazon Kinesis. I dati in ogni shard diventano una partizione. Quindi distribuirà queste attività tra gli esecutori/worker, a seconda del numero di core di ciascun worker (il numero di core per worker dipende dal tipo di worker selezionato, come `G.025X`, `G.1X` e così via). Tuttavia, il modo in cui le attività vengono distribuite non è deterministico. Tutte le attività vengono eseguite in parallelo sui rispettivi core. Se

sono presenti più shard rispetto al numero di core esecutori disponibili, le attività vengono messe in coda.

È possibile utilizzare una combinazione dei parametri precedenti e del numero di shard per fornire agli esecutori un carico stabile con un certo margine per eventuali picchi. Si consiglia di eseguire alcune iterazioni del processo per determinare il numero approssimativo di worker. Per un carico di lavoro instabile/con picchi, puoi ottenere il medesimo risultato impostando il dimensionamento automatico e il numero massimo di worker.

Imposta il valore di `windowSize` in base ai requisiti SLA della tua azienda. Ad esempio, se la tua azienda richiede che i dati elaborati non possano essere più vecchi di 120 secondi, imposta un valore di `windowSize` di almeno 60 secondi, in modo che il ritardo medio dei consumatori sia inferiore a 120 secondi (consulta la sezione precedente sul ritardo dei consumatori). A questo punto, in base a `numRecords` e al numero di shard, pianifica la capacità in DPU assicurandoti che il valore di `batchProcessingTimeInMs` sia inferiore al 70% del valore di `windowSize` per la maggior parte del tempo.

> **Note**
>
> Gli shard caldi possono causare una distorsione dei dati, il che significa che alcuni shard/partizioni risultano molto più grandi di altri. Ciò può far sì che alcune attività eseguite in parallelo richiedano più tempo, cosicché alcune attività restano indietro. Di conseguenza, il batch successivo non può iniziare fino al completamento di tutte le attività del precedente, il che influirà sul valore di `batchProcessingTimeInMillis` e sul ritardo massimo.

AWS Glue Qualità dei dati

AWS Glue La qualità dei dati consente di misurare e monitorare la qualità dei dati in modo da poter prendere buone decisioni aziendali. Basato su un DeeQu framework open source, AWS Glue Data Quality offre un'esperienza gestita e senza server. AWS Glue Data Quality funziona con Data Quality Definition Language (DQDL), un linguaggio specifico del dominio utilizzato per definire le regole di qualità dei dati. Per ulteriori informazioni su DQDL e sui tipi di regole supportati, consulta la pagina Riferimento a Data Quality Definition Language (DQDL).

Per informazioni aggiuntive sul prodotto e sui prezzi, consulta la pagina del servizio Qualità dei dati di AWS Glue.

Vantaggi e funzionalità principali

I vantaggi e le caratteristiche principali di AWS Glue Data Quality includono:

- Serverless: non è necessaria alcuna installazione, applicazione di patch o manutenzione.
- Inizia subito: AWS Glue Data Quality analizza rapidamente i tuoi dati e crea regole di qualità dei dati per te. È possibile iniziare con due clic: "Crea regole sulla qualità dei dati → Regole suggerite".
- Rileva problemi di qualità dei dati: utilizza l'apprendimento automatico (ML) per rilevare anomalie e problemi di qualità hard-to-detect dei dati.
- Improvvisa le tue regole: con più di 25 regole out-of-the-box DQ da cui partire, puoi creare regole adatte alle tue esigenze specifiche.
- Valuta la qualità e prendi decisioni aziendali con fiducia: una volta valutate le regole, ottieni un punteggio di qualità dei dati che fornisce una panoramica dello stato dei tuoi dati. Utilizza il punteggio di qualità dei dati per prendere decisioni aziendali con fiducia.
- Concentrati sui dati errati: AWS Glue Data Quality ti aiuta a identificare i record esatti che hanno causato il calo dei punteggi di qualità. Identificali, mettili in quarantena e correggili facilmente.
- Pagamento in base al consumo: non sono necessarie licenze annuali per utilizzare AWS Glue Data Quality.
- Nessun vincolo: AWS Glue Data Quality è basato sull'open source DeeQu e ti consente di mantenere le regole che stai creando in un linguaggio aperto.
- Controlli della qualità AWS Glue dei dati — Qualità dei dati È possibile applicare i controlli di qualità dei dati sulle pipeline AWS Glue ETL Data Catalog e gestire la qualità dei dati inattivi e in transito.

- Rilevamento della qualità dei dati basato su ML: utilizza l'apprendimento automatico (ML) per rilevare anomalie e problemi di qualità dei dati. hard-to-detect

Come funziona

Esistono due punti di accesso per la qualità AWS Glue dei dati: i job AWS Glue Data Catalog ed AWS Glue ETL. Questa sezione fornisce una panoramica dei casi d'uso e delle AWS Glue funzionalità supportate da ciascun punto di ingresso.

Qualità dei dati per AWS Glue Data Catalog

AWS Glue Data Quality valuta gli oggetti archiviati in e offre ai AWS Glue Data Catalog non programmatori un modo semplice per impostare regole di qualità dei dati. Queste figure includono amministratori di dati e analisti aziendali.

È possibile scegliere questa opzione per i seguenti casi d'uso:

- Desideri eseguire attività relative alla qualità dei dati su set di dati che hai già catalogato in AWS Glue Data Catalog.
- Ti occupi di governance dei dati e devi identificare o valutare i problemi di qualità dei dati nel tuo data lake su base continuativa.

È possibile gestire la qualità dei dati per Catalogo dati utilizzando le seguenti interfacce:

- La console di gestione AWS Glue
- AWS Glue API

Per iniziare a usare AWS Glue Data Quality for the AWS Glue Data Catalog seeNozioni di base su AWS Glue Data Quality per Data Catalog.

Qualità dei dati per AWS Glue lavori ETL

AWS Glue Data Quality for AWS Glue ETL Jobs consente di eseguire attività proattive sulla qualità dei dati. Le attività proattive ti aiutano a identificare e filtrare i dati errati prima di caricare un set di dati nel tuo data lake.

Video: Presentazione della qualità AWS Glue dei dati per le pipeline ETL

È possibile scegliere la qualità dei dati per i processi ETL per i seguenti casi d'uso:

- Desideri integrare attività relative alla qualità dei dati nei tuoi processi ETL
- Desideri scrivere codice che definisca le attività relative alla qualità dei dati negli script ETL
- Vuoi gestire la qualità dei dati che fluiscono nelle tue pipeline di dati visive

È possibile gestire la qualità dei dati per i processi ETL utilizzando le seguenti interfacce:

- AWS Glue Studio, AWS Glue Studio notebook e sessioni interattive AWS Glue
- AWS Glue librerie per lo scripting ETL
- AWS Glue API

Per iniziare a utilizzare la qualità dei dati per i processi ETL, consulta la pagina Tutorial: Getting started with Data Quality nella Guida per l'utente di AWS Glue Studio .

Confronto della qualità dei dati per Catalogo dati con la qualità dei dati per i processi ETL

Questa tabella fornisce una panoramica delle funzionalità supportate da ogni punto di ingresso di AWS Glue Data Quality.

Funzionalità	Qualità dei dati per Catalogo dati	Qualità dei dati per i processi ETL
Origini dati	Amazon S3, Amazon Redshift, origini JDBC compatibili con Catalogo dati e formati di data lake transazionali come Apache Iceberg, Apache Hudi e Delta Lake. Nota che se le tabelle sono AWS Lake Formation gestite, le tabelle Iceberg, Delta e HUDI non sono supportate. Amazon Athena le viste catalogate in	Tutte le fonti di dati supportat e da AWS Glue, inclusi connettori personalizzati e connettori di terze parti.

Funzionalità	Qualità dei dati per Catalogo dati	Qualità dei dati per i processi ETL
	non AWS Glue Data Catalog sono supportate.	
Suggerimenti di regole di Qualità dei dati	Supportato	Non supportato
Creazione ed esecuzione di regole DQDL	Supportato	Supportato
Dimensionamento automatico	Non supportato	Supportata
AWS Glue Supporto Flex	Non supportato	Supportata
Pianificazione	Supportato durante la valutazione delle regole di Qualità dei dati e tramite Step Functions.	Supportato durante l'utilizzo di Step Functions e flussi di lavoro.
Identificazione dei record che non hanno superato i controlli di qualità dei dati	Non supportato	Supportata
Integrazione con Amazon EventBridge	Supportato	Supportato
Integrazione con Cloudwatch AWS	Supportato	Supportato
Scrittura dei risultati di qualità dei dati in Amazon S3	Supportato	Supportato
Qualità incrementale dei dati	Supportato tramite predicati pushdown	Supportato tramite segnalibri AWS Glue
AWS CloudFormation supporto	Supportato	Supportato

Funzionalità	Qualità dei dati per Catalogo dati	Qualità dei dati per i processi ETL
Rilevamento delle anomalie basato su ML	Non supportato	Anteprima
Regole dinamiche	Non supportato	Supportata

Considerazioni

Prendi in considerazione i seguenti elementi prima di utilizzare AWS Glue Data Quality:

- Le regole di qualità dei dati non possono valutare origini dati annidate o di tipo elenco. Per informazioni, consulta Appiattimento di strutture annidate.

Terminologia

L'elenco seguente definisce i termini correlati alla qualità AWS Glue dei dati.

Data Quality Definition Language (DQDL)

Linguaggio specifico del dominio che è possibile utilizzare per scrivere regole di qualità AWS Glue dei dati.

Per ulteriori informazioni su DQDL, consulta la guida di Riferimento a Data Quality Definition Language (DQDL).

qualità dei dati

Descrive in che modo un set di dati soddisfa il suo scopo specifico. AWS Glue Data Quality valuta le regole rispetto a un set di dati per misurare la qualità dei dati. Ogni regola verifica caratteristiche particolari come la freschezza o l'integrità dei dati. Per quantificare la qualità dei dati, è possibile utilizzare un punteggio di qualità dei dati.

punteggio di qualità dei dati

La percentuale di regole sulla qualità dei dati che vengono rispettate (risultano vere) quando si valuta un set di regole con Data Quality. AWS Glue

regola

Un'espressione DQDL che controlla i dati per una caratteristica specifica e restituisce un valore booleano. Per ulteriori informazioni, consulta Struttura delle regole.

analyzer

Un'espressione DQDL che raccoglie statistiche sui dati. Un analizzatore raccoglie statistiche sui dati che possono essere utilizzate dagli algoritmi ML per rilevare anomalie e problemi di qualità dei dati nel tempo. hard-to-detect

set di regole

Una AWS Glue risorsa che comprende una serie di regole sulla qualità dei dati. Un set di regole deve essere associato a una tabella in AWS Glue Data Catalog. Quando salvi un set di regole, AWS Glue assegna un nome della risorsa Amazon (ARN) al set di regole.

punteggio di qualità dei dati

La percentuale di regole di qualità dei dati che vengono approvate (risultano vere) quando si valuta un set di regole con AWS Glue Data Quality.

osservazione

Informazioni non confermate generate da AWS Glue analizzando le statistiche sui dati raccolte da regole e analizzatori nel tempo.

Note di rilascio per la qualità AWS Glue dei dati

Questo argomento descrive le funzionalità introdotte in AWS Glue Data Quality.

Disponibilità generale: nuove funzionalità

Le seguenti nuove funzionalità sono disponibili con la disponibilità generale di AWS Glue Data Quality:

- La capacità di identificare quali record non hanno superato i controlli di qualità dei dati è ora supportata in AWS Glue Studio
- Nuovi tipi di regole sulla qualità dei dati, come la convalida dell'integrità referenziale dei dati tra due set di dati, il confronto dei dati tra due set di dati e il controllo dei tipi di dati
- Esperienza utente migliorata in AWS Glue Data Catalog
- Supporto per Apache Iceberg, Apache Hudi e Delta Lake

- Supporto per Amazon Redshift
- Invio semplificato di notifiche con Amazon EventBridge
- AWS CloudFormation supporto per la creazione di set di regole
- Miglioramenti delle prestazioni: opzione di memorizzazione nella cache in ETL e AWS Glue Studio per prestazioni più rapide nella valutazione della qualità dei dati

27 novembre 2023 (anteprima)

- Le funzionalità di rilevamento delle anomalie basate su ML sono ora disponibili in AWS Glue ETL e AWS Glue Studio. In questo modo, ora puoi rilevare anomalie e problemi di qualità dei dati. hard-to-detect
- Le regole dinamiche consentono di fornire soglie dinamiche (ad es. `RowCount> avg(last(10))`).

12 marzo 2024

- Support per parole chiave come NULL, BLANKS, WHITESPACES_ONLY
- Correzione di bug: ColumnValues ora fallirà quando le righe hanno valori NULL
- Opzione per valutare le regole composite

Rilevamento delle anomalie in Qualità dei dati di AWS Glue

> **Note**
>
> Qualità dei dati di AWS Glue è disponibile in anteprima nelle regioni seguenti:
>
> - Stati Uniti orientali (Ohio, Virginia settentrionale)
> - Stati Uniti occidentali (Oregon)
> - Asia Pacifico (Tokyo)
> - Europa (Irlanda)

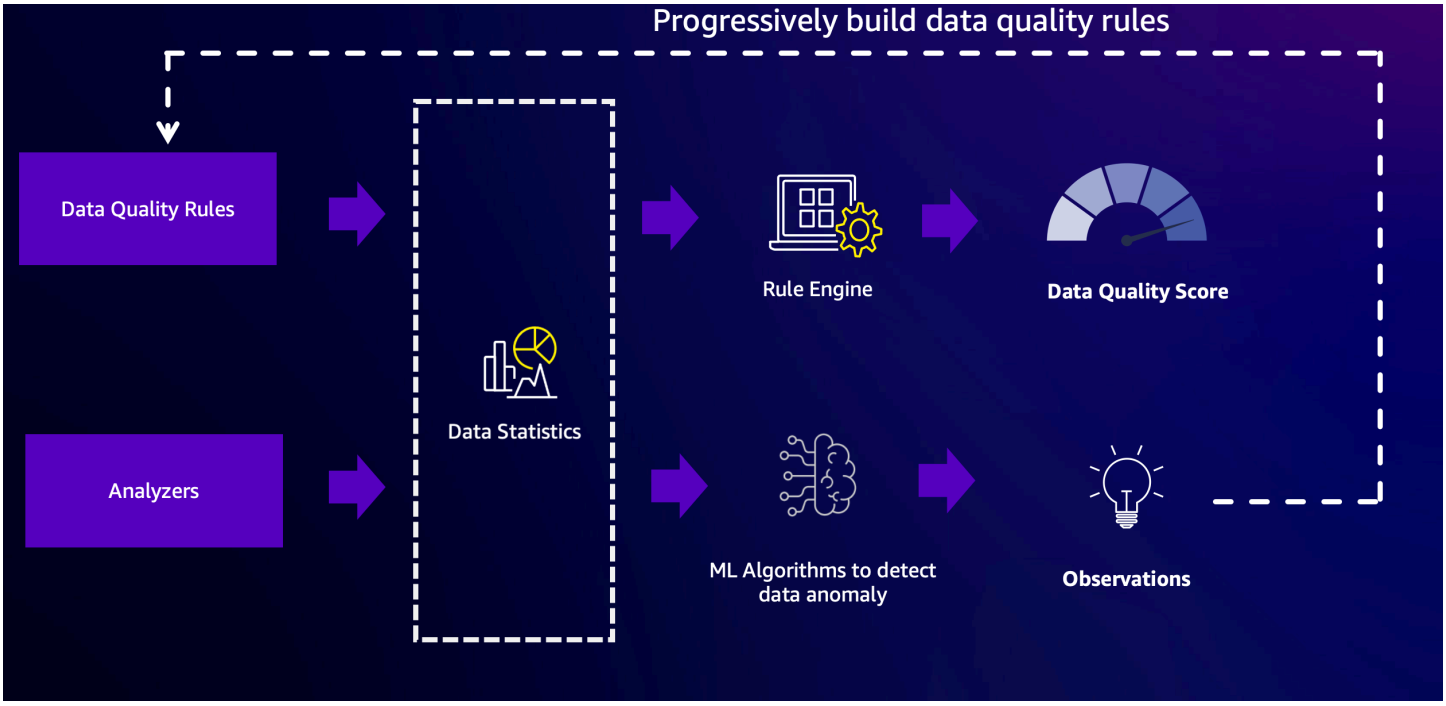
Il rilevamento delle anomalie relative a Qualità dei dati di AWS Glue applica nel tempo algoritmi di machine learning (ML) alle statistiche sui dati per rilevare modelli anomali e problemi nascosti di

qualità dei dati che sono difficili da individuare attraverso le regole. Al momento, il rilevamento delle anomalie è disponibile solo per AWS Glue 4.0. Questa funzionalità è attualmente disponibile solo nell'ETL visivo di AWS Glue Studio e nell'ETL di AWS Glue. Questa funzionalità non è disponibile su notebook AWS Glue Studio, Catalogo dati AWS Glue, sessioni interattive AWS Glue e anteprime dei dati AWS Glue.

Come funziona

Durante la valutazione delle regole di Qualità dei dati, AWS Glue acquisisce le statistiche sui dati necessarie per determinare se questi ultimi sono conformi alle regole. Ad esempio, Qualità dei dati calcolerà il numero di valori distinti in un set di dati e quindi confronterà tale valore con le aspettative.

Il motore delle regole di Qualità dei dati confronta il valore statistico con le soglie definite e valuta i requisiti di qualità. Poiché queste statistiche vengono raccolte nel tempo, puoi abilitare il rilevamento delle anomalie nelle pipeline di ETL per fare in modo che AWS Glue apprenda dalle statistiche precedenti e segnali modelli nascosti sotto forma di osservazioni. Le osservazioni sono informazioni non confermate identificate dall'algoritmo ML di AWS Glue. Tali informazioni vengono fornite insieme a regole di Qualità dei dati suggerite che puoi applicare al set di regole per monitorare il modello rilevato. Ti consigliamo di eseguire i processi a intervalli regolari (ad esempio, ogni ora e ogni giorno). Le esecuzioni irregolari potrebbero produrre informazioni di bassa qualità.



Utilizzo degli analizzatori per ispezionare i dati

A volte, potresti non avere il tempo per creare regole di qualità dei dati. È in questi casi che gli analizzatori sono utili. Gli analizzatori fanno parte del set di regole e sono molto semplici da configurare. Ad esempio, nel tuo set di regole puoi scrivere quanto segue:

```
Analzyers = [
      RowCount,
      Completeness “AllColumns”
]
```

In questo modo verranno raccolte le statistiche seguenti:

- Numero di righe per l'intero set di dati
- Completezza di ogni colonna del set di dati

Ti consigliamo di utilizzare gli analizzatori perché così facendo non dovrai preoccuparti delle soglie. Puoi eseguire le tue pipeline di dati e, dopo tre esecuzioni, Qualità dei dati di AWS Glue inizierà a generare osservazioni e suggerimenti di regole quando rileva eventuali anomalie. Puoi esaminare le osservazioni e le statistiche associate e incorporare facilmente questi suggerimenti nel tuo set di regole. Per iniziare, consulta Configurazione del rilevamento delle anomalie e generazione di informazioni . Tieni presente che gli analizzatori non influiranno sui punteggi di qualità dei dati. Essi producono statistiche che possono essere analizzate nel tempo per generare osservazioni.

Utilizzo della regola DetectAnomaly

Se desideri che un processo abbia esito negativo se rileva anomalie, dovrai applicare un vincolo, ossia configurare una regola. Gli analizzatori non arresteranno un processo, ma raccoglieranno statistiche e analizzeranno i dati. La configurazione della regola `DetectAnomaly` nella sezione relativa alle regole del set di regole confermerà che la scansione DQ riporta che il processo non ha superato tutte le regole della scansione.

Vantaggi e casi d'uso del rilevamento delle anomalie

Gli ingegneri possono gestire centinaia di pipeline di dati in qualsiasi momento. Ogni pipeline è in grado di estrarre dati da origini diverse e caricarli nel data lake. Poiché ogni pipeline può estrarre dati da un'origine diversa e caricarli in un data lake, è difficile ottenere un feedback immediato sui

dati e sapere, ad esempio, se la loro forma è cambiata in modo significativo o se si è discostata dalle tendenze esistenti.

In passato, le origini dati a monte sono cambiate senza che i team di ingegneria dei dati ricevessero alcun preavviso, introducendo così in questo processo "bug di dati" difficili da tracciare. L'aggiunta di nodi Qualità dei dati ai processi rende tutto più semplice, in quanto i processi falliscono quando vengono individuati problemi. Tuttavia, ciò non elimina tutte le modalità di errore che preoccupano i team di dati, il che non impedisce l'insorgenza di altri bug di dati.

Una delle modalità di errore riguarda il volume dei dati. Man mano che l'archivio dati di un'azienda cresce nel tempo, il numero di record prodotti dalle pipeline di dati può aumentare in modo esponenziale. Ogni settimana, i team di dati potrebbero dover aggiornare manualmente i processi ETL per aumentare ogni regola di Qualità dei dati che stabilisce un limite per il numero di righe importate.

Un'altra modalità di errore è rappresentata dal fatto che alcuni limiti delle regole di qualità dei dati sono molto ampi per tenere conto delle variazioni che il volume delle transazioni subisce in base al giorno della settimana. Nel fine settimana, infatti, non si verifica quasi alcuna transazione, mentre il lunedì ci sono circa tre volte più transazioni rispetto agli altri giorni feriali. I team di dati hanno due opzioni: implementare la logica per modificare in modo immediato il set di regole a seconda del giorno oppure impostare aspettative molto ampie.

Infine, i team si occupano anche di bug di dati meno definiti. I modelli sono stati addestrati su dati con caratteristiche specifiche e se questi dati iniziano a distorcersi in modi inaspettati, il team ha bisogno di esserne al corrente. Ad esempio, a febbraio un'azienda può espandersi nel Montana, quindi le transazioni avviate contenenti il codice "MT" vengono visualizzate più frequentemente. Ciò potrebbe interrompere l'inferenza ML e, di conseguenza, i modelli potrebbero prevedere erroneamente che ogni singola transazione nel Montana sia fraudolenta.

In questo caso il rilevamento delle anomalie della qualità dei dati può essere utile per risolvere il problema. Alcuni dei vantaggi del rilevamento delle anomalie di Qualità dei dati includono:

- La scansione dei dati pianificata, basata sugli eventi o manuale.
- Il rilevamento di anomalie che possono indicare un evento imprevisto, una stagionalità o un'anomalia statistica.
- I suggerimenti di regole per intervenire sulle osservazioni segnalate dal rilevamento di anomalie di Qualità dei dati.

Ciò è utile se:

- desideri rilevare automaticamente le anomalie nei dati, senza dover scrivere regole di qualità dei dati.
- desideri individuare potenziali problemi nei dati che le sole regole di qualità dei dati non sono in grado di rilevare.
- desideri automatizzare alcune attività che si evolvono nel tempo, come la limitazione del numero di righe importate per il monitoraggio della qualità dei dati.

Configurazione delle autorizzazioni IAM per Qualità dei dati di AWS Glue

Questo argomento fornisce informazioni sulle operazioni e le risorse che un amministratore IAM può utilizzare in una policy AWS Identity and Access Management (IAM) per Qualità dei dati di AWS Glue. Include anche policy IAM di esempio con le autorizzazioni minime necessarie per utilizzare AWS Glue Data Quality con il catalogo dati AWS Glue.

Per ulteriori informazioni sulla sicurezza in AWS Glue, consulta Sicurezza in AWS Glue.

Autorizzazioni IAM per AWS Glue Data Quality

La tabella seguente elenca le autorizzazioni richieste a un utente per eseguire operazioni di Qualità dei dati di AWS Glue specifiche. Per impostare un'autorizzazione dettagliata per Qualità dei dati di AWS Glue, puoi specificare le seguenti operazioni nell'elemento `Action` di un'istruzione di policy IAM.

Operazioni di AWS Glue Data Quality

Operazione	Descrizione	Tipi di risorsa
`glue:CreateDataQualityRuleset`	Concede l'autorizzazione per creare un set di regole di qualità dei dati.	`::dataQualityRuleset/<name>`
`glue:DeleteDataQualityRuleset`	Concede l'autorizzazione per eliminare un set di regole di qualità dei dati.	`::dataQualityRuleset/<name>`

Operazione	Descrizione	Tipi di risorsa
`glue:GetDataQualityRuleset`	Concede l'autorizzazione per recuperare un set di regole di qualità dei dati.	`::dataQualityRuleset/<name>`
`glue:ListDataQualityRulesets`	Concede l'autorizzazione per recuperare tutti i set di regole di qualità dei dati.	`::dataQualityRuleset/*`
`glue:UpdateDataQualityRuleset`	Concede l'autorizzazione per aggiornare un set di regole di qualità dei dati.	`::dataQualityRuleset/<name>`
`glue:GetDataQualityResult`	Concede l'autorizzazione per recuperare un risultato di esecuzione dell'attività di qualità dei dati.	`::dataQualityRuleset/<name>`
`glue:ListDataQualityResults`	Concede l'autorizzazione per recuperare i risultati di tutte le esecuzioni delle attività di qualità dei dati.	`::dataQualityRuleset/*`
`glue:CancelDataQualityRuleRecommendationRun`	Concede l'autorizzazione per interrompere l'esecuzione dell'attività di raccomandazione di qualità dei dati in corso.	`::dataQualityRuleset/*`
`glue:GetDataQualityRuleRecommendationRun`	Concede l'autorizzazione per recuperare l'esecuzione di un'attività di raccomandazione di qualità dei dati.	`::dataQualityRuleset/*`

Operazione	Descrizione	Tipi di risorsa
glue:ListDataQualityRuleRecommendationRuns	Concede l'autorizzazione per recuperare tutte le esecuzioni dell'attività di raccomandazione di qualità dei dati.	::dataQualityRuleset/*
glue:StartDataQualityRuleRecommendationRun	Concede l'autorizzazione per iniziare l'esecuzione di un'attività di raccomandazione di qualità dei dati.	::dataQualityRuleset/*
glue:CancelDataQualityRulesetEvaluationRun	Concede l'autorizzazione per interrompere l'esecuzione dell'attività di qualità dei dati in corso.	::dataQualityRuleset/*
glue:GetDataQualityRulesetEvaluationRun	Concede l'autorizzazione per recuperare una esecuzione dell'attività di qualità dei dati.	::dataQualityRuleset/*
glue:ListDataQualityRulesetEvaluationRuns	Concede l'autorizzazione per recuperare tutte le esecuzioni dell'attività di qualità dei dati.	::dataQualityRuleset/*
glue:StartDataQualityRulesetEvaluationRun	Concede l'autorizzazione per iniziare l'esecuzione di un'attività di qualità dei dati.	::dataQualityRuleset/<name>
glue:PublishDataQuality	Concede l'autorizzazione per pubblicare i risultati di qualità dei dati.	::dataQualityRuleset/<name>

Configurazione IAM richiesta per la pianificazione delle esecuzioni di valutazione

Autorizzazioni IAM

Per eseguire esecuzioni di valutazione pianificate della qualità dei dati, devi aggiungere l'operazione `IAM:PassRole` alla policy delle autorizzazioni.

Autorizzazioni richieste per il pianificatore AWS EventBridge

Operazione	Descrizione	Tipi di risorsa
`iam:PassRole`	Concede a IAM l'autorizzazione per consentire all'utente di trasmettere i ruoli approvati.	ARN del ruolo utilizzato per chiamare `StartDataQualityRulesetEvaluationRun`

Senza queste autorizzazioni si verifica il seguente errore:

```
"errorCode": "AccessDenied"
"errorMessage": "User: arn:aws:sts::account_id:assumed-role/AWSGlueServiceRole is not
authorized to perform: iam:PassRole on resource: arn:aws:iam::account_id:role/service-
role/AWSGlueServiceRole
because no identity-based policy allows the iam:PassRole action"
```

Entità attendibili di IAM

I servizi AWS Glue e Pianificatore AWS EventBridge devono essere elencati tra le entità attendibili per creare ed eseguire una `StartDataQualityEvaluationRun` pianificata.

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

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "scheduler.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Policy IAM di esempio

Un ruolo IAM per AWS Glue Data Quality richiede i seguenti tipi di autorizzazioni:

- Autorizzazioni per le operazioni di AWS Glue Data Quality in modo da poter ottenere le regole di qualità dei dati consigliate ed eseguire un'attività di qualità dei dati su una tabella nel catalogo dati AWS Glue. Gli esempi di policy IAM in questa sezione includono le autorizzazioni minime richieste per le operazioni di AWS Glue Data Quality.
- Autorizzazioni che concedono l'accesso alla tabella del catalogo dati e ai dati sottostanti. Queste autorizzazioni sono diverse a seconda del caso d'uso. Ad esempio, per i dati che cataloghi in Amazon S3, le autorizzazioni devono includere l'accesso ad Amazon S3.

> **Note**
>
> È necessario configurare le autorizzazioni Amazon S3 oltre alle autorizzazioni descritte in questa sezione.

Autorizzazioni minime per ottenere le regole di qualità dei dati consigliate

Questa policy di esempio include le autorizzazioni necessarie per generare regole di qualità dei dati consigliate.

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

```
        "glue:GetDataQualityRuleRecommendationRun",
        "glue:PublishDataQuality",
        "glue:CreateDataQualityRuleset"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*"
    },
 {
     "Sid": "AllowCatalogPermissions",
     "Effect": "Allow",
     "Action": [
        "glue:GetPartitions",
        "glue:GetTable"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowS3GetObjectToRunRuleRecommendationTask",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::aws-glue-*"
    },
 {   // Optional for Logs
        "Sid": "AllowPublishingCloudwatchLogs",
        "Effect": "Allow",
        "Action": [
          "logs:CreateLogStream",
          "logs:CreateLogGroup",
          "logs:PutLogEvents"
        ],
        "Resource": "*"
    },
  ]
}
```

Autorizzazioni minime per l'esecuzione di un'attività di qualità dei dati

Questa policy di esempio include le autorizzazioni necessarie per eseguire un'attività di valutazione della qualità dei dati.

Le seguenti dichiarazioni di policy sono facoltative, a seconda del caso d'uso:

- `AllowCloudWatchPutMetricDataToPublishTaskMetrics` - Obbligatorio se desideri pubblicare i parametri di esecuzione della qualità dei dati in Amazon CloudWatch.
- `AllowS3PutObjectToWriteTaskResults` - Obbligatorio se desideri scrivere i risultati di esecuzione della qualità dei dati su Amazon S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGlueGetDataQualityRuleset",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityRuleset"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/<YOUR-
RULESET-NAME>"
    },
    {
      "Sid": "AllowGlueRulesetEvaluationRunActions",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityRulesetEvaluationRun",
        "glue:PublishDataQuality"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*"
    },
 {
      "Sid": "AllowCatalogPermissions",
      "Effect": "Allow",
      "Action": [
        "glue:GetPartitions",
        "glue:GetTable"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowS3GetObjectForRulesetEvaluationRun",
      "Effect": "Allow",
```

```
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::aws-glue-*"
    },
    {
      "Sid": "AllowCloudWatchPutMetricDataToPublishTaskMetrics",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": "Glue Data Quality"
        }
      }
    },
    {
      "Sid": "AllowS3PutObjectToWriteTaskResults",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject*"
      ],
      "Resource": "arn:aws:s3:::<YOUR-BUCKET-NAME>/*"
    }
  ]
}
```

Nozioni di base su AWS Glue Data Quality per Data Catalog

Questa sezione introduttiva fornisce istruzioni per aiutarti a iniziare a utilizzare AWS Glue Data Quality sulla console AWS Glue. Imparerai come completare attività essenziali come la generazione di raccomandazioni di regole di qualità dei dati e la valutazione di un set di regole rispetto ai propri dati.

Argomenti

- Monitoraggio dei suggerimenti di regole
- Modifica dei set di regole suggeriti
- Creazione di un nuovo set di regole
- Esecuzione di un set di regole per valutare la qualità dei dati
- Visualizzazione del punteggio e dei risultati della qualità dei dati
- Argomenti correlati

Prerequisiti

Prima di utilizzare AWS Glue Data Quality, è necessario conoscere l'utilizzo di Data Catalog e dei crawler in AWS Glue. Con AWS Glue Data Quality, è possibile valutare la qualità delle tabelle in un database Data Catalog. Devi disporre anche dei seguenti elementi:

- Una tabella nel Data Catalog rispetto alla quale valutare il set di regole di qualità dei dati.
- Un ruolo IAM per AWS Glue fornito quando si generano i suggerimenti di regole o se si esegue un'attività di qualità dei dati. Questo ruolo deve disporre dell'autorizzazione per l'accesso alle risorse che vari processi AWS Glue Data Quality richiedono per l'esecuzione per tuo conto. Queste risorse includono AWS Glue Amazon S3 e. CloudWatch Per visualizzare policy di esempio che includono le autorizzazioni minime per AWS Glue Data Quality, consulta la pagina Policy IAM di esempio.

  Per ulteriori informazioni sui ruoli IAM per AWS Glue, consulta le pagine Create an IAM policy for the AWS Glue service e Create an IAM role for the AWS Glue service. È inoltre possibile consultare un elenco di tutte le autorizzazioni AWS Glue specifiche per la qualità dei dati nella pagina Authorization for AWS Glue Data Quality actions.
- Un database con almeno una tabella che contiene una varietà di dati. La tabella utilizzata in questo tutorial è denominata `yyz-tickets`, con la tabella `tickets`. Questi dati sono una raccolta di informazioni disponibili al pubblico dalla città di Toronto per le violazioni in materia di sosta. Se crei la tua tabella, assicurati che sia compilata con una serie di dati validi per ottenere il miglior set di regole suggerite.

Un tep-by-step esempio

Per un step-by-step esempio con set di dati di esempio, consulta il post sul blog AWS Glue Data Quality.

Generazione di raccomandazioni di regole

I suggerimenti di regole consentono di iniziare a utilizzare facilmente la qualità dei dati senza scrivere codice. Con Qualità dei dati di AWS Glue è possibile analizzare i dati, identificare le regole e creare un set di regole che possono essere valutate in un'attività di qualità dei dati. Le esecuzioni di consigli vengono eliminate automaticamente dopo 90 giorni.

Generazione di raccomandazioni di regole di qualità dei dati

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, seleziona Tables (Tabelle). Scegliere quindi la tabella per la quale si desidera generare le raccomandazioni di regole di qualità dei dati.
3. Nella pagina dei dettagli della tabella, scegli la scheda Qualità dei dati per accedere alle regole e alle impostazioni di Qualità dei dati di AWS Glue per la tabella.
4. Nella scheda Qualità dei dati, scegli Aggiungi regole e monitora la qualità dei dati.
5. Nella pagina Generatore set di regole, un avviso nella parte superiore della pagina ti chiederà di avviare un'attività di suggerimento se non sono presenti esecuzioni di suggerimenti di regole.
6. Scegli Regole suggerite per aprire il modale e inserisci i parametri per l'attività di suggerimento.
7. Scegli un ruolo IAM con accesso ad AWS Glue. Questo ruolo deve disporre dell'autorizzazione per l'accesso alle risorse che vari processi AWS Glue Data Quality richiedono per l'esecuzione a tuo nome.
8. Dopo aver completato i campi in base alle tue preferenze, scegli Suggerisci regole per avviare l'esecuzione dell'attività di suggerimento. Se le esecuzioni di suggerimento sono in corso o completate, puoi gestirle in questo avviso. Potrebbe essere necessario aggiornare l'avviso per visualizzare la modifica dello stato. Le esecuzioni delle attività di suggerimento completate e in corso vengono visualizzate nella pagina Cronologia delle esecuzioni, che elenca tutte le esecuzioni di suggerimento effettuate negli ultimi 90 giorni.

Cosa significano le regole suggerite

Qualità dei dati di AWS Glue genera regole basate sui dati di ogni colonna della tabella di input. Utilizza le regole per identificare i potenziali limiti entro i quali i dati possono essere filtrati per mantenere i requisiti di qualità. Il seguente elenco di regole generate include esempi utili per comprendere il significato delle regole e gli effetti che potrebbero avere se applicate ai dati.

Per un elenco completo dei tipi di regole Data Quality Definition Language (DQDL) generati, consulta la pagina DQDL rule type reference.

- `IsComplete "SET_FINE_AMOUNT"`: la regola `IsComplete` verifica che la colonna sia compilata in ogni riga specificata. Utilizza questa regola per contrassegnare le colonne come non facoltative nei dati.
- `Uniqueness "TICKET_NUMBER" > 0.95`: la regola `Uniqueness` verifica che i dati all'interno della colonna soddisfino una certa soglia di unicità. In questo esempio, è stato determinato che i dati che compongono una determinata riga per `"TICKET_NUMBER"` sono identici al massimo al 95% nel contenuto a tutte le altre righe, il che suggerisce questa regola.
- `ColumnValues "PROVINCE" in ["ON", "QC", "AB", "NY",...]`: la regola `ColumnValues` definisce valori validi per la colonna in base al contenuto della colonna esistente. In questo esempio, i dati per ogni riga sono una targa di 2 lettere per uno stato o una provincia.
- `ColumnLength "INFRACTION_DESCRIPTION" between 15 and 31`: la regola `ColumnLength` impone una limitazione di lunghezza sui dati di una colonna. Questa regola viene generata dai dati di esempio in base alle lunghezze minime e massime registrate per una colonna di stringhe.

Monitoraggio dei suggerimenti di regole

Quando sono in esecuzione i suggerimenti sulle regole di qualità dei dati, la pagina Aggiungi regole e monitora la qualità dei dati visualizza informazioni e operazioni aggiuntive che è possibile intraprendere nella barra superiore.

Quando sono in corso le esecuzioni dei suggerimenti sulle regole, puoi scegliere Interrompi esecuzione prima del completamento dell'attività di suggerimento. Mentre l'attività è in corso, vedrai lo stato In corso e la data e l'ora di inizio dell'esecuzione.

Una volta completati i suggerimenti di regole, la barra di suggerimento mostra il numero di regole suggerite, lo stato dell'ultima esecuzione di suggerimento e la data e l'ora del termine.

È possibile aggiungere le regole suggerite scegliendo Inserisci suggerimento di regola. Per visualizzare le regole precedentemente suggerite, seleziona una data specifica. Per eseguire un nuovo suggerimento, scegli Altre operazioni, quindi scegli Regole suggerite.

Configura le impostazioni predefinite scegliendo Gestisci impostazioni utente. È possibile impostare il percorso predefinito in cui Amazon S3 può archiviare i set di regole o configurare un ruolo predefinito per eseguire Catalogo dati.

Modifica dei set di regole suggeriti

Poiché Qualità dei dati di AWS Glue genera regole basate sui dati esistenti che hai a disposizione, potresti vedere alcune regole impreviste o indesiderate nei suggerimenti automatici. Per ottenere il massimo dai set di regole suggeriti, è necessario valutarli e modificarli. In questo passaggio del tutorial, prendi le regole generate nel passaggio precedente e le modifichi per applicare qualità più restrittive su alcuni dati. Inoltre, allenterai altre regole per garantire che dati corretti e univoci possano essere aggiunti in un secondo momento.

Modifica un set di regole suggerito

1. Nella console AWS Glue, scegli Catalogo dati, quindi scegli Tabelle database nel riquadro di navigazione. Seleziona la tabella `tickets`.
2. Nella pagina dei dettagli della tabella, scegli la scheda Qualità dei dati per accedere alle opzioni Qualità dei dati di AWS Glue per la tabella.
3. Nella sezione Set di regole, seleziona il set di regole generato in Generazione di raccomandazioni di regole.
4. Scegli Operazioni, quindi scegli Modifica nella finestra della console. L'editor del set di regole viene caricato nella console. Include un riquadro di modifica delle regole e un riferimento rapido per DQDL.
5. Rimuovi la riga 2 dello script. Ciò allenta il requisito che prevede che la dimensione del database sia limitata entro un certo numero di righe. Dopo la modifica, il file dovrebbe contenere quanto segue nelle righe 1-3:

```
Rules = [
    IsComplete "TAG_NUMBER_MASKED",
    ColumnLength "TAG_NUMBER_MASKED" between 6 and 9,
```

6. Rimuovi la riga 25 dello script. Ciò allenta il requisito che prevede che il 96% delle province registrate sia 0N. Dopo la modifica, il file dovrebbe contenere quanto segue dalla riga 24 alla fine del set di regole:

```
ColumnValues "PROVINCE" in ["ON", "QC", "AB", "NY", "AZ", "NS", "BC", "MI", "PQ",
 "MB", "PA", "FL", "SK", "NJ", "OH", "NB", "IL", "MA", "CA",
    "VA", "TX", "NF", "MD", "PE", "CT", "NC", "GA", "IN", "OR", "MN", "TN", "WI",
 "KY", "MO", "WA", "NH", "SC", "CO", "OK", "VT", "RI", "ME", "AL",
    "YT", "IA", "DE", "AR", "LA", "XX", "WV", "MT", "KS", "NT", "DC", "NV", "NE",
 "UT", "MS", "NM", "ID", "SD", "ND", "AK", "NU", "GO", "WY", "HI"],
```

```
ColumnLength "PROVINCE" = 2
]
```

7. Modifica la riga 14 come segue:

```
IsComplete "TIME_OF_INFRACTION",
```

   Ciò rafforza il requisito relativo alla colonna limitando il database ai soli ticket che contengono un orario di infrazione registrato. Nel contesto di questo set di dati, è importante considerare i ticket senza un orario di infrazione registrato come dati non validi. In alcune situazioni, potrebbe essere più appropriato considerare il partizionamento o la trasformazione dei dati al fine di consentire un ulteriore utilizzo o ispezione dei dati per determinare una regola di qualità.

8. Scegli Aggiorna set di regole nella parte inferiore della pagina della console.

Creazione di un nuovo set di regole

Un set di regole è un gruppo di regole di qualità dei dati che vengono valutate in base ai tuoi dati. Nella console AWS Glue, puoi creare set di regole personalizzati utilizzando Data Quality Definition Language (DQDL).

Creazione di un set di regole di qualità dei dati

1. Nella console AWS Glue, scegli Catalogo dati, scegli Database, quindi scegli Tabelle nel riquadro di navigazione. Seleziona la tabella `tickets`.
2. Apri la scheda Data quality (Qualità dei dati).
3. Nella sezione Set di regole, scegli Crea set di regole. L'editor DQDL viene avviato nella console. Dispone di un'area di testo per la modifica diretta e di un riferimento rapido alle regole DQDL e allo schema delle tabelle.
4. Inizia ad aggiungere regole all'area di testo dell'editor DQDL. Puoi scrivere le regole direttamente da questo tutorial o utilizzare la funzionalità Generatore di regole DQDL dell'editor delle regole sulla qualità dei dati.

> **Note**
>
> Come utilizzare il generatore di regole DQDL
>
> 1. Seleziona un tipo di regola dall'elenco e scegli il segno più per inserire la sintassi di esempio nel riquadro dell'editor.
> 2. Cambia i nomi delle colonne segnaposto con i nomi delle tue colonne. I nomi delle colonne della tabella sono disponibili nella scheda Schema.
> 3. Aggiorna il parametro dell'espressione per adattarlo al tuo caso. Per un elenco completo delle espressioni supportate da DQDL, consulta Espressioni.

Ad esempio, le seguenti regole sono vincoli per la convalida dei dati della colonna `ticket_number` nella tabella `tickets`. Per aggiungere le seguenti regole, utilizza il generatore di regole DQDL o modifica direttamente il tuo set di regole:

```
IsComplete "ticket_number",
IsUnique "ticket_number",
ColumnValues "ticket_number" > 9000000000
```

5. Fornisci un nome per il tuo nuovo set di regole nel campo Nome del set di regole.
6. Scegli Salva set di regole.

Valutazione della qualità dei dati su più set di dati

Puoi impostare regole di qualità dei dati su più set di dati utilizzando i set di regole ReferentialIntegrity e DatasetMatch . ReferentialIntegrityverifica se i dati del set di dati primario sono presenti in altri set di dati.

Per aggiungere un set di dati di riferimento, scegli la scheda Schema, quindi scegli Aggiorna tabelle di riferimento. Ti verrà richiesto di selezionare un database e una tabella. È possibile aggiungere la tabella e quindi impostare le regole di qualità dei dati. Tipi di regole come AggregateMatch,, RowCountMatch ReferentialIntegrity SchemaMatch, e DatasetMatch supportano la possibilità di eseguire controlli di qualità dei dati su più set di dati.

Esecuzione di un set di regole per valutare la qualità dei dati

Quando si esegue un'attività di qualità dei dati, AWS Glue Data Quality valuta un set di regole rispetto ai dati e calcola un punteggio di qualità dei dati. Questo punteggio rappresenta la percentuale di regole di qualità dei dati soddisfatte per l'input.

Esecuzione di un'attività di qualità dei dati

1. Nella console AWS Glue, scegli Catalogo dati, scegli Database, quindi scegli Tabelle nel riquadro di navigazione. Seleziona la tabella `tickets`.
2. Scegli la scheda Qualità dei dati.
3. Nell'elenco Set di regole, scegli il set di regole rispetto al quale desideri valutare la tabella. Per questo passaggio, ti consigliamo di utilizzare un set di regole che hai già scritto o modificato anziché regole generate. Scegli Esegui.
4. Nel modale, scegli il tuo ruolo IAM. Questo ruolo deve disporre dell'autorizzazione per l'accesso alle risorse che vari processi AWS Glue Data Quality richiedono per l'esecuzione a tuo nome. È possibile salvare il ruolo IAM come predefinito o modificarlo accedendo alla pagina Impostazioni predefinite.
5. In Azioni sulla qualità dei dati, scegli se pubblicare le metriche su Amazon CloudWatch. Se si seleziona questa opzione, Qualità dei dati di AWS Glue pubblica i parametri che indicano il numero di regole soddisfatte e il numero di regole non soddisfatte. Per intervenire sulle metriche archiviate in questo modo, puoi utilizzare CloudWatch gli allarmi. Su Amazon EventBridge vengono inoltre pubblicati i parametri chiave per consentirti di impostare gli avvisi. Per ulteriori informazioni, consulta la pagina Setting up alerts, deployments, and scheduling.
6. In Frequenza di esecuzione, scegli l'esecuzione on demand oppure pianifica il set di regole. Quando pianifichi un set di regole, ti viene richiesto un nome per l'attività. La pianificazione verrà creata in Amazon EventBridge. Puoi modificare la tua pianificazione in Amazon EventBridge.
7. Per salvare i risultati della qualità dei dati in Amazon S3, scegli una Posizione per i risultati della qualità dei dati. Il ruolo IAM selezionato in precedenza per questa attività deve avere accesso di scrittura a questa posizione.
8. In Configurazioni aggiuntive, inserisci il Numero di worker richiesto che desideri che AWS Glue allochi per la tua attività di qualità dei dati.
9. Facoltativamente, puoi impostare un filtro nell'origine dati. Questo contribuisce a ridurre i dati in fase di lettura. È inoltre possibile utilizzare un filtro per eseguire convalide incrementali selezionando le informazioni sulle partizioni e trasmettendole come parametri tramite chiamate API. Per migliorare le prestazioni, puoi fornire un predicato di partizione.

10. Scegli Esegui. La nuova attività dovrebbe essere riportata nell'elenco di esecuzioni delle attività relative alla qualità dei dati. Quando la colonna Stato di esecuzione dell'attività è visualizzata come Completata, è possibile visualizzare i risultati del punteggio di qualità. Potrebbe essere necessario aggiornare la finestra della console per visualizzare correttamente lo stato.
11. Per visualizzare la colonna con i dettagli dei risultati sulla qualità dei dati, scegli l'icona "+" per espandere il set di regole. I risultati mostrano le regole che hanno superato e quelle che non hanno superato la valutazione e cosa ha causato l'errore della regola.

Visualizzazione del punteggio e dei risultati della qualità dei dati

Consultazione dell'ultima esecuzione su tutti i set di regole creati

1. Nella console AWS Glue, scegli Tables (Tabelle) nel pannello di navigazione. Scegliere quindi la tabella per la quale si desidera eseguire un'attività di qualità dei dati.
2. Scegli la scheda Qualità dei dati.
3. Snapshot della qualità dei dati mostra una tendenza generale delle esecuzioni nel tempo. Per impostazione predefinita, vengono visualizzate le ultime 10 esecuzioni di tutti i set di regole. Per filtrare per set di regole, seleziona quello desiderato dall'elenco a discesa. Se ci sono meno di 10 esecuzioni, vengono visualizzate tutte le esecuzioni completate disponibili.
4. Nella tabella Qualità dei dati, viene mostrato ogni set di regole con l'ultima esecuzione, se presente, insieme al relativo punteggio. L'espansione del set di regole mostra le regole presenti in quel set di regole insieme ai risultati delle regole per tale esecuzione.

Consultazione dell'ultima esecuzione su un particolare set di regole

1. Nella console AWS Glue, scegli Tables (Tabelle) nel pannello di navigazione. Scegliere quindi la tabella per la quale si desidera eseguire un'attività di qualità dei dati.
2. Scegli la scheda Qualità dei dati.
3. Nella tabella Qualità dei dati, scegli un set di regole specifico.
4. Nella pagina Dettagli del set di regole, scegli la scheda Cronologia di esecuzione.

   Tutte le esecuzioni di valutazione per questo particolare set di regole sono elencate nella tabella all'interno di questa scheda. È possibile visualizzare la cronologia dei punteggi e lo stato delle esecuzioni.

5. Per visualizzare ulteriori informazioni su una determinata esecuzione, scegli ID esecuzione per accedere alla pagina Dettagli dell'esecuzione di valutazione. In questa pagina, puoi visualizzare informazioni specifiche sull'esecuzione e ulteriori dettagli sullo stato dei risultati delle singole regole.

Argomenti correlati

- Documentazione di riferimento del tipo di regola DQDL
- Riferimento a Data Quality Definition Language (DQDL)

Valutazione della qualità dei dati con AWS Glue Studio

AWS Glue Data Quality consente di valutare e monitorare la qualità dei dati in base alle regole definite. In questo modo è facile identificare i dati che richiedono un'azione. In AWS Glue Studio, puoi aggiungere nodi di qualità dei dati al tuo processo visivo per creare regole di qualità dei dati sulle tabelle del catalogo dati. Potrai quindi monitorare e valutare le modifiche ai set di dati nel corso del tempo. Per una panoramica su come utilizzare Qualità dei dati di AWS Glue in AWS Glue Studio, guarda il seguente video.

Di seguito sono riportati i passaggi di livello superiore per lavorare con Qualità dei dati di AWS Glue:

1. Create data quality rules (Crea regole di qualità dei dati): crea un set di regole di qualità dei dati utilizzando il generatore DQDL scegliendo i set di regole incorporati configurati.
2. Configure a data quality job (Configura un processo di qualità dei dati): definisci le azioni in base ai risultati della qualità dei dati e alle opzioni di output.
3. Salva ed esegui un processo con la qualità dei dati: crea ed esegui un processo. Il salvataggio del processo salverà i set di regole creati per il processo.
4. Monitor and review the data quality results (Monitora ed esamina i risultati della qualità dei dati): esamina i risultati della qualità dei dati al termine dell'esecuzione del processo. Facoltativamente, pianifica il processo per una data futura.

Vantaggi

Data analyst, data engineer e data scientist possono utilizzare il nodo di valutazione della qualità dei dati in AWS Glue Studio per analizzare, configurare, monitorare e migliorare la qualità dei dati dall'editor di processi visivi. I vantaggi dell'utilizzo del nodo di qualità dei dati includono i seguenti:

- È possibile rilevare problemi di qualità dei dati: puoi verificare la presenza di problemi creando regole che controllano le funzionalità dei set di dati.
- Iniziare è facile: puoi iniziare utilizzando regole e operazioni predefinite.
- Integrazione perfetta: è possibile utilizzare i nodi di qualità dei dati in AWS Glue Studio perché Qualità dei dati di AWS Glue viene eseguito su Catalogo dati AWS Glue.

Valutazione della qualità dei dati per i processi ETL in AWS Glue Studio

In questo tutorial, inizierai a usare Qualità dei dati di AWS Glue in AWS Glue Studio. Imparerai a:

- Creare regole utilizzando il generatore di regole Data Quality Definition Language (DQDL).
- Specificare le azioni di qualità dei dati, i dati da emettere e la posizione di output dei risultati della qualità dei dati.
- Esaminare i risultati della qualità dei dati.

Per fare pratica con un esempio, consulta il post del blog Getting started with AWS Glue Data Quality for ETL pipelines.

Passaggio 1: aggiunta del nodo Valuta la qualità dei dati al processo visivo

In questo passaggio, verrà aggiunto il nodo di valutazione della qualità dei dati all'editor del processo visivo.

Aggiunta del nodo di qualità dei dati

1. Nella console AWS Glue Studio, scegli Visivo con un'origine e una destinazione dalla sezione Crea processo , quindi scegli Crea.
2. Scegli un nodo al quale desideri applicare la trasformazione della qualità dei dati. In genere, si tratta di un nodo di trasformazione o di un'origine dati.
3. Apri il pannello delle risorse a sinistra scegliendo l'icona "+". È inoltre possibile digitare Valuta la qualità dei dati nella barra di ricerca e quindi scegliere Valuta la qualità dei dati dai risultati della ricerca.
4. L'editor del processo visivo mostrerà il nodo di trasformazione Valuta la qualità dei dati che si dirama dal nodo selezionato. Sul lato destro della console, la scheda Transform (Trasforma) è aperta automaticamente. Se devi modificare il nodo padre, scegli la scheda Proprietà del nodo, quindi scegli il nodo padre dal menu a discesa.

Quando si sceglie un nuovo nodo principale, viene stabilita una nuova connessione tra il nodo principale e il nodo Evaluate Data Quality (Valuta la qualità dei dati). Rimuovi tutti i nodi principali indesiderati. È possibile collegare un solo nodo principale a un nodo Evaluate Data Quality (Valuta la qualità dei dati).

5. La trasformazione Valuta la qualità dei dati supporta più padri per consentire di convalidare le regole di qualità dei dati su più set di dati. Le regole che supportano più set di dati includono ReferentialIntegrity, DataSetMatch, SchemaMatch, RowCountMatch e AggregateMatch.

   Se aggiungi più input alla trasformazione Valuta la qualità dei dati, devi selezionare l'input "primario". L'input primario è il set di dati del quale desideri convalidare la qualità dei dati. Tutti gli altri nodi o input vengono trattati come riferimenti.

   È possibile utilizzare la trasformazione Valuta la qualità dei dati per identificare record specifici che non hanno superato i controlli di qualità dei dati. Ti consigliamo di scegliere il set di dati primario perché le nuove colonne che segnalano i record non validi vengono aggiunte a tale set di dati.

6. È possibile specificare degli alias per le origini dati di input. Gli alias forniscono un altro modo per fare riferimento all'origine di input quando si utilizza la regola ReferentialIntegrity. Poiché è possibile designare una sola origine dati come origine principale, ogni ulteriore origine dati che aggiungi richiederà un alias.

   Nell'esempio seguente, la regola ReferentialIntegrity specifica l'origine dati di input in base al nome dell'alias ed esegue un confronto uno a uno con l'origine dati primaria.

   ```
   Rules = [
    ReferentialIntegrity “Aliasname.name” = 1
   ]
   ```

Fase 2: creazione di una regola con DQDL

In questa fase, viene creata una regola tramite DQDL. Per questo tutorial, verrà creata una singola regola utilizzando il tipo di regola Completezza. Questo tipo di regola verifica la percentuale di valori completi (non nulli) in una colonna rispetto a una determinata espressione. Per ulteriori informazioni sull'utilizzo di DQDL, consulta la pagina DQDL.

1. Nella scheda Trasforma, aggiungi un Tipo di regola facendo clic sul pulsante Inserisci. Questa operazione aggiunge il tipo di regola all'editor di regole, nel quale è possibile inserire i parametri per la regola.

   > **Note**
   >
   > Quando modifichi le regole, assicurati che le regole siano racchiuse tra parentesi e che siano separate da virgole. Ad esempio, un'espressione di regola completa avrà il seguente aspetto:
   >
   > ```
   > Rules= [
   >     Completeness "year">0.8, Completeness "month">0.8
   > ]
   > ```
   >
   > Questo esempio specifica il parametro di completezza per le colonne denominate "anno" e "mese". Affinché la regola venga soddisfatta, queste colonne devono essere complete per più dell'80% o devono contenere dati in oltre l'80% delle istanze per ogni rispettiva colonna.

   In questo esempio, cerca e inserisci il tipo di regola Completezza. Questa operazione aggiunge il tipo di regola all'editor di regole. Questo tipo di regola ha la seguente sintassi: `Completeness <COL_NAME> <EXPRESSION>`.

   La maggior parte dei tipi di regole richiede la specifica di un'espressione come parametro al fine di creare una risposta booleana. Per ulteriori informazioni sulle espressioni DQDL supportate, consulta la pagina DQDL expressions. Successivamente, aggiungerai il nome della colonna.

2. Nel generatore di regole DQDL, seleziona la scheda Schema. Usa la barra di ricerca per individuare il nome della colonna nello schema di input. Lo schema di input visualizza il nome della colonna e il tipo di dati.

3. Nell'editor di regole, fai clic sulla destra del tipo di regola per inserire il cursore nel punto in cui verrà inserita la colonna. In alternativa, è possibile digitare il nome della colonna nella regola.

   Ad esempio, dall'elenco di colonne nell'elenco dello schema di input, fai clic sul pulsante Inserisci accanto alla colonna (in questo esempio, anno). Questa operazione aggiunge la colonna alla regola.

4. Quindi, nell'editor di regole, aggiungi un'espressione per valutare la regola. Poiché il tipo Completezza verifica la percentuale di valori completi (non nulli) in una colonna rispetto a una determinata espressione, immetti un'espressione come `> 0.8`. Questa regola controlla se la colonna contiene almeno l'80% di valori completi (non nulli).

Passaggio 3: configurazione degli output di qualità dei dati

Dopo aver creato le regole di qualità dei dati, è possibile selezionare opzioni aggiuntive per specificare l'output del nodo della qualità dei dati.

1. In Data quality transform output (Output della trasformazione della qualità dei dati), scegli tra le seguenti opzioni:

   - Dati originali: scegli di emettere i dati di input originali. Quando si sceglie questa opzione, al processo viene aggiunto un nuovo nodo figlio "rowLevelOutcomes". Lo schema corrisponde allo schema del set di dati primario trasmesso come input alla trasformazione. Questa opzione è utile se si desidera soltanto trasmettere i dati e far sì che il processo abbia esito negativo se si verificano problemi di qualità.

     Un altro caso d'uso è quando si desidera rilevare record non validi che non hanno superato i controlli di qualità dei dati. Per rilevare i record non validi, scegli l'opzione Aggiungi nuove colonne per indicare gli errori di qualità dei dati. Questa operazione aggiunge quattro nuove colonne allo schema della trasformazione "rowLevelOutcomes".
     - DataQualityRulesPass (array di stringhe): fornisce una serie di regole che hanno superato i controlli di qualità dei dati.
     - DataQualityRulesFail (array di stringhe): fornisce una serie di regole che non hanno superato i controlli di qualità dei dati.
     - DataQualityRulesSkip (array di stringhe): fornisce una serie di regole che sono state ignorate. Le seguenti regole non possono identificare i record di errore perché vengono applicate a livello di set di dati.
       - AggregateMatch
       - ColumnCount
       - ColumnExists
       - ColumnNamesMatchPattern
       - CustomSql
       - RowCount

    - RowCountMatch
    - StandardDeviation
    - Mean
    - ColumnCorrelation
  - DataQualityEvaluationResult: fornisce lo stato "Riuscito" o "Non riuscito" a livello di riga. Tieni presente che i tuoi risultati complessivi possono essere non riusciti, ma un determinato record potrebbe essere riuscito. Ad esempio, la regola RowCount potrebbe non essere riuscita, ma tutte le altre regole potrebbero aver avuto successo. In questi casi, lo stato di questo campo è "Riuscito".
2. Risultati della qualità dei dati: scegli di visualizzare le regole configurate e il loro stato di riuscita o non riuscita. Questa opzione è utile se desideri scrivere i risultati su Amazon S3 o altri database.
3. Impostazioni di output della qualità dei dati (facoltativo): scegli Impostazioni di output della qualità dei dati per visualizzare il campo Posizione dei risultati della qualità dei dati. Quindi, fai clic su Sfoglia per cercare una posizione Amazon S3 da impostare come destinazione dell'output della qualità dei dati.

Fase 4. Configurazione delle operazioni di qualità dei dati

Sono disponibili operazioni che consentono di pubblicare i parametri su CloudWatch o di interrompere i processi in base a criteri specifici. Le operazioni sono disponibili solo dopo aver creato una regola. Se scegli questa opzione, gli stessi parametri vengono pubblicati anche su Amazon EventBridge. È possibile utilizzare queste opzioni per creare avvisi di notifica.

- In caso di errore del set di regole: è possibile scegliere cosa fare se un set di regole ha esito negativo mentre il processo è in esecuzione. Se desideri che il processo abbia esito negativo se la qualità dei dati non va a buon fine, puoi scegliere quando far fallire il processo selezionando una delle seguenti opzioni. Per impostazione predefinita, questa operazione non è selezionata e l'esecuzione del processo sarà completata anche se le regole di qualità dei dati hanno esito negativo.
  - Nessuno: se scegli Nessuno (impostazione predefinita), il processo non ha esito negativo e continua a essere eseguito nonostante gli errori del set di regole.
  - Abbandona il processo dopo il caricamento dei dati sulla destinazione: il processo ha esito negativo e non viene salvato alcun dato. Per salvare i risultati, scegli una posizione Amazon S3 in cui salvare i risultati sulla qualità dei dati.

- Abbandona il processo senza caricare i dati sulla destinazione: questa opzione determina immediatamente l'esito negativo del processo quando si verifica un errore di qualità dei dati. Non carica alcuna destinazione dati, inclusi i risultati della trasformazione di qualità dei dati.

Passaggio 5: visualizzazione dei risultati della qualità dei dati

Dopo aver eseguito il processo, visualizza i risultati relativi alla qualità dei dati facendo clic sulla scheda Qualità dei dati.

1. Per ogni esecuzione di processo, visualizza i risultati della qualità dei dati. Ogni nodo mostra lo stato della qualità dei dati e i dettagli dello stato. Scegli un nodo per visualizzare tutte le regole e lo stato di ciascuna regola.
2. Fai clic su Scarica risultati per scaricare un file CSV contenente informazioni sull'esecuzione del processo e sui risultati relativi alla qualità dei dati.
3. Se hai più di una esecuzione di processo con risultati di qualità dei dati, puoi filtrare i risultati per intervallo di data e ora. Scegli Filtra per intervallo di data e ora per espandere la finestra del filtro.
4. Puoi scegliere un intervallo relativo o un intervallo assoluto. Per gli intervalli assoluti, utilizza il calendario per selezionare i valori di data e ora per l'ora di inizio e l'ora di fine. Al termine, scegliere Applica.

Generatore di regole di qualità dei dati

Con il generatore di regole Data Quality Definition Language (DQDL), puoi creare regole di qualità dei dati per valutare i tuoi dati. Inizia selezionando un tipo di regola, quindi specifica i parametri nell'editor delle regole. Durante il processo di creazione, l'editor delle regole mostra anche eventuali errori e avvisi.

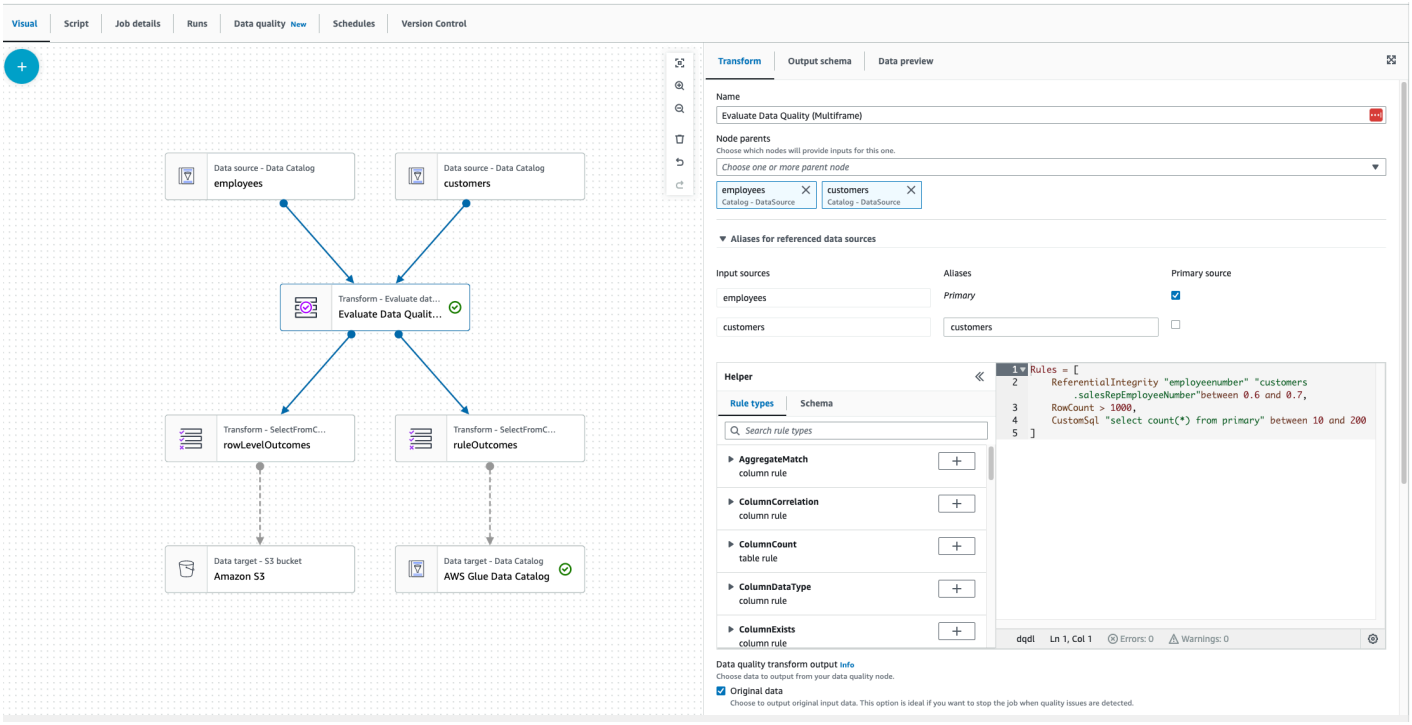
La Guida di DQDL fornisce una documentazione completa su come costruire regole utilizzando la sintassi DQDL, i tipi di regole integrati e gli esempi.

Nodo Evaluate Data Quality (Valuta la qualità dei dati)

Quando si lavora con il nodo di trasformazione Valuta la qualità dei dati e il generatore di regole DQDL, è possibile espandere lo spazio di lavoro.

- Per espandere la scheda Trasforma fino a riempire l'intero schermo, fai clic sull'icona di espansione nell'angolo in alto a destra del pannello dei dettagli del nodo.

- Per espandere l'editor delle regole DQDL, fai clic sull'icona << per espandere l'editor delle regole e comprimere le schede Tipi di regole e Schema.



Componenti

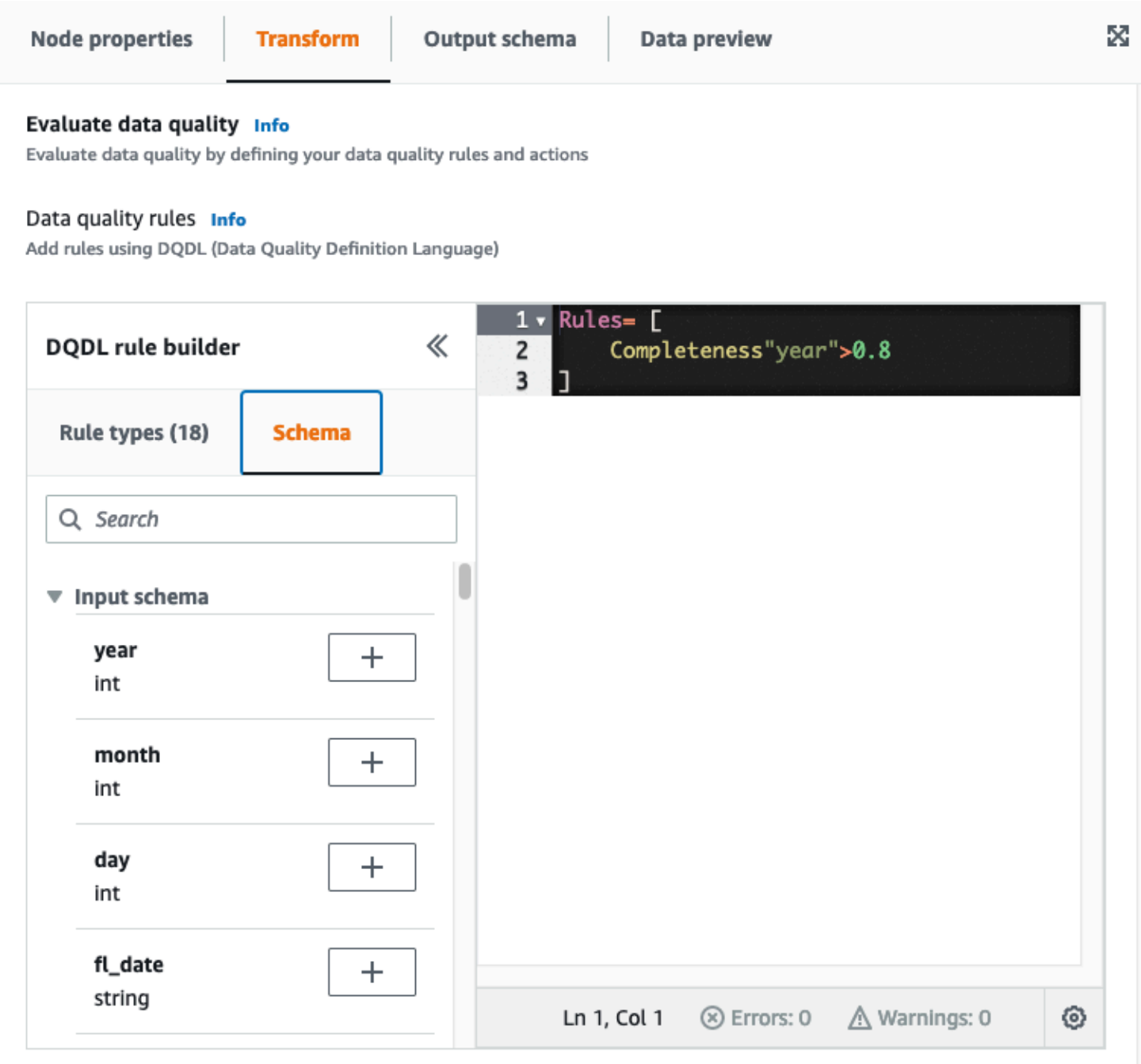
Esistono 26 tipi di regole incorporati in AWS Glue Studio. Ogni tipo di regola riporta una descrizione e vari esempi di come possono essere utilizzati.

Tipi di regole di qualità dei dati

AWS Glue Studio fornisce tipi di regole integrati per semplificare la creazione di una regola. Per ulteriori informazioni sui tipi di regole, consulta Riferimento ai tipi di regole DQDL.

Schema

La scheda Schema mostra i nomi delle colonne e il tipo di dati del nodo principale. Vengono visualizzati gli schemi di più nodi. È possibile visualizzare lo schema di input, effettuare una ricerca per nome della colonna e inserire la colonna nell'editor delle regole.



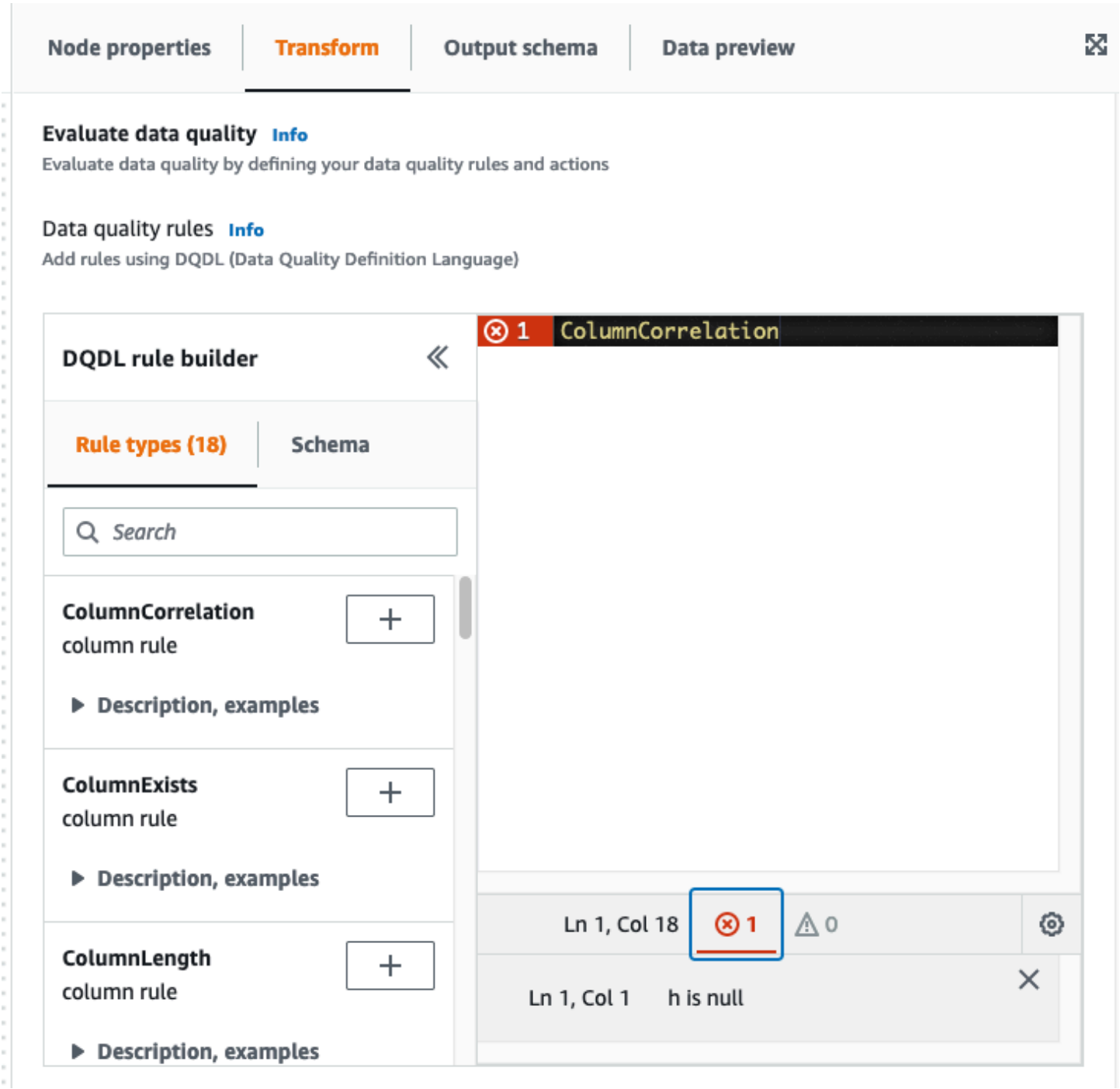
Editor delle regole

L'editor delle regole è un editor di testo in cui è possibile scrivere e modificare le regole. Se si seleziona un tipo di regola dal generatore di regole DQDL, questo verrà aggiunto all'editor delle regole. È quindi possibile specificare parametri, aggiungere e modificare le regole in base alle esigenze modificando il testo. AWS Glue Studio convalida le regole nell'editor delle regole e mostra errori e avvisi, se presenti.

Errori e avvertenze

Se una regola non segue la sintassi delle regole DQDL, l'editor delle regole mostra diversi indicatori visivi che segnalano la presenza di un errore:

- L'editor delle regole mostra un'icona di errore e un colore rosso sulla riga con l'errore.
- Il numero di errori viene mostrato accanto all'icona rossa di errore.
- Quando scegli la riga con l'errore, nella parte inferiore dell'editor delle regole vengono mostrate una descrizione e la sua posizione (riga e colonna).



Operazioni di qualità dei dati

Per impostazione predefinita, questa operazione non è selezionata e l'esecuzione del processo sarà completata anche se le regole di qualità dei dati hanno esito negativo.

Scegli tra le seguenti operazioni. È possibile utilizzare le operazioni per pubblicare i risultati su CloudWatch o interrompere i processi in base a criteri specifici. Le operazioni sono disponibili solo dopo aver creato una regola.

- Pubblica i risultati su CloudWatch: quando esegui un processo, aggiungi i risultati a CloudWatch.
- Processo fallito quando la qualità dei dati fallisce: se le regole sulla qualità dei dati falliscono, anche il processo fallisce di conseguenza.

Output di trasformazione della qualità dei dati

- Dati originali: scegli di emettere i dati di input originali. Questa opzione è ideale se si desidera interrompere il processo quando vengono rilevati problemi di qualità.
- Risultati della qualità dei dati: scegli di visualizzare le regole configurate per l'output e il loro stato di riuscita o non riuscita. Questa opzione è utile se desideri eseguire un'operazione personalizzata.

Impostazioni di output della qualità dei dati

Imposta la posizione dei risultati della qualità dei dati specificando la posizione Amazon S3 come destinazione dell'output della qualità dei dati.

Configurazione del rilevamento delle anomalie e generazione di informazioni

Qualità dei dati AWS Glue (DQ) valuta i dati in base alle regole di qualità pertinenti che scrivi e fornisce informazioni e osservazioni sui dati nel tempo in modo da consentire interventi immediati. DQ, dal momento che esegue la scansione dei dati, calcola parametri statistici, come il numero massimo o minimo di righe, quindi li confronta con le espressioni di soglia.

Alcuni dei vantaggi del rilevamento delle anomalie di Qualità dei dati includono:

- la scansione automatica continua dei dati
- il rilevamento di anomalie che possono indicare un evento imprevisto o un'anomalia statistica
- i suggerimenti di regole per intervenire sulle osservazioni segnalate dal rilevamento di anomalie di Qualità dei dati

Ciò è utile se:

- desideri rilevare automaticamente le anomalie nei dati, senza dover scrivere la qualità di questi ultimi
- desideri profilare i dati e visualizzarne le rappresentazioni visive
- desideri tenere traccia del modo in cui i dati cambiano nel tempo

Quali osservazioni posso visualizzare sui miei dati?

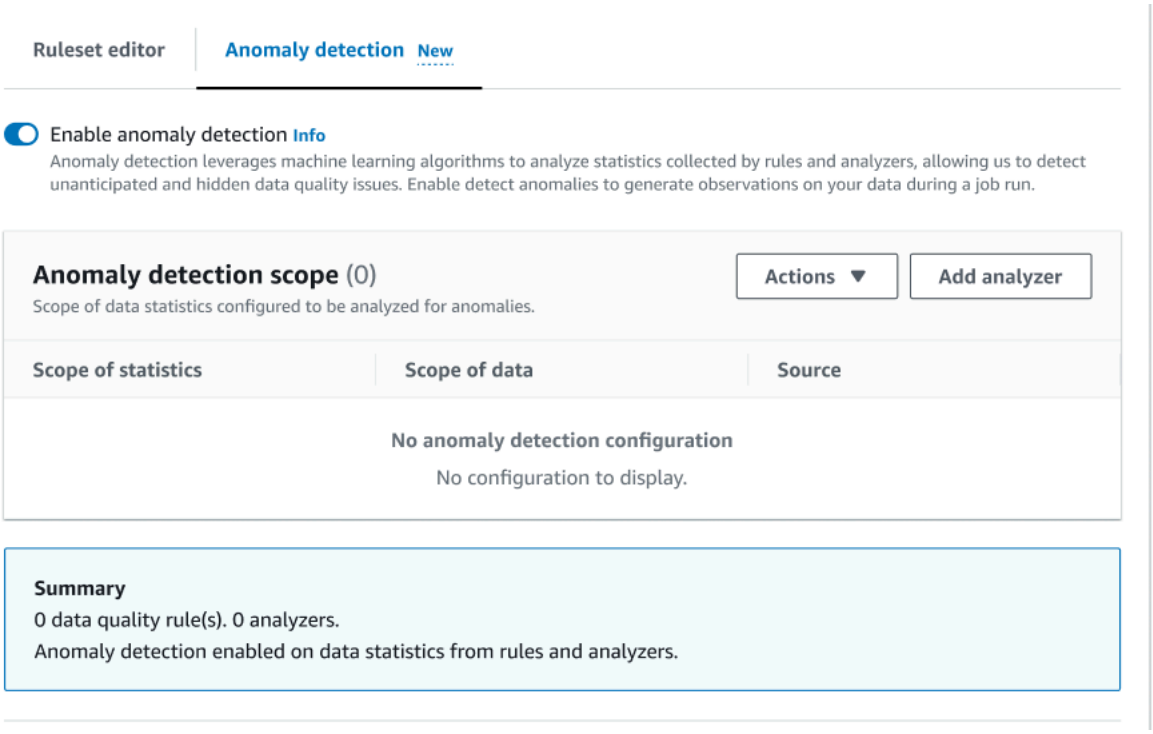
DQ identifica i valori anomali nelle statistiche sui dati raccolte, le modifiche nei formati dei dati, le deviazioni dei dati e le modifiche allo schema. Sulla base delle osservazioni, DQ consiglia regole di qualità dei dati che gli utenti possono rendere operative con facilità. Le statistiche includono completezza, unicità, media, somma StandardDeviation, entropia e. DistinctValuesCount UniqueValueRatio

Abilitazione del rilevamento delle anomalie in AWS Glue Studio

Per abilitare il rilevamento delle anomalie, puoi aprire un processo AWS Glue Studio e attivare "Abilita il rilevamento anomalie". L'attivazione di questa opzione abilita il rilevamento delle anomalie sui dati analizzandoli nel tempo e fornendo statistiche e osservazioni su cui è possibile intervenire.

Per abilitare il rilevamento delle anomalie in AWS Glue Studio:

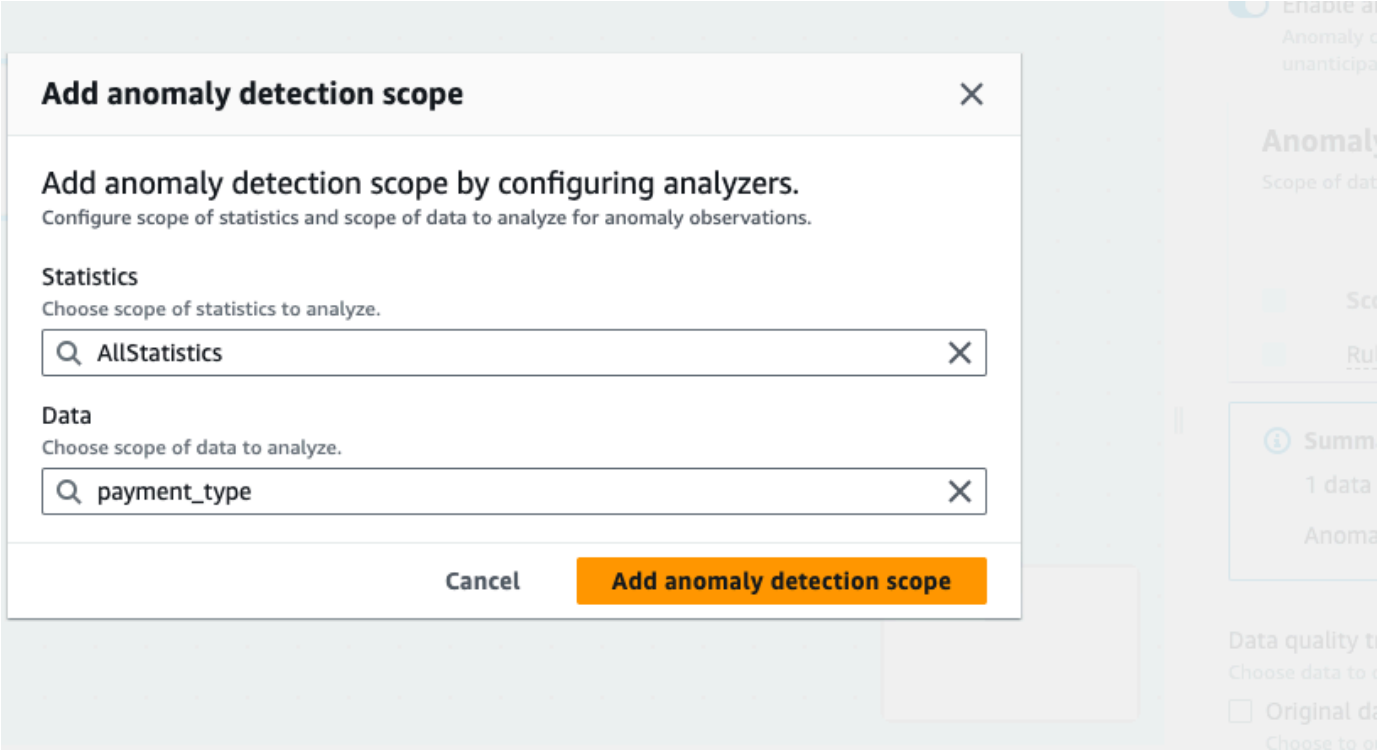
1. Scegli il nodo Qualità dei dati nel processo, quindi scegli la scheda Rilevamento delle anomalie. Attiva "Abilita il rilevamento anomalie".



2. Definisci i dati da monitorare per rilevare eventuali anomalie scegliendo Aggiungi analizzatore. È possibile compilare due campi: Statistiche e Dati.

   Le statistiche corrispondono alle informazioni sulla forma e su altre proprietà dei dati. È possibile scegliere una o più statistiche per volta oppure scegliere Tutte le statistiche. Le statistiche includono: completezza, unicità, media, somma, StandardDeviation entropia e. DistinctValuesCount UniqueValueRatio

   I dati corrispondono alle colonne del set di dati. Puoi scegliere colonne singole oppure sceglierle tutte.



3. Scegli Aggiungi ambito di rilevamento delle anomalie per salvare le modifiche. Dopo aver creato gli analizzatori, puoi visualizzarli nella sezione Ambito di rilevamento delle anomalie.

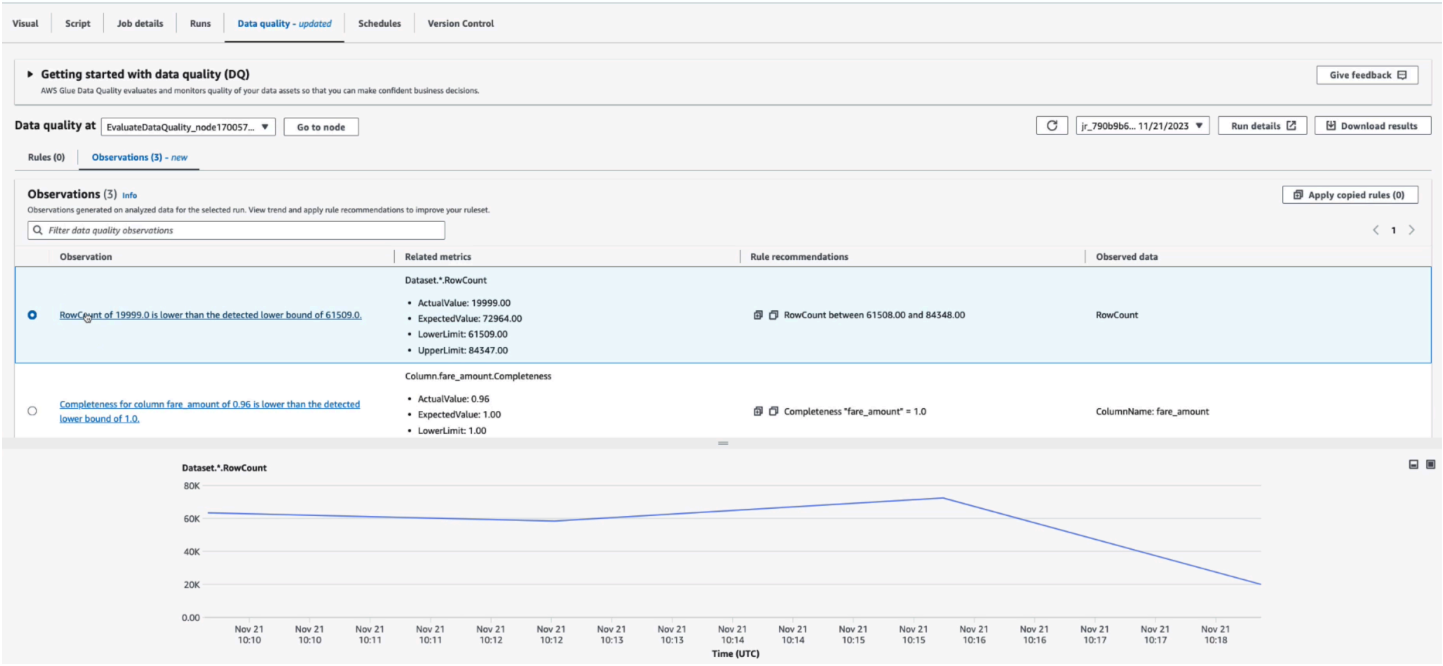
   Puoi anche utilizzare il menu Operazioni per modificare gli analizzatori oppure scegliere la scheda Editor del set di regole e modificare l'analizzatore direttamente nel blocco note dell'editor del set di regole. Visualizzerai gli analizzatori che hai salvato appena sotto le regole che hai creato.

```
Rules = [
]

Analyzers = [
    Completeness “id”
]
```

   Grazie al set di regole aggiornato e agli analizzatori, Qualità dei dati monitora continuamente i dati in entrata, segnalando eventuali anomalie tramite avvisi o interruzioni dei processi (a seconda delle impostazioni).

> **Note**
>
> Le osservazioni vengono generate quando nel set di dati vengono osservati almeno tre valori per ogni statistica sui dati. Se non ci sono osservazioni visibili, significa che Qualità dei dati non dispone di dati sufficienti per generare un'osservazione. Dopo diverse esecuzioni del processo, Qualità dei dati può fornire informazioni sui dati, mostrandole nella sezione Osservazioni.

Gli analizzatori generano osservazioni tramite il rilevamento di anomalie nei dati e forniscono consigli per la creazione progressiva di regole. Puoi visualizzare le osservazioni scegliendo la scheda Qualità dei dati. Le osservazioni sono specifiche per ogni esecuzione del processo. È possibile visualizzare il nodo Qualità dei dati e l'esecuzione del processo specifici nella parte superiore della sezione Osservazioni. Scegli un nuovo nodo o una nuova esecuzione del processo per visualizzarne le osservazioni specifiche.



Osservazione: ogni informazione si basa su un'esecuzione del processo ben precisa e configurata dai set di regole e dagli analizzatori specificati.

Parametri correlati: quando vengono generate le osservazioni, la colonna Parametri correlati mostra la regola e i valori effettivi previsti, nonché i limiti inferiori e superiori.

Suggerimenti di regole: successivamente, AWS Glue consiglia anche le regole per risolvere il problema. Ogni regola suggerita può essere copiata facendo clic sull'icona di copia. È possibile copiare tutte le regole suggerite facendo clic sull'icona di copia accanto a ciascuna di esse, quindi su Applica regole copiate.

Dati monitorati: la colonna Dati monitorati fornisce la colonna o la riga che è stata monitorata e che ha attivato l'osservazione.

Applicazione di una regola suggerita al nodo Qualità dei dati

Dopo la generazione di un'osservazione e il suggerimento di una regola, puoi applicare tale regola al nodo di qualità dei dati. Per farlo:

1. Fai clic sull'icona di copia accanto a ogni suggerimento di regola. In questo modo il suggerimento relativo alla regola verrà aggiunto a un blocco note che potrai recuperare in un secondo momento.
2. Fai clic su Applica i suggerimenti di regole. Si apre così il blocco note in cui è possibile visualizzare le regole copiate in precedenza.
3. Scegli Copia regole.
4. Scegli Applica all'editor del set di regole. Si apre così l'editor del set di regole in cui è possibile incollare le regole copiate.
5. Incolla le regole copiate nell'editor del set di regole.

Qualità dei dati per i processi ETL nei notebook AWS Glue Studio

In questo tutorial, imparerai a utilizzare Qualità dei dati di AWS Glue per i processi di estrazione, trasformazione e caricamento (ETL) nei notebook AWS Glue Studio.

È possibile utilizzare i notebook AWS Glue Studio per modificare gli script di processo e visualizzare l'output senza dover eseguire un processo completo. È inoltre possibile aggiungere markdown e salvare i notebook come file `.ipynb` e script di processo. Nota che è possibile avviare un notebook senza installare software localmente o gestire server. Quando hai completato il lavoro sul codice, potrai utilizzare AWS Glue Studio per convertire facilmente il tuo notebook in un processo AWS Glue.

Il set di dati utilizzato in questo esempio è costituito dai dati del pagamento del fornitore Medicare scaricati da due set di dati Data.CMS.gov: "Inpatient Prospective Payment System Provider Summary for the Top 100 Diagnosis-Related Groups - FY2011" e "Inpatient Charge Data FY 2011".

Dopo aver scaricato i dati, abbiamo apportato delle modifiche al set di dati al fine di introdurre alcuni record errati nella parte finale del file. Questo file modificato si trova in un bucket pubblico Amazon S3 in `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv`.

Prerequisiti

- Ruolo AWS Glue con l'autorizzazione Amazon S3 per scrivere nel bucket Amazon S3 di destinazione
- Un nuovo notebook (consulta la pagina Getting started with notebooks in AWS Glue Studio)

Creazione di un processo ETL in AWS Glue Studio

Creazione di un processo ETL

1. Modifica la versione della sessione in AWS Glue 3.0.

   Per fare ciò, rimuovi tutte le celle di codice boilerplate con il seguente magic ed esegui la cella. Nota che questo codice boilerplate viene fornito automaticamente nella prima cella quando viene creato un nuovo notebook.

   ```
   %glue_version 3.0
   ```

2. Copia il codice seguente e incollalo nella cella.

   ```
   import sys
   from awsglue.transforms import *
   from awsglue.utils import getResolvedOptions
   from pyspark.context import SparkContext
   from awsglue.context import GlueContext
   from awsglue.job import Job

   sc = SparkContext.getOrCreate()
   glueContext = GlueContext(sc)
   spark = glueContext.spark_session
   job = Job(glueContext)
   ```

3. Nella cella successiva, importa la classe `EvaluateDataQuality` che valuta Qualità dei dati di AWS Glue.

```
from awsgluedq.transforms import EvaluateDataQuality
```

4. Nella cella successiva, leggi i dati di origine utilizzando il file .csv archiviato nel bucket pubblico Amazon S3.

```
medicare = spark.read.format(
"csv").option(
"header", "true").option(
"inferSchema", "true").load(
's3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv')
medicare.printSchema()
```

5. Converti i dati in un file AWS Glue DynamicFrame.

```
from awsglue.dynamicframe import DynamicFrame
medicare_dyf = DynamicFrame.fromDF(medicare,glueContext,"medicare_dyf")
```

6. Crea il set di regole utilizzando Data Quality Definition Language (DQDL).

```
EvaluateDataQuality_ruleset = """
    Rules = [
        ColumnExists "Provider Id",
        IsComplete "Provider Id",
        ColumnValues  " Total Discharges " > 15
]
    ]
"""
```

7. Convalida il set di dati rispetto al set di regole.

```
EvaluateDataQualityMultiframe = EvaluateDataQuality().process_rows(
    frame=medicare_dyf,
    ruleset=EvaluateDataQuality_ruleset,
    publishing_options={
        "dataQualityEvaluationContext": "EvaluateDataQualityMultiframe",
        "enableDataQualityCloudWatchMetrics": False,
        "enableDataQualityResultsPublishing": False,
    },
```

```
    additional_options={"performanceTuning.caching": "CACHE_NOTHING"},
)
```

8. Rivedi i risultati.

```
ruleOutcomes = SelectFromCollection.apply(
    dfc=EvaluateDataQualityMultiframe,
    key="ruleOutcomes",
    transformation_ctx="ruleOutcomes",
)

ruleOutcomes.toDF().show(truncate=False)
```

Output:

```
-------------------------------------+-------
+-------------------------------------------------
+-----------------------------------------+
|Rule                                 |Outcome|FailureReason
                  |EvaluatedMetrics                         |
+-------------------------------------+-------
+-------------------------------------------------
+-----------------------------------------+
|ColumnExists "Provider Id"           |Passed |null
                  |{}                                       |
|IsComplete "Provider Id"             |Passed |null
                  |{Column.Provider Id.Completeness -> 1.0}   |
|ColumnValues " Total Discharges " > 15|Failed |Value: 11.0 does not meet the
 constraint requirement!|{Column. Total Discharges .Minimum -> 11.0}|
+-------------------------------------+-------
+-------------------------------------------------
+-----------------------------------------+
```

9. Filtra le righe trasmesse e rivedi le righe con errori tra i risultati a livello di riga di qualità dei dati.

```
owLevelOutcomes = SelectFromCollection.apply(
dfc=EvaluateDataQualityMultiframe,
key="rowLevelOutcomes",
transformation_ctx="rowLevelOutcomes",
```

```
)

rowLevelOutcomes_df = rowLevelOutcomes.toDF() # Convert Glue DynamicFrame to
 SparkSQL DataFrame
rowLevelOutcomes_df_passed =
 rowLevelOutcomes_df.filter(rowLevelOutcomes_df.DataQualityEvaluationResult ==
 "Passed") # Filter only the Passed records.
rowLevelOutcomes_df.filter(rowLevelOutcomes_df.DataQualityEvaluationResult ==
 "Failed").show(5, truncate=False) # Review the Failed records
```

Output:

```
+----------------------------------------+-----------
+-----------------------------------+-------------------------+-------------
+--------------+-----------------+-----------------------------------
+------------------+-------------------------+----------------------
+-------------------------+-------------------------
+----------------------------------------+---------------------------
+---------------------------+
|DRG Definition                            |Provider Id|Provider Name
           |Provider Street Address   |Provider City|Provider State|Provider Zip
 Code|Hospital Referral Region Description| Total Discharges | Average Covered
 Charges | Average Total Payments |Average Medicare Payments|DataQualityRulesPass
      |DataQualityRulesFail                    |DataQualityRulesSkip        |
DataQualityEvaluationResult|
+----------------------------------------+-----------
+-----------------------------------+-------------------------+-------------
+--------------+-----------------+-----------------------------------
+------------------+-------------------------+----------------------
+-------------------------+-------------------------
+----------------------------------------+---------------------------
+---------------------------+
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10005      |MARSHALL MEDICAL CENTER SOUTH
        |2505 U S HIGHWAY 431 NORTH|BOAZ         |AL            |35957
 |AL - Birmingham                     |14               |$15131.85
  |$5787.57               |$4976.71                 |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10046      |RIVERVIEW REGIONAL MEDICAL
 CENTER    |600 SOUTH THIRD STREET    |GADSDEN      |AL            |35901
   |AL - Birmingham                     |14               |$67327.92
   |$5461.57               |$4493.57                 |[IsComplete "Provider Id"]|
```

```
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10083      |SOUTH BALDWIN REGIONAL
 MEDICAL CENTER|1613 NORTH MCKENZIE STREET|FOLEY        |AL            |36535
        |AL - Mobile                          |15               |$25411.33
         |$5282.93               |$4383.73                |[IsComplete "Provider
 Id"]|[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
                   |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|30002      |BANNER GOOD SAMARITAN MEDICAL
 CENTER |1111 EAST MCDOWELL ROAD   |PHOENIX      |AZ            |85006
 |AZ - Phoenix                         |11               |$34803.81
  |$7768.90               |$6951.45                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|30010      |CARONDELET ST  MARYS HOSPITAL
        |1601 WEST ST MARY'S ROAD  |TUCSON       |AZ            |85745
 |AZ - Tucson                          |12               |$35968.50
  |$6506.50               |$5379.83                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
+---------------------------------------+-----------
+-----------------------------------+-------------------------+-------------
+--------------+-----------------+-----------------------------------
+------------------+-------------------------+-----------------------
+-------------------------+-------------------------
+---------------------------------------+---------------------------
+---------------------------+
only showing top 5 rows
```

Nota che AWS Glue Data Quality ha aggiunto quattro nuove colonne (DataQualityRulesPass DataQualityRulesFail DataQualityRulesSkip,, e DataQualityEvaluationResult). Indica i record con esito positivo e quelli con esito negativo, le regole ignorate per la valutazione a livello di riga e i risultati complessivi a livello di riga.

10. Scrivi l'output in un bucket Amazon S3 per analizzare i dati e visualizzare i risultati.

```
#Write the Passed records to the destination.

glueContext.write_dynamic_frame.from_options(
       frame = rowLevelOutcomes_df_passed,
       connection_type = "s3",
```

```
        connection_options = {"path": "s3://glue-sample-target/output-dir/
medicare_parquet"},
        format = "parquet")
```

Riferimento a Data Quality Definition Language (DQDL)

Il Data Quality Definition Language (DQDL) è un linguaggio specifico del dominio utilizzato per definire le regole per AWS Glue Data Quality.

Questa guida introduce i concetti chiave di DQDL per aiutarti a comprendere il linguaggio. Fornisce inoltre un riferimento per i tipi di regole DQDL con sintassi ed esempi. Prima di utilizzare questa guida, ti consigliamo di avere dimestichezza con AWS Glue Data Quality. Per ulteriori informazioni, consulta AWS Glue Qualità dei dati.

> **Note**
>
> DynamicRules sono supportati solo in AWS Glue ETL.

Indice

- ColumnCorrelation
- ColumnCount
- ColumnDataType
- ColumnExists
- ColumnLength
- ColumnNamesMatchPattern
- ColumnValues
- Completezza
- CustomSQL
- DataFreshness
- DatasetMatch
- DistinctValuesCount
- Entropia
- IsComplete
- IsPrimaryKey
- IsUnique
- Media
- ReferentialIntegrity
- RowCount
- RowCountMatch
- StandardDeviation
- Somma
- SchemaMatch
- Univocità
- UniqueValueRatio
- DetectAnomalies

Sintassi di DQDL

Un documento DQDL fa distinzione tra maiuscole e minuscole e contiene un set di regole che raggruppa le singole regole di qualità dei dati. Per costruire un set di regole, è necessario creare un

elenco denominato `Rules` (in maiuscolo), delimitato da una coppia di parentesi quadre. L'elenco deve contenere una o più regole DQDL separate da virgole come nell'esempio seguente.

```
Rules = [
   IsComplete "order-id",
   IsUnique "order-id"
]
```

Struttura delle regole

La struttura di una regola DQDL dipende dal tipo di regola. Tuttavia, le regole DQDL generalmente si adattano al seguente formato.

```
<RuleType> <Parameter> <Parameter> <Expression>
```

`RuleType` è il nome (sensibile al maiuscolo/minuscolo) del tipo di regola che si desidera configurare. Ad esempio, `IsComplete`, `IsUnique` o `CustomSql`. I parametri delle regole sono diversi per ogni tipo di regola. Per un riferimento completo ai tipi di regole DQDL e ai relativi parametri, consulta Documentazione di riferimento del tipo di regola DQDL.

Regole composite

DQDL supporta i seguenti operatori logici che possono essere utilizzati per combinare le regole. Queste regole sono chiamate Regole composite.

e

L'operatore logico `and` restituisce `true` se e solo se le regole che connette sono `true`. Altrimenti, la regola combinata darà come risultato `false`. Ogni regola connessa all'operatore `and` deve essere racchiusa tra parentesi.

L'esempio seguente utilizza l'operatore `and` per combinare due regole DQDL.

```
(IsComplete "id") and (IsUnique "id")
```

oppure

L'operatore logico `or` restituisce `true` se e solo se una o più regole che connette sono `true`. Ogni regola connessa all'operatore `or` deve essere racchiusa tra parentesi.

L'esempio seguente utilizza l'operatore `or` per combinare due regole DQDL.

```
(RowCount "id" > 100) or (IsPrimaryKey "id")
```

È possibile utilizzare lo stesso operatore per connettere più regole, quindi la seguente combinazione di regole è consentita.

```
(Mean "Star_Rating" > 3) and (Mean "Order_Total" > 500) and (IsComplete "Order_Id")
```

Tuttavia, non è possibile combinare gli operatori logici in un'unica espressione. Ad esempio, la combinazione seguente non è consentita.

```
(Mean "Star_Rating" > 3) and (Mean "Order_Total" > 500) or (IsComplete "Order_Id")
```

Come funzionano le regole composite

Per impostazione predefinita, le regole composite vengono valutate come regole individuali nell'intero set di dati o nella tabella e quindi i risultati vengono combinati. In altre parole, valuta prima l'intera colonna e poi applica l'operatore. Questo comportamento predefinito è spiegato di seguito con un esempio:

```
# Dataset

+------+------+
|myCol1|myCol2|
+------+------+
|     2|     1|
|     0|     3|
+------+------+

# Overall outcome

+-------------------------------------------------------------+-------+
|Rule                                                         |Outcome|
+-------------------------------------------------------------+-------+
|(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)|Failed |
+-------------------------------------------------------------+-------+
```

Nell'esempio precedente, valuta AWS Glue Data Quality `(ColumnValues "myCol1" > 1)` innanzitutto cosa comporterà un errore. Quindi valuterà anche `(ColumnValues "myCol2" > 2)` quale fallirà. La combinazione di entrambi i risultati verrà contrassegnata come NON RIUSCITA.

Tuttavia, se si preferisce un comportamento simile a SQL, in cui è necessario valutare l'intera riga, è necessario impostare esplicitamente il `ruleEvaluation.scope` parametro come mostrato `additionalOptions` nel frammento di codice riportato di seguito.

```
object GlueApp {
  val datasource = glueContext.getCatalogSource(
    database="<db>",
    tableName="<table>",
    transformationContext="datasource"
  ).getDynamicFrame()

  val ruleset = """
    Rules = [
        (ColumnValues "age" >= 26) OR (ColumnLength "name" >= 4)
    ]
  """

  val dq_results = EvaluateDataQuality.processRows(
    frame=datasource,
    ruleset=ruleset,
    additionalOptions=JsonOptions("""
        {
          "compositeRuleEvaluation.method":"ROW"
        }
      """
    )
  )
}
```

Una volta impostate, le regole composite si comporteranno come un'unica regola che valuta l'intera riga. L'esempio seguente illustra questo comportamento.

```


# Row Level outcome
```

```
+------+------+-------------------------------------------------------------
+---------------------------+
|myCol1|myCol2|DataQualityRulesPass                                          |
DataQualityEvaluationResult|
+------+------+-------------------------------------------------------------
+---------------------------+
|2     |1     |[(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)]|Passed
               |
|0     |3     |[(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)]|Passed
               |
+------+------+-------------------------------------------------------------
+---------------------------+
```

Alcune regole non possono essere supportate in questa funzionalità perché il loro risultato complessivo si basa su soglie o rapporti. Sono elencate di seguito.

Regole basate sui rapporti:

- Completezza
- DatasetMatch
- ReferentialIntegrity
- Univocità

Regole dipendenti dalle soglie:

Quando le seguenti regole includono una soglia, non sono supportate. Tuttavia, le regole che non lo prevedono `with threshold` rimangono supportate.

- ColumnDataType
- ColumnValues
- CustomSQL

Espressioni

Se un tipo di regola non produce una risposta booleana, è necessario fornire un'espressione come parametro per creare una risposta booleana. Ad esempio, la regola seguente controlla la media di tutti i valori di una colonna rispetto a un'espressione per restituire un risultato vero o falso.

```
Mean "colA" between 80 and 100
```

Alcuni tipi di regole, ad esempio `IsUnique` e `IsComplete`, restituiscono già una risposta booleana.

Nella tabella seguente sono riportate le espressioni che è possibile utilizzare nelle regole DQDL.

Espressioni DQDL supportate

Expression	Descrizione	Esempio
= *x*	Si risolve in `true` se la risposta del tipo di regola è uguale a *x*.	`Completeness "colA" = "1.0", ColumnValues "colA" = "2022-06-30"`
! = *x*	*x Risolve a true se la risposta del tipo di regola non è uguale a x.*	`ColumnValues "colA" != "a", ColumnValues "colA" != "2022-06-30"`
> *x*	Si risolve in `true` se la risposta del tipo di regola è maggiore di *x*.	`ColumnValues "colA" > 10`
< *x*	Si risolve in `true` se la risposta del tipo di regola è minore di *x*.	`ColumnValues "colA" < 1000, ColumnValues "colA" < "2022-06-30"`
>= *x*	Si risolve in `true` se la risposta del tipo di regola è maggiore o uguale a *x*.	`ColumnValues "colA" >= 10`
<= *x*	Si risolve in `true` se la risposta del tipo di regola è minore o uguale a *x*.	`ColumnValues "colA" <= 1000`
tra *x* e *y*	Si risolve in `true` se la risposta del tipo di regola rientra in un intervallo	`Mean "colA" between 8 and 100,`

Expression	Descrizione	Esempio
	specificato (esclusivo). Utilizza questo tipo di espressione solo per i tipi numerici e data.	`ColumnValues "colA" between "2022-05-31" and "2022-06-30"`
non compreso tra x e y	Risolve a true se la risposta del tipo di regola non rientra in un intervallo specifica to (incluso). È necessari o utilizzare questo tipo di espressione solo per i tipi numerici e data.	`ColumnValues "colA" not between "2022-05-31" and "2022-06-30"`
in [*a, b, c, ...*]	Si risolve in `true` se la risposta del tipo di regola è nel set specificato.	`ColumnValues "colA" in [ 1, 2, 3 ], ColumnValues "colA" in [ "a", "b", "c" ]`
non in [*a, b, c,* ...]	Risolve `true` se la risposta del tipo di regola non è inclusa nel set specificato.	`ColumnValues "colA" not in [ 1, 2, 3 ], ColumnValues "colA" not in [ "a", "b", "c" ]`
matches */ab+c/i*	Si risolve in `true` se la risposta del tipo di regola corrisponde a un'espressione regolare.	`ColumnValues "colA" matches "[a-ZA-Z]*"`
non corrisponde a /ab+c/i	Risolve `true` se la risposta del tipo di regola non corrisponde a un'espressione regolare.	`ColumnValues "colA" not matches "[a-ZA-Z]*"`

Expression	Descrizione	Esempio
now()	Funziona solo con il tipo di regola ColumnValues per creare un'espressione di data.	`ColumnValues "load_date" > (now() - 3 days)`
corrisponde/in [...] /non corrisponde/non in [...] with threshold	Specifica la percentua le di valori che corrispon dono alle condizioni della regola. Funziona solo con i tipi di CustomSQL regole ColumnValues ColumnDat aType , e.	`ColumnValues "colA" in ["A", "B"] with threshold > 0.8, ColumnValues "colA" matches "[a-zA-Z]*" with threshold between 0.2 and 0.9 ColumnDataType "colA" = "Timestamp" with threshold > 0.9`

Parole chiave per NULL, EMPTY e WHITESPACES_ONLY

Se vuoi verificare se una colonna di stringhe ha un valore nullo, vuoto o una stringa con solo spazi bianchi, puoi usare le seguenti parole chiave:

- NULL/null: questa parola chiave viene risolta in true per un valore in una colonna di stringhe. null

  `ColumnValues "colA" != NULL with threshold > 0.5`restituirebbe true se più del 50% dei dati non ha valori nulli.

  `(ColumnValues "colA" = NULL) or (ColumnLength "colA" > 5)`restituirebbe true per tutte le righe che hanno un valore nullo o hanno una lunghezza >5. Nota che ciò richiederà l'uso dell'opzione «compositeRuleEvaluation.method» = «ROW».
- EMPTY/ empty: questa parola chiave viene risolta in true per un valore di stringa vuota («») in una colonna di stringhe. Alcuni formati di dati trasformano i valori null in una colonna di stringhe in stringhe vuote. Questa parola chiave aiuta a filtrare le stringhe vuote nei dati.

  `(ColumnValues "colA" = EMPTY) or (ColumnValues "colA" in ["a", "b"])`restituirebbe true se una riga è vuota, «a» o «b». Nota che ciò richiede l'uso dell'opzione «compositeRuleEvaluation.method» = «ROW».

- WHITESPACES_ONLY/whitespaces_only — Questa parola chiave viene risolta in true per una stringa con solo valori di spazi bianchi («») in una colonna di stringhe.

  `ColumnValues "colA" not in ["a", "b", WHITESPACES_ONLY]`restituirebbe true se una riga non è né «a» o «b» né solo spazi bianchi.

  Regole supportate:

  - ColumnValues

Per un'espressione numerica o basata sulla data, se vuoi convalidare se una colonna ha un valore nullo, puoi usare le seguenti parole chiave.

- NULL/null: questa parola chiave viene risolta in true per un valore nullo in una colonna di stringhe.

  `ColumnValues "colA" in [NULL, "2023-01-01"]`restituirebbe true se le date nella colonna sono entrambe o nulle. `2023-01-01`

  `(ColumnValues "colA" = NULL) or (ColumnValues "colA" between 1 and 9)`restituirebbe true per tutte le righe che hanno un valore nullo o hanno valori compresi tra 1 e 9. Nota che ciò richiederà l'uso dell'opzione «compositeRuleEvaluation.method» = «ROW».

  Regole supportate:

  - ColumnValues

Regole dinamiche

Ora puoi creare regole dinamiche per confrontare le metriche correnti prodotte dalle tue regole con i relativi valori storici. Questi confronti storici sono abilitati utilizzando l'operatore `last()` nelle espressioni. Ad esempio, la regola `RowCount > last()` avrà esito positivo se il numero di righe nell'esecuzione corrente è maggiore del conteggio precedente più recente delle righe per lo stesso set di dati. `last()` utilizza un argomento facoltativo relativo ai numeri naturali che descrive il numero di metriche precedenti da prendere in considerazione; `last(k)` dove `k >= 1` farà riferimento alle ultime k metriche.

- Se non sono disponibili punti dati, `last(k)` restituirà il valore predefinito 0,0.
- Se sono disponibili meno di k metriche, `last(k)` restituirà tutte quelle precedenti.

Utilizza `last(k)` per formare espressioni valide, dove `k > 1` richiede una funzione di aggregazione per ridurre più risultati storici a un unico numero. Ad esempio, `RowCount > avg(last(5))` controllerà se il conteggio delle righe del set di dati corrente è strettamente maggiore della media dei conteggi delle ultime cinque righe per lo stesso set di dati. `RowCount > last(5)` produrrà un errore perché il conteggio delle righe del set di dati corrente non può essere confrontato in modo significativo con un elenco.

Funzioni di aggregazione supportate:

- `avg`
- `median`
- `max`
- `min`
- `sum`
- `std` (deviazione standard)
- `abs` (valore assoluto)
- `index(last(k), i)` consentirà di selezionare il `i`° valore più recente tra gli ultimi `k`. `i` è indicizzato a zero, quindi `index(last(3), 0)` restituirà il punto dati più recente e `index(last(3), 3)` genererà un errore poiché ci sono solo tre punti dati, mentre noi cerchiamo di indicizzare il 4° punto dati più recente.

Espressioni di esempio

ColumnCorrelation

- `ColumnCorrelation "colA" "colB" < avg(last(10))`

DistinctValuesCount

- `DistinctValuesCount "colA" between min(last(10))-1 and max(last(10))+1`

La maggior parte dei tipi di regole con condizioni o soglie numeriche supporta regole dinamiche; consulta la tabella fornita, Analizzatori e regole, per determinare se le regole dinamiche sono supportate per il tuo tipo di regola.

Analizzatori

Le regole DQDL utilizzano funzioni chiamate analizzatori per raccogliere informazioni sui dati. Queste informazioni vengono utilizzate dall'espressione booleana di una regola per determinare se quest'ultima deve avere esito positivo o negativo. Ad esempio, la RowCount regola `RowCount > 5` utilizzerà un analizzatore del conteggio delle righe per scoprire il numero di righe nel set di dati e confronterà tale conteggio con l'espressione `> 5` per verificare se esistono più di cinque righe nel set di dati corrente.

A volte, invece di creare regole, consigliamo di creare analizzatori e fare in modo che generino statistiche da utilizzare per rilevare anomalie. In questi casi, puoi creare analizzatori. Gli analizzatori differiscono dalle regole nei modi indicati di seguito.

Caratteristica	Analizzatori	Regolamento
Parte del set di regole	Sì	Sì
Genera statistiche	Sì	Sì
Genera osservazioni	Sì	Sì
Può valutare e verificare una condizione	No	Sì
È possibile configurare operazioni come l'interru zione dei processi in caso di errore o la prosecuzione di un processo di elaborazione	No	Sì

Gli analizzatori possono esistere indipendentemente senza regole, quindi puoi configurarli in modo rapido e creare regole di qualità dei dati in modo progressivo.

Alcuni tipi di regole possono essere inseriti nel blocco `Analyzers` del set di regole per eseguire quelle richieste per gli analizzatori e raccogliere informazioni senza applicare controlli per alcuna condizione. Esistono analizzatori che non sono associati ad alcuna regola e che possono essere inseriti solo nel blocco `Analyzers`. La tabella seguente indica se ogni elemento è supportato come regola o come analizzatore autonomo, insieme a dettagli aggiuntivi per ogni tipo di regola.

Set di regole di esempio con analizzatore

Il seguente set di regole utilizza:

- una regola dinamica per verificare se un set di dati è in crescita rispetto alla media finale delle ultime tre esecuzioni del processo
- un analizzatore `DistinctValuesCount` per registrare il numero di valori distinti nella colonna del `Name` del set di dati
- un analizzatore `ColumnLength` per tracciare le dimensioni minime e massime del `Name` nel tempo

I risultati delle metriche dell'analizzatore per l'esecuzione del processo possono essere visualizzati nella scheda Qualità dei dati.

```
Rules = [
   RowCount > avg(last(3))
],
Analyzers = [
   DistinctValuesCount "Name",
   ColumnLength "Name"
]
```

Commenti

È possibile utilizzare il carattere '#' per aggiungere un commento al documento DQDL. Qualsiasi elemento dopo il carattere '#' e fino alla fine della riga viene ignorato da DQDL.

```
Rules = [
    # More items should generally mean a higher price, so correlation should be
 positive
    ColumnCorrelation "price" "num_items" > 0
]
```

Documentazione di riferimento del tipo di regola DQDL

Questa sezione fornisce un riferimento per ogni tipo di regola supportato da AWS Glue Data Quality.

> **Note**
>
> - Attualmente DQDL non supporta dati di colonna annidati o di tipo elenco.
> - I valori tra parentesi nella tabella seguente verranno sostituiti con le informazioni fornite negli argomenti delle regole.
> - Le regole richiedono in genere un argomento aggiuntivo per l'espressione

RuleType	Descrizione	Argomenti	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
AggregateMatch	Verifica se due set di dati corrispondono confrontando parametri di riepilogo come l'importo totale delle vendite. È utile agli istituti finanziari per	Una o più aggregazioni	Quando i nomi della prima e della seconda colonna di aggregazione corrispondono: `Column.[Column].AggregateMatch` Quando i nomi della	Sì	No	No	No	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
	confrontare se tutti i dati vengono importati dai sistemi di origine.		prima e della seconda colonna di aggregazione non corrispondono: `Column.[Column1,Column2].AggregateMatch`					

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni

AllStatistics	Analizzatore autonomo per raccogliere più metriche per la colonna fornita o per tutte le colonne di un set di dati.	Un nome di colonna singola, OPPURE "AllColumns»	Per le colonne di tutti i tipi:
			`Dataset.RowCou`
			`Column.[Column].Completeness`
			`Column.[Column].Uniqueness`
			Metriche aggiuntive per le colonne con valori stringa:
			`ColumnLength metrics`
				No	Sì	No	No	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
			Metriche aggiuntive per le colonne con valori numerici: `ColumnValues metrics`					
ColumnCorrelation	Verifica il grado di correlazione tra due colonne.	Esattamente due nomi di colonne	`Multicolumn. [Column1], [Column2].ColumnCorrelation`	Sì	Sì	No	Sì	No
ColumnCount	Verifica se delle colonne vengono eliminate.	Nessuno	`Dataset.ColumnCount`	Sì	No	No	Sì	Sì

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
ColumnDataType	Verifica se una colonna è conforme a un tipo di dati.	Esattamente un nome di colonna	`Column.[Column].ColumnDataType.Compliance`	Sì	No	No	Sì, nell'espressione di soglia a livello di riga	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
ColumnExists	Verifica se esistono colonne in un set di dati. Ciò consente ai clienti di creare piattaforme di dati self-service per garantire la disponibilità di determinate colonne.	Esattamente un nome di colonna	N/D	Sì	No	No	No	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni

ColumnLength	Verifica se la lunghezza dei dati è coerente.	Esattamente un nome di colonna	`Column.[Column].MaximumLength`
			
			`Column.[Column].MinimumLength`
			
			Metrica aggiuntiva quando viene fornita la soglia a livello di riga:
			
			`Column.[Column].ColumnValues.Compliance`
				Sì	Sì	Sì, quando viene fornita la soglia a livello di riga	No	Sì. Genera solo osservazioni analizzando la lunghezza minima e quella massima

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
ColumnNaesMatchPttern	Verifica se i nomi delle colonne corrispondono ai modelli definiti. È utile ai team di governance per far rispettare la coerenza dei nomi delle colonne.	Un'espressione regolare per i nomi delle colonne	`Dataset.ColumnNamesPatternMatchRatio`	Sì	No	No	No	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni

ColumnVa ues	Verifica se i dati sono coerenti per valori definiti. Questa regola supporta le espressioni regolari.	Esattamei te un nome di colonna	`Column. [C olumn].I ximum`
			
			`Column. [C olumn].I nimum`
			
			Metrica aggiuntiva quando viene fornita la soglia a livello di riga:
			
			`Column. [C olumn].( lumnVal s.Compl nce`
				Sì	No	Sì, quando viene fornita la soglia a livello di riga	No	Sì. Genera solo osservazioni analizzando i valori minimi e quelli massimi

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
Completez	Verifica la presenza di eventuali spazi vuoti o NULL nei dati.	Esattamer te un nome di colonna	`Column.[Column].Completens`	Sì	Sì	Sì	Sì	Sì

CustomSc	I clienti possono implemen are quasi tutti i tipi di controlli di qualità dei dati in SQL.	Un'istruzione SQL
		
		(Facoltativo) Una soglia a livello di riga
			`Dataset.CustomSL`
			
			Metrica aggiuntiva quando viene fornita la soglia a livello di riga:
			
			`Dataset.CustomSL.Compliance`
				Sì	Sì	Sì, quando viene fornita la soglia a livello di riga	Sì	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
DataFresh ness	Verifica se i dati sono aggiornati.	Esattamer te un nome di colonna	`Column. [C olumn].I taFreshı ss.Comp: ance`	Sì	No	Sì	No	No
DatasetM tch	Confronta due set di dati e identifica se sono sincronizzati.	Nome di un set di dati di riferimento Una mappatura delle colonne (Facoltativo) Colonne da controllare per cercare corrispondenze	`Dataset [ Referen Dataset/ ias].Da setMatcl`	Sì	No	Sì	Sì	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
DistinctValuesCount	Verifica la presenza di valori duplicati.	Esattamente un nome di colonna	`Column.[Column].DistinctValuesCount`	Sì	Sì	Sì	Sì	Sì
DetectAnomalies	Verifica la presenza di anomalie nelle metriche riportate di un altro tipo di regola.	Un tipo di regola	Metriche riportate dall'argomento del tipo di regola	Sì	No	No	No	No
Entropia	Verifica l'entropia dei dati.	Esattamente un nome di colonna	`Column.[Column].Entropy`	Sì	Sì	No	Sì	No
IsComplete	Verifica se il 100% dei dati è completo.	Esattamente un nome di colonna	`Column.[Column].Completeness`	Sì	No	Sì	No	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
IsPrimaryKey	Verifica se una colonna è una chiave primaria (non NULL e univoca).	Esattamente un nome di colonna	Per colonna singola: `Column.[Column].Uniqueness` Per più colonne: `Multicolumn[Commaelimited Columns] Uniqueness`	Sì	No	Sì	No	No
IsUnique	Verifica se il 100% dei dati è univoco.	Esattamente un nome di colonna	`Column.[Column].Uniqueness`	Sì	No	Sì	No	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
Media	Verifica se la media corrisponde alla soglia impostata.	Esattamente un nome di colonna	`Column.[Column].Mean`	Sì	Sì	Sì	Sì	No
ReferentialIntegrity	Verifica se due set di dati hanno un'integrità referenziale.	Uno o più nomi di colonne dal set di dati Uno o più nomi di colonne dal set di dati di riferimento	`Column.[Reference].DatasetAlias].ReferentialIntegrity`	Sì	No	Sì	Sì	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
RowCoun	Verifica se il conteggio dei record corrisponde a una soglia.	Nessuno	`Dataset.RowCou`	Sì	Sì	No	Sì	Sì
RowCoun atch	Verifica se il conteggio dei record tra due set di dati corrisponde.	Alias del set di dati di riferimento	`Dataset[Referen Dataset/ ias].Ro ountMat`	Sì	No	No	Sì	No
StandardD eviation	Verifica se la deviazione standard corrisponde alla soglia.	Esattame te un nome di colonna	`Column.[C olumn].S andardDe iation`	Sì	Sì	Sì	Sì	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
SchemaM ch	Verifica se il numero di record tra due set di dati corrisponde.	Alias del set di dati di riferimento	`Dataset[ Referen Dataset/ ias].Sc maMatch`	Sì	No	No	Sì	No
Somma	Verifica se la somma corrisponde a una soglia impostata.	Esattame te un nome di colonna	`Column. [C olumn].S m`	Sì	Sì	No	Sì	No
Univocità	Verifica se l'unicità del set di dati corrisponde alla soglia.	Esattame te un nome di colonna	`Column. [C olumn].U iquenes`	Sì	Sì	Sì	Sì	No

RuleType	Descrizione	Argoment	Metriche riportate	Supportato come regola?	Supportato come analizzatore?	Restituisce risultati a livello di riga?	Supporto per regole dinamiche?	Genera osservazioni
UniqueValueRatio	Verifica se la porzione di valore univoco corrisponde alla soglia.	Esattamente un nome di colonna	`Column.[Column].UniqueValueRatio`	Sì	Sì	Sì	Sì	No

Argomenti

- AggregateMatch
- ColumnCorrelation
- ColumnCount
- ColumnDataType
- ColumnExists
- ColumnLength
- ColumnNamesMatchPattern
- ColumnValues
- Completezza
- CustomSQL
- DataFreshness
- DatasetMatch
- DistinctValuesCount
- Entropia
- IsComplete
- IsPrimaryKey

- IsUnique
- Media
- ReferentialIntegrity
- RowCount
- RowCountMatch
- StandardDeviation
- Somma
- SchemaMatch
- Univocità
- UniqueValueRatio
- DetectAnomalies

AggregateMatch

Verifica il rapporto di aggregazioni a due colonne rispetto a una determinata espressione. Questo tipo di regola funziona su più set di dati. Le aggregazioni a due colonne vengono valutate e viene prodotto un rapporto dividendo il risultato dell'aggregazione della prima colonna per il risultato dell'aggregazione della seconda colonna. Il rapporto viene confrontato con l'espressione fornita per produrre una risposta booleana.

Sintassi

Aggregazione di colonne

```
ColumnExists <AGG_OPERATION> (<OPTIONAL_REFERENCE_ALIAS>.<COL_NAME>)
```

- AGG_OPERATION: l'operazione da utilizzare per l'aggregazione. Attualmente, sono supportati `sum` e `avg`.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- OPTIONAL_REFERENCE_ALIAS: è necessario fornire questo parametro se la colonna proviene da un set di dati di riferimento e non dal set di dati primario. <database_name>Se utilizzi questa regola nel AWS Glue Data Catalog, il tuo alias di riferimento deve seguire il formato ". <table_name>. <column_name>

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short

- COL_NAME: il nome della colonna da aggregare.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short

Esempio: media

```
"avg(rating)"
```

Esempio: somma

```
"sum(amount)"
```

Esempio: media della colonna nel set di dati di riferimento

```
"avg(reference.rating)"
```

Regola

```
AggregateMatch <AGG_EXP_1> <AGG_EXP_2> <EXPRESSION>
```

- AGG_EXP_1: l'aggregazione della prima colonna.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- AGG_EXP_2: l'aggregazione della seconda colonna.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: aggrega corrispondenza utilizzando la somma

La seguente regola di esempio verifica se la somma dei valori nella colonna `amount` è esattamente uguale alla somma dei valori nella colonna `total_amount`.

```
AggregateMatch "sum(amount)" "sum(total_amount)" = 1.0
```

Esempio: aggrega corrispondenza utilizzando la media

La seguente regola di esempio verifica se la media dei valori nella colonna `ratings` è pari almeno al 90% della media dei valori nella colonna `ratings` del set di dati `reference`. Il set di dati di riferimento viene fornito come origine dati aggiuntiva nell'esperienza ETL o Catalogo dati.

In AWS Glue ETL, puoi usare:

```
AggregateMatch "avg(ratings)" "avg(reference.ratings)" >= 0.9
```

Nel AWS Glue Data Catalog, puoi usare:

```
AggregateMatch "avg(ratings)" "avg(database_name.tablename.ratings)" >= 0.9
```

Comportamento nullo

La `AggregateMatch` regola ignorerà le righe con valori NULL nel calcolo dei metodi di aggregazione (somma/media). Per esempio:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

La media della colonna `units` sarà (0 + 20 + 40)/3 = 20. Le righe 101 e 103 non vengono considerate in questo calcolo.

ColumnCorrelation

Verifica la correlazione tra due colonne rispetto a una determinata espressione. AWS Glue Data Quality utilizza il coefficiente di correlazione di Pearson per misurare la correlazione lineare tra due colonne. Il risultato è un numero compreso tra -1 e 1 che misura la forza e la direzione della relazione.

Sintassi

```
ColumnCorrelation <COL_1_NAME> <COL_2_NAME> <EXPRESSION>
```

- COL_1_NAME: il nome della prima colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- COL_2_NAME: il nome della seconda colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: correlazione tra colonne

La seguente regola di esempio verifica se il coefficiente di correlazione tra le colonne `height` e `weight` ha una forte correlazione positiva (un valore del coefficiente maggiore di 0,8).

```
ColumnCorrelation "height" "weight" > 0.8
```

Regole dinamiche di esempio

- `ColumnCorrelation "colA" "colB" between min(last(10)) and max(last(10))`
- `ColumnCorrelation "colA" "colB" < avg(last(5)) + std(last(5))`

Comportamento nullo

La `ColumnCorrelation` regola ignorerà le righe con `NULL` valori nel calcolo della correlazione. Per esempio:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
```

```
|103|null       |
|104|40         |
+---+-----------+
```

Le righe 101 e 103 verranno ignorate e `ColumnCorrelation` saranno 1,0.

ColumnCount

Verifica il conteggio delle righe del set di dati primario rispetto a una determinata espressione. Nell'espressione, è possibile specificare il numero di colonne o un intervallo di colonne utilizzando operatori come > e <.

Sintassi

```
ColumnCount <EXPRESSION>
```

- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: controllo numerico del conteggio delle colonne

La seguente regola di esempio controlla se il conteggio delle colonne rientra in un determinato intervallo.

```
ColumnCount between 10 and 20
```

Regole dinamiche di esempio

- `ColumnCount >= avg(last(10))`
- `ColumnCount between min(last(10))-1 and max(last(10))+1`

ColumnDataType

Controlla il tipo di dati intrinseco dei valori in una determinata colonna rispetto al tipo previsto fornito. Accetta un'espressione `with threshold` per verificare la presenza di un sottoinsieme di valori nella colonna.

Sintassi

```
ColumnDataType <COL_NAME> = <EXPECTED_TYPE>
   ColumnDataType <COL_NAME> = <EXPECTED_TYPE> with threshold <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonne supportati: tipo stringa

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPECTED_TYPE: il tipo di valori previsto nella colonna.

  Valori supportati: Boolean, Date, Timestamp, Integer, Double, Float, Long

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION: un'espressione facoltativa per specificare la percentuale di valori che devono essere del tipo previsto.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short

Esempio: il tipo di dato nella colonna rappresentato come stringa è intero

La seguente regola di esempio verifica se i valori nella colonna specificata, che è di tipo stringa, sono effettivamente numeri interi.

```
ColumnDataType "colA" = "INTEGER"
```

Esempio: verifica di un sottoinsieme dei valori delle colonne, di tipo intero ma rappresentati come stringhe

La seguente regola di esempio verifica se più del 90% dei valori nella colonna data, che è di tipo stringa, sono effettivamente numeri interi.

```
ColumnDataType "colA" = "INTEGER" with threshold > 0.9
```

ColumnExists

Verifica se esiste una colonna.

Sintassi

```
ColumnExists <COL_NAME>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna

Esempio: la colonna esiste

La seguente regola di esempio verifica se la colonna denominata `Middle_Name` esiste.

```
ColumnExists "Middle_Name"
```

ColumnLength

Verifica se la lunghezza di ogni riga in una colonna è conforme a una determinata espressione.

Sintassi

```
ColumnLength <COL_NAME><EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonne supportati: String
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: lunghezza della riga della colonna

La seguente regola di esempio verifica se il valore in ogni riga della colonna denominata `Postal_Code` è lungo 5 caratteri.

```
ColumnLength "Postal_Code" = 5
```

Comportamento nullo

La `ColumnLength` regola considera `NULL` s come stringhe di lunghezza pari a 0. Per una `NULL` riga:

```
ColumnLength "Postal_Code" > 4 # this will fail
```

```
ColumnLength "Postal_Code" < 6 # this will succeed
```

Il seguente esempio di regola composta fornisce un modo per fallire in modo esplicito NULL i valori:

```
(ColumnLength "Postal_Code" > 4) AND (ColumnValues != NULL)
```

ColumnNamesMatchPattern

Verifica se i nomi di tutte le colonne del set di dati primario corrispondono all'espressione regolare specificata.

Sintassi

```
ColumnNamesMatchPattern <PATTERN>
```

- PATTERN: il modello in base al quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short

Esempio: i nomi delle colonne corrispondono al modello

La seguente regola di esempio verifica se tutte le colonne iniziano con il prefisso "aws_"

```
ColumnNamesMatchPattern "aws_.*"
```

ColumnValues

Esegue un'espressione rispetto ai valori di una colonna.

Sintassi

```
ColumnValues <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna

- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: valori consentiti

La seguente regola di esempio verifica se ogni valore nella colonna specificata si trova in un insieme di valori consentiti (inclusi null, empty e stringhe con solo spazi bianchi).

```
ColumnValues "Country" in [ "US", "CA", "UK", NULL, EMPTY, WHITESPACES_ONLY ]
```

Esempio: espressione regolare

La seguente regola di esempio verifica i valori di una colonna rispetto a un'espressione regolare.

```
ColumnValues "First_Name" matches "[a-ZA-Z]*"
```

Esempio: valori data

La seguente regola di esempio verifica i valori di una colonna data rispetto a un'espressione data.

```
ColumnValues "Load_Date" > (now() - 3 days)
```

Esempio: valori numerici

La seguente regola di esempio verifica se i valori delle colonne corrispondono a un determinato vincolo numerico.

```
ColumnValues "Customer_ID" between 1 and 2000
```

Comportamento nullo

Per tutte le `ColumnValues` regole (diverse da `!=` e`NOT IN`), `NULL` le righe non soddisferanno la regola. Se la regola fallisce a causa di un valore nullo, il motivo dell'errore sarà il seguente:

```
Value: NULL does not meet the constraint requirement!
```

Il seguente esempio di regola composta fornisce un modo per `NULL` consentire esplicitamente i valori:

```
(ColumnValues "Age" > 21) OR (ColumnValues "Age" = NULL)
```

ColumnValues Le regole negate che utilizzano la `not in` sintassi `!=` and verranno valide per le righe. NULL Per esempio:

```
ColumnValues "Age" != 21
```

```
ColumnValues "Age" not in [21, 22, 23]
```

Gli esempi seguenti forniscono un modo per fallire in modo esplicito i valori NULL

```
(ColumnValues "Age" != 21) AND (ColumnValues "Age" != NULL)
```

```
ColumnValues "Age" not in [21, 22, 23, NULL]
```

Completezza

Verifica la percentuale di valori completi (non nulli) in una colonna rispetto a una determinata espressione.

Sintassi

```
Completeness <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: percentuale di valore nullo

Le seguenti regole di esempio controllano se più del 95% dei valori in una colonna sono completi.

```
Completeness "First_Name" > 0.95
```

Regole dinamiche di esempio

- `Completeness "colA" between min(last(5)) - 1 and max(last(5)) + 1`
- `Completeness "colA" <= avg(last(10))`

Comportamento nullo

Nota sui formati di dati CSV: le righe vuote nelle colonne CSV possono mostrare più comportamenti.

- Se una colonna è di `String` tipo, la riga vuota verrà riconosciuta come stringa vuota e non violerà la regola. `Completeness`
- Se una colonna è di un altro tipo di dati`Int`, la riga vuota verrà riconosciuta come tale `NULL` e non rispetterà la `Completeness` regola.

CustomSQL

Questo tipo di regola è stato esteso per supportare due casi d'uso:

- Esegui un'istruzione SQL personalizzata su un set di dati e controlla il valore restituito rispetto a una determinata espressione.
- Esegui un'istruzione SQL personalizzata specificando un nome di colonna nell'istruzione SELECT in base alla quale eseguire un confronto con alcune condizioni per ottenere risultati a livello di riga.

Sintassi

```
CustomSql <SQL_STATEMENT> <EXPRESSION>
```

- SQL_STATEMENT: un'istruzione SQL che restituisce un singolo valore numerico, racchiuso tra virgolette doppie.
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: SQL personalizzato per recuperare il risultato di una regola generale

Questa regola di esempio utilizza un'istruzione SQL per recuperare il numero di record per un set di dati. La regola verifica quindi che il conteggio dei record sia compreso tra 10 e 20.

```
CustomSql "select count(*) from primary" between 10 and 20
```

Esempio: SQL personalizzato per recuperare i risultati a livello di riga

Questa regola di esempio utilizza un'istruzione SQL personalizzata specificando un nome di colonna nell'istruzione SELECT in base alla quale eseguire un confronto con alcune condizioni per ottenere risultati a livello di riga. Un'espressione di condizione di soglia definisce una soglia di quanti record devono avere esito negativo perché l'intera regola abbia esito negativo. Tieni presente che una regola non può contenere contemporaneamente una condizione e una parola chiave.

```
CustomSql "select Name from primary where Age  > 18"
```

oppure

```
CustomSql "select Name from primary where Age > 18" with threshold  > 3
```

> **Important**
>
> L'alias `primary` sostituisce il nome del set di dati che si desidera valutare. Quando lavori con job ETL visivi sulla console, `primary` rappresenta sempre il `DynamicFrame` passato alla trasformazione `EvaluateDataQuality.apply()`. Quando si utilizza il AWS Glue Data Catalog per eseguire attività di qualità dei dati su una tabella, `primary` rappresenta la tabella.

Se utilizzi Catalogo dati AWS Glue, puoi anche utilizzare i nomi effettivi delle tabelle:

```
CustomSql "select count(*) from database.table" between 10 and 20
```

È inoltre possibile effettuare il join di più tabelle per confrontare diversi elementi di dati:

```
CustomSql "select count(*) from database.table inner join database.table2 on id1 = id2"
 between 10 and 20
```

In ETL di AWS Glue, CustomSQL è in grado di identificare i record che non hanno superato i controlli di qualità dei dati. Affinché ciò funzioni, è necessario restituire i record che fanno parte della tabella

primaria della quale si sta valutando la qualità dei dati. I record restituiti come parte della query sono considerati riusciti, mentre i record che non vengono restituiti sono considerati non riusciti.

La regola seguente garantirà che i record con età < 100 vengano identificati come riusciti e i record al di sopra vengano contrassegnati come non riusciti.

```
CustomSql "select id from primary where age < 100"
```

Questa regola CustomSQL sarà soddisfatta quando il 50% dei record hanno un'età > 10 e identificherà anche i record che non soddisfano la regola. I record restituiti da questa regola CustomSQL verranno considerati superati, mentre quelli non restituiti verranno considerati non superati.

```
CustomSQL "select ID, CustomerID from primary where age > 10" with threshold > 0.5
```

Nota: la regola CustomSQL avrà esito negativo se si restituiscono record che non sono disponibili nel set di dati.

DataFreshness

Verifica l'aggiornamento dei dati in una colonna valutando la differenza tra l'ora corrente e i valori di una colonna di date. È possibile specificare un'espressione basata sul tempo per questo tipo di regola per assicurarsi che i valori delle colonne siano aggiornati.

Sintassi

```
DataFreshness <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonne supportati: Data
- EXPRESSION: un'espressione numerica in ore o giorni. È necessario specificare l'unità di tempo nell'espressione.

Esempio: freschezza dei dati

Le seguenti regole di esempio controllano la freschezza dei dati, ovvero quanto sono aggiornati.

```
DataFreshness "Order_Date" <= 24 hours
DataFreshness "Order_Date" between 2 days and 5 days
```

Comportamento nullo

Le `DataFreshness` regole avranno esito negativo per le righe con `NULL` valori. Se la regola fallisce a causa di un valore nullo, il motivo dell'errore sarà il seguente:

```
80.00 % of rows passed the threshold
```

dove il 20% delle righe che hanno avuto esito negativo include le righe con`NULL`.

La seguente regola composta di esempio fornisce un modo per consentire esplicitamente `NULL` i valori:

```
(DataFreshness "Order_Date" <= 24 hours) OR (ColumnValues "Order_Date" = NULL)
```

DatasetMatch

Verifica se i dati nel set di dati primario corrispondono ai dati in un set di dati di riferimento. Il join dei due set di dati viene effettuato utilizzando le mappature delle colonne chiave fornite. È possibile fornire mappature di colonne aggiuntive se si desidera verificare l'uguaglianza dei dati solo in quelle colonne. Nota che, DataSetMatchper funzionare, le tue chiavi di join devono essere uniche e non devono essere NULL (deve essere una chiave primaria). Se non soddisfi queste condizioni, riceverai il messaggio di errore "Provided key map not suitable for given data frames". Nei casi in cui non è possibile avere chiavi di unione univoche, prendi in considerazione l'utilizzo di altri tipi di regole, ad esempio la corrispondenza nei dati AggregateMatchdi riepilogo.

Sintassi

```
DatasetMatch <REFERENCE_DATASET_ALIAS> <JOIN CONDITION WITH
 MAPPING> <OPTIONAL_MATCH_COLUMN_MAPPINGS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS: l'alias del set di dati di riferimento con cui confronti i dati del set di dati primario.
- KEY_COLUMN_MAPPINGS: un elenco separato da virgole di nomi di colonne che formano una chiave nei set di dati. Se i nomi delle colonne non sono uguali in entrambi i set di dati, è necessario separarli con un `->`

- OPTIONAL_MATCH_COLUMN_MAPPINGS: puoi fornire questo parametro se desideri verificare la corrispondenza dei dati solo in determinate colonne. Utilizza la stessa sintassi delle mappature delle colonne chiave. Se questo parametro non viene fornito, abbineremo i dati in tutte le colonne rimanenti. Le colonne rimanenti (non chiave) devono avere gli stessi nomi in entrambi i set di dati.
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: abbina i set di dati definiti utilizzando la colonna ID

La seguente regola di esempio verifica che più del 90% del set di dati primario corrisponda al set di dati di riferimento, utilizzando la colonna "ID" per il join dei due set di dati. In questo caso confronta tutte le colonne.

```
DatasetMatch "reference" "ID" >= 0.9
```

Esempio: abbina i set di dati del set utilizzando più colonne chiave

Nell'esempio seguente, il set di dati primario e il set di dati di riferimento hanno nomi diversi per le colonne chiave. `ID_1` e `ID_2` insieme formano una chiave composita nel set di dati primario. `ID_ref1` e `ID_ref2` insieme formano una chiave composita nel set di dati di riferimento. In questo scenario, è possibile utilizzare la sintassi speciale per fornire i nomi delle colonne.

```
DatasetMatch "reference" "ID_1->ID_ref1,ID_ref2->ID_ref2" >= 0.9
```

Esempio: abbina i set di dati del set utilizzando più colonne chiave e verifica che le colonne specifiche corrispondano

Questo esempio si basa sull'esempio precedente. Vogliamo verificare che solo la colonna contenente gli importi corrisponda. Questa colonna è denominata `Amount1` nel set di dati primario e `Amount2` nel set di dati di riferimento. Vuoi una corrispondenza esatta.

```
DatasetMatch "reference" "ID_1->ID_ref1,ID_ref2->ID_ref2" "Amount1->Amount2" >= 0.9
```

DistinctValuesCount

Seleziona il numero di valori distinti in una colonna rispetto a una determinata espressione.

Sintassi

```
DistinctValuesCount <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: conteggio dei valori distinti delle colonne

La seguente regola di esempio verifica che la colonna denominata `State` contenga più di 3 valori distinti.

```
DistinctValuesCount "State" > 3
```

Regole dinamiche di esempio

- `DistinctValuesCount "colA" between avg(last(10))-1 and avg(last(10))+1`
- `DistinctValuesCount "colA" <= index(last(10),2) + std(last(5))`

Entropia

Verifica se il valore di entropy di una colonna corrisponde a una determinata espressione. L'entropia misura il livello di informazioni contenute in un messaggio. Data la distribuzione della probabilità sui valori in una colonna, l'entropia descrive quanti bit sono necessari per identificare un valore.

Sintassi

```
Entropy <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: entropia delle colonne

La seguente regola di esempio verifica che la colonna denominata `Feedback` abbia un valore di entropia maggiore di uno.

```
Entropy "Star_Rating" > 1
```

Regole dinamiche di esempio

- `Entropy "colA" < max(last(10))`
- `Entropy "colA" between min(last(10)) and max(last(10))`

IsComplete

Verifica se tutti i valori in una colonna sono completi (non nulli).

Sintassi

```
IsComplete <COL_NAME>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna

Esempio: valori nulli

L'esempio seguente verifica se tutti i valori in una colonna denominata `email` non sono nulli.

```
IsComplete "email"
```

Comportamento nullo

Nota sui formati di dati CSV: le righe vuote nelle colonne CSV possono mostrare più comportamenti.

- Se una colonna è di `String` tipo, la riga vuota verrà riconosciuta come stringa vuota e non violerà la regola. `Completeness`
- Se una colonna è di un altro tipo di dati`Int`, la riga vuota verrà riconosciuta come tale `NULL` e non rispetterà la `Completeness` regola.

IsPrimaryKey

Verifica se una colonna contiene una chiave primaria. Una colonna contiene una chiave primaria se tutti i valori nella colonna sono univoci e completi (non nulli).

Sintassi

```
IsPrimaryKey <COL_NAME>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna

Esempio: chiave primaria

La seguente regola di esempio verifica se la colonna denominata `Customer_ID` contiene una chiave primaria.

```
IsPrimaryKey "Customer_ID"
```

Esempio: chiave primaria con più colonne. Ognuno degli esempi seguenti è valido.

```
IsPrimaryKey "colA" "colB"
IsPrimaryKey "colA" "colB" "colC"
IsPrimaryKey colA "colB" "colC"
```

IsUnique

Verifica se tutti i valori in una colonna sono univoci e restituisce un valore booleano.

Sintassi

```
IsUnique <COL_NAME>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

Tipi di colonna supportati: qualsiasi tipo di colonna

Esempio: valori di colonna univoci

La seguente regola di esempio verifica se tutti i valori in una colonna denominata `email` sono univoci.

```
IsUnique "email"
```

Media

Verifica se la media di tutti i valori in una colonna corrisponde a una determinata espressione.

Sintassi

```
Mean <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: valore medio

La seguente regola di esempio verifica se la media di tutti i valori di una colonna supera una soglia.

```
Mean "Star_Rating" > 3
```

Regole dinamiche di esempio

- `Mean "colA" > avg(last(10)) + std(last(2))`
- `Mean "colA" between min(last(5)) - 1 and max(last(5)) + 1`

Comportamento nullo

La `Mean` regola ignorerà le righe con `NULL` valori nel calcolo della media. Per esempio:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

La media della colonna `units` sarà (0 + 20 + 40)/3 = 20. Le righe 101 e 103 non vengono considerate in questo calcolo.

ReferentialIntegrity

Verifica in che misura i valori di un set di colonne nel set di dati primario siano un sottoinsieme dei valori di un set di colonne in un set di dati di riferimento.

Sintassi

```
ReferentialIntegrity <PRIMARY_COLS> <REFERENCE_DATASET_COLS> <EXPRESSION>
```

- PRIMARY_COLS: un elenco separato da virgole dei nomi delle colonne nel set di dati primario.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- REFERENCE_DATASET_COLS: questo parametro contiene due parti separate da un punto. La prima parte è l'alias del set di dati di riferimento. La seconda parte è l'elenco separato da virgole dei nomi delle colonne nel set di dati di riferimento racchiuso tra parentesi.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: verifica l'integrità referenziale di una colonna di codice postale

La seguente regola di esempio verifica che più del 90% dei valori nella colonna `zipcode` del set di dati primario siano presenti nella colonna `zipcode` del set di dati `reference`.

```
ReferentialIntegrity "zipcode" "reference.zipcode" >= 0.9
```

Esempio: verifica l'integrità referenziale delle colonne relative alla città e allo stato

Nell'esempio seguente, nel set di dati primario e nel set di dati di riferimento sono presenti colonne contenenti informazioni sulla città e sullo stato. I nomi delle colonne sono diversi in entrambi i set di dati. La regola verifica se il set di valori delle colonne nel set di dati primario è esattamente uguale al set di valori delle colonne nel set di dati di riferimento.

```
ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" = 1.0
```

Regole dinamiche di esempio

- `ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" > avg(last(10))`
- `ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" between min(last(10)) - 1 and max(last(10)) + 1`

RowCount

Verifica il numero di righe di un set di dati rispetto a una determinata espressione. Nell'espressione, è possibile specificare il numero di righe o un intervallo di righe utilizzando operatori come > e <.

Sintassi

```
RowCount <EXPRESSION>
```

- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: controllo numerico del conteggio delle righe

La seguente regola di esempio controlla se il conteggio delle righe rientra in un determinato intervallo.

```
RowCount between 10 and 100
```

Regole dinamiche di esempio

```
RowCount > avg(lats(10)) *0.8
```

RowCountMatch

Verifica il rapporto tra il conteggio delle righe del set di dati primario e il conteggio delle righe di un set di dati di riferimento rispetto all'espressione data.

Sintassi

```
RowCountMatch <REFERENCE_DATASET_ALIAS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS: l'alias del set di dati di riferimento con cui confrontare il conteggio delle righe.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: controllo del conteggio delle righe rispetto a un set di dati di riferimento

La seguente regola di esempio verifica se il conteggio delle righe del set di dati primario è almeno il 90% del conteggio delle righe del set di dati di riferimento.

```
RowCountMatch "reference" >= 0.9
```

StandardDeviation

Verifica la deviazione standard di tutti i valori di una colonna rispetto a una determinata espressione.

Sintassi

```
StandardDeviation <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: deviazione standard

La seguente regola di esempio verifica se la deviazione standard dei valori in una colonna denominata `colA` è inferiore a un valore specificato.

```
StandardDeviation "Star_Rating" < 1.5
```

Regole dinamiche di esempio

- `StandardDeviation "colA" > avg(last(10) + 0.1`
- `StandardDeviation "colA" between min(last(10)) - 1 and max(last(10)) + 1`

Comportamento nullo

La `StandardDeviation` regola ignorerà le righe con `NULL` valori nel calcolo della deviazione standard. Per esempio:

```
+---+-----------+-----------+
|id |units1     |units2     |
+---+-----------+-----------+
|100|0          |0          |
|101|null       |0          |
|102|20         |20         |
|103|null       |0          |
|104|40         |40         |
+---+-----------+-----------+
```

La deviazione standard della colonna non `units1` considererà le righe 101 e 103 e risulterà pari a 16,33. La deviazione standard per la colonna `units2` risulterà pari a 16.

Somma

Verifica la somma di tutti i valori in una colonna rispetto a una determinata espressione.

Sintassi

```
Sum <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short

- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: somma

La seguente regola di esempio verifica se la somma di tutti i valori in una colonna supera una determinata soglia.

```
Sum "transaction_total" > 500000
```

Regole dinamiche di esempio

- `Sum "ColA" > avg(last(10))`
- `Sum "colA" between min(last(10)) - 1 and max(last(10)) + 1`

Comportamento nullo

La `Sum` regola ignorerà le righe con `NULL` valori nel calcolo della somma. Per esempio:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

La somma della colonna non `units` prenderà in considerazione le righe 101 e 103 e darà come risultato (0 + 20 + 40) = 60.

SchemaMatch

Verifica se lo schema del set di dati primario corrisponde allo schema del set di dati di riferimento. Il controllo dello schema viene eseguito colonna per colonna. Lo schema di due colonne corrisponde se i nomi sono identici e i tipi sono identici. L'ordine delle colonne non è rilevante.

Sintassi

```
SchemaMatch <REFERENCE_DATASET_ALIAS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS: l'alias del set di dati di riferimento con cui confrontare gli schemi.

  Tipi di colonna supportati: Byte, Decimal, Double, Float, Integer, Long, Short
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: SchemaMatch

La seguente regola di esempio verifica se lo schema del set di dati primario corrisponde esattamente allo schema di un set di dati di riferimento.

```
SchemaMatch "reference" = 1.0
```

Univocità

Verifica la percentuale di valori univoci in una colonna rispetto a una determinata espressione. I valori univoci si verificano esattamente una volta.

Sintassi

```
Uniqueness <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: percentuale di univocità

La seguente regola di esempio verifica se la percentuale di valori univoci in una colonna corrisponde a determinati criteri numerici.

```
Uniqueness "email" = 1.0
```

Regole dinamiche di esempio

- `Uniqueness "colA" between min(last(10)) and max(last(10))`
- `Uniqueness "colA" >= avg(last(10))`

UniqueValueRatio

Verifica il rapporto di valori univoci in una colonna rispetto a una determinata espressione. Un rapporto di valori univoci è la frazione di valori univoci divisa per il numero di tutti i valori distinti in una colonna. I valori univoci si verificano esattamente una volta, mentre i valori distinti si verificano almeno una volta.

Ad esempio, il set `[a, a, b]` contiene un valore univoco (b) e due valori distinti (a e b). Quindi il rapporto di valori univoci del set è ½ = 0,5.

Sintassi

```
UniqueValueRatio <COL_NAME> <EXPRESSION>
```

- COL_NAME: il nome della colonna in base alla quale si desidera valutare la regola di qualità dei dati.

  Tipi di colonna supportati: qualsiasi tipo di colonna
- EXPRESSION: un'espressione da eseguire sulla risposta del tipo di regola per produrre un valore booleano. Per ulteriori informazioni, consulta Espressioni.

Esempio: rapporto di valori univoci

Questo esempio controlla il rapporto tra i valori univoci di una colonna rispetto a un intervallo di valori.

```
UniqueValueRatio "test_score" between 0 and 0.5
```

Regole dinamiche di esempio

- `UniqueValueRatio "colA" > avg(last(10))`

- `UniqueValueRatio "colA" <= index(last(10),2) + std(last(5))`

DetectAnomalies

Rileva le anomalie per una determinata regola Deequ. Ogni esecuzione della DetectAnomalies regola comporta il salvataggio del valore valutato per la regola data. Quando vengono raccolti dati sufficienti, l'algoritmo di rilevamento delle anomalie prende tutti i dati storici relativi a quella determinata regola ed esegue il rilevamento delle anomalie. DetectAnomalies la regola fallisce quando viene rilevata un'anomalia. È possibile ottenere ulteriori informazioni sull'anomalia rilevata tramite le osservazioni.

Sintassi

```
DetectAnomalies <RULE_NAME> <RULE_PARAMETERS>
```

`RULE_NAME`: il nome della regola che desideri valutare e per la quale desideri rilevare le anomalie. Regole supportate:

- "RowCount"
- "Completezza"
- "Univocità"
- "Media"
- "Somma"
- "StandardDeviation"
- "Entropia"
- "DistinctValuesCount"
- "UniqueValueRatio"
- "ColumnLength"
- "ColumnValues"
- "ColumnCorrelation"
- "CustomSql"

`RULE_PARAMETERS`: alcune regole richiedono parametri aggiuntivi per l'esecuzione. Fai riferimento alla documentazione fornita sulle regole per visualizzare i parametri richiesti.

Esempio: anomalie per RowCount

Ad esempio, se vogliamo rilevare RowCount anomalie, forniamo RowCount come regola il nome.

```
DetectAnomalies "RowCount"
```

Esempio: anomalie per ColumnLength

Ad esempio, se vogliamo rilevare ColumnLength anomalie, forniamo ColumnLength come regola il nome e il nome della colonna.

```
DetectAnomalies "ColumnLength" "id"
```

Utilizzo delle API per misurare e gestire la qualità dei dati

Questo argomento descrive come utilizzare le API per misurare e gestire la qualità dei dati.

Indice

Prerequisiti

- Assicurati che la tua versione di boto3 sia aggiornata in modo che includa l'ultima API di Qualità dei dati di AWS Glue.
- Assicurati che la tua versione di AWS CLI sia aggiornata, in modo che includa la CLI più recente.

Se stai usando un processo AWS Glue per eseguire queste API, puoi utilizzare la seguente opzione per aggiornare la libreria boto3 alla versione più recente:

```
—additional-python-modules boto3==<version>
```

Utilizzo dei suggerimenti di Qualità dei dati di AWS Glue

Per avviare un'esecuzione di suggerimento di Qualità dei dati di AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def start_data_quality_rule_recommendation_run(self, database_name, table_name,
 role_arn):
        """
        Starts a recommendation run that is used to generate rules when you don't
 know what rules to write. AWS Glue Data Quality analyzes the data and comes up with
 recommendations for a potential ruleset. You can then triage the ruleset and modify
 the generated ruleset to your liking.

        :param database_name: The name of the AWS Glue database which contains the
 dataset.
        :param table_name: The name of the AWS Glue table against which we want a
 recommendation
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and Access
 Management (IAM) role that grants permission to let AWS Glue access the resources it
 needs.

        """
        try:
            response = self.client.start_data_quality_rule_recommendation_run(
                DataSource={
                    'GlueTable': {
                        'DatabaseName': database_name,
                        'TableName': table_name
                    }
                },
                Role=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't start data quality recommendation run %s. Here's why: %s:
 %s", name,
```

```
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['RunId']
```

Per l'esecuzione di un suggerimento, puoi utilizzare `pushDownPredicates` o per `catalogPartitionPredicates` migliorare le prestazioni ed eseguire i suggerimenti solo su partizioni specifiche delle origini del catalogo.

```
client.start_data_quality_rule_recommendation_run(
            DataSource={
                'GlueTable': {
                    'DatabaseName': database_name,
                    'TableName': table_name,
                    'AdditionalOptions': {
                        'pushDownPredicate': "year=2022"
                    }
                }
            },
            Role=role_arn,
            NumberOfWorkers=2,
            CreatedRulesetName='<rule_set_name>'
  )
```

Per ottenere i risultati dell'esecuzione di un suggerimento di Qualità dei dati di AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_rule_recommendation_run(self, run_id):
        """
        Gets the specified recommendation run that was used to generate rules.

        :param run_id: The id of the data quality recommendation run

        """
        try:
```

```
            response =
 self.client.get_data_quality_rule_recommendation_run(RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get data quality recommendation run %. Here's why: %s: %s",
 run_id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Dall'oggetto di risposta precedente, è possibile estrarre il RuleSet suggerito dall'esecuzione da utilizzare nei passaggi successivi:

```
print(response['RecommendedRuleset'])

Rules = [
    RowCount between 2000 and 8000,
    IsComplete "col1",
    IsComplete "col2",
    StandardDeviation "col3" between 58138330.8 and 64258155.09,
    ColumnValues "col4" between 1000042965 and 1214474826,
    IsComplete "col5"
]
```

Per ottenere un elenco di tutte le esecuzioni suggerite che è possibile filtrare ed elencare:

```
response = client.list_data_quality_rule_recommendation_runs(
    Filter={
        'DataSource': {
            'GlueTable': {
                'DatabaseName': '<database_name>',
                'TableName': '<table_name>'
            }
        }
)
```

Per annullare le attività di suggerimento esistenti di Qualità dei dati di AWS Glue:

```
response = client.cancel_data_quality_rule_recommendation_run(
    RunId='dqrun-d4b6b01957fdd79e59866365bf9cb0e40fxxxxxxx'
)
```

Utilizzo dei set di regole di Qualità dei dati di AWS Glue

Per creare un set di regole di qualità dei dati di AWS Glue:

```
response = client.create_data_quality_ruleset(
    Name='<ruleset_name>',
    Ruleset='Rules = [IsComplete "col1", IsPrimaryKey "col2", RowCount between 2000 and
 8000]',
    TargetTable={
        'TableName': '<table_name>',
        'DatabaseName': '<database_name>'
    }
)
```

Per ottenere un set di regole di qualità dei dati:

```
response = client.get_data_quality_ruleset(
    Name='<ruleset_name>'
)
print(response)
```

Successivamente, puoi utilizzare quest'API per estrarre il set di regole:

```
print(response['Ruleset'])
```

Per elencare tutti i set di regole sulla qualità dei dati per una tabella:

```
response = client.list_data_quality_rulesets()
```

È possibile utilizzare la condizione di filtro all'interno dell'API per filtrare tutti i set di regole collegati a un database o una tabella specifici:

```
response = client.list_data_quality_rulesets(
    Filter={
        'TargetTable': {
            'TableName': '<table_name>',
            'DatabaseName': '<database_name>'
        }
    },
)
```

Per aggiornare un set di regole di qualità dei dati:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def update_data_quality_ruleset(self, ruleset_name, ruleset_string):
        """
        Update an AWS Glue Data Quality Ruleset

        :param ruleset_name: The name of the AWS Glue Data Quality ruleset to update
        :param ruleset_string: The DQDL ruleset string to update the ruleset with

        """
        try:
            response = self.client.update_data_quality_ruleset(
                Name=ruleset_name,
                Ruleset=ruleset_string
            )
        except ClientError as err:
            logger.error(
                "Couldn't update the AWS Glue Data Quality ruleset. Here's why: %s:
 %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Per eliminare un set di regole di qualità dei dati:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def delete_data_quality_ruleset(self, ruleset_name):
```

```
        """
        Delete a AWS Glue Data Quality Ruleset

        :param ruleset_name: The name of the AWS Glue Data Quality ruleset to delete

        """
        try:
            response = self.client.delete_data_quality_ruleset(
                Name=ruleset_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete the AWS Glue Data Quality ruleset. Here's why: %s:
%s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Utilizzo delle esecuzioni di Qualità dei dati di AWS Glue

Per avviare un'esecuzione di Qualità dei dati di AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def start_data_quality_ruleset_evaluation_run(self, database_name, table_name,
 role_name, ruleset_list):
        """
        Start an AWS Glue Data Quality evaluation run

        :param database_name: The name of the AWS Glue database which contains the
 dataset.
        :param table_name: The name of the AWS Glue table against which we want to
 evaluate.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and Access
 Management (IAM) role that grants permission to let AWS Glue access the resources it
 needs.
```

```
        :param ruleset_list: The list of AWS Glue Data Quality ruleset names to
 evaluate.

        """
        try:
            response = client.start_data_quality_ruleset_evaluation_run(
                DataSource={
                    'GlueTable': {
                        'DatabaseName': database_name,
                        'TableName': table_name
                    }
                },
                Role=role_name,
                RulesetNames=ruleset_list
            )
        except ClientError as err:
            logger.error(
                "Couldn't start the AWS Glue Data Quality Run. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['RunId']
```

Ricorda che puoi trasmettere un parametro `pushDownPredicate` o `catalogPartitionPredicate` per garantire che l'esecuzione della qualità dei dati riguardi solo un set specifico di partizioni all'interno della tabella del catalogo. Ad esempio:

```
response = client.start_data_quality_ruleset_evaluation_run(
    DataSource={
        'GlueTable': {
            'DatabaseName': '<database_name>',
            'TableName': '<table_name>',
            'AdditionalOptions': {
                'pushDownPredicate': 'year=2023'
            }
        }
    },
    Role='<role_name>',
    NumberOfWorkers=5,
    Timeout=123,
    AdditionalRunOptions={
        'CloudWatchMetricsEnabled': False
    },
```

```
    RulesetNames=[
        '<ruleset_name>',
    ]
)
```

Per ottenere informazioni su un'esecuzione di Qualità dei dati di AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_ruleset_evaluation_run(self, run_id):
        """
        Get details about an AWS Glue Data Quality Run

        :param run_id: The AWS Glue Data Quality run ID to look up

        """
        try:
            response = self.client.get_data_quality_ruleset_evaluation_run(
                RunId=run_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't look up the AWS Glue Data Quality run ID. Here's why: %s:
 %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Per ottenere i risultati di un'esecuzione di Qualità dei dati di AWS Glue:

Per una determinata esecuzione di Qualità dei dati di AWS Glue, è possibile estrarre i risultati della valutazione dell'esecuzione utilizzando il seguente metodo:

```
response = client.get_data_quality_ruleset_evaluation_run(
    RunId='d4b6b01957fdd79e59866365bf9cb0e40fxxxxxxx'
)
```

```
resultID = response['ResultIds'][0]

response = client.get_data_quality_result(
    ResultId=resultID
)

print(response['RuleResults'])
```

Per elencare tutte le esecuzioni di Qualità dei dati di AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def list_data_quality_ruleset_evaluation_runs(self, database_name, table_name):
        """
        Lists all the AWS Glue Data Quality runs against a given table

        :param database_name: The name of the database where the data quality runs
        :param table_name: The name of the table against which the data quality runs
 were created

        """
        try:
            response = self.client.list_data_quality_ruleset_evaluation_runs(
                Filter={
                    'DataSource': {
                        'GlueTable': {
                            'DatabaseName': database_name,
                            'TableName': table_name
                        }
                    }
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't list the AWS Glue Quality runs. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
```

```
            raise
        else:
            return response
```

È possibile modificare la clausola di filtro in modo che mostri solo i risultati compresi entro orari specifici o in base a tabelle specifiche.

Per interrompere un'esecuzione in corso di Qualità dei dati di AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def cancel_data_quality_ruleset_evaluation_run(self, result_id):
        """
        Cancels a given AWS Glue Data Quality run

        :param result_id: The result id of a AWS Glue Data Quality run to cancel

        """
        try:
            response = self.client.cancel_data_quality_ruleset_evaluation_run(
                ResultId=result_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't cancel the AWS Glue Data Quality run. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Utilizzo dei risultati di Qualità dei dati di AWS Glue

Per ottenere i risultati dell'esecuzione di Qualità dei dati di AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_result(self, result_id):
        """
        Outputs the result of an AWS Glue Data Quality Result

        :param result_id: The result id of an AWS Glue Data Quality run

        """
        try:
            response = self.client.get_data_quality_result(
                ResultId=result_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get the AWS Glue Data Quality result. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Per annullare le attività di suggerimento esistenti di Qualità dei dati di AWS Glue:

Dato un ID di esecuzione di Qualità dei dati di AWS Glue, puoi estrarre l'ID del risultato per poi ottenere i risultati effettivi, come mostrato di seguito:

```
response = client.get_data_quality_ruleset_evaluation_run(
    RunId='dqrun-abca77ee126abe1378c1da1ae0750xxxxxxxx'
)

resultID = response['ResultIds'][0]

response = client.get_data_quality_result(
    ResultId=resultID
)

print(resp['RuleResults'])
```

Configurazione di avvisi, implementazioni e pianificazioni

Questo argomento descrive come configurare avvisi, distribuzioni e pianificazione per AWS Glue Data Quality.

Indice

Configurazione di avvisi e notifiche nell'integrazione con Amazon EventBridge

AWS Glue Data Quality supporta la pubblicazione di EventBridge eventi, che vengono emessi al termine di un ciclo di valutazione del set di regole di Data Quality. In questo modo, è possibile impostare facilmente avvisi quando le regole di qualità dei dati non vengono soddisfatte.

Ecco un esempio di evento relativo alla valutazione dei set di regole sulla qualità dei dati in Catalogo dati. Con queste informazioni, puoi esaminare i dati resi disponibili con Amazon EventBridge. È possibile effettuare chiamate API aggiuntive per ottenere maggiori dettagli. Ad esempio, chiama l'API `get_data_quality_result` con l'ID del risultato per ottenere i dettagli di una particolare esecuzione.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Data Quality Evaluation Results Available",
    "source":"aws.glue-dataquality",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
```

```
        "context": {
                    "contextType": "GLUE_DATA_CATALOG",
                    "runId":"dqrun-12334567890",
                    "databaseName": "db-123",
                    "tableName": "table-123",
                    "catalogId": "123456789012"
                    },
        "resultID": "dqresult-12334567890",
        "rulesetNames": ["rulset1"],
        "state":"SUCCEEDED",
        "score": 1.00,
        "rulesSucceeded": 100,
        "rulesFailed": 0,
        "rulesSkipped": 0
    }
}
```

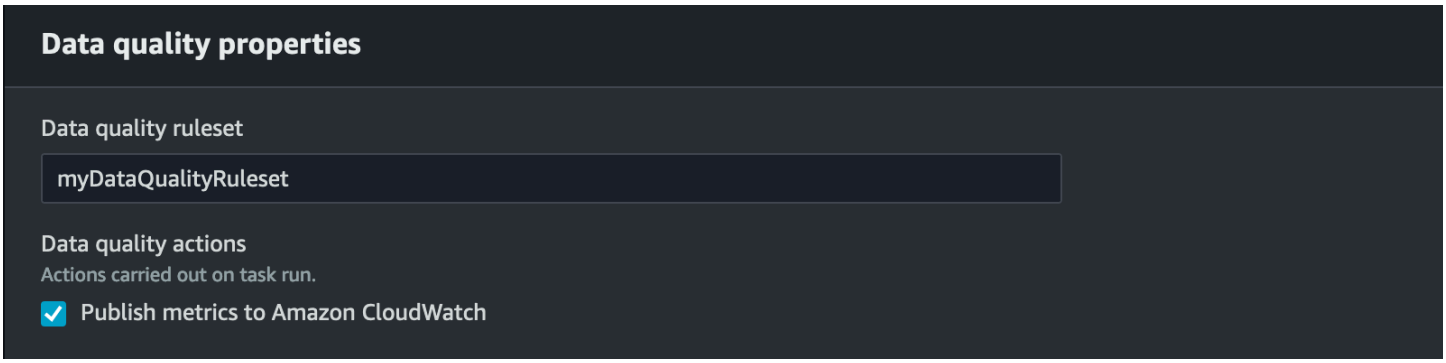
Ecco un esempio di evento che viene pubblicato quando si valutano i set di regole di qualità dei dati nei notebook Glue AWS ETL o AWS Glue Studio.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Data Quality Evaluation Results Available",
    "source":"aws.glue-dataquality",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
        "context": {
                    "contextType": "GLUE_JOB",
                    "jobId": "jr-12334567890",
                    "jobName": "dq-eval-job-1234",
                    "evaluationContext": "",
                    }
        "resultID": "dqresult-12334567890",
        "rulesetNames": ["rulset1"],
        "state":"SUCCEEDED",
        "score": 1.00
        "rulesSucceeded": 100,
        "rulesFailed": 0,
        "rulesSkipped": 0
```

```
    }
}
```

Affinché la valutazione della qualità dei dati venga eseguita sia nel Data Catalog che nei job ETL, l'opzione Publish metrics to Amazon Cloudwatch, selezionata per impostazione predefinita, deve rimanere selezionata affinché la EventBridge pubblicazione funzioni.

Configurazione delle notifiche EventBridge



Per ricevere gli eventi emessi e definire gli obiettivi, devi configurare EventBridge le regole di Amazon. Per creare regole:

1. Apri la EventBridge console Amazon.
2. Scegli Regole nella sezione Router della barra di navigazione.
3. Selezionare Create Rule (Crea regola).
4. In Definisci i dettagli della regola:
   a. In Nome, inserisci `myDQRule`.
   b. Inserisci una descrizione (facoltativa).
   c. Per Router di eventi, seleziona il tuo router. Se non ne hai uno, lascia quello predefinito.
   d. Per Tipo di regola, scegli Regola con un modello di eventi, quindi scegli Successivo.
5. In Crea un modello di eventi:
   a. Per l'origine dell'evento, seleziona AWS eventi o eventi dei EventBridge partner.
   b. Puoi saltare la sezione Evento di esempio.
   c. Per il metodo di creazione, seleziona Utilizza modulo del modello.
   d. Per il modello dell'evento:
      i. Seleziona Servizi AWS come Origine dell'evento.
      ii. Seleziona Glue Data Quality per l' AWS assistenza.

iii. Seleziona Risultati di valutazione della qualità dei dati disponibili per Tipo di evento.

iv. Seleziona NON RIUSCITO per Stati specifici. Viene quindi visualizzato un modello di eventi simile al seguente:

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "state": ["FAILED"]
  }
}
```

v. Per ulteriori opzioni di connessione, consulta la sezione Opzioni di configurazione aggiuntive per il modello di evento.

6. Su Seleziona destinazioni:
   a. Per Tipi di destinazione, seleziona Servizio AWS .
   b. Utilizza il menu a discesa Seleziona una destinazione per scegliere il AWS servizio desiderato a cui connetterti (SNS, Lambda, SQS, ecc.), quindi scegli Avanti.
7. In Configura tag, fai clic su Aggiungi nuovi tag per aggiungere tag facoltativi, quindi scegli Successivo.
8. Viene visualizzata una pagina di riepilogo di tutte le selezioni. Scegli Crea regola in basso.

Opzioni di configurazione aggiuntive per il modello di evento

Oltre a filtrare l'evento in base alla riuscita o al fallimento, potresti voler filtrare ulteriormente gli eventi in base a parametri diversi.

Per fare ciò, vai alla sezione Modello di evento e seleziona Modifica modello per specificare parametri aggiuntivi. Nota che i campi del modello di evento fanno distinzione tra maiuscole e minuscole. Di seguito sono riportati alcuni esempi di configurazione del modello di evento.

Per acquisire eventi da una particolare tabella valutando set di regole specifici, utilizza questo tipo di modello:

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
```

```
    "context": {
      "contextType": ["GLUE_DATA_CATALOG"],
      "databaseName": "db-123",
       "tableName": "table-123",
    },
    "rulesetNames": ["ruleset1", "ruleset2"]
    "state": ["FAILED"]
  }
}
```

Per acquisire eventi da processi specifici nell'esperienza ETL, utilizza questo tipo di modello:

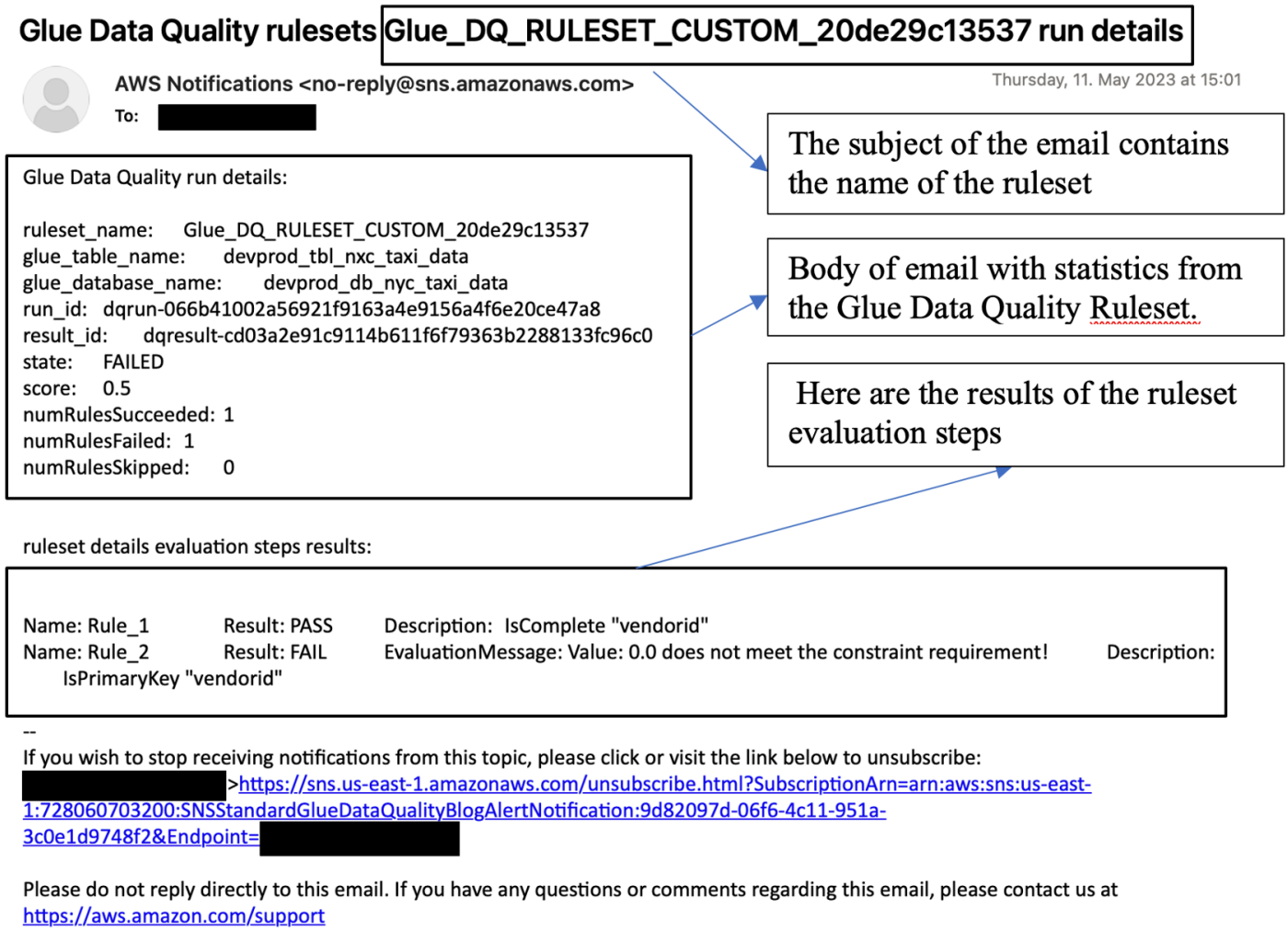
```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "context": {
      "contextType": ["GLUE_JOB"],
      "jobName": ["dq_evaluation_job1", "dq_evaluation_job2"]
    },
    "state": ["FAILED"]
  }
}
```

Per registrare eventi con un punteggio inferiore a una soglia specifica (ad esempio 70%):

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "score": [{
      "numeric": ["<=", 0.7]
    }]
  }
}
```

Formattazione delle notifiche in formato e-mail

A volte è necessario inviare una notifica e-mail ben formattata ai team aziendali. Puoi usare Amazon EventBridge e AWS Lambda per raggiungere questo obiettivo.



Il seguente codice di esempio può essere utilizzato per formattare le notifiche sulla qualità dei dati per generare e-mail.

```
import boto3
import json
from datetime import datetime

sns_client = boto3.client('sns')
glue_client = boto3.client('glue')

sns_topic_arn = 'arn:aws:sns:<region-code>:<account-id>:<sns-topic-name>'



def lambda_handler(event, context):
    log_metadata = {}
```

```
    message_text = ""
    subject_text = ""

    if event['detail']['context']['contextType'] == 'GLUE_DATA_CATALOG':
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['tableName'] = str(event['detail']['context']['tableName'])
        log_metadata['databaseName'] = str(event['detail']['context']['databaseName'])
        log_metadata['runId'] = str(event['detail']['context']['runId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])
        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_table_name: {}\n".format(log_metadata['tableName'])
        message_text += "glue_database_name: {}\n".format(log_metadata['databaseName'])
        message_text += "run_id: {}\n".format(log_metadata['runId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    else:
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['jobName'] = str(event['detail']['context']['jobName'])
        log_metadata['jobId'] = str(event['detail']['context']['jobId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])

        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
```

```
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_job_name: {}\n".format(log_metadata['jobName'])
        message_text += "job_id: {}\n".format(log_metadata['jobId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    resultID = str(event['detail']['resultId'])
    response = glue_client.get_data_quality_result(ResultId=resultID)
    RuleResults = response['RuleResults']
    message_text += "\n\nruleset details evaluation steps results:\n\n"
    subresult_info = []

    for dic in RuleResults:
        subresult = "Name: {}\t\tResult: {}\t\tDescription: \t{}".format(dic['Name'],
dic['Result'], dic['Description'])
        if 'EvaluationMessage' in dic:
            subresult += "\t\tEvaluationMessage: {}".format(dic['EvaluationMessage'])
        subresult_info.append({
            'Name': dic['Name'],
            'Result': dic['Result'],
            'Description': dic['Description'],
            'EvaluationMessage': dic.get('EvaluationMessage', '')
        })
        message_text += "\n" + subresult

    log_metadata['resultrun'] = subresult_info



    sns_client.publish(
        TopicArn=sns_topic_arn,
        Message=message_text,
        Subject=subject_text
    )

    return {
```

```
        'statusCode': 200,
        'body': json.dumps('Message published to SNS topic')
    }
```

Imposta avvisi e notifiche in integrazione CloudWatch

Il nostro approccio consigliato consiste nell'impostare avvisi sulla qualità dei dati utilizzando Amazon EventBridge, poiché Amazon EventBridge richiede una configurazione unica per avvisare i clienti. Tuttavia, alcuni clienti preferiscono Amazon CloudWatch per familiarità. Per tali clienti, offriamo l'integrazione con Amazon CloudWatch.

Ogni valutazione di AWS Glue Data Quality emette un paio di metriche denominate `glue.data.quality.rules.passed` (che indicano un numero di regole passate) e `glue.data.quality.rules.failed` (che indica il numero di regole non riuscite) per ogni esecuzione sulla qualità dei dati. È possibile utilizzare questo parametro emesso per creare allarmi per avvisare gli utenti se un determinato ciclo di qualità dei dati scende al di sotto di una soglia. Per iniziare a configurare un allarme che invii un'e-mail tramite una notifica Amazon SNS, procedi nel modo seguente:

Per iniziare a configurare un allarme che invii un'e-mail tramite una notifica Amazon SNS, procedi nel modo seguente:

1. Apri la CloudWatch console Amazon.
2. Scegli Tutti i parametri in Parametri. Vedrai uno spazio dei nomi aggiuntivo in Spazi dei nomi personalizzati intitolato Qualità dei dati di Glue.

   > **Note**
   >
   > Quando avvii un'esecuzione di AWS Glue Data Quality, assicurati che la CloudWatch casella di controllo Publish metrics to Amazon sia abilitata. Altrimenti, le metriche per quella particolare corsa non verranno pubblicate su Amazon CloudWatch.

   Nello spazio del nome `Glue Data Quality`, puoi visualizzare i parametri emessi per tabella e per set di regole. Ai fini di questo argomento, useremo la regola `glue.data.quality.rules.failed` e attiveremo l'allarme se questo valore supera 1 (indicando che, se riscontriamo un numero di valutazioni delle regole non riuscite superiore a 1, vogliamo ricevere una notifica).

3. Per creare l'allarme, scegli Tutti gli allarmi in Allarmi.
4. Scegli Crea allarme.
5. Scegli Select Metric (Seleziona parametro).
6. Seleziona il parametro `glue.data.quality.rules.failed` corrispondente alla tabella che hai creato, quindi scegli Seleziona parametro.
7. Nella scheda Specifica parametri e condizioni, nella sezione Parametri:
   a. Per Statistic (Statistica), scegliere Sum (Somma).
   b. Per Periodo, scegli 1 minuto.
8. Nella sezione Condizioni:
   a. For Threshold type (Tipo di soglia), scegli Static (Statica).
   b. Per Quando glue.data.quality.rules.failed è..., seleziona Maggiore di/uguale a.
   c. Per Oltre..., inserisci 1 come valore di soglia.

   Queste selezioni implicano che se il parametro `glue.data.quality.rules.failed` emette un valore maggiore o uguale a 1, attiveremo un allarme. Tuttavia, se non ci sono dati, lo considereremo accettabile.
9. Seleziona Avanti.
10. In Configura operazioni:
    a. Per Attivazione dello stato di allarme, scegli In allarme.
    b. Nella sezione Invia una notifica al seguente argomento SNS, scegli Crea un nuovo argomento per inviare una notifica tramite un nuovo argomento SNS.
    c. In Endpoint e-mail che riceveranno la notifica, inserisci il tuo indirizzo e-mail. Quindi, fai clic su Crea argomento.
    d. Seleziona Avanti.
11. In Nome dell'allarme, inserisci `myFirstDQAlarm`, quindi seleziona Successivo.
12. Viene visualizzata una pagina di riepilogo di tutte le selezioni. Scegli Crea allarme in basso.

Ora puoi vedere l'allarme creato dalla dashboard degli CloudWatch allarmi di Amazon.

Esecuzione di query di risultati sulla qualità dei dati per creare pannelli di controllo

Potresti voler creare un pannello di controllo per visualizzare i risultati di qualità dei dati. Ci sono due modi per effettuare questa operazione:

Configura Amazon EventBridge con il seguente codice per scrivere i dati su Amazon S3:

```
import boto3
import json
from datetime import datetime


s3_client = boto3.client('s3')
glue_client = boto3.client('glue')


s3_bucket = 's3-bucket-name'

def write_logs(log_metadata):
    try:
        filename = datetime.now().strftime("%m%d%Y%H%M%S") + ".json"
        key_opts = {
            'year': datetime.now().year,
            'month': "{:02d}".format(datetime.now().month),
            'day': "{:02d}".format(datetime.now().day),
            'filename': filename
        }
        s3key = "gluedataqualitylogs/year={year}/month={month}/day={day}/
{filename}".format(**key_opts)
        s3_client.put_object(Bucket=s3_bucket, Key=s3key,
 Body=json.dumps(log_metadata))
    except Exception as e:
        print(f'Error writing logs to S3: {e}')


def lambda_handler(event, context):
    log_metadata = {}
    message_text = ""
    subject_text = ""

    if event['detail']['context']['contextType'] == 'GLUE_DATA_CATALOG':
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
```

```
        log_metadata['tableName'] = str(event['detail']['context']['tableName'])
        log_metadata['databaseName'] = str(event['detail']['context']['databaseName'])
        log_metadata['runId'] = str(event['detail']['context']['runId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])
        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_table_name: {}\n".format(log_metadata['tableName'])
        message_text += "glue_database_name: {}\n".format(log_metadata['databaseName'])
        message_text += "run_id: {}\n".format(log_metadata['runId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    else:
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['jobName'] = str(event['detail']['context']['jobName'])
        log_metadata['jobId'] = str(event['detail']['context']['jobId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])

        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_job_name: {}\n".format(log_metadata['jobName'])
        message_text += "job_id: {}\n".format(log_metadata['jobId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
```

```
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    resultID = str(event['detail']['resultId'])
    response = glue_client.get_data_quality_result(ResultId=resultID)
    RuleResults = response['RuleResults']
    message_text += "\n\nruleset details evaluation steps results:\n\n"
    subresult_info = []

    for dic in RuleResults:
        subresult = "Name: {}\t\tResult: {}\t\tDescription: \t{}".format(dic['Name'],
dic['Result'], dic['Description'])
        if 'EvaluationMessage' in dic:
            subresult += "\t\tEvaluationMessage: {}".format(dic['EvaluationMessage'])
        subresult_info.append({
            'Name': dic['Name'],
            'Result': dic['Result'],
            'Description': dic['Description'],
            'EvaluationMessage': dic.get('EvaluationMessage', '')
        })
        message_text += "\n" + subresult

    log_metadata['resultrun'] = subresult_info

    write_logs(log_metadata)


    return {
        'statusCode': 200,
        'body': json.dumps('Message published to SNS topic')
    }
```

Dopo aver scritto su Amazon S3, puoi usare i crawler AWS Glue per registrarti su Athena e interrogare le tabelle.

Configurazione di una posizione Amazon S3 durante una valutazione della qualità dei dati:

Quando esegui attività di qualità dei dati in AWS Glue Data Catalog o AWS Glue ETL, puoi fornire una posizione Amazon S3 per scrivere i risultati sulla qualità dei dati su Amazon S3. Per creare una tabella facendo riferimento alla destinazione per leggere i risultati di qualità dei dati, puoi utilizzare la sintassi seguente.

Tieni presente che è necessario eseguire le query `CREATE EXTERNAL TABLE` e `MSCK REPAIR TABLE` separatamente.

```
CREATE EXTERNAL TABLE <my_table_name>(
    catalogid string,
    databasename string,
    tablename string,
    dqrunid string,
    evaluationstartedon timestamp,
    evaluationcompletedon timestamp,
    rule string,
    outcome string,
    failurereason string,
    evaluatedmetrics string)
PARTITIONED BY (
    `year` string,
    `month` string,
    `day` string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (

 'paths'='catalogId,databaseName,dqRunId,evaluatedMetrics,evaluationCompletedOn,evaluationStart
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://glue-s3-dq-bucket-us-east-2-results/'
TBLPROPERTIES (
    'classification'='json',
    'compressionType'='none',
    'typeOfData'='file');
```

```
MSCK REPAIR TABLE <my_table_name>;
```

Dopo aver creato la tabella precedente, puoi eseguire query analitiche utilizzando Amazon Athena.

Implementazione delle regole di qualità dei dati utilizzando AWS CloudFormation

È possibile utilizzare AWS CloudFormation per creare regole di qualità dei dati. Per ulteriori informazioni, vedere AWS CloudFormation AWS Glue.

Pianificazione delle regole di qualità dei dati

È possibile pianificare le regole di qualità dei dati utilizzando i seguenti metodi:

- Pianifica le regole di qualità dei dati dal Data Catalog: gli utenti senza codice possono utilizzare questa opzione per pianificare facilmente le scansioni della qualità dei dati. AWS Glue Data Quality creerà la pianificazione in Amazon EventBridge. Per pianificare le regole di qualità dei dati:
  - Vai al set di regole e fai clic su Esegui.
  - In Frequenza di esecuzione, seleziona la pianificazione desiderata e fornisci un Nome dell'attività. Questo nome dell'attività è il nome della tua pianificazione in EventBridge.
- Usa Amazon EventBridge e AWS Step Functions per orchestrare valutazioni e raccomandazioni per le regole di qualità dei dati.

Risoluzione degli errori di AWS Glue Data Quality

Se riscontri errori in AWS Glue Data Quality, utilizza le seguenti soluzioni per aiutarti a trovare l'origine dei problemi e risolverli.

Indice

- ERRORE DI AVVIO: errore durante il download da S3 per il bucket
- InvalidInputException (status: 400): DataQuality le regole non possono essere analizzate
- Errore: EventBridge non attiva i processi Qualità dei dati di Glue in base alla pianificazione che ho impostato
- Errori CustomSQL
- Regole dinamiche
- Eccezione nella classe utente: org.apache.spark.sql. AnalysisException: org.apache.hadoop.hive.ql.metadata. HiveException

Errore: modulo AWS Glue Data Quality mancante

Messaggio di errore: No module named 'awsgluedq'.

Risoluzione: questo errore si verifica quando si esegue AWS Glue Data Quality in una versione non supportata. AWS Glue Data Quality è supportato solo nella versione 3.0 e successive di Glue.

Errore: permessi di AWS Lake Formation insufficienti

Messaggio di errore: Eccezione nella classe utente:`com.amazonaws.services.glue.model.AccessDeniedException`: Autorizzazioni Lake Formation insufficienti su impact_sdg_engagement (Service:; Codice di stato: 400; Codice di errore:AWS Glue; ID richiesta: AccessDeniedException 465ae693-b7ba-4df0-a4e4-6b17xxxxxxxx; Proxy: null).

Risoluzione: è necessario fornire autorizzazioni sufficienti in AWS Lake Formation.

Errore: i set di regole non hanno un nome univoco

Messaggio di errore: eccezione nella classe utente:... services.glue.model. AlreadyExistsException: Esiste già un altro set di regole con lo stesso nome.

Risoluzione: i set di regole sono globali e devono essere univoci.

Errore: tabelle con caratteri speciali

Messaggio di errore: eccezione nella classe utente: org.apache.spark.sql. AnalysisException: impossibile risolvere le colonne di input «C» fornite: [primary.data_end_time, primary.data_start_time,

primary.end_time, primary.last_updated, primary.message, primary.process_date, primary.rowhash, primary.run_by, primary.run_id, primary.start_time, primary.status]; riga 1 pos 44;.

Risoluzione: attualmente esiste una limitazione per cui AWS Glue Data Quality non può essere eseguito su tabelle che contengono caratteri speciali come «.».

Errore: errore di overflow con un set di regole di grandi dimensioni

Messaggio di errore: Eccezione nella classe utente: java.lang. StackOverflowError.

Risoluzione: se disponi di un set di regole di grandi dimensioni con più di 2.000 regole, potresti riscontrare questo problema. Suddividi le tue regole in più set di regole.

Errore: lo stato generale della regola è non riuscito

Condizione di errore: il mio set di regole ha esito positivo, ma lo stato generale delle regole non è riuscito.

Risoluzione: questo errore si è probabilmente verificato perché hai scelto l'opzione per pubblicare le metriche su Amazon CloudWatch durante la pubblicazione. Se il tuo set di dati è in un VPC, il tuo VPC potrebbe non consentire a AWS Glue di pubblicare metriche su Amazon. CloudWatch In questo caso, devi >configurare un endpoint per consentire al tuo VPC di accedere ad Amazon. CloudWatch

AnalysisException: impossibile verificare l'esistenza del database predefinito

Condizione di errore AnalysisException: impossibile verificare l'esistenza del database predefinito: com.amazonaws.services.glue.model. AccessDeniedException: Autorizzazioni Lake Formation insufficienti per impostazione predefinita (Service:AWS Glue; Codice di stato: 400; Codice errore:; ID richiesta: XXXXXXXX-XXXX-XXXX-XXXX-XXXX-XXXXXXXX AccessDeniedException; Proxy: null)

Risoluzione: nell'integrazione del catalogo del processo AWS Glue, AWS Glue cerca sempre di verificare se il database predefinito esiste o meno utilizzando l'`GetDatabase API` AWS Glue. Quando l'autorizzazione `DESCRIBE` Lake Formation non viene concessa o viene concessa l'autorizzazione `GetDatabase IAM`, il processo ha esito negativo durante la verifica dell'esistenza del database predefinito.

Per risolvere:

1. Aggiungi l'autorizzazione `DESCRIBE` in Lake Formation per il database predefinito.

2. Configura il ruolo IAM associato al processo AWS Glue come Creatore di database in Lake Formation. Questo creerà automaticamente un database predefinito e concederà le autorizzazioni Lake Formation richieste per il ruolo.
3. Disabilita l'opzione `--enable-data-catalog`. Viene visualizzato come Utilizza Data Catalog come metastore Hive in AWS Glue Studio.

   Se l'integrazione Data Catalog Spark SQL non è necessaria nel processo, è possibile disabilitarla.

Messaggio di errore: Provided key map not suitable for given data frames

Condizione di errore: la mappa delle chiavi fornita non è adatta a determinati frame di dati.

Risoluzione: DataSetMatchstai usando il tipo di regola e le chiavi di join hanno dei duplicati. Le tue chiavi di join devono essere univoche e non possono essere NULL. Nei casi in cui non puoi avere chiavi di join univoche, prendi in considerazione l'utilizzo di altri tipi di regole, AggregateMatchad esempio la corrispondenza nei dati di riepilogo.

Eccezione nella classe utente: java.lang. RuntimeException : Impossibile recuperare i dati. Controlla i log in CloudWatch per avere maggiori dettagli

Condizione di errore: eccezione nella classe utente: java.lang. RuntimeException : Impossibile recuperare i dati. Controlla i log in CloudWatch per avere maggiori dettagli.

Risoluzione: questo accade quando crei regole DQ su una tabella basata su Amazon S3 confrontabile con Amazon RDS o. Amazon Redshift In questi casi, AWS Glue non è in grado di caricare la connessione. Prova invece a configurare la regola DQ sul set di dati Amazon Redshift o Amazon RDS. Si tratta di un bug noto.

ERRORE DI AVVIO: errore durante il download da S3 per il bucket

Condizione di errore: ERRORE DI AVVIO: Errore durante il download da S3 per il bucket: `aws-glue-ml-data-quality-assets-us-east-1, key: jars/aws-glue-ml-data-quality-etl.jar.Access Denied (Service: Amazon S3; Status Code: 403; Please refer logs for details)` .

Risoluzione: le autorizzazioni relative al ruolo passate a AWS Glue Data Quality devono consentire la lettura dalla precedente posizione Amazon S3. Al ruolo deve essere collegata questa policy IAM:

```
{
  "Sid": "allowS3",
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::aws-glue-ml-data-quality-assets-<region>/*"
}
```

Fai riferimento all'Autorizzazione di Qualità dei dati per le autorizzazioni dettagliate. Queste librerie sono necessarie per valutare la qualità dei dati per i tuoi set di dati.

InvalidInputException (status: 400): DataQuality le regole non possono essere analizzate

Condizione di errore: InvalidInputException (status: 400): DataQuality le regole non possono essere analizzate.

Risoluzione: sono molte le possibili cause di questo errore. Una possibilità è che le regole siano racchiuse tra virgolette singole. Verifica che siano racchiuse tra virgolette doppie. Per esempio:

```
Rules = [
ColumnValues "tipo_vinculo" in ["CODO", "DOCO", "COCO", "DODO"] AND "categoria" = 'ES'
    AND "cod_bandera" = 'CEP'
```

Modificalo con:

```
Rules = [
(ColumnValues "tipovinculo" in [ "CODO", "DOCO", "COCO", "DODO"]) AND (ColumnValues
 "categoria" = "ES")
    AND (ColumnValues "codbandera" = "CEP")
]
```

Errore: EventBridge non attiva i processi Qualità dei dati di Glue in base alla pianificazione che ho impostato

Condizione di errore: EventBridge non attiva i processi AWS Glue Data Quality in base alla pianificazione che ho impostato.

Risoluzione: il ruolo che attiva il processo potrebbe non avere le autorizzazioni corrette. Assicurati che il ruolo che stai utilizzando per avviare i processi disponga delle autorizzazioni menzionate nella sezione Configurazione IAM richiesta per la pianificazione delle esecuzioni di valutazione.

Errori CustomSQL

Condizione di errore: `The output from CustomSQL must contain at least one column that matches the input dataset for AWS Glue Data Quality to provide row level results. The SQL query is a valid query but no columns from the SQL result are present in the Input Dataset. Ensure that matching columns are returned from the SQL`.

Risoluzione: la query SQL è valida, ma assicurati di selezionare solo le colonne della tabella primaria. La selezione di funzioni aggregate come somma o conteggio delle colonne della tabella primaria può causare questo errore.

Condizione di errore: `There was a problem when executing your SQL statement: cannot resolve "Col"`.

Risoluzione: questa colonna non è presente nella tabella primaria.

Condizione di errore: `The columns that are returned from the SQL statement should only belong to the primary table. "In this case, some columns ( Col ) belong to reference table"`.

Risoluzione: nelle query SQL, quando esegui il join della tabella primaria con altre tabelle di riferimento, assicurati che l'istruzione select contenga solo i nomi di colonna della tabella primaria per generare risultati a livello di riga per tale tabella.

Regole dinamiche

Condizione di errore: `Dynamic rules require job context, and cannot be evaluated in interactive session or data preview.`.

Causa: questo messaggio di errore potrebbe apparire nei risultati dell'anteprima dei dati, o in altre sessioni interattive, quando nel set di regole sono presenti regole di qualità dei dati dinamiche. Le regole dinamiche fanno riferimento alle metriche storiche associate a un particolare nome di processo e contesto di valutazione, quindi non possono essere valutate nelle sessioni interattive.

Risoluzione: l'esecuzione del processo AWS Glue produrrà metriche storiche, a cui è possibile fare riferimento nelle esecuzioni successive per lo stesso processo.

Condizione di errore:

- `[RuleType] rule only supports simple atomic operands in thresholds.`.
- `Function last not yet implemented for [RuleType] rule.`

Risoluzione: le regole dinamiche sono generalmente supportate per tutti i tipi di regole DQDL nelle espressioni numeriche (consulta il Riferimento a Data Quality Definition Language (DQDL)). Tuttavia, alcune regole che producono più metriche non sono ancora supportate. ColumnValues ColumnLength

Condizione di errore: `Binary expression operands must resolve to a single number.`.

Causa: le regole dinamiche supportano le espressioni binarie, come `RowCount > avg(last(5)) * 0.9`. In questo caso, l'espressione binaria è `avg(last(5)) * 0.9`. Questa regola è valida perché entrambi gli operandi `avg(last(5))` e `0.9` si risolvono in un unico numero. Un esempio errato è `RowCount > last(5) * 0.9`, perché `last(5)` produrrà un elenco che non può essere confrontato in modo significativo con il conteggio delle righe corrente.

Risoluzione: utilizza le funzioni di aggregazione per ridurre un operando con valori di elenco a un unico numero.

Condizione di errore:

- `Rule threshold results in list, and a single value is expected. Use aggregation functions to produce a single value. Valid example: sum(last(10)), avg(last(10)).`
- `Rule threshold results in empty list, and a single value is expected.`

Causa: è possibile utilizzare regole dinamiche per confrontare alcune funzionalità del set di dati con i valori storici corrispondenti. L'ultima funzione consente il recupero di più valori storici, se viene fornito un argomento intero positivo. Ad esempio, `last(5)` recupererà i cinque valori più recenti osservati nelle esecuzioni dei processi per la regola.

Risoluzione: è necessario utilizzare una funzione di aggregazione per ridurre questi valori a un unico numero per effettuare un confronto significativo con il valore osservato nell'esecuzione del processo corrente.

Esempi validi:

- `RowCount >= avg(last(5))`
- `RowCount > last(1)`
- `RowCount < last()`

Esempio non valido: `RowCount > last(5)`.

Condizione di errore:

- `Function index used in threshold requires positive integer argument.`
- `Index argument must be an integer. Valid syntax example: RowCount > index(last(10, 2)), which means RowCount must be greater than third most recent execution from last 10 job runs.`

Risoluzione: durante la creazione di regole dinamiche, è possibile utilizzare la funzione di aggregazione `index` per selezionare un valore storico da un elenco. Ad esempio, `RowCount > index(last(5)`, 1) controllerà se il conteggio delle righe osservato nel processo corrente è strettamente maggiore del secondo conteggio di righe più recente osservato per il processo. `index` è indicizzato a zero.

Condizione di errore: `IllegalArgumentException: Parsing Error: Rule Type: DetectAnomalies is not valid`.

Risoluzione: il rilevamento delle anomalie è disponibile solo in AWS Glue 4.0.

Condizione di errore: `IllegalArgumentException: Parsing Error: Unexpected condition for rule of type ... no viable alternative at input ...`.

Nota: `...` è dinamico. Esempio: `IllegalArgumentException: Parsing Error: Unexpected condition for rule of type RowCount with number return type, line 4:19 no viable alternative at input '>last'`.

Risoluzione: il rilevamento delle anomalie è disponibile solo in AWS Glue 4.0.

Eccezione nella classe utente: org.apache.spark.sql. AnalysisException: org.apache.hadoop.hive.ql.metadata. HiveException

Condizione di errore : `Exception in User Class: org.apache.spark.sql.AnalysisException: org.apache.hadoop.hive.ql.metadata.HiveException: Unable to fetch table mailpiece_submitted. StorageDescriptor#InputFormat cannot be null for table: mailpiece_submitted (Service: null; Status Code: 0; Error Code: null; Request ID: null; Proxy: null)`

Causa: stai usando Apache Iceberg in AWS Glue Data Catalog e l'attributo Input Format in AWS Glue Data Catalog è vuoto.

Soluzione: questo problema si verifica quando si utilizza il tipo di regola CustomSQL nella regola DQ. Un modo per risolvere questo problema consiste nell'utilizzare «primario» o aggiungere il nome del catalogo a. `glue_catalog.<database>.<table> in Custom ruletype`

Integrazione dei dati di Amazon Q in AWS Glue

L'integrazione dei dati in Amazon Q AWS Glue è in versione di anteprima ed è soggetta a modifiche.

L'integrazione dei dati di Amazon Q AWS Glue è una nuova funzionalità di intelligenza artificiale generativa AWS Glue che consente ai data engineer e agli sviluppatori ETL di creare lavori di integrazione dei dati utilizzando il linguaggio naturale. Gli ingegneri e gli sviluppatori possono chiedere a Q di creare lavori, risolvere problemi e rispondere a domande sull' AWS Glue integrazione dei dati.

Che cos'è Amazon Q?

> **Note**
>
> Realizzato da Amazon Bedrock: AWS implementa il rilevamento automatico degli abusi. Poiché l'integrazione dei dati di Amazon Q è basata su Amazon Bedrock, gli utenti possono sfruttare appieno i controlli implementati in Amazon Bedrock per rafforzare la sicurezza e l'uso responsabile dell'intelligenza artificiale (AI).

Amazon Q è un assistente conversazionale basato sull'intelligenza artificiale generativa (AI) che può aiutarti a comprendere, creare, estendere e utilizzare le applicazioni. AWS Il modello alla base di Amazon Q è stato arricchito con AWS contenuti di alta qualità per fornirti risposte più complete, attuabili e referenziate per accelerare la tua crescita. AWS Per ulteriori informazioni, consulta Che cos'è Amazon Q?

Che cos'è l'integrazione dei dati di Amazon Q in AWS Glue?

L'integrazione dei dati di Amazon Q in AWS Glue include le seguenti funzionalità:

- Chat: Amazon Q Data Integration in AWS Glue può rispondere a domande in linguaggio naturale in inglese su AWS Glue domini di integrazione dei dati come connettori di AWS Glue origine e destinazione, job AWS Glue ETL, Data Catalog, crawler e AWS Lake Formation altra

documentazione sulle funzionalità e best practice. L'integrazione dei dati di Amazon Q AWS Glue risponde con step-by-step istruzioni e include riferimenti alle sue fonti di informazioni.

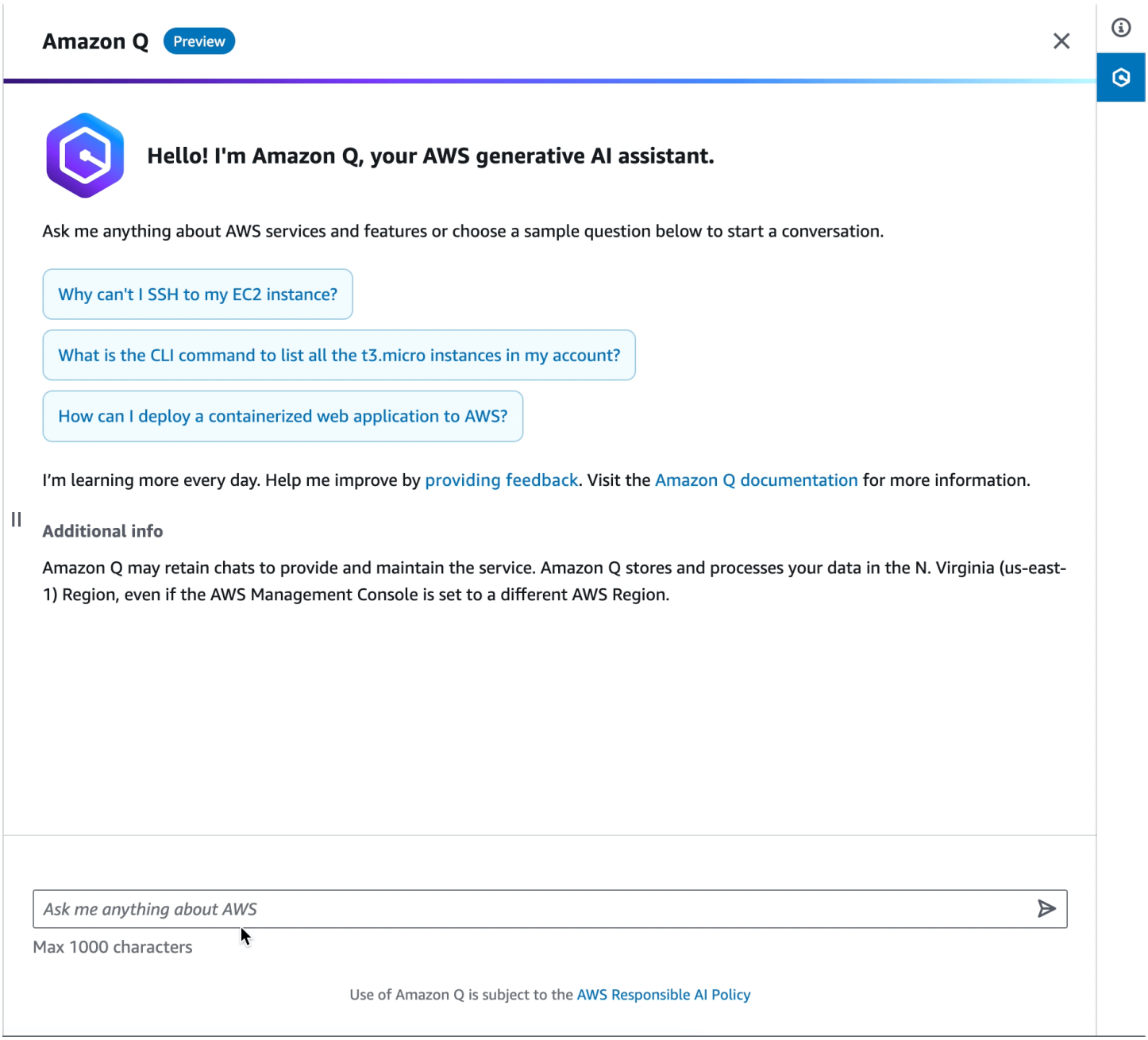
- Generazione di codice di integrazione dei dati: Amazon Q Data Integration in AWS Glue può rispondere a domande sugli script AWS Glue ETL e generare nuovo codice in base a una domanda in linguaggio naturale in inglese.
- Risoluzione dei problemi: l'integrazione dei dati di Amazon Q AWS Glue è stata creata appositamente per aiutarti a comprendere gli errori nei AWS Glue lavori e fornisce step-by-step istruzioni per la causa principale e la risoluzione dei problemi.

> **Note**
>
> L'integrazione dei dati di Amazon Q in AWS Glue non utilizza il contesto della conversazione per fornire informazioni sulle risposte future per tutta la durata della conversazione. Ogni conversazione con Amazon Q Data Integration in AWS Glue è indipendente dalle conversazioni precedenti o future.

Utilizzi l'integrazione dei dati di Amazon Q in AWS Glue?

Nel pannello Amazon Q puoi richiedere ad Amazon Q di generare codice per uno script AWS Glue ETL o rispondere a una domanda sulle AWS Glue funzionalità o risolvere un errore. La risposta è uno script ETL PySpark con step-by-step istruzioni per personalizzare lo script, esaminarlo ed eseguirlo. Per le domande, la risposta viene generata sulla base della knowledge base sull'integrazione dei dati con un riepilogo e un URL di origine per i riferimenti.

Ad esempio, puoi chiedere ad Amazon Q di "scrivere uno AWS Glue script che legga i dati CSV da S3, applicare la DropNullFields trasformazione e scrivere su Redshift" e, in risposta, l'integrazione dei dati di Amazon Q AWS Glue restituirà uno script di AWS Glue lavoro in grado di eseguire l'azione richiesta. Puoi esaminare il codice generato per assicurarti che soddisfi l'intento richiesto. Se sei soddisfatto, puoi implementarlo come processo di produzione. AWS Glue È possibile risolvere i problemi relativi ai processi chiedendo all'integrazione di spiegare gli errori e di proporre soluzioni. Amazon Q può rispondere a domande sulle nostre AWS Glue best practice di integrazione dei dati.



Di seguito sono riportati alcuni esempi di domande che dimostrano come l'integrazione dei dati di Amazon Q in AWS Glue può aiutarti a sviluppare AWS Glue:

AWS Glue Generazione di codice ETL:

- Scrivi uno AWS Glue script che legga JSON da S3, trasformi i campi utilizzando Apply Mapping e scriva su Amazon Redshift
- Come posso scrivere uno AWS Glue script per leggere da DynamoDB, applicare DropNullFields la trasformazione e scrivere su S3 come Parquet?

- Dammi uno AWS Glue script che legga da MySQL, rilasci alcuni campi in base alla mia logica aziendale e scriva su Snowflake
- Scrivi un AWS Glue lavoro da leggere da DynamoDB e scrivi su S3 come JSON
- Aiutami a sviluppare uno AWS Glue script per AWS Glue Data Catalog to S3
- Scrivi un AWS Glue lavoro per leggere JSON da S3, elimina i valori nulli e scrivi su Redshift

AWS Glue spiegazioni delle funzionalità:

- Come posso usare AWS Glue Data Quality?
- Come usare i segnalibri di AWS Glue lavoro?
- Come posso abilitare la scalabilità AWS Glue automatica?
- Qual è la differenza tra frame AWS Glue dinamici e frame di dati Spark?
- Quali sono i diversi tipi di connessioni supportati AWS Glue?

AWS Glue risoluzione dei problemi:

- Come risolvere gli errori di memoria esaurita (OOM) nei AWS Glue job?
- Quali sono alcuni messaggi di errore che potresti visualizzare durante la configurazione della qualità AWS Glue dei dati e come puoi risolverli?
- Come posso correggere un AWS Glue lavoro con l'errore Accesso negato ad Amazon S3?
- Come posso risolvere i problemi relativi allo shuffle dei dati nei lavori? AWS Glue

Configurazione dell'integrazione dei dati di Amazon Q in AWS Glue

L'integrazione dei dati di Amazon Q in AWS Glue è in versione di anteprima ed è soggetta a modifiche.

Nelle sezioni seguenti vengono fornite informazioni sulla configurazione dell'integrazione dei dati di Amazon Q in AWS Glue.

Argomenti

- Configurazione delle autorizzazioni IAM

Configurazione delle autorizzazioni IAM

L'integrazione dei dati di Amazon Q in AWS Glue è in versione di anteprima ed è soggetta a modifiche.

La concessione delle autorizzazioni alle API utilizzate dall'integrazione dei dati di Amazon Q AWS Glue richiede le autorizzazioni IAM (AWSIdentity and Access Management) appropriate. Puoi ottenere le autorizzazioni allegando la seguente AWS policy personalizzata alla tua identità IAM (ad esempio un utente, un ruolo o un gruppo):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartCompletion",
                "glue:GetCompletion"
            ],
            "Resource": [
                "arn:aws:glue:*:*:completion/*"
            ]
        }
    ]
}
```

> **Note**
>
> L'integrazione dei dati di Amazon Q in AWS Glue non dispone di API disponibili tramite l'SDK AWS da poter utilizzare in modo programmatico. Le seguenti due API vengono utilizzate nella policy IAM per abilitare questa esperienza tramite il pannello di chat di Amazon Q: `StartCompletion` e`GetCompletion`.

Assegnare le autorizzazioni

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

- Utenti e gruppi in AWS IAM Identity Center: crea un set di autorizzazioni. Segui le istruzioni contenute in Creare 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 Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.
- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di 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 di IAM.

Esempi

L'integrazione dei dati di Amazon Q in AWS Glue è in versione di anteprima ed è soggetta a modifiche.

Nelle sezioni seguenti vengono fornite informazioni sugli esempi di integrazione dei dati di Amazon Q in AWS Glue.

Argomenti

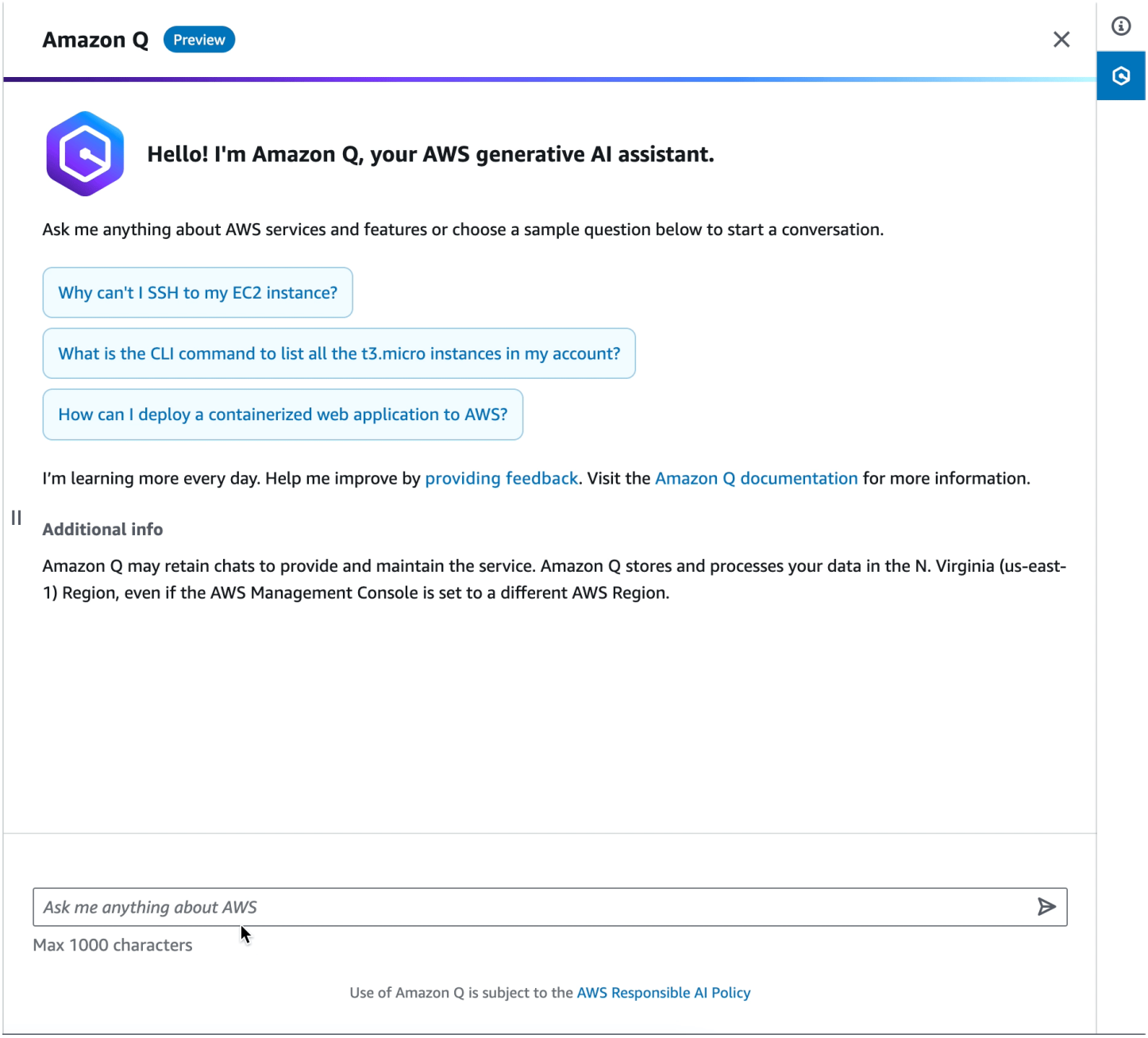
Interazioni di esempio

L'integrazione dei dati di Amazon Q in AWS Glue è in versione di anteprima ed è soggetta a modifiche.

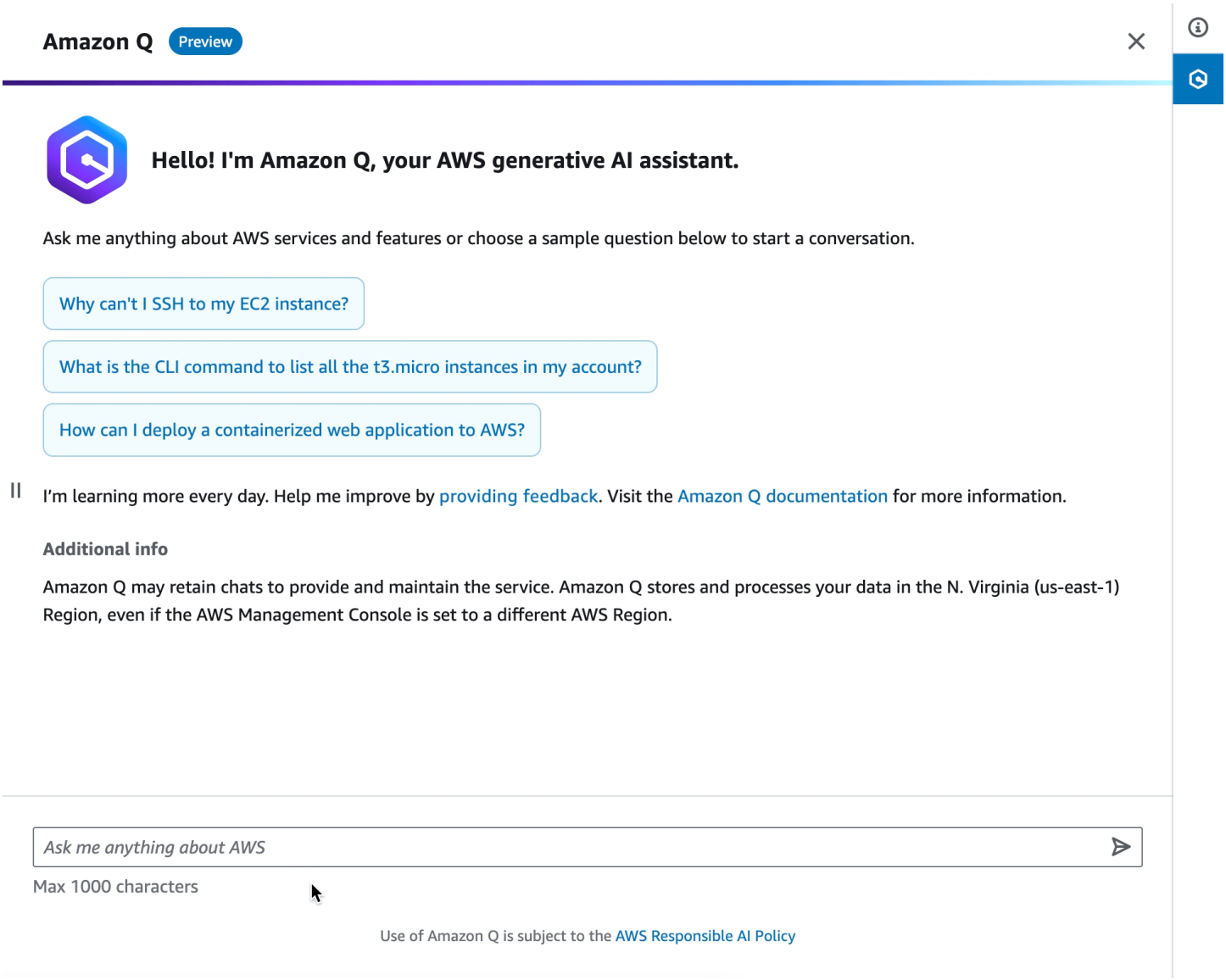
L'integrazione dei dati di Amazon Q ti AWS Glue consente di inserire la tua interrogazione/richiesta nel pannello Amazon Q. Puoi inserire una domanda sulla funzionalità di integrazione dei dati fornita da. AWS Glue Verrà restituita una risposta dettagliata, insieme ai documenti di riferimento.

Un altro caso d'uso è la generazione di script di processo ETL di AWS Glue. Puoi porre una domanda su come eseguire un processo di estrazione, trasformazione, caricamento dei dati. Verrà restituito PySpark uno script generato.

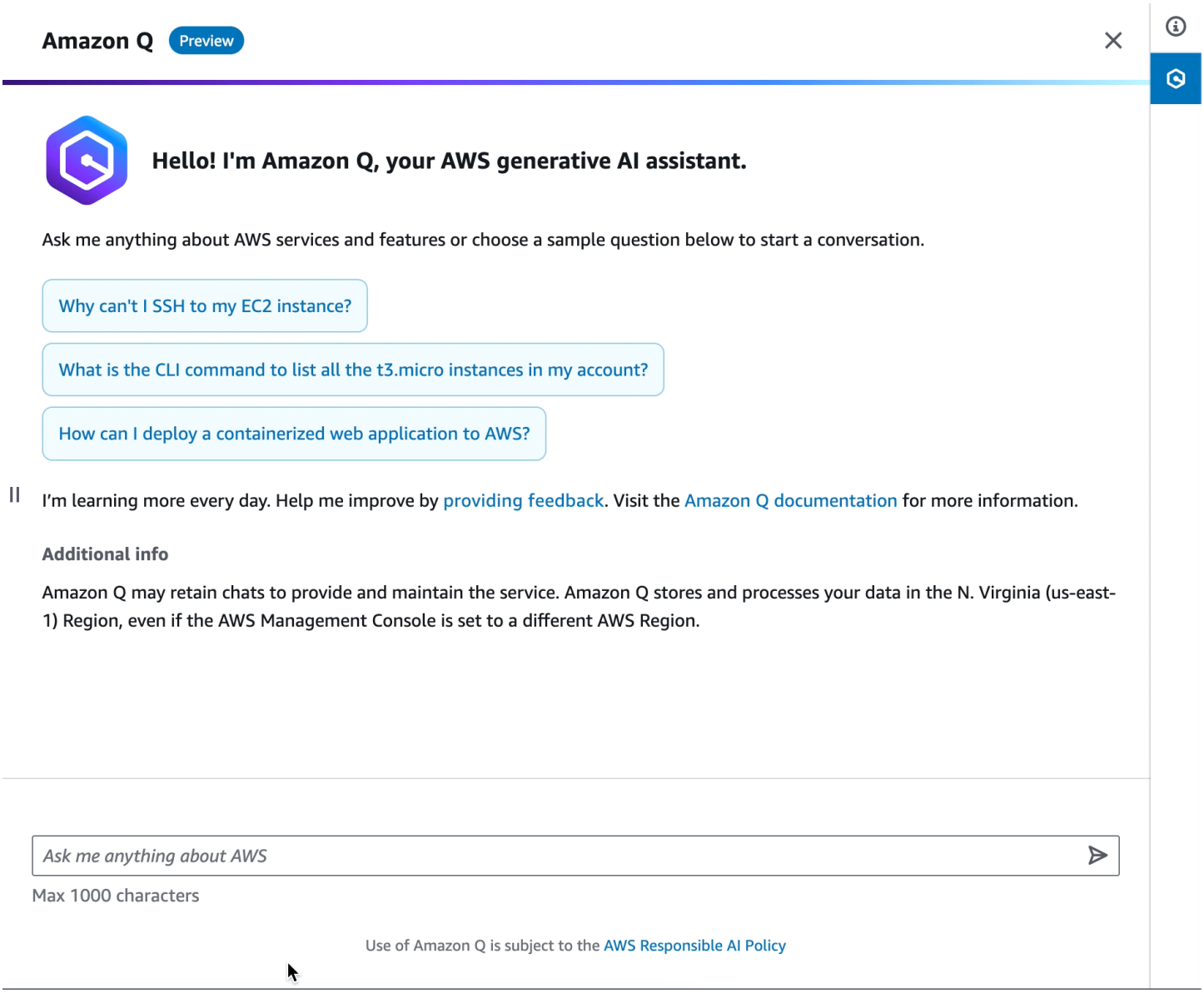
Richiesta di uno script ETL generato: scrivi uno script AWS Glue che legga JSON da S3, rimappi le colonne e scriva su Amazon Redshift.



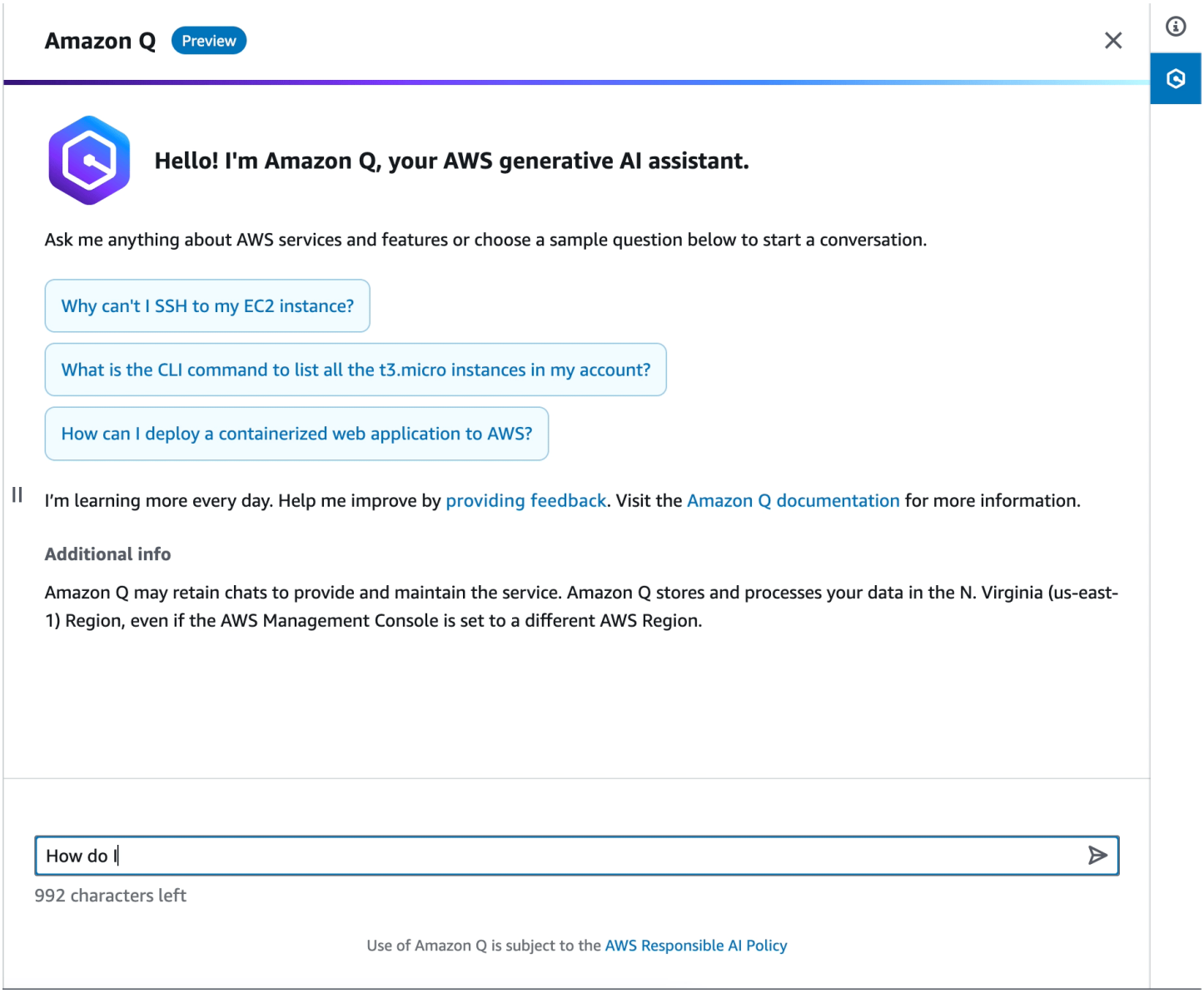
Un altro esempio di richiesta di uno script ETL generato: come posso scrivere uno AWS Glue script per leggere da DynamoDB, applicare DropNullFields la trasformazione e scrivere su S3 come Parquet? .



Chiedere una spiegazione per una AWS Glue funzionalità: come si utilizza la qualità AWS Glue dei dati?



Chiedere come risolvere un errore riscontrato in un AWS Glue processo: come posso correggere un AWS Glue lavoro con l'errore Accesso negato ad Amazon S3?



Considerazioni

Considera i seguenti elementi prima di utilizzare l'integrazione dei dati di Amazon Q in AWS Glue:

- Attualmente, la generazione di codice funziona solo con il kernel. PySpark Il codice generato è per i processi AWS Glue basati su Python Spark.
- Di seguito sono elencate le combinazioni delle capacità di generazione di codice dell'integrazione dei dati di Amazon Q in AWS Glue. Prevediamo di aggiungere in modo iterativo il supporto per nuove origini, trasformazioni e target in futuro.

Origine	Trasformazione	Target
S3 con i seguenti tipi di formato: json, csv, parquet, hudi, delta	ApplyMapping	S3 con i seguenti tipi di formato: json, csv, avro, orc, parquet, hudi, delta
Catalogo dati Glue	DropNullFields	Catalogo dati Glue
Amazon Redshift	DropFields	Amazon Redshift
MySQL	SelectFields	MySQL
Postgres	ResolveChoice	Postgres
Oracle	Filtro	Oracle
SQL Server	RenameFields	SQL Server
DynamoDB		DynamoDB
Snowflake		Snowflake

Orchestrazione in AWS Glue

Nelle sezioni seguenti vengono fornite informazioni sull'orchestrazione dei processi in AWS Glue.

Argomenti

Avvio di lavori e crawler utilizzando i trigger

In AWS Glue, è possibile creare oggetti del catalogo dati chiamati trigger, che è possibile utilizzare per avviare manualmente o automaticamente uno o più crawler o estrarre, trasformare e caricare lavori (ETL). Utilizzando i trigger, è possibile progettare una catena di crawler e lavori dipendenti.

> **Note**
>
> È possibile eseguire la stessa procedura definendo i flussi di lavoro. I flussi di lavoro sono i preferiti nella creazione di complesse operazioni ETL multi-processo. Per ulteriori informazioni, consulta the section called "Esecuzione di attività ETL complesse utilizzando gli schemi e i flussi di lavoro".

Argomenti

Trigger di AWS Glue

Quando viene attivato, un trigger può avviare processi e crawler specificati. Un trigger viene attivato on demand, in base a una pianificazione o in base a una combinazione di eventi.

> **Note**
>
> Solo due crawler possono essere attivati da un singolo trigger. Se vuoi eseguire il crawling di più datastore, utilizza più fonti per ogni crawler anziché eseguire più crawler contemporaneamente.

Esistono diversi stati di trigger. Un trigger è `CREATED`, `ACTIVATED` o `DEACTIVATED`. Esistono anche stati transitori, come ad esempio `ACTIVATING`. Per interrompere temporaneamente l'attivazione di un trigger, è possibile disattivarlo. È quindi possibile riattivarlo in un secondo momento.

Esistono tre tipi di trigger:

Pianificati

Un trigger basato sul tempo in `cron`.

È possibile creare un trigger per un set di lavori o crawler in base a una pianificazione. È possibile specificare i vincoli, ad esempio la frequenza in cui vengono eseguiti i lavori o i crawler, i giorni della settimana in cui vengono eseguiti e a che ora. Questi vincoli si basano sul comando `cron`. Quando si configura una pianificazione per un trigger, tenere conto delle caratteristiche e delle limitazioni del cron. Ad esempio, se vuoi eseguire il crawler il giorno 31 di ogni mese, devi ricordare che alcuni mesi non sono di 31 giorni. Per ulteriori informazioni sul cron, consulta Pianificazioni basate sul tempo per processi e crawler.

Condizionale

Un trigger che viene attivato quando un lavoro o crawler o più lavori o crawler precedenti soddisfano un elenco di condizioni.

Quando si crea un trigger condizionale, si specificano un elenco di lavori e un elenco di crawler da controllare. Per ogni lavoro o crawler controllato, è necessario specificare uno stato da controllare, ad esempio riuscito, non riuscito, timeout e così via. Il trigger viene attivato se i lavori o i crawler controllati terminano con gli stati specificati. È possibile configurare il trigger per l'attivazione quando si verificano uno o tutti gli eventi osservati.

Ad esempio, è possibile configurare un trigger T1 per avviare il lavoro J3 quando entrambi i lavori J1 e J2 vengono completati correttamente e un altro trigger T2 per avviare il lavoro J4 se il lavoro J1 o J2 non riesce.

Nella tabella seguente sono elencati gli stati di completamento del lavoro e del crawler (eventi) controllati dai trigger.

Stati di completamento del lavoro	Stati di completamento del crawler
• `SUCCEEDED` • `STOPPED` • `FAILED` • `TIMEOUT`	• `SUCCEEDED` • `FAILED` • `CANCELLED`

On demand

Un trigger che si attiva quando viene acceso. I trigger su richiesta non entrano mai nello stato `ACTIVATED` o `DEACTIVATED`. Rimangono sempre nello stato `CREATED`.

Affinché si attivino non appena creati, è possibile impostare un flag per attivare i trigger pianificati e condizionali al momento della creazione.

> **Important**
>
> I lavori o i crawler eseguiti dopo il completamento di altri processi o crawler vengono definiti dipendenti. I lavori o i crawler dipendenti vengono avviati solo se il lavoro o il crawler completato è stato avviato da un trigger. Tutti i lavori o i crawler in una catena di dipendenze devono discendere da una singola pianificazione o da un singolo trigger on-demand.

Passare i parametri del lavoro con i trigger

Un trigger può passare parametri ai lavori che avvia. I parametri includono argomenti del lavoro, valore di timeout, configurazione di sicurezza e altro ancora. Se il trigger avvia più lavori, i parametri vengono passati a ciascun lavoro.

Di seguito sono riportate le regole per argomenti di lavoro passati da un trigger:

- Se la chiave nella coppia chiave-valore corrisponde a un argomento di lavoro predefinito, l'argomento passato sostituisce l'argomento predefinito. Se la chiave non corrisponde a un argomento predefinito, l'argomento viene passato come argomento aggiuntivo al lavoro.

- Se la chiave nella coppia chiave-valore corrisponde a un argomento non sovrascrivibile, l'argomento passato viene ignorato.

Per ulteriori informazioni, consulta the section called “Trigger” nell'API AWS Glue.

Aggiunta di trigger

È possibile aggiungere un trigger utilizzando la console AWS Glue, l'AWS Command Line Interface (AWS CLI) o l'API AWS Glue.

> **Note**
>
> Attualmente, la console AWS Glue supporta solo i lavori, non i crawler, quando lavora con i trigger. È possibile utilizzare l'API AWS Glue o l'AWS CLI per configurare i trigger con lavori e crawler.

Per aggiungere un trigger (console)

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione, in ETL, scegliere Triggers (Trigger). Selezionare Add trigger (Aggiungi trigger).
3. Specificare le proprietà seguenti:

   Nome

   Assegna al trigger un nome univoco.

   Tipo di trigger

   Specifica una delle seguenti proprietà:

   - Schedule (Pianifica): il trigger si attiva a una frequenza e un tempo specifici.
   - Job events (Eventi di lavoro): un trigger condizionale. Il trigger viene attivato quando uno o tutti i lavori nell'elenco corrispondono agli stati designati. Per consentire l'attivazione del trigger, il lavoro in questione deve essere stato avviato da un trigger. Per qualsiasi lavoro selezionato, è possibile osservare un solo evento (stato di completamento).
   - On-demand (On demand): il trigger funziona se attivato.

4. Completare la procedura guidata del trigger. Nella pagina Review (Revisione) è possibile attivare immediatamente i trigger Schedule (Pianifica) e Job events (Eventi di lavoro) (condizionali), selezionando Enable trigger on creation (Attiva trigger alla creazione).

Per aggiungere un trigger (AWS CLI)

- Utilizzare un comando simile al seguente:

```
aws glue create-trigger --name MyTrigger --type SCHEDULED --schedule  "cron(0 12 *
 * ? *)" --actions CrawlerName=MyCrawler --start-on-creation
```

Questo comando crea un trigger di pianificazione denominato `MyTrigger`, che viene eseguito ogni giorno alle 12:00 UTC e avvia un crawler denominato `MyCrawler`. Il trigger viene creato nello stato attivato.

Per ulteriori informazioni, consulta the section called “Trigger di AWS Glue”.

Pianificazioni basate sul tempo per processi e crawler

Puoi definire una pianificazione basata sul tempo per i crawler e i processi in AWS Glue. La definizione di queste pianificazioni usa la sintassi cron di tipo Unix. Specifichi il tempo in Coordinated Universal Time (UTC) e la precisione minima per una pianificazione è 5 minuti.

Per ulteriori informazioni sulla configurazione di processi e crawler da eseguire utilizzando una pianificazione, consulta Avvio di lavori e crawler utilizzando i trigger.

Espressioni cron

Le espressioni cron hanno sei campi obbligatori che sono separati da uno spazio vuoto.

Sintassi

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

Campi	Valori	Caratteri jolly
Minuti	0-59	, - * /

Campi	Valori	Caratteri jolly
Ore	0-23	, - * /
Day-of-month (Giorno del mese)	1-31	, - * ? / L W
Mese	1-12 o JAN-DEC	, - * /
Day-of-week (Giorno della settimana)	1-7 o SUN-SAT	, - * ? / L
Anno	1970–2199	, - * /

Caratteri jolly

- Il carattere jolly , (virgola) include valori aggiuntivi. Nel campo `Month`, `JAN,FEB,MAR` includono gennaio, febbraio e marzo.
- Il carattere jolly - (trattino) specifica gli intervalli. Nel campo `Day`, 1-15 include i giorni dall'1 al 15 del mese specificato.
- Il carattere jolly * (asterisco) include tutti i valori nel campo. Nel campo `Hours`, * include ogni ora.
- Il carattere jolly / (barra) specifica gli incrementi. Nel campo `Minutes` puoi immettere **`1/10`** per specificare ogni decimo minuto, a partire dal primo minuto dell'ora (ad esempio, l'11°, il 21° e il 31° minuto).
- Il carattere jolly ? (punto interrogativo) specifica un valore. Nel campo `Day-of-month` puoi immettere 7 e, se è indifferente quale sia il settimo giorno della settimana, puoi specificare ?.
- Il carattere jolly L nel campo `Day-of-month` o `Day-of-week` specifica l'ultimo giorno del mese o della settimana.
- Il carattere jolly W nel campo `Day-of-month` specifica un giorno feriale. Nel campo `Day-of-month`, `3W` specifica il giorno più vicino al terzo giorno feriale del mese.

Limiti

- Non puoi specificare i campi `Day-of-month` e `Day-of-week` nella stessa espressione cron. Se specifichi un valore in uno dei campi, devi usare un carattere ? nell'altro campo.
- Le espressioni cron che indicano frequenze più rapide di 5 minuti non sono supportate.

Esempi

Quando crei una pianificazione puoi utilizzare le seguenti stringhe cron di esempio.

Minuti	Ore	Giorno del mese	Mese	Giorno della settimana	Anno	Significato
0	10	*	*	?	*	Esegui ogni giorno alle 10:00 (UTC)
15	12	*	*	?	*	Esegui ogni giorno alle 12:15 (UTC)
0	18	?	*	LUN-VEN	*	Esegui dal lunedì al venerdì alle 18:00 (UTC)
0	8	1	*	?	*	Esegui ogni primo giorno del mese alle 8.00 (UTC)
0/15	*	*	*	?	*	Esegui ogni 15 minuti
0/10	*	?	*	LUN-VEN	*	Esegui dal lunedì al venerdì

Minuti	Ore	Giorno del mese	Mese	Giorno della settimana	Anno	Significato
						ogni 10 minuti
0/5	8-17	?	*	LUN-VEN	*	Esegui dal lunedì al venerdì dalle 8:00 alle 17:55 (UTC) ogni 5 minuti

Ad esempio, per eseguire una pianificazione ogni giorno alle 12:15 UTC, specifica:

```
cron(15 12 * * ? *)
```

Attivazione e disattivazione dei trigger

È possibile attivare o disattivare un trigger utilizzando la console AWS Glue, l'AWS Command Line Interface (AWS CLI) o l'API AWS Glue.

> **Note**
>
> Attualmente, la console AWS Glue supporta solo i lavori, non i crawler, quando lavora con i trigger. È possibile utilizzare l'API AWS Glue o l'AWS CLI per configurare i trigger con lavori e crawler.

Per attivare o disattivare un trigger (console)

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione, in ETL, scegliere Triggers (Trigger).

3. Selezionare la casella di controllo accanto al trigger desiderato e nel menu Action (Operazione) scegliere Enable trigger (Abilita trigger) per attivare il trigger o Disable trigger (Disattiva trigger) per disattivare il trigger.

Per attivare o disattivare un trigger (AWS CLI)

- Inserisci uno dei comandi seguenti.

```
aws glue start-trigger --name MyTrigger

aws glue stop-trigger --name MyTrigger
```

  Un trigger viene attivato con l'avvio e viene disattivato con l'arresto. Quando un trigger si attiva on demand, il funzionamento è immediato.

Per ulteriori informazioni, consulta the section called “Trigger di AWS Glue”.

Esecuzione di attività ETL complesse utilizzando gli schemi e i flussi di lavoro in AWS Glue

Alcuni dei processi di estrazione, trasformazione e caricamento (ETL) della tua organizzazione potrebbero essere implementati utilizzando più processi e crawler dipendenti AWS Glue. Con i flussi di lavoro di AWS Glue, puoi progettare una complessa operazione ETL multi-processo e multi-crawler che AWS Glue può eseguire e tracciare come singola entità. Dopo aver creato un flusso di lavoro e averne specificato i processi, i crawler e i trigger, puoi eseguire il flusso di lavoro on demand o in base a una pianificazione.

Argomenti

- Recupero e impostazione delle proprietà di esecuzione del flusso di lavoro in AWS Glue
- Eseguire query sui flussi di lavoro utilizzando AWS Glue API
- Restrizioni dei flussi di lavoro e degli schemi in AWS Glue
- Risoluzione degli errori relativi agli schemi in AWS Glue
- Autorizzazioni per utenti e ruoli per gli schemi AWS Glue

Panoramica di flussi di lavoro in AWS Glue

In AWS Glue, è possibile utilizzare i flussi di lavoro per creare e visualizzare complesse attività di estrazione, trasformazione e caricamento (ETL) che coinvolgono più crawler, processi e trigger. Ogni flusso di lavoro gestisce l'esecuzione e il monitoraggio di tutti i suoi processi e crawler. Poiché un flusso di lavoro esegue ogni componente, registra l'avanzamento e lo stato di esecuzione. In questo modo viene fornita una panoramica dell'attività complessiva e i dettagli di ciascuna fase. La console di AWS Glue offre una rappresentazione visiva di un flusso di lavoro sotto forma di diagramma.

È possibile creare un flusso di lavoro da un piano AWS Glue oppure creare manualmente un flusso di lavoro aggiungendo un componente alla volta, utilizzando la AWS Management Console o AWS Glue API. Per ulteriori informazioni sui piani, consulta the section called “Panoramica degli schemi”.

I trigger all'interno dei flussi di lavoro possono attivare sia processi che crawler e possono attivarsi quando i processi o i crawler vengono completati. Utilizzando i trigger è possibile creare grandi catene di processi e crawler interdipendenti. Oltre ai trigger all'interno di un flusso di lavoro che definiscono le dipendenze dei processi e dei crawler, ogni flusso di lavoro dispone di un trigger di avvio. Esistono tre tipi di trigger di avvio:

- Pianificazione: il flusso di lavoro viene avviato secondo una pianificazione definita. La pianificazione può essere giornaliera, settimanale, mensile e così via oppure può essere una personalizzata in base a un'espressione `cron`.
- On demand: il flusso di lavoro viene avviato manualmente dalla console AWS Glue, dall'API, o da AWS CLI.
- Evento EventBridge: il flusso di lavoro viene avviato quando si verifica un singolo evento Amazon EventBridge o un batch di eventi Amazon EventBridge. Con questo tipo di trigger, AWS Glue può essere un consumer di eventi in un'architettura basata su eventi. Qualsiasi tipo di evento EventBridge può avviare un flusso di lavoro. Un caso d'uso ìcomune è l'arrivo di un nuovo oggetto in un bucket Amazon S3 (l'operazione `PutObject` di S3).

Avviare un flusso di lavoro con un batch di eventi significa attendere fino a quando non è stato ricevuto un numero specificato di eventi o fino a quando non è trascorso un determinato periodo di tempo. Quando si crea il trigger di evento EventBridge, è possibile specificare facoltativamente le condizioni di batch. Se si specificano le condizioni del batch, è necessario specificare la dimensione del batch (numero di eventi) e, facoltativamente, è possibile specificare una finestra batch (numero di secondi). La dimensione massima di default della finestra è di 900 secondi (15 minuti). La condizione batch che viene soddisfatta per prima avvia il flusso di lavoro. La finestra batch si avvia all'arrivo del primo evento. Se durante la creazione di un trigger non si specificano le condizioni di batch, la dimensione del batch viene impostata automaticamente su 1.

All'avvio del flusso di lavoro, le condizioni di batch vengono reimpostate e il trigger di evento inizia a monitorare la condizione di batch successiva da soddisfare per avviare nuovamente il flusso di lavoro.

Nella tabella seguente viene illustrato il modo in cui le dimensioni batch e la finestra batch operano insieme per attivare un flusso di lavoro.

Dimensione batch	Finestra batch	Condizione di attivazione risultante
10		Il flusso di lavoro viene attivato all'arrivo di 10 eventi EventBridge o 15 minuti dopo l'arrivo del primo evento, a seconda di quale evento si verifica per primo. (Se la dimensione della finestra non viene specificata, il valore predefinito è 15 minuti.)
10	2 minuti	Il flusso di lavoro viene attivato all'arrivo di 10 eventi EventBridge o 2 minuti dopo l'arrivo del primo evento, a seconda di quale evento si verifica per primo.
1		Il flusso di lavoro viene attivato all'arrivo del primo evento. La dimensione della finestra è irrilevante. Se durante la creazione di del trigger di evento EventBridge non si specificano le condizioni di batch, la dimensione del batch viene impostata automaticamente su 1.

L'operazione API `GetWorkflowRun` restituisce la condizione batch che ha attivato il flusso di lavoro.

Indipendentemente dalla modalità di avvio di un flusso di lavoro, è possibile specificare il numero massimo di esecuzioni simultanee durante la creazione del flusso di lavoro.

Se un evento o un batch di eventi avvia un'esecuzione del flusso di lavoro che alla fine ha esito negativo, tale evento o batch di eventi non viene più considerato per l'avvio di un'esecuzione del flusso di lavoro. Un nuovo flusso di lavoro viene avviato solo quando arriva l'evento o il batch di eventi successivo.

> **Important**
>
> Limita il numero totale di processi, crawler e attivazioni all'interno di un flusso di lavoro a 100 o meno. Se includi più di 100, potresti riscontrare errori durante il tentativo di riprendere o interrompere l'esecuzione del flusso di lavoro.

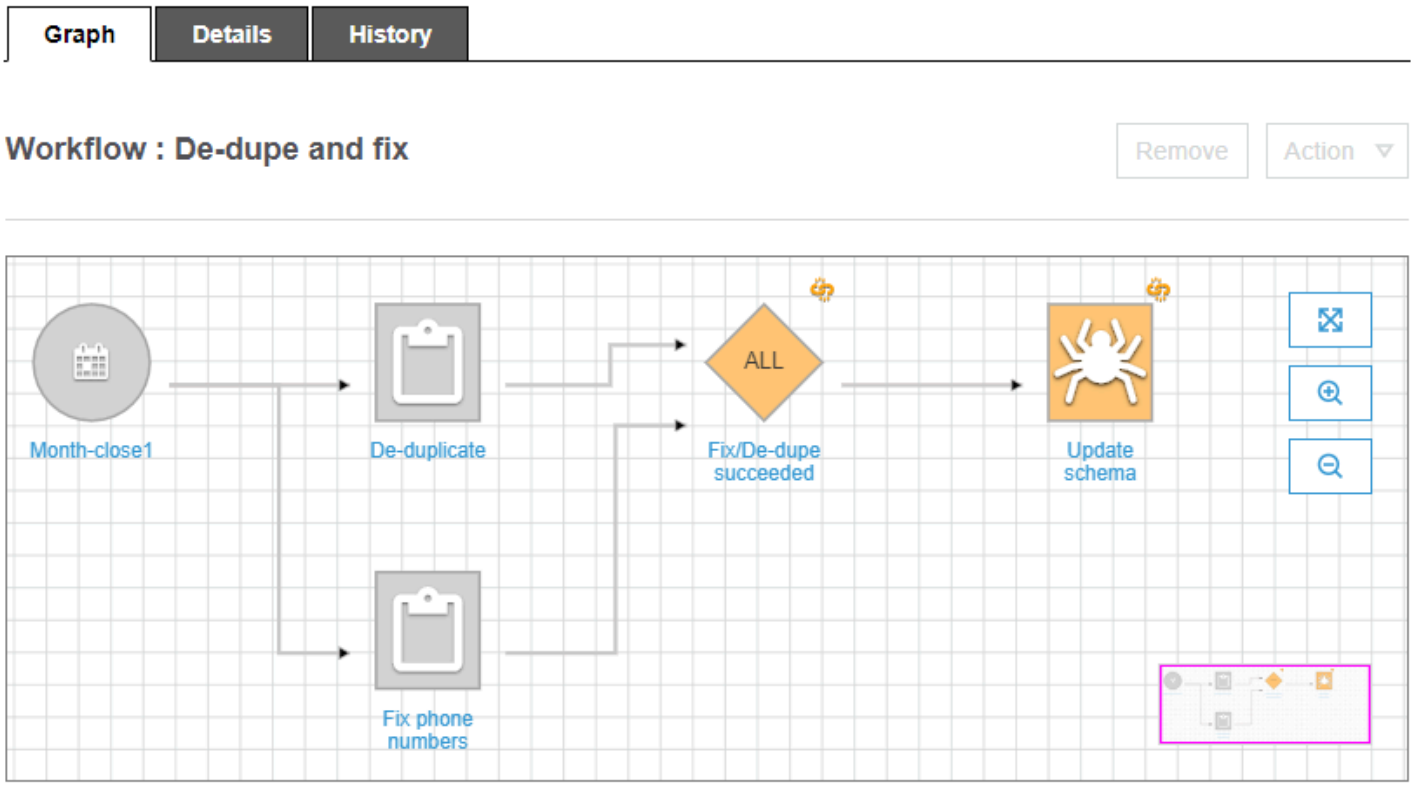
Un'esecuzione del flusso di lavoro non verrà avviata se supererà il limite di concorrenza impostato per il flusso di lavoro, anche se la condizione dell'evento è soddisfatta. È consigliabile modificare i limiti di simultaneità del flusso di lavoro in base al volume degli eventi previsti. AWS Glue non ritenta le esecuzioni del flusso di lavoro che non riescono a causa di limiti di simultaneità superati. Allo stesso modo, è consigliabile modificare i limiti di simultaneità per i processi e i crawler all'interno dei flussi di lavoro in base al volume degli eventi previsto.

Proprietà esecuzione flusso di lavoro

Per condividere e gestire lo stato di un flusso di lavoro in esecuzione, è possibile definire le proprietà dell'esecuzione di flussi di lavoro di default. Queste proprietà, che sono coppie nome/valore, sono disponibili per tutti i processi del flusso di lavoro. Utilizzando la AWS Glue API, i processi possono recuperare le proprietà di esecuzione e modificarle per i processi successivi nel flusso di lavoro.

Grafico del flusso di lavoro

L'immagine seguente mostra il grafico di un flusso di lavoro basato sulla console di AWS Glue. Un flusso di lavoro potrebbe essere composto da dozzine di componenti.



Questo flusso di lavoro viene avviato da un trigger di pianificazione, `Month-close1`, che avvia due processi, `De-duplicate` e `Fix phone numbers`. Al corretto completamento di entrambi i processi, un trigger di evento, `Fix/De-dupe succeeded`, avvia un crawler, `Update schema`.

Visualizzazioni del flusso di lavoro statica e dinamica

Per ogni flusso di lavoro, esiste il concetto di visualizzazione statica e visualizzazione dinamica. La visualizzazione statica descrive la struttura del flusso di lavoro. La visualizzazione dinamica è una visualizzazione in fase di runtime che include le informazioni sull'ultima esecuzione di ognuno dei processi e dei crawler. Le informazioni sull'esecuzione includono l'esito finale e i dettagli degli errori.

Quando un flusso di lavoro è in esecuzione, la console mostra la visualizzazione dinamica, che indica graficamente i processi che si sono conclusi e quelli che devono ancora essere eseguiti. È anche possibile recuperare una visualizzazione dinamica di un flusso di lavoro in esecuzione utilizzando la AWS Glue API. Per ulteriori informazioni, consulta Eseguire query sui flussi di lavoro utilizzando AWS Glue API.

Consulta anche

- the section called “Creazione di un flusso di lavoro da uno schema”

- the section called “Creazione e costruzione manuale di un flusso di lavoro”
- Flussi di lavoro (per l'API dei flussi di lavoro)

Creazione e costruzione manuale di un flusso di lavoro in AWS Glue

Puoi utilizzare la console AWS Glue per creare e costruire manualmente un flusso di lavoro, un nodo alla volta.

Un flusso di lavoro contiene processi, crawler e trigger. Prima di creare un flusso di lavoro manualmente, è necessario creare i processi e i crawler che devono essere inclusi nel flusso di lavoro. Per i flussi di lavoro è consigliabile specificare dei crawler di tipo run-on demand. È possibile creare nuovi trigger durante la creazione del flusso di lavoro oppure è possibile clonare i trigger già esistenti nel flusso di lavoro. Quando si clona un trigger, tutti gli oggetti catalogo associati al trigger, ovvero i processi o i crawler che lo attivano e i processi o crawler che avvia, vengono aggiunti al flusso di lavoro.

> **Important**
>
> Limita il numero totale di processi, crawler e attivazioni all'interno di un flusso di lavoro a 100 o meno. Se includi più di 100, potresti riscontrare errori durante il tentativo di riprendere o interrompere l'esecuzione del flusso di lavoro.

È possibile creare il proprio flusso di lavoro aggiungendo trigger al diagramma del flusso di lavoro e definendo gli eventi osservati e le operazioni di ogni trigger. Si inizia con un trigger di attivazione, che può essere un trigger on demand o pianificato, e si completa il diagramma aggiungendo trigger basati su evento (condizionali).

Fase 1: creazione del flusso di lavoro

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in ETL, scegliere Workflows (Flussi di lavoro).
3. Scegliere Add workflow (Aggiungi flusso di lavoro) e completare il modulo Add a new ETL workflow (Aggiungi un nuovo flusso di lavoro ETL).

Qualsiasi proprietà opzionale di default per l'esecuzione aggiuntiva viene resa disponibile come argomento a tutti i processi del flusso di lavoro. Per ulteriori informazioni, consulta Recupero e impostazione delle proprietà di esecuzione del flusso di lavoro in AWS Glue.

4. Scegliere Add workflow (Aggiungi flusso di lavoro).

   Il nuovo flusso di lavoro verrà visualizzato nell'elenco sulla pagina Workflows (Flussi di lavoro).

Fase 2: aggiunta di un trigger di attivazione

1. Nella pagina Workflows (Flussi di lavoro), selezionare il nuovo flusso di lavoro. Quindi, nella parte inferiore della pagina, assicurarsi che grafico sia selezionato.
2. Scegliere Add trigger (Aggiungi trigger) e nella finestra di dialogo Add trigger (Aggiungi trigger), procedere in uno dei seguenti modi:

   - Scegliere Clone existing (Clona esistente) e scegliere un trigger da clonare. Quindi scegliere Add (Aggiungi).

     Il trigger viene visualizzato sul diagramma, insieme ai processi e ai crawler che lo attivano e i processi o i crawler che lancia.

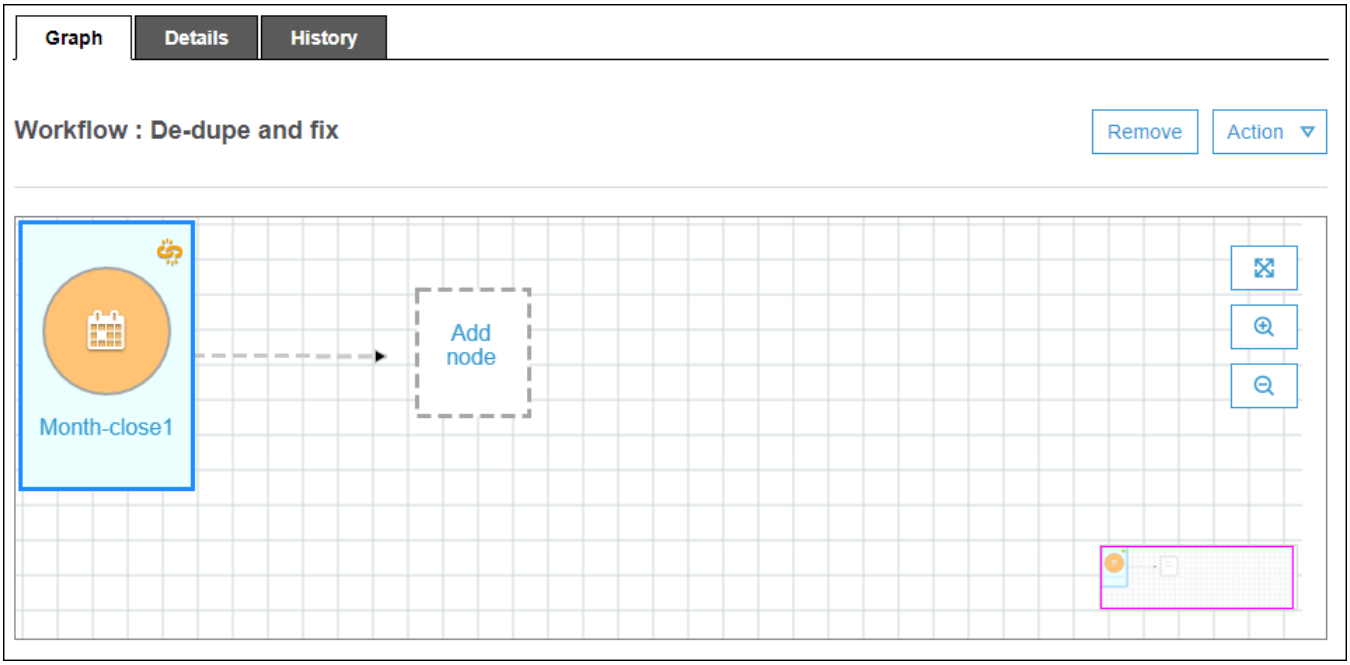
     Se è stato selezionato inavvertitamente il trigger sbagliato, selezionare il trigger nel diagramma e quindi scegliere Remove (Rimuovi).
   - Scegliere Add new (Aggiungi nuovo) e completare il modulo Add trigger (Aggiungi trigger).
     1. Per Trigger type (Tipo di trigger), selezionare Schedule (Pianificazione), On demand oppure EventBridge.

        Per il tipo di trigger Schedule (Pianificazione), scegliere una delle opzioni di Frequency (Frequenza). Scegliere Custom (Personalizza) per immettere un'espressione `cron`.

        Per il tipo di trigger EventBridge immettere il Number of events (Numero di eventi) (dimensione batch) e, facoltativamente, immettere Time delay (Ritardo) (finestra batch). Omettendo Time delay (Ritardo), la finestra batch viene impostata per impostazione predefinita su 15 minuti. Per ulteriori informazioni, consulta Panoramica di flussi di lavoro in AWS Glue.
     2. Scegli Add (Aggiungi).

Il trigger viene visualizzato sul diagramma, insieme a un nodo segnaposto nodo (etichettato Add node (Aggiungi nodo)). Nell'esempio seguente, il trigger di avvio è un trigger di pianificazione denominato `Month-close1`.

A questo punto, il trigger non è ancora salvato.



3. Se è stato aggiunto un nuovo trigger, completare i seguenti passaggi:

   a. Completa una delle seguenti operazioni:

      - Scegliere il nodo segnaposto (Add node (Aggiungi nodo)).
      - Verificare che il trigger di avvio sia selezionato e, dal menu Operazione sopra al diagramma, scegliere Add jobs/crawlers to trigger (Aggiungi processi/crawler al trigger).

   b. Nella finestra di dialogo Add jobs(s) and crawler(s) to trigger (Aggiungi processo/i e crawler al trigger) selezionare uno o più processi o crawler e quindi scegliere Add (Aggiungi).

      Il trigger viene salvato e i processi o crawler selezionati vengono visualizzati sul diagramma con connettori che hanno origine dal trigger.

      Se sono stati aggiunti inavvertitamente processi o crawler sbagliati, è possibile selezionare il trigger o un connettore e scegliere Remove (Rimuovi).

Fase 3: aggiunta di più trigger

Continuare a creare il flusso di lavoro aggiungendo ulteriori trigger di tipo Event (Evento). Per incrementare o ridurre il livello di dettaglio o per ingrandire l'area di disegno del diagramma, utilizzare le icone a destra. Per ogni trigger da aggiungere, completare i seguenti passaggi:

> **Note**
>
> Non viene eseguita alcuna azione per salvare il flusso di lavoro. Dopo aver aggiunto l'ultimo trigger e assegnato azioni al trigger, il flusso di lavoro è completato e salvato. È sempre possibile tornare successivamente e aggiungere altri nodi.

1. Completa una delle seguenti operazioni:

   - Per clonare un trigger esistente, accertarsi che non sia selezionato nessun nodo del diagramma e, nel menu Action (Operazione), scegliere Add trigger (Aggiungi trigger).
   - Per aggiungere un nuovo trigger che monitori un determinato processo o crawler del diagramma, selezionare il nodo del processo o del crawler e quindi scegliere il nodo segnaposto Add trigger (Aggiungi trigger).

     È possibile aggiungere ulteriori processi o crawler da far monitorare a questo trigger in un secondo momento.
2. Nella finestra di dialogo Add Folder (Aggiungi cartella), effettuare una delle operazioni indicate di seguito:

   - Scegliere Add new (Aggiungi nuovo) e completare il modulo Add trigger (Aggiungi trigger). Quindi scegliere Add (Aggiungi).

     Il trigger viene visualizzato sul diagramma. È possibile completare il trigger in un secondo momento.
   - Scegliere Clone existing (Clona esistente) e scegliere un trigger da clonare. Quindi scegliere Add (Aggiungi).

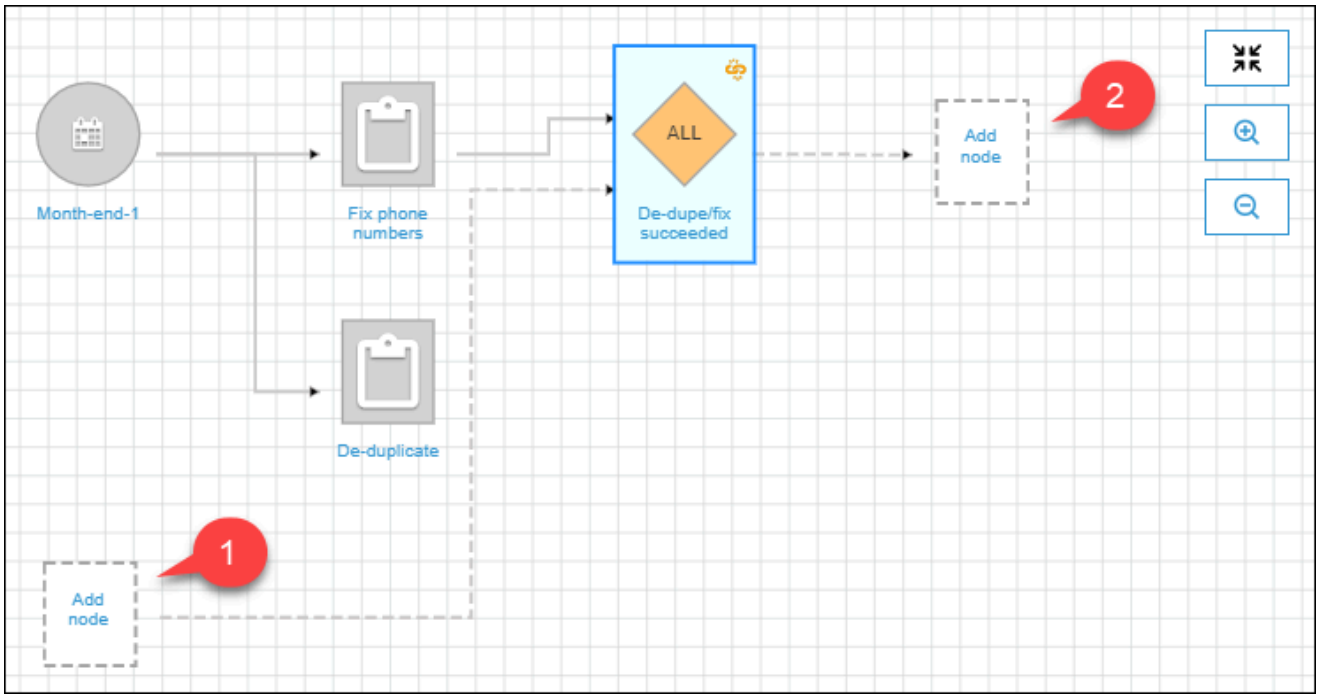
     Il trigger viene visualizzato sul diagramma, insieme ai processi e ai crawler che lo attivano e i processi o i crawler che lancia.

     Se è stato selezionato inavvertitamente il trigger sbagliato, selezionare il trigger nel diagramma e quindi scegliere Remove (Rimuovi).

3. Se è stato aggiunto un nuovo trigger, completare i seguenti passaggi:

   a. Selezionare il nuovo trigger.

      Come mostra il seguente diagramma, il trigger `De-dupe/fix succeeded` è selezionato e i nodi segnaposto appaiono per (1) eventi da monitorare e (2) operazioni.



   b. (Facoltativo se il trigger monitora già un evento e si desidera aggiungere ulteriori processi o crawler da monitorare) Scegliere il nodo segnaposto degli eventi da monitorare e, nella finestra di dialogo Add job(s) and crawler(s) to watch (Aggiungi processo/i e crawler da monitorare), selezionare uno o più processi o crawler. Scegli un evento da monitorare (SUCCEEDED, FAILED, ecc.) e scegliere Add (Aggiungi).

   c. Verificare che il trigger sia selezionato e scegliere il nodo segnaposto delle operazioni.

   d. Nella finestra di dialogo Add job(s) and crawler(s) to watch (Aggiungi processo/i e crawler da monitorare), selezionare uno o più processi o crawler e scegliere Add (Aggiungi).

      I processi e crawler selezionati vengono visualizzati sul diagramma con connettori che hanno origine dal trigger.

Per ulteriori informazioni sui flussi di lavoro e sui progetti, consulta i seguenti argomenti.

- Panoramica di flussi di lavoro in AWS Glue
- Esecuzione e monitoraggio di un flusso di lavoro in AWS Glue

- Creazione di un flusso di lavoro da uno schema in AWS Glue

Avvio di un flusso di lavoro AWS Glue con un evento Amazon EventBridge

Amazon EventBridge, conosciuto anche come CloudWatch Events, ti consente di automatizzare i servizi AWS e rispondere automaticamente a eventi di sistema, come i problemi relativi alla disponibilità delle applicazioni o le modifiche delle risorse. Gli eventi dei servizi AWS vengono recapitati a EventBridge quasi in tempo reale. 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 il supporto EventBridge, AWS Glue può servire come produttore e consumer di eventi in un'architettura basata sugli eventi. Per i flussi di lavoro, AWS Glue supporta qualsiasi tipo di evento EventBridge come consumer. Il caso d'uso più comune è l'arrivo di un nuovo oggetto in un bucket Amazon S3. Se si dispone di dati che arrivano a intervalli irregolari o non definiti, è possibile elaborare questi dati il più vicino possibile al loro arrivo.

> **Note**
>
> AWS Glue non fornisce la consegna garantita dei messaggi EventBridge. AWS Glue non esegue la deduplicazione se EventBridge fornisce messaggi duplicati. È necessario gestire l'idempotenza in base al proprio caso d'uso.
> Assicurati di configurare correttamente le regole EventBridge per evitare l'invio di eventi indesiderati.

Prima di iniziare

Se desideri avviare un flusso di lavoro con gli eventi di dati Amazon S3, devi assicurarti che gli eventi per il bucket S3 di interesse siano registrati su AWS CloudTrail ed EventBridge. A questo scopo, devi creare un persorso CloudTrail. Per ulteriori informazioni, consulta Creazione di un percorso per il tuo accountAWS.

Per avviare un flusso di lavoro con un evento EventBridge

> **Note**
>
> Nei comandi seguenti, sostituisci:

- *<workflow-name>* con il nome da assegnare al flusso di lavoro.
- *<trigger-name>* con il nome da assegnare al trigger.
- *<bucket-name>* con il nome del bucket Amazon S3.
- *<account-id>* con un ID dell'account AWS valido.
- *<region>* con il nome della regione (ad esempio `us-east-1`).
- *<rule-name>* con il nome da assegnare alla regola EventBridge.

1. Assicurati di disporre delle autorizzazioni AWS Identity and Access Management (IAM) per creare e visualizzare le regole e le destinazioni EventBridge. Di seguito è riportato una policy di esempio che è possibile allegare. Potresti ridurre il suo ambito per applicare restrizioni alle operazioni e alle risorse.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:DisableRule",
                "events:DeleteRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "events:EnableRule",
                "events:List*",
                "events:Describe*"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Crea un ruolo IAM che il servizio EventBridge può assumere nel passaggio di un evento a AWS Glue.

   a. Nella pagina Create role (Crea ruolo) della console IAM, seleziona Service AWS (Servizio AWS). Quindi scegli il servizio CloudWatch Events.

b. Completa la procedura guidata Create role (Crea ruolo). La procedura guidata allega automaticamente il `CloudWatchEventsBuiltInTargetExecutionAccess` e le policy `CloudWatchEventsInvocationAccess`.

c. Allega la seguente policy inline al ruolo. Questa policy consente al servizio EventBridge di indirizzare gli eventi ad AWS Glue.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:notifyEvent"
      ],
      "Resource": [
        "arn:aws:glue:<region>:<account-id>:workflow/<workflow-name>"
      ]
    }
  ]
}
```

3. Inserisci il seguente comando per creare il flusso di lavoro.

   Per ulteriori informazioni sui parametri della riga di comando aggiuntivi, consulta create-workflow nel Riferimento ai comandi AWS CLI.

```
aws glue create-workflow --name <workflow-name>
```

4. Inserisci il comando seguente per creare un trigger di evento EventBridge per il flusso di lavoro. Questo sarà il trigger di avvio per il flusso di lavoro. Sostituisci *<actions>* con le azioni da eseguire (i processi e i crawler da avviare).

   Consulta create-trigger nel Riferimento ai comandi AWS CLI per informazioni su come codificare l'argomento `actions`.

```
aws glue create-trigger --workflow-name <workflow-name> --type EVENT --
name <trigger-name> --actions <actions>
```

   Se svuoi che il flusso di lavoro venga attivato da un batch di eventi anziché da un singolo evento EventBridge, inserisci il comando seguente.

```
aws glue create-trigger --workflow-name <workflow-name> --type EVENT
 --name <trigger-name> --event-batching-condition BatchSize=<number-of-
events>,BatchWindow=<seconds> --actions <actions>
```

Per l'argomento `event-batching-condition`, `BatchSize` è obbligatorio e `BatchWindow` è facoltativo. Se `BatchWindow` viene omesso, il valore predefinito della finestra è 900 secondi, ovvero la dimensione massima della finestra.

Example

Nell'esempio seguente viene creato un trigger che avvia il flusso di lavoro `eventtest` dopo l'arrivo di tre eventi EventBridge o cinque minuti dopo l'arrivo del primo evento, a seconda di quale evento si verifica per primo.

```
aws glue create-trigger --workflow-name eventtest --type EVENT --name objectArrival
 --event-batching-condition BatchSize=3,BatchWindow=300 --actions JobName=test1
```

5. Crea una regola in Amazon EventBridge.

   a. Crea l'oggetto JSON per i dettagli della regola nell'editor di testo che preferisci.

      Nell'esempio seguente viene specificato Amazon S3 come origine evento, `PutObject`come nome dell'evento e il nome del bucket come parametro di richiesta. Questa regola avvia un flusso di lavoro quando un nuovo oggetto arriva nel bucket.

```
{
  "source": [
    "aws.s3"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3.amazonaws.com"
    ],
    "eventName": [
      "PutObject"
    ],
    "requestParameters": {
```

```
      "bucketName": [
        "<bucket-name>"
      ]
    }
  }
}
```

Per avviare il flusso di lavoro quando un nuovo oggetto arriva in una cartella all'interno del bucket, è possibile sostituire il codice seguente con `requestParameters`.

```
    "requestParameters": {
      "bucketName": [
        "<bucket-name>"
      ]
      "key" : [{ "prefix" : "<folder1>/<folder2>/*"}}]
  }
```

b. Utilizza lo strumento che preferisci per convertire l'oggetto JSON regola in una stringa di escape.

```
{\n  \"source\": [\n    \"aws.s3\"\n  ],\n  \"detail-type\": [\n    \"AWS API
 Call via CloudTrail\"\n  ],\n  \"detail\": {\n    \"eventSource\": [\n
 \"s3.amazonaws.com\"\n    ],\n    \"eventName\": [\n      \"PutObject\"\n
  ],\n    \"requestParameters\": {\n      \"bucketName\": [\n        \"<bucket-
name>\"\n      ]\n    }\n  }\n}
```

c. Esegui il comando seguente per creare un modello di parametro JSON che puoi modificare per specificare i parametri di input in un comando `put-rule`successivo. Salva l'output in un file. Per questo esempio, il file è denominato `ruleCommand`.

```
aws events put-rule --name <rule-name> --generate-cli-skeleton >ruleCommand
```

Per ulteriori informazioni sul parametro `--generate-cli-skeleton`, consulta Generazione di parametri di input e skeleton AWS CLI da un file di input JSON o YAML nella Guida per l'utente di AWS.

Il file di output deve essere simile al seguente.

```
{
    "Name": "",
    "ScheduleExpression": "",
```

```
    "EventPattern": "",
    "State": "ENABLED",
    "Description": "",
    "RoleArn": "",
    "Tags": [
        {
            "Key": "",
            "Value": ""
        }
    ],
    "EventBusName": ""
}
```

d. Modifica il file per rimuovere facoltativamente i parametri e specificare come minimo i parametri `Name`, `EventPattern`, e`State`. Per il parametro `EventPattern`, specifica la stringa di escape per i dettagli della regola creati in un passaggio precedente.

```
{
    "Name": "<rule-name>",
    "EventPattern": "{\n  \"source\": [\n    \"aws.s3\"\n  ],\n  \"detail-
type\": [\n    \"AWS API Call via CloudTrail\"\n  ],\n  \"detail\": {\n
 \"eventSource\": [\n      \"s3.amazonaws.com\"\n    ],\n    \"eventName\": [\n
      \"PutObject\"\n    ],\n    \"requestParameters\": {\n      \"bucketName
\": [\n        \"<bucket-name>\"\n      ]\n    }\n  }\n}",
    "State": "DISABLED",
    "Description": "Start an AWS Glue workflow upon new file arrival in an
 Amazon S3 bucket"
}
```

> **Note**
>
> È consigliabile lasciare disabilitata la regola fino a quando non si completa la creazione del flusso di lavoro.

e. Immetti la seguente comando `put-rule`, che legge i parametri di input dal file `ruleCommand`.

```
aws events put-rule --name <rule-name> --cli-input-json file://ruleCommand
```

Il seguente output indica il successo.

```
{
    "RuleArn": "<rule-arn>"
}
```

6. Immetti il seguente comando per collegare la regola alla destinazione. La destinazione è il flusso di lavoro in AWS Glue. Sostituisci *<role-name>* con il ruolo creato all'inizio di questa procedura.

```
aws events put-targets --rule <rule-name> --targets
 "Id"="1","Arn"="arn:aws:glue:<region>:<account-id>:workflow/<workflow-
name>","RoleArn"="arn:aws:iam::<account-id>:role/<role-name>" --region <region>
```

   Il seguente output indica il successo.

```
{
    "FailedEntryCount": 0,
    "FailedEntries": []
}
```

7. Conferma la corretta connessione della regola e della destinazione inserendo il comando seguente.

```
aws events list-rule-names-by-target --target-arn arn:aws:glue:<region>:<account-
id>:workflow/<workflow-name>
```

   Il seguente output indica il successo, nel quale *<rule-name>* è il nome della regola creata.

```
{
    "RuleNames": [
        "<rule-name>"
    ]
}
```

8. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
9. Seleziona il flusso di lavoro e verifica che il trigger di avvio e le relative azioni, ovvero i processi o i crawler che avvia, vengano visualizzati nel grafico del flusso di lavoro. Poi, continua con la procedura in Fase 3: aggiunta di più trigger. In alternativa, aggiungi altri componenti al flusso di lavoro utilizzando l'API AWS Glue oAWS Command Line Interface.

10. Quando il flusso di lavoro è specificato completamente, abilita la regola.

```
aws events enable-rule --name <rule-name>
```

Ora il flusso di lavoro è pronto per essere avviato da un evento EventBridge o un batch di eventi.

Consulta anche

- Guida per l'utente di Amazon EventBridge
- Panoramica di flussi di lavoro in AWS Glue
- Creazione e costruzione manuale di un flusso di lavoro in AWS Glue

Visualizzazione degli eventi EventBridge che hanno avviato un flusso di lavoro

Puoi visualizzare l'ID evento dell'evento Amazon EventBridge che ha avviato il tuo flusso di lavoro. Se il flusso di lavoro è stato avviato da un batch di eventi, è possibile visualizzare gli ID evento di tutti gli eventi del batch.

Per i flussi di lavoro con una dimensione batch maggiore di uno, è inoltre possibile verificare quale condizione batch ha avviato il flusso di lavoro: l'arrivo del numero di eventi nella dimensione batch o la scadenza del periodo batch.

Per visualizzare gli eventi EventBridge che hanno avviato un flusso di lavoro (console)

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, scegli Workflows (Flussi di lavoro).
3. Seleziona un flusso di lavoro. Quindi, nella parte inferiore, scegli la scheda History (Cronologia).
4. Seleziona un'esecuzione del flusso di lavoro, quindi scegli View run details (Visualizza i dettagli dell'esecuzione).
5. Nella pagina dei dettagli di esecuzione, individua il campo Run propertiese cerca il campo (Proprietà di esecuzione) e cerca la chiave aws:eventIds.

   Il valore per tale chiave è un elenco di ID evento EventBridge.

Per visualizzare gli eventi EventBridge che hanno avviato un flusso di lavoro (API AWS)

- Includi il seguente codice nello script Python.

```
workflow_params =
 glue_client.get_workflow_run_properties(Name=workflow_name,RunId=workflow_run_id)
batched_events = workflow_params['aws:eventIds']
```

`batched_events` sarà un elenco di stringhe, in cui ogni stringa è un ID evento.

> Consulta anche
>
> - Guida per l'utente di Amazon EventBridge
> - the section called “Panoramica di flussi di lavoro”

Esecuzione e monitoraggio di un flusso di lavoro in AWS Glue

Se il trigger di attivazione di un flusso di lavoro è un trigger on-demand, è possibile avviare il flusso di lavoro dalla console di AWS Glue. Utilizza la procedura seguente per eseguire e monitorare un flusso di lavoro. Se il flusso di lavoro non riesce, puoi visualizzare il grafico di esecuzione per individuare il nodo non riuscito. Per facilitare la risoluzione dei problemi, se il flusso di lavoro è stato creato da un piano, puoi visualizzare l'esecuzione del piano per vedere i valori dei parametri del piano utilizzati per creare il flusso di lavoro. Per ulteriori informazioni, consulta the section called “Visualizzazione delle esecuzioni dello schema”.

Puoi eseguire e monitorare l'esecuzione di un flusso di lavoro utilizzando la console AWS Glue, l'API o AWS Command Line Interface (AWS CLI).

Per eseguire e monitorare un flusso di lavoro (console)

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in ETL, scegliere Workflows (Flussi di lavoro).
3. Selezionare un flusso di lavoro. Nel menu Actions (Operazioni), scegliere Run (Esegui).
4. Controlla la colonna Last run status (Stato dell'ultima esecuzione) nell'elenco dei flussi di lavoro. Scegli il pulsante di aggiornamento per visualizzare lo stato del flusso di lavoro in corso.

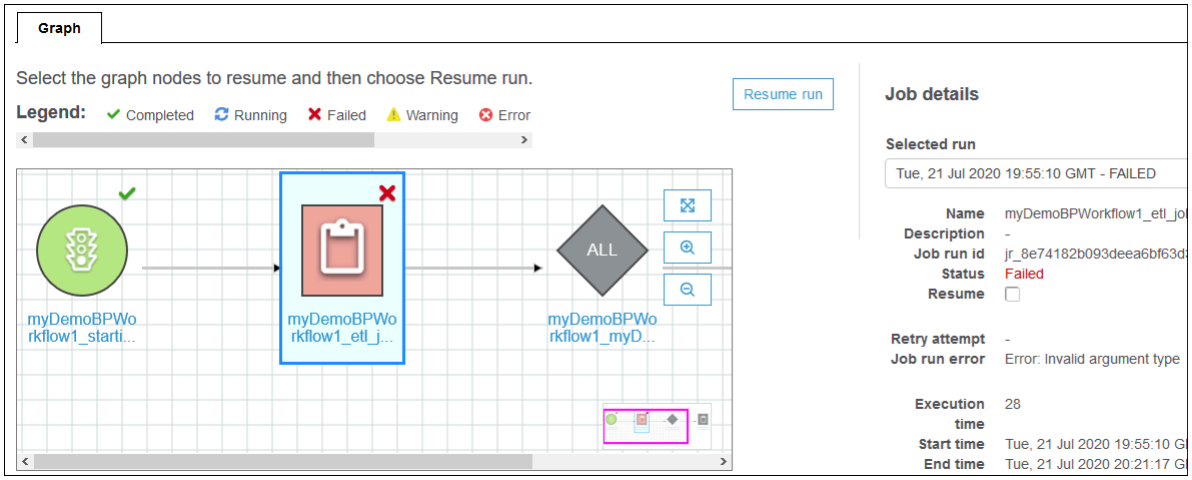
5. Mentre il flusso di lavoro è in esecuzione o dopo che è stato completato (o non riuscito), visualizza i dettagli dell'esecuzione completando la procedura seguente.

   a. Verifica che il flusso di lavoro sia selezionato e scegli la scheda History (Cronologia).

   b. Seleziona l'esecuzione del flusso di lavoro corrente o più recente, quindi seleziona View run details (Visualizza i dettagli dell'esecuzione).

      Il grafico del tempo di esecuzione del flusso di lavoro mostra lo stato corrente dell'esecuzione.

   c. Scegli un nodo nel grafico per visualizzare relativi i dettagli e lo stato.



Per eseguire e monitorare un flusso di lavoro (AWS CLI)

1. Inserire il seguente comando. Sostisci *<workflow-name>* con il flusso di lavoro da eseguire.

```
aws glue start-workflow-run --name <workflow-name>
```

   Se il flusso di lavoro viene avviato correttamente, il comando restituisce l'ID di esecuzione.

2. Visualizza lo stato di esecuzione del flusso di lavoro utilizzando il comando `get-workflow-run`. Specifica il nome del flusso di lavoro e l'ID di esecuzione.

```
aws glue get-workflow-run --name myWorkflow --run-id
 wr_d2af14217e8eae775ba7b1fc6fc7a42c795aed3cbcd8763f9415452e2dbc8705
```

   Di seguito è riportato un output del comando di esempio.

```
{
```

```
    "Run": {
        "Name": "myWorkflow",
        "WorkflowRunId":
 "wr_d2af14217e8eae775ba7b1fc6fc7a42c795aed3cbcd8763f9415452e2dbc8705",
        "WorkflowRunProperties": {
            "run_state": "COMPLETED",
            "unique_id": "fee63f30-c512-4742-a9b1-7c8183bdaae2"
        },
        "StartedOn": 1578556843.049,
        "CompletedOn": 1578558649.928,
        "Status": "COMPLETED",
        "Statistics": {
            "TotalActions": 11,
            "TimeoutActions": 0,
            "FailedActions": 0,
            "StoppedActions": 0,
            "SucceededActions": 9,
            "RunningActions": 0,
            "ErroredActions": 0
        }
    }
}
```

Consulta anche:

- the section called “Panoramica di flussi di lavoro”
- the section called “Panoramica degli schemi”

Arresto dell'esecuzione di un flusso di lavoro

Per arrestare una sessione, è possibile utilizzare la console AWS Glue, l’AWS Command Line Interface (AWS CLI) o l’API AWS Glue. Quando si interrompe l'esecuzione di un flusso di lavoro, tutti i processi e i crawler in esecuzione vengono immediatamente terminati e i processi e i crawler non ancora avviati non vengono mai avviati. Potrebbe essere necessario fino a un minuto prima che tutti i processi in esecuzione e i crawler si fermino. Lo stato di esecuzione del flusso di lavoro passa da Running (In esecuzione) a Stopping (Arresto in corso), e quando l'esecuzione del flusso di lavoro è completamente interrotta, lo stato passa a Stopped (Arrestato).

Dopo l'interruzione dell'esecuzione del flusso di lavoro, è possibile visualizzare il grafico di esecuzione per verificare quali processi e crawler sono stati completati e quali non sono mai stati avviati. È quindi possibile determinare se è necessario eseguire qualsiasi procedura per garantire l'integrità dei dati. L'arresto di un'esecuzione del flusso di lavoro non comporta l'esecuzione di operazioni di rollback automatico.

Per interrompere l'esecuzione di un flusso di lavoro (console)

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione, in ETL, scegliere Workflows (Flussi di lavoro).
3. Scegliere un flusso di lavoro in esecuzione, quindi scegliere la scheda History (Cronologia) .
4. Scegliere l'esecuzione del flusso di lavoro, quindi scegliere Stop run (Arresta esecuzione).

   Lo stato di esecuzione cambia in Stopping (Arresto in corso).
5. (Facoltativo) Scegliere l'esecuzione del flusso di lavoro, scegliere View run details (Visualizza dettagli esecuzione), ed esaminare il grafico di esecuzione.

Per interrompere l'esecuzione di un flusso di lavoro (AWS CLI)

- Inserire il seguente comando. Sostituire *<workflow-name>* con il nome del flusso di lavoro e *<run-id>* con l'ID di esecuzione dell'esecuzione del flusso di lavoro da interrompere.

  ```
  aws glue stop-workflow-run --name <workflow-name> --run-id <run-id>
  ```

  Di seguito è riportato un comando stop-workflow-run di esempio.

  ```
  aws glue stop-workflow-run --name my-workflow --run-id
   wr_137b88917411d128081069901e4a80595d97f719282094b7f271d09576770354
  ```

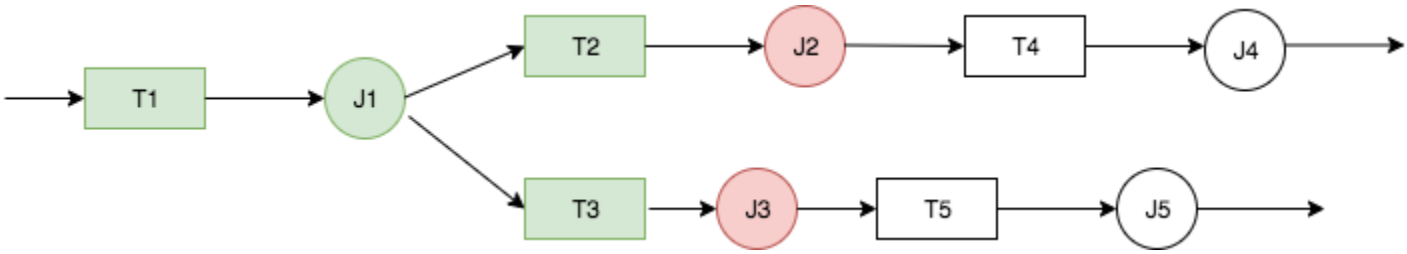
Ripresa e ripristino dell'esecuzione di un flusso di lavoro

Se uno o più nodi (processi o crawler) in un flusso di lavoro non vengono completati correttamente, ciò significa che il flusso di lavoro è stato eseguito solo parzialmente. Dopo aver individuato le cause principali e apportato le correzioni, è possibile selezionare uno o più nodi da cui riprendere l'esecuzione del flusso di lavoro e quindi riprendere l'esecuzione del flusso di lavoro. Vengono eseguiti i nodi selezionati e tutti i nodi che sono a valle dei nodi selezionati.

Argomenti

Riprendere un'esecuzione del flusso di lavoro: come funziona

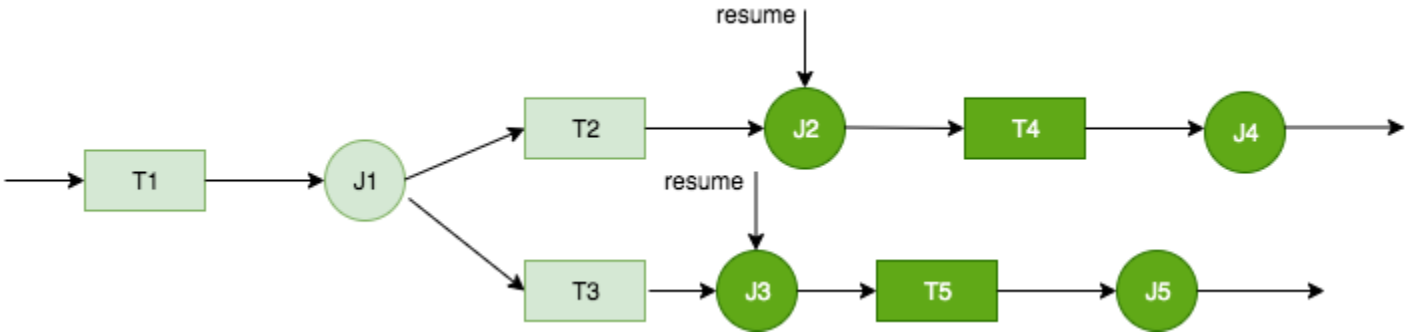
Considera il flusso di lavoro W1 nel diagramma seguente.



L'esecuzione del flusso di lavoro si svolge come segue:

1. Il trigger T1 avvia il processo J1.
2. Il completamento riuscito di J1 attiva i trigger T2 e T3, che eseguono i processi J2 e J3, rispettivamente.
3. I processi J2 e J3 non riescono.
4. I trigger T4 e T5 dipendono dal completamento riuscito di J2 e J3, quindi non si attivano e i processi J4 e J5 non vengono eseguiti. Il flusso di lavoro W1 viene eseguito solo parzialmente.

Ora supponiamo che i problemi che hanno causato la non riuscita di J2 e J3 vengano corretti. J2 e J3 sono selezionati come punti di partenza da cui riprendere l'esecuzione del flusso di lavoro.



L'esecuzione del flusso di lavoro riprende come segue:

1. I processi J2 e J3 vengono eseguiti correttamente.
2. I trigger T4 e T5 si attivano.
3. I processi J4 e J5 vengono eseguiti correttamente.

L'esecuzione del flusso di lavoro ripreso viene registrata come un'esecuzione separata del flusso di lavoro con un nuovo ID di esecuzione. Nella cronologia del flusso di lavoro, è possibile visualizzare l'ID dell' esecuzione precedente per qualsiasi esecuzione del flusso di lavoro. Nell'esempio nello screenshot seguente, un nodo dell'esecuzione del flusso di lavoro con ID di esecuzione `wr_c7a22...` (la seconda riga) non è stato completato. L'utente ha risolto il problema e ha ripreso l'esecuzione del flusso di lavoro, con ID di esecuzione `wr_a07e55...` (la prima riga).

> **Note**
>
> Per il resto di questo articolo, il termine "esecuzione del flusso di lavoro ripreso" si riferisce all'esecuzione del flusso di lavoro creata quando è stata ripresa l'esecuzione precedente del flusso di lavoro. L'"esecuzione del flusso di lavoro originale" indica l'esecuzione del flusso di lavoro che è stata eseguita solo parzialmente e che doveva essere ripresa.

Grafico dell'esecuzione del flusso di lavoro ripreso

In un'esecuzione del flusso di lavoro ripreso, nonostante venga eseguito solo un sottoinsieme di nodi, il grafico di esecuzione è un grafico completo. Cioè, i nodi che non sono stati eseguiti nel flusso di lavoro ripreso vengono copiati dal grafico dell'esecuzione del flusso di lavoro originale. I nodi del processo e del crawler copiati eseguiti nell'esecuzione del flusso di lavoro originale includono dettagli di esecuzione.

Considera nuovamente il flusso di lavoro W1 nel diagramma precedente. Quando l'esecuzione del flusso di lavoro viene ripresa a partire da J2 e J3, il grafico di esecuzione per l'esecuzione del flusso di lavoro ripreso mostra tutti i processi, da J1 a J5, e tutti i trigger, da T1 a T5. I dettagli del processo per J1 vengono copiati dall'esecuzione del flusso di lavoro originale.

Snapshot dell'esecuzione del flusso di lavoro

All'avvio di un'esecuzione del flusso di lavoro, AWS Glue acquisisce uno snapshot del grafico di progettazione del flusso di lavoro in quel momento. Lo snapshot viene utilizzato per la durata dell'esecuzione del flusso di lavoro. Se si apportano modifiche a qualsiasi trigger dopo l'avvio dell'esecuzione, tali modifiche non influiscono sull'esecuzione del flusso di lavoro corrente. Gli snapshot garantiscono che le esecuzioni del flusso di lavoro procedano in modo coerente.

Gli snapshot rendono immutabili solo i trigger. Le modifiche apportate ai processi a valle e ai crawler durante l'esecuzione del flusso di lavoro diventano effettive per l'esecuzione corrente.

Riprendere un'esecuzione di flusso di lavoro

Segui questi passaggi per riprendere un'esecuzione di flusso di lavoro. È possibile riprendere un'esecuzione di flusso di lavoro utilizzando la console AWS Glue, l'API o AWS Command Line Interface (AWS CLI).

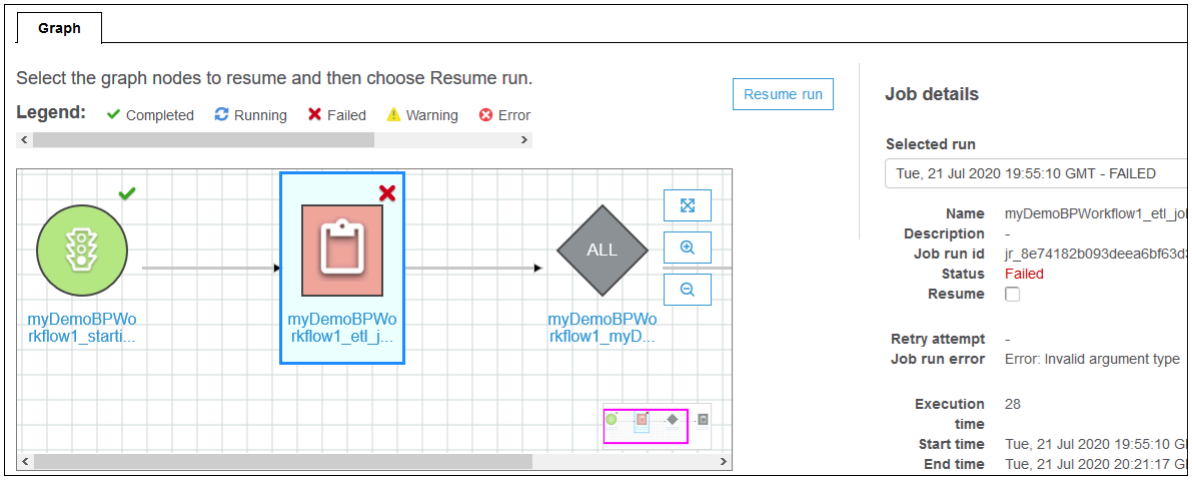
Per riprendere un flusso di lavoro (console)

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.

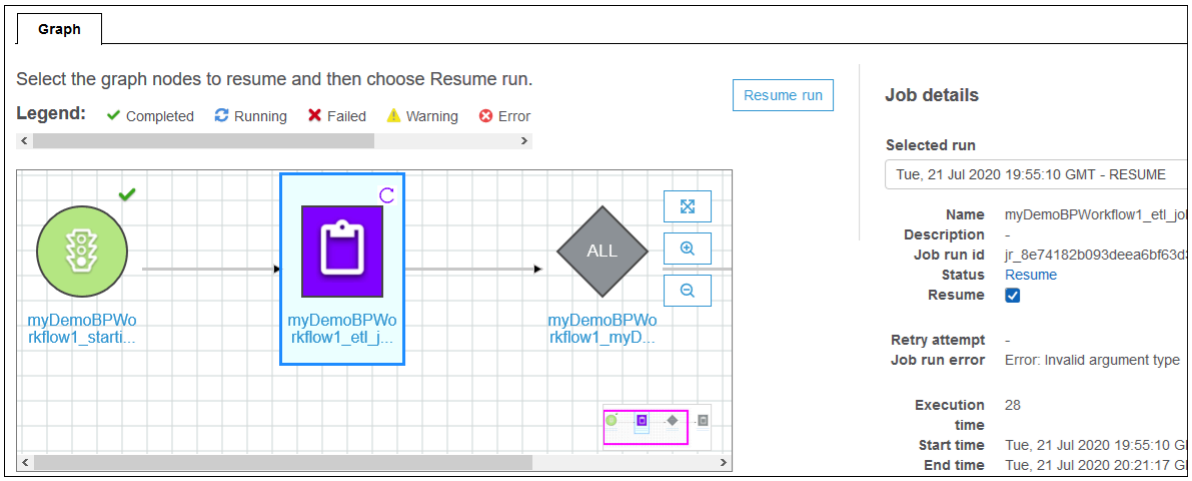
   Accedi come utente che dispone delle autorizzazioni per visualizzare i flussi di lavoro e riprendere le esecuzioni dei flussi di lavoro.

   > **Note**
   >
   > Per riprendere l'esecuzione del flusso di lavoro, è necessaria l'autorizzazione `glue:ResumeWorkflowRun` AWS Identity and Access Management (IAM).

2. Nel pannello di navigazione, scegli Workflows (Flussi di lavoro).
3. Seleziona un flusso di lavoro, quindi la scheda History (Cronologia).
4. Seleziona un'esecuzione del flusso di lavoro eseguita solo parzialmente, quindi seleziona View run details (Visualizza i dettagli dell'esecuzione).
5. Nel grafico di esecuzione seleziona il primo (o solo) nodo da riavviare e da cui riprendere l'esecuzione del flusso di lavoro.
6. Nel riquadro dei dettagli a destra del grafico, seleziona la casella di controlloResume (Riprendi).



Il nodo cambia colore e mostra una piccola icona di ripresa in alto a destra.



7. Completa i due passaggi precedenti per riavviare eventuali nodi aggiuntivi.
8. Seleziona Resume (Riprendi).

Per riprendere un flusso di lavoro (AWS CLI)

1. Assicurati di disporre dell'autorizzazione `glue:ResumeWorkflowRun` IAM.
2. Recupera un elenco degli ID dei i nodi da riavviare.

   a. Esegui il comando `get-workflow-run` per l'esecuzione del flusso di lavoro originale. Fornisci il nome del flusso di lavoro e l'ID di esecuzione e aggiungi l'opzione `--include-graph`, come mostrato nell'esempio seguente. Recupera l'ID di esecuzione dalla scheda History (Cronologia) sulla console o eseguendo il comando `get-workflow`.

```
aws glue get-workflow-run --name cloudtrailtest1 --run-id
 wr_a07e55f2087afdd415a404403f644a4265278f68b13ba3da08c71924ebe3c3a8 --include-
graph
```

Il comando restituisce i nodi e gli estremi del grafico come un oggetto JSON di grandi dimensioni.

b. Individua i nodi di interesse dalle proprietà `Type` e `Name` degli oggetti nodo.

Il seguente è un esempio di oggetto nodo dell'output.

```
{
    "Type": "JOB",
    "Name": "test1_post_failure_4592978",
    "UniqueId":
 "wnode_d1b2563c503078b153142ee76ce545fe5ceef66e053628a786ddd74a05da86fd",
    "JobDetails": {
        "JobRuns": [
            {
                "Id":
 "jr_690b9f7fc5cb399204bc542c6c956f39934496a5d665a42de891e5b01f59e613",
                "Attempt": 0,
                "TriggerName": "test1_aggregate_failure_649b2432",
                "JobName": "test1_post_failure_4592978",
                "StartedOn": 1595358275.375,
                "LastModifiedOn": 1595358298.785,
                "CompletedOn": 1595358298.785,
                "JobRunState": "FAILED",
                "PredecessorRuns": [],
                "AllocatedCapacity": 0,
                "ExecutionTime": 16,
                "Timeout": 2880,
                "MaxCapacity": 0.0625,
                "LogGroupName": "/aws-glue/python-jobs"
            }
        ]
    }
}
```

c. Recupera l'ID nodo dalla proprietà `UniqueId` dell'oggetto nodo.

3. Esegui il comando `resume-workflow-run`. Specifica il nome del flusso di lavoro, l'ID di esecuzione e l'elenco degli ID di nodo separati da spazi, come mostrato nell'esempio seguente.

```
aws glue resume-workflow-run --name cloudtrailtest1 --run-id
 wr_a07e55f2087afdd415a404403f644a4265278f68b13ba3da08c71924ebe3c3a8 --node-
ids wnode_ca1f63e918fb855e063aed2f42ec5762ccf71b80082ae2eb5daeb8052442f2f3
 wnode_d1b2563c503078b153142ee76ce545fe5ceef66e053628a786ddd74a05da86fd
```

Il comando restituisce l'ID di esecuzione della (nuova) esecuzione del flusso di lavoro ripresa e un elenco di nodi che verranno avviati.

```
{
    "RunId": "wr_2ada0d3209a262fc1156e4291134b3bd643491bcfb0ceead30bd3e4efac24de9",
    "NodeIds": [
        "wnode_ca1f63e918fb855e063aed2f42ec5762ccf71b80082ae2eb5daeb8052442f2f3"
    ]
}
```

Nota che anche se l'esempio `resume-workflow-run` elencava due nodi da riavviare, l'output di esempio indicava che un solo nodo sarebbe stato riavviato. Questo perché un nodo era a valle dell'altro nodo e il nodo a valle viene comunque riavviato dal flusso normale del flusso di lavoro.

Note e limitazioni per la ripresa delle esecuzioni del flusso di lavoro

Tieni presente le seguenti note e limitazioni per la ripresa delle esecuzioni del flusso di lavoro.

- È possibile riprendere l'esecuzione di un flusso di lavoro solo se si trova nello stato `COMPLETED`.

> **Note**
>
> Anche se uno o più nodi in un'esecuzione del flusso di lavoro non vengono completati, lo stato di esecuzione del flusso di lavoro viene visualizzato come `COMPLETED`. Accertati di controllare il grafico di esecuzione per individuare eventuali nodi che non sono stati completati correttamente.

- È possibile riprendere l'esecuzione di un flusso di lavoro da qualsiasi nodo del processo o del crawler che il flusso di lavoro originale ha tentato di eseguire. Non è possibile riprendere l'esecuzione di un flusso di lavoro da un nodo trigger.
- Il riavvio di un nodo non ne reimposta lo stato. Tutti i dati parzialmente elaborati non vengono ripristinati.

- È possibile riprendere la stessa esecuzione del flusso di lavoro più volte. Se un flusso di lavoro ripreso viene eseguito solo parzialmente, è possibile risolvere il problema e riprendere nuovamente l'esecuzione.
- Se si selezionano due nodi da riavviare e dipendono l'uno dall'altro, il nodo a monte viene eseguito prima del nodo a valle. Di fatto, la selezione del nodo a valle è ridondante, perché viene eseguito in base al normale flusso di lavoro.

Recupero e impostazione delle proprietà di esecuzione del flusso di lavoro in AWS Glue

Utilizzare le proprietà di esecuzione dei flussi di lavoro per condividere e gestire lo stato tra i processi del flusso di lavoro di AWS Glue. È possibile impostare proprietà di esecuzione di default al momento della creazione del flusso di lavoro. Quindi, nel momento in cui i processi sono eseguiti, è possibile recuperare i valori delle proprietà di esecuzione e, se necessario, modificarli come input per i processi successivi nel flusso di lavoro. Quando un processo modifica una proprietà di esecuzione, il nuovo valore esiste solo per il flusso di lavoro in esecuzione. Le proprietà di esecuzione predefinite non sono interessate.

Se il processo AWS Glue non fa parte di un workflow, queste proprietà non verranno impostate.

Il codice Python di esempio seguente relativo a un processo di estrazione, trasformazione e caricamento (ETL) mostra come recuperare le proprietà di esecuzione del flusso di lavoro.

```
import sys
import boto3
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from awsglue.context import GlueContext
from pyspark.context import SparkContext

glue_client = boto3.client("glue")
args = getResolvedOptions(sys.argv, ['JOB_NAME','WORKFLOW_NAME', 'WORKFLOW_RUN_ID'])
workflow_name = args['WORKFLOW_NAME']
workflow_run_id = args['WORKFLOW_RUN_ID']
workflow_params = glue_client.get_workflow_run_properties(Name=workflow_name,
                                       RunId=workflow_run_id)["RunProperties"]

target_database = workflow_params['target_database']
target_s3_location = workflow_params['target_s3_location']
```

Il codice seguente prosegue impostando la proprietà di esecuzione `target_format` al valore `'csv'`.

```
workflow_params['target_format'] = 'csv'
glue_client.put_workflow_run_properties(Name=workflow_name, RunId=workflow_run_id,
 RunProperties=workflow_params)
```

Per ulteriori informazioni, consulta gli argomenti seguenti:

- GetWorkflowRunProperties azione (Python: get_workflow_run_properties)
- PutWorkflowRunProperties azione (Python: put_workflow_run_properties)

Eseguire query sui flussi di lavoro utilizzando AWS Glue API

AWS Glue fornisce una ricca API per la gestione dei flussi di lavoro. Utilizzando la AWS Glue API è possibile recuperare una visualizzazione statica o dinamica di un flusso di lavoro in esecuzione. Per ulteriori informazioni, consulta Flussi di lavoro.

Argomenti

- Eseguire query sulle visualizzazioni statiche
- Eseguire query sulle visualizzazioni dinamiche

Eseguire query sulle visualizzazioni statiche

Per ottenere una visualizzazione statica che descrive la configurazione di un flusso di lavoro, utilizzare l'operazione API `GetWorkflow`. Questa operazione restituisce un diagramma orientato composto da nodi e archi, nel quale ogni nodo rappresenta un trigger, un processo o un crawler. Gli archi definiscono le relazioni tra i nodi. Sono rappresentati da connettori (frecce) all'interno del diagramma visualizzato nella console di AWS Glue.

È anche possibile usare questa operazione con popolari librerie per l'elaborazione di diagrammi come, ad esempio, NetworkX, igraph, JGraphT e il framework Java Universal Network/Graph (JUNG). Poiché tutte queste librerie rappresentano i diagrammi in modo analogo, sono necessarie trasformazioni minime.

La visualizzazione statica restituita da questa API è quella più aggiornata rispetto alle ultime definizioni di trigger associati al flusso di lavoro.

Definizione del diagramma

Un diagramma di un flusso di lavoro G è una coppia ordinata (N, E), dove N è costituito da un insieme di nodi ed E è costituito da un insieme di archi. Un Nodo è un vertice del diagramma identificato da un numero univoco. Un nodo può essere di tipo trigger, processo o crawler. Ad esempio: `{name:T1, type:Trigger, uniqueId:1}, {name:J1, type:Job, uniqueId:2}`.

Un Edge è una tupla nella forma (`src, dest`), dove `src` e `dest` sono nodi ed è presente un arco diretto che va da `src` a `dest`.

Esempio di esecuzione di query di una visualizzazione statica.

Considerare un trigger condizionale T, che attiva il processo J2 al completamento del processo J1.

```
J1 ---> T ---> J2
```

Nodi: J1, T, J2

Archi: (J1, T), (T, J2)

Eseguire query sulle visualizzazioni dinamiche

Per ottenere una visualizzazione dinamica di un flusso di lavoro in esecuzione, utilizzare l'operazione API `GetWorkflowRun`. Questa operazione restituisce una visualizzazione identica a quella statica con i metadati correlati all'esecuzione del flusso di lavoro.

Per l'esecuzione, i nodi che rappresentano i processi nella chiamata `GetWorkflowRun` sono associati a un elenco di esecuzioni dei processi derivanti dall'ultima esecuzione del flusso di lavoro. È possibile usare questo elenco per visualizzare lo stato di esecuzione di ogni processo nel diagramma stesso. Per dipendenze a valle non ancora eseguite, questo campo è impostato su `null`. Le informazioni rappresentate sul diagramma permettono di conoscere lo stato attuale di qualsiasi flusso di lavoro in qualsiasi momento.

La visualizzazione dinamica restituita da questa API si basa sulla visualizzazione statica presente al momento dell'avvio dell'esecuzione del flusso di lavoro.

Esempio del tempo di esecuzione dei nodi: `{name:T1, type: Trigger, uniqueId:1}`, `{name:J1, type:Job, uniqueId:2, jobDetails:{jobRuns}}`, `{name:C1, type:Crawler, uniqueId:3, crawlerDetails:{crawls}}`

Esempio 1: visualizzazione dinamica

L'esempio seguente illustra un semplice flusso di lavoro costituito da due trigger.

- Nodi: t1, j1, t2, j2
- Archi: (t1, j1), (j1, t2), (t2, j2)

La risposta di `GetWorkflow` contiene quanto segue.

```
{
    Nodes : [
        {
            "type" : Trigger,
            "name" : "t1",
            "uniqueId" : 1
        },
        {
            "type" : Job,
            "name" : "j1",
            "uniqueId" : 2
        },
        {
            "type" : Trigger,
            "name" : "t2",
            "uniqueId" : 3
        },
        {
            "type" : Job,
            "name" : "j2",
            "uniqueId" : 4
        }
    ],
    Edges : [
        {
            "sourceId" : 1,
            "destinationId" : 2
        },
        {
            "sourceId" : 2,
            "destinationId" : 3
        },
        {
```

```
            "sourceId" : 3,
            "destinationId" : 4
        }
}
```

La risposta di `GetWorkflowRun` contiene quanto segue.

```
{
    Nodes : [
        {
            "type" : Trigger,
            "name" : "t1",
            "uniqueId" : 1,
            "jobDetails" : null,
            "crawlerDetails" : null
        },
        {
            "type" : Job,
            "name" : "j1",
            "uniqueId" : 2,
            "jobDetails" : [
                {
                    "id" : "jr_12334",
                    "jobRunState" : "SUCCEEDED",
                    "errorMessage" : "error string"
                }
            ],
            "crawlerDetails" : null
        },
        {
            "type" : Trigger,
            "name" : "t2",
            "uniqueId" : 3,
            "jobDetails" : null,
            "crawlerDetails" : null
        },
        {
            "type" : Job,
            "name" : "j2",
            "uniqueId" : 4,
            "jobDetails" : [
                {
                    "id" : "jr_1233sdf4",
```

```
                    "jobRunState" : "SUCCEEDED",
                    "errorMessage" : "error string"
                }
            ],
            "crawlerDetails" : null
        }
    ],
    Edges : [
        {
            "sourceId" : 1,
            "destinationId" : 2
        },
        {
            "sourceId" : 2,
            "destinationId" : 3
        },
        {
            "sourceId" : 3,
            "destinationId" : 4
        }
}
```

Esempio 2: processi multipli con un trigger condizionale.

L'esempio seguente mostra un flusso di lavoro con processi multipli e un trigger condizionale (t3).

```
Consider Flow:
T(t1) ---> J(j1) ---> T(t2) ---> J(j2)
             |                    |
             |                    |
             >+------> T(t3) <-----+
                        |
                        |
                      J(j3)

Graph generated:
Nodes: t1, t2, t3, j1, j2, j3
Edges: (t1, j1), (j1, t2), (t2, j2), (j1, t3), (j2, t3), (t3, j3)
```

Restrizioni dei flussi di lavoro e degli schemi in AWS Glue

Di seguito sono riportate le restrizioni imposte sui piani e sui flussi di lavoro.

Restrizioni degli schemi

Tieni a mente le seguenti restrizioni relative al piano:

- Il piano deve essere registrato nella stessa regione AWS in si trova il bucket Amazon S3.
- Per condividere i piani tra gli account AWS devi concedere le autorizzazioni di lettura per l'archivio ZIP del piano in Amazon S3. I clienti che hanno l'autorizzazione di lettura su un archivio ZIP di un piano possono registrare il piano nel proprio account AWS e utilizzarlo.
- L'insieme di parametri del piano viene memorizzato come un singolo oggetto JSON. La lunghezza massima di questo oggetto è 128 KB.
- La dimensione massima non compressa dell'archivio ZIP del piano è 5 MB. La dimensione massima compressa è 1 MB.
- Limita il numero totale di processi, crawler e attivazioni all'interno di un flusso di lavoro a 100 o meno. Se includi più di 100, potresti riscontrare errori durante il tentativo di riprendere o interrompere l'esecuzione del flusso di lavoro.

Restrizioni dei flussi di lavoro

Tieni a mente le seguenti restrizioni relative ai flussi di lavoro. Alcuni di questi commenti sono principalmente indirizzati a utenti che creano flussi di lavoro manualmente.

- La dimensione massima del batch per un trigger di evento Amazon EventBridge è 100. La dimensione massima della finestra è di 900 secondi (15 minuti).
- Un trigger può essere associato a un solo flusso di lavoro.
- È permessa la configurazione di un solo trigger di attivazione (on demand o pianificato).
- Se un processo o un crawler in un flusso di lavoro viene avviato da un trigger esterno al flusso di lavoro, eventuali trigger all'interno del flusso di lavoro che dipendono dal completamento del processo o del crawler (completato o meno) non vengono attivati.
- Analogamente, se un processo o crawler in un flusso di lavoro dispone di trigger che dipendono dal completamento del processo o del crawler (completato o meno) sia all'interno del flusso di lavoro che all'esterno del flusso di lavoro e se il processo o il crawler viene avviato da un flusso di lavoro, solo i trigger all'interno del flusso di lavoro si attivano al completamento del processo o del crawler.

Risoluzione degli errori relativi agli schemi in AWS Glue

Se riscontri errori durante l'utilizzo dei piani AWS Glue, usa le seguenti soluzioni per trovare l'origine dei problemi e correggerli.

Argomenti

Errore: modulo PySpark mancante

AWS Glue restituisce l'errore "Unknown error executing layout generator function ModuleNotFoundError: No module named 'pyspark" (Errore sconosciuto durante l'esecuzione della funzione del generatore di layout moduleNotFoundError: nessun modulo chiamato 'pyspark').

Quando si decomprime l'archivio del piano, potrebbe verificarsi uno dei seguenti casi:

```
$ unzip compaction.zip
Archive:  compaction.zip
   creating: compaction/
  inflating: compaction/blueprint.cfg
  inflating: compaction/layout.py
  inflating: compaction/README.md
  inflating: compaction/compaction.py

$ unzip compaction.zip
Archive:  compaction.zip
  inflating: blueprint.cfg
  inflating: compaction.py
  inflating: layout.py
```

```
  inflating: README.md
```

Nel primo caso, tutti i file relativi al piano sono stati collocati in una cartella denominata compattazione convertita poi in un file zip denominato compaction.zip.

Nel secondo caso, tutti i file necessari per il piano non sono stati inclusi in una cartella e sono stati aggiunti come file root sotto il file zip compaction.zip.

È consentita la creazione di un file in uno dei formati sopra indicati. Accertati che `blueprint.cfg` abbia il percorso corretto al nome della funzione nello script che genera il layout.

Esempi

Nel caso 1: `blueprint.cfg` deve avere `layoutGenerator` come segue:

```
layoutGenerator": "compaction.layout.generate_layout"
```

Nel caso 2: `blueprint.cfg` deve avere `layoutGenerator` come segue

```
layoutGenerator": "layout.generate_layout"
```

Se questo percorso non è incluso correttamente, è possibile che venga visualizzato l'errore indicato. Ad esempio, se disponi della struttura di cartelle come indicato nel caso 2 e `layoutGenerator` come indicato come nel caso 1, potresti visualizzare l'errore di cui sopra.

Errore: file di configurazione del piano mancante

AWS Glue restituisce l'errore "Unknown error executing layout generator function FileNotFoundError: [Errno 2] No such file or directory: '/tmp/compaction/blueprint.cfg'". (Errore sconosciuto durante l'esecuzione della funzione del generatore di layout FileNotFoundError: [Errno 2] Nessun file o directory di questo tipo: '/tmp/compaction/blueprint.cfg').

blueprint.cfg deve essere posizionato al livello root dell'archivio ZIP o all'interno di una cartella che ha lo stesso nome dell'archivio ZIP.

Quando si estrae l'archivio ZIP del piano, blueprint.cfg deve essere trovato in uno dei seguenti percorsi. Se non viene trovato in uno dei seguenti percorsi, potresti visualizzare l'errore di cui sopra.

```
$ unzip compaction.zip
Archive:  compaction.zip
```

```
   creating: compaction/
  inflating: compaction/blueprint.cfg

$ unzip compaction.zip
Archive:  compaction.zip
  inflating: blueprint.cfg
```

Errore: file importato mancante

AWS Glue restituisce l'errore "Unknown error executing layout generator function FileNotFoundError: [Errno 2] No such file or directory:* *'demo-project/foo.py'". (Errore sconosciuto durante l'esecuzione della funzione del generatore di layout FileNotFoundError: [Errno 2] Nessun file o directory di questo tipo:* *'demo-project/foo.py').

Se lo script di generazione del layout dispone di funzionalità per leggere altri file, accertati di fornire un percorso completo per il file da importare. Ad esempio, Layout.py potrebbe fare riferimento allo script Conversion.py. Per ulteriori informazioni, consulta Progetto di schema di esempio.

Errore: non autorizzato a eseguire iamPassRole sulla risorsa

AWS Glue Restituisce l'errore "User: arn:aws:sts::123456789012:assumed-role/AWSGlueServiceRole/GlueSession is not authorized to perform: iam:PassRole on resource: arn:aws:iam::123456789012:role/AWSGlueServiceRole" (Utente: arn:aws:sts::123456789012:assumed-role/AWSGlueServiceRole/GlueSession non è autorizzato a eseguire: iam:PassRole sulla risorsa: arn:aws:iam::123456789012:role/AWSGlueServiceRole"

Se i processi e i crawler nel flusso di lavoro assumono lo stesso ruolo del ruolo passato per creare il flusso di lavoro dal piano, il ruolo del piano deve includere l'autorizzazione `iam:PassRole` su se stesso.

Se i processi e i crawler nel flusso di lavoro assumono un ruolo diverso da quello passato per creare le entità del flusso di lavoro dal piano, il ruolo del piano deve includere l'autorizzazione `iam:PassRole` sull'altro ruolo invece che sul ruolo del piano.

Per ulteriori informazioni, consulta Autorizzazioni per i ruoli degli schemi.

Errore: pianificazione cron non valida

AWS Glue restituisce l'errore "The schedule cron(0 0 * * * *) is invalid." (La pianificazione cron(0 0 * * * *) non è valida)

Fornisci un'espressione cron valida. Per ulteriori informazioni, consulta Pianificazioni basate sul tempo per processi e crawler.

Errore: esiste già un trigger con questo nome

AWS Glue restituisce l'errore "Trigger with name 'foo_starting_trigger' already submitted with different configuration" (Trigger con nome foo_starting_trigger' già inviato con configurazione diversa).

Un piano non richiede la definizione dei trigger nello script di layout per la creazione del flusso di lavoro. La creazione dei trigger viene gestita dalla libreria del piano in base alle dipendenze definite tra due operazioni.

La denominazione per i trigger è la seguente:

- Per il trigger iniziale nel flusso di lavoro, la denominazione è <workflow_name>_starting_trigger.
- Per un nodo (processo/crawler) nel flusso di lavoro che dipende dal completamento di uno o più nodi a monte; AWS Glue definisce un trigger con il nome <workflow_name>_<node_name>_trigger

Questo errore indica che esiste già un trigger con lo stesso nome. È possibile eliminare il trigger esistente ed eseguire nuovamente la creazione del flusso di lavoro.

> **Note**
>
> L'eliminazione di un flusso di lavoro non comporta l'eliminazione dei nodi all'interno del flusso di lavoro. È possibile che i trigger vengano lasciati anche se il flusso di lavoro viene eliminato. Per questo motivo, nel caso in cui crei un flusso di lavoro, lo elimini e quindi provi a ricrearlo con lo stesso nome dallo stesso piano, potresti non ricevere un errore che indica che il flusso di lavoro esiste già, ma potresti ricevere un errore che indica che il trigger esiste già.

Errore: un flusso di lavoro con nome “foo” esiste già.

Il nome del flusso di lavoro deve essere univoco. Prova con un nome diverso.

Errore: modulo non trovato nel percorso layoutGenerator specificato

AWS Glue restituisce l'errore "Unknown error executing layout generator function ModuleNotFoundError: No module named 'crawl_s3_locations'" (Errore sconosciuto durante

l'esecuzione della funzione del generatore di layout moduleNotFoundError: nessun modulo chiamato 'crawl_s3_locations').

```
layoutGenerator": "crawl_s3_locations.layout.generate_layout"
```

Ad esempio, disponi del percorso LayoutGenerator di cui sopra, quando decomprimi l'archivio del piano, questo deve essere simile al seguente:

```
$ unzip crawl_s3_locations.zip
Archive:  crawl_s3_locations.zip
   creating: crawl_s3_locations/
  inflating: crawl_s3_locations/blueprint.cfg
  inflating: crawl_s3_locations/layout.py
  inflating: crawl_s3_locations/README.md
```

Quando decomprimi l'archivio, se l'archivio del piano è simile al seguente, potresti visualizzare l'errore precedente.

```
$ unzip crawl_s3_locations.zip
Archive:  crawl_s3_locations.zip
  inflating: blueprint.cfg
  inflating: layout.py
  inflating: README.md
```

Non esiste una cartella denominata `crawl_s3_locations` e quando il percorso `layoutGenerator` fa riferimento al file di layout tramite il modulo `crawl_s3_locations`, potresti visualizzare l'errore di cui sopra.

Errore: errore di convalida nel campo Connections (Connessioni)

AWS Glue restituisce l'errore "Unknown error executing layout generator function TypeError: Value ['foo'] for key Connections should be of type <class 'dict'>!" (Errore sconosciuto durante l'esecuzione della funzione del generatore di layout TypeError: Value ['foo'] per le connessioni chiave dovrebbe essere di tipo <class 'dict'>!).

Si tratta di un errore di convalida. Il campo `Connections` nella classe `Job` si aspetta un dizionario e invece viene fornito un elenco di valori che causa l'errore.

```
User input was list of values
Connections= ['string']
```

```
Should be a dict like the following
Connections*=*{'Connections': ['string']}
```

Per evitare questi errori di runtime durante la creazione di un flusso di lavoro da uno schema, puoi convalidare le definizioni del flusso di lavoro, del processo e del crawler come descritto in Test di uno schema.

Fai riferimento alla sintassi in Riferimento alle classi di schemaAWS Glue per definire processo, crawler e flusso di lavoro AWS Glue nello script di layout.

Autorizzazioni per utenti e ruoli per gli schemi AWS Glue

Di seguito sono riportati gli utenti tipici e le autorizzazioni AWS Identity and Access Management (IAM) suggerite per utenti e ruoli per i progetti AWS Glue.

Argomenti

- Utenti dello schema
- Autorizzazioni per gli utenti dei piani
- Autorizzazioni per i ruoli degli schemi

Utenti dello schema

Di seguito sono riportati i utenti generalmente coinvolti nel ciclo di vita dei piani AWS Glue.

Utente	Descrizione
Sviluppatore AWS Glue	Sviluppa, verifica e pubblica progetti.
Amministratore di AWS Glue	Registra, mantiene e concede le autorizzazioni per i piani.
Analista dei dati	Esegue i piani per creare flussi di lavoro.

Per ulteriori informazioni, consulta the section called “Panoramica degli schemi”.

Autorizzazioni per gli utenti dei piani

Di seguito sono riportate le autorizzazioni suggerite per ogni utente del piano.

Autorizzazioni per i progetti per lo sviluppatore di AWS Glue per gli schemi

Lo sviluppatore di AWS Glue deve disporre delle autorizzazioni di scrittura sul bucket Amazon S3 utilizzato per pubblicare il progetto. Spesso, lo sviluppatore registra il piano dopo averlo caricato. In tal caso, lo sviluppatore necessita delle autorizzazioni elencate in the section called “Autorizzazioni per i progetti per l'amministratore di AWS Glue per gli schemi”. Inoltre, se lo sviluppatore desidera testare il piano dopo la registrazione, ha bisogno anche delle autorizzazioni elencate in the section called “Autorizzazioni per gli schemi per l'analista dati”.

Autorizzazioni per i progetti per l'amministratore di AWS Glue per gli schemi

La policy seguente concede le autorizzazioni per la registrazione, la visualizzazione e la gestione dei progetti AWS Glue.

> **Important**
>
> Nella policy riportata di seguito, sostituisci *<s3-bucket-name>* e *<prefix>* con il percorso Amazon S3 per gli archivi ZIP del piano caricati per la registrazione.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateBlueprint",
                "glue:UpdateBlueprint",
                "glue:DeleteBlueprint",
                "glue:GetBlueprint",
                "glue:ListBlueprints",
                "glue:BatchGetBlueprints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::<s3-bucket-name>/<prefix>/*"
```

```
        }
    ]
}
```

Autorizzazioni per gli schemi per l'analista dati

Il criterio seguente concede le autorizzazioni per eseguire i piani e per visualizzare il flusso di lavoro e i relativi i componenti risultanti. Concede inoltre `PassRole` al ruolo che AWS Glue assume per creare il flusso di lavoro e i relativi componenti.

La policy concede le autorizzazioni su qualsiasi risorsa. Per configurare l'accesso granulare a singoli piani, utilizza il seguente formato per gli ARN del piano:

```
arn:aws:glue:<region>:<account-id>:blueprint/<blueprint-name>
```

> **Important**
>
> Nella policy riportata di seguito, sostituisci *<account-id>*con un account AWS valido e sostituisci *<role-name>* con il nome del ruolo utilizzato per eseguire un piano. Consulta the section called “Autorizzazioni per i ruoli degli schemi” per le autorizzazioni richieste da questo ruolo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListBlueprints",
                "glue:GetBlueprint",
                "glue:StartBlueprintRun",
                "glue:GetBlueprintRun",
                "glue:GetBlueprintRuns",
                "glue:GetCrawler",
                "glue:ListTriggers",
                "glue:ListJobs",
                "glue:BatchGetCrawlers",
                "glue:GetTrigger",
                "glue:BatchGetWorkflows",
                "glue:BatchGetTriggers",
```

```
                "glue:BatchGetJobs",
                "glue:BatchGetBlueprints",
                "glue:GetWorkflowRun",
                "glue:GetWorkflowRuns",
                "glue:ListCrawlers",
                "glue:ListWorkflows",
                "glue:GetJob",
                "glue:GetWorkflow",
                "glue:StartWorkflowRun"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::<account-id>:role/<role-name>"
        }
    ]
}
```

Autorizzazioni per i ruoli degli schemi

Di seguito sono riportate le autorizzazioni suggerite per il ruolo IAM utilizzato per creare un flusso di lavoro da un piano. Il ruolo deve avere una relazione di trust con `glue.amazonaws.com`.

> **Important**
>
> Nella policy riportata di seguito, sostituisci *<account-id>* con un account AWS valido e *<role-name>* con il nome del ruolo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateJob",
                "glue:GetCrawler",
                "glue:GetTrigger",
                "glue:DeleteCrawler",
                "glue:CreateTrigger",
```

```
                "glue:DeleteTrigger",
                "glue:DeleteJob",
                "glue:CreateWorkflow",
                "glue:DeleteWorkflow",
                "glue:GetJob",
                "glue:GetWorkflow",
                "glue:CreateCrawler"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::<account-id>:role/<role-name>"
        }
    ]
}
```

> **Note**
>
> Se i processi e i crawler nel flusso di lavoro assumono un ruolo diverso da questo, questa policy deve includere l'autorizzazione `iam:PassRole` su quel ruolo invece che sul ruolo del piano.

Sviluppo di schemi in AWS Glue

L'organizzazione potrebbe avere un set di casi di utilizzo ETL simili che potrebbero trarre vantaggio dalla possibilità di definire i parametri di un singolo flusso di lavoro in grado di gestirli tutti. A questo scopo, AWS Glue ti consente di definire dei progetti, che puoi utilizzare per generare i flussi di lavoro. Un progetto accetta i parametri, in modo che da un singolo progetto, un analista di dati possa creare diversi flussi di lavoro per gestire casi di utilizzo ETL simili. Una volta creato un progetto, puoi riutilizzarlo per reparti, team e progetti diversi.

Argomenti

- Aggiornamento di uno schema in AWS Glue
- Creazione di un flusso di lavoro da uno schema in AWS Glue
- Visualizzazione delle esecuzioni dello schema in AWS Glue

Panoramica degli schemi in AWS Glue

> **Note**
>
> La funzionalità blueprints (schemi) non è attualmente disponibile nelle seguenti Regioni della consoleAWS Glue: Asia Pacific (Giacarta) e Medio Oriente (Emirati Arabi Uniti).

I progetti AWS Glue offrono un modo per creare e condividere i flussi di lavoro AWS Glue. Quando esiste un processo ETL complesso che potrebbe essere utilizzato per casi d'uso simili, piuttosto che creare un flusso di lavoro AWS Glue per ogni caso d'uso, è possibile creare un singolo progetto.

Il piano specifica i processi e i crawler da includere in un flusso di lavoro e specifica i parametri che l'utente fornisce quando esegue il piano per creare un flusso di lavoro. L'uso di parametri consente a un singolo piano di generare flussi di lavoro per vari casi d'uso simili. Per ulteriori informazioni sui flussi di lavoro, consulta Panoramica di flussi di lavoro in AWS Glue.

Di seguito sono riportati esempi di casi d'uso per i piani:

- Vuoi partizionare un set di dati esistente. I parametri di input del piano sono i percorsi di origine e di destinazione Amazon Simple Storage Service (Amazon S3) e un elenco di colonne di partizione.
- Vuoi creare uno snapshot di una tabella Amazon DynamoDB in un archivio dati SQL come Amazon Redshift. I parametri di input per il progetto sono il nome della tabella DynamoDB e una connessione AWS Glue, che indica un cluster Amazon Redshift e un database di destinazione.
- Vuoi convertire i dati CSV in più percorsi Amazon S3 in Parquet. È consigliabile che il flusso di lavoro AWS Glue includa un crawler e un processo separati per ogni percorso. I parametri di input sono il database di destinazione in AWS Glue Data Catalog e un elenco di percorsi Amazon S3 delimitati da virgola. In questo caso, il numero di crawler e processi creati dal flusso di lavoro è variabile.

Componenti dello schema

Un piano è un archivio ZIP contenente i seguenti componenti:

- Uno script generatore di layout Python

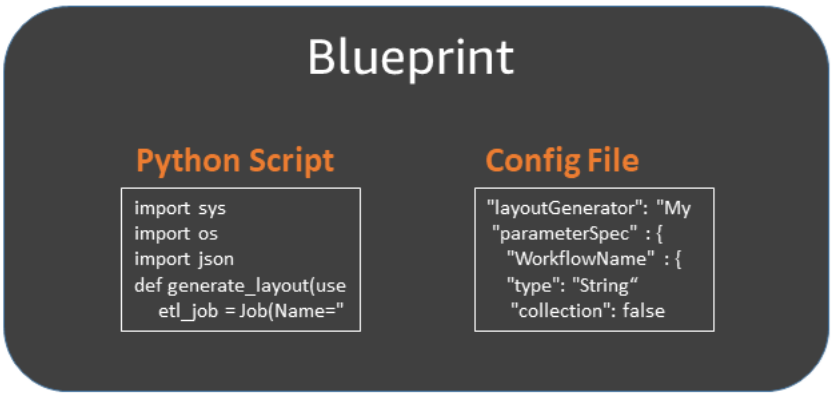
  Contiene una funzione che specifica il layout del flusso di lavoro: i crawler e i processi da creare per il flusso di lavoro, le proprietà del processo e del crawler e le dipendenze tra i processi e i crawler. La funzione accetta i parametri di progetto e restituisce una struttura del flusso di lavoro (oggetto JSON) che AWS Glue utilizza per generare il flusso di lavoro. Utilizzando uno script Python per generare il flusso di lavoro, puoi aggiungere la logica adatta ai tuoi casi d'uso.

- Un file di configurazione

  Specifica il nome completo della funzione Python che genera il layout del flusso di lavoro. Specifica inoltre i nomi, i tipi di dati e le altre proprietà di tutti i parametri del piano utilizzati dallo script.

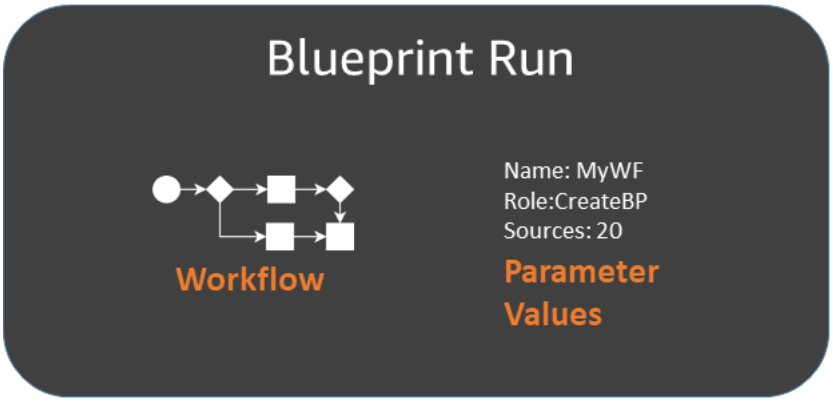
- (Facoltativo) Script ETL e file di supporto

  Come caso d'uso avanzato, è possibile definire i parametri della posizione degli script ETL utilizzati dai processi. Puoi includere i file di script di processo nell'archivio ZIP e specificare un parametro del piano per una posizione Amazon S3 in cui gli script devono essere copiati. Lo script generatore di layout può copiare gli script ETL nella posizione indicata e specificare tale posizione come proprietà della posizione dello script di processo. È inoltre possibile includere qualsiasi libreria o altri file di supporto, a condizione che lo script li gestisca.



Esecuzioni del piano

Quando crei un flusso di lavoro da un progetto, AWS Glue lo esegue, il che avvia un processo asincrono per creare il flusso di lavoro e i processi, i crawler e i trigger incapsulati dal flusso di lavoro. AWS Glue usa l'esecuzione del progetto per orchestrare la creazione del flusso di lavoro e dei suoi componenti. Puoi vedere lo stato del processo di creazione attraverso lo stato di esecuzione del piano. L'esecuzione del piano memorizza anche i valori forniti per i parametri del piano.



Puoi visualizzare le esecuzioni di un progetto utilizzando la console AWS Glue o la AWS Command Line Interface (AWS CLI). Durante la visualizzazione o la risoluzione dei problemi di un flusso di lavoro, puoi sempre tornare all'esecuzione del piano per visualizzare i valori dei parametri del piano utilizzati per creare il flusso di lavoro.

Ciclo di vita di uno schema

Gli schemi sono sviluppati, testati, registrati con AWS Glue ed eseguiti per creare flussi di lavoro. In genere tre utenti sono coinvolti nel ciclo di vita del piano.

Utente	Processi
Sviluppatore AWS Glue	• Scrive lo script del layout del flusso di lavoro e crea il file di configurazione. • Testa il progetto in locale utilizzando le librerie fornite dal servizio AWS Glue. • Crea un archivio ZIP dello script, del file di configura zione e dei file di supporto e pubblica l'archivio in una posizione in Amazon S3. • Aggiunge una policy del bucket al bucket Amazon S3 che concede le autorizzazioni di lettura per gli oggetti bucket all'account AWS dell'amministratore AWS Glue. • Concede le autorizzazioni di lettura IAM per l'archivio ZIP in Amazon S3 all'amministratore AWS Glue.
Amministratore di AWS Glue	• Registra il progetto con AWS Glue. AWS Glue crea una copia dell'archivio ZIP in una posizione Amazon S3 riservata.

Utente	Processi
	• Concede le autorizzazioni IAM per il piano agli analisti dei dati.
Analista dei dati	• Esegue il piano per creare un flusso di lavoro e fornisce i valori dei parametri del piano. Controlla lo stato di esecuzione del piano per assicurarsi che il flusso di lavoro e i relativi componenti siano stati generati correttamente. • Esegue e risolve i problemi relativi al flusso di lavoro. Prima di eseguire il flusso di lavoro, può verificarlo visualizzando il grafico di progettazione del flusso di lavoro nella console AWS Glue.

Consulta anche

- Sviluppo di schemi in AWS Glue
- Creazione di un flusso di lavoro da uno schema in AWS Glue
- Autorizzazioni per utenti e ruoli per gli schemi AWS Glue

Sviluppo di schemi in AWS Glue

In qualità di sviluppatore AWS Glue, puoi creare e pubblicare progetti che gli analisti di dati potranno utilizzare per generare flussi di lavoro.

Argomenti

- Esempi di schema

> Consulta anche
>
> - Panoramica degli schemi in AWS Glue

Panoramica sullo sviluppo di schemi

Il primo passo del processo di sviluppo consiste nell'identificare un caso d'uso comune che possa trarre vantaggio da un piano. Un tipico caso d'uso comporta un problema ETL ricorrente che ritieni debba essere risolto in modo generale. Quindi, progetta un piano che implementi il caso d'uso generalizzato e definisci i parametri di input del piano che insieme possono definire un caso d'uso specifico a partire dal caso d'uso generalizzato.

Un piano è costituito da un progetto che contiene un file di configurazione dei parametri del piano e uno script che definisce la proprietà di layout del flusso di lavoro da generare. Il layout definisce i processi e i crawler (o entità, nella terminologia dello script del piano) da creare.

Non è possibile specificare direttamente alcun trigger nello script di layout. È invece possibile scrivere codice per specificare le dipendenze tra i processi e i crawler creati dallo script. AWS Glue genera le attivazioni basate sulle specifiche delle dipendenze. L'output dello script di layout è un oggetto di flusso di lavoro che contiene le specifiche per tutte le entità del flusso di lavoro.

È possibile costruire l'oggetto flusso di lavoro utilizzando le seguenti librerie dei progetti AWS Glue:

- `awsglue.blueprint.base_resource`— Una libreria di risorse di base utilizzate dalle librerie.
- `awsglue.blueprint.workflow`— Una libreria per definire una classe di `Workflow`.
- `awsglue.blueprint.job`— Una libreria per definire una classe di `Job`.
- `awsglue.blueprint.crawler`— Una libreria per definire una classe di `Crawler`.

Le uniche altre librerie supportate per la generazione del layout sono quelle disponibili per la shell Python.

Prima di pubblicare il piano, è possibile utilizzare i metodi definiti nelle librerie dei piani per testarlo localmente.

Quando si è pronti a rendere il piano disponibile agli analisti dei dati, è possibile creare un pacchetto dello script, del file di configurazione dei parametri e di tutti i file di supporto, ad esempio script e librerie aggiuntivi, in un'unica risorsa distribuibile. Quindi si carica la risorsa su Amazon S3 e si chiede a un amministratore di registrarla su AWS Glue.

Per ulteriori informazioni su altri piani di esempio, consulta Progetto di schema di esempio e Esempi di schema.

Prerequisiti per lo sviluppo degli schemi

Per sviluppare i progetti, è bene avere familiarità con AWS Glue e con la scrittura di script per i processi Apache Spark ETL o di shell Python. È inoltre necessario completare le seguenti attività di configurazione.

- Scaricare quattro librerie Python AWS da utilizzare negli script di layout del piano.
- Configurare gli SDK di AWS.
- Impostare AWS CLI.

Scaricare le librerie Python

Scarica le seguenti librerie da GitHub e installale nel tuo progetto:

- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/base_resource.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/workflow.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/crawler.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/job.py

Configurare l'SDK Java AWS

Per l'SDK Java AWS, è necessario aggiungere un file `jar` che includa l'API per i piani.

1. Se non l'hai già fatto, configura l'SDK for Java AWS.
   - Per Java 1.x, segui le istruzioni in Impostare AWS SDK for Java nella Guida per gli sviluppatori di AWS SDK for Java.
   - Per Java 2.x, segui le istruzioni in Configurazione di AWS SDK for Java 2.x nella AWS SDK for Java 2.xGuida per gli sviluppatori di .

2. Scarica il file client `jar` che ha accesso alle API per i piani.

   - Per Java 1.x: s3://awsglue-custom-blueprints-preview-artifacts/awsglue-java-sdk-preview/AWSGlueJavaClient-1.11.x.jar
   - Per Java 2.x: s3://awsglue-custom-blueprints-preview-artifacts/awsglue-java-sdk-v2-preview/AwsJavaSdk-Glue-2.0.jar

3. Aggiungi il client `jar` nella parte anteriore del classpath Java per sovrascrivere il client AWS Glue fornito dall'SDK Java di AWS.

```
export CLASSPATH=<path-to-preview-client-jar>:$CLASSPATH
```

4. (Facoltativo) Testa l'SDK con la seguente applicazione Java. L'applicazione dovrebbe produrre un elenco vuoto.

   Sostituisci `accessKey` e `secretKey` con le tue credenziali e sostituisci `us-east-1` con la tua regione.

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.glue.AWSGlue;
import com.amazonaws.services.glue.AWSGlueClientBuilder;
import com.amazonaws.services.glue.model.ListBlueprintsRequest;

public class App{
    public static void main(String[] args) {
        AWSCredentials credentials = new BasicAWSCredentials("accessKey",
 "secretKey");
        AWSCredentialsProvider provider = new
 AWSStaticCredentialsProvider(credentials);
        AWSGlue glue = AWSGlueClientBuilder.standard().withCredentials(provider)
                .withRegion("us-east-1").build();
        ListBlueprintsRequest request = new
 ListBlueprintsRequest().withMaxResults(2);
        System.out.println(glue.listBlueprints(request));
    }
}
```

Configurazione dell'SDK Python AWS

I seguenti passaggi presuppongono che sul computer sia installato Python versione 2.7 o successiva o versione 3.6 o successiva.

1. Scarica il seguente file boto3 wheel. Se richiesto di aprire o salvare, salva il file. s3://awsglue-custom-blueprints-preview-artifacts/aws-python-sdk-preview/boto3-1.17.31-py2.py3-none-any.whl
2. Scarica il seguente file ruota botocore wheel: s3://awsglue-custom-blueprints-preview-artifacts/aws-python-sdk-preview/botocore-1.20.31-py2.py3-none-any.whl
3. Controlla la tua versione Python.

```
python --version
```

4. A seconda della versione di Python, immetti seguenti comandi (per Linux):

   - Per Python 2.7 o versioni successive.

```
python3 -m pip install --user virtualenv
source env/bin/activate
```

   - Per Python 3.6 o versioni successive.

```
python3 -m venv python-sdk-test
source python-sdk-test/bin/activate
```

5. Installa il file botocore wheel.

```
python3 -m pip install <download-directory>/botocore-1.20.31-py2.py3-none-any.whl
```

6. Installa il file boto3 wheel.

```
python3 -m pip install <download-directory>/boto3-1.17.31-py2.py3-none-any.whl
```

7. Configura le credenziali e la regione predefinita nei file `~/.aws/credentials` e `~/.aws/config`. Per ulteriori informazioni, consulta Configurazione della AWS CLI nella Guida per l'utente di AWS Command Line Interface.
8. (Facoltativo) Esegui il test della configurazione. I seguenti comandi devono restituire un elenco vuoto.

Sostituisci `us-east-1` con la tua regione.

```
$ python
>>> import boto3
>>> glue = boto3.client('glue', 'us-east-1')
>>> glue.list_blueprints()
```

Configurazione dell'anteprima della AWS CLI

1. Se non l'hai ancora fatto, installa e/o aggiorna AWS Command Line Interface (AWS CLI) sul computer. Il modo più semplice per eseguire questa operazione è utilizzare `pip`, l'utility di installazione Python:

   ```
   pip install awscli --upgrade --user
   ```

   Puoi trovare le istruzioni di installazione complete per AWS CLI qui: Installazione di AWS Command Line Interface.
2. Scarica il file wheel AWS CLI da: s3://awsglue-custom-blueprints-preview-artifacts/awscli-preview-build/awscli-1.19.31-py2.py3-none-any.whl
3. Installa il file wheel AWS CLI.

   ```
   python3 -m pip install awscli-1.19.31-py2.py3-none-any.whl
   ```
4. Esegui il comando `aws configure`. Configura le credenziali AWS (tra cui chiave di accesso e chiave segreta) e la ragione AWS. Puoi trovare informazioni sulla configurazione di AWS CLI qui: Configurazione di AWS CLI.
5. Testa AWS CLI. Il seguente comando dovrebbe restituire un elenco vuoto.

   Sostituisci `us-east-1` con la tua regione.

   ```
   aws glue list-blueprints --region us-east-1
   ```

Scrittura del codice dello schema

Ogni piano creato deve contenere almeno i seguenti file:

- Uno script di layout Python che definisce il flusso di lavoro. Lo script contiene una funzione che definisce le entità (processi e crawler) in un flusso di lavoro e le dipendenze tra di esse.
- Un file di configurazione, `blueprint.cfg`, che definisce:
  - Il percorso completo della funzione di definizione del layout del flusso di lavoro.
  - I parametri accettati dal piano.

Argomenti

Creazione dello script di layout dello schema

Lo script di layout del piano deve includere una funzione che genera le entità nel flusso di lavoro. Puoi nominare questa funzione come preferisci. AWS Glue utilizza il file di configurazione per determinare il nome completo della funzione.

La funzione di layout svolge le operazioni seguenti:

- (Facoltativo) Crea un'istanza della classe di `Job` per creare oggetti `Job` e passa argomenti come `Command` e `Role`. Si tratta di proprietà del processo da specificare se lo stai creando tramite la console o l'API AWS Glue.
- (Facoltativo) Crea un'istanza della classe di `Crawler` per creare oggetti `Crawler` e passa argomenti come il nome, il ruolo e la destinazione.
- Per indicare le dipendenze tra gli oggetti (entità del flusso di lavoro), passa gli argomenti aggiuntivi `DependsOn` e `WaitForDependencies` al `Job()` e al `Crawler()`. Questi argomenti sono descritti più avanti in questa sezione.
- Crea un'istanza della classe di `Workflow` per creare l'oggetto di flusso di lavoro che viene restituito ad AWS Glue, passando un argomento `Name`, un argomento `Entities` e un argomento facoltativo `OnSchedule`. L'argomento `Entities` specifica tutti i processi e i crawler da includere nel flusso di lavoro. Per sapere come costruire un oggetto `Entities`, vedi il progetto di esempio più avanti in questa sezione.
- Restituisce l'oggetto `Workflow`.

Per le definizioni delle classi `Job`, `Crawler` e`Workflow`, consulta Riferimento alle classi di schema AWS Glue.

La funzione di layout di deve accettare i seguenti argomenti.

Argomento	Descrizione
`user_params`	Dizionario Python di nomi e valori dei parametri del piano. Per ulteriori informazioni, consulta Specifica dei parametri dello schema.
`system_params`	Dizionario Python contenente due proprietà: `region` e `accountId` .

Ecco uno script generatore di layout di esempio in un file chiamato `Layout.py`:

```
import argparse
import sys
import os
import json
from awsglue.blueprint.workflow import *
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *


def generate_layout(user_params, system_params):

    etl_job = Job(Name="{}_etl_job".format(user_params['WorkflowName']),
                  Command={
                      "Name": "glueetl",
                      "ScriptLocation": user_params['ScriptLocation'],
                      "PythonVersion": "2"
                  },
                  Role=user_params['PassRole'])
    post_process_job = Job(Name="{}_post_process".format(user_params['WorkflowName']),
                           Command={
                               "Name": "pythonshell",
                               "ScriptLocation": user_params['ScriptLocation'],
                               "PythonVersion": "2"
                           },
                           Role=user_params['PassRole'],
                           DependsOn={
                               etl_job: "SUCCEEDED"
```

```
                            },
                            WaitForDependencies="AND")
    sample_workflow = Workflow(Name=user_params['WorkflowName'],
                            Entities=Entities(Jobs=[etl_job, post_process_job]))
    return sample_workflow
```

Lo script di esempio importa le librerie del piano richieste e include un `generate_layout` che genera un flusso di lavoro con due processi. Si tratta di uno script molto semplice. Uno script più complesso potrebbe impiegare logica e parametri aggiuntivi per generare un flusso di lavoro con molti processi e crawler, o anche un numero variabile di processi e crawler.

Utilizzo dell'argomento DependsOn

La'argomento `DependsOn` è una rappresentazione dizionario di una dipendenza di questa entità su altre entità all'interno del flusso di lavoro. Presenta il seguente formato.

```
DependsOn = {dependency1 : state, dependency2 : state, ...}
```

Le chiavi in questo dizionario rappresentano il riferimento all'oggetto dell'entità (non il nome), mentre i valori sono stringhe che corrispondono allo stato da controllare. AWS Glue deduce le attivazioni appropriate. Per gli stati validi, consulta Struttura Condition.

Ad esempio, un processo potrebbe dipendere dal completamento corretto di un crawler. Se definisci un oggetto crawler denominato `crawler2` come segue:

```
crawler2 = Crawler(Name="my_crawler", ...)
```

Allora un oggetto dipendente da `crawler2` includerebbe un argomento del costruttore come:

```
DependsOn = {crawler2 : "SUCCEEDED"}
```

Ad esempio:

```
job1 = Job(Name="Job1", ..., DependsOn = {crawler2 : "SUCCEEDED", ...})
```

Se `DependsOn` viene omesso per un'entità, tale entità dipende dal trigger di avvio del flusso di lavoro.

Utilizzo dell'argomento WaitForDependencies

L'argomento `WaitForDependencies` definisce se un processo o un'entità crawler deve attendere fino a che tutte le entità da cui dipende sono complete o fino a quando è completa una qualsiasi.

I valori consentiti sono "AND" o "ANY".

Utilizzo dell'argomento OnSchedule

L'argomento `OnSchedule` per il costruttore della classe `Workflow` è un'espressione `cron` che indica la definizione del trigger iniziale per un flusso di lavoro.

Se questo argomento è specificato, AWS Glue crea un'attivazione di pianificazione con la pianificazione corrispondente. Se non è specificato, il trigger di attivazione del flusso di lavoro è un trigger on demand.

Creare il file di configurazione

Il file di configurazione del piano è un file obbligatorio che definisce il punto di ingresso dello script per la generazione del flusso di lavoro e i parametri accettati dal piano. Il deve essere denominato `blueprint.cfg`.

Segue una configurazione di esempio.

```
{
    "layoutGenerator": "DemoBlueprintProject.Layout.generate_layout",
    "parameterSpec" : {
           "WorkflowName" : {
                "type": "String",
                "collection": false
           },
           "WorkerType" : {
                "type": "String",
                "collection": false,
                "allowedValues": ["G1.X", "G2.X"],
                "defaultValue": "G1.X"
           },
           "Dpu" : {
                "type" : "Integer",
                "allowedValues" : [2, 4, 6],
                "defaultValue" : 2
           },
           "DynamoDBTableName": {
                "type": "String",
```

```
                "collection" : false
            },
            "ScriptLocation" : {
                "type": "String",
                "collection": false
     }
    }
}
```

La proprietà `layoutGenerator` specifica il nome completo della funzione nello script che genera il layout.

La proprietà `parameterSpec` specifica i parametri accettati da questo piano. Per ulteriori informazioni, consulta Specifica dei parametri dello schema.

> **Important**
>
> Il file di configurazione deve includere il nome del flusso di lavoro come parametro del piano oppure è necessario generare un nome di flusso di lavoro univoco nello script di layout.

Specifica dei parametri dello schema

Il file di configurazione contiene le specifiche dei parametri del piano in un oggetto JSON `parameterSpec`. `parameterSpec` contiene uno o più oggetti parametro.

```
"parameterSpec": {
    "<parameter_name>": {
      "type": "<parameter-type>",
      "collection": true|false,
      "description": "<parameter-description>",
      "defaultValue": "<default value for the parameter if value not specified>"
      "allowedValues": "<list of allowed values>"
    },
    "<parameter_name>": {
       ...
    }
  }
```

Di seguito sono riportate le regole per la codifica di ogni oggetto parametro:

- Il nome e il `type` del parametro sono obbligatori. Tutte le altre proprietà sono facoltative.

- Se si specifica la proprietà `defaultValue`, il parametro è facoltativo. In caso contrario, il parametro è obbligatorio e l'analista dei dati che sta creando un flusso di lavoro dal piano deve fornire un valore per esso.
- Se si imposta la proprietà `collection` su `true`, il parametro può assumere un insieme di valori. Le raccolte possono essere di qualsiasi tipo di dati.
- Se specifichi `allowedValues`, la console AWS Glue mostra un elenco a discesa di valori tra cui l'analista dei dati può scegliere durante la creazione di un flusso di lavoro dal progetto.

Di seguito sono elencati i valori consentiti per `type`:

Tipo di dati dei parametri	Note
`String`	-
`Integer`	-
`Double`	-
`Boolean`	I valori possibili sono `true` e `false`. Genera una casella di controllo nella pagina Create a workflow from <blueprint> (Crea un flusso di lavoro da <blueprint>), sulla console AWS Glue.
`S3Uri`	Completa il percorso Amazon S3, iniziando con `s3://`. Genera un campo di testo e un pulsante Browse (Sfoglia) nella pagina Create a workflow from <blueprint> (Crea un flusso di lavoro da <blueprint>).
`S3Bucket`	Solo il nome del bucket Amazon S3. Genera un selettore bucket nella scheda Create a workflow from <blueprint> (Crea un flusso di lavoro da <blueprint>).
`IAMRoleArn`	L'Amazon Resource Name (ARN) di un ruolo AWS Identity and Access Management (IAM). Genera un selettore ruolo nella pagina Create a workflow from <blueprint> (Crea un flusso di lavoro da <blueprint>).
`IAMRoleName`	Nome di un ruolo IAM. Genera un selettore ruolo nella pagina Create a workflow from <blueprint> (Crea un flusso di lavoro da <blueprint>).

Progetto di schema di esempio

La conversione del formato dei dati è un caso d'uso frequente di estrazione, trasformazione e caricamento (ETL). Nei carichi di lavoro analitici tipici, i formati di file basati su colonne come Parquet o ORC sono preferiti rispetto ai formati di testo come CSV o JSON. Questo piano di esempio consente di convertire i dati da CSV/JSON/ecc. in Parquet per i file su Amazon S3.

Questo piano accetta un elenco di percorsi S3 definiti da un parametro del piano, converte i dati in formato Parquet e li scrive nella posizione S3 specificata da un altro parametro del piano. Lo script di layout crea un crawler e un processo per ogni percorso. Lo script di layout carica anche lo script ETL in `Conversion.py` in un bucket S3 specificato da un altro parametro del piano. Lo script di layout specifica quindi lo script caricato come script ETL per ogni processo. L'archivio ZIP del progetto contiene lo script di layout, lo script ETL e il file di configurazione del piano.

Per ulteriori informazioni su altri piani di esempio, consulta Esempi di schema.

Di seguito è riportato lo script di layout, nel file `Layout.py`.

```
from awsglue.blueprint.workflow import *
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *
import boto3

s3_client = boto3.client('s3')

# Ingesting all the S3 paths as Glue table in parquet format
def generate_layout(user_params, system_params):
    #Always give the full path for the file
    with open("ConversionBlueprint/Conversion.py", "rb") as f:
        s3_client.upload_fileobj(f, user_params['ScriptsBucket'], "Conversion.py")
    etlScriptLocation = "s3://{}/Conversion.py".format(user_params['ScriptsBucket'])

    crawlers = []
    jobs = []
    workflowName = user_params['WorkflowName']
    for path in user_params['S3Paths']:
      tablePrefix = "source_"
      crawler = Crawler(Name="{}_crawler".format(workflowName),
                        Role=user_params['PassRole'],
                        DatabaseName=user_params['TargetDatabase'],
                        TablePrefix=tablePrefix,
                        Targets= {"S3Targets": [{"Path": path}]})
```

```
      crawlers.append(crawler)
      transform_job = Job(Name="{}_transform_job".format(workflowName),
                          Command={"Name": "glueetl",
                                   "ScriptLocation": etlScriptLocation,
                                   "PythonVersion": "3"},
                          Role=user_params['PassRole'],
                          DefaultArguments={"--database_name":
 user_params['TargetDatabase'],
                                            "--table_prefix": tablePrefix,
                                            "--region_name": system_params['region'],
                                            "--output_path":
 user_params['TargetS3Location']},
                          DependsOn={crawler: "SUCCEEDED"},
                          WaitForDependencies="AND")
      jobs.append(transform_job)
   conversion_workflow = Workflow(Name=workflowName, Entities=Entities(Jobs=jobs,
 Crawlers=crawlers))
   return conversion_workflow
```

Di seguito è riportato il corrispondente file `blueprint.cfg` di configurazione del piano.

```
{
    "layoutGenerator": "ConversionBlueprint.Layout.generate_layout",
    "parameterSpec" : {
        "WorkflowName" : {
            "type": "String",
            "collection": false,
            "description": "Name for the workflow."
        },
        "S3Paths" : {
            "type": "S3Uri",
            "collection": true,
            "description": "List of Amazon S3 paths for data ingestion."
        },
        "PassRole" : {
            "type": "IAMRoleName",
            "collection": false,
            "description": "Choose an IAM role to be used in running the job/crawler"
        },
        "TargetDatabase": {
            "type": "String",
            "collection" : false,
            "description": "Choose a database in the Data Catalog."
```

```
        },
        "TargetS3Location": {
            "type": "S3Uri",
            "collection" : false,
            "description": "Choose an Amazon S3 output path: ex:s3://<target_path>/."
        },
        "ScriptsBucket": {
            "type": "S3Bucket",
            "collection": false,
            "description": "Provide an S3 bucket name(in the same AWS Region) to store
 the scripts."
        }
    }
}
```

Il seguente script nel file `Conversion.py` è lo script ETL caricato. Nota che durante la conversione mantiene lo schema di partizionamento.

```
import sys
from pyspark.sql.functions import *
from pyspark.context import SparkContext
from awsglue.transforms import *
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions
import boto3

args = getResolvedOptions(sys.argv, [
    'JOB_NAME',
    'region_name',
    'database_name',
    'table_prefix',
    'output_path'])
databaseName = args['database_name']
tablePrefix = args['table_prefix']
outputPath = args['output_path']

glue = boto3.client('glue', region_name=args['region_name'])

glue_context = GlueContext(SparkContext.getOrCreate())
spark = glue_context.spark_session
job = Job(glue_context)
job.init(args['JOB_NAME'], args)
```

```
def get_tables(database_name, table_prefix):
    tables = []
    paginator = glue.get_paginator('get_tables')
    for page in paginator.paginate(DatabaseName=database_name, Expression=table_prefix
+"*"):
        tables.extend(page['TableList'])
    return tables

for table in get_tables(databaseName, tablePrefix):
    tableName = table['Name']
    partitionList = table['PartitionKeys']
    partitionKeys = []
    for partition in partitionList:
        partitionKeys.append(partition['Name'])

    # Create DynamicFrame from Catalog
    dyf = glue_context.create_dynamic_frame.from_catalog(
        name_space=databaseName,
        table_name=tableName,
        additional_options={
            'useS3ListImplementation': True
        },
        transformation_ctx='dyf'
    )

    # Resolve choice type with make_struct
    dyf = ResolveChoice.apply(
        frame=dyf,
        choice='make_struct',
        transformation_ctx='resolvechoice_' + tableName
    )

    # Drop null fields
    dyf = DropNullFields.apply(
        frame=dyf,
        transformation_ctx="dropnullfields_" + tableName
    )

    # Write DynamicFrame to S3 in glueparquet
    sink = glue_context.getSink(
        connection_type="s3",
        path=outputPath,
        enableUpdateCatalog=True,
```

```
        partitionKeys=partitionKeys
    )
    sink.setFormat("glueparquet")

    sink.setCatalogInfo(
        catalogDatabase=databaseName,
        catalogTableName=tableName[len(tablePrefix):]
    )
    sink.writeFrame(dyf)

job.commit()
```

> **Note**
>
> Solo due percorsi Amazon S3 possono essere forniti come input per il piano di esempio. Questo perché le attivazioni di AWS Glue possono richiamare solo due operazioni crawler.

Test di uno schema

Durante lo sviluppo del codice, è necessario eseguire test locali per verificare che il layout del flusso di lavoro sia corretto.

Il test locale non genera processi, crawler o attivazioni AWS Glue. Invece, esegui lo script di layout localmente e utilizzi i metodi `to_json()` e `validate()` per stampare gli oggetti e trovare gli errori. Questi metodi sono disponibili in tutte e tre le classi definite nelle librerie.

Sono disponibili due modi per gestire gli argomenti `user_params` e `system_params` cheAWS Glue passa alla funzione di layout. Il codice test-bench può creare un dizionario di valori di esempio dei parametri del piano e passarli alla funzione di layout come argomento `user_params`. In alternativa, puoi rimuovere i riferimenti a `user_params` e sostituirli con stringhe hardcoded.

Se il tuo codice utilizza proprietà `region` e `accountId` nell'argomento `system_params`, puoi passare nel tuo dizionario per `system_params`.

Per testare un piano

1. Avvia un interprete Python in una directory con le librerie o carica i file del piano e le librerie fornite nel tuo ambiente di sviluppo integrato (IDE) preferito.
2. Assicurati che il tuo codice importi le librerie fornite.

3. Aggiungi codice alla tua funzione di layout per chiamare `validate()` o `to_json()` su qualsiasi entità o sull'oggetto `Workflow`. Ad esempio, se il codice crea un oggetto `Crawler` denominato `mycrawler`, è possibile chiamare `validate()` come segue.

   ```
   mycrawler.validate()
   ```

   Puoi stampare `mycrawler` come segue:

   ```
   print(mycrawler.to_json())
   ```

   Se chiami `to_json` su un oggetto, non è necessario chiamare anche `validate()`, perché `to_json()` chiama `validate()`.

   È molto utile chiamare questi metodi sull'oggetto flusso di lavoro. Supponendo che lo script denomini l'oggetto flusso di lavoro `my_workflow`, convalida e stampa l'oggetto flusso di lavoro come segue.

   ```
   print(my_workflow.to_json())
   ```

   Per ulteriori informazioni su `to_json()` e `validate()`, consulta Metodi di classe.

   Puoi anche importare `pprint` e stampare con precisione l'oggetto flusso di lavoro, come illustrato nell'esempio più avanti in questa sezione.

4. Esegui il codice, correggi gli errori e infine rimuovi tutte le chiamate a `validate()` o `to_json()`.

Example

L'esempio seguente mostra come costruire un dizionario di parametri di esempio del piano e passarli come argomento `user_params` alla funzione di layout `generate_compaction_workflow`. Viene inoltre illustrato come stampare con precisione l'oggetto flusso di lavoro generato.

```
from pprint import pprint
from awsglue.blueprint.workflow import *
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *

USER_PARAMS = {"WorkflowName": "compaction_workflow",
```

```
                "ScriptLocation": "s3://awsexamplebucket1/scripts/threaded-
compaction.py",
                "PassRole": "arn:aws:iam::111122223333:role/GlueRole-ETL",
                "DatabaseName": "cloudtrial",
                "TableName": "ct_cloudtrail",
                "CoalesceFactor": 4,
                "MaxThreadWorkers": 200}


def generate_compaction_workflow(user_params: dict, system_params: dict) -> Workflow:
    compaction_job = Job(Name=f"{user_params['WorkflowName']}_etl_job",
                         Command={"Name": "glueetl",
                                  "ScriptLocation": user_params['ScriptLocation'],
                                  "PythonVersion": "3"},
                         Role="arn:aws:iam::111122223333:role/
AWSGlueServiceRoleDefault",
                         DefaultArguments={"DatabaseName": user_params['DatabaseName'],
                                           "TableName": user_params['TableName'],
                                           "CoalesceFactor":
 user_params['CoalesceFactor'],
                                           "max_thread_workers":
 user_params['MaxThreadWorkers']})

    catalog_target = {"CatalogTargets": [{"DatabaseName": user_params['DatabaseName'],
 "Tables": [user_params['TableName']]}]}

    compacted_files_crawler = Crawler(Name=f"{user_params['WorkflowName']}_post_crawl",
                                      Targets = catalog_target,
                                      Role=user_params['PassRole'],
                                      DependsOn={compaction_job: "SUCCEEDED"},
                                      WaitForDependencies="AND",
                                      SchemaChangePolicy={"DeleteBehavior": "LOG"})

    compaction_workflow = Workflow(Name=user_params['WorkflowName'],
                                   Entities=Entities(Jobs=[compaction_job],

 Crawlers=[compacted_files_crawler]))
    return compaction_workflow

generated = generate_compaction_workflow(user_params=USER_PARAMS, system_params={})
gen_dict = generated.to_json()

pprint(gen_dict)
```

Pubblicazione di uno schema

Dopo aver sviluppato un piano, devi caricarlo su Amazon S3. Devi disporre delle autorizzazioni di scrittura sul bucket Amazon S3 utilizzato per pubblicare il piano. Devi inoltre assicurarti che l'amministratore AWS Glue, che registrerà il progetto disponga dell'accesso in lettura al bucket Amazon S3. Per suggerimenti sulle policy di autorizzazione AWS Identity and Access Management (IAM) per utenti e ruoli per i progetti AWS Glue, consulta Autorizzazioni per utenti e ruoli per gli schemi AWS Glue.

Per pubblicare un piano

1. Crea gli script, le risorse e il file di configurazione del piano necessari.
2. Aggiungi tutti i file a un archivio ZIP e carica il file ZIP su Amazon S3. Utilizza un bucket S3 che si trova nella regione in cui gli utenti registreranno ed eseguiranno il piano.

   È possibile creare un file ZIP dalla riga di comando utilizzando il comando seguente.

   ```
   zip -r folder.zip folder
   ```

3. Aggiungi una policy del bucket che conceda le autorizzazioni di lettura all'account AWS desiderato. Di seguito è riportata una policy di esempio.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "AWS": "arn:aws:iam::111122223333:root"
               },
               "Action": "s3:GetObject",
               "Resource": "arn:aws:s3:::my-blueprints/*"
           }
       ]
   }
   ```

4. Concedi l'autorizzazione `s3:GetObject` IAM sul bucket Amazon S3 all'amministratore AWS Glue o a chiunque registrerà i progetti. Per un esempio di policy da concedere agli amministratori, consulta Autorizzazioni per i progetti per l'amministratore di AWS Glue per gli schemi.

Dopo aver completato il test locale del progetto, potresti anche voler testare un progetto su AWS Glue. Per testare un progetto su AWS Glue, questo deve essere registrato. È possibile limitare chi vede il piano registrato utilizzando l'autorizzazione IAM o account di test separati.

> Consulta anche:
>
> - Registrazione di uno schema in AWS Glue

Riferimento alle classi di schema AWS Glue

Le librerie per i piani AWS Glue definiscono tre classi da utilizzare nello script di layout del flusso di lavoro: `Job`, `Crawler` e `Workflow`.

Argomenti

- Classe di processo
- Classe di crawler
- Classe di flusso di lavoro
- Metodi di classe

Classe di processo

La classe `Job` rappresenta un processo ETL AWS Glue.

Argomenti dei costruttori obbligatori

Di seguito sono illustrati gli argomenti dei costruttori obbligatori per la classe di `Job`.

Nome argomento	Type (Tipo)	Descrizione
`Name`	`str`	Nome da assegnare al processo. AWS Glue aggiunge un suffisso generato casualmente al nome per distinguere il processo da quelli creati da altre esecuzioni del progetto.
`Role`	`str`	L'Amazon Resource Name (ARN) del ruolo che deve assumere il processo durante l'esecuzione.

Nome argomento	Type (Tipo)	Descrizione
`Command`	`dict`	Comando del processo, come specificato nella documentazione API in JobCommand struttura.

Argomenti dei costruttori facoltativi

Di seguito sono illustrati gli argomenti dei costruttori facoltativi per la classe di `Job`.

Nome argomento	Type (Tipo)	Descrizione
`DependsOn`	`dict`	Elenco delle entità del flusso di lavoro da cui dipende il processo. Per ulteriori informazioni, consulta Utilizzo dell'argomento DependsOn.
`WaitForDependencies`	`str`	Indica se il processo deve attendere fino a che tutte le entità da cui dipende sono complete prima dell'esecuzione o fino a quando è completa una qualsiasi. Per ulteriori informazioni, consulta Utilizzo dell'argomento WaitForDependencies. Ometti se il processo dipende da una sola entità.
(Proprietà processo)	-	Qualsiasi proprietà del processo elencate in Struttura del processo nella documentazione API AWS Glue (eccetto `CreatedOn` e `LastModifiedOn`).

Classe di crawler

La classe `Crawler` rappresenta un crawler AWS Glue.

Argomenti dei costruttori obbligatori

Di seguito sono illustrati gli argomenti dei costruttori obbligatori per la classe di `Crawler`.

Nome argomento	Type (Tipo)	Descrizione
`Name`	`str`	Nome da assegnare al crawler. AWS Glue aggiunge un suffisso generato casualmente al nome per distinguere il crawler da quelli creati da altre esecuzioni del progetto.
`Role`	`str`	ARN del ruolo che il crawler deve assumere durante l'esecuzione.
`Targets`	`dict`	Raccolta di destinazioni da sottoporre al crawling. Gli argomenti dei costruttori della classe `Targets` sono definiti in CrawlerTargets struttura nella documentazione API. Tutti gli argomenti dei costruttori `Targets` sono facoltativi, ma è necessario passarne almeno uno.

Argomenti dei costruttori facoltativi

Di seguito sono illustrati gli argomenti dei costruttori facoltativi per la classe di `Crawler`.

Nome argomento	Type (Tipo)	Descrizione
`DependsOn`	`dict`	Elenco delle entità del flusso di lavoro da cui dipende il crawler. Per ulteriori informazioni, consulta Utilizzo dell'argomento DependsOn.
`WaitForDependencies`	`str`	Indica se il crawler deve attendere fino a che tutte le entità da cui dipende sono complete prima dell'esecuzione o fino a quando è completa una qualsiasi. Per ulteriori informazioni, consulta Utilizzo dell'argomento WaitForDependencies. Ometti se il crawler dipende da una sola entità.

Nome argomento	Type (Tipo)	Descrizione
(Proprietà dei crawler)	-	Qualsiasi proprietà del crawler elencata in Struttura dei crawler nella documentazione API AWS Glue, con le seguenti eccezioni: • `State` • `CrawlElapsedTime` • `CreationTime` • `LastUpdated` • `LastCrawl` • `Version`

Classe di flusso di lavoro

La classe `Workflow` rappresenta un flusso di lavoro AWS Glue. Lo script layout del flusso di lavoro restituisce un `Workflow`. AWS Glue crea un flusso di lavoro basato su questo oggetto.

Argomenti dei costruttori obbligatori

Di seguito sono illustrati gli argomenti dei costruttori obbligatori per la classe di `Workflow`.

Nome argomento	Type (Tipo)	Descrizione
`Name`	`str`	Nome da assegnare al flusso di lavoro.
`Entities`	`Entities`	Insieme di entità (processi e crawler) da includere nel flusso di lavoro. Il costruttore di classi `Entities` accetta un argomento `Jobs`, che è un elenco di oggetti `Job` e un argomento `Crawlers`, che è un elenco di oggetti `Crawler`.

Argomenti dei costruttori facoltativi

Di seguito sono illustrati gli argomenti dei costruttori facoltativi per la classe di `Workflow`.

Nome argomento	Type (Tipo)	Descrizione
`Description`	`str`	Per informazioni, consultare Struttura flusso di lavoro.
`DefaultRunProperties`	`dict`	Per informazioni, consultare Struttura flusso di lavoro.
`OnSchedule`	`str`	Un'espressione `cron`.

Metodi di classe

Tutte e tre le classi includono i seguenti metodi.

validate()

Convalida le proprietà dell'oggetto e, se vengono rilevati errori, genera un messaggio ed esce. Non genera alcun output se non ci sono errori. Per la classe di `Workflow`, si richiama su ogni entità nel flusso di lavoro.

to_json()

Serializza l'oggetto in JSON. Chiama anche `validate()`. Per la classe di `Workflow`, l'oggetto JSON include elenchi di processi e crawler e un elenco di trigger generati dalle specifiche di dipendenza del processo e del crawler.

Esempi di schema

Nel Repository Github per i progetti AWS Glue, sono disponibili diversi progetti di esempio . Questi esempi sono solo di riferimento e non sono destinati all'utilizzo.

I titoli dei progetti di esempio sono:

- Compattazione: questo piano crea un lavoro che compatta i file di input in blocchi più grandi in base alla dimensione di file desiderata.
- Conversione: questo piano converte i file di input in vari formati di file standard in formato Apache Parquet, ottimizzato per i carichi di lavoro analitici.
- Crawling di posizioni Amazon S3: questo piano esegue il crawling di più posizioni Amazon S3 per aggiungere tabelle di metadati al catalogo dati.

- Connessione personalizzata a Data Catalog: questo piano accede ai datastore utilizzando connettori personalizzati AWS Glue, legge i registri e popola le definizioni di tabella nell'AWS Glue Data Catalog in base allo schema del registro.
- Codifica: questo piano converte i file non UTF in file codificati UTF.
- Partizionamento: questo piano crea un processo di partizionamento che inserisce i file di output in partizioni basate su chiavi di partizione specifiche.
- Importazione di dati Amazon S3 in una tabella DynamoDB: questo progetto importa i dati da Amazon S3 in una tabella DynamoDB.
- Tabella standard da regolare: questo progetto importa una tabella AWS Glue Data Catalog in una tabella Lake Formation.

Registrazione di uno schema in AWS Glue

Dopo che lo sviluppatore di AWS Glue ha codificato il progetto e caricato un archivio in formato ZIP su Amazon Simple Storage Service (Amazon S3), un amministratore di AWS Glue dovrà registrarlo. La registrazione del piano lo rende disponibile per l'uso.

Quando si registra un progetto, AWS Glue ne copia l'archivio in una posizione Amazon S3 riservata. È quindi possibile eliminare l'archivio dalla posizione di caricamento.

Per registrare un piano, hai bisogno delle autorizzazioni di lettura per la posizione Amazon S3 che contiene l'archivio caricato. È inoltre necessario disporre dell'autorizzazione AWS Identity and Access Management (IAM) `glue:CreateBlueprint`. Per le autorizzazioni suggerite da concedere a un amministratore di AWS Glue che deve registrare, visualizzare e gestire i progetti, consulta Autorizzazioni per i progetti per l'amministratore di AWS Glue per gli schemi.

Puoi registrare un progetto utilizzando la console AWS Glue, l'API AWS Glue o la AWS Command Line Interface (AWS CLI).

Per registrare un piano (console)

1. Accertati di disporre delle autorizzazioni di lettura (`s3:GetObject`) per l'archivio ZIP del piano in Amazon S3.
2. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.

   Accedi come un utente che dispone delle autorizzazioni per registrare un piano. Passa alla stessa regione AWS del bucket Amazon S3 che contiene l'archivio ZIP del piano.

3. Nel pannello di navigazione seleziona schemi. Quindi, nella pagina schemi, seleziona Add blueprint (aggiungi schema).
4. Immetti un nome e, facoltativamente, una descrizione.
5. Per ZIP archive location (S3) (Posizione archivio ZIP [S3]), inserisci il percorso Amazon S3 dell'archivio ZIP del piano caricato. Includi il nome del file di archivio nel percorso e inizia il percorso con `s3://`.
6. (Facoltativo) Aggiungi uno o più tag.
7. Scegli Add blueprint (Aggiungi piano).

   La pagina schemi restituisce e mostra che lo stato del piano è `CREATING`. Seleziona il pulsante di aggiornamento fino a quando lo stato non cambia in `ACTIVE` o `FAILED`.
8. Se lo stato è `FAILED`, seleziona il piano e nella scheda Actions (Operazioni), scegli View (Visualizza).

   La pagina dei dettagli mostra il motivo dell'errore. Se il messaggio dell'errore indica che è impossibile accedere all'oggetto nella posizione... o che è negato l'accesso sull'oggetto nella posizione..., verifica i requisiti seguenti:

   - L'utente con cui hai effettuato l'accesso deve disporre dell'autorizzazione di lettura per l'archivio ZIP del piano in Amazon S3.
   - Il bucket Amazon S3 che contiene l'archivio ZIP deve disporre di una policy di bucket che conceda l'autorizzazione di lettura sull'oggetto al tuo ID account AWS. Per ulteriori informazioni, consulta Sviluppo di schemi in AWS Glue.
   - Il bucket Amazon S3 che stai utilizzando deve trovarsi nella stessa regione di quella alla quale hai eseguito l'accesso sulla console.
9. Assicurati che gli analisti dei dati dispongano delle autorizzazioni per il piano.

   La policy IAM suggerita per gli analisti di dati è mostrata in Autorizzazioni per gli schemi per l'analista dati. Questa policy concede `glue:GetBlueprint` su qualsiasi risorsa. Se i criteri sono più granulari a livello di risorsa, concedi agli analisti di dati le autorizzazioni per questa risorsa appena creata.

Per registrare un piano (AWS CLI)

1. Inserisci il comando seguente.

```
aws glue create-blueprint --name <blueprint-name> [--description <description>] --
blueprint-location s3://<s3-path>/<archive-filename>
```

2. Immetti il seguente comando per verificare lo stato del piano. Ripeti il comando fino a quando lo stato non diventa `ACTIVE` o `FAILED`.

```
aws glue get-blueprint --name <blueprint-name>
```

Se lo stato è `FAILED` e il messaggio dell'errore indica che è impossibile accedere all'oggetto nella posizione... o che è negato l'accesso sull'oggetto nella posizione..., verifica i requisiti seguenti:

- L'utente con cui hai effettuato l'accesso deve disporre dell'autorizzazione di lettura per l'archivio ZIP del piano in Amazon S3.
- Il bucket Amazon S3 che contiene l'archivio ZIP deve disporre di una policy di bucket che conceda l'autorizzazione di lettura sull'oggetto al tuo ID account AWS. Per ulteriori informazioni, consulta Pubblicazione di uno schema.
- Il bucket Amazon S3 che stai utilizzando deve trovarsi nella stessa regione di quella alla quale hai eseguito l'accesso sulla console.

Consulta anche:

- Panoramica degli schemi in AWS Glue

Visualizzazione degli schemi in AWS Glue

Visualizza un piano per esaminare la descrizione, lo stato e le specifiche dei parametri del piano e scaricare l'archivio ZIP del piano.

Puoi visualizzare un progetto utilizzando la console AWS Glue, l'API AWS Glue o la AWS Command Line Interface (AWS CLI).

Per visualizzare un piano (console)

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.

2. Nel pannello di navigazione scegli schemi.
3. Nella pagina schemi seleziona uno schema. Quindi nel menu Actions (Operazioni), scegli View (Visualizza).

Per visualizzare un piano (AWS CLI)

- Immetti il comando seguente per visualizzare solo il nome, la descrizione e lo stato del piano. Sostituisci *<blueprint-name>* con il nome del piano da visualizzare.

```
aws glue get-blueprint --name <blueprint-name>
```

L'output dei comandi è simile al seguente.

```
{
    "Blueprint": {
        "Name": "myDemoBP",
        "CreatedOn": 1587414516.92,
        "LastModifiedOn": 1587428838.671,
        "BlueprintLocation": "s3://awsexamplebucket1/demo/
DemoBlueprintProject.zip",
        "Status": "ACTIVE"
    }
}
```

Immetti il seguente comando per visualizzare anche le specifiche di parametro.

```
aws glue get-blueprint --name <blueprint-name>  --include-parameter-spec
```

L'output dei comandi è simile al seguente.

```
{
    "Blueprint": {
        "Name": "myDemoBP",
        "CreatedOn": 1587414516.92,
        "LastModifiedOn": 1587428838.671,
        "ParameterSpec": "{\"WorkflowName\":{\"type\":\"String\",\"collection
\":false,\"description\":null,\"defaultValue\":null,\"allowedValues\":null},
\"PassRole\":{\"type\":\"String\",\"collection\":false,\"description\":null,
\"defaultValue\":null,\"allowedValues\":null},\"DynamoDBTableName\":{\"type
```

```
\":\"String\",\"collection\":false,\"description\":null,\"defaultValue\":null,
\"allowedValues\":null},\"ScriptLocation\":{\"type\":\"String\",\"collection
\":false,\"description\":null,\"defaultValue\":null,\"allowedValues\":null}}",
        "BlueprintLocation": "s3://awsexamplebucket1/demo/
DemoBlueprintProject.zip",
        "Status": "ACTIVE"
    }
}
```

Aggiungi l'argomento `--include-blueprint` per includere nell'output un URL che è possibile incollare nel browser per scaricare l'archivio ZIP del progetto archiviato in AWS Glue.

Consulta anche:

- Panoramica degli schemi in AWS Glue

Aggiornamento di uno schema in AWS Glue

Puoi aggiornare un piano se si hai uno script di layout revisionato, un set di parametri del piano revisionato o file di supporto revisionati. L'aggiornamento di un piano crea una nuova versione.

L'aggiornamento di un piano non influisce sui flussi di lavoro esistenti creati dal piano.

Puoi aggiornare un progetto utilizzando la console AWS Glue, l'API AWS Glue o la AWS Command Line Interface (AWS CLI).

La procedura seguente presuppone che lo sviluppatore di AWS Glue abbia creato e caricato un archivio in formato ZIP del progetto aggiornato su Amazon S3.

Per aggiornare un piano (console)

1. Accertati di disporre delle autorizzazioni di lettura (`s3:GetObject`) per l'archivio ZIP del piano in Amazon S3.
2. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.

   Accedi come utente che dispone delle autorizzazioni per aggiornare un piano. Passa alla stessa regione AWS del bucket Amazon S3 che contiene l'archivio ZIP del piano.
3. Nel pannello di navigazione scegli schemi.

4. Nella pagina schemi, seleziona un piano e nella scheda Actions (operazioni) scegli Edit (modifica).
5. Nella pagina Edit a blueprint (Modifica un piano), aggiorna la Description (Descrizione) del piano o la ZIP archive location (S3) (Posizione dell'archivio ZIP [S3]). Assicurati di includere il nome del file di archivio nel percorso.
6. Seleziona Save (Salva).

   La pagina schemi restituisce e mostra che lo stato dello schema è `UPDATING`. Seleziona il pulsante di aggiornamento fino a quando lo stato non cambia in `ACTIVE` o `FAILED`.
7. Se lo stato è `FAILED`, seleziona il piano e nella scheda Actions (Operazioni), scegli View (Visualizza).

   La pagina dei dettagli mostra il motivo dell'errore. Se il messaggio dell'errore indica che è impossibile accedere all'oggetto nella posizione... o che è negato l'accesso sull'oggetto nella posizione..., verifica i requisiti seguenti:

   - L'utente con cui hai effettuato l'accesso deve disporre dell'autorizzazione di lettura per l'archivio ZIP del piano in Amazon S3.
   - Il bucket Amazon S3 che contiene l'archivio ZIP deve disporre di una policy di bucket che conceda l'autorizzazione di lettura sull'oggetto al tuo ID account AWS. Per ulteriori informazioni, consulta Pubblicazione di uno schema.
   - Il bucket Amazon S3 che stai utilizzando deve trovarsi nella stessa regione di quella alla quale hai eseguito l'accesso sulla console.

   > **Note**
   >
   > Se l'aggiornamento non riesce, l'esecuzione successiva del piano utilizza la versione più recente del piano correttamente registrata o aggiornata.

Per aggiornare un piano (AWS CLI)

1. Inserisci il comando seguente.

```
aws glue update-blueprint --name <blueprint-name> [--description <description>] --
blueprint-location s3://<s3-path>/<archive-filename>
```

2. Immetti il seguente comando per verificare lo stato del piano. Ripeti il comando fino a quando lo stato non diventa `ACTIVE` o `FAILED`.

   ```
   aws glue get-blueprint --name <blueprint-name>
   ```

   Se lo stato è `FAILED` e il messaggio dell'errore indica che è impossibile accedere all'oggetto nella posizione... o che è negato l'accesso sull'oggetto nella posizione..., verifica i requisiti seguenti:

   - L'utente con cui hai effettuato l'accesso deve disporre dell'autorizzazione di lettura per l'archivio ZIP del piano in Amazon S3.
   - Il bucket Amazon S3 che contiene l'archivio ZIP deve disporre di una policy di bucket che conceda l'autorizzazione di lettura sull'oggetto al tuo ID account AWS. Per ulteriori informazioni, consulta Pubblicazione di uno schema.
   - Il bucket Amazon S3 che stai utilizzando deve trovarsi nella stessa regione di quella alla quale hai eseguito l'accesso sulla console.

> **Consulta anche**
>
> - Panoramica degli schemi in AWS Glue

Creazione di un flusso di lavoro da uno schema in AWS Glue

Puoi creare un flusso di lavoro AWS Glue manualmente, aggiungendo un componente alla volta, oppure puoi creare un flusso di lavoro da un progetto AWS Glue. AWS Glue include progetti per casi d'uso comuni. Gli sviluppatori di AWS Glue possono creare progetti aggiuntivi.

> **Important**
>
> Limita il numero totale di processi, crawler e attivazioni all'interno di un flusso di lavoro a 100 o meno. Se includi più di 100, potresti riscontrare errori durante il tentativo di riprendere o interrompere l'esecuzione del flusso di lavoro.

Quando utilizzi un progetto, puoi generare rapidamente un flusso di lavoro per uno specifico caso d'uso basato sul caso d'uso generalizzato definito dal progetto. Puoi definire il caso d'uso specifico

fornendo valori per i parametri del progetto. Ad esempio, un progetto che partiziona un set di dati potrebbe avere i percorsi di origine e destinazione di Amazon S3 come parametri.

AWS Glue crea un flusso di lavoro da un progetto eseguendo il progetto. L'esecuzione del progetto salva i valori dei parametri forniti e viene utilizzata per tenere traccia dell'avanzamento e dell'esito della creazione del flusso di lavoro e dei relativi componenti. Durante la risoluzione dei problemi di un flusso di lavoro, puoi sempre visualizzare l'esecuzione del progetto per determinare i valori dei parametri del progetto utilizzati per creare un flusso di lavoro.

Per creare e visualizzare i flussi di lavoro, è necessario disporre di determinate autorizzazioni IAM. Per la policy IAM suggerita, consulta Autorizzazioni per gli schemi per l'analista dati.

Puoi creare un piano utilizzando la console AWS Glue, l'API AWS Glue o la AWS Command Line Interface (AWS CLI).

Per creare un flusso di lavoro da un progetto (console)

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.

   Accedi come un utente che dispone delle autorizzazioni per creare un flusso di lavoro.
2. Nel pannello di navigazione seleziona schemi.
3. Seleziona un progetto e nel menu Actions (Operazioni), scegli Create workflow (Crea flusso di lavoro).
4. Nella pagina Create a workflow from <blueprint-name> (Crea un flusso di lavoro da <blueprint-name>), inserisci le seguenti informazioni:

   Parametri del progetto

   Questi variano in base alla progettazione del progetto. Per domande sui parametri, consulta lo sviluppatore. Gli schemi in genere includono un parametro per il nome del flusso di lavoro.

   Ruolo IAM

   Il ruolo che AWS Glue assume per creare il flusso di lavoro e i relativi componenti. Il ruolo deve disporre delle autorizzazioni per creare ed eliminare flussi di lavoro, processi, crawler e trigger. Per una policy suggerita per il ruolo, consulta Autorizzazioni per i ruoli degli schemi.
5. Scegli Submit (Invia).

   Viene visualizzata la pagina Blueprint Details (Dettagli progetto), che mostra un elenco di esecuzioni del piano nella parte inferiore.

6. Nell'elenco delle esecuzioni del progetto, controlla lo stato della creazione del flusso di lavoro nell'esecuzione del progetto che si trova più in alto.

   Lo stato iniziale è `RUNNING`. Seleziona il pulsante di aggiornamento fino a quando lo stato non diventa `SUCCEEDED` o `FAILED`.

7. Scegli una delle seguenti operazioni:

   - Se lo stato di completamento è `SUCCEEDED`, puoi passare alla pagina Workflows (Flussi di lavoro), selezionare il flusso di lavoro appena creato ed eseguirlo. Prima di eseguire il flusso di lavoro, è possibile esaminare il grafico di progettazione.
   - Se lo stato di completamento è `FAILED`, seleziona l'esecuzione del progetto e nel menu Actions (Operazioni), scegli View (Visualizza) per vedere il messaggio di errore.

Per ulteriori informazioni sui flussi di lavoro e sui progetti, consulta i seguenti argomenti.

- Panoramica di flussi di lavoro in AWS Glue
- Aggiornamento di uno schema in AWS Glue
- Creazione e costruzione manuale di un flusso di lavoro in AWS Glue

Visualizzazione delle esecuzioni dello schema in AWS Glue

Visualizza l'esecuzione di un piano per vedere le seguenti informazioni:

- Nome del flusso di lavoro che è stato creato.
- Valori dei parametri dello schema utilizzati per creare il flusso di lavoro.
- Stato dell'operazione di creazione del flusso di lavoro.

Puoi visualizzare l'esecuzione di un progetto utilizzando la console AWS Glue, l'API AWS Glue o la AWS Command Line Interface (AWS CLI).

Per visualizzare l'esecuzione di un piano (console)

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel pannello di navigazione scegli schemi.
3. Nella pagina schemi seleziona uno schema. Quindi nel menu Actions (Operazioni), scegli View (Visualizza).

4. Nella parte inferiore della finestra Blueprint Details (Dettagli piano), seleziona un'esecuzione del piano e nel menu Actions (Operazioni), scegli View (Visualizza).

Per visualizzare l'esecuzione di un piano (AWS CLI)

- Inserire il seguente comando. Sostituisci *<blueprint-name>* con il nome del piano. Sostituisci *<blueprint-run-id>* con l'ID di esecuzione del piano.

```
aws glue get-blueprint-run --blueprint-name <blueprint-name> --run-id <blueprint-run-id>
```

Consulta anche:

- Panoramica degli schemi in AWS Glue

AWS CloudFormation per AWS Glue

AWS CloudFormation è un servizio che può creare molte risorse AWS. AWS Glue fornisce operazioni API per la creazione di oggetti nel AWS Glue Data Catalog. Tuttavia, può essere utile definire e creare oggetti AWS Glue e altri oggetti risorsa AWS correlati in un file di modello AWS CloudFormation. È quindi possibile automatizzare il processo di creazione degli oggetti.

AWS CloudFormation fornisce una sintassi semplificata, JSON (JavaScript Object Notation) o YAML (YAML Ain't Markup Language), per esprimere la creazione di risorse AWS. Puoi usare modelli AWS CloudFormation per definire oggetti del catalogo dati come database, tabelle, partizioni, crawler, classificatori e connessioni. È anche possibile definire oggetti ETL, come processi, trigger ed endpoint di sviluppo. Puoi creare un modello che descrive tutte le risorse AWS che desideri perché AWS CloudFormation si occupi del provisioning e della configurazione di queste risorse per te.

Per ulteriori informazioni, consulta Cos'è AWS CloudFormation? e Utilizzo di modelli AWS CloudFormation nella Guida per l'utente di AWS CloudFormation.

Se prevedi di usare modelli AWS CloudFormation compatibili con AWS Glue, in qualità di amministratore devi concedere l'accesso a AWS CloudFormation e alle operazioni e ai servizi AWS da cui dipende. Per concedere le autorizzazioni per la creazione di risorse AWS CloudFormation, collega la policy seguente agli utenti che usano AWS CloudFormation:

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

La tabella seguente contiene le operazioni che un modello AWS CloudFormation può eseguire per conto tuo. Sono inclusi collegamenti a informazioni sui tipi di risorsa AWS e sui rispettivi tipi di proprietà che puoi aggiungere a un modello AWS CloudFormation.

Risorsa AWS Glue	Modello AWS CloudFormation	Esempi di AWS Glue
Classificatore	AWS::Glue::Classifier	Classificatore Grok, classificatore JSON, classificatore XML
Connessione	AWS::Glue::Connection	Connessione MySQL
Crawler	AWS::Glue::Crawler	Crawler Amazon S3, crawler MySQL
Database	AWS::Glue::Database	Database vuoto, database con tabelle
Endpoint di sviluppo	AWS::Glue::DevEndpoint	Endpoint di sviluppo
Processo	AWS::Glue::Job	Processo Amazon S3, Processo JDBC
Trasformazione basata su machine learning	AWS::Glue::MLTransform	Trasformazione basata su machine learning
Set di regole sulla qualità dei dati	AWS::Glue::DataQualityRuleset	Set di regole sulla qualità dei dati, Set di regole sulla qualità dei dati con Pianificatore EventBridge
Partizione	AWS::Glue::Partition	Partizioni di una tabella
Tabella	AWS::Glue::Table	Tabella in un database
Trigger	AWS::Glue::Trigger	Trigger on demand, trigger pianificato, trigger condizionale

Per iniziare, usa i modelli di esempio seguenti e personalizzali con i tuoi metadati. Usa quindi la console AWS CloudFormation per creare uno stack AWS CloudFormation per l'aggiunta di oggetti ad AWS Glue e a tutti i servizi associati. Molti campi di un oggetto AWS Glue sono facoltativi. Questi modelli indicano i campi obbligatori o necessari per un oggetto AWS Glue funzionante e funzionale.

Un modello AWS CloudFormation può avere il formato JSON o YAML. In questi esempi viene usato il formato YAML per semplificare la lettura. Gli esempi contengono commenti (#) per descrivere i valori definiti nei modelli.

I modelli AWS CloudFormation possono includere una sezione `Parameters`. Questa sezione può essere modificata nel testo di esempio o quando il file YAML viene inviato alla console AWS CloudFormation per creare uno stack. La sezione `Resources` del modello contiene la definizione di AWS Glue e degli oggetti correlati. Le definizioni della sintassi dei modelli AWS CloudFormation possono contenere proprietà che includono una sintassi più dettagliata. Non tutte le proprietà potrebbero essere necessarie per creare un oggetto AWS Glue. Questi esempi mostrano valori di esempio per alcune proprietà comuni per la creazione di un oggetto AWS Glue.

Modello AWS CloudFormation di esempio per un database AWS Glue

Un database AWS Glue nel catalogo dati contiene tabelle di metadati. Il database è costituito da pochissime proprietà e può essere creato nel catalogo dati con un modello AWS CloudFormation. Il modello di esempio seguente viene fornito per iniziare e per mostrare l'uso di stack AWS CloudFormation con AWS Glue. L'unica risorsa creata dal modello di esempio è un database denominato `cfn-mysampledatabase`. Puoi modificarlo modificando il testo dell'esempio o cambiando il valore nella console AWS CloudFormation quando invii il codice YAML.

L'esempio seguente mostra valori di esempio per alcune proprietà comuni per la creazione di un database AWS Glue. Per ulteriori informazioni sul modello di database AWS CloudFormation per AWS Glue, consulta AWS::Glue::Database.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CloudFormation template in YAML to demonstrate creating a database named
 mysampledatabase
# The metadata created in the Data Catalog points to the flights public S3 bucket
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  CFNDatabaseName:
    Type: String
    Default: cfn-mysampledatabse
```

```
# Resources section defines metadata for the Data Catalog
Resources:
# Create an AWS Glue database
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      # The database is created in the Data Catalog for your account
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        # The name of the database is defined in the Parameters section above
        Name: !Ref CFNDatabaseName
        Description: Database to hold tables for flights data
        LocationUri: s3://crawler-public-us-east-1/flight/2016/csv/
        #Parameters: Leave AWS database parameters blank
```

Modello AWS CloudFormation di esempio per un database, una tabella e una partizione AWS Glue

Una tabella AWS Glue contiene i metadati che definiscono la struttura e la posizione dei dati che vuoi elaborare con gli script ETL. In una tabella è possibile definire partizioni per parallelizzare l'elaborazione dei dati. Una partizione è un blocco di dati definito con una chiave. Se, ad esempio, usi il mese come chiave, tutti i dati per gennaio vengono inclusi nella stessa partizione. In AWS Glue i database possono contenere tabelle e le tabelle possono contenere partizioni.

L'esempio seguente mostra come popolare un database, una tabella e le partizioni usando un modello AWS CloudFormation. Il formato dei dati di base è `csv`, con valori delimitati da una virgola (,). Poiché un database deve esistere per poter contenere una tabella e una tabella deve esistere per poter creare le partizioni, il modello usa l'istruzione `DependsOn` per definire la dipendenza di questi oggetti quando vengono creati.

I valori in questo esempio definiscono una tabella che contiene dati di voli da un bucket Amazon S3 disponibile pubblicamente. A scopo illustrativo, sono definite solo alcune colonne di dati e una chiave di partizionamento. Vengono definite anche quattro partizioni nel catalogo dati. Nei campi `StorageDescriptor` sono mostrati anche alcuni campi per descrivere lo storage dei dati di base.

```
---
AWSTemplateFormatVersion: '2010-09-09'
```

```
# Sample CloudFormation template in YAML to demonstrate creating a database, a table,
 and partitions
# The metadata created in the Data Catalog points to the flights public S3 bucket
#
# Parameters substituted in the Resources section
# These parameters are names of the resources created in the Data Catalog
Parameters:
  CFNDatabaseName:
    Type: String
    Default: cfn-database-flights-1
  CFNTableName1:
    Type: String
    Default: cfn-manual-table-flights-1
# Resources to create metadata in the Data Catalog
Resources:
###
# Create an AWS Glue database
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        Name: !Ref CFNDatabaseName
        Description: Database to hold tables for flights data
###
# Create an AWS Glue table
  CFNTableFlights:
    # Creating the table waits for the database to be created
    DependsOn: CFNDatabaseFlights
    Type: AWS::Glue::Table
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableInput:
        Name: !Ref CFNTableName1
        Description: Define the first few columns of the flights table
        TableType: EXTERNAL_TABLE
        Parameters: {
    "classification": "csv"
  }
#       ViewExpandedText: String
        PartitionKeys:
        # Data is partitioned by month
        - Name: mon
```

```
          Type: bigint
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: year
            Type: bigint
          - Name: quarter
            Type: bigint
          - Name: month
            Type: bigint
          - Name: day_of_month
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 1
# Create an AWS Glue partition
  CFNPartitionMon1:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 1
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=1/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 2
# Create an AWS Glue partition
  CFNPartitionMon2:
```

```
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 2
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=2/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 3
# Create an AWS Glue partition
  CFNPartitionMon3:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 3
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=3/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 4
```

```
# Create an AWS Glue partition
  CFNPartitionMon4:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 4
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=4/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
```

Modello AWS CloudFormation di esempio per un classificatore Grok AWS Glue

Un classificatore AWS Glue determina lo schema dei dati. Un tipo di classificatore personalizzato usa un pattern grok per trovare la corrispondenza con i dati. Se il pattern corrisponde, il classificatore personalizzato viene usato per creare lo schema della tabella e impostare `classification` sul valore impostato nella definizione del classificatore.

Questo esempio crea un classificatore che crea a sua volta uno schema con una colonna denominata `message` e imposta la classificazione su `greedy`.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a classifier
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
```

```
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-grok-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses grok pattern to put all data in one column and classifies
 it as "greedy".
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      GrokClassifier:
        #Grok classifier that puts all data in one column
        Name: !Ref CFNClassifierName
        Classification: greedy

        GrokPattern: "%{GREEDYDATA:message}"
        #CustomPatterns: none
```

Modello AWS CloudFormation di esempio per un classificatore JSON AWS Glue

Un classificatore AWS Glue determina lo schema dei dati. Un tipo di classificatore personalizzato utilizza una stringa `JsonPath` che definisce i dati JSON per il classificatore da classificare. AWS Glue supporta un sottoinsieme di operatori per `JsonPath`, come descritto in Scrittura di classificatori JsonPath personalizzati.

Se il modello corrisponde, il classificatore personalizzato viene utilizzato per creare il tuo schema della tabella.

Questo esempio crea un classificatore che a sua volta crea uno schema con ogni record nella matrice `Records3` in un oggetto.

```
---
AWSTemplateFormatVersion: '2010-09-09'
```

```
# Sample CFN YAML to demonstrate creating a JSON classifier
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-json-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses a JSON pattern.
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      JSONClassifier:
        #JSON classifier
        Name: !Ref CFNClassifierName
        JsonPath: $.Records3[*]
```

Modello AWS CloudFormation di esempio per un classificatore XML AWS Glue

Un classificatore AWS Glue determina lo schema dei dati. Un tipo di classificatore personalizzato specifica un tag XML per designare l'elemento che contiene ogni record in un documento XML sottoposto ad analisi. Se il pattern corrisponde, il classificatore personalizzato viene usato per creare lo schema della tabella e impostare `classification` sul valore impostato nella definizione del classificatore.

Questo esempio crea un classificatore che crea a sua volta uno schema con ciascun record nel tag `Record` e imposta la classificazione su `XML`.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating an XML classifier
```

```
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-xml-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses the XML pattern and classifies it as "XML".
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      XMLClassifier:
        #XML classifier
        Name: !Ref CFNClassifierName
        Classification: XML
        RowTag: <Records>
```

Modello AWS CloudFormation di esempio per un crawler AWS Glue per Amazon S3

Un crawler AWS Glue crea nel catalogo dati tabelle di metadati che corrispondono ai dati. Puoi quindi usare queste definizioni di tabella come origini e target nei processi ETL.

Questo esempio crea un crawler, il ruolo IAM necessario e un database AWS Glue nel catalogo dati. Quando il crawler viene eseguito, assume il ruolo IAM e crea una tabella del database per i dati dei voli pubblici. La tabella viene creata con il prefisso "`cfn_sample_1_`". Il ruolo IAM creato da questo modello concede autorizzazioni globali. Potresti voler creare un ruolo personalizzato. Tramite questo classificatore non vengono definiti classificatori personalizzati. Per impostazione predefinita, vengono usati classificatori AWS Glue predefiniti.

Quando invii questo esempio alla console AWS CloudFormation, devi confermare di voler creare il ruolo IAM.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a crawler
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNCrawlerName:
    Type: String
    Default: cfn-crawler-flights-1
  CFNDatabaseName:
    Type: String
    Default: cfn-database-flights-1
  CFNTablePrefixName:
    Type: String
    Default: cfn_sample_1_
#
#
# Resources section defines metadata for the Data Catalog
Resources:
#Create IAM Role assumed by the crawler. For demonstration, this role is given all
 permissions.
  CFNRoleFlights:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: "Allow"
            Principal:
              Service:
                - "glue.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        -
          PolicyName: "root"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
```

```
              -
                Effect: "Allow"
                Action: "*"
                Resource: "*"
  # Create a database to contain tables created by the crawler
   CFNDatabaseFlights:
     Type: AWS::Glue::Database
     Properties:
       CatalogId: !Ref AWS::AccountId
       DatabaseInput:
         Name: !Ref CFNDatabaseName
         Description: "AWS Glue container to hold metadata tables for the flights
  crawler"
  #Create a crawler to crawl the flights data on a public S3 bucket
   CFNCrawlerFlights:
     Type: AWS::Glue::Crawler
     Properties:
       Name: !Ref CFNCrawlerName
       Role: !GetAtt CFNRoleFlights.Arn
       #Classifiers: none, use the default classifier
       Description: AWS Glue crawler to crawl flights data
       #Schedule: none, use default run-on-demand
       DatabaseName: !Ref CFNDatabaseName
       Targets:
         S3Targets:
           # Public S3 bucket with the flights data
           - Path: "s3://crawler-public-us-east-1/flight/2016/csv"
       TablePrefix: !Ref CFNTablePrefixName
       SchemaChangePolicy:
         UpdateBehavior: "UPDATE_IN_DATABASE"
         DeleteBehavior: "LOG"
       Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":
{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":{\"AddOrUpdateBehavior\":
\"MergeNewColumns\"}}}"
```

Modello AWS CloudFormation di esempio per una connessione AWS Glue

Una connessione AWS Glue nel catalogo dati contiene le informazioni di rete e JDBC necessarie per la connessione a un database JDBC. Queste informazioni vengono usate per la connessione a un database JDBC per il crawling o l'esecuzione di processi ETL.

In questo esempio viene creata una connessione a un database Amazon RDS MySQL denominato devdb. Quando la connessione viene usata, è necessario fornire anche un ruolo IAM, le credenziali del database e i valori per la connessione di rete. Consulta i dettagli dei campi necessari nel modello.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a connection
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the connection to be created
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
  CFNJDBCString:
    Type: String
    Default: "jdbc:mysql://xxx-mysql.yyyyyyyyyyyyyy.us-east-1.rds.amazonaws.com:3306/
devdb"
  CFNJDBCUser:
    Type: String
    Default: "master"
  CFNJDBCPassword:
    Type: String
    Default: "12345678"
    NoEcho: true
#
#
# Resources section defines metadata for the Data Catalog
Resources:
  CFNConnectionMySQL:
    Type: AWS::Glue::Connection
    Properties:
      CatalogId: !Ref AWS::AccountId
      ConnectionInput:
        Description: "Connect to MySQL database."
        ConnectionType: "JDBC"
        #MatchCriteria: none
        PhysicalConnectionRequirements:
          AvailabilityZone: "us-east-1d"
          SecurityGroupIdList:
```

```
           - "sg-7d52b812"
          SubnetId: "subnet-84f326ee"
        ConnectionProperties: {
          "JDBC_CONNECTION_URL": !Ref CFNJDBCString,
          "USERNAME": !Ref CFNJDBCUser,
          "PASSWORD": !Ref CFNJDBCPassword
        }
        Name: !Ref CFNConnectionName
```

Modello AWS CloudFormation di esempio per un crawler AWS Glue per JDBC.

Un crawler AWS Glue crea nel catalogo dati tabelle di metadati che corrispondono ai dati. Puoi quindi usare queste definizioni di tabella come origini e target nei processi ETL.

Questo esempio crea un crawler, il ruolo IAM necessario e un database AWS Glue nel catalogo dati. Quando il crawler viene eseguito, assume il ruolo IAM e crea una tabella nel database per i dati dei voli pubblici archiviati in un database MySQL. La tabella viene creata con il prefisso "`cfn_jdbc_1_`". Il ruolo IAM creato da questo modello concede autorizzazioni globali. Potresti voler creare un ruolo personalizzato. Per i dati JDBC non è possibile definire classificatori personalizzati. Per impostazione predefinita, vengono usati classificatori AWS Glue predefiniti.

Quando invii questo esempio alla console AWS CloudFormation, devi confermare di voler creare il ruolo IAM.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a crawler
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNCrawlerName:
    Type: String
    Default: cfn-crawler-jdbc-flights-1
# The name of the database to be created to contain tables
  CFNDatabaseName:
    Type: String
```

```
    Default: cfn-database-jdbc-flights-1
# The prefix for all tables crawled and created
  CFNTablePrefixName:
    Type: String
    Default: cfn_jdbc_1_
# The name of the existing connection to the MySQL database
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
# The name of the JDBC path (database/schema/table) with wildcard (%) to crawl
  CFNJDBCPath:
    Type: String
    Default: saldev/%
#
#
# Resources section defines metadata for the Data Catalog
Resources:
#Create IAM Role assumed by the crawler. For demonstration, this role is given all
 permissions.
  CFNRoleFlights:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: "Allow"
            Principal:
              Service:
                - "glue.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        -
          PolicyName: "root"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              -
                Effect: "Allow"
                Action: "*"
                Resource: "*"
 # Create a database to contain tables created by the crawler
```

```
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        Name: !Ref CFNDatabaseName
        Description: "AWS Glue container to hold metadata tables for the flights
 crawler"
 #Create a crawler to crawl the flights data in MySQL database
  CFNCrawlerFlights:
    Type: AWS::Glue::Crawler
    Properties:
      Name: !Ref CFNCrawlerName
      Role: !GetAtt CFNRoleFlights.Arn
      #Classifiers: none, use the default classifier
      Description: AWS Glue crawler to crawl flights data
      #Schedule: none, use default run-on-demand
      DatabaseName: !Ref CFNDatabaseName
      Targets:
        JdbcTargets:
          # JDBC MySQL database with the flights data
          - ConnectionName: !Ref CFNConnectionName
            Path: !Ref CFNJDBCPath
          #Exclusions: none
      TablePrefix: !Ref CFNTablePrefixName
      SchemaChangePolicy:
        UpdateBehavior: "UPDATE_IN_DATABASE"
        DeleteBehavior: "LOG"
   Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":
{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":{\"AddOrUpdateBehavior\":
\"MergeNewColumns\"}}}"
```

Modello AWS CloudFormation di esempio per un processo AWS Glue da Amazon S3 ad Amazon S3

Un processo AWS Glue nel catalogo dati contiene i valori dei parametri necessari per eseguire uno script in AWS Glue.

Questo esempio crea un processo che legge i dati dei voli da un bucket Amazon S3 in formato `csv` e li scrive in un file Parquet in Amazon S3. Lo script eseguito da questo processo deve esistere già.

Puoi generare uno script ETL per l'ambiente con la console AWS Glue. Quando questo processo viene eseguito, è necessario fornire anche un ruolo IAM con le autorizzazioni appropriate.

I valori dei parametri comuni sono mostrati nel modello. Ad esempio, il valore predefinito di `AllocatedCapacity` (DPU) è 5.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a job using the public flights S3 table in a
 public bucket
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the job to be created
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-2
# The name of the IAM role that the job assumes. It must have access to data, script,
 temporary directory
  CFNIAMRoleName:
    Type: String
    Default: AWSGlueServiceRoleGA
# The S3 path where the script for this job is located
  CFNScriptLocation:
    Type: String
    Default: s3://aws-glue-scripts-123456789012-us-east-1/myid/sal-job-test2
#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create job to run script which accesses flightscsv table and write to S3 file as
 parquet.
# The script already exists and is called by this job
  CFNJobFlights:
    Type: AWS::Glue::Job
    Properties:
      Role: !Ref CFNIAMRoleName
      #DefaultArguments: JSON object
      # If script written in Scala, then set DefaultArguments={'--job-language';
 'scala', '--class': 'your scala class'}
```

```
      #Connections:  No connection needed for S3 to S3 job
      #  ConnectionsList
      #MaxRetries: Double
      Description: Job created with CloudFormation
      #LogUri: String
      Command:
        Name: glueetl
        ScriptLocation: !Ref CFNScriptLocation
             # for access to directories use proper IAM role with permission to buckets
and folders that begin with "aws-glue-"
             # script uses temp directory from job definition if required (temp
directory not used S3 to S3)
             # script defines target for output as s3://aws-glue-target/sal
      AllocatedCapacity: 5
      ExecutionProperty:
        MaxConcurrentRuns: 1
      Name: !Ref CFNJobName
```

Modello AWS CloudFormation di esempio per un processo AWS Glue per il trasferimento da JDBC ad Amazon S3

Un processo AWS Glue nel catalogo dati contiene i valori dei parametri necessari per eseguire uno script in AWS Glue.

Questo esempio crea un processo che legge i dati dei voli da un database JDBC MySQL in base a quanto definito dalla connessione denominata `cfn-connection-mysql-flights-1` e li scrive in un file Parquet in Amazon S3. Lo script eseguito da questo processo deve esistere già. Puoi generare uno script ETL per l'ambiente con la console AWS Glue. Quando questo processo viene eseguito, è necessario fornire anche un ruolo IAM con le autorizzazioni appropriate.

I valori dei parametri comuni sono mostrati nel modello. Ad esempio, il valore predefinito di `AllocatedCapacity` (DPU) è 5.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a job using a MySQL JDBC DB with the flights
 data to an S3 file
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
```

```
Parameters:

# The name of the job to be created
  CFNJobName:
    Type: String
    Default: cfn-job-JDBC-to-S3-1
# The name of the IAM role that the job assumes. It must have access to data, script,
 temporary directory
  CFNIAMRoleName:
    Type: String
    Default: AWSGlueServiceRoleGA
# The S3 path where the script for this job is located
  CFNScriptLocation:
    Type: String
    Default: s3://aws-glue-scripts-123456789012-us-east-1/myid/sal-job-dec4a
# The name of the connection used for JDBC data source
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create job to run script which accesses JDBC flights table via a connection and write
 to S3 file as parquet.
# The script already exists and is called by this job
  CFNJobFlights:
    Type: AWS::Glue::Job
    Properties:
      Role: !Ref CFNIAMRoleName
      #DefaultArguments: JSON object
      # For example, if required by script, set temporary directory as
 DefaultArguments={'--TempDir'; 's3://aws-glue-temporary-xyc/sal'}
      Connections:
        Connections:
        - !Ref CFNConnectionName
      #MaxRetries: Double
      Description: Job created with CloudFormation using existing script
      #LogUri: String
      Command:
        Name: glueetl
        ScriptLocation: !Ref CFNScriptLocation
             # for access to directories use proper IAM role with permission to buckets
 and folders that begin with "aws-glue-"
```

```
            # if required, script defines temp directory as argument TempDir and used
 in script like redshift_tmp_dir = args["TempDir"]
            # script defines target for output as s3://aws-glue-target/sal
      AllocatedCapacity: 5
      ExecutionProperty:
        MaxConcurrentRuns: 1
      Name: !Ref CFNJobName
```

Modello AWS CloudFormation di esempio per un trigger on demand AWS Glue

Un trigger AWS Glue nel catalogo dati contiene i valori dei parametri necessari per avviare l'esecuzione di un processo quando viene attivato il trigger. Un trigger on demand viene attivato quando lo si abilita.

In questo esempio viene creato un trigger on demand che avvia un processo denominato `cfn-job-S3-to-S3-1`.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating an on-demand trigger
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-1
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-ondemand-flights-1
#
# Resources section defines metadata for the Data Catalog
# Sample CFN YAML to demonstrate creating an on-demand trigger for a job
Resources:
# Create trigger to run an existing job (CFNJobName) on an on-demand schedule.
  CFNTriggerSample:
    Type: AWS::Glue::Trigger
```

```
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: ON_DEMAND
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      #Schedule:
      #Predicate:
```

Modello AWS CloudFormation di esempio per un trigger pianificato AWS Glue

Un trigger AWS Glue nel catalogo dati contiene i valori dei parametri necessari per avviare l'esecuzione di un processo quando viene attivato il trigger. Un trigger pianificato viene attivato quando è abilitato e il timer cron raggiunge il valore definito.

In questo esempio viene creato un trigger pianificato che avvia un processo denominato `cfn-job-S3-to-S3-1`. Il timer è un'espressione cron per l'esecuzione del processo ogni 10 minuti nei giorni feriali.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a scheduled trigger
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-1
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-scheduled-flights-1
#
# Resources section defines metadata for the Data Catalog
# Sample CFN YAML to demonstrate creating a scheduled trigger for a job
```

```
#
Resources:
# Create trigger to run an existing job (CFNJobName) on a cron schedule.
  TriggerSample1CFN:
    Type: AWS::Glue::Trigger
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: SCHEDULED
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      # # Run the trigger every 10 minutes on Monday to Friday
      Schedule: cron(0/10 * ? * MON-FRI *)
      #Predicate:
```

Modello AWS CloudFormation di esempio per un trigger condizionale AWS Glue

Un trigger AWS Glue nel catalogo dati contiene i valori dei parametri necessari per avviare l'esecuzione di un processo quando viene attivato il trigger. Un trigger condizionale viene attivato quando è abilitato e le relative condizioni vengono soddisfatte, ad esempio un processo viene completato correttamente.

In questo esempio viene creato un trigger condizionale che avvia un processo denominato `cfn-job-S3-to-S3-1`. Questo processo viene avviato quando il processo denominato `cfn-job-S3-to-S3-2` viene completato correttamente.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a conditional trigger for a job, which starts
 when another job completes
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
    Type: String
```

```
    Default: cfn-job-S3-to-S3-1
  # The existing job that when it finishes causes trigger to fire
  CFNJobName2:
    Type: String
    Default: cfn-job-S3-to-S3-2
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-conditional-1
#
Resources:
# Create trigger to run an existing job (CFNJobName) when another job completes
 (CFNJobName2).
  CFNTriggerSample:
    Type: AWS::Glue::Trigger
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: CONDITIONAL
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      #Schedule: none
      Predicate:
        #Value for Logical is required if more than 1 job listed in Conditions
        Logical: AND
        Conditions:
          - LogicalOperator: EQUALS
            JobName: !Ref CFNJobName2
            State: SUCCEEDED
```

Modello AWS CloudFormation di esempio per un endpoint di sviluppo AWS Glue

Una trasformazione basata su machine learning di AWS Glue è una trasformazione personalizzata per ripulire i dati. Attualmente è disponibile una trasformazione denominata FindMatches. La trasformazione FindMatches consente di identificare record duplicati o corrispondenti nel set di dati, anche quando i record non dispongono di un identificatore univoco comune e nessun campo corrisponde esattamente.

Questo esempio mostra come creare una trasformazione basata su machine learning. Per ulteriori informazioni sui parametri necessari per creare una trasformazione basata su machine learning, consulta Corrispondenza dei record con FindMatches AWS Lake Formation.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a machine learning transform
#
# Resources section defines metadata for the machine learning transform
Resources:
  MyMLTransform:
    Type: "AWS::Glue::MLTransform"
    Condition: "isGlueMLGARegion"
    Properties:
      Name: !Sub "MyTransform"
      Description: "The bestest transform ever"
      Role: !ImportValue MyMLTransformUserRole
      GlueVersion: "1.0"
      WorkerType: "Standard"
      NumberOfWorkers: 5
      Timeout: 120
      MaxRetries: 1
      InputRecordTables:
        GlueTables:
          - DatabaseName: !ImportValue MyMLTransformDatabase
            TableName: !ImportValue MyMLTransformTable
      TransformParameters:
        TransformType: "FIND_MATCHES"
        FindMatchesParameters:
          PrimaryKeyColumnName: "testcolumn"
          PrecisionRecallTradeoff: 0.5
          AccuracyCostTradeoff: 0.5
          EnforceProvidedLabels: True
      Tags:
        key1: "value1"
        key2: "value2"
      TransformEncryption:
        TaskRunSecurityConfigurationName: !ImportValue
 MyMLTransformSecurityConfiguration
        MLUserDataEncryption:
          MLUserDataEncryptionMode: "SSE-KMS"
          KmsKeyId: !ImportValue MyMLTransformEncryptionKey
```

Modello di esempio di AWS CloudFormation per un set di regole AWS Glue Data Quality

Un set di regole di Qualità dei dati di AWS Glue contiene regole che possono essere valutate su una tabella all'interno di Catalogo dati. Una volta che il set di regole è stato inserito nella tabella di destinazione, è possibile accedere a Catalogo dati ed eseguire una valutazione che esamina i dati in base a tali regole all'interno del set di regole. Queste regole spaziano dalla valutazione del conteggio delle righe alla valutazione dell'integrità referenziale dei dati.

L'esempio seguente è un modello CloudFormation che crea un set di regole con una varietà di regole sulla tabella di destinazione specificata.

```
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a DataQualityRuleset
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

  # The name of the ruleset to be created
  RulesetName:
    Type: String
    Default: "CFNRulesetName"
  RulesetDescription:
    Type: String
    Default: "CFN DataQualityRuleset"
  # Rules that will be associated with this ruleset
  Rules:
    Type: String
    Default: 'Rules = [
        RowCount > 100,
        IsUnique "id",
        IsComplete "nametype"
        ]'
  # Name of database and table within Data Catalog which the ruleset will
  # be applied too
  DatabaseName:
    Type: String
    Default: "ExampleDatabaseName"
  TableName:
    Type: String
```

```
    Default: "ExampleTableName"

# Resources section defines metadata for the Data Catalog
Resources:
  # Creates a Data Quality ruleset under specified rules
  DQRuleset:
    Type: AWS::Glue::DataQualityRuleset
    Properties:
      Name: !Ref RulesetName
      Description: !Ref RulesetDescription
      # The String within rules must be formatted in DQDL, a language
      # used specifically to make rules
      Ruleset: !Ref Rules
      # The targeted table must exist within Data Catalog alongside
      # the correct database
      TargetTable:
        DatabaseName: !Ref DatabaseName
        TableName: !Ref TableName
```

Modello di esempio di AWS CloudFormation per un set di regole AWS Glue Data Quality con Pianificatore EventBridge

Un set di regole di Qualità dei dati di AWS Glue contiene regole che possono essere valutate su una tabella all'interno di Catalogo dati. Una volta che il set di regole è stato inserito nella tabella di destinazione, è possibile accedere a Catalogo dati ed eseguire una valutazione che esamina i dati in base a tali regole all'interno del set di regole. Invece di dover accedere manualmente a Catalogo dati per valutare il set di regole, puoi anche aggiungere un Pianificatore EventBridge all'interno del modello CloudFormation, pianificando così le valutazioni del set di regole per tuo conto a intervalli di tempo.

L'esempio seguente è un modello CloudFormation che crea un set di regole Qualità dei dati e un pianificatore EventBridge per valutare il suddetto set di regole ogni cinque minuti.

```
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a DataQualityRuleset
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

  # The name of the ruleset to be created
```

```
  RulesetName:
    Type: String
    Default: "CFNRulesetName"
  # Rules that will be associated with this Ruleset
  Rules:
    Type: String
    Default: 'Rules = [
        RowCount > 100,
        IsUnique "id",
        IsComplete "nametype"
        ]'
  # The name of the Schedule to be created
  ScheduleName:
    Type: String
    Default: "ScheduleDQRulsetEvaluation"
  # This expression determines the rate at which the Schedule will evaluate
  # your data using the above ruleset
  ScheduleRate:
    Type: String
    Default: "rate(5 minutes)"
  # The Request that being sent must match the details of the Data Quality Ruleset
  ScheduleRequest:
    Type: String
    Default: '
        { "DataSource": { "GlueTable": { "DatabaseName": "ExampleDatabaseName",
         "TableName": "ExampleTableName" } },
         "Role": "role/AWSGlueServiceRoleDefault",
          "RulesetNames": [ ""CFNRulesetName"" ] }
        '

# Resources section defines metadata for the Data Catalog
Resources:
  # Creates a Data Quality ruleset under specified rules
  DQRuleset:
    Type: AWS::Glue::DataQualityRuleset
    Properties:
      Name: !Ref RulesetName
      Description: "CFN DataQualityRuleset"
      # The String within rules must be formatted in DQDL, a language
      # used specifically to make rules
      Ruleset: !Ref Rules
      # The targeted table must exist within Data Catalog alongside
      # the correct database
      TargetTable:
```

```
        DatabaseName: "ExampleDatabaseName"
        TableName: "ExampleTableName"
  # Create a Scheduler to schedule evaluation runs on the above ruleset
  ScheduleDQEval:
    Type: AWS::Scheduler::Schedule
    Properties:
      Name: !Ref ScheduleName
      Description: "Schedule DataQualityRuleset Evaluations"
      FlexibleTimeWindow:
        Mode: "OFF"
      ScheduleExpression: !Ref ScheduleRate
      ScheduleExpressionTimezone: "America/New_York"
      State: "ENABLED"
      Target:
        # The ARN is the API that will be run, since we want to evaluate our ruleset
        # we want this specific ARN
        Arn: "arn:aws:scheduler:::aws-sdk:glue:startDataQualityRulesetEvaluationRun"
        # Your RoleArn must have approval to schedule
        RoleArn: "arn:aws:iam::123456789012:role/AWSGlueServiceRoleDefault"
        # This is the Request that is being sent to the Arn
        Input: '
        { "DataSource": { "GlueTable": { "DatabaseName": "sampledb", "TableName":
"meteorite" } },
         "Role": "role/AWSGlueServiceRoleDefault",
          "RulesetNames": [ "TestCFN" ] }
        '
```

Modello AWS CloudFormation di esempio per un endpoint di sviluppo AWS Glue

Un endpoint di sviluppo AWS Glue è un ambiente che puoi usare per sviluppare e testare gli script AWS Glue.

In questo esempio viene creato un endpoint di sviluppo con i valori dei parametri di rete minimi necessari per la creazione. Per ulteriori informazioni sui parametri necessari per configurare un endpoint di sviluppo, consulta Creazione di reti per lo sviluppo per AWS Glue.

Per creare l'endpoint di sviluppo, puoi fornire un ARN (Amazon Resource Name) di un ruolo IAM esistente. Fornisci una chiave pubblica RSA valida e tieni a disposizione la chiave privata corrispondente se prevedi di creare un server notebook nell'endpoint di sviluppo.

> **Note**
>
> Effettui la gestione di qualsiasi server notebook che hai creato e che è associato a un endpoint di sviluppo. Di conseguenza, se elimini l'endpoint di sviluppo, per eliminare il server notebook devi eliminare lo stack AWS CloudFormation nella console AWS CloudFormation.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a development endpoint
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNEndpointName:
    Type: String
    Default: cfn-devendpoint-1
  CFNIAMRoleArn:
    Type: String
    Default: arn:aws:iam::123456789012/role/AWSGlueServiceRoleGA
#
#
# Resources section defines metadata for the Data Catalog
Resources:
  CFNDevEndpoint:
    Type: AWS::Glue::DevEndpoint
    Properties:
      EndpointName: !Ref CFNEndpointName
      #ExtraJarsS3Path: String
      #ExtraPythonLibsS3Path: String
      NumberOfNodes: 5
      PublicKey: ssh-rsa public.....key myuserid-key
      RoleArn: !Ref CFNIAMRoleArn
      SecurityGroupIds:
        - sg-64986c0b
      SubnetId: subnet-c67cccac
```

AWS Glue guida alla programmazione

Uno script contiene il codice che estrae i dati dalle fonti, li trasforma e li carica in obiettivi. AWS Glue esegue uno script quando avvia un processo.

Gli script di AWS Glue ETL possono essere codificati in Python o Scala. Sebbene tutti i tipi di processo possano essere scritti in Python, in AWS Glue per Spark i processi possono essere scritti anche in Scala. Quando si genera automaticamente la logica del codice sorgente per il job in AWS Glue Studio, viene creato uno script. Puoi modificare questo script oppure puoi fornire il tuo script per elaborare il lavoro ETL.

Fornire i propri script personalizzati

Gli script eseguono le operazioni di estrazione, trasformazione e caricamento (ETL). AWS Glue Uno script viene creato quando si genera automaticamente la logica del codice origine per un processo. Puoi modificare questo script generato oppure fornire il tuo script personalizzato.

Per specificare script personalizzati in AWS Glue, segui questa procedura generale:

1. Accedi AWS Management Console e apri la AWS Glue console all'indirizzo https://console.aws.amazon.com/glue/.
2. Scegli la scheda Processi ETL, quindi visualizza la sezione Crea processo. Scegli un'opzione per l'editor di script.
3. In This job runs (Questo processo viene eseguito), seleziona una delle opzioni seguenti:
   - Creazione di un nuovo script con codice boilerplate
   - Caricamento e modifica di uno script esistente
4. Nella schermata Dettagli del processo, seleziona il Ruolo IAM richiesto per l'esecuzione dello script personalizzato. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per AWS Glue.
5. Scegli qualsiasi connessione a cui fa riferimento lo script. Questi oggetti sono richiesti per connettersi ai datastore JDBC necessari.

   Un'interfaccia di rete elastica è un'interfaccia di rete virtuale che è possibile collegare a un'istanza in un Virtual Private Cloud (VPC). Scegli l'interfaccia di rete elastica necessaria per connetterti al datastore utilizzato nello script.

6. Fornisci i dettagli di configurazione aggiuntivi, inclusi i parametri, specifici per il tuo tipo di processo. Per ulteriori informazioni sulla configurazione in base al tipo di processo, consulta la sezione Creazione di processi ETL visivi con AWS Glue Studio.
7. Nella scheda Script, incolla o scrivi lo script personalizzato.

Utilizza il contenuto di questa sezione per guidare il processo di scrittura dello script personalizzato.

Per ulteriori informazioni sull'aggiunta di processi in AWS Glue, consulta Creazione di processi ETL visivi con AWS Glue Studio.

Per step-by-step informazioni, consulta il tutorial Aggiungi lavoro nella AWS Glue console.

Script di programmazione Spark

AWS Glue permette di scrivere o generare automaticamente in modo semplice script di estrazione, trasformazione e caricamento (ETL), nonché di testarli ed eseguirli. Questa sezione descrive le estensioni di Apache Spark introdotte da AWS Glue e fornisce esempi di come codificare ed eseguire gli script ETL in Python e Scala.

> **Important**
>
> Versioni diverse di AWS Glue supportano versioni diverse di Apache Spark. Lo script personalizzato deve essere compatibile con la versione di Apache Spark supportata. Per ulteriori informazioni sulle versioni AWS Glue, consulta Glue version job property.

Argomenti

- Tutorial: Scrivere uno script di AWS Glue for Spark
- Programmazione di script ETL AWS Glue in PySpark
- Programmazione di script ETL AWS Glue in Scala
- Funzionalità e ottimizzazioni per la programmazione degli script ETL di AWS Glue per Spark

Tutorial: Scrivere uno script di AWS Glue for Spark

Questo tutorial ti introduce al processo di scrittura degli script AWS Glue. È possibile eseguire script in base a una pianificazione con processi o in modo interattivo con sessioni. Per ulteriori

informazioni sui processi, consulta Creazione di processi ETL visivi con AWS Glue Studio. Per ulteriori informazioni, sulle sessioni interattive, consulta the section called “Panoramica delle sessioni interattive in AWS Glue”.

L'editor visivo di AWS Glue Studio offre un'interfaccia grafica senza codice per la creazione di lavori AWS Glue. AWS Glue gli script ai lavori visivi. Danno accesso al set esteso di strumenti disponibili per lavorare con i programmi Apache Spark. Puoi accedere alle API Spark native e alle librerie AWS Glue che facilitano i flussi di lavoro di estrazione, trasformazione e caricamento (ETL) dall'interno di uno script Glue. AWS

In questo tutorial, estrai, trasformi e carichi un set di dati di multe per il parcheggio. Lo script che esegue questo lavoro è identico nella forma e nella funzione a quello generato in Making ETL easy with AWS Glue Studio sul AWS Big Data Blog, che introduce l'editor visivo di AWS Glue Studio. Eseguendo questo script in un job, è possibile confrontarlo con i lavori visivi e vedere come funzionano gli script AWS Glue ETL. Questo ti prepara a utilizzare funzionalità aggiuntive che non sono ancora disponibili nei processi visivi.

Questo tutorial utilizza il linguaggio e le librerie Python. Funzionalità simili sono disponibili in Scala. Dopo aver seguito questo tutorial, dovresti essere in grado di generare e ispezionare uno script Scala di esempio per capire come eseguire il processo di scrittura dello script Scala AWS Glue ETL.

Prerequisiti

Di seguito sono elencati i requisiti per questo tutorial:

- Gli stessi prerequisiti del post del blog di AWS Glue Studio, che ti spiega come eseguire un AWS CloudFormation modello.

  Questo modello utilizza il AWS Glue Data Catalog per gestire il set di dati dei biglietti di parcheggio disponibile in`s3://aws-bigdata-blog/artifacts/gluestudio/`. Questa operazione crea le risorse seguenti, a cui verrà fatto riferimento:

- AWS Glue StudioRuolo: il ruolo IAM per eseguire i processi AWS Glue
- AWS Glue StudioBucket Amazon S3: il nome del bucket Amazon S3 per archiviare i file relativi al blog
- AWS Glue StudioTicketsYYZDB: il database del Catalogo dati di AWS Glue
- AWS Glue StudioTableTickets— Tabella Data Catalog da utilizzare come fonte
- AWS Glue StudioTableTrials— Tabella Data Catalog da utilizzare come fonte

- AWS Glue StudioParkingTicketCount — Tabella Data Catalog da utilizzare come destinazione
- Lo script generato nel post del blog di AWS Glue Studio. Nel caso in cui il post venga modificato, è possibile trovare lo script anche nel testo seguente.

Generazione di uno script di esempio

Puoi usare l'editor visivo di AWS Glue Studio come potente strumento di generazione di codice per creare uno scaffold per lo script che desideri scrivere. Questo strumento verrà utilizzato per creare uno script di esempio.

Se preferisci saltare queste fasi, puoi utilizzare lo script seguente.

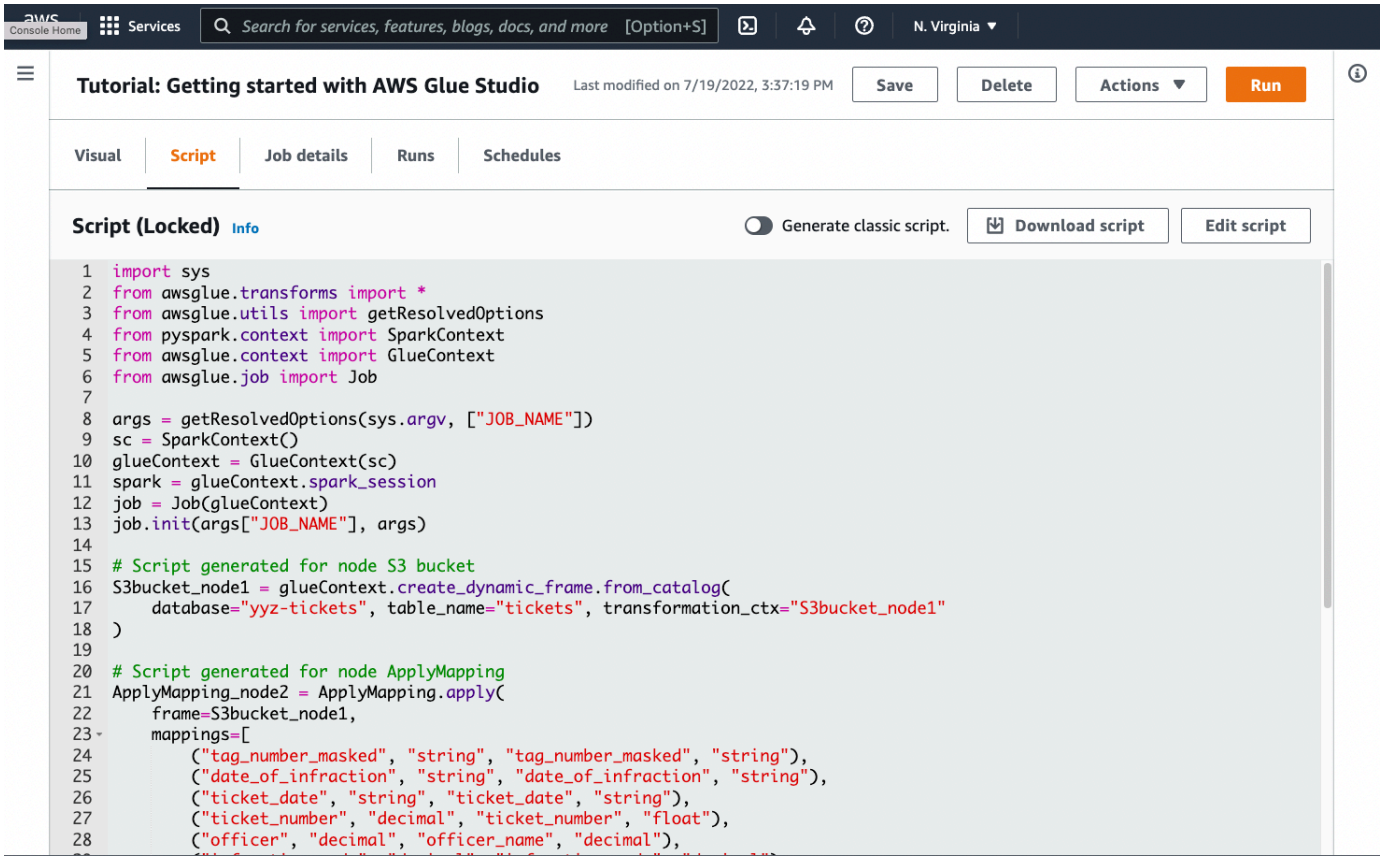
Script di esempio per il tutorial

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 bucket
S3bucket_node1 = glueContext.create_dynamic_frame.from_catalog(
    database="yyz-tickets", table_name="tickets", transformation_ctx="S3bucket_node1"
)

# Script generated for node ApplyMapping
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3bucket_node1,
    mappings=[
        ("tag_number_masked", "string", "tag_number_masked", "string"),
        ("date_of_infraction", "string", "date_of_infraction", "string"),
        ("ticket_date", "string", "ticket_date", "string"),
        ("ticket_number", "decimal", "ticket_number", "float"),
        ("officer", "decimal", "officer_name", "decimal"),
```

```
        ("infraction_code", "decimal", "infraction_code", "decimal"),
        ("infraction_description", "string", "infraction_description", "string"),
        ("set_fine_amount", "decimal", "set_fine_amount", "float"),
        ("time_of_infraction", "decimal", "time_of_infraction", "decimal"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node S3 bucket
S3bucket_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="glueparquet",
    connection_options={"path": "s3://DOC-EXAMPLE-BUCKET", "partitionKeys": []},
    format_options={"compression": "gzip"},
    transformation_ctx="S3bucket_node3",
)

job.commit()
```

Generazione di uno script di esempio

1. Completa il tutorial di AWS Glue Studio. Per completare questo tutorial, vedi Creazione di un lavoro in AWS Glue Studio da un lavoro di esempio.
2. Passa alla scheda Script nella pagina del processo, come mostrato nello screenshot seguente:



3. Copia tutto il contenuto della scheda Script. Impostando il linguaggio dello script nella sezione Job details (Dettagli del processo), puoi passare dalla generazione di codice Python a Scala e viceversa.

Fase 1. Creare un processo e incollare lo script

In questo passaggio, si crea un lavoro AWS Glue in AWS Management Console. Questo imposta una configurazione che consente a AWS Glue di eseguire lo script. e si crea contemporaneamente uno spazio dove archiviarlo e modificarlo.

Per creare un lavoro

1. Nel AWS Management Console, vai alla landing page di AWS Glue.
2. Nel riquadro di navigazione laterale, scegli Jobs (Processi).
3. Scegli Editor di script Spark in Creazione di processo, quindi scegli Crea.
4. Facoltativo: incolla il testo completo dello script nel riquadro Script. In alternativa, puoi seguire il tutorial.

Fase 2. Importa librerie AWS Glue

È necessario impostare lo script in modo che interagisca con il codice e la configurazione che sono stati definiti all'esterno dello script. Questo lavoro viene svolto dietro le quinte di AWS Glue Studio.

In questa fase, procedi secondo quanto descritto di seguito.

- Importa e inizializza un oggetto `GlueContext`. Questa è l'importazione più importante dal punto di vista della scrittura dello script Ciò espone i metodi standard per la definizione dei set di dati di origine e di destinazione, ossia il punto di partenza per qualsiasi script ETL. Per ulteriori informazioni sulla classe `GlueContext`, consulta la pagina Classe GlueContext.
- Inizializza `SparkContext` e `SparkSession`. Questi consentono di configurare il motore Spark disponibile all'interno del lavoro AWS Glue. Non sarà necessario utilizzarli direttamente negli script introduttivi di AWS Glue.
- Richiama `getResolvedOptions` per preparare gli argomenti del processo da utilizzare all'interno dello script. Per ulteriori informazioni sulla risoluzione dei parametri di processo, consulta the section called “getResolvedOptions”.
- Inizializza un `Job`. L'`Job`oggetto imposta la configurazione e tiene traccia dello stato di varie funzioni opzionali di AWS Glue. Lo script può essere eseguito senza un oggetto `Job`, tuttavia la procedura consigliata consiste nell'inizializzarlo in modo da non essere confusi da un'eventuale integrazione successiva di queste funzionalità.

  Una di queste è rappresentata dai segnalibri di processo, che puoi configurare facoltativamente in questo tutorial. Per informazioni sui segnalibri di processo, consulta la sezione the section called “Facoltativo - Abilita i segnalibri di processo”.

In questa procedura si scriverà il codice seguente. Questo codice è una parte dello script di esempio generato.

```
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
```

```
job = Job(glueContext)
job.init(args["JOB_NAME"], args)
```

Per importare le librerie AWS Glue

- Copia questa sezione del codice e incollala nell'editor Script.

> **Note**
>
> La copia del codice potrebbe essere considerata una pratica di ingegneria non consigliata. In questo tutorial, ti suggeriamo questo per incoraggiarti a denominare in modo coerente le tue variabili principali in tutti gli script ETL di AWS Glue.

Fase 3. Estrazione di dati da un'origine

In qualsiasi processo ETL, è innanzitutto necessario definire un set di dati di origine che si vuole modificare. Nell'editor visivo di AWS Glue Studio, fornisci queste informazioni creando un nodo Source.

In questa fase, fornisci al metodo `create_dynamic_frame.from_catalog` i parametri `database` e `table_name` per estrarre i dati da un'origine configurata nel Data Catalog AWS Glue.

Nella fase precedente hai inizializzato un oggetto `GlueContext` Utilizzi questo oggetto per trovare i metodi usati per configurare le origini, ad esempio `create_dynamic_frame.from_catalog`.

In questa procedura si scriverà il codice seguente utilizzando `create_dynamic_frame.from_catalog`. Questo codice è una parte dello script di esempio generato.

```
S3bucket_node1 = glueContext.create_dynamic_frame.from_catalog(
    database="yyz-tickets", table_name="tickets", transformation_ctx="S3bucket_node1"
    )
```

Estrazione di dati da un'origine

1. Esamina la documentazione per trovare un metodo `GlueContext` per estrarre dati da una fonte definita nel AWS Glue Data Catalog. Questi metodi sono documentati in the section called “GlueContext”. Scegli il metodo create_dynamic_frame.from_catalog. Richiama questo metodo in `glueContext`.

2. Esamina la documentazione relativa a `create_dynamic_frame.from_catalog`. Questo metodo richiede i parametri `database` e `table_name`. Fornisci i parametri necessari per `create_dynamic_frame.from_catalog`.

   Il AWS Glue Data Catalog memorizza le informazioni sulla posizione e il formato dei dati di origine ed è stato impostato nella sezione dei prerequisiti. Non è necessario fornire direttamente allo script tali informazioni.

3. Facoltativo: fornisci al metodo il parametro `transformation_ctx` per supportare i segnalibri di processo. Per informazioni sui segnalibri di processo, consulta la sezione the section called “Facoltativo - Abilita i segnalibri di processo”.

> **Note**
>
> Metodi comuni per l'estrazione dei dati
> the section called “create_dynamic_frame_from_catalog”viene utilizzato per connettersi alle tabelle nel AWS Glue Data Catalog.
> Se hai bisogno di fornire direttamente al processo una configurazione che descriva la struttura e la posizione dell'origine, consulta il metodo the section called “create_dynamic_frame_from_options”. Dovrai fornire parametri più dettagliati per la descrizione dei dati rispetto a quando utilizzi `create_dynamic_frame.from_catalog`. Consulta la documentazione supplementare su `format_options` e `connection_parameters` per identificare i parametri obbligatori. Per una spiegazione su come fornire allo script informazioni sul formato dei dati di origine, consulta la sezione the section called “Opzioni del formato dei dati”. Per una spiegazione su come fornire allo script informazioni sulla posizione dei dati di origine, consulta la sezione the section called “Parametri di connessione”.
> Se stai leggendo le informazioni da un'origine di streaming, fornisci al processo le informazioni di origine tramite i metodi the section called “create_data_frame_from_catalog” o the section called “create_data_frame_from_options”. Nota: questi metodi restituiscono `DataFrames` di Apache Spark.
> Il codice generato effettua una chiamata a `create_dynamic_frame.from_catalog`, mentre la documentazione fa riferimento a `create_dynamic_frame_from_catalog`. Questi metodi richiamano in definitiva lo stesso codice e sono stati inclusi per consentire di scrivere un codice più pulito. Puoi verificarlo visualizzando il sorgente del nostro wrapper Python, disponibile all'indirizzo aws-glue-libs.

Fase 4. Trasformare i dati con AWS Glue

Dopo aver estratto i dati di origine in un processo ETL, è necessario specificare il modo in cui modificare i dati. Fornisci queste informazioni creando un nodo Transform nell'editor visivo di AWS Glue Studio.

In questa fase, fornisci al metodo `ApplyMapping` una mappa dei nomi e dei tipi di campo attuali e desiderati per trasformare il `DynamicFrame`.

Esegui le trasformazioni seguenti.

- Rilascia le quattro chiavi `location` e `province`.
- Modifica il nome di `officer` in `officer_name`.
- Modifica il tipo di `ticket_number` e `set_fine_amount` in `float`.

`create_dynamic_frame.from_catalog` fornisce un oggetto `DynamicFrame`. A `DynamicFrame` rappresenta un set di dati in AWS Glue. AWS Le Glue Transform sono operazioni che cambianoDynamicFrames.

> **Note**
>
> Che cos'è una `DynamicFrame`?
> Un `DynamicFrame` è un'astrazione che consente di collegare un set di dati con una descrizione dei nomi e dei tipi di voci presenti nei dati. In Apache Spark esiste un'astrazione simile chiamata a. DataFrame Per una spiegazione di DataFrames, consulta Spark SQL Guide.
> I `DynamicFrames`, consentono di descrivere gli schemi del set di dati in modo dinamico. Prendiamo in considerazione un set di dati con una colonna dei prezzi, in cui alcune voci memorizzano il prezzo come stringa e altre il prezzo come doppio. AWS Glue calcola uno schema on-the-fly: crea un record autodescrittivo per ogni riga.
> I campi non coerenti (come il prezzo) sono rappresentati esplicitamente con un tipo (`ChoiceType`) nello schema del riquadro. Puoi affrontare il problema dei campi non coerenti eliminandoli con `DropFields` o risolvendoli con `ResolveChoice`. Queste trasformazioni che sono disponibili su `DynamicFrame`. Puoi quindi riscrivere i dati sul data lake con `writeDynamicFrame`.

È possibile richiamare la maggior parte di queste trasformazioni dai metodi della classe `DynamicFrame`, ottenendo così script più leggibili. Per ulteriori informazioni su `DynamicFrame`, consulta the section called “DynamicFrame”.

In questa procedura si scriverà il codice seguente utilizzando `ApplyMapping`. Questo codice è una parte dello script di esempio generato.

```
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3bucket_node1,
    mappings=[
        ("tag_number_masked", "string", "tag_number_masked", "string"),
        ("date_of_infraction", "string", "date_of_infraction", "string"),
        ("ticket_date", "string", "ticket_date", "string"),
        ("ticket_number", "decimal", "ticket_number", "float"),
        ("officer", "decimal", "officer_name", "decimal"),
        ("infraction_code", "decimal", "infraction_code", "decimal"),
        ("infraction_description", "string", "infraction_description", "string"),
        ("set_fine_amount", "decimal", "set_fine_amount", "float"),
        ("time_of_infraction", "decimal", "time_of_infraction", "decimal"),
    ],
    transformation_ctx="ApplyMapping_node2",
)
```

Per trasformare i dati con AWS Glue

1. Esamina la documentazione per identificare una trasformazione volta a modificare ed eliminare i campi. Per informazioni dettagliate, vedi the section called “GlueTransform”. Seleziona la trasformazione `ApplyMapping`. Per ulteriori informazioni su `ApplyMapping`, consulta the section called “ApplyMapping”. Richiama `apply` nell'oggetto di trasformazione `ApplyMapping`.

   > **Note**
   >
   > Cos'è `ApplyMapping`?
   > `ApplyMapping` prende un `DynamicFrame` e lo trasforma. Prende un elenco di tuple che rappresentano le trasformazioni sui campi, una "mappatura". I primi due elementi della tupla, ovvero un nome e un tipo di campo, vengono utilizzati per identificare un campo nel riquadro. Anche gli altri due parametri rappresentano un nome e un tipo di campo.

ApplyMapping converte il campo di origine nel nome di destinazione e ne digita uno nuovo`DynamicFrame`, che restituisce. I campi non forniti verranno eliminati dal valore restituito.
Invece di effettuare una chiamata ad `apply`, è possibile richiamare la stessa trasformazione con il metodo `apply_mapping` nell'oggetto `DynamicFrame`, creando così un codice più fluido e leggibile. Per ulteriori informazioni, consulta the section called "apply_mapping".

2. Esamina la documentazione relativa ad `ApplyMapping` per identificare i parametri obbligatori. Per informazioni, consulta the section called "ApplyMapping". Noterai che questo metodo richiede i parametri `frame` e `mappings`. Fornisci i parametri necessari per `ApplyMapping`.
3. Facoltativo – Fornisci `transformation_ctx` al metodo per supportare i segnalibri di processo. Per informazioni sui segnalibri di processo, consulta la sezione the section called "Facoltativo - Abilita i segnalibri di processo".

> **Note**
>
> Funzionalità di Apache Spark
> Forniamo trasformazioni per semplificare i flussi di lavoro ETL all'interno del tuo processo. Avrai inoltre accesso alle librerie che sono state create per scopi più generali e disponibili in un programma Spark nel processo. Per utilizzarle, è necessario eseguire la conversione tra `DynamicFrame` e `DataFrame`.
> È possibile creare un `DataFrame` con the section called "toDF". Quindi, puoi utilizzare i metodi disponibili su DataFrame per trasformare il tuo set di dati. Per ulteriori informazioni su questi metodi, vedere DataFrame. È quindi possibile eseguire la conversione all'indietro con the section called "fromDF" per utilizzare le operazioni AWS Glue per caricare la cornice su un bersaglio.

Fase 5. Caricare i dati in una destinazione

In genere, i dati trasformati vengono archiviati in una posizione diversa da quella di origine. Questa operazione viene eseguita creando un nodo di destinazione nell'editor visivo di AWS Glue Studio.

In questa fase fornisci al metodo `write_dynamic_frame.from_options` i parametri `connection_type`, `connection_options`, `format` e `format_options` per caricare i dati in un bucket di destinazione di Amazon S3.

Nella fase 1 hai inizializzato un oggetto `GlueContext`. In AWS Glue, è qui che troverai i metodi utilizzati per configurare gli obiettivi, proprio come i sorgenti.

In questa procedura si scriverà il codice seguente utilizzando `write_dynamic_frame.from_options`. Questo codice è una parte dello script di esempio generato.

```
S3bucket_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="glueparquet",
    connection_options={"path": "s3://DOC-EXAMPLE-BUCKET", "partitionKeys": []},
    format_options={"compression": "gzip"},
    transformation_ctx="S3bucket_node3",
    )
```

Caricare i dati in una destinazione

1. Esamina la documentazione per trovare un metodo adatto a caricare i dati in un bucket Amazon S3 di destinazione. Questi metodi sono documentati in the section called “GlueContext”. Scegli il metodo the section called “write_dynamic_frame_from_options”. Richiama questo metodo in `glueContext`.

   > **Note**
   >
   > Metodi comuni per il caricamento dei dati
   > `write_dynamic_frame.from_options` è il metodo più comunemente usato per caricare i dati. Supporta tutti i target disponibili in AWS Glue.
   > Se stai scrivendo su un target JDBC definito in una connessione AWS Glue, usa il the section called “write_dynamic_frame_from_jdbc_conf” metodo. AWS Le connessioni Glue memorizzano informazioni su come connettersi a un'origine dati. Ciò elimina la necessità di fornire tali informazioni in `connection_options`. Tuttavia, devi comunque utilizzare `connection_options` per fornire `dbtable`.
   > `write_dynamic_frame.from_catalog` non è un metodo comune per caricare i dati. Questo metodo aggiorna il AWS Glue Data Catalog senza aggiornare il set di dati sottostante e viene utilizzato in combinazione con altri processi che modificano il set di dati sottostante. Per ulteriori informazioni, consulta the section called “Creazione di

tabelle, aggiornamento dello schema e aggiunta di nuove partizioni nel catalogo dati da processi ETL AWS Glue”.

2. Esamina la documentazione relativa a the section called “write_dynamic_frame_from_options”. Questo metodo richiede `frame`, `connection_type`, `format`, `connection_options`, `format_options`. Richiama questo metodo in `glueContext`.

   a. Consulta la documentazione aggiuntiva relativa a `format_options` e `format` per identificare i parametri necessari. Per una spiegazione dei formati di dati, consulta la sezione the section called “Opzioni del formato dei dati”.

   b. Consulta la documentazione aggiuntiva relativa a `connection_type` e `connection_options` per identificare i parametri necessari. Per una spiegazione delle connessioni, consulta la sezione the section called “Parametri di connessione”.

   c. Fornisci i parametri necessari per `write_dynamic_frame.from_options`. Questo metodo ha una configurazione simile a `create_dynamic_frame.from_options`.

3. Facoltativo – Fornisci `transformation_ctx` al metodo `write_dynamic_frame.from_options` per supportare i segnalibri di processo. Per informazioni sui segnalibri di processo, consulta la sezione the section called “Facoltativo - Abilita i segnalibri di processo”.

Fase 6. Commit dell'oggetto **`Job`**

Nella fase 1 hai inizializzato un oggetto `Job`. Dovrai concludere manualmente il suo ciclo di vita alla fine dello script. Alcune funzionalità facoltative richiedono tale oggetto per funzionare correttamente. Questo lavoro viene svolto dietro le quinte di AWS Glue Studio.

In questa fase, effettua una chiamata al metodo `commit` nell'oggetto `Job`.

In questa procedura si scriverà il codice seguente. Questo codice è una parte dello script di esempio generato.

```
job.commit()
```

Commit dell'oggetto **`Job`**

1. Se non l'hai già fatto, esegui le fasi facoltative descritte nelle sezioni precedenti per includere `transformation_ctx`.

2. Chiama `commit`.

Facoltativo - Abilita i segnalibri di processo

In ogni fase precedente, ti è stato chiesto di impostare i parametri `transformation_ctx`. Questa operazione è correlata a una funzionalità denominata segnalibri di processo.

Con i segnalibri di processo risparmi tempo e denaro grazie a processi eseguiti su base ricorrente rispetto a set di dati in cui è possibile tracciare facilmente il lavoro precedente. I segnalibri Job tengono traccia dell'avanzamento di una trasformazione AWS Glue su un set di dati delle esecuzioni precedenti. Tracciando dove sono terminate le esecuzioni precedenti, AWS Glue può limitare il lavoro alle righe che non ha mai elaborato prima. Per ulteriori informazioni sui segnalibri di processo, consultare the section called “Monitoraggio dei dati elaborati mediante segnalibri di processo”.

Per abilitare i segnalibri di processo, aggiungi prima le informazioni `transformation_ctx` nelle funzioni fornite, come descritto negli esempi precedenti. Lo stato del segnalibro di processo viene mantenuto tra le esecuzioni. I parametri `transformation_ctx` sono chiavi utilizzate per accedere a tale stato. Da sole, queste istruzioni non servono a nulla. È necessario attivare la funzionalità nella configurazione del processo.

In questa procedura attivi i segnalibri di processo utilizzando la AWS Management Console.

Abilitazione dei segnalibri di processo

1. Passa alla sezione Job details (Dettagli del processo) del processo corrispondente.
2. Imposta Job bookmark (Segnalibro di processo) su Enable (Abilita).

Fase 7. Esecuzione del codice come processo

In questa fase, esegui il processo per verificare di aver completato correttamente questo tutorial. Questo viene fatto con il clic di un pulsante, come nell'editor visivo di AWS Glue Studio.

Esecuzione del codice come processo

1. Scegli Untitled job (Processo senza titolo) sulla barra del titolo per modificare e impostare il nome del processo.
2. Passa alla scheda Job details (Dettagli del lavoro). Assegna al processo un IAM Role (Ruolo IAM). Puoi usare quello creato dal AWS CloudFormation modello nei prerequisiti per il tutorial di

AWS Glue Studio. Se hai completato quel tutorial, dovrebbe essere disponibile come `AWS Glue StudioRole`.

3. Scegli Save (Salva) per salvare lo script.
4. Scegli Run (Esegui) per eseguire il processo.
5. Passa alla scheda Runs (Esecuzioni) per verificare il completamento del processo.
6. Passa a *`DOC-EXAMPLE-BUCKET`*, vale a dire la destinazione di `write_dynamic_frame.from_options`. Verifica che l'output corrisponda alle tue aspettative.

Per ulteriori informazioni sulla configurazione e la gestione dei processi, consulta la sezione the section called “Fornire i propri script personalizzati”.

Ulteriori informazioni

Le librerie e i metodi Apache Spark sono disponibili negli script AWS Glue. Per comprendere quali operazioni è possibile eseguire con le librerie incluse, consulta la documentazione di Spark. Per ulteriori informazioni, consulta la sezione esempi del repository di orgine di Spark.

AWS Glue 2.0+ include diverse librerie Python comuni per impostazione predefinita. Esistono anche meccanismi per caricare le proprie dipendenze in un lavoro AWS Glue in un ambiente Scala o Python. Per ulteriori informazioni sulle dipendenze Python, consulta the section called “Librerie Python”.

Per altri esempi di come usare le funzionalità di AWS Glue in Python, vedi. the section called “Esempi Python” I processi in Scala e Python presentano una condizione di parità di funzioni, per cui gli esempi relativi a Python consentono di comprendere l'esecuzione di un lavoro simile in Scala.

Programmazione di script ETL AWS Glue in PySpark

Puoi trovare esempi di codice Python e utilità per AWS Glue nel repository di esempi AWS Glue nel sito Web GitHub.

Uso di Python con AWS Glue

AWS Glue supporta un'estensione del dialetto Python PySpark per lo scripting dei processi di estrazione, trasformazione e caricamento (ETL). In questa sezione viene descritto come usare Python negli script ETL con l'API AWS Glue.

- Configurazione per l'uso di Python con AWS Glue

- Chiamata di API AWS Glue in Python
- Uso di librerie Python con AWS Glue
- Esempi di codice Python in AWS Glue

Estensioni PySpark in AWS Glue

AWS Glue ha creato le estensioni seguenti per il dialetto PySpark Python.

- Accesso ai parametri utilizzando `getResolvedOptions`
- Tipi di estensione PySpark
- DynamicFrame classe
- Classe DynamicFrameCollection
- Classe DynamicFrameWriter
- DynamicFrameReader classe
- Classe GlueContext

Trasformazioni PySpark AWS Glue

AWS Glue ha creato le classi di trasformazione seguenti da usare nelle operazioni ETL PySpark.

- Classe di base GlueTransform
- Classe ApplyMapping
- Classe DropFields
- Classe DropNullFields
- Classe ErrorsAsDynamicFrame
- Classe FillMissingValues
- Classe filtro
- Classe FindIncrementalMatches
- Classe FindMatches
- Classe FlatMap
- Classe join
- Classe mappatura

- Classe MapToCollection
- mergeDynamicFrame
- Classe relazionalizzazione
- Classe RenameField
- Classe ResolveChoice
- Classe SelectFields
- Classe SelectFromCollection
- Classe Spigot
- Classe SplitFields
- Classe SplitRows
- Classe unbox
- Classe UnnestFrame

Configurazione per l'uso di Python con AWS Glue

Utilizza Python per sviluppare script ETL per processi Spark. Le versioni Python supportate per i processi ETL dipendono dalla versione AWS Glue del processo. Per ulteriori informazioni sulle versioni di AWS Glue, consulta Glue version job property.

Per configurare il sistema per l'uso di Python con AWS Glue

Segui queste fasi per installare Python e poter richiamare le API AWS Glue.

1. Se non disponi ancora di Python installato, scaricalo e installalo dalla Python.org download page.
2. Installa AWS Command Line Interface (AWS CLI) come spiegato nella documentazione su AWS CLI.

   Non è necessario disporre di AWS CLI per usare Python. Tuttavia, la sua installazione e la sua configurazione permettono di configurare facilmente AWS con le credenziali dell'account e verificare il funzionamento.
3. Installa SDK for Python (Boto 3) AWS, come documentato nella Boto3 Quickstart.

   Le API delle risorse Boto 3 non sono ancora disponibili per AWS Glue. Al momento, solo le API del client di Boto 3 possono essere utilizzate.

   Per ulteriori informazioni su Boto 3, consulta AWSGuida introduttiva di SDK for Python (Boto3) .

Puoi trovare esempi di codice Python e utilità per AWS Glue nel repository di esempi AWS Glue nel sito Web di GitHub.

Chiamata di API AWS Glue in Python

Le API delle risorse Boto 3 non sono ancora disponibili per AWS Glue. Al momento, solo le API del client di Boto 3 possono essere utilizzate.

Nomi delle API AWS Glue in Python

I nomi delle API AWS Glue in Java e altri linguaggi di programmazione sono in genere costituiti da combinazioni di lettere maiuscole e minuscole. Tuttavia, quando vengono chiamati da Python, questi nomi generici vengono modificati in minuscolo, con le parti del nome separate da caratteri di sottolineatura per renderli più adatti a Python. Nella documentazione di riferimento API AWS Glue, questi nomi adatti a Python sono elencati in parentesi dopo i nomi generici CamelCased.

Tuttavia, anche se i nomi delle API AWS Glue vengono trasformati in lettere minuscole, i nomi dei relativi parametri rimangono in maiuscolo. È importante ricordarlo, perché i parametri devono essere passati usando il nome quando si chiamano le API AWS Glue, come descritto nella sezione seguente.

Passaggio di parametri Python in AWS Glue e accesso ai parametri

Quando in Python si chiamano le API AWS Glue, è preferibile passare i parametri in modo esplicito usando il nome. Ad esempio:

```
job = glue.create_job(Name='sample', Role='Glue_DefaultRole',
                      Command={'Name': 'glueetl',
                               'ScriptLocation': 's3://my_script_bucket/scripts/
my_etl_script.py'})
```

È utile comprendere che Python crea un dizionario delle tuple nome/valore specificate come argomenti di uno script ETL in una Struttura del processo o JobRun struttura. Boto 3 esegue quindi il passaggio ad AWS Glue in formato JSON mediante una chiamata API REST. Questo significa che non puoi fare affidamento sull'ordine degli argomenti al momento dell'accesso nello script.

Ad esempio, supponiamo che stai avviando un `JobRun` in una funzione del gestore Lambda di Python e che desideri specificare più parametri. Il tuo codice potrebbe essere simile a quanto segue:

```
from datetime import datetime, timedelta
```

```
client = boto3.client('glue')

def lambda_handler(event, context):
  last_hour_date_time = datetime.now() - timedelta(hours = 1)
  day_partition_value = last_hour_date_time.strftime("%Y-%m-%d")
  hour_partition_value = last_hour_date_time.strftime("%-H")

  response = client.start_job_run(
              JobName = 'my_test_Job',
              Arguments = {
                '--day_partition_key':   'partition_0',
                '--hour_partition_key':  'partition_1',
                '--day_partition_value':  day_partition_value,
                '--hour_partition_value': hour_partition_value } )
```

Per accedere a questi parametri in modo affidabile nello script ETL, specificali per nome usando la funzione AWS Glue di `getResolvedOptions` e quindi esegui l'accesso dal dizionario risultante:

```
import sys
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv,
                          ['JOB_NAME',
                           'day_partition_key',
                           'hour_partition_key',
                           'day_partition_value',
                           'hour_partition_value'])
print "The day partition key is: ", args['day_partition_key']
print "and the day partition value is: ", args['day_partition_value']
```

Se desideri passare un argomento che è una stringa JSON annidata, per preservare il valore del parametro man mano che viene passato al processo ETL AWS Glue, devi codificare la stringa del parametro prima di avviare l'esecuzione del processo, quindi decodificare la stringa del parametro prima di riferirla allo script di processo. Ad esempio, considera la seguente stringa di argomento:

```
glue_client.start_job_run(JobName = "gluejobname", Arguments={
"--my_curly_braces_string": '{"a": {"b": {"c": [{"d": {"e": 42}}]}}}'
})
```

Per passare correttamente questo parametro, devi codificare l'argomento come una stringa codificata Base64.

```
import base64
...
sample_string='{"a": {"b": {"c": [{"d": {"e": 42}}]}}}'
sample_string_bytes = sample_string.encode("ascii")

base64_bytes = base64.b64encode(sample_string_bytes)
base64_string = base64_bytes.decode("ascii")
...
glue_client.start_job_run(JobName = "gluejobname", Arguments={
"--my_curly_braces_string": base64_bytes})
...
sample_string_bytes = base64.b64decode(base64_bytes)
sample_string = sample_string_bytes.decode("ascii")
print(f"Decoded string: {sample_string}")
...
```

Esempio: creazione ed esecuzione di un processo

L'esempio seguente mostra come chiamare le API AWS Glue usando Python, per creare ed eseguire un processo ETL.

Per creare ed eseguire un processo

1. Crea un'istanza del client AWS Glue:

   ```
   import boto3
   glue = boto3.client(service_name='glue', region_name='us-east-1',
                 endpoint_url='https://glue.us-east-1.amazonaws.com')
   ```

2. Crea un processo. Devi utilizzare `glueetl` come nome per il comando ETL, come mostrato nel codice seguente:

   ```
   myJob = glue.create_job(Name='sample', Role='Glue_DefaultRole',
                           Command={'Name': 'glueetl',
                                    'ScriptLocation': 's3://my_script_bucket/
   scripts/my_etl_script.py'})
   ```

3. Avvia una nuova esecuzione del processo creato nella fase precedente:

   ```
   myNewJobRun = glue.start_job_run(JobName=myJob['Name'])
   ```

4. Ottieni lo stato del processo:

```
status = glue.get_job_run(JobName=myJob['Name'], RunId=myNewJobRun['JobRunId'])
```

5. Stampa lo stato attuale del processo eseguito:

```
print(status['JobRun']['JobRunState'])
```

Uso di librerie Python con AWS Glue

AWS Glue consente di installare moduli e librerie Python aggiuntivi da utilizzare con ETL AWS Glue.

Argomenti

- Installazione di moduli Python aggiuntivi in AWS Glue 2.0 con pip
- Inclusione di file Python con funzionalità native PySpark
- Script di programmazione che utilizzano trasformazioni visive
- Moduli Python già forniti in AWS Glue
- Compressione delle librerie per l'inclusione
- Caricamento di librerie Python in un endpoint di sviluppo
- Usare le librerie Python in un lavoro o JobRun

Installazione di moduli Python aggiuntivi in AWS Glue 2.0 con pip

AWS Glue utilizza Python Package Installer (pip3) per installare moduli aggiuntivi utilizzati da ETL AWS Glue. Puoi utilizzare il parametro `--additional-python-modules` con un elenco di moduli Python separati da virgole per aggiungere un nuovo modulo o modificare la versione di un modulo esistente. Puoi installare distribuzioni personalizzate di una libreria caricando la distribuzione in Amazon S3 e successivamente includendo il percorso dell'oggetto Amazon S3 nell'elenco dei moduli.

Puoi passare opzioni aggiuntive a pip3 tramite il parametro `--python-modules-installer-option`. Ad esempio, è possibile passare `"--upgrade"` per aggiornare i pacchetti specificati da `"--additional-python-modules"`. Per altri esempi, consulta Creazione di moduli Python da una ruota per carichi di lavoro Spark ETL con Glue 2.0. AWS

Se le tue dipendenze in Python dipendono transitivamente dal codice compilato nativo, potresti riscontrare la seguente limitazione: AWS Glue non supporta la compilazione di codice nativo nell'ambiente di lavoro. Tuttavia, i job AWS Glue vengono eseguiti in un ambiente Amazon Linux 2.

Potresti essere in grado di fornire le tue dipendenze native in un formato compilato tramite un file wheel distribuibile.

Ad esempio, per aggiornare o aggiungere un nuovo modulo `scikit-learn` usa la seguente chiave-valore: `"--additional-python-modules", "scikit-learn==0.21.3"`.

Inoltre, all'interno dell'opzione `--additional-python-modules` puoi specificare un percorso Amazon S3 per un modulo ruota Python. Per esempio:

```
--additional-python-modules s3://aws-glue-native-spark/tests/j4.2/ephem-3.7.7.1-cp37-cp37m-linux_x86_64.whl,s3://aws-glue-native-spark/tests/j4.2/fbprophet-0.6-py3-none-any.whl,scikit-learn==0.21.3
```

È possibile `--additional-python-modules` specificarli nel campo Job parameters della AWS Glue console o modificando gli argomenti del lavoro nell' AWS SDK. Per ulteriori informazioni sulla configurazione dei parametri di processo, consulta the section called “Parametri del processo”.

Inclusione di file Python con funzionalità native PySpark

AWS Glueutilizza PySpark per includere file Python nei lavori ETL di AWS Glue. Quando possibile, ti consigliamo di usare `--additional-python-modules` per gestire le dipendenze. Puoi utilizzare il parametro del processo `--extra-py-files` per includere i file Python. Le dipendenze devono essere ospitate in Amazon S3 e il valore dell'argomento deve essere un elenco delimitato da virgole di percorsi Amazon S3 senza spazi. Questa funzionalità si comporta come la gestione delle dipendenze Python che useresti con Spark. Per ulteriori informazioni sulla gestione delle dipendenze di Python in Spark, consulta la pagina Utilizzo delle funzionalità PySpark native nella documentazione di Apache Spark. `--extra-py-files`è utile nei casi in cui il codice aggiuntivo non è incluso nel pacchetto o quando si sta migrando un programma Spark con una toolchain esistente per la gestione delle dipendenze. Affinché gli strumenti di dipendenza siano gestibili, sarà necessario raggruppare le dipendenze prima di inviarle.

Script di programmazione che utilizzano trasformazioni visive

Quando crei un lavoro AWS Glue utilizzando l'interfaccia visiva di AWS Glue Studio, puoi trasformare i tuoi dati con nodi di trasformazione dati gestiti e trasformazioni visive personalizzate. Per ulteriori informazioni sui nodi di trasformazione dei dati gestiti, consultathe section called “Modifica dei nodi di trasformazione dei dati gestiti da AWS Glue”. Per ulteriori informazioni sulle trasformazioni visive personalizzate, vederethe section called “ Trasformazioni visive personalizzate ”. Gli script che utilizzano trasformazioni visive possono essere generati solo quando il linguaggio del lavoro è impostato per utilizzare Python.

Quando si genera un lavoro AWS Glue utilizzando trasformazioni visive, AWS Glue Studio includerà queste trasformazioni nell'ambiente di runtime utilizzando il `--extra-py-files` parametro nella configurazione del lavoro. Per ulteriori informazioni sui parametri di processo, consulta the section called “Parametri del processo”. Quando si apportano modifiche a uno script o a un ambiente di runtime generato, è necessario mantenere questa configurazione del lavoro affinché lo script venga eseguito correttamente.

Moduli Python già forniti in AWS Glue

Puoi modificare la versione dei moduli disponibili con il parametro di processo `--additional-python-modules`.

AWS Glue version 2.0

AWS Glue versione 2.0 supporta i moduli Python seguenti per impostazione predefinita:

- avro-python3==1.10.0
- awscli==1.27.60
- boto3==1.12.4
- botocore==1.15.4
- certifi==2019.11.28
- chardet==3.0.4
- click==8.1.3
- colorama==0.4.4
- cycler==0.10.0
- Cython==0.29.15
- docutils==0.15.2
- enum34==1.1.9
- fsspec==0.6.2
- idna==2.9
- importlib-metadata==6.0.0
- jmespath==0.9.4
- joblib==0.14.1
- kiwisolver==1.1.0

- matplotlib==3.1.3
- mpmath==1.1.0
- nltk==3.5
- numpy==1.18.1
- pandas==1.0.1
- patsy==0.5.1
- pmdarima==1.5.3
- ptvsd==4.3.2
- pyarrow==0.16.0
- pyasn1==0.4.8
- pydevd==1.9.0
- pyhocon==0.3.54
- PyMySQL==0.9.3
- pyparsing==2.4.6
- python-dateutil==2.8.1
- pytz==2019.3
- PyYAML==5.3.1
- regex==2022.10.31
- requests==2.23.0
- rsa==4.7.2
- s3fs==0.4.0
- s3transfer==0.3.3
- scikit-learn==0.22.1
- scipy==1.4.1
- setuptools==45.2.0
- six==1.14.0
- Spark==1.0
- statsmodels==0.11.1
- subprocess32==3.5.4
- sympy==1.5.1

- tbats==1.0.9
- tqdm==4.64.1
- typing-extensions==4.4.0
- urllib3==1.25.8
- wheel==0.35.1
- zipp==3.12.0

AWS Glue versione 3.0

AWS Glue versione 3.0 supporta i moduli Python seguenti per impostazione predefinita:

- aiobotocore==1.4.2
- aiohttp==3.8.3
- aioitertools==0.11.0
- aiosignal==1.3.1
- async-timeout==4.0.2
- asynctest==0.13.0
- attrs==22.2.0
- avro-python3==1.10.2
- boto3==1.18.50
- botocore==1.21.50
- certifi==2021.5.30
- chardet==3.0.4
- charset-normalizer==2.1.1
- click==8.1.3
- cycler==0.10.0
- Cython==0.29.4
- docutils==0.17.1
- enum34==1.1.10
- frozenlist==1.3.3
- fsspec==2021.8.1

- idna==2.10
- importlib-metadata==6.0.0
- jmespath==0.10.0
- joblib==1.0.1
- kiwisolver==1.3.2
- matplotlib==3.4.3
- mpmath==1.2.1
- multidict==6.0.4
- nltk==3.6.3
- numpy==1.19.5
- packaging==23.0
- pandas==1.3.2
- patsy==0.5.1
- Pillow==9.4.0
- pip==23.0
- pmdarima==1.8.2
- ptvsd==4.3.2
- pyarrow==5.0.0
- pydevd==2.5.0
- pyhocon==0.3.58
- PyMySQL==1.0.2
- pyparsing==2.4.7
- python-dateutil==2.8.2
- pytz==2021.1
- PyYAML==5.4.1
- regex==2022.10.31
- requests==2.23.0
- s3fs==2021.8.1
- s3transfer==0.5.0
- scikit-learn==0.24.2

- scipy==1.7.1
- six==1.16.0
- Spark==1.0
- statsmodels==0.12.2
- subprocess32==3.5.4
- sympy==1.8
- tbats==1.1.0
- threadpoolctl==3.1.0
- tqdm==4.64.1
- typing_extensions==4.4.0
- urllib3==1.25.11
- wheel==0.37.0
- wrapt==1.14.1
- yarl==1.8.2
- zipp==3.12.0

AWS Glue versione 4.0

AWS Glue versione 4.0 supporta i moduli Python seguenti per impostazione predefinita:

- aiobotocore==2.4.1
- aiohttp==3.8.3
- aioitertools==0.11.0
- aiosignal==1.3.1
- async-timeout==4.0.2
- asynctest==0.13.0
- attrs==22.2.0
- avro-python3==1.10.2
- boto3==1.24.70
- botocore==1.27.59
- certifi==2021.5.30

- chardet==3.0.4
- charset-normalizer==2.1.1
- click==8.1.3
- cycler==0.10.0
- Cython==0.29.32
- docutils==0.17.1
- enum34==1.1.10
- frozenlist==1.3.3
- fsspec==2021.8.1
- idna==2.10
- importlib-metadata==5.0.0
- jmespath==0.10.0
- joblib==1.0.1
- caleido==0.2.1
- kiwisolver==1.4.4
- matplotlib==3.4.3
- mpmath==1.2.1
- multidict==6.0.4
- nltk==3.7
- numpy==1.23.5
- packaging==23.0
- pandas == 1.5.1
- patsy==0.5.1
- Pillow==9.4.0
- pip==23.0.1
- trame ==5.16.0
- pmdarima==2.0.1
- ptvsd==4.3.2
- pyarrow==10.0.0
- pydevd==2.5.0

- pyhocon==0.3.58
- PyMySQL==1.0.2
- pyparsing==2.4.7
- python-dateutil==2.8.2
- pytz==2021.1
- PyYAML==6.0.1
- regex==2022.10.31
- requests==2.23.0
- s3fs==2022.11.0
- s3transfer==0.6.0
- scikit-learn==0.24.2
- scipy==1.9.3
- setuptools==49.1.3
- six==1.16.0
- statsmodels==0.13.5
- subprocess32==3.5.4
- sympy==1.8
- tbats==1.1.0
- threadpoolctl==3.1.0
- tqdm==4.64.1
- typing_extensions==4.4.0
- urllib3==1.25.11
- wheel==0.37.0
- wrapt==1.14.1
- yarl==1.8.2
- zipp==3.10.0

Compressione delle librerie per l’inclusione

A meno che una libreria non sia contenuta in un singolo file `.py`, deve essere compressa in un archivio `.zip`. La directory del pacchetto deve trovarsi al livello radice dell'archivio e deve contenere

un file `__init__.py` per il pacchetto. Python sarà in grado di importare il pacchetto nel modo normale.

Se la tua libreria è composta da un singolo modulo Python in un file `.py`, non è necessario trasferirla in un file `.zip`.

Caricamento di librerie Python in un endpoint di sviluppo

Se utilizzi diversi set di librerie per diversi script ETL, puoi impostare un endpoint di sviluppo separato per ciascun set oppure sovrascrivere i file `.zip` della libreria che l'endpoint di sviluppo carica ogni volta che si cambia script.

Puoi utilizzare la console per specificare uno o più file .zip di libreria per un endpoint di sviluppo al momento della creazione. Dopo l'assegnazione di un nome e un ruolo IAM, scegli Script Libraries and job parameters (optional) (Librerie di Script e parametri di processo -opzionale) e immetti il percorso Amazon S3 completo per i tuoi file `.zip` della libreria nella casella Python library path (Percorso libreria Python). Ad esempio:

```
s3://bucket/prefix/site-packages.zip
```

Se lo desideri, puoi specificare più percorsi completi per i file, separandoli con virgole ma non spazi, in questo modo:

```
s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/lib_X.zip
```

Se aggiorni questi file `.zip` in un secondo momento, puoi utilizzare la console per importarli nuovamente nell'endpoint di sviluppo. Individua l'endpoint dello sviluppatore in questione, verifica la casella a esso corrispondente e scegli Update ETL libraries (Aggiorna librerie ETL) dal menu Action (Operazione).

Analogamente, puoi specificare i file della libreria usando le API AWS Glue. Quando crei un endpoint di sviluppo chiamando Operazione CreateDevEndpoint (Python: create_dev_endpoint), puoi specificare uno o più percorsi completi per le librerie nel parametro `ExtraPythonLibsS3Path` in una chiamata come la seguente:

```
dep = glue.create_dev_endpoint(
            EndpointName="testDevEndpoint",
            RoleArn="arn:aws:iam::123456789012",
            SecurityGroupIds="sg-7f5ad1ff",
```

```
            SubnetId="subnet-c12fdba4",
            PublicKey="ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCtp04H/y...",
            NumberOfNodes=3,
            ExtraPythonLibsS3Path="s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/
lib_X.zip")
```

Quando aggiorni un endpoint di sviluppo, puoi anche aggiornare le librerie caricate utilizzando un oggetto DevEndpointCustomLibraries e impostare il parametro `UpdateEtlLibraries` su `True` durante la chiamata UpdateDevEndpoint (update_dev_endpoint).

Usare le librerie Python in un lavoro o JobRun

Quando crei un nuovo processo nella console, puoi specificare uno o più file ZIP di libreria scegliendo Script Libraries and job parameters (optional) (Librerie di script e parametri di processo - opzionale) e immettendo percorsi di librerie Amazon S3 completi, analogamente a come faresti quando crei un endpoint di sviluppo:

```
s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/lib_X.zip
```

Se stai chiamando CreateJob (create_job), puoi specificare uno o più percorsi completi alle librerie predefinite utilizzando il parametro predefinito `--extra-py-files`, come segue:

```
job = glue.create_job(Name='sampleJob',
                      Role='Glue_DefaultRole',
                      Command={'Name': 'glueetl',
                               'ScriptLocation': 's3://my_script_bucket/scripts/
my_etl_script.py'},
                      DefaultArguments={'--extra-py-files': 's3://bucket/prefix/
lib_A.zip,s3://bucket_B/prefix/lib_X.zip'})
```

Quindi, quando avvii un JobRun, puoi sovrascrivere l'impostazione della libreria predefinita con una diversa:

```
runId = glue.start_job_run(JobName='sampleJob',
                           Arguments={'--extra-py-files': 's3://bucket/prefix/
lib_B.zip'})
```

Esempi di codice Python in AWS Glue

- Esempio di codice: unione e relazioni dei dati

- Esempio di codice: preparazione dei dati utilizzando ResolveChoice, Lambda e ApplyMapping

Esempio di codice: unione e relazioni dei dati

In questo esempio viene usato un set di dati scaricato da http://everypolitician.org/ nel bucket `sample-dataset` in Amazon Simple Storage Service (Amazon S3): `s3://awsglue-datasets/examples/us-legislators/all`. Il set di dati contiene i dati in formato JSON sui legislatori degli Stati Uniti e sui seggi che hanno occupato nella Camera dei rappresentanti e al Senato che sono stati modificati leggermente e resi disponibili in un bucket Amazon S3 pubblico a fini di questo tutorial.

Puoi trovare il codice sorgente di questo esempio nel file `join_and_relationalize.py` presente nel repository degli esempi di AWS Glue sul sito Web di GitHub.

L'esercitazione illustra con questi dati come:

- Usa un crawler AWS Glue per classificare gli oggetti archiviati in un bucket Amazon S3 pubblico e salvare i relativi schemi nel catalogo dati di AWS Glue.
- Esaminare gli schemi e i metadati della tabella restituiti dal crawling.
- Scrivere uno script di estrazione, trasferimento e caricamento (ETL) Python che usa i metadati del catalogo dati per:
  - Unire insieme i dati dei diversi file di origine in un'unica tabella di dati (ovvero denormalizzare i dati).
  - Filtrare la tabella unita in tabelle separate in base al tipo di legislatore.
  - Scrivere i dati risultanti per separare i file di Apache Parquet per analisi successive.

Il modo più indicato per eseguire il debug degli script Python o PySpark durante l'esecuzione su AWS consiste nell'utilizzare Notebook su AWS Glue Studio.

Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3

1. Accedi alla AWS Management Console e apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Seguendo le fasi in Uso di crawler nella console AWS Glue, crea un nuovo crawler che esegue il crawling del set di dati `s3://awsglue-datasets/examples/us-legislators/all` in un database denominato `legislators` nel in AWS Glue Data Catalog. I dati di esempio sono già in questo bucket Amazon S3 pubblico.
3. Esegui il nuovo crawler e controlla il database `legislators`.

Il crawler crea le seguenti tabelle di metadati:

- `persons_json`
- `memberships_json`
- `organizations_json`
- `events_json`
- `areas_json`
- `countries_r_json`

Si tratta di una raccolta di tabelle semi-normalizzata contenenti i legislatori e le relative storie.

Fase 2: aggiunta dello script Boilerplate al notebook degli endpoint di sviluppo

Incolla lo script boilerplate seguente nel notebook degli endpoint di sviluppo per importare le librerie AWS Glue necessarie e configurare un singolo oggetto `GlueContext`:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

glueContext = GlueContext(SparkContext.getOrCreate())
```

Fase 3: esame degli schemi dai dati nel catalogo dati

Successivamente, è possibile creare facilmente un DynamicFrame da AWS Glue Data Catalog ed esaminare gli schemi dei dati. Ad esempio, per visualizzare lo schema della tabella `persons_json`, aggiungi quanto segue nel notebook:

```
persons = glueContext.create_dynamic_frame.from_catalog(
             database="legislators",
             table_name="persons_json")
print "Count: ", persons.count()
persons.printSchema()
```

Ecco l'output dalle chiamate di stampa:

```
Count:  1961
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

Ogni persona nella tabella è membro di alcuni enti del Congresso degli Stati Uniti.

Per visualizzare lo schema della tabella `memberships_json`, digita quando segue:

```
memberships = glueContext.create_dynamic_frame.from_catalog(
                database="legislators",
                table_name="memberships_json")
```

```
print "Count: ", memberships.count()
memberships.printSchema()
```

L'output è il seguente:

```
Count:  10439
root
|-- area_id: string
|-- on_behalf_of_id: string
|-- organization_id: string
|-- role: string
|-- person_id: string
|-- legislative_period_id: string
|-- start_date: string
|-- end_date: string
```

Gli elementi `organizations` sono i partiti e le due camere del Congresso, il Senato e la Camera dei rappresentanti. Per visualizzare lo schema della tabella `organizations_json`, digita quando segue:

```
orgs = glueContext.create_dynamic_frame.from_catalog(
           database="legislators",
           table_name="organizations_json")
print "Count: ", orgs.count()
orgs.printSchema()
```

L'output è il seguente:

```
Count:  13
root
|-- classification: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
```

```
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- name: string
|-- seats: int
|-- type: string
```

Fase 4: filtrare i dati

A questo punto mantieni solo i campi che desideri e rinomina `id` in `org_id`. Il set di dati è sufficientemente piccolo da poterlo visualizzare tutto insieme.

L'elemento `toDF()` converte un oggetto `DynamicFrame` in un elemento `DataFrame` di Apache Spark in modo da poter applicare le trasformazioni già esistenti in Apache Spark SQL:

```
orgs = orgs.drop_fields(['other_names',
                         'identifiers']).rename_field(
                             'id', 'org_id').rename_field(
                                'name', 'org_name')
orgs.toDF().show()
```

Di seguito è riportato l'output:

```
+--------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
|classification|              org_id|            org_name|               links|seats|
     type|               image|
+--------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
|         party|            party/al|                  AL|                null| null|
     null|                null|
|         party|      party/democrat|            Democrat|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/democrat-li...|    Democrat-Liberal|[[website,http://...| null|
     null|                null|
|   legislature|d56acebe-8fdc-47b...|House of Represen...|                null|  435|
lower house|                null|
```

```
|         party|   party/independent|         Independent|                null| null|
     null|                null|
|         party|party/new_progres...|     New Progressive|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/popular_dem...|    Popular Democrat|[[website,http://...| null|
     null|                null|
|         party|    party/republican|          Republican|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/republican-...|Republican-Conser...|[[website,http://...| null|
     null|                null|
|         party|      party/democrat|            Democrat|[[website,http://...| null|
     null|https://upload.wi...|
|         party|   party/independent|         Independent|                null| null|
     null|                null|
|         party|    party/republican|          Republican|[[website,http://...| null|
     null|https://upload.wi...|
|   legislature|8fa6c3d2-71dc-478...|              Senate|                null|  100|
upper house|                null|
+--------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
```

Digita quanto segue per visualizzare gli elementi `organizations` presenti nell'oggetto `memberships`:

```
memberships.select_fields(['organization_id']).toDF().distinct().show()
```

Di seguito è riportato l'output:

```
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

Fase 5: unione dei dati

Usa quindi AWS Glue per unire queste tabelle relazionali e creare una tabella con la cronologia completa per l'oggetto `memberships` dei legislatori e i relativi elementi `organizations`.

1. In primo luogo, unisci `persons` e `memberships` in `id` e `person_id`.
2. Quindi, unisci il risultato a `orgs` in `org_id` e `organization_id`.
3. Quindi, rilascia i campi ridondanti, `person_id` e `org_id`.

Puoi eseguire tutte queste operazioni in una sola riga di codice estesa:

```
l_history = Join.apply(orgs,
                       Join.apply(persons, memberships, 'id', 'person_id'),
                       'org_id', 'organization_id').drop_fields(['person_id',
 'org_id'])
print "Count: ", l_history.count()
l_history.printSchema()
```

L'output è il seguente:

```
Count:  10439
root
|-- role: string
|-- seats: int
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- type: string
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
```

```
|    |    |-- value: string
|-- name: string
|-- birth_date: string
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- death_date: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- family_name: string
|-- id: string
|-- start_date: string
|-- end_date: string
```

Ora hai la tabella finale che puoi utilizzare per l'analisi. Puoi scrivere in un formato compatto ed efficiente per le analisi dei dati, vale a dire Parquet, che puoi usare per eseguire SQL in AWS Glue, Amazon Athena o Amazon Redshift Spectrum.

La seguente chiamata scrive la tabella in più file per supportare le operazioni di lettura parallela veloce nella fase di analisi successiva:

```
glueContext.write_dynamic_frame.from_options(frame = l_history,
          connection_type = "s3",
          connection_options = {"path": "s3://glue-sample-target/output-dir/
legislator_history"},
          format = "parquet")
```

Per inserire tutti i dati cronologici in un singolo file, devi convertirli in un frame di dati, suddividerlo in partizioni e scriverlo:

```
s_history = l_history.toDF().repartition(1)
s_history.write.parquet('s3://glue-sample-target/output-dir/legislator_single')
```

In alternativa, se vuoi separarlo dal Senato e dalla Camera:

```
l_history.toDF().write.parquet('s3://glue-sample-target/output-dir/legislator_part',
                               partitionBy=['org_name'])
```

Fase 6: trasformare i dati per i database relazionali

AWS Glue permette di scrivere in modo semplice i dati, anche semistrutturati, in database relazionali come Amazon Redshift,. Offre una trasformazione di tipo `relationalize`, che appiattisce gli elementi `DynamicFrames` indipendentemente dalla complessità degli oggetti in frame.

Utilizzando `l_history` `DynamicFrame` in questo esempio, passi il nome di una tabella radice (`hist_root`) e un percorso temporaneo a `relationalize`. Viene restituito un elemento `DynamicFrameCollection`. Puoi quindi elencare i nomi degli elementi `DynamicFrames` nella raccolta:

```
dfc = l_history.relationalize("hist_root", "s3://glue-sample-target/temp-dir/")
dfc.keys()
```

Di seguito è riportato l'output della chiamata `keys`:

```
[u'hist_root', u'hist_root_contact_details', u'hist_root_links',
 u'hist_root_other_names', u'hist_root_images', u'hist_root_identifiers']
```

`Relationalize` suddivide la tabella della cronologia in sei nuove tabelle: una tabella radice che contiene un record per ogni oggetto dell'elemento `DynamicFrame` e le tabelle ausiliarie per le matrici. La gestione delle matrici nei database relazionali spesso non è ottimali, soprattutto quando le matrici diventano grandi. Separando le matrici in tabelle diverse velocizza l'esecuzione delle query.

A questo punto, controlla la separazione esaminando `contact_details`:

```
l_history.select_fields('contact_details').printSchema()
dfc.select('hist_root_contact_details').toDF().where("id = 10 or id =
 75").orderBy(['id','index']).show()
```

Di seguito è riportato l'output della chiamata `show`:

```
root
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
| 10|    0|                    fax|                       |
| 10|    1|                       |           202-225-1314|
| 10|    2|                  phone|                       |
| 10|    3|                       |           202-225-3772|
| 10|    4|                twitter|                       |
| 10|    5|                       |         MikeRossUpdates|
| 75|    0|                    fax|                       |
| 75|    1|                       |           202-225-7856|
| 75|    2|                  phone|                       |
| 75|    3|                       |           202-225-2711|
| 75|    4|                twitter|                       |
| 75|    5|                       |              SenCapito|
+---+-----+-----------------------+-----------------------+
```

Il campo `contact_details` era una matrice di strutture nell'elemento `DynamicFrame` originale. Ogni elemento di tali matrici è una riga separata nella tabella ausiliaria, indicizzata da `index`. L'`id` qui è una chiave esterna nella tabella `hist_root` con la chiave `contact_details`:

```
dfc.select('hist_root').toDF().where(
    "contact_details = 10 or contact_details = 75").select(
       ['id', 'given_name', 'family_name', 'contact_details']).show()
```

Di seguito è riportato l'output:

```
+--------------------+----------+-----------+---------------+
|                  id|given_name|family_name|contact_details|
+--------------------+----------+-----------+---------------+
|f4fc30ee-7b42-432...|      Mike|       Ross|             10|
|e3c60f34-7d1b-4c0...|   Shelley|     Capito|             75|
+--------------------+----------+-----------+---------------+
```

In questi comandi vengono utilizzati `toDF()` e un'espressione `where` per filtrare le righe che vuoi vedere.

Quindi, unendo la tabella `hist_root` con le tabelle ausiliarie ti consente di effettuare le operazioni descritte di seguito.

- Caricare i dati nei database senza il supporto di matrici.
- Eseguire la query di ogni singolo elemento in una matrice con SQL.

Archivia e accedi in modo sicuro alle tue credenziali Amazon Redshift con una connessione AWS Glue. Per informazioni su come creare la tua connessione, vedi Connessione ai dati.

Siete ora pronti a scrivere i vostri dati su una connessione, scorrendo uno alla volta i `DynamicFrames`:

```
for df_name in dfc.keys():
  m_df = dfc.select(df_name)
  print "Writing to table: ", df_name
  glueContext.write_dynamic_frame.from_jdbc_conf(frame = m_df, connection settings
 here)
```

Le impostazioni di connessione variano in base al tipo di database relazionale:

- Per istruzioni su come scrivere su Amazon Redshift, consultare the section called “Connessioni Redshift”.
- Per altri database, consultare Tipi e opzioni di connessione per ETL in AWS Glue per Spark.

Conclusioni

Complessivamente, AWS Glue è molto flessibile. Con poche righe di codice, ti consente di eseguire ciò che normalmente ti potrebbe richiedere giorni di scrittura. Puoi trovare gli interi script ETL origine-destinazione nel file Python `join_and_relationalize.py` negli esempi di AWS Glue su GitHub.

Esempio di codice: preparazione dei dati utilizzando ResolveChoice, Lambda e ApplyMapping

Il set di dati utilizzati in questo esempio è costituito dai dati del pagamento del provider Medicare scaricati da due set di dati Data.CMS.gov: "Inpatient Prospective Payment System Provider Summary for the Top 100 Diagnosis-Related Groups - FY2011" (Riepilogo Provider sistema di pagamenti

prospettici ospedalieri per i primi 100 gruppi correlati alla diagnosi - FY2011) e "Inpatient Charge Data FY 2011" (Dati di fatturazione ospedaliera FY 2011). Dopo aver scaricato i dati, abbiamo apportato delle modifiche al set di dati al fine di introdurre alcuni record errati nella parte finale del file. Questo file modificato si trova in un bucket pubblico Amazon S3 in `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv`.

Puoi trovare il codice sorgente di questo esempio nel `data_cleaning_and_lambda.py` file dell' GitHub archivio degli AWS Glueesempi.

Il modo preferito per eseguire il debug di Python PySpark o degli script durante l'esecuzione consiste nell'utilizzare AWS Notebooks su Glue Studio. AWS

Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Dopo il processo descritto in Uso di crawler nella console AWS Glue, crea un nuovo crawler in grado di eseguire il crawling del file `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv` e posizionare i metadati risultanti in un database denominato `payments` in AWS Glue Data Catalog.
3. Esegui il nuovo crawler e controlla il database `payments`. Il crawler dovrebbe aver creato una tabella di metadati denominata `medicare` nel database dopo aver letto l'inizio del file per determinarne il formato e il delimitatore.

   Lo schema della nuova tabella `medicare` è il seguente:

```
Column  name                                Data type
================================================
drg definition                                 string
provider id                                    bigint
provider name                                  string
provider street address                        string
provider city                                  string
provider state                                 string
provider zip code                              bigint
hospital referral region description           string
total discharges                               bigint
average covered charges                        string
average total payments                         string
average medicare payments                      string
```

Fase 2: aggiunta dello script boilerplate al notebook degli endpoint di sviluppo

Incolla lo script boilerplate seguente nel notebook degli endpoint di sviluppo per importare le librerie AWS Glue necessarie e configurare un singolo oggetto `GlueContext`:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

glueContext = GlueContext(SparkContext.getOrCreate())
```

Fase 3: confronta differenti analisi di schema

Puoi quindi verificare se lo schema riconosciuto da un oggetto `DataFrame` Apache Spark è uguale a quello registrato dal crawler AWS Glue. Esegui questo codice:

```
medicare = spark.read.format(
   "com.databricks.spark.csv").option(
   "header", "true").option(
   "inferSchema", "true").load(
   's3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv')
medicare.printSchema()
```

Ecco l'output dalla chiamata `printSchema`:

```
root
 |-- DRG Definition: string (nullable = true)
 |-- Provider Id: string (nullable = true)
 |-- Provider Name: string (nullable = true)
 |-- Provider Street Address: string (nullable = true)
 |-- Provider City: string (nullable = true)
 |-- Provider State: string (nullable = true)
 |-- Provider Zip Code: integer (nullable = true)
 |-- Hospital Referral Region Description: string (nullable = true)
 |--  Total Discharges : integer (nullable = true)
 |--  Average Covered Charges : string (nullable = true)
 |--  Average Total Payments : string (nullable = true)
```

```
 |-- Average Medicare Payments: string (nullable = true)
```

Controlla quindi lo schema generato da un oggetto AWS Glue `DynamicFrame`:

```
medicare_dynamicframe = glueContext.create_dynamic_frame.from_catalog(
       database = "payments",
       table_name = "medicare")
medicare_dynamicframe.printSchema()
```

L'output `printSchema` è il seguente:

```
root
 |-- drg definition: string
 |-- provider id: choice
 |    |-- long
 |    |-- string
 |-- provider name: string
 |-- provider street address: string
 |-- provider city: string
 |-- provider state: string
 |-- provider zip code: long
 |-- hospital referral region description: string
 |-- total discharges: long
 |-- average covered charges: string
 |-- average total payments: string
 |-- average medicare payments: string
```

Il `DynamicFrame` genera uno schema in cui `provider id` potrebbe essere un tipo `long` o `string`. Lo schema `DataFrame` elenca `Provider Id` come tipo `string` e il catalogo dati elenca `provider id` come tipo `bigint`.

Qual è corretto? Sono disponibili due record alla fine del file (su 160.000 record) con i valori `string` nella colonna. Questi sono i record errati che sono stati introdotti per illustrare un problema.

Per risolvere questo problema, l'oggetto AWS Glue `DynamicFrame` introduce il concetto di tipo choice. In questo caso, `DynamicFrame` mostra che entrambi i valori `long` e `string` possono essere visualizzati nella colonna. Il crawler AWS Glue ha saltato i valori `string` perché ha considerato solo un prefisso di 2 MB di dati. L'Apache Spark `DataFrame` ha considerato l'intero set di dati, ma è stato

costretto ad assegnare il tipo più generale alla colonna, ossia `string`. Infatti, Spark spesso ricorre al caso più generale quando non ci sono tipi complessi o variazioni con cui non è familiare.

Per eseguire una query sulla colonna `provider id`, risolvi prima il tipo di scelta. Puoi utilizzare il metodo di trasformazione `resolveChoice` in `DynamicFrame` per convertire quei valori `string` in valori `long` con un'opzione `cast:long`:

```
medicare_res = medicare_dynamicframe.resolveChoice(specs = [('provider
 id','cast:long')])
medicare_res.printSchema()
```

L'output `printSchema` è ora:

```
root
 |-- drg definition: string
 |-- provider id: long
 |-- provider name: string
 |-- provider street address: string
 |-- provider city: string
 |-- provider state: string
 |-- provider zip code: long
 |-- hospital referral region description: string
 |-- total discharges: long
 |-- average covered charges: string
 |-- average total payments: string
 |-- average medicare payments: string
```

Nel caso di un valore `string` di cui non è stato possibile eseguire il cast, AWS Glue ha inserito `null`.

Un'altra opzione consiste nel convertire il tipo di scelta in `struct`, che mantiene i valori di entrambi i tipi.

Quindi, esaminare le righe anomale:

```
medicare_res.toDF().where("'provider id' is NULL").show()
```

Verrà visualizzato quanto segue:

```
+--------------------+----------+--------------+----------------------+------------
+-------------+----------------+-----------------------------------+---------------
+----------------------+---------------------+------------------------+
|      drg definition|provider id|  provider name|provider street address|provider
 city|provider state|provider zip code|hospital referral region description|total
 discharges|average covered charges|average total payments|average medicare payments|
+--------------------+----------+--------------+----------------------+------------
+-------------+----------------+-----------------------------------+---------------
+----------------------+---------------------+------------------------+
|948 - SIGNS & SYM...|       null|            INC|       1050 DIVISION ST|
 MAUSTON|            WI|            53948|                        WI - Madison|
      12|              $11961.41|              $4619.00|                $3775.33|
|948 - SIGNS & SYM...|       null| INC- ST JOSEPH|     5000 W CHAMBERS ST|
 MILWAUKEE|            WI|            53210|                        WI - Milwaukee|
        14|              $10514.28|              $5562.50|                 $4522.78|
+--------------------+----------+--------------+----------------------+------------
+-------------+----------------+-----------------------------------+---------------
+----------------------+---------------------+------------------------+
```

Ora rimuovi i due record difettosi, come segue:

```
medicare_dataframe = medicare_res.toDF()
medicare_dataframe = medicare_dataframe.where("'provider id' is NOT NULL")
```

Fase 4: mappatura dei dati e utilizzo di funzioni Lambda Apache Spark

AWS Glue non supporta ancora direttamente le funzioni Lambda, note anche come funzioni definite dall'utente. Tuttavia, puoi sempre convertire un `DynamicFrame` in e da un `DataFrame` Apache Spark per trarre vantaggio dalle funzionalità Spark, oltre alle funzionalità speciali di `DynamicFrames`.

Trasforma quindi i dati di pagamento in numeri, in modo che i motori di analisi come Amazon Redshift o Amazon Athena possano eseguire i calcoli più rapidamente:

```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

chop_f = udf(lambda x: x[1:], StringType())
medicare_dataframe = medicare_dataframe.withColumn(
        "ACC", chop_f(
            medicare_dataframe["average covered charges"])).withColumn(
                "ATP", chop_f(
```

```
                    medicare_dataframe["average total payments"])).withColumn(
                        "AMP", chop_f(
                            medicare_dataframe["average medicare payments"]))
medicare_dataframe.select(['ACC', 'ATP', 'AMP']).show()
```

L'output dalla chiamata show è:

```
+--------+-------+-------+
|     ACC|    ATP|    AMP|
+--------+-------+-------+
|32963.07|5777.24|4763.73|
|15131.85|5787.57|4976.71|
|37560.37|5434.95|4453.79|
|13998.28|5417.56|4129.16|
|31633.27|5658.33|4851.44|
|16920.79|6653.80|5374.14|
|11977.13|5834.74|4761.41|
|35841.09|8031.12|5858.50|
|28523.39|6113.38|5228.40|
|75233.38|5541.05|4386.94|
|67327.92|5461.57|4493.57|
|39607.28|5356.28|4408.20|
|22862.23|5374.65|4186.02|
|31110.85|5366.23|4376.23|
|25411.33|5282.93|4383.73|
| 9234.51|5676.55|4509.11|
|15895.85|5930.11|3972.85|
|19721.16|6192.54|5179.38|
|10710.88|4968.00|3898.88|
|51343.75|5996.00|4962.45|
+--------+-------+-------+
only showing top 20 rows
```

Questi sono ancora tutte stringhe nei dati. Puoi utilizzare il potente metodo di trasformazione `apply_mapping` per eliminare, rinominare, trasmettere e nidificare i dati in modo che i dati di altri linguaggi di programmazione e sistemi possano accedere facilmente:

```
from awsglue.dynamicframe import DynamicFrame
medicare_tmp_dyf = DynamicFrame.fromDF(medicare_dataframe, glueContext, "nested")
medicare_nest_dyf = medicare_tmp_dyf.apply_mapping([('drg definition', 'string', 'drg',
 'string'),
                ('provider id', 'long', 'provider.id', 'long'),
```

```
                ('provider name', 'string', 'provider.name', 'string'),
                ('provider city', 'string', 'provider.city', 'string'),
                ('provider state', 'string', 'provider.state', 'string'),
                ('provider zip code', 'long', 'provider.zip', 'long'),
                ('hospital referral region description', 'string','rr', 'string'),
                ('ACC', 'string', 'charges.covered', 'double'),
                ('ATP', 'string', 'charges.total_pay', 'double'),
                ('AMP', 'string', 'charges.medicare_pay', 'double')])
medicare_nest_dyf.printSchema()
```

L'output `printSchema` è il seguente:

```
root
 |-- drg: string
 |-- provider: struct
 |    |-- id: long
 |    |-- name: string
 |    |-- city: string
 |    |-- state: string
 |    |-- zip: long
 |-- rr: string
 |-- charges: struct
 |    |-- covered: double
 |    |-- total_pay: double
 |    |-- medicare_pay: double
```

Trasformando i dati in un`DataFrame` Spark, puoi visualizzare quello che appare ora:

```
medicare_nest_dyf.toDF().show()
```

L'output è il seguente:

```
+--------------------+--------------------+--------------+--------------------+
|                 drg|            provider|            rr|             charges|
+--------------------+--------------------+--------------+--------------------+
|039 - EXTRACRANIA...|[10001,SOUTHEAST ...|    AL - Dothan|[32963.07,5777.24...|
|039 - EXTRACRANIA...|[10005,MARSHALL M...|AL - Birmingham|[15131.85,5787.57...|
|039 - EXTRACRANIA...|[10006,ELIZA COFF...|AL - Birmingham|[37560.37,5434.95...|
|039 - EXTRACRANIA...|[10011,ST VINCENT...|AL - Birmingham|[13998.28,5417.56...|
|039 - EXTRACRANIA...|[10016,SHELBY BAP...|AL - Birmingham|[31633.27,5658.33...|
|039 - EXTRACRANIA...|[10023,BAPTIST ME...|AL - Montgomery|[16920.79,6653.8,...|
|039 - EXTRACRANIA...|[10029,EAST ALABA...|AL - Birmingham|[11977.13,5834.74...|
```

```
|039 - EXTRACRANIA...|[10033,UNIVERSITY...|AL - Birmingham|[35841.09,8031.12...|
|039 - EXTRACRANIA...|[10039,HUNTSVILLE...|AL - Huntsville|[28523.39,6113.38...|
|039 - EXTRACRANIA...|[10040,GADSDEN RE...|AL - Birmingham|[75233.38,5541.05...|
|039 - EXTRACRANIA...|[10046,RIVERVIEW ...|AL - Birmingham|[67327.92,5461.57...|
|039 - EXTRACRANIA...|[10055,FLOWERS HO...|    AL - Dothan|[39607.28,5356.28...|
|039 - EXTRACRANIA...|[10056,ST VINCENT...|AL - Birmingham|[22862.23,5374.65...|
|039 - EXTRACRANIA...|[10078,NORTHEAST ...|AL - Birmingham|[31110.85,5366.23...|
|039 - EXTRACRANIA...|[10083,SOUTH BALD...|    AL - Mobile|[25411.33,5282.93...|
|039 - EXTRACRANIA...|[10085,DECATUR GE...|AL - Huntsville|[9234.51,5676.55,...|
|039 - EXTRACRANIA...|[10090,PROVIDENCE...|    AL - Mobile|[15895.85,5930.11...|
|039 - EXTRACRANIA...|[10092,D C H REGI...|AL - Tuscaloosa|[19721.16,6192.54...|
|039 - EXTRACRANIA...|[10100,THOMAS HOS...|    AL - Mobile|[10710.88,4968.0,...|
|039 - EXTRACRANIA...|[10103,BAPTIST ME...|AL - Birmingham|[51343.75,5996.0,...|
+--------------------+--------------------+---------------+--------------------+
only showing top 20 rows
```

Fase 5: scrittura dei dati in Apache Parquet

AWS Glue semplifica la scrittura dei dati in un formato come Apache Parquet, che può essere usato in modo efficiente dai database relazionali:

```
glueContext.write_dynamic_frame.from_options(
       frame = medicare_nest_dyf,
       connection_type = "s3",
       connection_options = {"path": "s3://glue-sample-target/output-dir/
medicare_parquet"},
       format = "parquet")
```

Informazioni di riferimento sulle estensioni PySpark AWS Glue

AWS Glue ha creato le estensioni seguenti per il dialetto PySpark Python.

- Accesso ai parametri utilizzando `getResolvedOptions`
- Tipi di estensione PySpark
- DynamicFrame classe
- Classe DynamicFrameCollection
- Classe DynamicFrameWriter
- DynamicFrameReader classe
- Classe GlueContext

Accesso ai parametri utilizzando **`getResolvedOptions`**

La funzione dell'utilità AWS Glue `getResolvedOptions(args, options)` ti permette di accedere agli argomenti passati allo script quando esegui un processo. Per usare questa funzione, esegui l'importazione dal modulo AWS Glue `utils` insieme al modulo `sys`:

```
import sys
from awsglue.utils import getResolvedOptions
```

`getResolvedOptions(args, options)`

- `args`: elenco degli argomenti contenuti in `sys.argv`.
- `options`: una matrice Python dei nomi degli argomenti da recuperare.

Example Recupero degli argomenti passati a un JobRun

Supponiamo che sia stato creato un oggetto JobRun in uno script, magari all'interno di una funzione Lambda:

```
response = client.start_job_run(
             JobName = 'my_test_Job',
             Arguments = {
               '--day_partition_key':   'partition_0',
               '--hour_partition_key':  'partition_1',
               '--day_partition_value':  day_partition_value,
               '--hour_partition_value': hour_partition_value } )
```

Per recuperare gli argomenti passati, puoi usare la funzione `getResolvedOptions` come segue:

```
import sys
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv,
                          ['JOB_NAME',
                           'day_partition_key',
                           'hour_partition_key',
                           'day_partition_value',
                           'hour_partition_value'])
print "The day-partition key is: ", args['day_partition_key']
print "and the day-partition value is: ", args['day_partition_value']
```

Si noti che gli argomenti vengono definiti con due trattini iniziali ma viene fatto riferimento a essi nello script senza i trattini Gli argomenti utilizzano solo trattini bassi, non trattini. I tuoi argomenti devono seguire questa convenzione per poter essere risolti.

Tipi di estensione PySpark

Tipi usati dalle estensioni PySpark AWS Glue.

DataType

Classe di base per gli altri tipi AWS Glue.

`__init__(properties={})`

- `properties`: proprietà del tipo di dati (opzionale).

`typeName(cls)`

Restituisce il tipo per la classe di tipo AWS Glue (ovvero il nome della classe senza "Type" nella parte finale).

- `cls`: un'istanza di classe AWS Glue derivata da `DataType`.

`jsonValue( )`

Restituisce un oggetto JSON contenente il tipo di dati e le proprietà della classe:

```
{
  "dataType": typeName,
  "properties": properties
}
```

AtomicType e derivate semplici

Eredita dalla classe DataType e la estende e funge da classe di base per tutti i tipi di dati atomici AWS Glue.

`fromJsonValue(cls, json_value)`

Inizializza un'istanza di classe con valori da un oggetto JSON.

- `cls`: un'istanza di classe di tipo AWS Glue da inizializzare.
- `json_value`: l'oggetto JSON dal quale caricare coppie chiave-valore.

I seguenti tipi sono derivate semplici della classe AtomicType:

- `BinaryType`: i dati binari.
- `BooleanType`: i valori booleani.
- `ByteType`: un valore di byte.
- `DateType`: un valore datetime.
- `DoubleType`: un valore doppio in virgola mobile.
- `IntegerType`: un valore intero.
- `LongType`: un valore intero lungo.
- `NullType`: un valore nullo.
- `ShortType`: un valore intero breve.
- `StringType`: una stringa di testo.
- `TimestampType`: un valore di timestamp (in genere in secondi dal 1/1/1970).
- `UnknownType`: un valore di tipo non identificato.

DecimalType(AtomicType)

Eredita la classe AtomicType e la estende per rappresentare un numero decimale (un numero espresso in cifre decimali, opposto ai numeri binari in base 2).

`__init__(precision=10, scale=2, properties={})`

- `precision`: il numero di cifre nel numero decimale (opzionale; il valore predefinito è 10).
- `scale`: il numero di cifre alla destra del punto decimale (opzionale; il valore predefinito è 2).
- `properties`: le proprietà del numero decimale (opzionale).

EnumType(AtomicType)

Eredita la classe AtomicType e la estende per rappresentare un'enumerazione delle opzioni valide.

`__init__(options)`

- `options`: un elenco delle opzioni enumerate.

Tipi di raccolta

- ArrayType(DataType)
- ChoiceType(DataType)
- MapType(DataType)
- Field(Object)
- StructType(DataType)
- EntityType(DataType)

ArrayType(DataType)

`__init__(elementType=UnknownType(), properties={})`

- `elementType`: il tipo di elementi nella matrice (opzionale; l'impostazione predefinita è UnknownType).
- `properties`: proprietà del tipo di matrice (opzionale).

ChoiceType(DataType)

`__init__(choices=[], properties={})`

- `choices`: un elenco di possibili scelte (opzionale).
- `properties`: proprietà di queste opzioni (opzionale).

`add(new_choice)`

Aggiunge una nuova opzione all'elenco di scelte possibili.

- `new_choice`: l'opzione da aggiungere all'elenco di scelte possibili.

merge(new_choices)

Unisce un elenco di nuove opzioni con quello esistente.

- `new_choices`: un elenco di nuove opzioni da unire con quelle esistenti.

MapType(DataType)

__init__(valueType=UnknownType, properties={})

- `valueType`: il tipo di valori nella mappa (opzionale; l'impostazione predefinita è UnknownType).
- `properties`: proprietà della mappa (opzionale).

Field(Object)

Consente di creare un oggetto campo al di fuori di un oggetto che deriva da DataType.

__init__(name, dataType, properties={})

- `name`: il nome da assegnare al campo.
- `dataType`: l'oggetto dal quale creare un campo.
- `properties`: proprietà del campo (opzionale).

StructType(DataType)

Definisce una struttura di dati (`struct`).

__init__(fields=[], properties={})

- `fields`: un elenco dei campi (di tipo `Field`) da includere nella struttura (opzionale).
- `properties`: proprietà della struttura (opzionale).

add(field)

- `field`: un oggetto di tipo `Field` da aggiungere alla struttura.

`hasField(field)`

Restituisce `True` se questa struttura ha un campo con lo stesso nome, altrimenti `False`.

- `field`: un nome campo o un oggetto di tipo `Field` di cui viene utilizzato il nome.

`getField(field)`

- `field`: un nome campo o un oggetto di tipo `Field` di cui viene utilizzato il nome. Se la struttura ha un campo con lo stesso nome, viene restituito.

EntityType(DataType)

`__init__(entity, base_type, properties)`

Questa classe non è ancora implementata.

Altri tipi

- DataSource(object)
- DataSink(object)

DataSource(object)

`__init__(j_source, sql_ctx, name)`

- `j_source`: l'origine dei dati.
- `sql_ctx`: il contesto SQL.
- `name`: il nome data-source.

`setFormat(format, **options)`

- `++format`: il formato da impostare per l'origine dei dati.
- `options`: un insieme di opzioni da impostare per l'origine dati. Per ulteriori informazioni sulle opzioni di formato, consulta la pagina the section called “Opzioni del formato dei dati”.

`getFrame()`

Restituisce un `DynamicFrame` per l'origine dati.

DataSink(object)

`__init__(j_sink, sql_ctx)`

- `j_sink`: il sink da creare.
- `sql_ctx`: il contesto SQL per il sink dei dati.

`setFormat(format, **options)`

- `format`: il formato da impostare per il sink dei dati.
- `options`: insieme di opzioni da impostare per il sink dei dati. Per ulteriori informazioni sulle opzioni di formato, consulta la pagina the section called “Opzioni del formato dei dati”.

`setAccumulableSize(size)`

- `size`: la dimensione accumulabile da impostare, in byte.

`writeFrame(dynamic_frame, info="")`

- `dynamic_frame`: il `DynamicFrame` da scrivere.
- `info`: informazioni sul `DynamicFrame` (opzionale).

`write(dynamic_frame_or_dfc, info="")`

Scrive un `DynamicFrame` o una `DynamicFrameCollection`.

- `dynamic_frame_or_dfc`: un oggetto `DynamicFrame` o un oggetto `DynamicFrameCollection` da scrivere.
- `info`: informazioni sulla `DynamicFrame` o `DynamicFrames` da scrivere (opzionale).

DynamicFrame classe

Una delle principali astrazioni in Apache Spark è SparkSQL `DataFrame`, che è simile al costrutto `DataFrame` di R e Pandas. Un oggetto `DataFrame` è simile a una tabella e supporta operazioni di tipo funzionale (mappatura, riduzione, filtro e così via) e operazioni SQL (selezione, proiezione, aggregazione).

I `DataFrames` sono potenti e ampiamente utilizzati, ma presentano delle limitazioni riguardo operazioni di estrazione, trasformazione e caricamento (ETL). Principalmente, richiedono che venga specificato uno schema prima di caricare qualsiasi dato. SparkSQL risolve il problema eseguendo due passaggi sui dati: il primo per dedurre lo schema e il secondo per caricare i dati. Tuttavia, l'inferenza è limitata e non gestisce i casi di dati non organizzati. Lo stesso campo, ad esempio, potrebbe essere di tipo diverso in record diversi. Apache Spark spesso si ferma e definisce il tipo come `string` utilizzando il testo del campo originale. Questo potrebbe non essere corretto e potrebbe essere richiesto un controllo più preciso sulle modalità di risoluzione delle discrepanze dello schema. Inoltre, per i set di dati di grandi dimensioni, un ulteriore passaggio sui dati di origine potrebbe essere proibitivo in termini di costi.

Per ovviare a queste limitazioni, AWS Glue introduce il`DynamicFrame`. Un `DynamicFrame` è simile a un `DataFrame`, con la differenza che ogni record è autodescrittivo, quindi inizialmente non è richiesto alcuno schema. Al contrario, AWS Glue calcola uno schema on-the-fly quando richiesto e codifica esplicitamente le incongruenze dello schema utilizzando un tipo di scelta (o unione). Puoi risolvere queste incongruenze per rendere i set di dati compatibili con i datastore che richiedono uno schema fisso.

Analogamente, un `DynamicRecord` rappresenta un record logico all'interno di un `DynamicFrame`. È come una riga in un `DataFrame` Spark, con la differenza che è autodescrittivo e può essere utilizzato per dati non conformi a uno schema fisso. Quando si utilizza AWS Glue with PySpark, in genere non si manipola in modo indipendente`DynamicRecords`. Viceversa, di solito si trasforma il set di dati nel suo complesso attraverso il rispettivo `DynamicFrame`.

Dopo aver risolto eventuali incongruenze dello schema, puoi convertire `DynamicFrames` in e da `DataFrames`.

— construction —

- __init__
- fromDF
- toDF

__init__

`__init__(jdf, glue_ctx, name)`

- `jdf`: un riferimento al frame di dati nella JVM (Java Virtual Machine).
- `glue_ctx`: un oggetto Classe GlueContext.
- `name`: una stringa nome opzionale, vuota per impostazione predefinita.

fromDF

`fromDF(dataframe, glue_ctx, name)`

Converte un `DataFrame` in un `DynamicFrame` convertendo campi del `DataFrame` in campi del `DynamicRecord`. Restituisce il nuovo `DynamicFrame`.

Un `DynamicRecord` rappresenta un record logico all'interno di un `DynamicFrame`. È simile a una riga in un `DataFrame` Spark, con la differenza che è autodescrittivo e può essere utilizzato per dati non conformi a uno schema fisso.

Questa funzione prevede che le colonne con nomi duplicati nel `DataFrame` siano già state risolte.

- `dataframe`: il `DataFrame` Apache Spark SQL da convertire (obbligatorio).
- `glue_ctx`: l'oggetto Classe GlueContext che specifica il contesto di questa trasformazione (richiesto).
- `name`— Il nome del risultato `DynamicFrame` (opzionale a partire da AWS Glue 3.0).

toDF

`toDF(options)`

Converte un `DynamicFrame` in un `DataFrame` Apache Spark, convertendo `DynamicRecords` in campi di `DataFrame`. Restituisce il nuovo `DataFrame`.

Un `DynamicRecord` rappresenta un record logico all'interno di un `DynamicFrame`. È simile a una riga in un `DataFrame` Spark, con la differenza che è autodescrittivo e può essere utilizzato per dati non conformi a uno schema fisso.

- `options`: un elenco di opzioni. Se scegli il tipo di operazione `Project` e `Cast`, devi specificare il tipo di destinazione. Gli esempi includono quanto segue.

```
>>>toDF([ResolveOption("a.b.c", "KeepAsStruct")])
>>>toDF([ResolveOption("a.b.c", "Project", DoubleType())])
```

— information —

- count
- schema
- printSchema
- show
- repartition
- coalesce

count

`count( )`: restituisce il numero di righe nell'oggetto sottostante `DataFrame`.

schema

`schema( )`: restituisce lo schema di questo `DynamicFrame` oppure, se non è disponibile, lo schema del `DataFrame` sottostante.

Per ulteriori informazioni sui tipi di `DynamicFrame` che compongono questo schema, consulta la pagina the section called “Tipi”.

printSchema

`printSchema( )`: stampa lo schema dell'oggetto sottostante `DataFrame`.

show

`show(num_rows)`: stampa un numero di righe specificato dall'oggetto sottostante `DataFrame`.

repartition

`repartition(numPartitions)`: restituisce un nuovo oggetto `DynamicFrame` con partizioni `numPartitions`.

coalesce

`coalesce(numPartitions)` – Restituisce un nuovo oggetto `DynamicFrame` con partizioni `numPartitions`.

— transforms —

- apply_mapping
- drop_fields
- filter
- join
- map
- mergeDynamicFrame
- relationalize
- rename_field
- resolveChoice
- select_fields
- spigot
- split_fields
- split_rows
- unbox
- the section called “unione”
- unnest
- unnest_ddb_json
- write

apply_mapping

`apply_mapping(mappings, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Applica una mappatura dichiarativa a `DynamicFrame` e restituisce un nuovo `DynamicFrame` con tali mappature applicate ai campi specificati. I campi non specificati vengono omessi dal nuovo `DynamicFrame`.

- `mappings`: un elenco di tuple di mappatura (obbligatorio). Ognuna è costituito da: colonna di origine, tipo di origine, colonna di destinazione, tipo di destinazione.

  Se il nome della colonna di origine include un punto (`.`), esso deve essere racchiuso tra apici inversi (``). Ad esempio, per mappare `this.old.name` (stringa) a `thisNewName`, devi utilizzare la tupla seguente:

  ```
  ("`this.old.name`", "string", "thisNewName", "string")
  ```

- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usa apply_mapping per rinominare campi e modificare tipi di campo

L'esempio di codice seguente mostra come utilizzare il metodo `apply_mapping` per rinominare i campi selezionati e modificare tipi di campo.

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: unione e relazioni dei dati e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

```
# Example: Use apply_mapping to reshape source data into
# the desired column names and types as a new DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
```

```
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Select and rename fields, change field type
print("Schema for the persons_mapped DynamicFrame, created with apply_mapping:")
persons_mapped = persons.apply_mapping(
    [
        ("family_name", "String", "last_name", "String"),
        ("name", "String", "first_name", "String"),
        ("birth_date", "String", "date_of_birth", "Date"),
    ]
)
persons_mapped.printSchema()
```

Output

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
```

```
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the persons_mapped DynamicFrame, created with apply_mapping:
root
|-- last_name: string
|-- first_name: string
|-- date_of_birth: date
```

drop_fields

`drop_fields(paths, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Richiama la trasformazione Classe FlatMap per rimuovere campi da un `DynamicFrame`. Restituisce un nuovo `DynamicFrame` con i campi specificati rimossi.

- `paths`: un elenco di stringhe. Ognuna contiene il percorso completo di un nodo del campo da rimuovere. Puoi utilizzare la notazione a punti per specificare campi nidificati. Ad esempio, se il campo `first` è figlio del campo `name` nell'albero, specifica `"name.first"` per il percorso.

  Se il nome di un nodo di campo contiene un punto (`.`) letterale, è necessario racchiudere il nome tra apici inversi (`` ` ``).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: utilizza drop_fields per rimuovere campi da un **`DynamicFrame`**

Questo esempio di codice utilizza il metodo `drop_fields` per rimuovere i campi di primo livello e i campi nidificati selezionati da un `DynamicFrame`.

Set di dati di esempio

L'esempio utilizza il set di dati seguente rappresentato dalla tabella `EXAMPLE-FRIENDS-DATA` nel codice:

```
{"name": "Sally", "age": 23, "location": {"state": "WY", "county": "Fremont"},
 "friends": []}
{"name": "Varun", "age": 34, "location": {"state": "NE", "county": "Douglas"},
 "friends": [{"name": "Arjun", "age": 3}]}
{"name": "George", "age": 52, "location": {"state": "NY"}, "friends": [{"name":
 "Fred"}, {"name": "Amy", "age": 15}]}
{"name": "Haruki", "age": 21, "location": {"state": "AK", "county": "Denali"}}
{"name": "Sheila", "age": 63, "friends": [{"name": "Nancy", "age": 22}]}
```

Esempio di codice

```
# Example: Use drop_fields to remove top-level and nested fields from a DynamicFrame.
# Replace MY-EXAMPLE-DATABASE with your Glue Data Catalog database name.
# Replace EXAMPLE-FRIENDS-DATA with your table name.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame from Glue Data Catalog
glue_source_database = "MY-EXAMPLE-DATABASE"
glue_source_table = "EXAMPLE-FRIENDS-DATA"

friends = glueContext.create_dynamic_frame.from_catalog(
```

```
    database=glue_source_database, table_name=glue_source_table
)
print("Schema for friends DynamicFrame before calling drop_fields:")
friends.printSchema()

# Remove location.county, remove friends.age, remove age
friends = friends.drop_fields(paths=["age", "location.county", "friends.age"])
print("Schema for friends DynamicFrame after removing age, county, and friend age:")
friends.printSchema()
```

Output

```
Schema for friends DynamicFrame before calling drop_fields:
root
|-- name: string
|-- age: int
|-- location: struct
|    |-- state: string
|    |-- county: string
|-- friends: array
|    |-- element: struct
|    |    |-- name: string
|    |    |-- age: int

Schema for friends DynamicFrame after removing age, county, and friend age:
root
|-- name: string
|-- location: struct
|    |-- state: string
|-- friends: array
|    |-- element: struct
|    |    |-- name: string
```

filter

`filter(f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Restituisce un nuovo `DynamicFrame` contenente tutti i `DynamicRecords` nel `DynamicFrame` di input che soddisfano la funzione predicato specificata `f`.

- `f`: funzione predicato da applicare all'oggetto `DynamicFrame`. La funzione deve richiedere un `DynamicRecord` come argomento e restituire True se il `DynamicRecord` soddisfa i requisiti del filtro o False in caso contrario (obbligatorio).

  Un `DynamicRecord` rappresenta un record logico all'interno di un `DynamicFrame`. È simile a una riga in un `DataFrame` Spark, con la differenza che è autodescrittivo e può essere utilizzato per dati non conformi a uno schema fisso.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usa il filtro per ottenere una selezione filtrata di campi

In questo esempio viene utilizzato il metodo `filter` per creare un nuovo `DynamicFrame` che include una selezione filtrata di campi di un altro `DynamicFrame`.

Come il metodo `map`, `filter` assume una funzione come argomento che viene applicato a ogni record nel `DynamicFrame` originario. La funzione accetta un record come input e restituisce un valore booleano. Se il valore restituito è vero, il record viene incluso nel `DynamicFrame` risultante. Se è falso, il record viene escluso.

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: preparazione dei dati utilizzando ResolveChoice, Lambda e ApplyMapping e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

```
# Example: Use filter to create a new DynamicFrame
# with a filtered selection of records
```

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create DynamicFrame from Glue Data Catalog
medicare = glueContext.create_dynamic_frame.from_options(
    "s3",
    {
        "paths": [
            "s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv"
        ]
    },
    "csv",
    {"withHeader": True},
)

# Create filtered DynamicFrame with custom lambda
# to filter records by Provider State and Provider City
sac_or_mon = medicare.filter(
    f=lambda x: x["Provider State"] in ["CA", "AL"]
    and x["Provider City"] in ["SACRAMENTO", "MONTGOMERY"]
)

# Compare record counts
print("Unfiltered record count: ", medicare.count())
print("Filtered record count:  ", sac_or_mon.count())
```

Output

```
Unfiltered record count:  163065
Filtered record count:   564
```

join

`join(paths1, paths2, frame2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Esegue un equi join con un altro `DynamicFrame` e restituisce il `DynamicFrame` risultante.

- `paths1`: un elenco delle chiavi di questo frame da unire.

- `paths2`: un elenco delle chiavi dell'altro frame da unire.
- `frame2`: l'altro `DynamicFrame` da unire.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usa join per combinare **`DynamicFrames`**

Questo esempio utilizza il `join` metodo per eseguire un'unione su tre`DynamicFrames`. AWS Glue esegue l'unione in base alle chiavi di campo fornite. Il `DynamicFrame` risultante contiene le righe dei due frame originali in cui le chiavi specificate corrispondono.

Tieni presente che la trasformazione `join` mantiene intatti tutti i campi. Ciò significa che i campi specificati per la corrispondenza vengono visualizzati nel risultato DynamicFrame, anche se sono ridondanti e contengono le stesse chiavi. In questo esempio viene utilizzato `drop_fields` per rimuovere tali chiavi ridondanti dopo l'unione.

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: unione e relazioni dei dati e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

```
# Example: Use join to combine data from three DynamicFrames

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
```

```
glueContext = GlueContext(sc)

# Load DynamicFrames from Glue Data Catalog
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
memberships = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="memberships_json"
)
orgs = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="organizations_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()
print("Schema for the memberships DynamicFrame:")
memberships.printSchema()
print("Schema for the orgs DynamicFrame:")
orgs.printSchema()

# Join persons and memberships by ID
persons_memberships = persons.join(
    paths1=["id"], paths2=["person_id"], frame2=memberships
)

# Rename and drop fields from orgs
# to prevent field name collisions with persons_memberships
orgs = (
    orgs.drop_fields(["other_names", "identifiers"])
    .rename_field("id", "org_id")
    .rename_field("name", "org_name")
)

# Create final join of all three DynamicFrames
legislators_combined = orgs.join(
    paths1=["org_id"], paths2=["organization_id"], frame2=persons_memberships
).drop_fields(["person_id", "org_id"])

# Inspect the schema for the joined data
print("Schema for the new legislators_combined DynamicFrame:")
legislators_combined.printSchema()
```

Output

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the memberships DynamicFrame:
root
|-- area_id: string
|-- on_behalf_of_id: string
|-- organization_id: string
|-- role: string
|-- person_id: string
|-- legislative_period_id: string
|-- start_date: string
|-- end_date: string
```

```
Schema for the orgs DynamicFrame:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- classification: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string

Schema for the new legislators_combined DynamicFrame:
root
|-- role: string
|-- seats: int
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- type: string
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
```

```
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- name: string
|-- birth_date: string
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- start_date: string
|-- family_name: string
|-- id: string
|-- death_date: string
|-- end_date: string
```

map

`map(f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Restituisce un nuovo `DynamicFrame` ottenuto applicando la funzione di mappatura specificata a tutti i record nel `DynamicFrame` originale.

- `f`: funzione di mappatura da applicare a tutti i record nell'oggetto `DynamicFrame`. La funzione deve richiedere un `DynamicRecord` come argomento e restituire un nuovo `DynamicRecord` (obbligatorio).

  Un `DynamicRecord` rappresenta un record logico all'interno di un `DynamicFrame`. È simile a una riga in un `DataFrame` Apache Spark, con la differenza che è autodescrittivo e può essere utilizzato per dati non conformi a uno schema fisso.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (facoltativo).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.

- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

Esempio: utilizza la mappa per applicare una funzione a ogni record in un **`DynamicFrame`**

In questo esempio viene utilizzato il metodo `map` per applicare una funzione a ogni record di un `DynamicFrame`. Nello specifico, questo esempio applica una funzione denominata `MergeAddress` a ogni record per unire diversi campi indirizzo in un singolo tipo `struct`.

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: preparazione dei dati utilizzando ResolveChoice, Lambda e ApplyMapping e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

```
# Example: Use map to combine fields in all records
# of a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
medicare = glueContext.create_dynamic_frame.from_options(
        "s3",
        {"paths": ["s3://awsglue-datasets/examples/medicare/
Medicare_Hospital_Provider.csv"]},
        "csv",
        {"withHeader": True})
print("Schema for medicare DynamicFrame:")
medicare.printSchema()

# Define a function to supply to the map transform
# that merges address fields into a single field
def MergeAddress(rec):
  rec["Address"] = {}
  rec["Address"]["Street"] = rec["Provider Street Address"]
```

```
  rec["Address"]["City"] = rec["Provider City"]
  rec["Address"]["State"] = rec["Provider State"]
  rec["Address"]["Zip.Code"] = rec["Provider Zip Code"]
  rec["Address"]["Array"] = [rec["Provider Street Address"], rec["Provider City"],
 rec["Provider State"], rec["Provider Zip Code"]]
  del rec["Provider Street Address"]
  del rec["Provider City"]
  del rec["Provider State"]
  del rec["Provider Zip Code"]
  return rec


# Use map to apply MergeAddress to every record
mapped_medicare = medicare.map(f = MergeAddress)
print("Schema for mapped_medicare DynamicFrame:")
mapped_medicare.printSchema()
```

Output

```
Schema for medicare DynamicFrame:
root
|-- DRG Definition: string
|-- Provider Id: string
|-- Provider Name: string
|-- Provider Street Address: string
|-- Provider City: string
|-- Provider State: string
|-- Provider Zip Code: string
|-- Hospital Referral Region Description: string
|-- Total Discharges: string
|-- Average Covered Charges: string
|-- Average Total Payments: string
|-- Average Medicare Payments: string

Schema for mapped_medicare DynamicFrame:
root
|-- Average Total Payments: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
```

```
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

mergeDynamicFrame

`mergeDynamicFrame(stage_dynamic_frame, primary_keys, transformation_ctx = "", options = {}, info = "", stageThreshold = 0, totalThreshold = 0)`

Unisce questo `DynamicFrame` con un `DynamicFrame` temporaneo basato sulle chiavi primarie specificate per identificare i record. I record duplicati (record con le stesse chiavi primarie) non vengono deduplicati. Se non è presente alcun record corrispondente nel frame temporaneo, tutti i record (inclusi i duplicati) vengono mantenuti dall'origine. Se il frame di staging dispone di record corrispondenti, i suoi registri sovrascrivono i registri nell'origine in AWS Glue.

- `stage_dynamic_frame`: il `DynamicFrame` di gestione temporanea da unire.
- `primary_keys`: l'elenco dei campi chiave primaria per abbinare i record dall'origine e dai frame dinamici di gestione temporanei.
- `transformation_ctx`: una stringa univoca utilizzata per recuperare i metadati relativi alla trasformazione corrente (opzionale).
- `options`: una stringa di coppie nome-valore JSON che forniscono informazioni aggiuntive per questa trasformazione. Questo argomento non è attualmente utilizzato.
- `info`: un `String`. Qualsiasi stringa da associare agli errori in questa trasformazione.
- `stageThreshold`: un `Long`. Il numero di errori nella trasformazione specificata per cui l'elaborazione deve restituire un errore.
- `totalThreshold`: un `Long`. Il numero totale di errori fino a questa trasformazione inclusa per i quali l'elaborazione deve restituire un errore.

Questo metodo restituisce un nuovo `DynamicFrame` ottenuto unendo questo `DynamicFrame` con il `DynamicFrame` temporaneo.

Il `DynamicFrame` restituito contiene il record A in questi casi:

- Se A esiste sia nel frame di origine che nel frame temporaneo, viene restituito A nel frame temporaneo.
- Se A si trova nella tabella di origine e `A.primaryKeys` non si trova nel `stagingDynamicFrame`, A non viene aggiornato nella tabella temporanea.

Il frame di origine e il frame temporaneo non devono avere lo stesso schema.

Esempio: mergeDynamicFrame da utilizzare per unire due in **`DynamicFrames`** base a una chiave primaria

Il seguente esempio di codice mostra come utilizzare il metodo `mergeDynamicFrame` per unire un `DynamicFrame` con un `DynamicFrame` di staging, in base alla chiave primaria `id`.

Set di dati di esempio

L'esempio utilizza due `DynamicFrames` da una `DynamicFrameCollection` chiamata `split_rows_collection`. Di seguito è riportato un elenco di chiavi in `split_rows_collection`.

```
dict_keys(['high', 'low'])
```

Esempio di codice

```
# Example: Use mergeDynamicFrame to merge DynamicFrames
# based on a set of specified primary keys

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import SelectFromCollection

# Inspect the original DynamicFrames
frame_low = SelectFromCollection.apply(dfc=split_rows_collection, key="low")
print("Inspect the DynamicFrame that contains rows where ID < 10")
frame_low.toDF().show()

frame_high = SelectFromCollection.apply(dfc=split_rows_collection, key="high")
print("Inspect the DynamicFrame that contains rows where ID > 10")
frame_high.toDF().show()

# Merge the DynamicFrames based on the "id" primary key
merged_high_low = frame_high.mergeDynamicFrame(
```

```
    stage_dynamic_frame=frame_low, primary_keys=["id"]
)

# View the results where the ID is 1 or 20
print("Inspect the merged DynamicFrame that contains the combined rows")
merged_high_low.toDF().where("id = 1 or id= 20").orderBy("id").show()
```

Output

```
Inspect the DynamicFrame that contains rows where ID < 10
+---+-----+----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-----------------------+
|  1|    0|                   fax|           202-225-3307|
|  1|    1|                 phone|           202-225-5731|
|  2|    0|                   fax|           202-225-3307|
|  2|    1|                 phone|           202-225-5731|
|  3|    0|                   fax|           202-225-3307|
|  3|    1|                 phone|           202-225-5731|
|  4|    0|                   fax|           202-225-3307|
|  4|    1|                 phone|           202-225-5731|
|  5|    0|                   fax|           202-225-3307|
|  5|    1|                 phone|           202-225-5731|
|  6|    0|                   fax|           202-225-3307|
|  6|    1|                 phone|           202-225-5731|
|  7|    0|                   fax|           202-225-3307|
|  7|    1|                 phone|           202-225-5731|
|  8|    0|                   fax|           202-225-3307|
|  8|    1|                 phone|           202-225-5731|
|  9|    0|                   fax|           202-225-3307|
|  9|    1|                 phone|           202-225-5731|
| 10|    0|                   fax|           202-225-6328|
| 10|    1|                 phone|           202-225-4576|
+---+-----+----------------------+-----------------------+
only showing top 20 rows

Inspect the DynamicFrame that contains rows where ID > 10
+---+-----+----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-----------------------+
| 11|    0|                   fax|           202-225-6328|
| 11|    1|                 phone|           202-225-4576|
| 11|    2|               twitter|          RepTrentFranks|
```

```
| 12|    0|                    fax|             202-225-6328|
| 12|    1|                  phone|             202-225-4576|
| 12|    2|                twitter|           RepTrentFranks|
| 13|    0|                    fax|             202-225-6328|
| 13|    1|                  phone|             202-225-4576|
| 13|    2|                twitter|           RepTrentFranks|
| 14|    0|                    fax|             202-225-6328|
| 14|    1|                  phone|             202-225-4576|
| 14|    2|                twitter|           RepTrentFranks|
| 15|    0|                    fax|             202-225-6328|
| 15|    1|                  phone|             202-225-4576|
| 15|    2|                twitter|           RepTrentFranks|
| 16|    0|                    fax|             202-225-6328|
| 16|    1|                  phone|             202-225-4576|
| 16|    2|                twitter|           RepTrentFranks|
| 17|    0|                    fax|             202-225-6328|
| 17|    1|                  phone|             202-225-4576|
+---+-----+----------------------+-------------------------+
only showing top 20 rows

Inspect the merged DynamicFrame that contains the combined rows
+---+-----+----------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-------------------------+
|  1|    0|                    fax|             202-225-3307|
|  1|    1|                  phone|             202-225-5731|
| 20|    0|                    fax|             202-225-5604|
| 20|    1|                  phone|             202-225-6536|
| 20|    2|                twitter|                USRepLong|
+---+-----+----------------------+-------------------------+
```

relationalize

`relationalize(root_table_name, staging_path, options, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Converte un `DynamicFrame` in un modulo che si inserisce in un database relazionale. La relazionalizzazione di un `DynamicFrame` è particolarmente utile quando si desidera spostare dati da un ambiente NoSQL come DynamoDB a un database relazionale come MySQL.

La trasformazione genera un elenco di frame rimuovendo le colonne annidate e ruotando le colonne dell'array. La colonna matrice trasformata mediante pivot può essere unita alla tabella root utilizzando la join-key generata durante la fase di annullamento dell'annidamento.

- `root_table_name`: il nome della tabella root.
- `staging_path`: il percorso in cui il metodo può archiviare le partizioni di tabelle trasformate mediante pivot in formato CSV (facoltativo). Le tabelle trasformate mediante pivot vengono rilette da questo percorso.
- `options`: un dizionario dei parametri opzionali.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usa la relazionalizzazione per livellare uno schema annidato in un **`DynamicFrame`**

Questo esempio di codice utilizza il metodo `relationalize` per livellare uno schema annidato in una forma adatta a un database relazionale.

Set di dati di esempio

L'esempio utilizza un `DynamicFrame` chiamato `legislators_combined` con lo schema seguente. `legislators_combined` ha più campi annidati come `links`, `images`, `econtact_details`, che verranno livellati dalla trasformazione `relationalize`.

```
root
|-- role: string
|-- seats: int
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
```

```
|-- type: string
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- name: string
|-- birth_date: string
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- start_date: string
|-- family_name: string
|-- id: string
|-- death_date: string
|-- end_date: string
```

Esempio di codice

```
# Example: Use relationalize to flatten
# a nested schema into a format that fits
# into a relational database.
# Replace DOC-EXAMPLE-S3-BUCKET/tmpDir with your own location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext
```

```
# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Apply relationalize and inspect new tables
legislators_relationalized = legislators_combined.relationalize(
    "l_root", "s3://DOC-EXAMPLE-BUCKET/tmpDir"
)
legislators_relationalized.keys()

# Compare the schema of the contact_details
# nested field to the new relationalized table that
# represents it
legislators_combined.select_fields("contact_details").printSchema()
legislators_relationalized.select("l_root_contact_details").toDF().where(
    "id = 10 or id = 75"
).orderBy(["id", "index"]).show()
```

Output

L'output seguente consente di confrontare lo schema del campo annidato chiamato `contact_details` con la tabella creata dalla trasformazione `relationalize`. Si noti che i record della tabella rimandano alla tabella principale utilizzando una chiave esterna chiamata `id` e una colonna `index` che rappresenta le posizioni dell'array.

```
dict_keys(['l_root', 'l_root_images', 'l_root_links', 'l_root_other_names',
 'l_root_contact_details', 'l_root_identifiers'])

root
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string

+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
| 10|    0|                    fax|           202-225-4160|
| 10|    1|                  phone|           202-225-3436|
| 75|    0|                    fax|           202-225-6791|
| 75|    1|                  phone|           202-225-2861|
| 75|    2|                twitter|             RepSamFarr|
```

```
+---+-----+-----------------------+-----------------------+
```

rename_field

`rename_field(oldName, newName, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Rinomina un campo in questo `DynamicFrame` e restituisce un nuovo `DynamicFrame` con il campo rinominato.

- `oldName`: il percorso completo al nodo che desideri rinominare.

  Se il vecchio nome contiene dei punti, `RenameField` non funziona a meno che non venga racchiuso tra virgolette (`). Ad esempio, per sostituire `this.old.name` con `thisNewName`, chiamare rename_field come segue:

  ```
  newDyF = oldDyF.rename_field("`this.old.name`", "thisNewName")
  ```

- `newName`: il nuovo nome, come un percorso completo.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usa rename_field per rinominare i campi in un **`DynamicFrame`**

Questo esempio di codice utilizza il metodo `rename_field` per rinominare i campi in un `DynamicFrame`. Si noti che l'esempio utilizza il concatenamento di metodi per rinominare più campi contemporaneamente.

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: unione e relazioni dei dati e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

Esempio di codice

```
# Example: Use rename_field to rename fields
# in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Inspect the original orgs schema
orgs = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="organizations_json"
)
print("Original orgs schema: ")
orgs.printSchema()

# Rename fields and view the new schema
orgs = orgs.rename_field("id", "org_id").rename_field("name", "org_name")
print("New orgs schema with renamed fields: ")
orgs.printSchema()
```

Output

```
Original orgs schema:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
```

```
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- classification: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string

New orgs schema with renamed fields:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- classification: string
|-- org_id: string
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string
```

resolveChoice

`resolveChoice(specs = None, choice = "" , database = None , table_name = None , transformation_ctx="", info="", stageThreshold=0, totalThreshold=0, catalog_id = None)`

Risolve un tipo di scelta all'interno di questo `DynamicFrame` e restituisce il nuovo `DynamicFrame`.

- `specs`: elenco di ambiguità specifiche da risolvere, ognuna sotto forma di tupla: `(field_path, action)`.

  Ci sono due modi per utilizzare `resolveChoice`. Il primo consiste nell'indicare un argomento `specs` per specificare una sequenza di campi specifici e come risolverli. L'altra modalità per `resolveChoice` è usare un singolo argomento `choice` per specificare una singola risoluzione per tutti i `ChoiceTypes`.

  I valori per `specs` vengono specificati come tuple costituiti da coppie `(field_path, action)`. Il valore `field_path` identifica un elemento ambiguo specifico e il valore `action` identifica la soluzione corrispondente. Sono disponibili le operazioni seguenti:

  - `cast:`*`type`*: tenta di trasmettere tutti i valori al tipo specificato. Ad esempio: `cast:int`.
  - `make_cols`: converte ogni tipo distinto in colonna con il nome *`columnName_type`*. Risolve una potenziale ambiguità livellando i dati. Ad esempio, se `columnA` fosse un `int` o una `string`, la soluzione consisterebbe nel produrre due colonne denominate `columnA_int` e `columnA_string` nel `DynamicFrame` risultante.
  - `make_struct`: risolve una potenziale ambiguità utilizzando una `struct` per rappresentare i dati. Ad esempio, se i dati in una colonna sono un `int` o una `string`, con l'utilizzo dell'operazione `make_struct` viene prodotta una colonna di strutture nel risultante `DynamicFrame`. Ogni struttura contiene sia un `int` che un `string`.
  - `project:`*`type`*: risolve una potenziale ambiguità proiettando tutti i dati su uno dei tipi di dati possibili. Ad esempio, se i dati in una colonna sono un `int` o una `string`, utilizzando un'operazione `project:string` viene prodotta una colonna nel `DynamicFrame` risultante, dove tutti i valori `int` sono stati convertiti in stringhe.

  Se il `field_path` identifica un array, inserisci parentesi quadre vuote dopo il nome dell'array per evitare ambiguità. Ad esempio, supponiamo che tu stia lavorando con dati strutturati nel seguente modo:

```
"myList": [
  { "price": 100.00 },
  { "price": "$100.00" }
]
```

  Puoi selezionare la versione numerica invece di quella di stringa del prezzo impostando il `field_path` su `"myList[].price"` e la `action` su `"cast:double"`.

> **Note**
>
> Può essere utilizzato solo uno dei parametri `specs` e `choice`. Se il parametro `specs` non è `None`, allora il parametro `choice` deve essere una stringa vuota. Viceversa, se `choice` non è una stringa vuota, allora il parametro `specs` deve essere `None`.

- `choice`: specifica una singola risoluzione per tutti i `ChoiceTypes`. Puoi usare questa modalità nei casi in cui l'elenco completo di `ChoiceTypes` non è noto prima del runtime. Oltre alle operazioni elencate in precedenza per `specs`, questa modalità supporta anche l'operazione seguente:
  - `match_catalog`: tenta di trasmettere ogni `ChoiceType` al tipo corrispondente nella tabella del catalogo specificata.
- `database`: il database del catalogo dati da usare con l'operazione `match_catalog`.
- `table_name`: la tabella del catalogo dati da usare con l'operazione `match_catalog`.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero di errori riscontrati fino a questa trasformazione compresa, raggiunto il quale il processo dovrebbe interrompersi (opzionale: impostazione predefinita: zero, a indicare che il processo non dovrebbe interrompersi).
- `catalog_id`: l'ID catalogo del catalogo dati a cui si accede (l'ID account del catalogo dati). Se impostato su `None` (valore predefinito), utilizza l'ID catalogo dell'account chiamante.

Esempio: utilizzare resolveChoice per gestire una colonna che contiene più tipi

Questo esempio di codice utilizza il metodo `resolveChoice` per specificare come gestire una colonna `DynamicFrame` che contiene valori di più tipi. L'esempio mostra due modi comuni per gestire una colonna con tipi diversi:

- Trasforma la colonna in un singolo tipo di dati.
- Conserva tutti i tipi in colonne separate.

Set di dati di esempio

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: preparazione dei dati utilizzando ResolveChoice, Lambda e ApplyMapping e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

L'esempio utilizza un `DynamicFrame` chiamato `medicare` con il seguente schema:

```
root
|-- drg definition: string
|-- provider id: choice
|    |-- long
|    |-- string
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
|-- average total payments: string
|-- average medicare payments: string
```

Esempio di codice

```
# Example: Use resolveChoice to handle
# a column that contains multiple types

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load the input data and inspect the "provider id" column
medicare = glueContext.create_dynamic_frame.from_catalog(
    database="payments", table_name="medicare_hospital_provider_csv"
```

```
)
print("Inspect the provider id column:")
medicare.toDF().select("provider id").show()

# Cast provider id to type long
medicare_resolved_long = medicare.resolveChoice(specs=[("provider id", "cast:long")])
print("Schema after casting provider id to type long:")
medicare_resolved_long.printSchema()
medicare_resolved_long.toDF().select("provider id").show()

# Create separate columns
# for each provider id type
medicare_resolved_cols = medicare.resolveChoice(choice="make_cols")
print("Schema after creating separate columns for each type:")
medicare_resolved_cols.printSchema()
medicare_resolved_cols.toDF().select("provider id_long", "provider id_string").show()
```

Output

```
Inspect the 'provider id' column:
+-----------+
|provider id|
+-----------+
|   [10001,]|
|   [10005,]|
|   [10006,]|
|   [10011,]|
|   [10016,]|
|   [10023,]|
|   [10029,]|
|   [10033,]|
|   [10039,]|
|   [10040,]|
|   [10046,]|
|   [10055,]|
|   [10056,]|
|   [10078,]|
|   [10083,]|
|   [10085,]|
|   [10090,]|
|   [10092,]|
|   [10100,]|
|   [10103,]|
```

```
+-----------+
only showing top 20 rows

Schema after casting 'provider id' to type long:
root
|-- drg definition: string
|-- provider id: long
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
|-- average total payments: string
|-- average medicare payments: string

+-----------+
|provider id|
+-----------+
|      10001|
|      10005|
|      10006|
|      10011|
|      10016|
|      10023|
|      10029|
|      10033|
|      10039|
|      10040|
|      10046|
|      10055|
|      10056|
|      10078|
|      10083|
|      10085|
|      10090|
|      10092|
|      10100|
|      10103|
+-----------+
only showing top 20 rows
```

```
Schema after creating separate columns for each type:
root
|-- drg definition: string
|-- provider id_string: string
|-- provider id_long: long
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
|-- average total payments: string
|-- average medicare payments: string

+----------------+------------------+
|provider id_long|provider id_string|
+----------------+------------------+
|           10001|              null|
|           10005|              null|
|           10006|              null|
|           10011|              null|
|           10016|              null|
|           10023|              null|
|           10029|              null|
|           10033|              null|
|           10039|              null|
|           10040|              null|
|           10046|              null|
|           10055|              null|
|           10056|              null|
|           10078|              null|
|           10083|              null|
|           10085|              null|
|           10090|              null|
|           10092|              null|
|           10100|              null|
|           10103|              null|
+----------------+------------------+
only showing top 20 rows
```

select_fields

`select_fields(paths, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Restituisce un nuovo `DynamicFrame` contenente i campi selezionati.

- `paths`: un elenco di stringhe. Ogni stringa è un percorso completo di un nodo di livello superiore da selezionare.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usa select_fields per creare un nuovo **`DynamicFrame`** con i campi scelti

L'esempio di codice seguente mostra come utilizzare il metodo `select_fields` per creare un nuovo `DynamicFrame` con un elenco di campi scelto da un `DynamicFrame` esistente.

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: unione e relazioni dei dati e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

```
# Example: Use select_fields to select specific fields from a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
```

```
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Create a new DynamicFrame with chosen fields
names = persons.select_fields(paths=["family_name", "given_name"])
print("Schema for the names DynamicFrame, created with select_fields:")
names.printSchema()
names.toDF().show()
```

Output

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
```

```
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the names DynamicFrame:
root
|-- family_name: string
|-- given_name: string

+-----------+----------+
|family_name|given_name|
+-----------+----------+
|    Collins|   Michael|
|   Huizenga|      Bill|
|    Clawson|    Curtis|
|    Solomon|    Gerald|
|     Rigell|    Edward|
|      Crapo|   Michael|
|      Hutto|      Earl|
|      Ertel|     Allen|
|     Minish|    Joseph|
|    Andrews|    Robert|
|     Walden|      Greg|
|      Kazen|   Abraham|
|     Turner|   Michael|
|      Kolbe|     James|
|  Lowenthal|      Alan|
|    Capuano|   Michael|
|   Schrader|      Kurt|
|     Nadler|   Jerrold|
|     Graves|       Tom|
|   McMillan|      John|
+-----------+----------+
only showing top 20 rows
```

simply_ddb_json

`simplify_ddb_json(): DynamicFrame`

Semplifica le colonne annidate in un ambiente `DynamicFrame` che si trova specificamente nella struttura JSON di DynamoDB e ne restituisce una nuova semplificata. `DynamicFrame` Se ci sono

più tipi o tipi di mappa in un tipo di elenco, gli elementi nell'elenco non verranno semplificati. Tieni presente che si tratta di un tipo specifico di trasformazione che si comporta in modo diverso dalla `unnest` trasformazione normale e richiede che i dati siano già presenti nella struttura JSON di DynamoDB. Per ulteriori informazioni, consulta DynamoDB JSON.

Ad esempio, lo schema di lettura di un'esportazione con la struttura JSON DynamoDB potrebbe apparire come segue:

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

La trasformazione di `simplify_ddb_json()` lo convertirebbe in:

```
root
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
|    |    |-- packageName: string
|    |    |-- updatedAt: string
```

```
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
|-- nullValue: null
```

Esempio: utilizza simply_ddb_json per richiamare un DynamoDB JSON simple

Questo esempio di codice utilizza il `simplify_ddb_json` metodo per utilizzare il connettore di esportazione AWS Glue DynamoDB, richiamare un DynamoDB JSON simple e stampare il numero di partizioni.

Esempio di codice

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type = "dynamodb",
    connection_options = {
        'dynamodb.export': 'ddb',
        'dynamodb.tableArn': '<table arn>',
        'dynamodb.s3.bucket': '<bucket name>',
        'dynamodb.s3.prefix': '<bucket prefix>',
        'dynamodb.s3.bucketOwner': '<account_id of bucket>'
    }
)
simplified = dynamicFrame.simplify_ddb_json()
print(simplified.getNumPartitions())
```

spigot

`spigot(path, options={})`

Scrive record di esempio in una destinazione specificata per aiutarti a verificare le trasformazioni eseguite dal tuo lavoro.

- `path`: il percorso della destinazione in cui scrivere (obbligatorio).
- `options`: coppie chiave-valore che specificano opzioni (opzionale). L'opzione `"topk"` specifica che devono essere scritti i primi record k. L'opzione `"prob"` specifica la probabilità (sotto forma di valore decimale) di scelta di un dato record. Puoi usarlo per selezionare i record da scrivere.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).

Esempio: usa lo spigot per scrivere campi di esempio da **`DynamicFrame`** ad Amazon S3

Questo esempio di codice utilizza il metodo `spigot` per scrivere record di esempio in un bucket Amazon S3 dopo aver applicato la trasformazione `select_fields`.

Set di dati di esempio

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: unione e relazioni dei dati e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

L'esempio utilizza un `DynamicFrame` chiamato `persons` con il seguente schema:

```
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
```

```
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

Esempio di codice

```
# Example: Use spigot to write sample records
# to a destination during a transformation
# from pyspark.context import SparkContext.
# Replace DOC-EXAMPLE-BUCKET with your own location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load table data into a DynamicFrame
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)

# Perform the select_fields on the DynamicFrame
persons = persons.select_fields(paths=["family_name", "given_name", "birth_date"])

# Use spigot to write a sample of the transformed data
# (the first 10 records)
spigot_output = persons.spigot(
    path="s3://DOC-EXAMPLE-BUCKET", options={"topk": 10}
)
```

Output

Di seguito è riportato un esempio di dati che `spigot` scrive su Amazon S3. Poiché è stato specificato il codice di esempio `options={"topk": 10}`, i dati di esempio contengono i primi 10 record.

```
{"family_name":"Collins","given_name":"Michael","birth_date":"1944-10-15"}
{"family_name":"Huizenga","given_name":"Bill","birth_date":"1969-01-31"}
{"family_name":"Clawson","given_name":"Curtis","birth_date":"1959-09-28"}
{"family_name":"Solomon","given_name":"Gerald","birth_date":"1930-08-14"}
{"family_name":"Rigell","given_name":"Edward","birth_date":"1960-05-28"}
{"family_name":"Crapo","given_name":"Michael","birth_date":"1951-05-20"}
{"family_name":"Hutto","given_name":"Earl","birth_date":"1926-05-12"}
{"family_name":"Ertel","given_name":"Allen","birth_date":"1937-11-07"}
{"family_name":"Minish","given_name":"Joseph","birth_date":"1916-09-01"}
{"family_name":"Andrews","given_name":"Robert","birth_date":"1957-08-04"}
```

split_fields

`split_fields(paths, name1, name2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Restituisce un nuovo `DynamicFrameCollection` che ne contiene due `DynamicFrames`. Il primo `DynamicFrame` contiene tutti i nodi che sono stati separati e il secondo contiene i nodi rimanenti.

- `paths`: elenco di stringhe, ciascuna delle quali è un percorso completo di un nodo da separare in un nuovo oggetto `DynamicFrame`.
- `name1`: una stringa nome per il `DynamicFrame` separato.
- `name2`: una stringa nome per il `DynamicFrame` che rimane dopo aver separato i nodi specificati.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usare split_fields per dividere i campi selezionati in un campo separato **`DynamicFrame`**

Questo esempio di codice utilizza il metodo `split_fields` per dividere un elenco di campi specificati in un campo separato `DynamicFrame`.

Set di dati di esempio

L'esempio utilizza un `DynamicFrame` chiamato `l_root_contact_details` che proviene da una raccolta denominata `legislators_relationalized`.

`l_root_contact_details` ha il seguente schema e le seguenti voci.

```
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                  phone|             202-225-5265|
|  1|    1|                twitter|              kathyhochul|
|  2|    0|                  phone|             202-225-3252|
|  2|    1|                twitter|            repjackyrosen|
|  3|    0|                    fax|             202-225-1314|
|  3|    1|                  phone|             202-225-3772|
...
```

Esempio di codice

```
# Example: Use split_fields to split selected
# fields into a separate DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load the input DynamicFrame and inspect its schema
frame_to_split = legislators_relationalized.select("l_root_contact_details")
```

```
print("Inspect the input DynamicFrame schema:")
frame_to_split.printSchema()

# Split id and index fields into a separate DynamicFrame
split_fields_collection = frame_to_split.split_fields(["id", "index"], "left", "right")

# Inspect the resulting DynamicFrames
print("Inspect the schemas of the DynamicFrames created with split_fields:")
split_fields_collection.select("left").printSchema()
split_fields_collection.select("right").printSchema()
```

Output

```
Inspect the input DynamicFrame's schema:
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

Inspect the schemas of the DynamicFrames created with split_fields:
root
|-- id: long
|-- index: int

root
|-- contact_details.val.type: string
|-- contact_details.val.value: string
```

split_rows

`split_rows(comparison_dict, name1, name2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Suddivide una o più righe di un `DynamicFrame` in un nuovo `DynamicFrame`.

Il metodo restituisce un nuovo `DynamicFrameCollection` che contiene due `DynamicFrames`. Il primo `DynamicFrame` contiene tutti i nodi che sono stati separati e il secondo contiene i nodi rimanenti.

- `comparison_dict`: un dizionario in cui la chiave è un percorso verso una colonna e il valore è un altro dizionario per la mappatura di comparatori rispetto a valori con i quali vengono confrontati i

valori di colonna. Ad esempio, `{"age": {">": 10, "<": 20}}` divide tutte le righe il cui valore nella colonna età è superiore a 10 e inferiore a 20.

- `name1`: una stringa nome per il `DynamicFrame` separato.
- `name2`: una stringa nome per il `DynamicFrame` che rimane dopo aver separato i nodi specificati.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usare split_rows per dividere le righe in un **`DynamicFrame`**

Questo esempio di codice utilizza il metodo `split_rows` per dividere le righe in un `DynamicFrame` in base al valore del campo `id`.

Set di dati di esempio

L'esempio utilizza un `DynamicFrame` chiamato `l_root_contact_details` che proviene da una raccolta denominata `legislators_relationalized`.

`l_root_contact_details` ha il seguente schema e le seguenti voci.

```
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                  phone|             202-225-5265|
|  1|    1|                twitter|              kathyhochul|
|  2|    0|                  phone|             202-225-3252|
```

```
|  2|    1|                twitter|            repjackyrosen|
|  3|    0|                    fax|             202-225-1314|
|  3|    1|                  phone|             202-225-3772|
|  3|    2|                twitter|          MikeRossUpdates|
|  4|    0|                    fax|             202-225-1314|
|  4|    1|                  phone|             202-225-3772|
|  4|    2|                twitter|          MikeRossUpdates|
|  5|    0|                    fax|             202-225-1314|
|  5|    1|                  phone|             202-225-3772|
|  5|    2|                twitter|          MikeRossUpdates|
|  6|    0|                    fax|             202-225-1314|
|  6|    1|                  phone|             202-225-3772|
|  6|    2|                twitter|          MikeRossUpdates|
|  7|    0|                    fax|             202-225-1314|
|  7|    1|                  phone|             202-225-3772|
|  7|    2|                twitter|          MikeRossUpdates|
|  8|    0|                    fax|             202-225-1314|
+---+-----+-----------------------+-------------------------+
```

Esempio di codice

```
# Example: Use split_rows to split up
# rows in a DynamicFrame based on value

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Retrieve the DynamicFrame to split
frame_to_split = legislators_relationalized.select("l_root_contact_details")

# Split up rows by ID
split_rows_collection = frame_to_split.split_rows({"id": {">": 10}}, "high", "low")

# Inspect the resulting DynamicFrames
print("Inspect the DynamicFrame that contains IDs < 10")
split_rows_collection.select("low").toDF().show()
print("Inspect the DynamicFrame that contains IDs > 10")
split_rows_collection.select("high").toDF().show()
```

Output

```
Inspect the DynamicFrame that contains IDs < 10
+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                  phone|             202-225-5265|
|  1|    1|                twitter|              kathyhochul|
|  2|    0|                  phone|             202-225-3252|
|  2|    1|                twitter|            repjackyrosen|
|  3|    0|                    fax|             202-225-1314|
|  3|    1|                  phone|             202-225-3772|
|  3|    2|                twitter|          MikeRossUpdates|
|  4|    0|                    fax|             202-225-1314|
|  4|    1|                  phone|             202-225-3772|
|  4|    2|                twitter|          MikeRossUpdates|
|  5|    0|                    fax|             202-225-1314|
|  5|    1|                  phone|             202-225-3772|
|  5|    2|                twitter|          MikeRossUpdates|
|  6|    0|                    fax|             202-225-1314|
|  6|    1|                  phone|             202-225-3772|
|  6|    2|                twitter|          MikeRossUpdates|
|  7|    0|                    fax|             202-225-1314|
|  7|    1|                  phone|             202-225-3772|
|  7|    2|                twitter|          MikeRossUpdates|
|  8|    0|                    fax|             202-225-1314|
+---+-----+-----------------------+-----------------------+
only showing top 20 rows

Inspect the DynamicFrame that contains IDs > 10
+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
| 11|    0|                  phone|             202-225-5476|
| 11|    1|                twitter|             RepDavidYoung|
| 12|    0|                  phone|             202-225-4035|
| 12|    1|                twitter|           RepStephMurphy|
| 13|    0|                    fax|             202-226-0774|
| 13|    1|                  phone|             202-225-6335|
| 14|    0|                    fax|             202-226-0774|
| 14|    1|                  phone|             202-225-6335|
| 15|    0|                    fax|             202-226-0774|
| 15|    1|                  phone|             202-225-6335|
| 16|    0|                    fax|             202-226-0774|
```

```
| 16|    1|                  phone|             202-225-6335|
| 17|    0|                    fax|             202-226-0774|
| 17|    1|                  phone|             202-225-6335|
| 18|    0|                    fax|             202-226-0774|
| 18|    1|                  phone|             202-225-6335|
| 19|    0|                    fax|             202-226-0774|
| 19|    1|                  phone|             202-225-6335|
| 20|    0|                    fax|             202-226-0774|
| 20|    1|                  phone|             202-225-6335|
+---+-----+-----------------------+-------------------------+
only showing top 20 rows
```

unbox

`unbox(path, format, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0, **options)`

Esegue la conversione unboxing di un campo stringa in un oggetto `DynamicFrame` e restituisce un nuovo oggetto `DynamicFrame` che contiene gli oggetti `DynamicRecords` sottoposti a conversione unboxing.

Un `DynamicRecord` rappresenta un record logico all'interno di un `DynamicFrame`. È simile a una riga in un `DataFrame` Apache Spark, con la differenza che è autodescrittivo e può essere utilizzato per dati non conformi a uno schema fisso.

- `path`: un percorso completo al nodo stringa che desideri cancellare.
- `format`: una specifica del formato (facoltativa). Viene usata per una connessione Amazon S3 o AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

- `options`: una o più delle seguenti:
  - `separator`: una stringa che contiene il carattere separatore.
  - `escaper`: una stringa che contiene il carattere escape.
  - `skipFirst`: un valore Boolean che indica se saltare la prima istanza.
  - `withSchema`: una stringa contenente una rappresentazione JSON dello schema del nodo. Il formato della rappresentazione JSON di uno schema è definito dall'output di `StructType.json()`.
  - `withHeader`: un valore Boolean che indica se è inclusa un'intestazione.

Esempio: usare unbox per decomprimere un campo di stringa in un campo struct

Questo esempio di codice utilizza il metodo `unbox` per decomprimere o riformattare un campo di tipo stringa `DynamicFrame` in un campo di tipo struct.

Set di dati di esempio

L'esempio utilizza un `DynamicFrame` chiamato `mapped_with_string` con i seguenti schema e voci:

Nota il campo denominato `AddressString`. Questo è il campo che di cui l'esempio esegue l'unboxing in un campo struct.

```
root
|-- Average Total Payments: string
|-- AddressString: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

```
+----------------------+--------------------+-----------------------
+--------------------+-------------------------+-----------------------------------
+--------------------+-----------+---------------+--------------------+
|Average Total Payments|       AddressString|Average Covered Charges|      DRG
 Definition|Average Medicare Payments|Hospital Referral Region Description|
 Address|Provider Id|Total Discharges|       Provider Name|
+----------------------+--------------------+-----------------------
+--------------------+-------------------------+-----------------------------------
+--------------------+-----------+---------------+--------------------+
|              $5777.24|{"Street": "1108 ...|             $32963.07|039 -
 EXTRACRANIA...|                $4763.73|                        AL - Dothan|[36301,
 DOTHAN, [...|      10001|             91|SOUTHEAST ALABAMA...|
|              $5787.57|{"Street": "2505 ...|             $15131.85|039 -
 EXTRACRANIA...|                $4976.71|                    AL - Birmingham|[35957,
 BOAZ, [25...|      10005|             14|MARSHALL MEDICAL ...|
|              $5434.95|{"Street": "205 M...|             $37560.37|039 -
 EXTRACRANIA...|                $4453.79|                    AL - Birmingham|[35631,
 FLORENCE,...|      10006|             24|ELIZA COFFEE MEMO...|
|              $5417.56|{"Street": "50 ME...|             $13998.28|039 -
 EXTRACRANIA...|                $4129.16|                    AL - Birmingham|[35235,
 BIRMINGHA...|      10011|             25|   ST VINCENT'S EAST|
...
```

Esempio di codice

```
# Example: Use unbox to unbox a string field
# into a struct in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

unboxed = mapped_with_string.unbox("AddressString", "json")
unboxed.printSchema()
unboxed.toDF().show()
```

Output

```
root
```

```
|-- Average Total Payments: string
|-- AddressString: struct
|    |-- Street: string
|    |-- City: string
|    |-- State: string
|    |-- Zip.Code: string
|    |-- Array: array
|    |    |-- element: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string

+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+-----------+---------------+--------------------+
|Average Total Payments|       AddressString|Average Covered Charges|      DRG
 Definition|Average Medicare Payments|Hospital Referral Region Description|
 Address|Provider Id|Total Discharges|       Provider Name|
+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+-----------+---------------+--------------------+
|              $5777.24|[1108 ROSS CLARK ...|             $32963.07|039 -
 EXTRACRANIA...|                $4763.73|                        AL - Dothan|[36301,
 DOTHAN, [...|      10001|             91|SOUTHEAST ALABAMA...|
|              $5787.57|[2505 U S HIGHWAY...|             $15131.85|039 -
 EXTRACRANIA...|                $4976.71|                    AL - Birmingham|[35957,
 BOAZ, [25...|      10005|             14|MARSHALL MEDICAL ...|
|              $5434.95|[205 MARENGO STRE...|             $37560.37|039 -
 EXTRACRANIA...|                $4453.79|                    AL - Birmingham|[35631,
 FLORENCE,...|      10006|             24|ELIZA COFFEE MEMO...|
|              $5417.56|[50 MEDICAL PARK ...|             $13998.28|039 -
 EXTRACRANIA...|                $4129.16|                    AL - Birmingham|[35235,
 BIRMINGHA...|      10011|             25|   ST VINCENT'S EAST|
```

```
|              $5658.33|[1000 FIRST STREE...|              $31633.27|039 -
 EXTRACRANIA...|                 $4851.44|                     AL - Birmingham|[35007,
 ALABASTER...|      10016|              18|SHELBY BAPTIST ME...|
|              $6653.80|[2105 EAST SOUTH ...|              $16920.79|039 -
 EXTRACRANIA...|                 $5374.14|                     AL - Montgomery|[36116,
 MONTGOMER...|      10023|              67|BAPTIST MEDICAL C...|
|              $5834.74|[2000 PEPPERELL P...|              $11977.13|039 -
 EXTRACRANIA...|                 $4761.41|                     AL - Birmingham|[36801,
 OPELIKA, ...|      10029|              51|EAST ALABAMA MEDI...|
|              $8031.12|[619 SOUTH 19TH S...|              $35841.09|039 -
 EXTRACRANIA...|                 $5858.50|                     AL - Birmingham|[35233,
 BIRMINGHA...|      10033|              32|UNIVERSITY OF ALA...|
|              $6113.38|[101 SIVLEY RD, H...|              $28523.39|039 -
 EXTRACRANIA...|                 $5228.40|                     AL - Huntsville|[35801,
 HUNTSVILL...|      10039|             135| HUNTSVILLE HOSPITAL|
|              $5541.05|[1007 GOODYEAR AV...|              $75233.38|039 -
 EXTRACRANIA...|                 $4386.94|                     AL - Birmingham|[35903,
 GADSDEN, ...|      10040|              34|GADSDEN REGIONAL ...|
|              $5461.57|[600 SOUTH THIRD ...|              $67327.92|039 -
 EXTRACRANIA...|                 $4493.57|                     AL - Birmingham|[35901,
 GADSDEN, ...|      10046|              14|RIVERVIEW REGIONA...|
|              $5356.28|[4370 WEST MAIN S...|              $39607.28|039 -
 EXTRACRANIA...|                 $4408.20|                         AL - Dothan|[36305,
 DOTHAN, [...|      10055|              45|    FLOWERS HOSPITAL|
|              $5374.65|[810 ST VINCENT'S...|              $22862.23|039 -
 EXTRACRANIA...|                 $4186.02|                     AL - Birmingham|[35205,
 BIRMINGHA...|      10056|              43|ST VINCENT'S BIRM...|
|              $5366.23|[400 EAST 10TH ST...|              $31110.85|039 -
 EXTRACRANIA...|                 $4376.23|                     AL - Birmingham|[36207,
 ANNISTON,...|      10078|              21|NORTHEAST ALABAMA...|
|              $5282.93|[1613 NORTH MCKEN...|              $25411.33|039 -
 EXTRACRANIA...|                 $4383.73|                         AL - Mobile|[36535,
 FOLEY, [1...|      10083|              15|SOUTH BALDWIN REG...|
|              $5676.55|[1201 7TH STREET ...|               $9234.51|039 -
 EXTRACRANIA...|                 $4509.11|                     AL - Huntsville|[35609,
 DECATUR, ...|      10085|              27|DECATUR GENERAL H...|
|              $5930.11|[6801 AIRPORT BOU...|              $15895.85|039 -
 EXTRACRANIA...|                 $3972.85|                         AL - Mobile|[36608,
 MOBILE, [...|      10090|              27| PROVIDENCE HOSPITAL|
|              $6192.54|[809 UNIVERSITY B...|              $19721.16|039 -
 EXTRACRANIA...|                 $5179.38|                     AL - Tuscaloosa|[35401,
 TUSCALOOS...|      10092|              31|D C H REGIONAL ME...|
```

```
|              $4968.00|[750 MORPHY AVENU...|              $10710.88|039 -
 EXTRACRANIA...|                 $3898.88|                        AL - Mobile|[36532,
 FAIRHOPE,...|      10100|              18|     THOMAS HOSPITAL|
|              $5996.00|[701 PRINCETON AV...|              $51343.75|039 -
 EXTRACRANIA...|                 $4962.45|                    AL - Birmingham|[35211,
 BIRMINGHA...|      10103|              33|BAPTIST MEDICAL C...|
+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+----------+---------------+--------------------+
only showing top 20 rows
```

unione

union(frame1, frame2, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Unione due. DynamicFrames Restituzioni DynamicFrame contenenti tutti i record di entrambi gli input DynamicFrames. Questa trasformazione può restituire risultati diversi dall'unione di due DataFrames con dati equivalenti. Se hai bisogno del comportamento dell' DataFrame unione Spark, prendi in considerazione l'utilizzo `toDF` di.

- `frame1`— I primi DynamicFrame a unirsi.
- `frame2`— Seconda dopo DynamicFrame l'unione.
- `transformation_ctx`: (facoltativo) una stringa univoca utilizzata per identificare informazioni su statistiche/stato
- `info`: (facoltativo) qualsiasi stringa da associare agli errori nella trasformazione
- `stageThreshold`: (facoltativo) numero massimo di errori nella trasformazione fino a che l'elaborazione si interrompe a causa di un errore
- `totalThreshold`: (facoltativo) numero massimo di errori totali fino a che l'elaborazione si interrompe a causa di un errore.

unnest

unnest(transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

Annulla l'annidamento di oggetti nidificati in un `DynamicFrame` rendendoli oggetti di primo livello e restituendo un nuovo `DynamicFrame` non nidificato.

- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.
- `totalThreshold`: il numero massimo di errori rilevati durante questa trasformazione raggiunto il quale il processo deve interrompersi (facoltativo). L'impostazione predefinita è zero, indicando che il processo non deve terminare.

Esempio: usare unnest per trasformare i campi annidati in campi di primo livello

Questo esempio di codice utilizza il metodo `unnest` per raggruppare tutti i campi annidati di a`DynamicFrame` in campi di primo livello.

Set di dati di esempio

L'esempio utilizza un `DynamicFrame` chiamato `mapped_medicare` con il seguente schema. Nota che il campo `Address` è l'unico campo che contiene dati annidati.

```
root
|-- Average Total Payments: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

Esempio di codice

```
# Example: Use unnest to unnest nested
```

```
# objects in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Unnest all nested fields
unnested = mapped_medicare.unnest()
unnested.printSchema()
```

Output

```
root
|-- Average Total Payments: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address.Zip.Code: string
|-- Address.City: string
|-- Address.Array: array
|    |-- element: string
|-- Address.State: string
|-- Address.Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

unnest_ddb_json

Snidifica le colonne nidificate in un `DynamicFrame` che si trovano specificamente nella struttura JSON di DynamoDB e restituisce un nuovo `DynamicFrame` non annidato. Le colonne che sono di un array di struct non verranno annidate. Si noti che si tratta di un tipo specifico di trasformazione di snidamento che si comporta in modo diverso dalla normale trasformazione di `unnest` e richiede che i dati siano già nella struttura JSON di DynamoDB. Per ulteriori informazioni, consulta DynamoDB JSON.

`unnest_ddb_json(transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa da associare alla segnalazione errori per questa trasformazione (opzionale).
- `stageThreshold`: il numero di errori riscontrati durante questa trasformazione, raggiunto il quale il processo dovrebbe interrompersi (opzionale impostazione predefinita: zero, a indicare che il processo non dovrebbe interrompersi).
- `totalThreshold`: il numero di errori riscontrati fino a questa trasformazione compresa, raggiunto il quale il processo dovrebbe interrompersi (opzionale: impostazione predefinita: zero, a indicare che il processo non dovrebbe interrompersi).

Ad esempio, lo schema di lettura di un'esportazione con la struttura JSON DynamoDB potrebbe apparire come segue:

```
root
|-- Item: struct
|    |-- ColA: struct
|    |    |-- S: string
|    |-- ColB: struct
|    |    |-- S: string
|    |-- ColC: struct
|    |    |-- N: string
|    |-- ColD: struct
|    |    |-- L: array
|    |    |    |-- element: null
```

La trasformazione di `unnest_ddb_json()` lo convertirebbe in:

```
root
|-- ColA: string
|-- ColB: string
|-- ColC: string
|-- ColD: array
|    |-- element: null
```

L'esempio di codice seguente mostra come utilizzare il connettore di esportazione AWS Glue DynamoDB, richiamare un unnest JSON di DynamoDB e stampare il numero di partizioni:

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dynamicFrame = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": "<test_source>",
        "dynamodb.s3.bucket": "<bucket name>",
        "dynamodb.s3.prefix": "<bucket prefix>",
        "dynamodb.s3.bucketOwner": "<account_id>",
    }
)
unnested = dynamicFrame.unnest_ddb_json()
print(unnested.getNumPartitions())

job.commit()
```

write

`write(connection_type, connection_options, format, format_options, accumulator_size)`

Ottiene un DataSink(object) del tipo di connessione specificata da Classe GlueContext di questo `DynamicFrame` e lo utilizza per formattare e scrivere i contenuti di questo `DynamicFrame`. Restituisce il nuovo `DynamicFrame` formattato e scritto come specificato.

- `connection_type`: il tipo di connessione da utilizzare. I valori validi sono `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` e `oracle`.
- `connection_options`: l'opzione di connessione da utilizzare (opzionale). Per un `connection_type` di `s3` è definito un percorso Amazon S3.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Per le connessioni JDBC, diverse proprietà devono essere definite. Il nome del database deve fare parte dell'URL. Puoi opzionalmente essere incluso nelle opzioni di connessione.

> **Warning**
>
> Si consiglia di non archiviare le password nello script. Valuta la possibilità `boto3` di utilizzarli per recuperarli da AWS Secrets Manager o dal AWS Glue Data Catalog.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-
path"}
```

- `format`: una specifica del formato (facoltativa). Viene utilizzato per un servizio Amazon Simple Storage Service (Amazon S3) o una connessione AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `format_options`: opzioni di formato per il formato specificato. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `accumulator_size`: la dimensione accumulabile da utilizzare, in byte (facoltativa).

— errori —

- assertErrorThreshold
- errorsAsDynamicCornice
- errorsCount
- stageErrorsCount

assertErrorThreshold

`assertErrorThreshold( )`: asserzione per gli errori nelle trasformazioni che hanno creato questo oggetto `DynamicFrame`. Restituisce una `Exception` dal `DataFrame` sottostante.

errorsAsDynamicCornice

`errorsAsDynamicFrame( )`: restituisce un `DynamicFrame` che ha record di errore nidificati al suo interno.

Esempio: utilizzare errorsAsDynamic Frame per visualizzare i record di errori

L'esempio di codice seguente mostra come utilizzare il metodo `errorsAsDynamicFrame` per visualizzare un record degli errori per un `DynamicFrame`.

Set di dati di esempio

L'esempio utilizza il set di dati seguente che puoi caricare in Amazon S3 come JSON. Tieni presente che il formato del secondo record non è corretto. I dati con formato non corretto generalmente interrompono l'analisi dei file quando utilizzi SparkSQL. `DynamicFrame`, tuttavia, riconosce i problemi di formato non corretto e trasforma le righe con formato non corretto in record degli errori che puoi gestire singolarmente.

```
{"id": 1, "name": "george", "surname": "washington", "height": 178}
{"id": 2, "name": "benjamin", "surname": "franklin",
{"id": 3, "name": "alexander", "surname": "hamilton", "height": 171}
{"id": 4, "name": "john", "surname": "jay", "height": 190}
```

Esempio di codice

```
# Example: Use errorsAsDynamicFrame to view error records.
# Replace s3://DOC-EXAMPLE-S3-BUCKET/error_data.json with your location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create errors DynamicFrame, view schema
errors = glueContext.create_dynamic_frame.from_options(
    "s3", {"paths": ["s3://DOC-EXAMPLE-S3-BUCKET/error_data.json"]}, "json"
)
print("Schema of errors DynamicFrame:")
errors.printSchema()

# Show that errors only contains valid entries from the dataset
print("errors contains only valid records from the input dataset (2 of 4 records)")
errors.toDF().show()

# View errors
```

```
print("Errors count:", str(errors.errorsCount()))
print("Errors:")
errors.errorsAsDynamicFrame().toDF().show()

# View error fields and error data
error_record = errors.errorsAsDynamicFrame().toDF().head()

error_fields = error_record["error"]
print("Error fields: ")
print(error_fields.asDict().keys())

print("\nError record data:")
for key in error_fields.asDict().keys():
    print("\n", key, ": ", str(error_fields[key]))
```

Output

```
Schema of errors DynamicFrame:
root
|-- id: int
|-- name: string
|-- surname: string
|-- height: int

errors contains only valid records from the input dataset (2 of 4 records)
+---+------+----------+------+
| id|  name|   surname|height|
+---+------+----------+------+
|  1|george|washington|   178|
|  4|  john|       jay|   190|
+---+------+----------+------+

Errors count: 1
Errors:
+--------------------+
|               error|
+--------------------+
|[[  File "/tmp/20...|
+--------------------+

Error fields:
dict_keys(['callsite', 'msg', 'stackTrace', 'input', 'bytesread', 'source',
 'dynamicRecord'])
```

```
Error record data:

 callsite :  Row(site='  File "/tmp/2060612586885849088", line 549, in <module>\n
 sys.exit(main())\n  File "/tmp/2060612586885849088", line 523, in main\n    response
 = handler(content)\n  File "/tmp/2060612586885849088", line 197, in execute_request
\n    result = node.execute()\n  File "/tmp/2060612586885849088", line 103, in
 execute\n    exec(code, global_dict)\n  File "<stdin>", line 10, in <module>\n
 File "/opt/amazon/lib/python3.6/site-packages/awsglue/dynamicframe.py", line 625, in
 from_options\n    format_options, transformation_ctx, push_down_predicate, **kwargs)\n
  File "/opt/amazon/lib/python3.6/site-packages/awsglue/context.py", line 233, in
 create_dynamic_frame_from_options\n    source.setFormat(format, **format_options)\n',
 info='')

 msg :  error in jackson reader

 stackTrace :  com.fasterxml.jackson.core.JsonParseException: Unexpected character
 ('{' (code 123)): was expecting either valid name character (for unquoted name) or
 double-quote (for quoted) to start field name
 at [Source: com.amazonaws.services.glue.readers.BufferedStream@73492578; line: 3,
 column: 2]
 at com.fasterxml.jackson.core.JsonParser._constructError(JsonParser.java:1581)
 at
 com.fasterxml.jackson.core.base.ParserMinimalBase._reportError(ParserMinimalBase.java:533)
 at
 com.fasterxml.jackson.core.base.ParserMinimalBase._reportUnexpectedChar(ParserMinimalBase.java
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser._handleOddName(UTF8StreamJsonParser.java:
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser._parseName(UTF8StreamJsonParser.java:1650
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser.nextToken(UTF8StreamJsonParser.java:740)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun$hasNextGoodToken
$1.apply(JacksonReader.scala:57)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun$hasNextGoodToken
$1.apply(JacksonReader.scala:57)
 at scala.collection.Iterator$$anon$9.next(Iterator.scala:162)
 at scala.collection.Iterator$$anon$16.hasNext(Iterator.scala:599)
 at scala.collection.Iterator$$anon$16.hasNext(Iterator.scala:598)
 at scala.collection.Iterator$class.foreach(Iterator.scala:891)
 at scala.collection.AbstractIterator.foreach(Iterator.scala:1334)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun
$1.apply(JacksonReader.scala:120)
```

```
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun
$1.apply(JacksonReader.scala:116)
 at
 com.amazonaws.services.glue.DynamicRecordBuilder.handleErr(DynamicRecordBuilder.scala:209)
 at
 com.amazonaws.services.glue.DynamicRecordBuilder.handleErrorWithException(DynamicRecordBuilder
 at
 com.amazonaws.services.glue.readers.JacksonReader.nextFailSafe(JacksonReader.scala:116)
 at com.amazonaws.services.glue.readers.JacksonReader.next(JacksonReader.scala:109)
 at com.amazonaws.services.glue.readers.JSONReader.next(JSONReader.scala:247)
 at
 com.amazonaws.services.glue.hadoop.TapeHadoopRecordReaderSplittable.nextKeyValue(TapeHadoopRec
 at org.apache.spark.rdd.NewHadoopRDD$$anon$1.hasNext(NewHadoopRDD.scala:230)
 at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$13.hasNext(Iterator.scala:462)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$13.hasNext(Iterator.scala:462)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at org.apache.spark.sql.execution.SparkPlan$$anonfun$2.apply(SparkPlan.scala:255)
 at org.apache.spark.sql.execution.SparkPlan$$anonfun$2.apply(SparkPlan.scala:247)
 at org.apache.spark.rdd.RDD$$anonfun$mapPartitionsInternal$1$$anonfun$apply
$24.apply(RDD.scala:836)
 at org.apache.spark.rdd.RDD$$anonfun$mapPartitionsInternal$1$$anonfun$apply
$24.apply(RDD.scala:836)
 at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
 at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324)
 at org.apache.spark.rdd.RDD.iterator(RDD.scala:288)
 at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
 at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324)
 at org.apache.spark.rdd.RDD.iterator(RDD.scala:288)
 at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
 at org.apache.spark.scheduler.Task.run(Task.scala:121)
 at org.apache.spark.executor.Executor$TaskRunner$$anonfun$10.apply(Executor.scala:408)
 at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
 at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:414)
 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
 at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
 at java.lang.Thread.run(Thread.java:750)
```

```
input :

bytesread :  252

source :

dynamicRecord :  Row(id=2, name='benjamin', surname='franklin')
```

errorsCount

`errorsCount( )`: restituisce il numero totale di errori in un oggetto `DynamicFrame`.

stageErrorsCount

`stageErrorsCount`: restituisce il numero di errori che si sono verificati nel processo di generazione di questo oggetto `DynamicFrame`.

Classe DynamicFrameCollection

Un `DynamicFrameCollection` è un dizionario di oggetti DynamicFrame classe, in cui le chiavi sono i nomi di `DynamicFrames` e i valori sono gli oggetti `DynamicFrame`.

__init__

`__init__(dynamic_frames, glue_ctx)`

- `dynamic_frames`: un dizionario di oggetti DynamicFrame classe.
- `glue_ctx`: un oggetto Classe GlueContext.

Chiavi

`keys( )`: restituisce un elenco di chiavi in questa raccolta, che in genere contiene i nomi dei valori `DynamicFrame` corrispondenti.

Valori

`values(key)`: restituisce un elenco di valori `DynamicFrame` in questa raccolta.

Select

`select(key)`

Restituisce il `DynamicFrame` corrispondente alla chiave specificata (che in genere è il nome di `DynamicFrame`).

- `key`: una chiave in `DynamicFrameCollection`, che in genere rappresenta il nome di un `DynamicFrame`.

Eseguire la mappatura

`map(callable, transformation_ctx="")`

Utilizza una funzione passata per creare e restituire un nuovo `DynamicFrameCollection` basato su `DynamicFrames` in questa raccolta.

- `callable`: funzione che impiega `DynamicFrame` e il contesto di trasformazione specificato come parametri e restituisce un `DynamicFrame`.
- `transformation_ctx`: un contesto di trasformazione che deve essere utilizzato dal callable (facoltativo).

Flatmap

`flatmap(f, transformation_ctx="")`

Utilizza una funzione passata per creare e restituire un nuovo `DynamicFrameCollection` basato su `DynamicFrames` in questa raccolta.

- `f`: funzione che impiega `DynamicFrame` come parametro e restituisce uno `DynamicFrame` o `DynamicFrameCollection`.
- `transformation_ctx`: un contesto di trasformazione che deve essere utilizzato dalla funzione (facoltativo).

Classe DynamicFrameWriter

Metodi

- __init__

- from_options
- from_catalog
- from_jdbc_conf

__init__

`__init__(glue_context)`

- `glue_context`: il Classe GlueContext da usare.

from_options

`from_options(frame, connection_type, connection_options={}, format=None, format_options={}, transformation_ctx="")`

Scrive un `DynamicFrame` usando la connessione e il formato specificati.

- `frame`: il `DynamicFrame` da scrivere.
- `connection_type`: il tipo di connessione. I valori validi sono `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` e `oracle`.
- `connection_options`: opzioni di connessione, come tabella di database e percorso (opzionale). Per un `connection_type` di `s3` è definito un percorso Amazon S3.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Per le connessioni JDBC, diverse proprietà devono essere definite. Il nome del database deve fare parte dell'URL. Puoi opzionalmente essere incluso nelle opzioni di connessione.

> **Warning**
>
> Si consiglia di non archiviare le password nello script. Valuta la possibilità di utilizzare `boto3` per recuperarle da AWS Secrets Manager o da Catalogo dati AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

La proprietà `dbtable` è il nome della tabella JDBC. Per i archivi dati JDBC che supportano schemi all'interno di un database, specifica `schema.table-name`. Se non viene fornito alcuno schema, viene usato lo schema "pubblico" predefinito.

Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.

- `format`: una specifica del formato (facoltativa). Viene utilizzato per un servizio Amazon Simple Storage Service (Amazon S3) o una connessione AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `format_options`: opzioni di formato per il formato specificato. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `transformation_ctx`: un contesto di trasformazione da usare (opzionale).

from_catalog

`from_catalog(frame, name_space, table_name, redshift_tmp_dir="", transformation_ctx="")`

Scrive un `DynamicFrame` utilizzando il nome della tabella e il database del catalogo specificati.

- `frame`: il `DynamicFrame` da scrivere.
- `name_space`: il database da usare.
- `table_name`: il `table_name` da usare.
- `redshift_tmp_dir`: una directory temporanea Amazon Redshift da usare (opzionale).
- `transformation_ctx`: un contesto di trasformazione da usare (opzionale).
- `additional_options`: opzioni aggiuntive fornite a AWS Glue.

Per scrivere su tabelle governate da Lake Formation, è possibile utilizzare queste opzioni aggiuntive:

- `transactionId`: (stringa) l'ID transazione in cui eseguire la scrittura nella tabella Governed. Di questa transazione non può essere già stato eseguito il commit, né può essere interrotta, diversamente la scrittura non andrà a buon fine.
- `callDeleteObjectsOnCancel` : (booleano, facoltativo) Se impostato su `true` (default), AWS Glue chiama automaticamente l'API `DeleteObjectsOnCancel` dopo che l'oggetto è stato

scritto su Amazon S3. Per ulteriori informazioni, consulta DeleteObjectsOnCancel nella Guida per gli sviluppatori di AWS Lake Formation.

Example Esempio: scrittura su una tabella governata in Lake Formation

```
txId = glueContext.start_transaction(read_only=False)
glueContext.write_dynamic_frame.from_catalog(
    frame=dyf,
    database = db,
    table_name = tbl,
    transformation_ctx = "datasource0",
    additional_options={"transactionId":txId})
...
glueContext.commit_transaction(txId)
```

from_jdbc_conf

`from_jdbc_conf(frame, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx="")`

Scrive un `DynamicFrame` usando le informazioni sulla connessione JDBC specificate.

- `frame`: il `DynamicFrame` da scrivere.
- `catalog_connection`: una connessione del catalogo da utilizzare.
- `connection_options`: opzioni di connessione, come tabella di database e percorso (opzionale).
- `redshift_tmp_dir`: una directory temporanea Amazon Redshift da usare (opzionale).
- `transformation_ctx`: un contesto di trasformazione da usare (opzionale).

Esempio di write_dynamic_frame

Questo esempio scrive l'output localmente utilizzando un `connection_type` di S3 con un argomento percorso POSIX in `connection_options`, che consente di scrivere su storage locale.

```
glueContext.write_dynamic_frame.from_options(\
frame = dyf_splitFields,\
connection_options = {'path': '/home/glue/GlueLocalOutput/'},\
connection_type = 's3',\
format = 'json')
```

DynamicFrameReader classe

— metodi —

__init__

`__init__(glue_context)`

- `glue_context`: il Classe GlueContext da usare.

from_rdd

`from_rdd(data, name, schema=None, sampleRatio=None)`

Legge un `DynamicFrame` da un Resilient Distributed Dataset (RDD).

- `data`: il set di dati da cui leggere.
- `name`: il nome da cui leggere.
- `schema`: lo schema da leggere (opzionale).
- `sampleRatio`: il rapporto di esempio (facoltativo).

from_options

`from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx="")`

Legge un `DynamicFrame` usando la connessione e il formato specificati.

- `connection_type`: il tipo di connessione. I valori validi includono `s3`,`mysql`,`postgresql`,`redshift`,, `sqlserver` `oracle`,`dynamodb`, e. `snowflake`
- `connection_options`: opzioni di connessione, come tabella di database e percorso (opzionale). Per ulteriori informazioni, consulta Tipi di connessione e opzioni per ETL in AWS Glue for Spark. Per un `connection_type` di `s3`, i percorsi Amazon S3 sono definiti in una matrice.

```
connection_options = {"paths": [ "s3://mybucket/object_a", "s3://mybucket/object_b"]}
```

Per le connessioni JDBC, diverse proprietà devono essere definite. Il nome del database deve fare parte dell'URL. Puoi opzionalmente essere incluso nelle opzioni di connessione.

> **Warning**
>
> Si consiglia di non archiviare le password nello script. Valuta la possibilità `boto3` di utilizzarli per recuperarli da AWS Secrets Manager o dal AWS Glue Data Catalog.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

Per una connessione JDBC che esegue letture parallele, è possibile impostare l'opzione hashfield. Ad esempio:

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path" , "hashfield": "month"}
```

Per ulteriori informazioni, consulta Lettura in parallelo dalle tabelle JDBC.

- `format`: una specifica del formato (facoltativa). Viene utilizzato per un servizio Amazon Simple Storage Service (Amazon S3) o una connessione AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `format_options`: opzioni di formato per il formato specificato. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `push_down_predicate`: filtra le partizioni senza dover elencare e leggere tutti i file nel set di dati. Per ulteriori informazioni, consulta Prefiltraggio con i predicati pushdown.

from_catalog

`from_catalog(database, table_name, redshift_tmp_dir="", transformation_ctx="", push_down_predicate="", additional_options={})`

Legge un `DynamicFrame` utilizzando il nome della tabella e il namespace del catalogo specificati.

- `database`: il database da cui leggere.
- `table_name`: il nome della tabella da cui leggere.
- `redshift_tmp_dir`: una directory Amazon Redshift temporanea da utilizzare (facoltativa se non si leggono i dati da Redshift).
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `push_down_predicate`: filtra le partizioni senza dover elencare e leggere tutti i file nel set di dati. Per ulteriori informazioni, consulta Prefiltraggio con i predicati pushdown.
- `additional_options`: opzioni aggiuntive fornite a AWS Glue.
  - Per usare una connessione JDBC che esegue letture parallele, è possibile impostare l'opzione `hashfield`, `hashexpression` o `hashpartitions`. Ad esempio:

    ```
    additional_options = {"hashfield": "month"}
    ```

    Per ulteriori informazioni, consulta Lettura in parallelo dalle tabelle JDBC.
  - Per passare un'espressione di catalogo per filtrare in base alle colonne di indice, vedi l'opzione `catalogPartitionPredicate`.

    `catalogPartitionPredicate` — È possibile passare un'espressione di catalogo per filtrare in base alle colonne di indice. Questo esegue il push down del filtro sul lato server. Per ulteriori informazioni, consulta la pagina relativa agli indici di partizionamento di AWS Glue. Tieni presente che `push_down_predicate` e `catalogPartitionPredicate` usano sintassi diverse. Il primo utilizza la sintassi standard Spark SQL e il secondo utilizza il parser JSQL.

    Per ulteriori informazioni, consulta Gestione delle partizioni per l'output ETL in AWS Glue.
  - Per leggere da tabelle governate da Lake Formation, è possibile utilizzare queste opzioni aggiuntive:
    - `transactionId`: (stringa) l'ID della transazione in cui leggere il contenuto della tabella governata. Se non viene eseguito il commit di questa transazione, la lettura verrà trattata come parte di tale transazione e ne vedrà le scritture. Se viene eseguito il commit di questa

transazione, le sue scritture saranno visibili in questa lettura. Se questa transazione viene interrotta, verrà restituito un errore. Non può essere specificato insieme a `asOfTime`.

> **Note**
>
> `transactionId` o `asOfTime` deve essere impostato per accedere alla tabella governata.

- `asOfTime`— (TimeStamp: yyyy- [m] m- [d] d hh:mm:ss) L'ora a partire dalla quale leggere il contenuto della tabella. Non può essere specificato insieme a `transactionId`.
- `query`: (facoltativo) un'istruzione di query PartiQL utilizzata come input per il servizio di pianificazione Lake Formation. Se non è impostata, l'impostazione di default è quella di selezionare tutti i dati dalla tabella. Per ulteriori dettagli su PartiQL, consulta Supporto PartiQL nelle espressioni di filtro riga nella Guida per gli sviluppatori di AWS Lake Formation .

Example Esempio: utilizzo di un'istruzione di query PartiQL durante la lettura da una tabella governata in Lake Formation

```
txId = glueContext.start_transaction(read_only=False)
datasource0 = glueContext.create_dynamic_frame.from_catalog(
    database = db,
    table_name = tbl,
    transformation_ctx = "datasource0",
    additional_options={
    "transactionId":txId,
    "query":"SELECT * from tblName WHERE partitionKey=value;"
    })
...
glueContext.commit_transaction(txId)
```

Classe GlueContext

Esegue il wrapping dell'oggetto Apache Spark SparkContext e quindi fornisce meccanismi per interagire con la piattaforma di Apache Spark.

__init__

`__init__(sparkContext)`

- `sparkContext`: il contesto Apache Spark da usare.

Creazione

- __init__
- getSource
- create_dynamic_frame_from_rdd
- create_dynamic_frame_from_catalog
- create_dynamic_frame_from_options
- create_sample_dynamic_frame_from_catalog
- create_sample_dynamic_frame_from_options
- add_ingestion_time_columns
- create_data_frame_from_catalog
- create_data_frame_from_options
- forEachBatch

getSource

`getSource(connection_type, transformation_ctx = "", **options)`

Crea un oggetto `DataSource` che può essere utilizzato per leggere `DynamicFrames` da fonti esterne.

- `connection_type`: il tipo di connessione da utilizzare, ad esempio Amazon Simple Storage Service (Amazon S3), Amazon Redshift e JDBC. I valori validi includono `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` e dynamodb.
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `options`: una raccolta di coppie nome/valore opzionali. Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.

Di seguito è riportato un esempio dell'utilizzo di `getSource`:

```
>>> data_source = context.getSource("file", paths=["/in/path"])
>>> data_source.setFormat("json")
>>> myFrame = data_source.getFrame()
```

create_dynamic_frame_from_rdd

`create_dynamic_frame_from_rdd(data, name, schema=None, sample_ratio=None, transformation_ctx="")`

Restituisce un `DynamicFrame` che viene creato da un Apache Spark Resilient Distributed Dataset (RDD).

- `data`: l'origine dati da usare.
- `name`: il nome dei dati da usare.
- `schema`: lo schema da usare (opzionale).
- `sample_ratio`: il rapporto di esempio da usare (opzionale).
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).

create_dynamic_frame_from_catalog

`create_dynamic_frame_from_catalog(database, table_name, redshift_tmp_dir, transformation_ctx = "", push_down_predicate= "", additional_options = {}, catalog_id = None)`

Restituisce un `DynamicFrame` creato utilizzando un database del catalogo dati e un nome della tabella. Quando si utilizza questo metodo, si forniscono `format_options` tramite le proprietà della tabella sulla tabella Catalogo dati AWS Glue specificata e altre opzioni tramite l'argomento `additional_options`.

- `Database`: il database da cui leggere.
- `table_name`: il nome della tabella da cui leggere.
- `redshift_tmp_dir`: una directory temporanea Amazon Redshift da usare (opzionale).
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `push_down_predicate`: filtra le partizioni senza dover elencare e leggere tutti i file nel set di dati. Per ulteriori informazioni, consulta Prefiltraggio con i predicati pushdown.
- `additional_options`: una raccolta di coppie nome/valore opzionali. Le opzioni possibili includono quelle elencate in Tipi e opzioni di connessione per ETL in AWS Glue per Spark ad eccezione di `endpointUrl`, `streamName`, `bootstrap.servers`, `security.protocol`, `topicName`, `classification` e `delimiter`. Un'altra opzione supportata è `catalogPartitionPredicate`:

`catalogPartitionPredicate` — È possibile passare un'espressione di catalogo per filtrare in base alle colonne di indice. Questo esegue il push down del filtro sul lato server. Per ulteriori informazioni, consulta la pagina relativa agli indici di partizionamento di AWS Glue. Tieni presente che `push_down_predicate` e `catalogPartitionPredicate` usano sintassi diverse. Il primo utilizza la sintassi standard Spark SQL e il secondo utilizza il parser JSQL.

- `catalog_id`: l'ID catalogo (ID account) del catalogo dati a cui si accede. Se Nessuno, viene utilizzato l'ID account predefinito del chiamante.

create_dynamic_frame_from_options

`create_dynamic_frame_from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")`

Restituisce un `DynamicFrame` creato con la connessione e il formato specificati.

- `connection_type`: il tipo di connessione, come Amazon S3, Amazon Redshift e JDBC. I valori validi includono `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` e `dynamodb`.
- `connection_options`: opzioni di connessione, come tabella di database e percorsi (facoltativo). Per un oggetto `connection_type` di `s3`, viene definito un elenco di percorsi Amazon S3.

```
connection_options = {"paths": ["s3://aws-glue-target/temp"]}
```

Per le connessioni JDBC, diverse proprietà devono essere definite. Il nome del database deve fare parte dell'URL. Puoi opzionalmente essere incluso nelle opzioni di connessione.

> **Warning**
>
> Si consiglia di non archiviare le password nello script. Valuta la possibilità di utilizzare `boto3` per recuperarle da AWS Secrets Manager o da Catalogo dati AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

La proprietà `dbtable` è il nome della tabella JDBC. Per i archivi dati JDBC che supportano schemi all'interno di un database, specifica `schema.table-name`. Se non viene fornito alcuno schema, viene usato lo schema "pubblico" predefinito.

Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.

- `format`: una specifica del formato (facoltativa). Viene usata per una connessione Amazon S3 o AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `format_options`: opzioni di formato per il formato specificato. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `push_down_predicate`: filtra le partizioni senza dover elencare e leggere tutti i file nel set di dati. Per ulteriori informazioni, consulta Prefiltraggio con i predicati pushdown.

create_sample_dynamic_frame_from_catalog

`create_sample_dynamic_frame_from_catalog(database, table_name, num, redshift_tmp_dir, transformation_ctx = "", push_down_predicate= "", additional_options = {}, sample_options = {}, catalog_id = None)`

Restituisce un `DynamicFrame` di esempio creato utilizzando un database del catalogo dati e un nome della tabella. La `DynamicFrame` contiene solo i primi `num` registri da un'origine dati.

- `database`: il database da cui leggere.
- `table_name`: il nome della tabella da cui leggere.
- `num`: il numero massimo di registri nel frame dinamico di esempio restituito.
- `redshift_tmp_dir`: una directory temporanea Amazon Redshift da usare (opzionale).
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `push_down_predicate`: filtra le partizioni senza dover elencare e leggere tutti i file nel set di dati. Per ulteriori informazioni, consulta Prefiltraggio con i predicati pushdown.
- `additional_options`: una raccolta di coppie nome/valore opzionali. Le opzioni possibili includono quelle elencate in Tipi e opzioni di connessione per ETL in AWS Glue per Spark ad eccezione di `endpointUrl`, `streamName`, `bootstrap.servers`, `security.protocol`, `topicName`, `classification` e `delimiter`.

- `sample_options`: parametri per controllare il comportamento di campionamento (facoltativo). Parametri attuali disponibili per le origini Amazon S3:
  - `maxSamplePartitions`: il numero massimo di partizioni che il campionamento leggerà. Il valore predefinito è 10
  - `maxSampleFilesPerPartition`: il numero massimo di file che il campionamento leggerà in una partizione. Il valore predefinito è 10.

    Questi parametri aiutano a ridurre il tempo impiegato dall'elenco dei file. Ad esempio, supponiamo che il set di dati contenga 1000 partizioni e ogni partizione contenga 10 file. Se hai impostato `maxSamplePartitions` = 10 e `maxSampleFilesPerPartition` = 10, invece di elencare tutti i 10.000 file, il campionamento elencherà e leggerà solo le prime 10 partizioni con i primi 10 file in ognuna di esse: 10*10 = 100 file in totale.
- `catalog_id`: l'ID catalogo del catalogo dati a cui si accede (l'ID account del catalogo dati). Impostato su `None` per default. L'impostazione predefinita di `None` è l'ID catalogo dell'account chiamante nel servizio.

create_sample_dynamic_frame_from_options

`create_sample_dynamic_frame_from_options(connection_type, connection_options={}, num, sample_options={}, format=None, format_options={}, transformation_ctx = "")`

Restituisce un `DynamicFrame` di esempio creato con la connessione e il formato specificati. La `DynamicFrame` contiene solo i primi `num` registri da un'origine dati.

- `connection_type`: il tipo di connessione, come Amazon S3, Amazon Redshift e JDBC. I valori validi includono `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` e `dynamodb`.
- `connection_options`: opzioni di connessione, come tabella di database e percorsi (facoltativo). Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.
- `num`: il numero massimo di registri nel frame dinamico di esempio restituito.
- `sample_options`: parametri per controllare il comportamento di campionamento (facoltativo). Parametri attuali disponibili per le origini Amazon S3:
  - `maxSamplePartitions`: il numero massimo di partizioni che il campionamento leggerà. Il valore predefinito è 10
  - `maxSampleFilesPerPartition`: il numero massimo di file che il campionamento leggerà in una partizione. Il valore predefinito è 10.

Questi parametri aiutano a ridurre il tempo impiegato dall'elenco dei file. Ad esempio, supponiamo che il set di dati contenga 1000 partizioni e ogni partizione contenga 10 file. Se hai impostato `maxSamplePartitions` = 10 e `maxSampleFilesPerPartition` = 10, invece di elencare tutti i 10.000 file, il campionamento elencherà e leggerà solo le prime 10 partizioni con i primi 10 file in ognuna di esse: 10*10 = 100 file in totale.

- `format`: una specifica del formato (facoltativa). Viene usata per una connessione Amazon S3 o AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `format_options`: opzioni di formato per il formato specificato. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `push_down_predicate`: filtra le partizioni senza dover elencare e leggere tutti i file nel set di dati. Per ulteriori informazioni, consulta Prefiltraggio con i predicati pushdown.

add_ingestion_time_columns

`add_ingestion_time_columns(dataFrame, timeGranularity = "")`

Aggiunge colonne del tempo di importazione dati come `ingest_year`, `ingest_month`, `ingest_day`, `ingest_hour`, `ingest_minute` al `DataFrame` di input. Questa funzione viene generata automaticamente nello script generato da AWS Glue quando si specifica una tabella del catalogo dati con Amazon S3 come destinazione. Questa funzione aggiorna automaticamente la partizione con le colonne del tempo di importazione dati nella tabella di output. Ciò consente ai dati di output di venire partizionati automaticamente nel tempo di importazione dati senza necessitare di colonne di tempo di inserimento esplicite nei dati di input.

- `dataFrame`: il `dataFrame` al quale aggiungere le colonne del tempo di importazione dati.
- `timeGranularity`: la granularità delle colonne temporali. I valori validi sono "`day`", "`hour`" e "`minute`". Ad esempio, se "`hour`" viene passato alla funzione, il `dataFrame` originale avrà "`ingest_year`", "`ingest_month`", "`ingest_day`" e "`ingest_hour`" colonne temporali aggiunte.

Restituisce il frame di dati dopo l'aggiunta di colonne di granularità di tempo.

Esempio:

```
dynamic_frame = DynamicFrame.fromDF(glueContext.add_ingestion_time_columns(dataFrame,
 "hour"))
```

create_data_frame_from_catalog

`create_data_frame_from_catalog(database, table_name, transformation_ctx = "", additional_options = {})`

Restituisce un `DataFrame` creato utilizzando le informazioni da una tabella del catalogo dati.

- `database`: il database del catalogo dati da cui leggere.
- `table_name`: il nome della tabella de catalogo dati da cui leggere.
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `additional_options`: una raccolta di coppie nome/valore opzionali. Le opzioni possibili includono quelle elencate in Tipi e opzioni di connessione per ETL in AWS Glue per Spark per le origini di streaming, ad esempio `startingPosition`, `maxFetchTimeInMs`, e `startingOffsets`.
  - `useSparkDataSource`: se impostato su true, forza AWS Glue a utilizzare l'API nativa di Spark Data Source per leggere la tabella. L'API Spark Data Source supporta i seguenti formati: AVRO, binario, CSV, JSON, ORC, Parquet e testo. In una tabella del catalogo dati, il formato può essere specificato utilizzando la proprietà `classification`. Per ulteriori informazioni sull'API Spark Data Source, consulta la documentazione ufficiale di Apache Spark.

    L'uso di `create_data_frame_from_catalog` con `useSparkDataSource` offre i seguenti vantaggi:

    - Restituisce direttamente un `DataFrame` e fornisce un'alternativa a `create_dynamic_frame.from_catalog().toDF()`.
    - Supporta il controllo delle autorizzazioni a livello di tabella AWS Lake Formation per i formati nativi.
    - Supporta la lettura di formati di data lake senza il controllo delle autorizzazioni a livello di tabella AWS Lake Formation. Per ulteriori informazioni, consulta Utilizzo di framework data lake con processi ETL di AWS Glue.

    Quando abiliti `useSparkDataSource`, puoi anche aggiungere una qualsiasi delle opzioni Spark Data Source in `additional_options` a seconda delle necessità. AWS Glue passa queste opzioni direttamente al lettore Spark.

- `useCatalogSchema`: se impostato su true, AWS Glue applica lo schema del catalogo dati al `DataFrame` risultante. Altrimenti, il lettore deduce lo schema dai dati. Se abiliti l'opzione `useCatalogSchema`, dovrai impostare anche `useSparkDataSource` su true.

Limitazioni

Quando utilizzi l'opzione `useSparkDataSource` considera le seguenti limitazioni:

- Se usi `useSparkDataSource`, AWS Glue crea un nuovo `DataFrame` in una sessione Spark separata che è diversa dalla sessione Spark originale.
- Il filtro delle partizioni Spark DataFrame non funziona con le seguenti funzionalità di AWS Glue.
  - Segnalibri di processo
  - Esclusione delle classi di archiviazione di Amazon S3
  - Predicati di partizione di catalogo

  Per utilizzare il filtro delle partizioni con queste funzionalità, puoi usare il predicato pushdown di AWS Glue. Per ulteriori informazioni, consulta Prefiltraggio con i predicati pushdown. Il filtraggio sulle colonne non partizionate non viene modificato.

  Lo script di esempio seguente dimostra il modo errato di eseguire il filtraggio delle partizioni con l'opzione `excludeStorageClasses`.

```
// Incorrect partition filtering using Spark filter with excludeStorageClasses
read_df = glueContext.create_data_frame.from_catalog(
    database=database_name,
    table_name=table_name,
    additional_options = {
      "useSparkDataSource": True,
      "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
)

//  Suppose year and month are partition keys.
//  Filtering on year and month won't work, the filtered_df will still
//  contain data with other year/month values.
filtered_df = read_df.filter("year == '2017 and month == '04' and 'state == 'CA'")
```

  Lo script di esempio seguente dimostra il modo corretto di utilizzare un predicato pushdown in modo da eseguire il filtraggio delle partizioni con l'opzione `excludeStorageClasses`.

```
// Correct partition filtering using the AWS Glue pushdown predicate
// with excludeStorageClasses
read_df = glueContext.create_data_frame.from_catalog(
    database=database_name,
    table_name=table_name,
    //  Use AWS Glue pushdown predicate to perform partition filtering
    push_down_predicate = "(year=='2017' and month=='04')"
    additional_options = {
      "useSparkDataSource": True,
      "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
)

//  Use Spark filter only on non-partitioned columns
filtered_df = read_df.filter("state == 'CA'")
```

Esempio: creazione di una tabella CSV utilizzando il lettore di origini dati Spark

```
//  Read a CSV table with '\t' as separator
read_df = glueContext.create_data_frame.from_catalog(
    database=<database_name>,
    table_name=<table_name>,
    additional_options = {"useSparkDataSource": True,  "sep": '\t'}
)
```

create_data_frame_from_options

`create_data_frame_from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")`

Questa API è obsoleta. Utilizza invece le API `getSource()`. Restituisce un `DataFrame` creato con la connessione e il formato specificati. Utilizza questa funzione solo con origini di streaming AWS Glue.

- `connection_type`: il tipo di connessione streaming. I valori validi includono `kinesis` e `kafka`.
- `connection_options`: opzioni di connessione, che sono diverse per Kinesis e Kafka. È possibile trovare l'elenco di tutte le opzioni di connessione per ogni origine dati di streaming all'indirizzo Tipi e opzioni di connessione per ETL in AWS Glue per Spark. Di seguito vengono illustrate le differenze delle opzioni di connessione di streaming:

  - Le origini di streaming di Kinesis richiedono `streamARN`, `startingPosition`, `inferSchema` e `classification`.
  - Le origini di streaming di Kafka richiedono `connectionName`, `topicName`, `startingOffsets`, `inferSchema` e `classification`.
- `format`: una specifica del formato (facoltativa). Viene usata per una connessione Amazon S3 o AWS Glue che supporta più formati. Per ulteriori informazioni sui formati supportati, consulta Opzioni del formato dati per input e output in AWS Glue per Spark.
- `format_options`: opzioni di formato per il formato specificato. Per ulteriori informazioni sulle opzioni di formato supportate, consulta Opzioni del formato dati per input e output in AWS Glue per Spark.
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).

Esempio per l'origine di streaming Amazon Kinesis:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Esempio per l'origine di streaming Kafka:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

forEachBatch

`forEachBatch(frame, batch_function, options)`

Applica il `batch_function` passato a ogni micro batch che viene letto dall'origine di streaming.

- `frame`: il frame di dati contenente il micro batch corrente.
- `batch_function`: una funzione che verrà applicata per ogni micro batch.
- `options`: una raccolta di coppie chiave-valore che contiene informazioni su come elaborare micro batch. Sono richieste le seguenti opzioni:
  - `windowSize`: la quantità di tempo da dedicare all'elaborazione di ciascun batch.
  - `checkpointLocation`: la posizione in cui sono archiviati i checkpoint per il processo ETL di streaming.
  - `batchMaxRetries`: numero massimo di tentativi per riprovare il processo se il batch ha esito negativo. Il valore predefinito è 3. Questa opzione è configurabile solo per Glue versione 2.0 e successive.

Esempio:

```
glueContext.forEachBatch(
    frame = data_frame_datasource0,
    batch_function = processBatch,
    options = {
        "windowSize": "100 seconds",
        "checkpointLocation": "s3://kafka-auth-dataplane/confluent-test/output/
checkpoint/"
    }
)

def processBatch(data_frame, batchId):
    if (data_frame.count() > 0):
        datasource0 = DynamicFrame.fromDF(
          glueContext.add_ingestion_time_columns(data_frame, "hour"),
          glueContext, "from_data_frame"
        )
        additionalOptions_datasink1 = {"enableUpdateCatalog": True}
        additionalOptions_datasink1["partitionKeys"] = ["ingest_yr", "ingest_mo",
 "ingest_day"]
        datasink1 = glueContext.write_dynamic_frame.from_catalog(
          frame = datasource0,
```

```
          database = "tempdb",
          table_name = "kafka-auth-table-output",
          transformation_ctx = "datasink1",
          additional_options = additionalOptions_datasink1
        )
```

Utilizzo di set di dati in Amazon S3

- purge_table
- purge_s3_path
- transition_table
- transition_s3_path

purge_table

`purge_table(catalog_id=None, database="", table_name="", options={}, transformation_ctx="")`

Elimina i file da Amazon S3 per il database e la tabella del catalogo specificati. Se tutti i file in una partizione vengono eliminati, anche la partizione viene eliminata dal catalogo.

Per poter recuperare gli oggetti eliminati, puoi abilitare la funzione di controllo delle versioni degli oggetti nel bucket Amazon S3. Quando un oggetto viene eliminato da un bucket per il quale non è abilitata la funzione Versioni multiple degli oggetti, l'oggetto non può essere recuperato. Per ulteriori informazioni su come recuperare gli oggetti eliminati in un bucket abilitato per le versioni, consulta In che modo può essere recuperato un oggetto Amazon S3 che è stato eliminato? nel Portale del sapere di AWS Support.

- `catalog_id`: l'ID catalogo del catalogo dati a cui si accede (l'ID account del catalogo dati). Impostato su `None` per default. L'impostazione predefinita di `None` è l'ID catalogo dell'account chiamante nel servizio.
- `database`: il database da usare.
- `table_name`: il nome della tabella da usare.
- `options`: opzioni per filtrare i file da eliminare e per la generazione di file manifesto.
  - `retentionPeriod`: specifica un periodo in numero di ore per la conservazione dei file. I file più recenti del periodo di conservazione vengono mantenuti. Impostato su 168 ore (7 giorni) per impostazione predefinita.

- `partitionPredicate`: le partizioni che soddisfano questo predicato vengono eliminate. I file all'interno del periodo di conservazione in queste partizioni non vengono eliminati. Impostato su `""`: vuoto per impostazione predefinita.
- `excludeStorageClasses`: i file con classe di storage nel `excludeStorageClasses` non vengono eliminati. L'impostazione di default è `Set()`: un set vuoto.
- `manifestFilePath`: un percorso facoltativo per la generazione di file manifesto. Tutti i file che sono stati eliminati correttamente vengono registrati in `Success.csv` e quelli che non sono riusciti in `Failed.csv`

- `transformation_ctx`: il contesto di trasformazione da usare (opzionale). Utilizzato nel percorso del file manifest.

Example

```
glueContext.purge_table("database", "table", {"partitionPredicate": "(month=='march')",
 "retentionPeriod": 1, "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath":
 "s3://bucketmanifest/"})
```

purge_s3_path

`purge_s3_path(s3_path, options={}, transformation_ctx="")`

Elimina i file dal percorso Amazon S3 specificato in modo ricorsivo.

Per poter recuperare gli oggetti eliminati, puoi abilitare la funzione di controllo delle versioni degli oggetti nel bucket Amazon S3. Quando un oggetto viene eliminato da un bucket per il quale non è abilitata la funzione di controllo delle versioni degli oggetti, l'oggetto non può essere recuperato. Per ulteriori informazioni su come recuperare gli oggetti eliminati in un bucket abilitato per il controllo delle versioni, consulta In che modo può essere recuperato un oggetto Amazon S3 che è stato eliminato? nel Portale del sapere di AWS Support.

- `s3_path`: il percorso in Amazon S3 dei file da eliminare nel formato `s3://<bucket>/<prefix>/`
- `options`: opzioni per filtrare i file da eliminare e per la generazione di file manifesto.
  - `retentionPeriod`: specifica un periodo in numero di ore per la conservazione dei file. I file più recenti del periodo di conservazione vengono mantenuti. Impostato su 168 ore (7 giorni) per impostazione predefinita.

- `excludeStorageClasses`: i file con classe di storage nel `excludeStorageClasses` non vengono eliminati. L'impostazione di default è `Set()`: un set vuoto.
- `manifestFilePath`: un percorso facoltativo per la generazione di file manifesto. Tutti i file che sono stati eliminati correttamente vengono registrati in `Success.csv` e quelli che non sono riusciti in `Failed.csv`

- `transformation_ctx`: il contesto di trasformazione da usare (opzionale). Utilizzato nel percorso del file manifest.

Example

```
glueContext.purge_s3_path("s3://bucket/path/", {"retentionPeriod": 1,
 "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath": "s3://bucketmanifest/"})
```

transition_table

`transition_table(database, table_name, transition_to, options={}, transformation_ctx="", catalog_id=None)`

Esegue la transizione della classe di storage dei file archiviati su Amazon S3 per il database e la tabella del catalogo specificati.

Puoi eseguire la transizione tra due classi di archiviazione qualsiasi. Per le classi di archiviazione `GLACIER` e `DEEP_ARCHIVE`, puoi passare a queste classi. Tuttavia, dovresti utilizzare un `S3 RESTORE` per eseguire la transizione dalle classi di archiviazione `GLACIER` a `DEEP_ARCHIVE`.

Se esegui processi ETL AWS Glue che leggono file o partizioni da Amazon S3, puoi escludere alcuni tipi di classe di archiviazione Amazon S3. Per ulteriori informazioni, consulta Esclusione delle classi di archiviazione Amazon S3.

- `database`: il database da usare.
- `table_name`: il nome della tabella da usare.
- `transition_to`: la classe di storage Amazon S3 in cui eseguire la transizione.
- `options`: opzioni per filtrare i file da eliminare e per la generazione di file manifesto.
  - `retentionPeriod`: specifica un periodo in numero di ore per la conservazione dei file. I file più recenti del periodo di conservazione vengono mantenuti. Impostato su 168 ore (7 giorni) per impostazione predefinita.

- `partitionPredicate`: le partizioni che soddisfano questo predicato vengono trasferite. I file all'interno del periodo di conservazione in queste partizioni non vengono passati. Impostato su `""`: vuoto per impostazione predefinita.
- `excludeStorageClasses`: i file con classe di storage nel set `excludeStorageClasses` non vengono passati. L'impostazione di default è `Set()`: un set vuoto.
- `manifestFilePath`: un percorso facoltativo per la generazione di file manifesto. Tutti i file che sono stati passati correttamente vengono registrati in `Success.csv` e quelli che non sono riusciti in `Failed.csv`
- `accountId`: l'ID account Amazon Web Services per eseguire la trasformazione di transizione. Obbligatorio per questa trasformazione.
- `roleArn`: il ruolo AWS per eseguire la trasformazione di transizione. Obbligatorio per questa trasformazione.

- `transformation_ctx`: il contesto di trasformazione da usare (opzionale). Utilizzato nel percorso del file manifest.
- `catalog_id`: l'ID catalogo del catalogo dati a cui si accede (l'ID account del catalogo dati). Impostato su `None` per default. L'impostazione predefinita di `None` è l'ID catalogo dell'account chiamante nel servizio.

Example

```
glueContext.transition_table("database", "table", "STANDARD_IA", {"retentionPeriod": 1,
 "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath": "s3://bucketmanifest/",
 "accountId": "12345678901", "roleArn": "arn:aws:iam::123456789012:user/example-
username"})
```

transition_s3_path

`transition_s3_path(s3_path, transition_to, options={}, transformation_ctx="")`

Esegue la transizione della classe di storage nel percorso Amazon S3 specificato in modo ricorsivo.

Puoi eseguire la transizione tra due classi di archiviazione qualsiasi. Per le classi di archiviazione `GLACIER` e `DEEP_ARCHIVE`, puoi passare a queste classi. Tuttavia, dovresti utilizzare un `S3 RESTORE` per eseguire la transizione dalle classi di archiviazione `GLACIER` a `DEEP_ARCHIVE`.

Se esegui processi ETL AWS Glue che leggono file o partizioni da Amazon S3, puoi escludere alcuni tipi di classe di archiviazione Amazon S3. Per ulteriori informazioni, consulta Esclusione delle classi di archiviazione Amazon S3.

- `s3_path`: il percorso in Amazon S3 dei file da convertire nel formato `s3://<bucket>/<prefix>/`
- `transition_to`: la classe di storage Amazon S3 in cui eseguire la transizione.
- `options`: opzioni per filtrare i file da eliminare e per la generazione di file manifesto.
  - `retentionPeriod`: specifica un periodo in numero di ore per la conservazione dei file. I file più recenti del periodo di conservazione vengono mantenuti. Impostato su 168 ore (7 giorni) per impostazione predefinita.
  - `partitionPredicate`: le partizioni che soddisfano questo predicato vengono trasferite. I file all'interno del periodo di conservazione in queste partizioni non vengono passati. Impostato su `""`: vuoto per impostazione predefinita.
  - `excludeStorageClasses`: i file con classe di storage nel set `excludeStorageClasses` non vengono passati. L'impostazione di default è `Set()`: un set vuoto.
  - `manifestFilePath`: un percorso facoltativo per la generazione di file manifesto. Tutti i file che sono stati passati correttamente vengono registrati in `Success.csv` e quelli che non sono riusciti in `Failed.csv`
  - `accountId`: l'ID account Amazon Web Services per eseguire la trasformazione di transizione. Obbligatorio per questa trasformazione.
  - `roleArn`: il ruolo AWS per eseguire la trasformazione di transizione. Obbligatorio per questa trasformazione.
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale). Utilizzato nel percorso del file manifest.

Example

```
glueContext.transition_s3_path("s3://bucket/prefix/", "STANDARD_IA",
 {"retentionPeriod": 1, "excludeStorageClasses": ["STANDARD_IA"],
 "manifestFilePath": "s3://bucketmanifest/", "accountId": "12345678901", "roleArn":
 "arn:aws:iam::123456789012:user/example-username"})
```

Estrazione in corso

- extract_jdbc_conf

extract_jdbc_conf

`extract_jdbc_conf(connection_name, catalog_id = None)`

Restituisce un `dict` con chiavi con le proprietà di configurazione dall'oggetto di connessione AWS Glue nel catalogo dati.

- `user`: il nome utente del database.
- `password`: la password del database.
- `vendor`: specifica un fornitore (`mysql`, `postgresql`, `oracle`, `sqlserver` e così via).
- `enforceSSL`: una stringa booleana che indica se è necessaria una connessione sicura.
- `customJDBCCert`: utilizza un certificato client specifico dal percorso Amazon S3 indicato.
- `skipCustomJDBCCertValidation`: una stringa booleana che indica se `customJDBCCert` deve essere convalidato da una CA.
- `customJDBCCertString`: informazioni aggiuntive sul certificato personalizzato, specifico per il tipo di driver.
- `url` (obsoleto): l'URL JDBC con solo protocollo, server e porta.
- `fullUrl`: l'URL JDBC immesso al momento della creazione della connessione (disponibile in AWS Glueversione 3.0 o successive).

Esempio di recupero delle configurazioni JDBC:

```
jdbc_conf = glueContext.extract_jdbc_conf(connection_name="your_glue_connection_name")
print(jdbc_conf)
>>> {'enforceSSL': 'false', 'skipCustomJDBCCertValidation': 'false', 'url':
 'jdbc:mysql://myserver:3306', 'fullUrl': 'jdbc:mysql://myserver:3306/mydb',
 'customJDBCCertString': '', 'user': 'admin', 'customJDBCCert': '', 'password': '1234',
 'vendor': 'mysql'}
```

Transazioni

- start_transaction
- commit_transaction
- cancel_transaction

start_transaction

`start_transaction(read_only)`

Avvia una nuova transazione. Chiama internamente l'API Lake Formation startTransaction.

- `read_only`: (booleano) indica se questa transazione debba essere di sola lettura o lettura e scrittura. Le scritture effettuate utilizzando un ID transazione di sola lettura verranno rifiutate. Il commit delle transazioni di sola lettura non deve essere eseguito.

Restituisce l'ID transazione.

commit_transaction

`commit_transaction(transaction_id, wait_for_commit = True)`

Tenta di eseguire il commit della transazione specificata. `commit_transaction` può restituire prima che la transazione abbia terminato il commit. Chiama internamente l'API Lake Formation commitTransaction.

- `transaction_id` : (stringa) la transazione di cui eseguire il commit.
- `wait_for_commit`: (booleano) determina se il `commit_transaction` restituisce immediatamente. Il valore di default è true. Se false, `commit_transaction` effettua il polling e aspetta che sia stato eseguito il commit della transazione. Il tempo di attesa è limitato a 1 minuto utilizzando il backoff esponenziale con un massimo di 6 tentativi.

Restituisce un valore booleano per indicare se il commit sia stato eseguito o meno.

cancel_transaction

`cancel_transaction(transaction_id)`

Tenta di annullare la transazione specificata. Restituisce un'eccezione `TransactionCommittedException` se è stato precedentemente eseguito il commit della transazione. Chiama internamente l'API Lake Formation CancelTransaction.

- `transaction_id`: (stringa) la transazione da annullare.

Scrittura

- getSink
- write_dynamic_frame_from_options
- write_from_options
- write_dynamic_frame_from_catalog
- write_data_frame_from_catalog
- write_dynamic_frame_from_jdbc_conf
- write_from_jdbc_conf

getSink

`getSink(connection_type, format = None, transformation_ctx = "", **options)`

Ottiene un oggetto `DataSink` che può essere utilizzato per scrivere `DynamicFrames` su fonti esterne. Verifica prima il `format` SparkSQL per essere certo di ricevere il sink previsto.

- `connection_type`: il tipo di connessione da utilizzare, come Amazon S3, Amazon Redshift e JDBC. I valori validi sono `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` e `oracle`.
- `format`: il formato SparkSQL da utilizzare (opzionale).
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `options`: raccolta di coppie nome-valore utilizzate per specificare le opzioni di connessione. Alcuni dei valori possibili sono:
  - `user` e `password`: per l'autorizzazione
  - `url`: l'endpoint per il archivio dati
  - `dbtable`: il nome della tabella di destinazione
  - `bulkSize`: il grado di parallelismo per le operazioni di inserimento

Le opzioni che è possibile specificare dipendono dal tipo di connessione. Per ulteriori valori ed esempi, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.

Esempio:

```
>>> data_sink = context.getSink("s3")
>>> data_sink.setFormat("json"),
```

```
>>> data_sink.writeFrame(myFrame)
```

write_dynamic_frame_from_options

`write_dynamic_frame_from_options(frame, connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")`

Legge e restituisce un `DynamicFrame` usando la connessione e il formato specificati.

- `frame`: il `DynamicFrame` da scrivere.
- `connection_type`: il tipo di connessione, come Amazon S3, Amazon Redshift e JDBC. I valori validi sono `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` e `oracle`.
- `connection_options`: opzioni di connessione, come tabella di database e percorso (opzionale). Per un `connection_type` di `s3` è definito un percorso Amazon S3.

  ```
  connection_options = {"path": "s3://aws-glue-target/temp"}
  ```

  Per le connessioni JDBC, diverse proprietà devono essere definite. Il nome del database deve fare parte dell'URL. Puoi opzionalmente essere incluso nelle opzioni di connessione.

  > **Warning**
  >
  > Si consiglia di non archiviare le password nello script. Valuta la possibilità di utilizzare `boto3` per recuperarle da AWS Secrets Manager o da Catalogo dati AWS Glue.

  ```
  connection_options = {"url": "jdbc-url/database", "user": "username",
   "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
  ```

  La proprietà `dbtable` è il nome della tabella JDBC. Per i archivi dati JDBC che supportano schemi all'interno di un database, specifica `schema.table-name`. Se non viene fornito alcuno schema, viene usato lo schema "pubblico" predefinito.

  Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.

- `format`: una specifica del formato (facoltativa). Viene usata per una connessione Amazon S3 o AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `format_options`: opzioni di formato per il formato specificato. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `transformation_ctx`: un contesto di trasformazione da usare (opzionale).

write_from_options

`write_from_options(frame_or_dfc, connection_type, connection_options={}, format={}, format_options={}, transformation_ctx = "")`

Scrive e restituisce un `DynamicFrame` o una `DynamicFrameCollection` creati con la connessione e le informazioni di formattazione specificati.

- `frame_or_dfc`: il `DynamicFrame` o la `DynamicFrameCollection` per scrivere.
- `connection_type`: il tipo di connessione, come Amazon S3, Amazon Redshift e JDBC. I valori validi sono `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` e `oracle`.
- `connection_options`: opzioni di connessione, come tabella di database e percorso (opzionale). Per un `connection_type` di `s3` è definito un percorso Amazon S3.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Per le connessioni JDBC, diverse proprietà devono essere definite. Il nome del database deve fare parte dell'URL. Puoi opzionalmente essere incluso nelle opzioni di connessione.

> **Warning**
>
> Si consiglia di non archiviare le password nello script. Valuta la possibilità di utilizzare `boto3` per recuperarle da AWS Secrets Manager o da Catalogo dati AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

La proprietà `dbtable` è il nome della tabella JDBC. Per i archivi dati JDBC che supportano schemi all'interno di un database, specifica `schema.table-name`. Se non viene fornito alcuno schema, viene usato lo schema "pubblico" predefinito.

Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.

- `format`: una specifica del formato (facoltativa). Viene usata per una connessione Amazon S3 o AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `format_options`: opzioni di formato per il formato specificato. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `transformation_ctx`: un contesto di trasformazione da usare (opzionale).

write_dynamic_frame_from_catalog

`write_dynamic_frame_from_catalog(frame, database, table_name, redshift_tmp_dir, transformation_ctx = "", additional_options = {}, catalog_id = None)`

Scrive e restituisce un `DynamicFrame` utilizzando un database del catalogo dati e una tabella.

- `frame`: il `DynamicFrame` da scrivere.
- `Database`: il database del catalogo dati che contiene la tabella.
- `table_name`: il nome della tabella del catalogo dati associata alla destinazione.
- `redshift_tmp_dir`: una directory temporanea Amazon Redshift da usare (opzionale).
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `additional_options`: una raccolta di coppie nome/valore opzionali.
- `catalog_id`: l'ID catalogo (ID account) del catalogo dati a cui si accede. Se Nessuno, viene utilizzato l'ID account predefinito del chiamante.

write_data_frame_from_catalog

`write_data_frame_from_catalog(frame, database, table_name, redshift_tmp_dir, transformation_ctx = "", additional_options = {}, catalog_id = None)`

Scrive e restituisce un `DataFrame` utilizzando un database del catalogo dati e una tabella. Questo metodo supporta la scrittura nei formati di data lake (Hudi, Iceberg e Delta Lake). Per ulteriori informazioni, consulta Utilizzo di framework data lake con processi ETL di AWS Glue.

- `frame`: il `DataFrame` da scrivere.
- `Database`: il database del catalogo dati che contiene la tabella.
- `table_name`: il nome della tabella del catalogo dati associata alla destinazione.
- `redshift_tmp_dir`: una directory temporanea Amazon Redshift da usare (opzionale).
- `transformation_ctx`: il contesto di trasformazione da usare (opzionale).
- `additional_options`: una raccolta di coppie nome/valore opzionali.
  - `useSparkDataSink`: se impostato su true, forza AWS Glue a utilizzare l'API nativa di Spark Data Sink per scrivere sulla tabella. Quando abiliti questa opzione, puoi anche aggiungere una qualsiasi delle opzioni Spark Data Source a `additional_options` a seconda delle necessità. AWS Glue passa queste opzioni direttamente allo scrittore Spark.
- `catalog_id`: l'ID catalogo (ID account) del catalogo dati a cui si accede. Se non specifichi un valore, verrà utilizzato l'ID account predefinito del chiamante.

Limitazioni

Quando utilizzi l'opzione `useSparkDataSink` considera le seguenti limitazioni:

- L'opzione `enableUpdateCatalog` non è supportata quando si utilizza l'opzione `useSparkDataSink`.

Esempio: scrittura su una tabella Hudi utilizzando lo scrittore Spark Data Source

```
hudi_options = {
    'useSparkDataSink': True,
    'hoodie.table.name': <table_name>,
    'hoodie.datasource.write.storage.type': 'COPY_ON_WRITE',
    'hoodie.datasource.write.recordkey.field': 'product_id',
```

```
    'hoodie.datasource.write.table.name': <table_name>,
    'hoodie.datasource.write.operation': 'upsert',
    'hoodie.datasource.write.precombine.field': 'updated_at',
    'hoodie.datasource.write.hive_style_partitioning': 'true',
    'hoodie.upsert.shuffle.parallelism': 2,
    'hoodie.insert.shuffle.parallelism': 2,
    'hoodie.datasource.hive_sync.enable': 'true',
    'hoodie.datasource.hive_sync.database': <database_name>,
    'hoodie.datasource.hive_sync.table': <table_name>,
    'hoodie.datasource.hive_sync.use_jdbc': 'false',
    'hoodie.datasource.hive_sync.mode': 'hms'}

glueContext.write_data_frame.from_catalog(
    frame = <df_product_inserts>,
    database = <database_name>,
    table_name = <table_name>,
    additional_options = hudi_options
)
```

write_dynamic_frame_from_jdbc_conf

`write_dynamic_frame_from_jdbc_conf(frame, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx = "", catalog_id = None)`

Legge e restituisce un `DynamicFrame` usando le informazioni sulla connessione JDBC specificate.

- `frame`: il `DynamicFrame` da scrivere.
- `catalog_connection`: una connessione del catalogo da utilizzare.
- `connection_options`: opzioni di connessione, come tabella di database e percorso (opzionale). Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.
- `redshift_tmp_dir`: una directory temporanea Amazon Redshift da usare (opzionale).
- `transformation_ctx`: un contesto di trasformazione da usare (opzionale).
- `catalog_id`: l'ID catalogo (ID account) del catalogo dati a cui si accede. Se Nessuno, viene utilizzato l'ID account predefinito del chiamante.

write_from_jdbc_conf

`write_from_jdbc_conf(frame_or_dfc, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx = "", catalog_id = None)`

Legge e restituisce un `DynamicFrame` o una `DynamicFrameCollection` usando le informazioni sulla connessione JDBC specificate.

- `frame_or_dfc`: il `DynamicFrame` o la `DynamicFrameCollection` per scrivere.
- `catalog_connection`: una connessione del catalogo da utilizzare.
- `connection_options`: opzioni di connessione, come tabella di database e percorso (opzionale). Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark.
- `redshift_tmp_dir`: una directory temporanea Amazon Redshift da usare (opzionale).
- `transformation_ctx`: un contesto di trasformazione da usare (opzionale).
- `catalog_id`: l'ID catalogo (ID account) del catalogo dati a cui si accede. Se Nessuno, viene utilizzato l'ID account predefinito del chiamante.

AWS Glue PySpark trasforma il riferimento

AWS Glue fornisce le seguenti trasformazioni integrate che è possibile utilizzare nelle operazioni PySpark ETL. I dati passano da una trasformazione all'altra in una struttura di dati chiamata a DynamicFrame, che è un'estensione di Apache Spark SQL. `DataFrame` `DynamicFrame` contiene i tuoi dati e il suo schema di riferimento per elaborare i dati.

La maggior parte di queste trasformazioni esiste anche come metodi della classe `DynamicFrame`. Per ulteriori informazioni, consulta DynamicFrame trasformazioni.

- Classe di base GlueTransform
- Classe ApplyMapping
- Classe DropFields
- Classe DropNullFields
- Classe ErrorsAsDynamicFrame
- Classe EvaluateDataQuality
- Classe FillMissingValues
- Classe filtro

- Classe FindIncrementalMatches
- Classe FindMatches
- Classe FlatMap
- Classe join
- Classe mappatura
- Classe MapToCollection
- mergeDynamicFrame
- Classe relazionalizzazione
- Classe RenameField
- Classe ResolveChoice
- Classe SelectFields
- Classe SelectFromCollection
- Classe Simplify_DDB_JSON
- Classe Spigot
- Classe SplitFields
- Classe SplitRows
- Classe unbox
- Classe UnnestFrame

Classe di base GlueTransform

La classe di base che tutte le classi `awsglue.transforms` ereditano.

Tutte le classi definiscono un `__call__` metodo. Sostituiscono i metodi della classe `GlueTransform` elencati nelle seguenti sezioni, oppure sono denominate utilizzando il nome della classe per impostazione predefinita.

Metodi

- apply(cls, *args, **kwargs)
- name(cls)
- describeArgs(cls)
- describeReturn(cls)
- describeTransform(cls)

- describeErrors(cls)
- describe(cls)

apply(cls, *args, **kwargs)

Applica la trasformazione chiamando la classe di trasformazione e restituisce il risultato.

- `cls`: l'oggetto `self` della classe.

name(cls)

Restituisce il nome della classe di trasformazione derivata.

- `cls`: l'oggetto `self` della classe.

describeArgs(cls)

- `cls`: l'oggetto `self` della classe.

Restituisce un elenco di dizionari, ciascuno corrispondente a un argomento denominato, nel formato seguente:

```
[
  {
    "name": "(name of argument)",
    "type": "(type of argument)",
    "description": "(description of argument)",
    "optional": "(Boolean, True if the argument is optional)",
    "defaultValue": "(Default value string, or None)(String; the default value, or
 None)"
  },
...
]
```

Solleva un'eccezione `NotImplementedError` quando viene chiamato in una trasformazione derivata dove non è stato implementato.

describeReturn(cls)

- `cls`: l'oggetto `self` della classe.

Restituisce un dizionario con informazioni sul tipo di restituzione, nel formato seguente:

```
{
  "type": "(return type)",
  "description": "(description of output)"
}
```

Solleva un'eccezione `NotImplementedError` quando viene chiamato in una trasformazione derivata dove non è stato implementato.

describeTransform(cls)

Restituisce una stringa che descrive la trasformazione.

- `cls`: l'oggetto `self` della classe.

Solleva un'eccezione `NotImplementedError` quando viene chiamato in una trasformazione derivata dove non è stato implementato.

describeErrors(cls)

- `cls`: l'oggetto `self` della classe.

Restituisce un elenco di dizionari, ognuno dei quali descrive una possibile eccezione generata da questa trasformazione, nel formato seguente:

```
[
  {
    "type": "(type of error)",
    "description": "(description of error)"
  },
...
]
```

describe(cls)

- `cls`: l'oggetto `self` della classe.

Restituisce un oggetto con il seguente formato:

```
{
  "transform" : {
    "name" : cls.name( ),
    "args" : cls.describeArgs( ),
    "returns" : cls.describeReturn( ),
    "raises" : cls.describeErrors( ),
    "location" : "internal"
  }
}
```

Classe ApplyMapping

Applica una mappatura in un `DynamicFrame`.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.apply_mapping()` per applicare una mappatura in un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usa apply_mapping per rinominare campi e modificare tipi di campo.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame, mappings, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Applica una mappatura dichiarativa a un `DynamicFrame` specificato.

- `frame`: il `DynamicFrame` in cui applicare la mappatura (obbligatorio).
- `mappings`: un elenco di tuple di mappatura (obbligatorio). Ognuna è costituito da: colonna di origine, tipo di origine, colonna di destinazione, tipo di destinazione.

Se la colonna di origine include un punto "`.`" nel nome, deve essere racchiuso tra apici inversi "```". Ad esempio, per mappare `this.old.name` (stringa) a `thisNewName`, devi utilizzare la tupla seguente:

```
("`this.old.name`", "string", "thisNewName", "string")
```

- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (facoltativo).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

Restituisce solo i campi del `DynamicFrame` specificato nelle tuple di "mappatura".

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe DropFields

Elimina i campi all'interno di un `DynamicFrame`.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.drop_fields()` per eliminare i campi da un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: utilizza drop_fields per rimuovere campi da un `DynamicFrame`.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__ (frame, percorsi, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Elimina i nodi all'interno di un `DynamicFrame`.

- `frame`: il `DynamicFrame` in cui rimuovere i nodi (obbligatorio).
- `paths`: un elenco di percorsi completi dei nodi da rilasciare (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

Restituisce un nuovo `DynamicFrame` senza i campi specificati.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe DropNullFields

Elimina tutti i campi nulli in un `DynamicFrame` il cui tipo è `NullType`. Questi sono campi con valori mancanti o nulli in ogni record nel set di dati `DynamicFrame`.

Esempio

Questo esempio usa `DropNullFields` per crearne una nuovo `DynamicFrame` dove sono stati rimossi i campi di tipo `NullType`. Per dimostrare `DropNullFields`, aggiungiamo una nuova colonna `empty_column` di tipo null al set di dati `persons` già caricato.

> **Note**
>
> Per accedere al set di dati utilizzato in questo esempio, consulta Esempio di codice: unione e relazioni dei dati e segui le istruzioni in Fase 1: esecuzione del crawling sui dati nel bucket Amazon S3.

```
# Example: Use DropNullFields to create a new DynamicFrame without NullType fields

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from pyspark.sql.functions import lit
from pyspark.sql.types import NullType
from awsglue.dynamicframe import DynamicFrame
from awsglue.transforms import DropNullFields

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create DynamicFrame
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Add new column "empty_column" with NullType
persons_with_nulls = persons.toDF().withColumn("empty_column",
 lit(None).cast(NullType()))
persons_with_nulls_dyf = DynamicFrame.fromDF(persons_with_nulls, glueContext,
 "persons_with_nulls")
print("Schema for the persons_with_nulls_dyf DynamicFrame:")
persons_with_nulls_dyf.printSchema()

# Remove the NullType field
persons_no_nulls = DropNullFields.apply(persons_with_nulls_dyf)
print("Schema for the persons_no_nulls DynamicFrame:")
persons_no_nulls.printSchema()
```

Output

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the persons_with_nulls_dyf DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
```

```
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
|-- empty_column: null

null_fields ['empty_column']
Schema for the persons_no_nulls DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
```

```
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Elimina tutti i campi nulli in un `DynamicFrame` il cui tipo è `NullType`. Questi sono campi con valori mancanti o nulli in ogni record nel set di dati `DynamicFrame`.

- `frame`: il `DynamicFrame` in cui rimuovere i campi nulli (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

Restituisce un nuovo `DynamicFrame` senza campi nulli.

apply(cls, *args, **kwargs)

- `cls`: cls

name(cls)

- `cls`: cls

describeArgs(cls)

- `cls`: cls

describeReturn(cls)

- `cls`: cls

describeTransform(cls)

- `cls`: cls

describeErrors(cls)

- `cls`: cls

describe(cls)

- `cls`: cls

Classe ErrorsAsDynamicFrame

Restituisce un `DynamicFrame` contenente record nidificati per gli errori che si sono verificati durante la creazione del `DynamicFrame` di origine.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.errorsAsDynamicFrame()` per recuperare e visualizzare i record degli errori. Per visualizzare un esempio di codice, consulta Esempio: utilizzare errorsAsDynamic Frame per visualizzare i record di errori.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame)

Restituisce un `DynamicFrame` contenente record nidificati degli errori correlati al `DynamicFrame` di origine.

- `frame`: il `DynamicFrame` di origine (obbligatorio).

apply(cls, *args, **kwargs)

- `cls`: cls

name(cls)

- `cls`: cls

describeArgs(cls)

- `cls`: cls

describeReturn(cls)

- `cls`: cls

describeTransform(cls)

- `cls`: cls

describeErrors(cls)

- `cls`: cls

describe(cls)

- `cls`: cls

Classe EvaluateDataQuality

Valuta un set di regole di qualità dei dati rispetto ai dati in un `DynamicFrame` e restituisce un nuovo `DynamicFrame` con i risultati della valutazione.

Esempio

Il seguente codice di esempio dimostra come valutare la qualità dei dati per un `DynamicFrame` e quindi visualizzare i risultati.

```
from awsglue.transforms import *
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsgluedq.transforms import EvaluateDataQuality

#Create Glue context
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Define DynamicFrame
legislatorsAreas = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="areas_json")

# Create data quality ruleset
```

```
ruleset = """Rules = [ColumnExists "id", IsComplete "id"]"""

# Evaluate data quality
dqResults = EvaluateDataQuality.apply(
    frame=legislatorsAreas,
    ruleset=ruleset,
    publishing_options={
        "dataQualityEvaluationContext": "legislatorsAreas",
        "enableDataQualityCloudWatchMetrics": True,
        "enableDataQualityResultsPublishing": True,
        "resultsS3Prefix": "DOC-EXAMPLE-BUCKET1",
    },
)


# Inspect data quality results
dqResults.printSchema()
dqResults.toDF().show()
```

Output

```
root
|-- Rule: string
|-- Outcome: string
|-- FailureReason: string
|-- EvaluatedMetrics: map
|    |-- keyType: string
|    |-- valueType: double


+----------------------+-------+-------------+--------------------------------------+
|Rule                  |Outcome|FailureReason|EvaluatedMetrics                      |
+----------------------+-------+-------------+--------------------------------------+
|ColumnExists "id"     |Passed |null         |{}                                    |
|IsComplete "id"       |Passed |null         |{Column.first_name.Completeness -> 1.0}|
+----------------------+-------+-------------+--------------------------------------+
```

Metodi

- __call__
- apply
- name

- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, ruleset, publishing_options = {})

- `frame`: il `DynamicFrame` di cui desideri valutare la qualità dei dati.
- `ruleset`: un set di regole del Data Quality Definition Language (DQDL) in formato stringa. Per ulteriori informazioni su DQDL, consulta la guida di Riferimento a Data Quality Definition Language (DQDL).
- `publishing_options`: un dizionario che specifica le seguenti opzioni per la pubblicazione dei risultati e dei parametri di valutazione:
  - `dataQualityEvaluationContext`: una stringa che specifica lo spazio dei nomi in cui AWS Glue deve pubblicare i parametri Amazon CloudWatch e i risultati della qualità dei dati. I parametri aggregati vengono visualizzati in CloudWatch, mentre i risultati completi vengono visualizzati nell'interfaccia di AWS Glue Studio.
    - Campo obbligatorio: no
    - Valore predefinito: `default_context`
  - `enableDataQualityCloudWatchMetrics`: specifica se i risultati della valutazione della qualità dei dati devono essere pubblicati su CloudWatch. Uno spazio dei nomi per i parametri viene specificato utilizzando l'opzione `dataQualityEvaluationContext`.
    - Campo obbligatorio: no
    - Valore predefinito: False
  - `enableDataQualityResultsPublishing`: specifica se i risultati della qualità dei dati devono essere visibili nella scheda Data Quality (Qualità dei dati) nell'interfaccia di AWS Glue Studio.
    - Campo obbligatorio: no
    - Valore predefinito: true
  - `resultsS3Prefix`: specifica la posizione Amazon S3 in cui AWS Glue può scrivere i risultati della valutazione della qualità dei dati.
    - Campo obbligatorio: no
    - Valore predefinito: "" (stringa vuota)

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe FillMissingValues

La classe `FillMissingValues` individua i valori null e stringhe vuote in un `DynamicFrame` specificato e utilizza metodi di machine learning, come la regressione lineare e la foresta casuale, per prevedere i valori mancanti. Il processo ETL utilizza i valori nel set di dati di input per addestrare il modello di machine learning, che prevede quindi quali devono essere i valori mancanti.

> **Tip**
>
> Se si utilizzano set di dati incrementali, ogni set incrementale viene utilizzato come dati di addestramento per il modello di machine learning, pertanto i risultati potrebbero non essere molto accurati.

Per l'importazione:

```
from awsglueml.transforms import FillMissingValues
```

Metodi

- Applica

apply(frame, missing_values_column, output_column ="", transformation_ctx ="", info ="", stageThreshold = 0, totalThreshold = 0)

Riempie i valori mancanti di un frame dinamico in una colonna specificata e restituisce un frame dinamico con stime in una nuova colonna. Per le righe senza valori mancanti, il valore della colonna specificato viene duplicato nella nuova colonna.

- `frame` il `DynamicFrame` in cui inserire i valori mancanti. Campo obbligatorio.
- `missing_values_column`: la colonna contenente valori mancanti (valori `null` e stringhe vuote). Campo obbligatorio.
- `output_column`: il nome della nuova colonna che conterrà i valori stimati per tutte le righe il cui valore era mancante. Facoltativo; il valore di default è il nome di `missing_values_column` con suffisso formato da `"_filled"`.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (opzionale; il numero predefinito è zero).
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (opzionale; il numero predefinito è zero).

Restituisce un nuovo `DynamicFrame` con una colonna aggiuntiva che contiene stime per le righe con valori mancanti e il valore attuale per le altre righe.

Classe filtro

Crea un nuovo `DynamicFrame` contenente record dall'input `DynamicFrame` che soddisfano una funzione predicato specificata.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.filter()` per filtrare i record in un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usa il filtro per ottenere una selezione filtrata di campi.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0))

Restituisce un nuovo `DynamicFrame` creato selezionando i record dall'input `DynamicFrame` che soddisfano una funzione predicato specificata.

- `frame`: l'origine `DynamicFrame` da applicare alla funzione filtro specificata (campo obbligatorio).
- `f`: la funzione di predicato da applicare a ciascun `DynamicRecord` nella `DynamicFrame`. La funzione deve assumere un `DynamicRecord` come argomento e restituire True se il `DynamicRecord` soddisfa i requisiti del filtro, altrimenti False (obbligatorio).

  Un `DynamicRecord` rappresenta un record logico all'interno di un `DynamicFrame`. È simile a una riga in un `DataFrame` Spark, con la differenza che è autodescrittivo e può essere utilizzato per dati non conformi a uno schema fisso.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (facoltativo).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.

- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe FindIncrementalMatches

Identifica i record corrispondenti nel `DynamicFrame` esistente e incrementale e crea un nuovo `DynamicFrame` con un identificatore univoco assegnato a ciascun gruppo di record corrispondenti.

Per l'importazione:

```
from awsglueml.transforms import FindIncrementalMatches
```

Metodi

- Applica

apply(existingFrame, incrementalFrame, transformId, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0, enforcedMatches = None, computeMatchConfidenceScores = 0)

Identifica i record corrispondenti nel `DynamicFrame` di input e crea un nuovo `DynamicFrame` con un identificatore univoco assegnato a ciascun gruppo di record corrispondenti.

- `existingFrame`: il `DynamicFrame` attuale e pre-abbinato al quale applicare la trasformazione FindIncrementalMatches. Campo obbligatorio.
- `incrementalFrame`: il `DynamicFrame` incrementale al quale applicare la trasformazione FindIncrementalMatches in modo che corrisponda al `existingFrame`. Campo obbligatorio.
- `transformId`: l'ID univoco associato alla trasformazione FindIncrementalMatches da applicare ai record nel `DynamicFrames`. Campo obbligatorio.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni su statistiche/stato. Facoltativo.
- `info`: una stringa associata a errori nella trasformazione. Facoltativo.
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata. Facoltativo. Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata. Facoltativo. Il valore di default è zero.
- `enforcedMatches`: il `DynamicFrame` utilizzato per applicare le corrispondenze. Facoltativo. Il valore predefinito è None (Nessuno).
- `computeMatchConfidenceScores`: un valore booleano che indica se calcolare un punteggio di confidenza per ciascun gruppo di record corrispondenti. Facoltativo. Il valore di default è false.

Restituisce un nuovo `DynamicFrame` con un identificatore univoco assegnato a ciascun gruppo di record corrispondenti.

Classe FindMatches

Identifica i record corrispondenti nel `DynamicFrame` di input e crea un nuovo `DynamicFrame` con un identificatore univoco assegnato a ciascun gruppo di record corrispondenti.

Per l'importazione:

```
from awsglueml.transforms import FindMatches
```

Metodi

- Applica

apply(frame, transformId, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0, enforcedMatches = none, computeMatchConfidenceScores = 0)

Identifica i record corrispondenti nel `DynamicFrame` di input e crea un nuovo `DynamicFrame` con un identificatore univoco assegnato a ciascun gruppo di record corrispondenti.

- `frame`: il `DynamicFrame` per applicare la trasformazione FindMatches. Campo obbligatorio.
- `transformId`: l'ID univoco associato alla trasformazione FindMatches da applicare ai record nel `DynamicFrame`. Campo obbligatorio.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni su statistiche/stato. Facoltativo.
- `info`: una stringa associata a errori nella trasformazione. Facoltativo.
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata. Facoltativo. Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata. Facoltativo. Il valore di default è zero.
- `enforcedMatches`: il `DynamicFrame` utilizzato per applicare le corrispondenze. Facoltativo. Il valore predefinito è None (Nessuno).
- `computeMatchConfidenceScores`: un valore booleano che indica se calcolare un punteggio di confidenza per ciascun gruppo di record corrispondenti. Facoltativo. Il valore di default è false.

Restituisce un nuovo `DynamicFrame` con un identificatore univoco assegnato a ciascun gruppo di record corrispondenti.

Classe FlatMap

Applica una trasformazione a ogni `DynamicFrame` in una raccolta. I risultati non vengono appiattiti in un unico `DynamicFrame` ma conservati come una raccolta.

Esempi per FlatMap

Il seguente frammento di esempio mostra come utilizzare la trasformazione `ResolveChoice` su una raccolta di frame dinamici quando applicata a una `FlatMap`. I dati utilizzati per l'input sono nel JSON

situato all'indirizzo segnaposto `s3://bucket/path-for-data/sample.json` di Amazon S3 e contengono i seguenti dati.

Dati JSON di esempio


```
[{
    "firstname": "Arnav",
    "lastname": "Desai",
    "address": {
        "street": "6 Anyroad Avenue",
        "city": "London",
        "state": "England",
        "country": "UK"
    },
    "phone": 17235550101,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Independent Research",
        "Government Department of Examples"
    ]
},
{
    "firstname": "Mary",
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
        "state": "OK",
        "country": "US"
    },
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
        "Example.io"
    ]
},
{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
```

```
        "country": "CA"
    },
    "phone": 12175550181,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
    ]
}]
```

Example Applica ResolveChoice a una DynamicFrameCollection e mostra l'output.

```
#Read DynamicFrame
datasource = glueContext.create_dynamic_frame_from_options("s3", connection_options =
 {"paths":["s3://bucket/path/to/file/mysamplejson.json"]}, format="json")
datasource.printSchema()
datasource.show()

## Split to create a DynamicFrameCollection
split_frame=datasource.split_fields(["firstname","lastname","address"],"personal_info","busines
split_frame.keys()
print("---")

## Use FlatMap to run ResolveChoice
kwargs = {"choice": "cast:string"}
flat = FlatMap.apply(split_frame, ResolveChoice, frame_name="frame",
 transformation_ctx='tcx', **kwargs)
flat.keys()

##Select one of the DynamicFrames
personal_info = flat.select("personal_info")
personal_info.printSchema()
personal_info.show()
print("---")

business_info = flat.select("business_info")
business_info.printSchema()
business_info.show()
```

> **Important**
>
> Quando si chiama `FlatMap.apply`, il parametro `frame_name` deve essere `"frame"`. Nessun altro valore è attualmente accettato.

Output di esempio

```
root
|-- firstname: string
|-- lastname: string
|-- address: struct
|    |-- street: string
|    |-- city: string
|    |-- state: string
|    |-- country: string
|-- phone: long
|-- affiliations: array
|    |-- element: string
---
{
    "firstname": "Mary",
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
        "state": "OK",
        "country": "US"
    },
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
        "Example.io"
    ]
}

{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
        "country": "CA"
    },
    "phone": 12175550181,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
```

```
    ]
}
---
root
|-- firstname: string
|-- lastname: string
|-- address: struct
|    |-- street: string
|    |-- city: string
|    |-- state: string
|    |-- country: string

{
    "firstname": "Mary",
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
        "state": "OK",
        "country": "US"
    }
}

{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
        "country": "CA"
    }
}
---
root
|-- phone: long
|-- affiliations: array
|    |-- element: string

{
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
```

```
        "Example.io"
    ]
}

{
    "phone": 12175550181,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
    ]
}
```

Metodi

- __call__
- Applica
- Nome
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__ (dfc, BaseTransform, frame_name, transformation_ctx = "",** base_kwargs)

Si applica alla trasformazione per ogni `DynamicFrame` in una raccolta e appiattisce i risultati.

- `dfc`: la `DynamicFrameCollection` su cui applicare la classe flatmap (obbligatorio).
- `BaseTransform`: una trasformazione derivata da `GlueTransform` da applicare a ciascun membro della raccolta (obbligatorio).
- `frame_name`: Il nome dell'argomento a cui passare gli elementi della raccolta (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `base_kwargs`: argomenti per passare alla trasformazione di base (obbligatorio).

Restituisce una nuova `DynamicFrameCollection` creata applicando la trasformazione a ciascun `DynamicFrame` nella `DynamicFrameCollection` di origine.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe join

Esegue un equi join su due `DynamicFrames`.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.join()` per unire `DynamicFrames`. Per visualizzare un esempio di codice, consulta Esempio: usa join per combinare `DynamicFrames`.

Metodi

- __call__
- apply
- name

- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__ (frame1, frame2, keys1, keys2, transformation_ctx = "")

Esegue un equi join su due `DynamicFrames`.

- `frame1`: il primo `DynamicFrame` da unire (obbligatorio).
- `frame2`: il secondo `DynamicFrame` da unire (obbligatorio).
- `keys1`: le chiavi da unire per il primo frame (obbligatorio).
- `keys2`: le chiavi da unire per il secondo frame (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).

Restituisce un nuovo `DynamicFrame` che viene creato unendo i due `DynamicFrames`.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe mappatura

Crea un nuovo `DynamicFrame` applicando una funzione a tutti i record nell'input `DynamicFrame`.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.map()` per applicare una funzione a tutti i record in un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: utilizza la mappa per applicare una funzione a ogni record in un `DynamicFrame`.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame, f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

Restituisce un nuovo `DynamicFrame` ottenuto applicando la funzione specificata a tutti i `DynamicRecords` nel `DynamicFrame` originale.

- `frame`: il `DynamicFrame` originale a cui applicare la funzione di mappatura (obbligatorio).
- `f`: la funzione da applicare a tutti i `DynamicRecords` nel `DynamicFrame`. La funzione deve assumere un `DynamicRecord` come argomento e restituire un nuovo `DynamicRecord` prodotto dalla mappatura (obbligatorio).

Un `DynamicRecord` rappresenta un record logico all'interno di un `DynamicFrame`. È simile a una riga in un `DataFrame` Apache Spark, con la differenza che è autodescrittivo e può essere utilizzato per dati non conformi a uno schema fisso.

- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

Restituisce un nuovo `DynamicFrame` ottenuto applicando la funzione specificata a tutti i `DynamicRecords` nel `DynamicFrame` originale.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe MapToCollection

Applica una trasformazione a ogni `DynamicFrame` nella `DynamicFrameCollection` specificata.

Metodi

- __call__
- Applica
- Nome
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__ (dfc, BaseTransform, frame_name, transformation_ctx = "",** base_kwargs)

Applica una funzione di trasformazione a ogni `DynamicFrame` nella `DynamicFrameCollection` specificata.

- `dfc`: la `DynamicFrameCollection` a cui applicare la funzione di trasformazione (obbligatorio).
- `callable`: la funzione di trasformazione chiamabile da applicare a ciascun membro della raccolta (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).

Restituisce una nuova `DynamicFrameCollection` creata applicando la trasformazione a ciascun `DynamicFrame` nella `DynamicFrameCollection` di origine.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe relazionalizzazione

Livella uno schema nidificato in un `DynamicFrame` e trasforma le colonne della matrice tramite pivoting da un frame appiattito.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.relationalize()` per relazionare `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usa la relazionalizzazione per livellare uno schema annidato in un `DynamicFrame`.

Metodi

- __call__
- apply
- name

- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, staging_path=None, name='roottable', options=None, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Mette in relazione un `DynamicFrame` e produce un elenco di frame che vengono generati annullando l'annidamento di colonne nidificate e trasformando colonne della matrice mediante pivot. La colonna matrice trasformata mediante pivot può essere unita alla tabella root utilizzando la chiave di join generata durante la fase di annullamento dell'annidamento.

- `frame`: il `DynamicFrame` da mettere in relazione (obbligatorio).
- `staging_path`: il percorso in cui il metodo può archiviare le partizioni di tabelle trasformate mediante pivot in formato CSV (facoltativo). Le tabelle trasformate mediante pivot vengono rilette da questo percorso.
- `name`: il nome della tabella root (opzionale).
- `options`: un dizionario dei parametri opzionali. Attualmente inutilizzato.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe RenameField

Rinomina un nodo all'interno di un `DynamicFrame`.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.rename_field()` per rinominare un campo in un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usa rename_field per rinominare i campi in un `DynamicFrame`.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, old_name, new_name, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Rinomina un nodo all'interno di un `DynamicFrame`.

- `frame`: il `DynamicFrame` in cui rinominare un nodo (obbligatorio).
- `old_name`: il percorso completo del nodo da rinominare (obbligatorio).

  Se il nome precedente contiene dei punti, RenameField non funzionerà a meno che non sia racchiuso tra virgolette inversi (``). Ad esempio, per sostituire `this.old.name` con `thisNewName`, chiami RenameField come segue:

  ```
  newDyF = RenameField(oldDyF, "`this.old.name`", "thisNewName")
  ```

- `new_name`: il nuovo nome, incluso il percorso completo (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe ResolveChoice

Risolve un tipo di scelta all'interno di un `DynamicFrame`.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.resolveChoice()` per gestire campi che contengono più tipi in un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: utilizzare resolveChoice per gestire una colonna che contiene più tipi.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, specs = none, choice = "", transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Fornisce informazioni per risolvere tipi ambigui all'interno di un `DynamicFrame`. Restituisce il `DynamicFrame` risultante.

- `frame`: il `DynamicFrame` in cui risolvere il tipo di scelta (obbligatorio).
- `specs`: elenco di ambiguità specifiche da risolvere, ognuna sotto forma di tupla:`(path, action)`. Il valore `path` identifica un elemento ambiguo specifico e il valore `action` identifica la soluzione corrispondente.

  Può essere utilizzato solo uno dei parametri `spec` e `choice`. Se il parametro `spec` non è `None`, allora il parametro `choice` deve essere una stringa vuota. Viceversa, se `choice` non è una stringa vuota, allora il parametro `spec` deve essere `None`. Se non viene fornito alcun parametro, AWS Glue cerca di analizzare lo schema e di usarlo per risolvere le ambiguità.

  La parte `action` di una tupla `specs` può specificare una delle quattro strategie di risoluzione possibili:

  - `cast`: consente di specificare un tipo verso cui trasmettere (ad esempio, `cast:int`).
  - `make_cols`: risolve una potenziale ambiguità appiattendo i dati. Ad esempio, se `columnA` è un `int` o una `string`, la soluzione consiste nel produrre due colonne denominate `columnA_int` e `columnA_string` nel `DynamicFrame` risultante.
  - `make_struct`: risolve una potenziale ambiguità utilizzando una struttura per rappresentare i dati. Ad esempio, se i dati in una colonna sono un `int` o una `string`, utilizzando l'operazione `make_struct` viene prodotta una colonna di strutture nel `DynamicFrame` risultante, ognuna contenente sia un `int` che una `string`.
  - `project`: risolve un potenziale ambiguità conservando solo i valori di un tipo specificato nel `DynamicFrame` risultante. Ad esempio, se i dati in una colonna `ChoiceType` possono essere di tipo `int` o `string`, specificando un'operazione `project:string` si rimuovono i valori dal `DynamicFrame` risultante che non sono di tipo `string`.

  Se il `path` identifica un array, inserisci parentesi quadre vuote dopo il nome dell'array per evitare ambiguità. Ad esempio, supponiamo che tu stia lavorando con dati strutturati nel seguente modo:

  ```
  "myList": [
    { "price": 100.00 },
    { "price": "$100.00" }
  ]
  ```

  Puoi selezionare la versione numerica invece di quella di stringa del prezzo impostando il `path` su `"myList[].price"` e la `action` su `"cast:double"`.
- `choice`: l'operazione di risoluzione di default se il parametro `specs` è `None`. Se il parametro `specs` non è `None`, allora deve essere impostato solo su una stringa vuota.

Oltre alle operazioni elencate in precedenza per `specs`, questo argomento supporta anche l'operazione seguente:

  - `MATCH_CATALOG`: tenta di trasmettere ogni `ChoiceType` al tipo corrispondente nella tabella del catalogo specificata.
- `database`: il database AWS Glue Data Catalog da utilizzare con la scelta `MATCH_CATALOG` (obbligatorio per`MATCH_CATALOG`).
- `table_name`: il nome della tabella AWS Glue Data Catalog da utilizzare con l'operazione `MATCH_CATALOG` (obbligatorio per `MATCH_CATALOG`).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe SelectFields

La classe `SelectFields` crea un nuovo `DynamicFrame` da un `DynamicFrame` esistente e mantiene solo i campi specificati. `SelectFields` fornisce funzionalità simili a quelle di un'istruzione `SELECT` SQL.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.select_fields()` per selezionare i campi da `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usa select_fields per creare un nuovo `DynamicFrame` con i campi scelti.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__ (frame, percorsi, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Ottiene i campi (nodi) in un `DynamicFrame`.

- `frame`: il `DynamicFrame` in cui selezionare i campi (obbligatorio).
- `paths`: un elenco di percorsi completi ai campi da selezionare (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (facoltativo).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.

- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

Restituisce un nuovo `DynamicFrame` contenente solo i campi specificati.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe SelectFromCollection

Seleziona un `DynamicFrame` in una `DynamicFrameCollection`.

Esempio

Questo esempio utilizza `SelectFromCollection` per selezionare un `DynamicFrame` da una `DynamicFrameCollection`.

Set di dati di esempio

L'esempio seleziona due `DynamicFrames` da una `DynamicFrameCollection` chiamata `split_rows_collection`. Di seguito è riportato un elenco di chiavi in `split_rows_collection`.

```
dict_keys(['high', 'low'])
```

Esempio di codice

```
# Example: Use SelectFromCollection to select
# DynamicFrames from a DynamicFrameCollection

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import SelectFromCollection

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Select frames and inspect entries
frame_low = SelectFromCollection.apply(dfc=split_rows_collection, key="low")
frame_low.toDF().show()

frame_high = SelectFromCollection.apply(dfc=split_rows_collection, key="high")
frame_high.toDF().show()
```

Output

```
+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                    fax|           202-225-3307|
|  1|    1|                  phone|           202-225-5731|
|  2|    0|                    fax|           202-225-3307|
|  2|    1|                  phone|           202-225-5731|
|  3|    0|                    fax|           202-225-3307|
|  3|    1|                  phone|           202-225-5731|
|  4|    0|                    fax|           202-225-3307|
|  4|    1|                  phone|           202-225-5731|
|  5|    0|                    fax|           202-225-3307|
|  5|    1|                  phone|           202-225-5731|
|  6|    0|                    fax|           202-225-3307|
```

```
|  6|    1|                  phone|             202-225-5731|
|  7|    0|                    fax|             202-225-3307|
|  7|    1|                  phone|             202-225-5731|
|  8|    0|                    fax|             202-225-3307|
|  8|    1|                  phone|             202-225-5731|
|  9|    0|                    fax|             202-225-3307|
|  9|    1|                  phone|             202-225-5731|
| 10|    0|                    fax|             202-225-6328|
| 10|    1|                  phone|             202-225-4576|
+---+-----+----------------------+-------------------------+
only showing top 20 rows

+---+-----+----------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-------------------------+
| 11|    0|                    fax|             202-225-6328|
| 11|    1|                  phone|             202-225-4576|
| 11|    2|                twitter|           RepTrentFranks|
| 12|    0|                    fax|             202-225-6328|
| 12|    1|                  phone|             202-225-4576|
| 12|    2|                twitter|           RepTrentFranks|
| 13|    0|                    fax|             202-225-6328|
| 13|    1|                  phone|             202-225-4576|
| 13|    2|                twitter|           RepTrentFranks|
| 14|    0|                    fax|             202-225-6328|
| 14|    1|                  phone|             202-225-4576|
| 14|    2|                twitter|           RepTrentFranks|
| 15|    0|                    fax|             202-225-6328|
| 15|    1|                  phone|             202-225-4576|
| 15|    2|                twitter|           RepTrentFranks|
| 16|    0|                    fax|             202-225-6328|
| 16|    1|                  phone|             202-225-4576|
| 16|    2|                twitter|           RepTrentFranks|
| 17|    0|                    fax|             202-225-6328|
| 17|    1|                  phone|             202-225-4576|
+---+-----+----------------------+-------------------------+
only showing top 20 rows
```

Metodi

- __call__
- apply

- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (dfc, chiave, transformation_ctx = "")

Ottiene un `DynamicFrame` da una `DynamicFrameCollection`.

- `dfc`: la `DynamicFrameCollection` da cui il `DynamicFrame` deve essere selezionato (obbligatorio).
- `key`: la chiave del `DynamicFrame` da selezionare (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe Simplify_DDB_JSON

Semplifica le colonne annidate in un ambiente `DynamicFrame` che si trova specificamente nella struttura JSON di DynamoDB e ne restituisce una nuova semplificata. `DynamicFrame`

Esempio

Si consiglia di utilizzare il `DynamicFrame.simplify_ddb_json()` metodo per semplificare le colonne annidate in un `DynamicFrame` formato specifico nella struttura JSON di DynamoDB. Per visualizzare un esempio di codice, consulta Esempio: utilizza simply_ddb_json per richiamare un DynamoDB JSON simple.

Classe Spigot

Scrive record di esempio in una destinazione specificata per aiutarti a verificare le trasformazioni eseguite dal processo AWS Glue.

Esempio

Si consiglia di utilizzare il metodo `DynamicFrame.spigot()` per scrivere un sottoinsieme di record da un `DynamicFrame` a una destinazione specificata. Per visualizzare un esempio di codice, consulta Esempio: usa lo spigot per scrivere campi di esempio da `DynamicFrame` ad Amazon S3.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, path, options, transformation_ctx = "")

Scrive record di esempio in una destinazione specificata durante una trasformazione.

- `frame`: il `DynamicFrame` da sottoporre allo spigot (obbligatorio).
- `path`: il percorso della destinazione in cui scrivere (obbligatorio).
- `options`: coppie chiave-valore JSON che specificano opzioni (opzionale). L'opzione `"topk"` specifica che devono essere scritti i primi record k. L'opzione `"prob"` specifica la probabilità (sotto forma di valore decimale) di selezione di un dato record. Puoi usarlo per selezionare i record da scrivere.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply

name(cls)

Ereditato da `GlueTransform` name

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors

describe(cls)

Ereditato da `GlueTransform` describe

Classe SplitFields

Divide un `DynamicFrame` in due, in base ai campi specificati.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.split_fields()` per dividere i campi in un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usare split_fields per dividere i campi selezionati in un campo separato `DynamicFrame`.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, paths, name1 = none, name2 = none, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Divide uno o più campi in un `DynamicFrame` disattivato in un nuovo `DynamicFrame` e crea un altro nuovo `DynamicFrame` che contiene i campi che rimangono.

- `frame`: il `DynamicFrame` di origine da dividere in due nuovi elementi (obbligatorio).
- `paths`: un elenco di percorsi completi per campi da suddividere (obbligatorio).
- `name1`: il nome da assegnare al `DynamicFrame` che contiene i campi che devono essere separati (facoltativo). Se non viene fornito nessun nome, il nome del frame di origine viene utilizzato con "1" aggiunto.
- `name2`: il nome da assegnare al `DynamicFrame` che contiene i campi che rimangono una volta che i campi specificati vengono suddivisi (facoltativo). Se non viene fornito alcun nome, il nome del frame di origine viene utilizzato con "2" aggiunto.

- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe SplitRows

Crea una `DynamicFrameCollection` che contiene due `DynamicFrames`. Un `DynamicFrame` contiene solo le righe specificate da suddividere e l'altro con tutte le righe rimanenti.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.split_rows()` per dividere le righe in un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usare split_rows per dividere le righe in un `DynamicFrame`.

Metodi

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, comparison_dict, name1="frame1", name2="frame2", transformation_ctx = "", info = none, stageThreshold = 0, totalThreshold = 0)

Suddivide una o più righe di un `DynamicFrame` in un nuovo `DynamicFrame`.

- `frame`: il `DynamicFrame` di origine da dividere in due nuovi elementi (obbligatorio).
- `comparison_dict`: un dizionario in cui la chiave è il percorso completo verso una colonna e il valore è un altro dizionario per la mappatura di comparatori rispetto a valori con i quali vengono confrontati i valori di colonna. Ad esempio, `{"age": {">": 10, "<": 20}}` divide le righe in cui il valore di "age" (età) è compreso tra 10 e 20 (non inclusi) dalle righe dove "age" non è compreso in tale intervallo (obbligatorio).
- `name1`: il nome da assegnare al `DynamicFrame` che contiene le righe da dividere (opzionale).
- `name2`: il nome da assegnare al `DynamicFrame` che contiene le righe che rimangono dopo la divisione delle righe specificate (opzionale).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.

- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe unbox

Sottopone a conversione unboxing (riformatta) un campo stringa in un oggetto `DynamicFrame`.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.unbox()` per eseguire la conversione unboxing di un campo in un `DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usare unbox per decomprimere un campo di stringa in un campo struct.

Metodi

- __call__

- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, path, format, transformation_ctx = "", info="", stageThreshold=0, totalThreshold=0, **options)

Sottopone a conversione unboxing un campo stringa in un oggetto `DynamicFrame`.

- `frame`: il `DynamicFrame` nel quale sottoporre a conversione unboxing un campo (obbligatorio).
- `path`: il percorso completo del `StringNode` da cancellare (obbligatorio).
- `format`: una specifica del formato (facoltativa). Viene usata per una connessione Amazon S3 o AWS Glue che supporta più formati. Consulta Opzioni del formato dati per input e output in AWS Glue per Spark per informazioni sui formati supportati.
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.
- `separator`: un token di separazione (opzionale).
- `escaper`: un token di escape (opzionale).
- `skipFirst`: `True` se la prima riga di dati deve essere ignorata oppure `False` se non deve essere ignorata (opzionale).
- withSchema: una stringa che contiene lo schema per i dati da rimuovere (opzionale). Deve essere sempre creato utilizzando `StructType.json`.
- `withHeader`: `True` se i dati decompressi includono un'intestazione oppure `False` se non la includono (opzionale).

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Classe UnnestFrame

Annulla l'annidamento di un `DynamicFrame`, livella gli oggetti nidificati a elementi di primo livello e genera chiavi di join per oggetti di array.

Esempio

Ti consigliamo di utilizzare il metodo `DynamicFrame.unnest()` per livellare le strutture annidate in un`DynamicFrame`. Per visualizzare un esempio di codice, consulta Esempio: usare unnest per trasformare i campi annidati in campi di primo livello.

Metodi

- __call__
- apply
- name

- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, transformation_ctx = "", info="", stageThreshold=0, totalThreshold=0)

Annulla l'annidamento di un `DynamicFrame`, livella gli oggetti nidificati a elementi di primo livello e genera chiavi di join per oggetti di array.

- `frame`: il `DynamicFrame` per annullare l'annidamento (obbligatorio).
- `transformation_ctx`: una stringa univoca utilizzata per identificare informazioni sullo stato (opzionale).
- `info`: una stringa associata a errori nella trasformazione (opzionale).
- `stageThreshold`: il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo). Il valore di default è zero.
- `totalThreshold`: il numero massimo di errori che si possono verificare in totale prima che l'elaborazione venga arrestata (facoltativo). Il valore di default è zero.

apply(cls, *args, **kwargs)

Ereditato da `GlueTransform` apply.

name(cls)

Ereditato da `GlueTransform` name.

describeArgs(cls)

Ereditato da `GlueTransform` describeArgs.

describeReturn(cls)

Ereditato da `GlueTransform` describeReturn.

describeTransform(cls)

Ereditato da `GlueTransform` describeTransform.

describeErrors(cls)

Ereditato da `GlueTransform` describeErrors.

describe(cls)

Ereditato da `GlueTransform` describe.

Programmazione di script ETL AWS Glue in Scala

Puoi trovare codici di esempio Scala e le utilità per AWS Glue nel repository di esempio AWS Glue nel sito Web GitHub.

AWS Glue supporta un'estensione del dialetto Scala PySpark per lo scripting dei processi di estrazione, trasformazione e caricamento (ETL). Le sezioni seguenti descrivono come usare la libreria Scala AWS Glue e l'API AWS Glue negli script ETL e forniscono la documentazione di riferimento per la libreria.

Indice

Uso di Scala per programmare gli script ETL AWS Glue

È possibile generare automaticamente un programma di estrazione, trasformazione e caricamento (ETL) Scala usando la console AWS Glue ed effettuare le modifiche necessarie prima dell'assegnazione a un processo. In alternativa, è possibile scrivere il proprio programma da zero. Per altre informazioni, consulta Aggiunta di processi in AWS Glue. AWS Glue compila quindi il programma Scala nel server prima di eseguire il processo associato.

Per fare in modo che il programma venga compilato senza errori e venga eseguito correttamente, è molto importante caricarlo su un endpoint di sviluppo in una REPL (Read-Eval-Print Loop) o su un notebook Jupyter e testarlo prima di eseguirlo in un processo. Poiché il processo di compilazione viene effettuato sul server, non sarà possibile individuare chiaramente eventuali problemi.

Test di un programma Scala ETL in un notebook Jupyter su un endpoint di sviluppo

Per testare un programma Scala in un endpoint di sviluppo AWS Glue, configura l'endpoint di sviluppo come descritto in Aggiunta di un endpoint di sviluppo.

Connettilo quindi a un notebook Jupyter in esecuzione sul computer in locale o in remoto su un server notebook Amazon EC2. Per installare una versione locale di un notebook Jupyter, segui le istruzioni riportate alla pagina Tutorial: notebook Jupyter in JupyterLab.

L'unica differenza tra l'esecuzione del codice Scala e l'esecuzione del codice PySpark sul notebook è che è necessario iniziare ciascun paragrafo sul notebook con quanto segue:

```
%spark
```

In questo modo si impedisce che il server del notebook passi per impostazione predefinita al flavor PySpark dell'interprete Spark.

Test di un programma Scala ETL in una REPL Scala

Come testare un programma Scala in un endpoint di sviluppo usando un ciclo REPL Scala AWS Glue. Segui le istruzioni in Tutorial: utilizzo di un notebook SageMaker; alla fine del comando SSH-to-REPL, sostituisci `-t gluepyspark` con `-t glue-spark-shell`. In questo modo viene richiamato il ciclo REPL Scala AWS Glue.

Per chiudere la REPL quando si è terminato, digitare `sys.exit`.

Esempio di script Scala - Streaming ETL

Example

Lo script di esempio seguente si connette ad Amazon Kinesis Data Streams, utilizza uno schema del catalogo dati per analizzare un flusso dei dati, unisce il flusso a un set di dati statico su Amazon S3 e genera i risultati uniti in Amazon S3 in formato parquet.

```
// This script connects to an Amazon Kinesis stream, uses a schema from the data
 catalog to parse the stream,
```

```
// joins the stream to a static dataset on Amazon S3, and outputs the joined results to
 Amazon S3 in parquet format.
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import java.util.Calendar
import org.apache.spark.SparkContext
import org.apache.spark.sql.Dataset
import org.apache.spark.sql.Row
import org.apache.spark.sql.SaveMode
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions.from_json
import org.apache.spark.sql.streaming.Trigger
import scala.collection.JavaConverters._

object streamJoiner {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val sparkSession: SparkSession = glueContext.getSparkSession
    import sparkSession.implicits._
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val staticData = sparkSession.read          // read() returns type DataFrameReader
      .format("csv")
      .option("header", "true")
      .load("s3://awsexamplebucket-streaming-demo2/inputs/productsStatic.csv")  //
 load() returns a DataFrame

    val datasource0 = sparkSession.readStream   // readstream() returns type
 DataStreamReader
      .format("kinesis")
      .option("streamName", "stream-join-demo")
      .option("endpointUrl", "https://kinesis.us-east-1.amazonaws.com")
      .option("startingPosition", "TRIM_HORIZON")
      .load                                     // load() returns a DataFrame

    val selectfields1 = datasource0.select(from_json($"data".cast("string"),
 glueContext.getCatalogSchemaAsSparkSchema("stream-demos", "stream-join-demo2")) as
 "data").select("data.*")
```

```
    val datasink2 = selectfields1.writeStream.foreachBatch { (dataFrame: Dataset[Row],
 batchId: Long) => {   //foreachBatch() returns type DataStreamWriter
      val joined = dataFrame.join(staticData, "product_id")
      val year: Int = Calendar.getInstance().get(Calendar.YEAR)
      val month :Int = Calendar.getInstance().get(Calendar.MONTH) + 1
      val day: Int = Calendar.getInstance().get(Calendar.DATE)
      val hour: Int = Calendar.getInstance().get(Calendar.HOUR_OF_DAY)

      if (dataFrame.count() > 0) {
        joined.write                            // joined.write returns type
 DataFrameWriter
          .mode(SaveMode.Append)
          .format("parquet")
          .option("quote", " ")
          .save("s3://awsexamplebucket-streaming-demo2/output/" + "/year=" +
 "%04d".format(year) + "/month=" + "%02d".format(month) + "/day=" + "%02d".format(day)
 + "/hour=" + "%02d".format(hour) + "/")
      }
    }
    }  // end foreachBatch()
      .trigger(Trigger.ProcessingTime("100 seconds"))
      .option("checkpointLocation", "s3://awsexamplebucket-streaming-demo2/
checkpoint/")
      .start().awaitTermination()             // start() returns type StreamingQuery
    Job.commit()
  }
}
```

API nella libreria Scala AWS Glue

AWS Glue supporta un'estensione del dialetto Scala PySpark per lo scripting dei processi ETL (Extract, Transform and Load, estrazione, trasformazione e caricamento). Le sezioni seguenti descrivono le API nella libreria Scala AWS Glue.

com.amazonaws.services.glue

Il pacchetto com.amazonaws.services.glue nella libreria Scala AWS Glue contiene le API seguenti:

- ChoiceOption
- DataSink
- Proprietà DataSource
- DynamicFrame

- DynamicRecord
- GlueContext
- MappingSpec
- ResolveSpec

com.amazonaws.services.glue.ml

Il pacchetto com.amazonaws.services.glue.ml nella libreria Scala AWS Glue contiene le API seguenti:

- FillMissingValues
- FindIncrementalMatches
- FindMatches

com.amazonaws.services.glue.dq

Il pacchetto com.amazonaws.services.glue.dq nella libreria Scala AWS Glue contiene le seguenti API:

- EvaluateDataQuality

com.amazonaws.services.glue.types

Il pacchetto com.amazonaws.services.glue.types nella libreria Scala AWS Glue contiene le API seguenti:

- ArrayNode
- BinaryNode
- BooleanNode
- ByteNode
- DateNode
- DecimalNode
- DoubleNode
- DynamicNode

- FloatNode
- IntegerNode
- LongNode
- MapLikeNode
- MapNode
- NullNode
- ObjectNode
- ScalarNode
- ShortNode
- StringNode
- TimestampNode

com.amazonaws.services.glue.util

Il pacchetto com.amazonaws.services.glue.util nella libreria Scala AWS Glue contiene le API seguenti:

- GlueArgParser
- Processo

API Scala ChoiceOption di AWS Glue

Argomenti

- Proprietà ChoiceOption
- Oggetto ChoiceOption
- Case class ChoiceOptionWithResolver
- Case class MatchCatalogSchemaChoiceOption

Pacchetto: com.amazonaws.services.glue

Proprietà ChoiceOption

```
trait ChoiceOption extends Serializable
```

Oggetto ChoiceOption

ChoiceOption

```
object ChoiceOption
```

Una strategia generale per risolvere la scelta applicabile a tutti i nodi `ChoiceType` in un `DynamicFrame`.

- `val CAST`
- `val MAKE_COLS`
- `val MAKE_STRUCT`
- `val MATCH_CATALOG`
- `val PROJECT`

Applicazione di def

```
def apply(choice: String): ChoiceOption
```

Case class ChoiceOptionWithResolver

```
case class ChoiceOptionWithResolver(name: String, choiceResolver: ChoiceResolver)
 extends ChoiceOption {}
```

Case class MatchCatalogSchemaChoiceOption

```
case class MatchCatalogSchemaChoiceOption() extends ChoiceOption {}
```

Classe DataSink Abstract

Argomenti

- Def writeDynamicFrame
- Def pyWriteDynamicFrame
- Def writeDataFrame

Pacchetto: com.amazonaws.services.glue

```
abstract class DataSink
```

Writer analogo a `DataSource`. `DataSink` incapsula una destinazione e un formato in cui può essere scritto un oggetto `DynamicFrame`.

Def writeDynamicFrame

```
def writeDynamicFrame( frame : DynamicFrame,
                       callSite : CallSite = CallSite("Not provided", "")
                     ) : DynamicFrame
```

Def pyWriteDynamicFrame

```
def pyWriteDynamicFrame( frame : DynamicFrame,
                         site : String = "Not provided",
                         info : String = "" )
```

Def writeDataFrame

```
def writeDataFrame(frame: DataFrame,
                   glueContext: GlueContext,
                   callSite: CallSite = CallSite("Not provided", "")
                   ): DataFrame
```

Def pyWriteDataFrame

```
def pyWriteDataFrame(frame: DataFrame,
                     glueContext: GlueContext,
                     site: String = "Not provided",
                     info: String = ""
                     ): DataFrame
```

Def setCatalogInfo

```
def setCatalogInfo(catalogDatabase: String,
                   catalogTableName : String,
                   catalogId : String = "")
```

Def supportsFormat

```
def supportsFormat( format : String ) : Boolean
```

Def setFormat

```
def setFormat( format : String,
               options : JsonOptions
             ) : Unit
```

Def withFormat

```
def withFormat( format : String,
                options : JsonOptions = JsonOptions.empty
              ) : DataSink
```

Def setAccumulableSize

```
def setAccumulableSize( size : Int ) : Unit
```

Def getOutputErrorRecordsAccumulable

```
def getOutputErrorRecordsAccumulable : Accumulable[List[OutputError], OutputError]
```

Def errorsAsDynamicFrame

```
def errorsAsDynamicFrame : DynamicFrame
```

Oggetto DataSink

```
object DataSink
```

Def recordMetrics

```
def recordMetrics( frame : DynamicFrame,
                   ctxt : String
                 ) : DynamicFrame
```

Proprietà Scala DataSource in AWS Glue

Pacchetto: com.amazonaws.services.glue

Un'interfaccia di alto livello per la produzione di un `DynamicFrame`.

```
trait DataSource {

  def getDynamicFrame : DynamicFrame

  def getDynamicFrame( minPartitions : Int,
                       targetPartitions : Int
                     ) : DynamicFrame
  def getDataFrame : DataFrame

  /** @param num: the number of records for sampling.
    * @param options: optional parameters to control sampling behavior. Current
 available parameter for Amazon S3 sources in options:
    *  1. maxSamplePartitions: the maximum number of partitions the sampling will
 read.
    *  2. maxSampleFilesPerPartition: the maximum number of files the sampling will
 read in one partition.
```

```
    */
  def getSampleDynamicFrame(num:Int, options: JsonOptions = JsonOptions.empty):
 DynamicFrame

  def glueContext : GlueContext

  def setFormat( format : String,
                 options : String
               ) : Unit

  def setFormat( format : String,
                 options : JsonOptions
               ) : Unit

  def supportsFormat( format : String ) : Boolean

  def withFormat( format : String,
                  options : JsonOptions = JsonOptions.empty
                ) : DataSource
}
```

API Scala DynamicFrame di AWS Glue

Pacchetto: com.amazonaws.services.glue

Indice

- Def calcolato getSchemaIf
- Def isSchemaComputed
- Def javaToPython
- Def join
- Def mappa
- Def mergeDynamicFrames
- Def printSchema
- Def recomputeSchema
- Def relazionalizzazione
- Def renameField
- Def ripartizione
- Def resolveChoice
- Def schema
- Def selectField
- Def selectFields
- Def mostra
- Def semplifica DDBJSON
- Def spigot
- Def splitFields
- Def splitRows
- Def stageErrorsCount
- Def toDF
- Def unbox
- Def unnest
- Def unnestDDBJson
- Def withFrameSchema
- Def withName
- Def withTransformationContext

- L'oggetto DynamicFrame
  - Applicazione di def

- Def emptyDynamicFrame
- Def fromPythonRDD
- Def ignoreErrors
- Def inlineErrors
- Def newFrameWithErrors

AWS Glue DynamicFrameClasse Scala

Pacchetto: com.amazonaws.services.glue

```
class DynamicFrame extends Serializable with Logging  (
           val glueContext : GlueContext,
           _records : RDD[DynamicRecord],
           val name : String = s"",
           val transformationContext : String = DynamicFrame.UNDEFINED,
           callSite : CallSite = CallSite("Not provided", ""),
           stageThreshold : Long = 0,
           totalThreshold : Long = 0,
           prevErrors : => Long = 0,
           errorExpr : => Unit = {} )
```

Un `DynamicFrame` è una raccolta distribuita di oggetti DynamicRecord autodescrittivi.

`DynamicFrame` sono stati progettati per fornire un modello di dati flessibile per le operazioni ETL (estrazione, trasformazione e caricamento). Questi oggetti non richiedono la creazione di uno schema e possono essere usati per leggere e trasformare i dati che contengono valori e tipi non organizzati e non coerenti. Un schema può essere calcolato on demand per le operazioni che ne richiedono uno.

`DynamicFrame` offrono un'ampia gamma di trasformazioni per la pulizia dei dati e operazioni ETL. Supportano anche la conversione da e verso SparkSQL DataFrames per l'integrazione con il codice esistente e le numerose operazioni di analisi che DataFrames forniscono.

I parametri seguenti vengono condivisi tra molte trasformazioni AWS Glue che costituiscono oggetti `DynamicFrame`:

- `transformationContext` — Identificatore per questo `DynamicFrame`. Il `transformationContext` viene usato come chiave per lo stato dei segnalibro di processo che viene mantenuto tra esecuzioni.

- `callSite` — Fornisce informazioni sul contesto per la segnalazione degli errori. Questi valori vengono impostati automaticamente durante la chiamata da Python.
- `stageThreshold` — Numero massimo di record di errore consentiti nel calcolo di questo `DynamicFrame` prima di generare un'eccezione, esclusi i record presenti nell'oggetto `DynamicFrame` precedente.
- `totalThreshold` — numero massimo di record di errore totali prima di generare un'eccezione, inclusi quelli dei frame precedenti.

Val errorsCount

```
val errorsCount
```

Numero di record di errore in questo oggetto `DynamicFrame`. Include gli errori restituiti dalle operazioni precedenti.

Def applyMapping

```
def applyMapping( mappings : Seq[Product4[String, String, String, String]],
                  caseSensitive : Boolean = true,
                  transformationContext : String = "",
                  callSite : CallSite = CallSite("Not provided", ""),
                  stageThreshold : Long = 0,
                  totalThreshold : Long = 0
                ) : DynamicFrame
```

- `mappings` — Sequenza di mappature per creare un nuovo oggetto `DynamicFrame`.
- `caseSensitive` — Specifica se considerare o meno le colonne di origine come colonne che fanno distinzione tra maiuscole e minuscole. L'impostazione di questo parametro su false può essere utile per l'integrazione con archivi che non fanno distinzione tra maiuscole e minuscole, come il catalogo dati AWS Glue.

Seleziona, proietta e trasmette le colonne in base a una sequenza di mappature.

Ogni mappatura è costituita da una colonna e un tipo di origine e da una colonna e un tipo target. Le mappature possono essere specificate come un 4-tuple (`source_path`, `source_type`, `target_path`, `target_type`) o un oggetto MappingSpec contenente le stesse informazioni.

Oltre che per semplici proiezioni e trasferimenti, le mappature possono essere usate per annidare campi o annullarne l'annidamento separando i componenti del percorso con "`.`" (punto).

Ad esempio, supponiamo di avere un `DynamicFrame` con lo schema seguente.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- zip: int
}}}
```

Puoi effettuare la chiamata seguente per annullare l'annidamento dei campi `state` e `zip`.

```
{{{
  df.applyMapping(
   Seq(("name", "string", "name", "string"),
       ("age", "int", "age", "int"),
       ("address.state", "string", "state", "string"),
       ("address.zip", "int", "zip", "int")))
}}}
```

Lo schema risultante è il seguente.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- state: string
  |-- zip: int
}}}
```

Puoi anche usare `applyMapping` per riannidare le colonne. Ad esempio, il codice seguente inverte la trasformazione precedente e crea una struttura denominata `address` nel target.

```
{{{
  df.applyMapping(
   Seq(("name", "string", "name", "string"),
       ("age", "int", "age", "int"),
       ("state", "string", "address.state", "string"),
```

```
        ("zip", "int", "address.zip", "int")))
 }}}
```

I nomi dei campi che contengono i caratteri "`.`" (periodo) possono essere quotati utilizzando le virgolette (` `).

> **Note**
>
> Al momento, non puoi utilizzare il metodo `applyMapping` per mappare colonne annidate all'interno di matrici.

Def assertErrorThreshold

```
def assertErrorThreshold : Unit
```

Operazione che forza il calcolo e verifica che il numero di record di errore sia inferiore a `stageThreshold` e `totalThreshold`. Genera un'eccezione se una delle due condizioni non è vera.

Def conteggio

```
lazy
def count
```

Restituisce il numero di elementi inclusi in questo oggetto `DynamicFrame`.

Def dropField

```
def dropField( path : String,
               transformationContext : String = "",
               callSite : CallSite = CallSite("Not provided", ""),
               stageThreshold : Long = 0,
               totalThreshold : Long = 0
             ) : DynamicFrame
```

Restituisce un nuovo oggetto `DynamicFrame` con la colonna specificata rimossa.

Def dropFields

```
def dropFields( fieldNames : Seq[String],   // The column names to drop.
```

```
                transformationContext : String = "",
                callSite : CallSite = CallSite("Not provided", ""),
                stageThreshold : Long = 0,
                totalThreshold : Long = 0
              ) : DynamicFrame
```

Restituisce un nuovo oggetto `DynamicFrame` con le colonne specificate rimosse.

Questo metodo può essere usato per eliminare colonne annidate, incluse quelle all'interno di matrici, ma non per eliminare elementi di matrice specifici.

Def dropNulls

```
def dropNulls( transformationContext : String = "",
               callSite : CallSite = CallSite("Not provided", ""),
               stageThreshold : Long = 0,
               totalThreshold : Long = 0 )
```

Restituisce un nuovo oggetto `DynamicFrame` con tutte le colonne null rimosse.

> **Note**
>
> Rimuove solo le colonne di tipo `NullType`. I singoli valori null in altre colonne non vengono rimossi o modificati.

Cornice Def errorsAsDynamic

```
def errorsAsDynamicFrame
```

Restituisce un nuovo oggetto `DynamicFrame` contenente i record degli errori da questo `DynamicFrame`.

Def filtro

```
def filter( f : DynamicRecord => Boolean,
            errorMsg : String = "",
            transformationContext : String = "",
            callSite : CallSite = CallSite("Not provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
```

```
          ) : DynamicFrame
```

Crea un nuovo oggetto `DynamicFrame` contenente solo i record per i quali la funzione "`f`" restituisce `true`. La funzione di filtro "f" non dovrebbe modificare il record di input.

Def getName

```
def getName : String
```

Restituisce il nome di questo oggetto `DynamicFrame`.

Def getNumPartitions

```
def getNumPartitions
```

Restituisce il numero di partizioni incluse in questo oggetto `DynamicFrame`.

Def calcolato getSchemaIf

```
def getSchemaIfComputed : Option[Schema]
```

Restituisce lo schema se è già stato calcolato. Non analizza i dati se lo schema non è stato ancora calcolato.

Def isSchemaComputed

```
def isSchemaComputed : Boolean
```

Restituisce `true` se lo schema è stato calcolato per questo oggetto `DynamicFrame` oppure restituisce `false` in caso contrario. Se questo metodo restituisce false, la chiamata del metodo `schema` richiede un altro passaggio sui record in questo oggetto `DynamicFrame`.

Def javaToPython

```
def javaToPython : JavaRDD[Array[Byte]]
```

Def join

```
def join( keys1 : Seq[String],
          keys2 : Seq[String],
```

```
          frame2 : DynamicFrame,
          transformationContext : String = "",
          callSite : CallSite = CallSite("Not provided", ""),
          stageThreshold : Long = 0,
          totalThreshold : Long = 0
        ) : DynamicFrame
```

- `keys1` — Le colonne in questo `DynamicFrame` da utilizzare per l'unione.
- `keys2` — Le colonne in `frame2` da utilizzare per l'unione. Deve avere la stessa lunghezza di `keys1`.
- `frame2` — `DynamicFrame` da unire.

Restituisce il risultato dell'esecuzione di una query equijoin con `frame2` usando le chiavi specificate.

Def mappa

```
def map( f : DynamicRecord => DynamicRecord,
         errorMsg : String = "",
         transformationContext : String = "",
         callSite : CallSite = CallSite("Not provided", ""),
         stageThreshold : Long = 0,
         totalThreshold : Long = 0
       ) : DynamicFrame
```

Restituisce un nuovo oggetto `DynamicFrame` creato applicando la funzione "`f`" specificata a ogni record in questo oggetto `DynamicFrame`.

Questo metodo copia ogni record prima di applicare la funzione specificata e di conseguenza è sicuro per la modifica dei record. Se la funzione di mappatura genera un'eccezione per un determinato record, il record verrà contrassegnato come errore e l'analisi dello stack verrà salvata come colonna nel record di errore.

Def mergeDynamicFrames

```
def mergeDynamicFrames( stageDynamicFrame: DynamicFrame,  primaryKeys: Seq[String],
 transformationContext: String = "",
                         options: JsonOptions = JsonOptions.empty, callSite: CallSite =
 CallSite("Not provided"),
                         stageThreshold: Long = 0, totalThreshold: Long = 0):
 DynamicFrame
```

- `stageDynamicFrame` — Il `DynamicFrame` di gestione temporanea da unire.
- `primaryKeys` — L'elenco dei campi chiave primaria per abbinare i record dall'origine e dal `DynamicFrame` di gestione temporanea.
- `transformationContext` — Una stringa univoca utilizzata per recuperare i metadati relativi alla trasformazione corrente (opzionale).
- `options`: una stringa di coppie nome-valore JSON che forniscono informazioni aggiuntive per questa trasformazione.
- `callSite` — Usato per fornire informazioni sul contesto per la segnalazione degli errori.
- `stageThreshold` — Un `Long`. Il numero di errori nella trasformazione specificata per cui l'elaborazione deve restituire un errore.
- `totalThreshold` — Un `Long`. Il numero totale di errori fino a questa trasformazione inclusa per i quali l'elaborazione deve restituire un errore.

Unisce questo `DynamicFrame` con un `DynamicFrame` temporaneo basato sulle chiavi primarie specificate per identificare i record. I registri duplicati (registri con le stesse chiavi primarie) non vengono deduplicati. Se non è presente alcun record corrispondente nel frame temporaneo, tutti i record (inclusi i duplicati) vengono mantenuti dall'origine. Se il frame di staging dispone di record corrispondenti, i suoi registri sovrascrivono i registri nell'origine in AWS Glue.

Il `DynamicFrame` restituito contiene il record A in questi casi:

1. Se `A` esiste sia nel frame di origine che nel frame temporaneo, viene restituito `A` nel frame temporaneo.
2. Se `A` si trova nella tabella di origine e `A.primaryKeys` non si trova nel `stagingDynamicFrame` (ciò significa che `A` non viene aggiornato nella tabella temporanea).

Il frame di origine e il frame temporaneo non devono avere lo stesso schema.

Example

```
val mergedFrame: DynamicFrame = srcFrame.mergeDynamicFrames(stageFrame, Seq("id1",
 "id2"))
```

Def printSchema

```
def printSchema : Unit
```

Stampa lo schema di questo oggetto `DynamicFrame` in `stdout` in un formato leggibile.

Def recomputeSchema

```
def recomputeSchema : Schema
```

Forza un nuovo calcolo dello schema. Questa operazione richiede una scansione dei dati, ma potrebbe "limitare" lo schema se lo schema corrente include dati che nono sono presenti nei dati.

Restituisce lo schema ricalcolato.

Def relazionalizzazione

```
def relationalize( rootTableName : String,
                   stagingPath : String,
                   options : JsonOptions = JsonOptions.empty,
                   transformationContext : String = "",
                   callSite : CallSite = CallSite("Not provided"),
                   stageThreshold : Long = 0,
                   totalThreshold : Long = 0
                 ) : Seq[DynamicFrame]
```

- `rootTableName`: il nome da usare per l'oggetto `DynamicFrame` di base nell'output. Gli oggetti `DynamicFrame` creati tramite il pivoting di matrici usano questo nome come prefisso.
- `stagingPath` — il percorso Amazon Simple Storage Service (Amazon S3) per la scrittura di dati intermedi.
- `options` — opzioni e configurazione per l'applicazione di relazioni. Attualmente inutilizzato.

Appiattisce tutte le strutture annidate e trasforma tramite pivoting le matrici in tabelle separate.

Questa operazione può essere usata per preparare dati annidati a più livelli per l'inserimento in un database relazionale. Le strutture annidate vengono appiattite allo stesso modo della trasformazione Unnest. Inoltre, le matrici vengono trasformate tramite pivoting in tabelle separate attraverso un'operazione in cui ogni elemento di matrice diventa una riga. Ad esempio, supponiamo di avere un `DynamicFrame` con i dati seguenti.

```
{"name": "Nancy", "age": 47, "friends": ["Fred", "Lakshmi"]}
{"name": "Stephanie", "age": 28, "friends": ["Yao", "Phil", "Alvin"]}
{"name": "Nathan", "age": 54, "friends": ["Nicolai", "Karen"]}
```

Eseguire il seguente codice.

```
{{{
  df.relationalize("people", "s3:/my_bucket/my_path", JsonOptions.empty)
}}}
```

Il codice produce due tabelle. La prima tabella è denominata "persone" e contiene quanto segue.

```
{{{
  {"name": "Nancy", "age": 47, "friends": 1}
  {"name": "Stephanie", "age": 28, "friends": 2}
  {"name": "Nathan", "age": 54, "friends": 3)
}}}
```

Qui le matrici friends sono state sostituite con una chiave di join generata automaticamente. Viene creata una tabella separata denominata `people.friends` con il contenuto seguente.

```
{{{
  {"id": 1, "index": 0, "val": "Fred"}
  {"id": 1, "index": 1, "val": "Lakshmi"}
  {"id": 2, "index": 0, "val": "Yao"}
  {"id": 2, "index": 1, "val": "Phil"}
  {"id": 2, "index": 2, "val": "Alvin"}
  {"id": 3, "index": 0, "val": "Nicolai"}
  {"id": 3, "index": 1, "val": "Karen"}
}}}
```

In questa tabella "`id`" è una chiave di join che identifica il record da cui proviene l'elemento della matrice, "`index`" fa riferimento alla posizione nella matrice originale e "`val`" è l'effettiva voce della matrice.

Il metodo `relationalize` restituisce la sequenza di oggetti `DynamicFrame` creati applicando questo processo in modo ricorsivo a tutte le matrici.

> **Note**
>
> La libreria AWS Glue genera automaticamente le chiavi di join per le nuove tabelle. Per garantire che le chiavi di join siano univoche tra esecuzioni di processi, devono essere abilitati i segnalibro di processo.

Def renameField

```
def renameField( oldName : String,
                 newName : String,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

- `oldName`: nome originale della colonna.
- `newName`: nuovo nome della colonna.

Restituisce un nuovo `DynamicFrame` con il campo specificato rinominato.

Questo metodo può essere usato per rinominare campi annidati. Ad esempio, il codice seguente rinominerebbe `state` in `state_code` all'interno della struttura address.

```
{{{
  df.renameField("address.state", "address.state_code")
}}}
```

Def ripartizione

```
def repartition( numPartitions : Int,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

Restituisce un nuovo oggetto `DynamicFrame` con partizioni `numPartitions`.

Def resolveChoice

```
def resolveChoice( specs : Seq[Product2[String, String]] = Seq.empty[ResolveSpec],
                   choiceOption : Option[ChoiceOption] = None,
                   database : Option[String] = None,
                   tableName : Option[String] = None,
```

```
    transformationContext : String = "",
    callSite : CallSite = CallSite("Not provided", ""),
    stageThreshold : Long = 0,
    totalThreshold : Long = 0
  ) : DynamicFrame
```

- `choiceOption` — Un'operazione da applicare a tutte le colonne `ChoiceType` non elencate nella sequenza delle specifiche.
- `database` — Il database del catalogo dati da usare con l'operazione `match_catalog`.
- `tableName` — La tabella del catalogo dati da usare con l'operazione `match_catalog`.

Restituisce un nuovo oggetto `DynamicFrame` sostituendo uno o più oggetti `ChoiceType` con un tipo più specifico.

Ci sono due modi per utilizzare `resolveChoice`. Il primo consiste nell'indicare una sequenza di colonne specifiche e come risolverle. Queste vengono specificate come tuple costituite da coppie (colonna, operazione).

Sono disponibili le operazioni seguenti:

- `cast:type`: tenta di trasmettere tutti i valori al tipo specificato.
- `make_cols`: converte ogni tipo distinto in colonna con il nome `columnName_type`.
- `make_struct`: converte una colonna in struttura con chiavi per ogni tipo distinto.
- `project:type` — mantiene solo i valori del tipo specificato.

L'altra modalità per `resolveChoice` è specificare una singola risoluzione per tutti gli oggetti `ChoiceType`. Puoi usare questa modalità nei casi in cui l'elenco completo di oggetti `ChoiceType` non è noto prima dell'esecuzione. Oltre alle operazioni elencate in precedenza, questa modalità supporta anche l'operazione seguente:

- `match_catalogChoiceType`: tenta di trasmettere ogni oggetto al tipo corrispondente nella tabella del catalogo specificata.

Esempi:

Risoluzione della colonna `user.id` mediante casting a un tipo int, facendo in modo che il campo `address` mantenga solo le strutture:

```
{{{
  df.resolveChoice(specs = Seq(("user.id", "cast:int"), ("address", "project:struct")))
}}}
```

Risoluzione di tutti gli oggetti `ChoiceType` mediante conversione di ogni scelta in una colonna separata:

```
{{{
  df.resolveChoice(choiceOption = Some(ChoiceOption("make_cols")))
}}}
```

Risoluzione di tutte gli oggetti `ChoiceType` mediante casting ai tipi nella tabella del catalogo specificata:

```
{{{
  df.resolveChoice(choiceOption = Some(ChoiceOption("match_catalog")),
                   database = Some("my_database"),
                   tableName = Some("my_table"))
}}}
```

Def schema

```
def schema : Schema
```

Restituisce lo schema di questo oggetto `DynamicFrame`.

Lo schema restituito è garantito per contenere ogni campo presente in un record in questo `DynamicFrame`. Tuttavia, in un esiguo numero di casi, può anche contenere campi aggiuntivi. Puoi utilizzare il metodo Unnest per "ridurre" lo schema in base ai record in questo `DynamicFrame`.

Def selectField

```
def selectField( fieldName : String,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

Restituisce un singolo campo come `DynamicFrame`.

Def selectFields

```
def selectFields( paths : Seq[String],
                  transformationContext : String = "",
                  callSite : CallSite = CallSite("Not provided", ""),
                  stageThreshold : Long = 0,
                  totalThreshold : Long = 0
                ) : DynamicFrame
```

- `paths` — La sequenza dei nomi delle colonne da selezionare.

Restituisce un nuovo oggetto `DynamicFrame` contenente le colonne specificate.

> **Note**
>
> Il metodo `selectFields` può essere usato solo per selezionare colonne di primo livello. Puoi utilizzare il metodo applyMapping per selezionare colonne annidate.

Def mostra

```
def show( numRows : Int = 20 ) : Unit
```

- `numRows` — Numero di righe da stampare.

Stampa le righe di questo oggetto `DynamicFrame` in formato JSON.

Def semplifica DDBJSON

Le esportazioni DynamoDB con il connettore di esportazione AWS Glue DynamoDB producono file JSON di strutture annidate specifiche. Per ulteriori informazioni, consulta Data objects. `simplifyDDBJson` Semplifica le colonne annidate in un tipo DynamicFrame di dati di questo tipo e ne restituisce uno nuovo semplificato. DynamicFrame Se ci sono più tipi o un tipo di mappa contenuto in un tipo di elenco, gli elementi nell'elenco non verranno semplificati. Questo metodo supporta solo i dati nel formato JSON di esportazione DynamoDB. Prendi in considerazione `unnest` la possibilità di apportare modifiche simili su altri tipi di dati.

```
def simplifyDDBJson() : DynamicFrame
```

Questo metodo non accetta alcun parametro.

Input di esempio

Prendi in considerazione lo schema seguente generato da un'esportazione DynamoDB:

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

Esempio di codice

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContextimport scala.collection.JavaConverters._
```

```
object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> "ddbTableARN",
        "dynamodb.s3.bucket" -> "exportBucketLocation",
        "dynamodb.s3.prefix" -> "exportBucketPrefix",
        "dynamodb.s3.bucketOwner" -> "exportBucketAccountID",
      ))
    ).getDynamicFrame()

    val simplified = dynamicFrame.simplifyDDBJson()
    simplified.printSchema()

    Job.commit()
  }

}
```

Output di esempio

La trasformazione `simplifyDDBJson` semplificherà questo processo in:

```
root
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
|    |    |-- packageName: string
|    |    |-- updatedAt: string
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
```

```
|-- nullValue: null
```

Def spigot

```
def spigot( path : String,
            options : JsonOptions = new JsonOptions("{}"),
            transformationContext : String = "",
            callSite : CallSite = CallSite("Not provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

Trasformazione passthrough che restituisce gli stessi record, ma scrive un sottoinsieme di record come effetto secondario.

- `path` — Il percorso in Amazon S3 in cui scrivere l'output, nel formato `s3://bucket//path`.
- `options` — Una mappa `JsonOptions` opzionale che descrive il comportamento di campionamento.

Restituisce un oggetto `DynamicFrame` contenente gli stessi record di questo.

Per impostazione predefinita, scrive 100 record arbitrari nel percorso specificato da `path`. Questo comportamento può essere personalizzato usando la mappa `options`. Le chiavi valide includono le seguenti:

- `topk` — Specifica il numero totale di record scritti. Il valore di default è 100.
- `prob`: specifica la probabilità (sotto forma di valore decimale) di inclusione di un singolo record. Il valore predefinito è 1.

Ad esempio, la chiamata seguente esegue il campionamento del set di dati selezionando ogni record con una probabilità del 20% e arrestandosi dopo la scrittura di 200 record.

```
{{{
  df.spigot("s3://my_bucket/my_path", JsonOptions(Map("topk" -&gt; 200, "prob" -&gt;
 0.2)))
}}}
```

Def splitFields

```
def splitFields( paths : Seq[String],
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : Seq[DynamicFrame]
```

- `paths` — Percorsi da includere nel primo `DynamicFrame`.

Restituisce una sequenza di due oggetti `DynamicFrame`. Il primo oggetto `DynamicFrame` contiene i percorsi specificati, mentre il secondo contiene tutte le altre colonne.

Esempio

Questo esempio prende una tabella DynamicFrame creata dalla `persons` tabella nel `legislators` database del AWS Glue Data Catalog e la DynamicFrame divide in due, con i campi specificati che entrano nel primo DynamicFrame e i campi rimanenti in un secondo DynamicFrame. L'esempio sceglie quindi il primo DynamicFrame dal risultato.

```
val InputFrame = glueContext.getCatalogSource(database="legislators",
 tableName="persons",
transformationContext="InputFrame").getDynamicFrame()

val SplitField_collection = InputFrame.splitFields(paths=Seq("family_name", "name",
 "links.note",
"links.url", "gender", "image", "identifiers.scheme", "identifiers.identifier",
 "other_names.lang",
"other_names.note", "other_names.name"), transformationContext="SplitField_collection")

val ResultFrame = SplitField_collection(0)
```

Def splitRows

```
def splitRows( paths : Seq[String],
               values : Seq[Any],
               operators : Seq[String],
               transformationContext : String,
               callSite : CallSite,
               stageThreshold : Long,
```

```
              totalThreshold : Long
            ) : Seq[DynamicFrame]
```

Suddivide le righe in base a predicati che confrontano colonne e costanti.

- `paths` — Colonne da usare per il confronto.
- `values` — I valori di costante da usare per il confronto.
- `operators` — Gli operatori da usare per il confronto.

Restituisce una sequenza di due oggetti `DynamicFrame`. Il primo contiene le righe per cui il predicato è true, il secondo contiene quelle per cui è false.

I predicati vengono specificati usando tre sequenze: "`paths`" contiene i nomi delle colonne (possibilmente annidate), "`values`" contiene i valori di costante rispetto ai quali eseguire il confronto e "`operators`" contiene gli operatori da usare per il confronto. Le tre sequenze devono essere tutte della stessa lunghezza: l'nesimo operatore verrà usato per confrontare la nesima colonna con l'nesimo valore.

Gli operatori consentiti sono: "`!=`", "`=`", "`<=`", "`<`", "`>=`" o "`>`".

Ad esempio, la chiamata seguente divide un oggetto `DynamicFrame` in modo che il primo frame di output contenga i record di persone degli Stati Uniti di età maggiore di 65 anni e che il secondo contenga tutti gli altri record.

```
{{{
  df.splitRows(Seq("age", "address.country"), Seq(65, "USA"), Seq("&gt;=", "="))
}}}
```

Def stageErrorsCount

```
def stageErrorsCount
```

Restituisce il numero di record di errore creati durante il calcolo di questo oggetto `DynamicFrame`. Sono esclusi gli errori restituiti dalla operazioni precedenti passate a questo oggetto `DynamicFrame` come input.

Def toDF

```
def toDF( specs : Seq[ResolveSpec] = Seq.empty[ResolveSpec] ) : DataFrame
```

Converte questo `DynamicFrame` in un Apache Spark SQL `DataFrame` con lo stesso schema e gli stessi record.

> **Note**
>
> Poiché gli oggetti `DataFrame` non supportano oggetti `ChoiceType`, questo metodo converte automaticamente le colonne `ChoiceType` in oggetti `StructType`. Per ulteriori informazioni sulle opzioni per le scelte di risoluzione, consulta resolveChoice.

Def unbox

```
def unbox( path : String,
           format : String,
           optionString : String = "{}",
           transformationContext : String = "",
           callSite : CallSite = CallSite("Not provided"),
           stageThreshold : Long = 0,
           totalThreshold : Long = 0
         ) : DynamicFrame
```

- `path` — La colonna da analizzare. Deve essere di tipo String o Binary.
- `format`: formato da usare per l'analisi.
- `optionString`: opzioni da passare al formato, ad esempio il separatore per file CSV.

Analizza una stringa incorporata o una colonna binaria in base al formato specificato. Le colonne analizzate vengono annidate all'interno di una struttura con il nome di colonna originale.

Supponi, ad esempio, di avere un file CSV con una colonna JSON incorporata.

```
name, age, address
Sally, 36, {"state": "NE", "city": "Omaha"}
...
```

Dopo un'analisi iniziale, avresti un oggetto `DynamicFrame` con lo schema seguente.

```
{{{
  root
```

```
  |-- name: string
  |-- age: int
  |-- address: string
}}}
```

Puoi chiamare `unbox` nella colonna address per analizzare i componenti specifici.

```
{{{
  df.unbox("address", "json")
}}}
```

Otterrai un oggetto `DynamicFrame` con lo schema seguente.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- city: string
}}}
```

Def unnest

```
def unnest( transformationContext : String = "",
            callSite : CallSite = CallSite("Not Provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

Restituisce un nuovo oggetto `DynamicFrame` con tutte le strutture annidate appiattite. I nomi vengono creati usando il carattere `"."` (punto).

Ad esempio, supponiamo di avere un `DynamicFrame` con lo schema seguente.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
```

```
  |    |-- state: string
  |    |-- city: string
}}}
```

La chiamata seguente annulla l'annidamento della struttura address.

```
{{{
  df.unnest()
}}}
```

Lo schema risultante è il seguente.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address.state: string
  |-- address.city: string
}}}
```

Questo metodo, inoltre, annulla l'annidamento delle strutture all'interno di array. Tuttavia, per motivi storici, ai nomi di tali campi vengono anteposti il nome della matrice di chiusura e "`.val`".

Def unnestDDBJson

```
unnestDDBJson(transformationContext : String = "",
            callSite : CallSite = CallSite("Not Provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0): DynamicFrame
```

Snidifica le colonne nidificate in un `DynamicFrame` che si trovano specificamente nella struttura JSON di DynamoDB e restituisce un nuovo `DynamicFrame` non annidato. Le colonne che sono di un array di struct non verranno annidate. Si noti che si tratta di un tipo specifico di trasformazione di snidamento che si comporta in modo diverso dalla normale trasformazione di `unnest` e richiede che i dati siano già nella struttura JSON di DynamoDB. Per ulteriori informazioni, consulta DynamoDB JSON.

Ad esempio, lo schema di lettura di un'esportazione con la struttura JSON DynamoDB potrebbe apparire come segue:

```
root
|-- Item: struct
|    |-- ColA: struct
|    |    |-- S: string
|    |-- ColB: struct
|    |    |-- S: string
|    |-- ColC: struct
|    |    |-- N: string
|    |-- ColD: struct
|    |    |-- L: array
|    |    |    |-- element: null
```

La trasformazione di `unnestDDBJson()` lo convertirebbe in:

```
root
|-- ColA: string
|-- ColB: string
|-- ColC: string
|-- ColD: array
|    |-- element: null
```

L'esempio di codice seguente mostra come utilizzare il connettore di esportazione DynamoDB AWS Glue, richiamare un JSON di DynamoDB unnest e stampare il numero di partizioni:

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
```

```
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> "<test_source>",
        "dynamodb.s3.bucket" -> "<bucket name>",
        "dynamodb.s3.prefix" -> "<bucket prefix>",
        "dynamodb.s3.bucketOwner" -> "<account_id of bucket>",
      ))
    ).getDynamicFrame()

    val unnested = dynamicFrame.unnestDDBJson()
    print(unnested.getNumPartitions())

    Job.commit()
  }

}
```

Def withFrameSchema

```
def withFrameSchema( getSchema : () => Schema ) : DynamicFrame
```

- `getSchema` — Una funzione che restituisce lo schema da usare. È specificata come funzione a zero parametri per posticipare i calcoli potenzialmente onerosi.

Imposta lo schema di questo oggetto `DynamicFrame` sul valore specificato. Viene usato per lo più internamente per evitare ricalcoli dello schema onerosi. Lo schema passato deve contenere tutte le colonne presenti nei dati.

Def withName

```
def withName( name : String ) : DynamicFrame
```

- `name` — Il nuovo nome da usare.

Restituisce una copia di questo oggetto `DynamicFrame` con un nuovo nome.

Def withTransformationContext

```
def withTransformationContext( ctx : String ) : DynamicFrame
```

Restituisce una copia di questo oggetto `DynamicFrame` con il contesto di trasformazione specificato.

L'oggetto DynamicFrame

Pacchetto: com.amazonaws.services.glue

```
object DynamicFrame
```

Applicazione di def

```
def apply( df : DataFrame,
           glueContext : GlueContext
         ) : DynamicFrame
```

Def emptyDynamicFrame

```
def emptyDynamicFrame( glueContext : GlueContext ) : DynamicFrame
```

Def fromPythonRDD

```
def fromPythonRDD( rdd : JavaRDD[Array[Byte]],
                   glueContext : GlueContext
                 ) : DynamicFrame
```

Def ignoreErrors

```
def ignoreErrors( fn : DynamicRecord => DynamicRecord ) : DynamicRecord
```

Def inlineErrors

```
def inlineErrors( msg : String,
                  callSite : CallSite
                ) : (DynamicRecord => DynamicRecord)
```

Def newFrameWithErrors

```
def newFrameWithErrors( prevFrame : DynamicFrame,
                        rdd : RDD[DynamicRecord],
                        name : String = "",
                        transformationContext : String = "",
                        callSite : CallSite,
                        stageThreshold : Long,
                        totalThreshold : Long
                      ) : DynamicFrame
```

Classe Scala DynamicRecord in AWS Glue

Argomenti

Pacchetto: com.amazonaws.services.glue

```
class DynamicRecord extends Serializable with Writable with Cloneable
```

Un `DynamicRecord` è una struttura di dati autodescrittiva che rappresenta una riga di dati nel set di dati in fase di elaborazione. È autodescrittiva nel senso che puoi ottenere lo schema della riga rappresentata da `DynamicRecord` controllando il record stesso. Un `DynamicRecord` è simile a un `Row` in Apache Spark.

Def addField

```
def addField( path : String,
              dynamicNode : DynamicNode
            ) : Unit
```

Aggiunge un DynamicNode al percorso specificato.

- `path` — Percorso per il campo da aggiungere.
- `dynamicNode` — Il DynamicNode da aggiungere al percorso specificato.

Def dropField

```
 def dropField(path: String, underRename: Boolean = false): Option[DynamicNode]
```

Rilascia un DynamicNode dal percorso specificato e restituisce il nodo rilasciato se non c'è un array nel percorso specificato.

- `path` — Percorso per il campo da rilasciare.
- `underRenamedropField` — True se fa parte di un'azione di rinomina o false in caso contrario (false per impostazione predefinita).

Restituisce una `scala.Option Option` (DynamicNode).

Def setError

```
def setError( error : Error )
```

Imposta il record come un record di errore, come specificato dal parametro `error`.

Restituisce una `DynamicRecord`.

Def isError

```
def isError
```

Verifica se il record è un record di errore.

Def getError

```
def getError
```

Ottiene `Error` se il record è un record di errore. Restituisce `scala.Some` Some (Error) se il record è un record di errore o altrimenti `scala.None`.

Def clearError

```
def clearError
```

Imposta `Error` su `scala.None.None`.

Def scrittura

```
override def write( out : DataOutput ) : Unit
```

Def readFields

```
override def readFields( in : DataInput ) : Unit
```

Def clone

```
override def clone : DynamicRecord
```

Clona questo record in un nuovo `DynamicRecord` e lo restituisce.

Def schema

```
def schema
```

Ottiene lo Schema controllando il record.

Def getRoot

```
def getRoot : ObjectNode
```

Ottiene la radice ObjectNode per il record.

Def toJson

```
def toJson : String
```

Ottiene la stringa JSON per il record.

Def getFieldNode

```
def getFieldNode( path : String ) : Option[DynamicNode]
```

Ottiene il valore del campo nel path specificato come opzione di DynamicNode.

Restituisce scala.Some Some (DynamicNode) se il campo esiste, altrimenti scala.None.None.

Def getField

```
def getField( path : String ) : Option[Any]
```

Ottiene il valore del campo nel path specificato come opzione di DynamicNode.

Restituisce scala.Some Some (valore).

Def hashCode

```
override def hashCode : Int
```

Def equals

```
override def equals( other : Any )
```

Oggetto DynamicRecord

```
object DynamicRecord
```

Applicazione di def

```
def apply( row : Row,
           schema : SparkStructType )
```

Applica il metodo per convertire un Apache Spark SQL `Row` in un DynamicRecord.

- `row` — Una `Row` Spark SQL.
- `schema` — Lo `Schema` della riga in questione.

Restituisce una `DynamicRecord`.

Proprietà RecordTraverser

```
trait RecordTraverser {
  def nullValue(): Unit
  def byteValue(value: Byte): Unit
  def binaryValue(value: Array[Byte]): Unit
  def booleanValue(value: Boolean): Unit
  def shortValue(value: Short) : Unit
  def intValue(value: Int) : Unit
  def longValue(value: Long) : Unit
  def floatValue(value: Float): Unit
  def doubleValue(value: Double): Unit
  def decimalValue(value: BigDecimal): Unit
  def stringValue(value: String): Unit
  def dateValue(value: Date): Unit
  def timestampValue(value: Timestamp): Unit
  def objectStart(length: Int): Unit
  def objectKey(key: String): Unit
  def objectEnd(): Unit
  def mapStart(length: Int): Unit
```

```
  def mapKey(key: String): Unit
  def mapEnd(): Unit
  def arrayStart(length: Int): Unit
  def arrayEnd(): Unit
}
```

API di AWS Glue Scala GlueContext

Pacchetto: com.amazonaws.services.glue

```
class GlueContext extends SQLContext(sc) (
           @transient val sc : SparkContext,
           val defaultSourcePartitioner : PartitioningStrategy )
```

`GlueContext` è il punto di ingresso per la lettura e la scrittura di un DynamicFrame da e verso Amazon Simple Storage Service (Amazon S3), il catalogo dati AWS Glue, JDBC e così via. Questa classe fornisce funzioni di utilità per creare oggetti Proprietà DataSource e DataSink che possono in cambio essere utilizzati per leggere e scrivere `DynamicFrame`.

`GlueContext` può anche essere usato per impostare un numero target di partizioni (per impostazione predefinita 20) nel `DynamicFrame` se il numero di partizioni create dalla sorgente è inferiore alla soglia minima delle partizioni (per impostazione predefinita 10).

def addIngestionTimeColumns

```
def addIngestionTimeColumns(
         df : DataFrame,
         timeGranularity : String = "") : dataFrame
```

Aggiunge colonne del tempo di importazione dati come `ingest_year`, `ingest_month`, `ingest_day`, `ingest_hour`, `ingest_minute` al `DataFrame` di input. Questa funzione viene generata automaticamente nello script generato da AWS Glue quando si specifica una tabella del catalogo dati con Amazon S3 come destinazione. Questa funzione aggiorna automaticamente la partizione con le colonne del tempo di importazione dati nella tabella di output. Ciò consente ai dati di output di venire partizionati automaticamente nel tempo di importazione dati senza necessitare di colonne di tempo di inserimento esplicite nei dati di input.

- `dataFrame`: il `dataFrame` al quale aggiungere le colonne del tempo di importazione dati.
- `timeGranularity`: la granularità delle colonne temporali. I valori validi sono "`day`", "`hour`" e "`minute`". Ad esempio, se "`hour`" viene passato alla funzione, il `dataFrame` originale avrà

"ingest_year", "ingest_month", "ingest_day" e "ingest_hour" colonne temporali aggiunte.

Restituisce il frame di dati dopo l'aggiunta di colonne di granularità di tempo.

Esempio:

```
glueContext.addIngestionTimeColumns(dataFrame, "hour")
```

def createDataFrameFromOptions

```
def createDataFrameFromOptions( connectionType : String,
                        connectionOptions : JsonOptions,
                        transformationContext : String = "",
                        format : String = null,
                        formatOptions : JsonOptions = JsonOptions.empty
                      ) : DataSource
```

Restituisce un DataFrame creato con la connessione e il formato specificati. Utilizza questa funzione solo con origini di streaming AWS Glue.

- connectionType: il tipo di connessione streaming. I valori validi includono kinesis e kafka.
- connectionOptions: opzioni di connessione, che sono diverse per Kinesis e Kafka. È possibile trovare l'elenco di tutte le opzioni di connessione per ogni origine dati di streaming all'indirizzo Tipi e opzioni di connessione per ETL in AWS Glue per Spark. Di seguito vengono illustrate le differenze delle opzioni di connessione di streaming:
  - Le origini di streaming di Kinesis richiedono streamARN, startingPosition, inferSchema e classification.
  - Le origini di streaming di Kafka richiedono connectionName, topicName, startingOffsets, inferSchema e classification.
- transformationContext: il contesto di trasformazione da utilizzare (facoltativo).
- format: una specifica del formato (facoltativo). Viene usata per una connessione Amazon S3 o AWS Glue che supporta più formati. Per ulteriori informazioni sui formati supportati, consulta Opzioni del formato dati per input e output in AWS Glue per Spark.
- formatOptions: opzioni di formattazione per il formato specificato. Per ulteriori informazioni sulle opzioni di formato supportate, consulta Opzioni del formato dei dati.

Esempio per l'origine di streaming Amazon Kinesis:

```
val data_frame_datasource0 =
glueContext.createDataFrameFromOptions(transformationContext = "datasource0",
 connectionType = "kinesis",
connectionOptions = JsonOptions("""{"streamName": "example_stream", "startingPosition":
 "TRIM_HORIZON", "inferSchema": "true", "classification": "json"}}""""))
```

Esempio per l'origine di streaming Kafka:

```
val data_frame_datasource0 =
glueContext.createDataFrameFromOptions(transformationContext = "datasource0",
 connectionType = "kafka",
connectionOptions = JsonOptions("""{"connectionName": "example_connection",
 "topicName": "example_topic", "startingPosition": "earliest", "inferSchema": "false",
 "classification": "json", "schema":"`column1` STRING, `column2` STRING"}""""))
```

forEachBatch

`forEachBatch(frame, batch_function, options)`

Applica il `batch_function` passato a ogni micro batch che viene letto dall'origine di streaming.

- `frame`: il frame di dati contenente il micro batch corrente.
- `batch_function`: una funzione che verrà applicata per ogni micro batch.
- `options`: una raccolta di coppie chiave-valore che contiene informazioni su come elaborare micro batch. Sono richieste le seguenti opzioni:
  - `windowSize`: la quantità di tempo da dedicare all'elaborazione di ciascun batch.
  - `checkpointLocation`: la posizione in cui sono archiviati i checkpoint per il processo ETL di streaming.
  - `batchMaxRetries`: numero massimo di tentativi per riprovare il processo se il batch ha esito negativo. Il valore predefinito è 3. Questa opzione è configurabile solo per Glue versione 2.0 e successive.

Esempio:

```
glueContext.forEachBatch(data_frame_datasource0, (dataFrame: Dataset[Row], batchId:
 Long) =>
```

```
   {
      if (dataFrame.count() > 0)
        {
          val datasource0 = DynamicFrame(glueContext.addIngestionTimeColumns(dataFrame,
 "hour"), glueContext)
          // @type: DataSink
          // @args: [database = "tempdb", table_name = "fromoptionsoutput",
 stream_batch_time = "100 seconds",
          //      stream_checkpoint_location = "s3://from-options-testing-eu-central-1/
fromOptionsOutput/checkpoint/",
          //      transformation_ctx = "datasink1"]
          // @return: datasink1
          // @inputs: [frame = datasource0]
          val options_datasink1 = JsonOptions(
             Map("partitionKeys" -> Seq("ingest_year", "ingest_month","ingest_day",
 "ingest_hour"),
             "enableUpdateCatalog" -> true))
          val datasink1 = glueContext.getCatalogSink(
             database = "tempdb",
             tableName = "fromoptionsoutput",
             redshiftTmpDir = "",
             transformationContext = "datasink1",
             additionalOptions = options_datasink1).writeDynamicFrame(datasource0)
        }
   }, JsonOptions("""{"windowSize" : "100 seconds",
         "checkpointLocation" : "s3://from-options-testing-eu-central-1/
fromOptionsOutput/checkpoint/"}"""))
```

def getCatalogSink

```
def getCatalogSink( database : String,
        tableName : String,
        redshiftTmpDir : String = "",
        transformationContext : String = ""
        additionalOptions: JsonOptions = JsonOptions.empty,
        catalogId: String = null
) : DataSink
```

Crea un DataSink che scrive in una posizione specificata di una tabella definita nel catalogo dati.

- `database` — Il nome del database nel catalogo di dati.
- `tableName` — Il nome della tabella nel catalogo dati.

- `redshiftTmpDir` — La directory di gestione temporanea da utilizzare con alcuni sink di dati. Impostato su per impostazione predefinita.
- `transformationContext` — Il contesto di trasformazione associato al sink che i segnalibri di processo utilizzano. Impostato su per impostazione predefinita.
- `additionalOptions`: opzioni aggiuntive fornite a AWS Glue.
- `catalogId` — L'ID catalogo (ID account) relativo al catalogo dati a cui si accede. Se null, viene utilizzato l'ID account predefinito del chiamante.

Restituisce il `DataSink`.

def getCatalogSource

```
def getCatalogSource( database : String,
                      tableName : String,
                      redshiftTmpDir : String = "",
                      transformationContext : String = ""
                      pushDownPredicate : String = " "
                      additionalOptions: JsonOptions = JsonOptions.empty,
                      catalogId: String = null
                    ) : DataSource
```

Crea un Proprietà DataSource che legge dati da una definizione di tabella nel catalogo di dati.

- `database` — Il nome del database nel catalogo di dati.
- `tableName` — Il nome della tabella nel catalogo dati.
- `redshiftTmpDir` — La directory di gestione temporanea da utilizzare con alcuni sink di dati. Impostato su per impostazione predefinita.
- `transformationContext` — Il contesto di trasformazione associato al sink che i segnalibri di processo utilizzano. Impostato su per impostazione predefinita.
- `pushDownPredicate`: filtra le partizioni senza dover elencare e leggere tutti i file nel set di dati. Per ulteriori informazioni, consulta Prefiltraggio con i predicati pushdown.
- `additionalOptions`: una raccolta di coppie nome/valore opzionali. Le opzioni possibili includono quelle elencate in Tipi e opzioni di connessione per ETL in AWS Glue per Spark ad eccezione di `endpointUrl`, `streamName`, `bootstrap.servers`, `security.protocol`, `topicName`, `classification` e `delimiter`. Un'altra opzione supportata è `catalogPartitionPredicate`:

`catalogPartitionPredicate` — È possibile passare un'espressione di catalogo per filtrare in base alle colonne di indice. Questo esegue il push down del filtro sul lato server. Per ulteriori informazioni, consulta la pagina relativa agli indici di partizionamento di AWS Glue. Tieni presente che `push_down_predicate` e `catalogPartitionPredicate` usano sintassi diverse. Il primo utilizza la sintassi standard Spark SQL e il secondo utilizza il parser JSQL.

- `catalogId` — L'ID catalogo (ID account) relativo al catalogo dati a cui si accede. Se null, viene utilizzato l'ID account predefinito del chiamante.

Restituisce il `DataSource`.

Esempio di origine di streaming

```
val data_frame_datasource0 = glueContext.getCatalogSource(
    database = "tempdb",
    tableName = "test-stream-input",
    redshiftTmpDir = "",
    transformationContext = "datasource0",
    additionalOptions = JsonOptions("""{
        "startingPosition": "TRIM_HORIZON", "inferSchema": "false"}""")
    ).getDataFrame()
```

def getJDBCSink

```
def getJDBCSink( catalogConnection : String,
                 options : JsonOptions,
                 redshiftTmpDir : String = "",
                 transformationContext : String = "",
                 catalogId: String = null
               ) : DataSink
```

Crea un oggetto DataSink che scrive in un database JDBC specificato in un oggetto `Connection` nel catalogo dati. L'oggetto `Connection` dispone di informazioni per connettersi a un sink JDBC, compresi URL, nome utente, password, VPC, sottorete e gruppi di sicurezza.

- `catalogConnection` — Il nome della connessione nel catalogo dati contenente l'URL JDBC su cui scrivere.
- `options`: una stringa di coppie nome-valore JSON che forniscono informazioni aggiuntive necessarie per scrivere su un datastore JDBC. Questo include:

- dbtable (obbligatorio): il nome della tabella JDBC. Per i archivi dati JDBC che supportano schemi all'interno di un database, specifica `schema.table-name`. Se non viene fornito alcuno schema, viene usato lo schema "pubblico" predefinito. L'esempio seguente mostra un parametro options che punta a uno schema denominato `test` e a una tabella denominata `test_table` nel database `test_db`.

  ```
  options = JsonOptions("""{"dbtable": "test.test_table", "database": "test_db"}""")
  ```

- database (obbligatorio): il nome del database JDBC.
- Le eventuali opzioni aggiuntive trasmesse direttamente allo scrittore SparkSQL JDBC. Per ulteriori informazioni, consulta la pagina relativa all'origine dati Redshift per Spark.

- `redshiftTmpDir` — Una directory di gestione temporanea da utilizzare con alcuni sink di dati. Impostato su per impostazione predefinita.
- `transformationContext` — Il contesto di trasformazione associato al sink che i segnalibri di processo utilizzano. Impostato su per impostazione predefinita.
- `catalogId` — L'ID catalogo (ID account) relativo al catalogo dati a cui si accede. Se null, viene utilizzato l'ID account predefinito del chiamante.

Codice di esempio:

```
getJDBCSink(catalogConnection = "my-connection-name", options =
 JsonOptions("""{"dbtable": "my-jdbc-table", "database": "my-jdbc-db"}"""),
 redshiftTmpDir = "", transformationContext = "datasink4")
```

Restituisce il `DataSink`.

def getSink

```
def getSink( connectionType : String,
             connectionOptions : JsonOptions,
             transformationContext : String = ""
           ) : DataSink
```

Crea un DataSink che scrive dati su una destinazione come Amazon Simple Storage Service (Amazon S3), JDBC o il catalogo dati AWS Glue.

- `connectionType` — Il tipo di connessione. Per informazioni, consultare the section called "Parametri di connessione".
- `connectionOptions`: una stringa di coppie nome-valore JSON che forniscono informazioni aggiuntive per stabilire la connessione con il sink dei dati. Per informazioni, consultare the section called "Parametri di connessione".
- `transformationContext` — Il contesto di trasformazione associato al sink che i segnalibri di processo utilizzano. Impostato su per impostazione predefinita.

Restituisce il `DataSink`.

def getSinkWithFormat

```
def getSinkWithFormat( connectionType : String,
                       options : JsonOptions,
                       transformationContext : String = "",
                       format : String = null,
                       formatOptions : JsonOptions = JsonOptions.empty
                     ) : DataSink
```

Crea un DataSink che scrive dati su una destinazione, ad esempio Amazon S3, JDBC o sul catalogo dati e in più imposta il formato dei dati da scrivere sulla destinazione.

- `connectionType` — Il tipo di connessione. Per informazioni, consultare the section called "Parametri di connessione".
- `options`: una stringa di coppie nome-valore JSON che forniscono informazioni aggiuntive per stabilire connessioni con il sink dei dati. Per informazioni, consultare the section called "Parametri di connessione".
- `transformationContext` — Il contesto di trasformazione associato al sink che i segnalibri di processo utilizzano. Impostato su per impostazione predefinita.
- `format` — Il formato dei dati da scrivere sulla destinazione.
- `formatOptions`: una stringa di coppie nome-valore JSON che forniscono opzioni aggiuntive per la formattazione dei dati nella destinazione. Per informazioni, consultare Opzioni del formato dei dati.

Restituisce il `DataSink`.

def getSource

```
def getSource( connectionType : String,
               connectionOptions : JsonOptions,
               transformationContext : String = ""
               pushDownPredicate
             ) : DataSource
```

Crea un Proprietà DataSource che legge i dati da un'origine, ad esempio Amazon S3, JDBC o il catalogo dati AWS Glue. Supporta anche origini dati di streaming Kafka e Kinesis.

- `connectionType` — Il tipo di origine dati. Per informazioni, consultare the section called "Parametri di connessione".
- `connectionOptions`: una stringa di coppie nome-valore JSON che forniscono informazioni aggiuntive per stabilire una connessione con l'origine dati. Per ulteriori informazioni, consulta the section called "Parametri di connessione".

  Un'origine di streaming Kinesis richiede le seguenti opzioni di connessione: `streamARN`, `startingPosition`, `inferSchema` e `classification`.

  Un'origine di streaming Kinesis richiede le seguenti opzioni di connessione: `connectionName`, `topicName`, `startingOffsets`, `inferSchema` e `classification`.
- `transformationContext` — Il contesto di trasformazione associato al sink che i segnalibri di processo utilizzano. Impostato su per impostazione predefinita.
- `pushDownPredicate` — Predicato sulle colonne delle partizioni.

Restituisce il `DataSource`.

Esempio per l'origine di streaming Amazon Kinesis:

```
val kinesisOptions = jsonOptions()
data_frame_datasource0 = glueContext.getSource("kinesis",
 kinesisOptions).getDataFrame()

private def jsonOptions(): JsonOptions = {
    new JsonOptions(
      s"""{"streamARN": "arn:aws:kinesis:eu-central-1:123456789012:stream/
fromOptionsStream",
         |"startingPosition": "TRIM_HORIZON",
         |"inferSchema": "true",
```

```
         |"classification": "json"}""".stripMargin)
}
```

Esempio per l'origine di streaming Kafka:

```
val kafkaOptions = jsonOptions()
val data_frame_datasource0 = glueContext.getSource("kafka",
 kafkaOptions).getDataFrame()

private def jsonOptions(): JsonOptions = {
    new JsonOptions(
      s"""{"connectionName": "ConfluentKafka",
         |"topicName": "kafka-auth-topic",
         |"startingOffsets": "earliest",
         |"inferSchema": "true",
         |"classification": "json"}""".stripMargin)
 }
```

def getSourceWithFormat

```
def getSourceWithFormat( connectionType : String,
                         options : JsonOptions,
                         transformationContext : String = "",
                         format : String = null,
                         formatOptions : JsonOptions = JsonOptions.empty
                       ) : DataSource
```

Crea un Proprietà DataSource che legge dati da un'origine, ad esempio Amazon S3, JDBC o il catalogo dati AWS Glue e in più imposta il formato dei dati archiviati nell'origine.

- `connectionType` – Il tipo dell'origine dati. Per informazioni, consultare the section called “Parametri di connessione”.
- `options`: una stringa di coppie nome/valore JSON che forniscono informazioni aggiuntive per stabilire una connessione all'origine dati. Per informazioni, consultare the section called “Parametri di connessione”.
- `transformationContext` – Il contesto di trasformazione associato al sink che deve essere usato dai segnalibri dei processi. Impostato su per impostazione predefinita.
- `format` – Il formato dei dati archiviati nell'origine. Quando il `connectionType` è "s3", è anche possibile specificare `format`. Le possibilità sono “avro”, “csv”, “grokLog”, “ion”, “json”, “xml”, “parquet” oppure “orc”.

- `formatOptions`: una stringa di coppie nome/valore JSON che forniscono opzioni aggiuntive per l'analisi dei dati nell'origine. Per informazioni, consultare Opzioni del formato dei dati.

Restituisce il `DataSource`.

Examples (Esempi)

Crea un DynamicFrame da un'origine dati, che è un file con valori separati da virgole (CSV) su Amazon S3:

```
val datasource0 = glueContext.getSourceWithFormat(
    connectionType="s3",
    options =JsonOptions(s"""{"paths": [ "s3://csv/nycflights.csv"]}"""),
    transformationContext = "datasource0",
    format = "csv",
    formatOptions=JsonOptions(s"""{"withHeader":"true","separator": ","}""")
    ).getDynamicFrame()
```

Crea un DynamicFrame da un'origine dati, che è un PostgreSQL utilizzando una connessione JDBC:

```
val datasource0 = glueContext.getSourceWithFormat(
    connectionType="postgresql",
    options =JsonOptions(s"""{
      "url":"jdbc:postgresql://databasePostgres-1.rds.amazonaws.com:5432/testdb",
      "dbtable": "public.company",
      "redshiftTmpDir":"",
      "user":"username",
      "password":"password123"
    }"""),
    transformationContext = "datasource0").getDynamicFrame()
```

Crea un DynamicFrame da un'origine dati, che è un MySQL utilizzando una connessione JDBC:

```
 val datasource0 = glueContext.getSourceWithFormat(
    connectionType="mysql",
    options =JsonOptions(s"""{
      "url":"jdbc:mysql://databaseMysql-1.rds.amazonaws.com:3306/testdb",
      "dbtable": "athenatest_nycflights13_csv",
      "redshiftTmpDir":"",
      "user":"username",
      "password":"password123"
```

```
    }"""),
    transformationContext = "datasource0").getDynamicFrame()
```

def getSparkSession

```
def getSparkSession : SparkSession
```

Riceve l'oggetto `SparkSession` associato a questo GlueContext. Utilizza questo oggetto SparkSession per registrare tabelle e UDF da utilizzare con il `DataFrame` creato da DynamicFrames.

Restituisce l'oggetto SparkSession.

def startTransaction

```
def startTransaction(readOnly: Boolean):String
```

Avvia una nuova transazione. Chiama internamente l'API Lake Formation startTransaction.

- `readOnly`: (booleano) indica se questa transazione debba essere di sola lettura o lettura e scrittura. Le scritture effettuate utilizzando un ID transazione di sola lettura verranno rifiutate. Il commit delle transazioni di sola lettura non deve essere eseguito.

Restituisce l'ID transazione.

def commitTransaction

```
def commitTransaction(transactionId: String, waitForCommit: Boolean): Boolean
```

Tenta di eseguire il commit della transazione specificata. `commitTransaction` può restituire prima che la transazione abbia terminato il commit. Chiama internamente l'API Lake Formation commitTransaction.

- `transactionId`: (stringa) la transazione di cui eseguire il commit.
- `waitForCommit`: (booleano) determina se il `commitTransaction` restituisce immediatamente. Il valore di default è true. Se false, `commitTransaction` effettua il polling e aspetta che sia stato eseguito il commit della transazione. Il tempo di attesa è limitato a 1 minuto utilizzando il backoff esponenziale con un massimo di 6 tentativi.

Restituisce un valore booleano per indicare se il commit sia stato eseguito o meno.

def cancelTransaction

```
def cancelTransaction(transactionId: String): Unit
```

Tenta di annullare la transazione specificata. Chiama internamente l'API Lake Formation CancelTransaction.

- `transactionId`: (stringa) la transazione da annullare.

Restituisce un'eccezione `TransactionCommittedException` se è stato precedentemente eseguito il commit della transazione.

def this

```
def this( sc : SparkContext,
          minPartitions : Int,
          targetPartitions : Int )
```

Crea un oggetto `GlueContext` utilizzando lo `SparkContext` specificato, le partizioni minime e quelle target.

- `sc` — Il carattere `SparkContext`.
- `minPartitions` — Il numero minimo di partizioni.
- `targetPartitions` — Il numero target di partizioni.

Restituisce il `GlueContext`.

def this

```
def this( sc : SparkContext )
```

Crea un oggetto `GlueContext` con il `SparkContext` fornito. Imposta il numero minimo di partizioni a 10 e a le partizioni target a 20.

- `sc` — Il carattere `SparkContext`.

Restituisce il `GlueContext`.

def this

```
def this( sparkContext : JavaSparkContext )
```

Crea un oggetto `GlueContext` con il `JavaSparkContext` fornito. Imposta il numero minimo di partizioni a 10 e a le partizioni target a 20.

- `sparkContext` — Il carattere `JavaSparkContext`.

Restituisce il `GlueContext`.

MappingSpec

Pacchetto: com.amazonaws.services.glue

Case Class MappingSpec

```
case class MappingSpec( sourcePath: SchemaPath,
                        sourceType: DataType,
                        targetPath: SchemaPath,
                        targetType: DataTyp
                       ) extends Product4[String, String, String, String] {
  override def _1: String = sourcePath.toString
  override def _2: String = ExtendedTypeName.fromDataType(sourceType)
  override def _3: String = targetPath.toString
  override def _4: String = ExtendedTypeName.fromDataType(targetType)
}
```

- `sourcePath`: il `SchemaPath` del campo di origine.
- `sourceType`: il `DataType` del campo di origine.
- `targetPath`: il `SchemaPath` del campo di destinazione.
- `targetType`: il `DataType` del campo di destinazione.

Un `MappingSpec` specifica una mappatura da un percorso di origine e un tipo di dati di origine a un percorso di destinazione e un tipo di dati di destinazione. Il valore al percorso di origine nel frame di origine viene visualizzato nel frame di destinazione presso il percorso di destinazione. Il tipo di dati di origine è trasmesso al tipo di dati di destinazione.

Si estende da `Product4` in modo che sia possibile gestire qualsiasi `Product4` all'interno dell'interfaccia `applyMapping`.

Oggetto MappingSpec

```
object MappingSpec
```

L'oggetto `MappingSpec` dispone dei seguenti membri:

Val orderingByTarget

```
val orderingByTarget: Ordering[MappingSpec]
```

Applicazione di def

```
def apply( sourcePath : String,
           sourceType : DataType,
           targetPath : String,
           targetType : DataType
         ) : MappingSpec
```

Crea un `MappingSpec`.

- `sourcePath`: rappresentazione di stringa del percorso di origine.
- `sourceType`: sorgente `DataType`.
- `targetPath`: rappresentazione di stringa del percorso di destinazione.
- `targetType`: destinazione `DataType`.

Restituisce un `MappingSpec`.

Applicazione di def

```
def apply( sourcePath : String,
           sourceTypeString : String,
           targetPath : String,
           targetTypeString : String
         ) : MappingSpec
```

Crea un `MappingSpec`.

- `sourcePath`: rappresentazione di stringa del percorso di origine.
- `sourceType`: rappresentazione di stringa del tipo di dati di origine.
- `targetPath`: rappresentazione di stringa del percorso di destinazione.
- `targetType`: rappresentazione di stringa del tipo di dati di destinazione.

Restituisce un MappingSpec.

Applicazione di def

```
def apply( product : Product4[String, String, String, String] ) : MappingSpec
```

Crea un `MappingSpec`.

- `product`: il `Product4` del percorso di origine, il tipo di dati di origine, il percorso di destinazione e il tipo di dati di destinazione.

Restituisce una `MappingSpec`.

API Scala ResolveSpec di AWS Glue

Argomenti

- Oggetto ResolveSpec
- Case class ResolveSpec

Pacchetto: com.amazonaws.services.glue

Oggetto ResolveSpec

ResolveSpec

```
object ResolveSpec
```

Def

```
def apply( path : String,
           action : String
```

```
        ) : ResolveSpec
```

Crea un `ResolveSpec`.

- `path`: rappresentazione di stringa del campo di scelta che deve essere risolto.
- `action`: un'operazione di risoluzione. L'operazione può essere una delle seguenti: `Project`, `KeepAsStruct` o `Cast`.

Restituisce il `ResolveSpec`.

Def

```
def apply( product : Product2[String, String] ) : ResolveSpec
```

Crea un `ResolveSpec`.

- `product` — `Product2` di: percorso di origine, operazione di risoluzione.

Restituisce il `ResolveSpec`.

Case class ResolveSpec

```
case class ResolveSpec extends Product2[String, String]  (
           path : SchemaPath,
           action : String )
```

Crea un `ResolveSpec`.

- `path`: `SchemaPath` del campo di scelta che deve essere risolto.
- `action`: un'operazione di risoluzione. L'operazione può essere una delle seguenti: `Project`, `KeepAsStruct` o `Cast`.

Metodi def ResolveSpec

```
def _1 : String
```

```
def _2 : String
```

API Scala ArrayNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class ArrayNode

ArrayNode

```
case class ArrayNode extends DynamicNode  (
           value : ArrayBuffer[DynamicNode] )
```

Metodi def ArrayNode

```
def add( node : DynamicNode )
```

```
def clone
```

```
def equals( other : Any )
```

```
def get( index : Int ) : Option[DynamicNode]
```

```
def getValue
```

```
def hashCode : Int
```

```
def isEmpty : Boolean
```

```
def nodeType
```

```
def remove( index : Int )
```

```
def this
```

```
def toIterator : Iterator[DynamicNode]
```

```
def toJson : String
```

```
def update( index : Int,
            node : DynamicNode )
```

API Scala BinaryNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class BinaryNode

BinaryNode

```
case class BinaryNode extends ScalarNode(value, TypeCode.BINARY)  (
           value : Array[Byte] )
```

Campi val BinaryNode

- `ordering`

Metodi def BinaryNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

API Scala BooleanNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class BooleanNode

BooleanNode

```
case class BooleanNode extends ScalarNode(value, TypeCode.BOOLEAN)  (
           value : Boolean )
```

Campi val BooleanNode

- `ordering`

Metodi def BooleanNode

```
def equals( other : Any )
```

API Scala ByteNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class ByteNode

ByteNode

```
case class ByteNode extends ScalarNode(value, TypeCode.BYTE)  (
           value : Byte )
```

Campi val ByteNode

- `ordering`

Metodi def ByteNode

```
def equals( other : Any )
```

API Scala DateNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class DateNode

DateNode

```
case class DateNode extends ScalarNode(value, TypeCode.DATE)  (
           value : Date )
```

Campi val DateNode

- `ordering`

Metodi def DateNode

```
def equals( other : Any )
```

```
def this( value : Int )
```

API Scala DecimalNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class DecimalNode

DecimalNode

```
case class DecimalNode extends ScalarNode(value, TypeCode.DECIMAL)  (
           value : BigDecimal )
```

Campi val DecimalNode

- `ordering`

Metodi def DecimalNode

```
def equals( other : Any )
```

```
def this( value : Decimal )
```

API Scala DoubleNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class DoubleNode

DoubleNode

```
case class DoubleNode extends ScalarNode(value, TypeCode.DOUBLE)  (
           value : Double )
```

Campi val DoubleNode

- `ordering`

Metodi def DoubleNode

```
def equals( other : Any )
```

API Scala DynamicNode di AWS Glue

Argomenti

- Classe DynamicNode
- Oggetto DynamicNode

Pacchetto: com.amazonaws.services.glue.types

Classe DynamicNode

DynamicNode

```
class DynamicNode extends Serializable with Cloneable
```

Metodi def DynamicNode

```
def getValue : Any
```

Ottenere un valore normale associato al record corrente:

```
def nodeType : TypeCode
```

```
def toJson : String
```

Metodo per il debug:

```
def toRow( schema : Schema,
           options : Map[String, ResolveOption]
         ) : Row
```

```
def typeName : String
```

Oggetto DynamicNode

DynamicNode

```
object DynamicNode
```

Metodi def DynamicNode

```
def quote( field : String,
           useQuotes : Boolean
         ) : String
```

```
def quote( node : DynamicNode,
           useQuotes : Boolean
         ) : String
```

Classe EvaluateDataQuality

AWS Glue Data Quality è in versione di anteprima per AWS Glue ed è soggetto a modifica.

Pacchetto: com.amazonaws.services.glue.dq

```
object EvaluateDataQuality
```

Applicazione di def

```
def apply(frame: DynamicFrame,
            ruleset: String,
            publishingOptions: JsonOptions = JsonOptions.empty): DynamicFrame
```

Valuta un set di regole di qualità dei dati rispetto ai dati in un `DynamicFrame` e restituisce un nuovo `DynamicFrame` con i risultati della valutazione. Per ulteriori informazioni su AWS Glue Data Quality, consulta AWS Glue Qualità dei dati.

- `frame`: il `DynamicFrame` di cui desideri valutare la qualità dei dati.
- `ruleset`: un set di regole del Data Quality Definition Language (DQDL) in formato stringa. Per ulteriori informazioni su DQDL, consulta la guida di Riferimento a Data Quality Definition Language (DQDL).

- `publishingOptions`: un dizionario che specifica le seguenti opzioni per la pubblicazione dei risultati e dei parametri di valutazione:
  - `dataQualityEvaluationContext`: una stringa che specifica lo spazio dei nomi in cui AWS Glue deve pubblicare i parametri Amazon CloudWatch e i risultati della qualità dei dati. I parametri aggregati vengono visualizzati in CloudWatch, mentre i risultati completi vengono visualizzati nell'interfaccia di AWS Glue Studio.
    - Campo obbligatorio: no
    - Valore predefinito: `default_context`
  - `enableDataQualityCloudWatchMetrics`: specifica se i risultati della valutazione della qualità dei dati devono essere pubblicati su CloudWatch. Uno spazio dei nomi per i parametri viene specificato utilizzando l'opzione `dataQualityEvaluationContext`.
    - Campo obbligatorio: no
    - Valore predefinito: False
  - `enableDataQualityResultsPublishing`: specifica se i risultati della qualità dei dati devono essere visibili nella scheda Data Quality (Qualità dei dati) nell'interfaccia di AWS Glue Studio.
    - Campo obbligatorio: no
    - Valore predefinito: true
  - `resultsS3Prefix`: specifica la posizione Amazon S3 in cui AWS Glue può scrivere i risultati della valutazione della qualità dei dati.
    - Campo obbligatorio: no
    - Valore predefinito: "" (stringa vuota)

Esempio

Il codice di esempio seguente dimostra come valutare la qualità dei dati per un `DynamicFrame` prima di eseguire una trasformazione `SelectFields`. Lo script verifica che tutte le regole di qualità dei dati siano rispettate prima di tentare la trasformazione.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
```

```
import com.amazonaws.services.glue.dq.EvaluateDataQuality

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Create DynamicFrame with data
    val Legislators_Area = glueContext.getCatalogSource(database="legislators",
 tableName="areas_json", transformationContext="S3bucket_node1").getDynamicFrame()

    // Define data quality ruleset
    val DQ_Ruleset = """
      Rules = [ColumnExists "id"]
    """

    // Evaluate data quality
    val DQ_Results = EvaluateDataQuality.apply(frame=Legislators_Area,
 ruleset=DQ_Ruleset, publishingOptions=JsonOptions("""{"dataQualityEvaluationContext":
 "Legislators_Area", "enableDataQualityMetrics": "true",
 "enableDataQualityResultsPublishing": "true"}"""))
    assert(DQ_Results.filter(_.getField("Outcome").contains("Failed")).count == 0,
 "Failing DQ rules for Legislators_Area caused the job to fail.")

    // Script generated for node Select Fields
    val SelectFields_Results = Legislators_Area.selectFields(paths=Seq("id", "name"),
 transformationContext="Legislators_Area")

    Job.commit()
  }
}
```

API Scala FloatNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class FloatNode

FloatNode

```
case class FloatNode extends ScalarNode(value, TypeCode.FLOAT)  (
```

```
            value : Float )
```

Campi val FloatNode

- `ordering`

Metodi def FloatNode

```
def equals( other : Any )
```

Classe FillMissingValues

Pacchetto: com.amazonaws.services.glue.ml

```
object FillMissingValues
```

Applicazione di def

```
def apply(frame: DynamicFrame,
          missingValuesColumn: String,
          outputColumn: String = "",
          transformationContext: String = "",
          callSite: CallSite = CallSite("Not provided", ""),
          stageThreshold: Long = 0,
          totalThreshold: Long = 0): DynamicFrame
```

Riempie i valori mancanti di un frame dinamico in una colonna specificata e restituisce un frame dinamico con stime in una nuova colonna. Per le righe senza valori mancanti, il valore della colonna specificato viene duplicato nella nuova colonna.

- `frame` — Il DynamicFrame i cui inserire i valori mancanti. Campo obbligatorio.
- `missingValuesColumn` — La colonna contenente valori mancanti (valori `null` e stringhe vuote). Campo obbligatorio.
- `outputColumn` — Il nome della nuova colonna che conterrà i valori stimati per tutte le righe il cui valore era mancante. Facoltativo; il valore predefinito è il valore di `missingValuesColumn` con suffisso formato da `"_filled"`.
- `transformationContext` — Una stringa univoca usata per identificare le informazioni sullo stato (facoltativo).

- `callSite` — Usato per fornire informazioni sul contesto per la segnalazione degli errori (facoltativo).
- `stageThreshold` — Il numero massimo di errori che si possono verificare nella trasformazione prima che venga arrestata (facoltativo, il valore di default è zero).
- `totalThreshold` — Il numero massimo di errori che si possono verificare in generale prima che l'elaborazione venga arrestata (facoltativo, il valore di default è zero).

Restituisce un nuovo frame dinamico con una colonna aggiuntiva che contiene stime per le righe con valori mancanti e il valore attuale per le altre righe.

Classe FindMatches

Pacchetto: com.amazonaws.services.glue.ml

```
object FindMatches
```

Applicazione di def

```
def apply(frame: DynamicFrame,
            transformId: String,
            transformationContext: String = "",
            callSite: CallSite = CallSite("Not provided", ""),
            stageThreshold: Long = 0,
            totalThreshold: Long = 0,
            enforcedMatches: DynamicFrame = null): DynamicFrame,
   computeMatchConfidenceScores: Boolean
```

Trova le corrispondenze in un frame di input e restituisci un nuovo frame con una nuova colonna contenente un ID univoco per gruppo di corrispondenza.

- `frame` — Il DynamicFrame in cui trovare le corrispondenze. Campo obbligatorio.
- `transformId` — Un ID univoco associato alla trasformazione FindMatches da applicare al frame di input. Campo obbligatorio.
- `transformationContext` — Identificatore per questo `DynamicFrame`. Il `transformationContext` viene usato come chiave per lo stato dei segnalibro di processo che viene mantenuto tra esecuzioni. Facoltativo.
- `callSite` — Usato per fornire informazioni sul contesto per la segnalazione degli errori. Questi valori vengono impostati automaticamente durante la chiamata da Python. Facoltativo.

- `stageThreshold` — Il numero massimo di record di errore consentiti nel calcolo di questo `DynamicFrame` prima di generare un'eccezione, esclusi i record presenti nell'oggetto `DynamicFrame` precedente. Facoltativo. Il valore di default è zero.
- `totalThreshold` — Il numero massimo di record di errore totali prima di generare un'eccezione, inclusi quelli dei frame precedenti. Facoltativo. Il valore di default è zero.
- `enforcedMatches` — Il frame per le corrispondenze applicate. Facoltativo. Il valore predefinito è `null`.
- `computeMatchConfidenceScores`: un valore booleano che indica se calcolare un punteggio di confidenza per ciascun gruppo di record corrispondenti. Facoltativo. Il valore di default è false.

Restituisce un nuovo frame dinamico con un identificatore univoco assegnato a ciascun gruppo di record corrispondenti.

Classe FindIncrementalMatches

Pacchetto: com.amazonaws.services.glue.ml

```
object FindIncrementalMatches
```

Applicazione di def

```
apply(existingFrame: DynamicFrame,
            incrementalFrame: DynamicFrame,
            transformId: String,
            transformationContext: String = "",
            callSite: CallSite = CallSite("Not provided", ""),
            stageThreshold: Long = 0,
            totalThreshold: Long = 0,
            enforcedMatches: DynamicFrame = null): DynamicFrame,
   computeMatchConfidenceScores: Boolean
```

Trova le corrispondenze in frame incrementali esistenti e restituisci un nuovo frame con una colonna contenente un ID univoco per gruppo di corrispondenza.

- `existingframe` — Un frame esistente a cui è stato assegnato un ID corrispondente per ogni gruppo. Campo obbligatorio.
- `incrementalframe` — Un frame incrementale utilizzato per trovare corrispondenze rispetto al frame esistente. Campo obbligatorio.

- `transformId` — Un ID univoco associato alla trasformazione FindIncrementalMatches da applicare ai frame di input. Campo obbligatorio.
- `transformationContext` — Identificatore per questo `DynamicFrame`. Il `transformationContext` viene usato come chiave per lo stato dei segnalibro di processo che viene mantenuto tra esecuzioni. Facoltativo.
- `callSite` — Usato per fornire informazioni sul contesto per la segnalazione degli errori. Questi valori vengono impostati automaticamente durante la chiamata da Python. Facoltativo.
- `stageThreshold` — Il numero massimo di record di errore consentiti nel calcolo di questo `DynamicFrame` prima di generare un'eccezione, esclusi i record presenti nell'oggetto `DynamicFrame` precedente. Facoltativo. Il valore di default è zero.
- `totalThreshold` — Il numero massimo di record di errore totali prima di generare un'eccezione, inclusi quelli dei frame precedenti. Facoltativo. Il valore di default è zero.
- `enforcedMatches` — Il frame per le corrispondenze applicate. Facoltativo. Il valore predefinito è `null`.
- `computeMatchConfidenceScores`: un valore booleano che indica se calcolare un punteggio di confidenza per ciascun gruppo di record corrispondenti. Facoltativo. Il valore di default è false.

Restituisce un nuovo frame dinamico con un identificatore univoco assegnato a ciascun gruppo di record corrispondenti.

API Scala IntegerNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class IntegerNode

IntegerNode

```
case class IntegerNode extends ScalarNode(value, TypeCode.INT)  (
           value : Int )
```

Campi val IntegerNode

- `ordering`

Metodi def IntegerNode

```
def equals( other : Any )
```

API Scala LongNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case Class LongNode

LongNode

```
case class LongNode extends ScalarNode(value, TypeCode.LONG)  (
           value : Long )
```

Campi val LongNode

- `ordering`

Metodi def LongNode

```
def equals( other : Any )
```

API Scala MapLikeNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Classe MapLikeNode

MapLikeNode

```
class MapLikeNode extends DynamicNode  (
           value : mutable.Map[String, DynamicNode] )
```

Metodi def MapLikeNode

```
def clear : Unit
```

```
def get( name : String ) : Option[DynamicNode]
```

```
def getValue
```

```
def has( name : String ) : Boolean
```

```
def isEmpty : Boolean
```

```
def put( name : String,
         node : DynamicNode
       ) : Option[DynamicNode]
```

```
def remove( name : String ) : Option[DynamicNode]
```

```
def toIterator : Iterator[(String, DynamicNode)]
```

```
def toJson : String
```

```
def toJson( useQuotes : Boolean ) : String
```

Esempio: considerato questo JSON:

```
{"foo": "bar"}
```

Se `useQuotes == true`, `toJson` genera `{"foo": "bar"}`. Se `useQuotes == false`, `toJson` genera `{foo: bar}` @return.

API Scala MapNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class MapNode

MapNode

```
case class MapNode extends MapLikeNode(value)  (
           value : mutable.Map[String, DynamicNode] )
```

Metodi def MapNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

```
def nodeType
```

```
def this
```

API Scala NullNode di AWS Glue

Argomenti

- Classe NullNode
- Oggetto del case NullNode

Pacchetto: com.amazonaws.services.glue.types

Classe NullNode

NullNode

```
class NullNode
```

Oggetto del case NullNode

NullNode

```
case object NullNode extends NullNode
```

API Scala ObjectNode di AWS Glue

Argomenti

- Oggetto ObjectNode
- Case class ObjectNode

Pacchetto: com.amazonaws.services.glue.types

Oggetto ObjectNode

ObjectNode

```
object ObjectNode
```

Metodi def ObjectNode

```
def apply( frameKeys : Set[String],
           v1 : mutable.Map[String, DynamicNode],
           v2 : mutable.Map[String, DynamicNode],
           resolveWith : String
         ) : ObjectNode
```

Case class ObjectNode

ObjectNode

```
case class ObjectNode extends MapLikeNode(value)  (
           val value : mutable.Map[String, DynamicNode] )
```

Metodi def ObjectNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

```
def nodeType
```

```
def this
```

API Scala ScalarNode di AWS Glue

Argomenti

- Classe ScalarNode

- Oggetto ScalarNode

Pacchetto: com.amazonaws.services.glue.types

Classe ScalarNode

ScalarNode

```
class ScalarNode extends DynamicNode  (
           value : Any,
           scalarType : TypeCode )
```

Metodi def ScalarNode

```
def compare( other : Any,
             operator : String
           ) : Boolean
```

```
def getValue
```

```
def hashCode : Int
```

```
def nodeType
```

```
def toJson
```

Oggetto ScalarNode

ScalarNode

```
object ScalarNode
```

Metodi def ScalarNode

```
def apply( v : Any ) : DynamicNode
```

```
def compare( tv : Ordered[T],
             other : T,
             operator : String
```

```
            ) : Boolean
```

```
def compareAny( v : Any,
                y : Any,
                o : String )
```

```
def withEscapedSpecialCharacters( jsonToEscape : String ) : String
```

API Scala ShortNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Case class ShortNode

ShortNode

```
case class ShortNode extends ScalarNode(value, TypeCode.SHORT)  (
            value : Short )
```

Campi val ShortNode

- `ordering`

Metodi def ShortNode

```
def equals( other : Any )
```

API Scala StringNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Classe del caso StringNode

StringNode

```
case class StringNode extends ScalarNode(value, TypeCode.STRING)  (
            value : String )
```

Campi valore StringNode

- `ordering`

Metodi definizione StringNode

```
def equals( other : Any )
```

```
def this( value : UTF8String )
```

API Scala TimestampNode di AWS Glue

Pacchetto: com.amazonaws.services.glue.types

Classe del caso TimestampNode

TimestampNode

```
case class TimestampNode extends ScalarNode(value, TypeCode.TIMESTAMP)  (
           value : Timestamp )
```

Campi di valore TimestampNode

- `ordering`

Metodi di definizione TimestampNode

```
def equals( other : Any )
```

```
def this( value : Long )
```

API Scala GlueArgParser di AWS Glue

Pacchetto: com.amazonaws.services.glue.util

Oggetto GlueArgParser

GlueArgParser

```
object GlueArgParser
```

Questo oggetto è rigorosamente coerente con la versione Python di `utils.getResolvedOptions` nel pacchetto `AWSGlueDataplanePython`.

Metodi def GlueArgParser

```
def getResolvedOptions( args : Array[String],
                        options : Array[String]
                      ) : Map[String, String]
```

```
def initParser( userOptionsSet : mutable.Set[String] ) : ArgumentParser
```

Example Recupero degli argomenti trasmessi a un processo

Per recuperare gli argomenti del processo, è possibile utilizzare il metodo `getResolvedOptions`. Esamina l'esempio seguente, che recupera un argomento del processo denominato `aws_region`.

```
val args = GlueArgParser.getResolvedOptions(sysArgs,
 Seq("JOB_NAME","aws_region").toArray)
Job.init(args("JOB_NAME"), glueContext, args.asJava)
val region = args("aws_region")
println(region)
```

API processo Scala di AWS Glue

Pacchetto: com.amazonaws.services.glue.util

Oggetto del processo

Processo

```
object Job
```

Metodi def del processo

```
def commit
```

```
def init( jobName : String,
          glueContext : GlueContext,
          args : java.util.Map[String, String] = Map[String, String]().asJava
        ) : this.type
```

```
def init( jobName : String,
```

```
          glueContext : GlueContext,
          endpoint : String,
          args : java.util.Map[String, String]
        ) : this.type
```

```
def isInitialized
```

```
def reset
```

```
def runId
```

Funzionalità e ottimizzazioni per la programmazione degli script ETL di AWS Glue per Spark

Le sezioni seguenti descrivono le tecniche e i valori che si applicano generalmente alla programmazione ETL (estrazione, trasformazione e caricamento) di AWS Glue per Spark in qualsiasi linguaggio.

Argomenti

- Tipi e opzioni di connessione per ETL in AWS Glue per Spark
- Opzioni del formato dati per input e output in AWS Glue per Spark
- Supporto del catalogo dati di AWS Glue per i processi Spark SQL
- Utilizzo di segnalibri di processo
- Utilizzo del rilevamento dei dati sensibili fuori da AWS Glue Studio
- API Visual Job di AWS Glue

Tipi e opzioni di connessione per ETL in AWS Glue per Spark

In AWS Glue per Spark, diversi metodi e trasformazioni PySpark e Scala specificano il tipo di connessione utilizzando un parametro `connectionType`. Specificano le opzioni di connessione utilizzando un parametro `connectionOptions` o `options`.

Il parametro `connectionType` può assumere i valori indicati nella tabella seguente. I valori dei parametri associati `connectionOptions` (o `options`) per ciascun tipo sono documentati nelle sezioni seguenti. Salvo indicazione contraria, i parametri si applicano quando la connessione viene utilizzata come sorgente o sink.

Per il codice di esempio che illustra l'impostazione e l'utilizzo delle opzioni di connessione, consulta la home page per ogni tipo di connessione.

`connectionType`	Si connette a
dynamodb	Amazon DynamoDB database
kinesis	Flusso di dati Amazon Kinesis
s3	Amazon S3
documentdb	Amazon DocumentDB (con compatibilità MongoDB) database
opensearch	Servizio OpenSearch di Amazon.
redshift	Database Amazon Redshift
kafka	Kafka o Amazon Managed Streaming for Apache Kafka
azurecosmos	Azure Cosmos per NoSQL.
azuresql	Azure SQL.
bigquery	Google BigQuery.
mongodb	Database MongoDB, incluso MongoDB Atlas.
sqlserver	Microsoft SQL Server database (vedere Connessioni JDBC)
mysql	MySQL database (vedere Connessioni JDBC)
oracle	Oracle database (vedere Connessioni JDBC)
postgresql	PostgreSQL database (vedere Connessioni JDBC)
saphana	SAP HANA.
snowflake	Data lake Snowflake
teradata	Teradata Vantage.
vertica	Vertica.

connectionType	Si connette a
personalizzato.*	Archivi dati Spark, Athena o JDBC (consulta Valori di personalizzazione e connectionType Marketplace AWS
marketplace.*	Archivi dati Spark, Athena o JDBC (consulta Valori di personalizzazione e connectionType Marketplace AWS)

Connessioni a DynamoDB

Puoi utilizzare AWS Glue per Spark per la lettura e la scrittura su tabelle in DynamoDB per AWS Glue. Puoi connetterti a DynamoDB utilizzando le autorizzazioni IAM allegate al processo AWS Glue. AWS Glue supporta la scrittura di dati nella tabella DynamoDB di un altro account AWS. Per ulteriori informazioni, consulta the section called “Accesso multi-account in più regioni alle tabelle DynamoDB”.

Oltre al connettore ETL per DynamoDB AWS Glue, è possibile leggere da DynamoDB utilizzando il connettore di esportazione DynamoDB, che richiama una richiesta DynamoDB `ExportTableToPointInTime` e la archivia in una posizione Amazon S3 da te fornita, nel formato JSON di DynamoDB. AWS Glue crea quindi un oggetto DynamicFrame leggendo i dati dalla posizione di esportazione di Amazon S3.

Il writer DynamoDB è disponibile nella versione 1.0 di AWS Glue Glue o nelle versioni successive. Il connettore di esportazione DynamoDB per AWS Glue è disponibile nella versione 2.0 di AWS Glue o nelle versioni successive.

Per ulteriori informazioni su DynamoDB, consulta la documentazione di Amazon DynamoDB.

> **Note**
>
> Il lettore DynamoDB ETL non supporta filtri o predicati pushdown.

Configurazione delle connessioni a MongoDB

Per connetterti a DynamoDB da AWS Glue, concedi al ruolo IAM associato al processo AWS Glue l'autorizzazione a interagire con DynamoDB. Per ulteriori informazioni sulle autorizzazioni necessarie per leggere o scrivere da DynamoDB, consulta Operazioni, risorse e chiavi di condizione per Amazon DynamoDB nella documentazione di IAM.

Nelle seguenti situazioni, potresti aver bisogno di una configurazione aggiuntiva:

- Quando utilizzi il connettore di esportazione DynamoDB, devi configurare IAM in modo che il processo possa richiedere l'esportazione di tabelle DynamoDB. Inoltre, dovrai identificare un bucket Amazon S3 per l'esportazione e fornire le autorizzazioni appropriate in IAM per fare in modo che DynamoDB sia in grado di scrivere sul bucket e il processo AWS Glue sia in grado di leggere da esso. Per ulteriori informazioni, consulta Richiesta di esportazione di una tabella in DynamoDB.
- Se il processo AWS Glue dispone di requisiti di connettività Amazon VPC specifici, usa il tipo di connessione `NETWORK` di AWS Glue per fornire opzioni di rete. Poiché l'accesso a DynamoDB è autorizzato da IAM, non è necessario un tipo di connessione DynamoDB per AWS Glue.

Lettura e scrittura su DynamoDB

Gli esempi di codice seguenti mostrano come leggere (tramite il connettore ETL) e scrivere tabelle DynamoDB. Mostrano la lettura da una tabella e la scrittura su un'altra tabella.

Python

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={"dynamodb.input.tableName": test_source,
        "dynamodb.throughput.read.percent": "1.0",
        "dynamodb.splits": "100"
    }
)
print(dyf.getNumPartitions())

glue_context.write_dynamic_frame_from_options(
    frame=dyf,
    connection_type="dynamodb",
```

```
    connection_options={"dynamodb.output.tableName": test_sink,
        "dynamodb.throughput.write.percent": "1.0"
    }
)

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.input.tableName" -> test_source,
        "dynamodb.throughput.read.percent" -> "1.0",
        "dynamodb.splits" -> "100"
      ))
    ).getDynamicFrame()

    print(dynamicFrame.getNumPartitions())

    val dynamoDbSink: DynamoDbDataSink =  glueContext.getSinkWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.output.tableName" -> test_sink,
        "dynamodb.throughput.write.percent" -> "1.0"
      ))
    ).asInstanceOf[DynamoDbDataSink]
```

```
    dynamoDbSink.writeDynamicFrame(dynamicFrame)

    Job.commit()
  }

}
```

Utilizzo del connettore di esportazione DynamoDB

Il connettore di esportazione ha prestazioni migliori rispetto al connettore ETL quando le dimensioni della tabella DynamoDB sono superiori a 80 GB. Inoltre, dato che la richiesta di esportazione è condotta al di fuori dei processi Spark in un processo AWS Glue, puoi abilitare Scalabilità automatica di processi Glue AWS per salvare l'utilizzo della DPU durante la richiesta di esportazione. Con il connettore di esportazione, non è inoltre necessario configurare il numero di divisioni per il parallelismo dell'esecutore Spark o la percentuale di lettura del throughput DynamoDB.

> **Note**
>
> DynamoDB ha requisiti specifici per richiamare le richieste `ExportTableToPointInTime`. Per ulteriori informazioni, consulta Richiesta di esportazione di una tabella in DynamoDB. Ad esempio, è necessario abilitare Point-in-Time-Restore (PITR) sulla tabella per utilizzare questo connettore. Il connettore DynamoDB supporta la crittografia AWS KMS per le esportazioni DynamoDB in Amazon S3. Specificando la configurazione di sicurezza nella configurazione del processo AWS Glue, si abilita la crittografia AWS KMS per un'esportazione DynamoDB. La chiave KMS deve essere nella stessa regione del bucket Amazon S3.
> Tieni presente che si applicano costi aggiuntivi per l'esportazione DynamoDB e i costi di storage Amazon S3. I dati esportati in Amazon S3 persistono al termine dell'esecuzione di un processo in modo da poterli riutilizzare senza ulteriori esportazioni DynamoDB. Per utilizzare questo connettore, è necessario che il ripristino point-in-time (PITR) sia abilitato per la tabella. Il connettore ETL DynamoDB o il connettore di esportazione non supportano filtri o predicati pushdown da applicare all'origine DynamoDB.

Gli esempi di codice seguenti mostrano come leggere (tramite il connettore di esportazione) e stampare il numero di partizioni.

Python

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": test_source,
        "dynamodb.s3.bucket": bucket_name,
        "dynamodb.s3.prefix": bucket_prefix,
        "dynamodb.s3.bucketOwner": account_id_of_bucket,
    }
)
print(dyf.getNumPartitions())

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
```

```
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> test_source,
        "dynamodb.s3.bucket" -> bucket_name,
        "dynamodb.s3.prefix" -> bucket_prefix,
        "dynamodb.s3.bucketOwner" -> account_id_of_bucket,
      ))
    ).getDynamicFrame()

    print(dynamicFrame.getNumPartitions())

    Job.commit()
  }

}
```

Questi esempi mostrano come leggere (tramite il connettore di esportazione) e stampare il numero di partizioni da una Tabella Glue Data Catalog AWS che ha una classificazione dynamodb:

Python

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dynamicFrame = glue_context.create_dynamic_frame.from_catalog(
        database=catalog_database,
        table_name=catalog_table_name,
        additional_options={
            "dynamodb.export": "ddb",
            "dynamodb.s3.bucket": s3_bucket,
```

```
            "dynamodb.s3.prefix": s3_bucket_prefix
        }
    )
print(dynamicFrame.getNumPartitions())

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getCatalogSource(
        database = catalog_database,
        tableName = catalog_table_name,
        additionalOptions = JsonOptions(Map(
            "dynamodb.export" -> "ddb",
            "dynamodb.s3.bucket" -> s3_bucket,
            "dynamodb.s3.prefix" -> s3_bucket_prefix
        ))
    ).getDynamicFrame()
    print(dynamicFrame.getNumPartitions())
)
```

Semplificazione dell'utilizzo del JSON di esportazione DynamoDB

Le esportazioni di DynamoDB con il connettore di esportazione DynamoDB AWS Glue possono risultare in file JSON con specifiche strutture annidate. Per ulteriori informazioni, consulta Oggetti

dati. AWS Glue fornisce una trasformazione DynamicFrame, che può snidificare tali strutture in una forma più facile da usare per le applicazioni a valle.

La trasformazione può essere invocata in uno dei due modi possibili. Puoi impostare l'opzione di connessione `"dynamodb.simplifyDDBJson"` sul valore `"true"` quando effettui una chiamata a un metodo per leggere da DynamoDB. Puoi anche effettuare una chiamata alla trasformazione come metodo disponibile in modo indipendente nella libreria di AWS Glue.

Prendi in considerazione lo schema seguente generato da un'esportazione DynamoDB:

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

La trasformazione `simplifyDDBJson` semplificherà questo processo in:

```
root
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
```

```
|    |    |-- packageName: string
|    |    |-- updatedAt: string
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
|-- nullValue: null
```

> **Note**
>
> `simplifyDDBJson` è disponibile solo in AWS Glue 3.0 e versioni successive. Per semplificare il JSON di esportazione DynamoDB è disponibile anche la trasformazione `unnestDDBJson`. Incoraggiamo gli utenti a passare da `unnestDDBJson` a `simplifyDDBJson`.

Configurazione del parallelismo nelle operazioni DynamoDB

Per migliorare le prestazioni, è possibile regolare alcuni parametri disponibili per il connettore DynamoDB. L'obiettivo dell'ottimizzazione dei parametri di parallelismo è massimizzare l'utilizzo dei worker AWS Glue di cui è stato effettuato il provisioning. Se hai bisogno di prestazioni più elevate, ti consigliamo quindi di impiegare la scalabilità orizzontale per il processo aumentando il numero di DPU.

Puoi modificare il parallelismo in un'operazione di lettura di DynamoDB con il parametro `dynamodb.splits` quando utilizzi il connettore ETL. La lettura con il connettore di esportazione non richiede la configurazione del numero di divisioni per il parallelismo dell'esecutore Spark. Puoi modificare il parallelismo in un'operazione di scrittura DynamoDB con `dynamodb.output.numParallelTasks`.

Lettura con il connettore ETL per DynamoDB

Ti consigliamo di calcolare `dynamodb.splits` in base al numero massimo di worker impostato nella configurazione del lavoro e al seguente calcolo `numSlots`. In caso di dimensionamento automatico, il numero effettivo di worker disponibili potrebbe variare al di sotto di tale limite. Per ulteriori informazioni sull'impostazione del numero massimo di worker, consulta Numero di worker (`NumberOfWorkers`) in the section called “Aggiunta di processi”.

- `numExecutors = NumberOfWorkers - 1`

  Per il contesto, un esecutore è riservato per il driver Spark; altri esecutori sono utilizzati per elaborare i dati.
- `numSlotsPerExecutor =`

  AWS Glue 3.0 and later versions
  - 4: se `WorkerType` è `G.1X`.
  - 8: se `WorkerType` è `G.2X`.
  - 16: se `WorkerType` è `G.4X`.
  - 32: se `WorkerType` è `G.8X`.

  AWS Glue 2.0 and legacy versions
  - 8: se `WorkerType` è `G.1X`.
  - 16: se `WorkerType` è `G.2X`.
- `numSlots = numSlotsPerExecutor * numExecutors`

Ti consigliamo di impostare `dynamodb.splits` sul numero di slot disponibili, `numSlots`.

Scrittura su DynamoDB

Il parametro `dynamodb.output.numParallelTasks` viene utilizzato per determinare il valore WCU per ogni attività Spark, utilizzando il seguente calcolo:

`permittedWcuPerTask = ( TableWCU * dynamodb.throughput.write.percent ) / dynamodb.output.numParallelTasks`

Il writer DynamoDB funzionerà al meglio se la configurazione rappresenta accuratamente il numero di attività Spark che scrivono su DynamoDB. In alcuni casi, potrebbe essere necessario sovrascrivere il calcolo predefinito per migliorare le prestazioni di scrittura. Se questo parametro non viene specificato, il valore WCU consentito per i processi Spark verrà calcolato automaticamente mediante la seguente formula:

-
  - `numPartitions = dynamicframe.getNumPartitions()`
  - `numSlots` (come definito in precedenza in questa sezione)
  - `numParallelTasks = min(numPartitions, numSlots)`
- Esempio 1. DPU=10, WorkerType=Standard. L'input DynamicFrame ha 100 partizioni RDD.
  - `numPartitions = 100`

  - `numExecutors = (10 - 1) * 2 - 1 = 17`
  - `numSlots = 4 * 17 = 68`
  - `numParallelTasks = min(100, 68) = 68`
- Esempio 2. DPU=10, WorkerType=Standard. L'input DynamicFrame ha 20 partizioni RDD.
  - `numPartitions = 20`
  - `numExecutors = (10 - 1) * 2 - 1 = 17`
  - `numSlots = 4 * 17 = 68`
  - `numParallelTasks = min(20, 68) = 20`

> **Note**
>
> I processi sulle versioni precedenti di AWS Glue e quelli che utilizzano worker standard richiedono metodi diversi per calcolare il numero di slot. Se hai bisogno di ottimizzare le prestazioni di tali processi, ti consigliamo di passare alle versioni di AWS Glue supportate.

Indicazioni di riferimento per le opzioni di connessione a DynamoDB

Indica una connessione ad Amazon DynamoDB.

Le opzioni di connessione differiscono per una connessione sorgente e una connessione sink.

"connectionType": "dynamodb" con il connettore ETL come origine

Utilizzare le seguenti opzioni di connessione con `"connectionType": "dynamodb"` come origine, quando utilizzi Connettore ETL per Glue DynamoDB AWS:

- `"dynamodb.input.tableName"`: (Obbligatorio) la tabella DynamoDB da cui leggere.
- `"dynamodb.throughput.read.percent"`: (Facoltativo) percentuale di unità di capacità di lettura (RCU) da usare. Il valore predefinito è "0,5". I valori accettabili vanno da "0,1" a "1,5", inclusi.
  - `0.5` rappresenta la velocità di lettura di default, il che significa che AWS Glue cercherà di consumare metà della capacità di lettura della tabella. Se si aumenta il valore sopra `0.5`, AWS Glue aumenterà il tasso di richiesta; diminuendo il valore al di sotto di `0.5`, riduce la frequenza di richiesta di lettura. La velocità di lettura effettiva varia in base a fattori come la presenza di una distribuzione uniforme delle chiavi nella tabella DynamoDB.

  - Quando la tabella DynamoDB è in modalità on demand, AWS Glue gestisce la capacità di lettura della tabella a 40000. Per esportare una tabella di grandi dimensioni, si consiglia di passare alla modalità su richiesta della tabella DynamoDB.
- `"dynamodb.splits"`: (Facoltativo) Definisce il numero di partizioni applicate a questa tabella DynamoDB durante la lettura. Il valore predefinito è "1". I valori accettabili vanno da "1" a "1,000,000", inclusi.

  1 indica che non c'è parallelismo. Si consiglia vivamente di specificare un valore maggiore per migliorare le prestazioni utilizzando la formula riportata di seguito. Per ulteriori informazioni sull'impostazione corretta di un valore, consulta the section called “Parallelismo di DynamoDB”.
- `"dynamodb.sts.roleArn"`: (Facoltativo) il ruolo IAM ARN da assumere per l'accesso multi-account. Questo parametro è disponibile in AWS Glue 1.0 o versione successiva.
- `"dynamodb.sts.roleSessionName"`: (Facoltativo) nome della sessione STS. Il valore predefinito è impostato su "glue-dynamodb-read-sts-session". Questo parametro è disponibile in AWS Glue 1.0 o versione successiva.

"connectionType": "dynamodb" con il connettore di esportazione DynamoDB AWS Glue come sorgente

Utilizzare le seguenti opzioni di connessione con «connectionType»: «dynamodb» come origine, quando utilizzi il Connettore di esportazione DynamoDB AWS Glue, disponibile solo per la versione AWS Glue 2.0 in poi:

- `"dynamodb.export"`: (Richiesto) Un valore stringa:
  - Se impostato su `ddb` abilita il Connettore di esportazione DynamoDB AWS Glue dove un nuovo `ExportTableToPointInTimeRequest` verrà invocato durante il processo AWS Glue. Verrà generata una nuova esportazione con la posizione passata da `dynamodb.s3.bucket` e `dynamodb.s3.prefix`.
  - Se impostato su `s3` abilita il Connettore di esportazione DynamoDB AWS Glue ma salta la creazione di una nuova esportazione DynamoDB e utilizza invece il `dynamodb.s3.bucket` e il `dynamodb.s3.prefix` come posizione di Amazon S3 di un'esportazione precedente di quella tabella.
- `"dynamodb.tableArn"`: (Obbligatorio) la tabella DynamoDB da cui leggere.
- `"dynamodb.unnestDDBJson"`: false per impostazione predefinita (facoltativo). Valori validi: booleani. Se impostato su true, esegue una trasformazione non nidificata della struttura JSON DynamoDB presente nelle esportazioni. È un errore impostare contemporaneamente

`"dynamodb.unnestDDBJson"` e `"dynamodb.simplifyDDBJson"` su true. In AWS Glue 3.0 e versioni successive, ti consigliamo di utilizzare `"dynamodb.simplifyDDBJson"` per ottenere un comportamento migliore durante la semplificazione dei tipi di mappe DynamoDB. Per ulteriori informazioni, consulta the section called “Semplificazione dell'utilizzo del JSON di esportazione DynamoDB”.

- `"dynamodb.simplifyDDBJson"`: false per impostazione predefinita (facoltativo). Valori validi: booleani. Se impostato su true, esegue una trasformazione per semplificare la struttura JSON DynamoDB presente nelle esportazioni. Questa opzione ha lo stesso scopo di `"dynamodb.unnestDDBJson"`, ma fornisce un supporto migliore per i tipi di mappe DynamoDB o per i tipi di mappe annidate nella tabella DynamoDB. Questa opzione è disponibile in AWS Glue 3.0 e versioni successive. È un errore impostare contemporaneamente `"dynamodb.unnestDDBJson"` e `"dynamodb.simplifyDDBJson"` su true. Per ulteriori informazioni, consulta the section called “Semplificazione dell'utilizzo del JSON di esportazione DynamoDB”.
- `"dynamodb.s3.bucket"`: (facoltativo) indica la posizione del bucket Amazon S3 in cui deve essere condotto il processo DynamoDB `ExportTableToPointInTime`. Il formato del file per l'esportazione è DynamoDB JSON.
  - `"dynamodb.s3.prefix"`: (facoltativo) indica la posizione del prefisso di Amazon S3 all'interno del bucket Amazon S3 in cui devono essere archiviati i carichi `ExportTableToPointInTime` di DynamoDB. Se non si specificano i valori `dynamodb.s3.prefix` e `dynamodb.s3.bucket`, per impostazione predefinita sarà utilizzata la posizione della directory temporanea specificata nella configurazione del processo AWS Glue. Per ulteriori informazioni, consulta Parametri speciali usati da AWS Glue.
  - `"dynamodb.s3.bucketOwner"`: indica il proprietario del bucket necessario per l'accesso di Amazon S3 tra account.
- `"dynamodb.sts.roleArn"`: (facoltativo) l'ARN del ruolo IAM da assumere per l'accesso multi-account e/o l'accesso tra regioni per la tabella DynamoDB. Nota: lo stesso ARN del ruolo IAM verrà utilizzato per accedere alla posizione Amazon S3 specificata per la richiesta `ExportTableToPointInTime`.
- `"dynamodb.sts.roleSessionName"`: (Facoltativo) nome della sessione STS. Il valore predefinito è impostato su "glue-dynamodb-read-sts-session".
- `"dynamodb.exportTime"` (facoltativo). Valori validi: stringhe che rappresentano istanti ISO-8601. Un point-in-time in cui deve essere effettuata l'esportazione.
- `"dynamodb.sts.region"`: la regione che ospita la tabella DynamoDB da leggere (obbligatorio se si effettua una chiamata tra regioni utilizzando un endpoint regionale).

"connectionType": "dynamodb" con il connettore ETL come sink

Utilizzare le seguenti opzioni di connessione con `"connectionType": "dynamodb"` come sink:

- `"dynamodb.output.tableName"`: (Obbligatorio) la tabella DynamoDB su cui scrivere.
- `"dynamodb.throughput.write.percent"`: (Facoltativo) percentuale di unità di capacità di scrittura (WCU) da usare. Il valore predefinito è "0,5". I valori accettabili vanno da "0,1" a "1,5", inclusi.
  - `0.5` rappresenta la velocità di scrittura di default, il che significa che AWS Glue cercherà di consumare metà della capacità di scrittura della tabella. Se si aumenta il valore sopra 0,5, AWS Glue aumenterà il tasso di richiesta; diminuendo il valore al di sotto di 0,5, riduce la frequenza di richiesta di scrittura. (La velocità di scrittura effettiva varia in base a fattori come la presenza di una distribuzione uniforme delle chiavi nella tabella DynamoDB).
  - Quando la tabella DynamoDB è in modalità on demand, AWS Glue gestisce la capacità di lettura della tabella come `40000`. Per importare una tabella di grandi dimensioni, si consiglia di passare alla modalità su richiesta della tabella DynamoDB.
- `"dynamodb.output.numParallelTasks"`: (Facoltativo) definisce il numero di attività parallele scritte contemporaneamente in DynamoDB. Utilizzato per calcolare WCU permissivo per i processi Spark. Nella maggior parte dei casi, AWS Glue calcolerà un valore predefinito ragionevole per questo valore. Per ulteriori informazioni, consulta the section called “Parallelismo di DynamoDB”.
- `"dynamodb.output.retry"`: (Facoltativo) definisce il numero di tentativi eseguiti quando esiste una `ProvisionedThroughputExceededException` da DynamoDB. Il valore predefinito è "10".
- `"dynamodb.sts.roleArn"`: (facoltativo) il ruolo IAM ARN da assumere per l'accesso multi-account.
- `"dynamodb.sts.roleSessionName"`: (Facoltativo) nome della sessione STS. Il valore predefinito è impostato su "glue-dynamodb-write-sts-session".

Accesso multi-account in più regioni alle tabelle DynamoDB

I processi ETL AWS Glue supportano sia l'accesso multi-account che in più regioni alle tabelle DynamoDB. I processi ETL AWS Glue supportano sia la lettura dei dati dalla tabella DynamoDB di un altro account AWS che la scrittura dei dati in una tabella DynamoDB in un altro account AWS. AWS Glue supporta inoltre sia la lettura da una tabella DynamoDB in un'altra regione, sia la scrittura in una tabella DynamoDB in un'altra regione. Questa sezione fornisce istruzioni su come configurare l'accesso e fornisce uno script di esempio.

Le procedure descritte in questa sezione fanno riferimento a un tutorial IAM per la creazione di un ruolo IAM e la concessione dell'accesso al ruolo. Il tutorial discute anche l'assunzione di un ruolo, ma qui si utilizzerà invece uno script di processo per assumere il ruolo in AWS Glue. Questo tutorial contiene anche informazioni sulle pratiche generali multi-account. Per ulteriori informazioni, consulta Tutorial: Accesso tra account AWS tramite i ruoli IAM nella Guida per l'utente di IAM.

Creare un ruolo

Segui la fase 1 del tutorial per creare un ruolo IAM nell'account A. Quando si definiscono le autorizzazioni del ruolo, è possibile scegliere di allegare criteri esistenti quali `AmazonDynamoDBReadOnlyAccess` oppure `AmazonDynamoDBFullAccess` per consentire al ruolo di leggere/scrivere DynamoDB. L'esempio seguente mostra la creazione di un ruolo denominato `DynamoDBCrossAccessRole`, con la policy di autorizzazione `AmazonDynamoDBFullAccess`.

Concedi autorizzazione per l'accesso al ruolo

Segui la fase 2 del tutorial nella Guida per l'utente di IAM per consentire all'account B di passare al ruolo appena creato. L'esempio seguente crea una nuova policy con l'istruzione riportata di seguito:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "<DynamoDBCrossAccessRole's ARN>"
    }
}
```

Quindi, è possibile allegare questa policy al gruppo/ruolo/utente da utilizzare per accedere a DynamoDB.

Assumi il ruolo nello script di processo AWS Glue

Ora è possibile accedere all'account B e creare un processo AWS Glue. Per creare un processo, fai riferimento alle istruzioni in Aggiunta di processi in AWS Glue.

Nello script del processo è necessario utilizzare il parametro `dynamodb.sts.roleArn` per assumere il ruolo `DynamoDBCrossAccessRole`. Supponendo che questo ruolo consenta di ottenere le credenziali temporanee, che devono essere utilizzate per accedere a DynamoDB nell'account B, esamina questi script di esempio.

Per una lettura multi-account tra regioni (connettore ETL):

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
    "dynamodb.region": "us-east-1",
    "dynamodb.input.tableName": "test_source",
    "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
    }
)
dyf.show()
job.commit()
```

Per una lettura multi-account tra regioni (connettore ELT):

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": "<test_source ARN>",
        "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
```

```
    }
)
dyf.show()
job.commit()
```

Per una lettura e scrittura multi-account tra regioni:

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.region": "us-east-1",
        "dynamodb.input.tableName": "test_source"
    }
)
dyf.show()

glue_context.write_dynamic_frame_from_options(
    frame=dyf,
    connection_type="dynamodb",
    connection_options={
        "dynamodb.region": "us-west-2",
        "dynamodb.output.tableName": "test_sink",
        "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
    }
)

job.commit()
```

Connessioni Kinesis

Puoi leggere e scrivere su flussi di dati Amazon Kinesis utilizzando le informazioni archiviate in una tabella catalogo dati o fornendo informazioni per accedere direttamente al flusso di dati. Puoi leggere

le informazioni da Kinesis in Spark DataFrame, quindi convertirle in un Glue. AWS DynamicFrame Puoi DynamicFrames scrivere su Kinesis in formato JSON. Se accedi direttamente al flusso di dati, utilizza queste opzioni per fornire le informazioni su come accedere al flusso di dati.

Se utilizzi `getCatalogSource` o `create_data_frame_from_catalog` per consumare i registri da una sorgente di streaming Kinesis, il processo avrà le informazioni sul database catalogo dati e sul nome della tabella, e potrà usarle per ottenere alcuni parametri di base per la lettura dalla sorgente di streaming Kinesis. Se utilizzi `getSource`, `getSourceWithFormat`, `createDataFrameFromOptions` o `create_data_frame_from_options`, dovrai specificare questi parametri di base utilizzando le opzioni di connessione descritte qui.

È possibile specificare le opzioni di connessione per Kinesis utilizzando i seguenti argomenti per i metodi specificati nella classe `GlueContext`.

- Scala
  - `connectionOptions`: utilizza con `getSource`, `createDataFrameFromOptions` e `getSink`
  - `additionalOptions`: utilizza con `getCatalogSource`, `getCatalogSink`
  - `options`: utilizza con `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: utilizza con `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: utilizza con `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: utilizza con `getSource`, `getSink`

Per osservazioni e restrizioni sui processi ETL dei flussi di dati, consulta la pagina the section called “Streaming di note e restrizioni ETL”.

Configurazione di Kinesis

Per connetterti a un flusso di dati Kinesis in un job AWS Glue Spark, avrai bisogno di alcuni prerequisiti:

- In caso di lettura, il job AWS Glue deve disporre delle autorizzazioni IAM di livello di accesso Read per il flusso di dati Kinesis.
- In fase di scrittura, il job AWS Glue deve disporre delle autorizzazioni IAM di livello di accesso Write per il flusso di dati Kinesis.

In alcuni casi, è necessario configurare ulteriori prerequisiti:

- Se il tuo job AWS Glue è configurato con connessioni di rete aggiuntive (in genere per connettersi ad altri set di dati) e una di queste connessioni offre opzioni di rete Amazon VPC, questo indirizzerà il tuo lavoro a comunicare tramite Amazon VPC. In questo caso, per comunicare tramite Amazon VPC dovrai configurare anche il flusso di dati Kinesis. È possibile farlo creando un endpoint VPC di interfaccia tra l'Amazon VPC e il flusso di dati Kinesis. Per ulteriori informazioni, consulta la pagina Using Amazon Kinesis Data Streams with Interface VPC Endpoints.
- Quando si specifica un flusso di dati Amazon Kinesis in un altro account, è necessario impostare i ruoli e le policy per consentire l'accesso multi-account. Per ulteriori informazioni, consulta Esempio: lettura da un flusso Kinesis in un account diverso.

Per ulteriori informazioni sui prerequisiti dei processi ETL dei flussi di dati, consulta la pagina the section called “Aggiunta di processi di streaming ETL”.

Esempio: lettura da flussi Kinesis

Esempio: lettura da flussi Kinesis

Usato in combinazione con the section called “forEachBatch”.

Esempio per l'origine di streaming Amazon Kinesis:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Indicazioni di riferimento alle opzioni di connessione a Kinesis

Indica le opzioni di connessione ad Amazon Kinesis Data Streams.

Utilizza le seguenti opzioni di connessione per le origini dati in streaming Kinesis:

- `"streamARN"`: (obbligatorio) utilizzato per la lettura/scrittura. L'ARN del flusso di dati di Kinesis.

- `"classification"`: (obbligatorio per la lettura) utilizzato per la lettura. Il formato di file utilizzato dai dati nel record. Obbligatorio, a meno che non sia fornito tramite Catalogo dati.
- `"streamName"`: (facoltativo) utilizzato per la lettura. Il nome di un flusso di dati Kinesis da cui leggere. Usato con `endpointUrl`.
- `"endpointUrl"`: (facoltativo) utilizzato per la lettura. Predefinito: "https://kinesis.us-east-1.amazonaws.com". L' AWS endpoint del flusso Kinesis. Non è necessario modificarlo a meno che non ci si stia connettendo a una regione speciale.
- `"partitionKey"`: (facoltativo) utilizzato per la scrittura. La chiave di partizione di Kinesis utilizzata per la produzione dei record.
- `"delimiter"`: (facoltativo) utilizzato per la lettura. Il separatore di valori utilizzato quando `classification` è CSV. Il valore predefinito è `","`.
- `"startingPosition"`: (facoltativo) utilizzato per la lettura. La posizione di partenza nel flusso dei dati Kinesis da cui leggere i dati. I valori possibili sono `"latest"`, `"trim_horizon"`, `"earliest"` o una stringa di timestamp in formato UTC con il modello `yyyy-mm-ddTHH:MM:SSZ`, dove `Z` rappresenta uno scostamento del fuso orario UTC con un +/- (ad esempio: "2023-04-04T08:00:00-04:00"). Il valore predefinito è `"latest"`. Nota: la stringa Timestamp in formato UTC per `"startingPosition"` è supportata solo per AWS Glue versione 4.0 o successiva.
- `"failOnDataLoss"`: (facoltativo) non è possibile eseguire il processo se una partizione attiva è mancante o scaduta. Il valore predefinito è `"false"`.
- `"awsSTSRoleARN"`: (facoltativo) utilizzato per la lettura/scrittura. L'Amazon Resource Name (ARN) del ruolo da assumere utilizzando AWS Security Token Service (AWS STS). Questo ruolo deve disporre delle autorizzazioni per descrivere o leggere le operazioni dei registri per il flusso di dati Kinesis. Quando si accede a un flusso di dati in un altro account, è necessario utilizzare questo parametro. Usato in combinazione con `"awsSTSSessionName"`.
- `"awsSTSSessionName"`: (facoltativo) utilizzato per la lettura/scrittura. Un identificatore della sessione che assume il ruolo usando AWS STS. Quando si accede a un flusso di dati in un altro account, è necessario utilizzare questo parametro. Usato in combinazione con `"awsSTSRoleARN"`.
- `"awsSTSEndpoint"`: (Facoltativo) L' AWS STS endpoint da utilizzare quando ci si connette a Kinesis con un ruolo presunto. Ciò consente di utilizzare l' AWS STS endpoint regionale in un VPC, cosa non possibile con l'endpoint globale predefinito.

- `"maxFetchTimeInMs"`: (facoltativo) utilizzato per la lettura. Il tempo massimo impiegato nell'esecutore del processo per recuperare un registro dal flusso dei dati Kinesis per partizione, specificato in millisecondi (ms). Il valore predefinito è `1000`.
- `"maxFetchRecordsPerShard"`: (facoltativo) utilizzato per la lettura. Il numero massimo di record da recuperare per shard nel flusso di dati Kinesis per microbatch. Nota: il client può superare questo limite se il job di streaming ha già letto record aggiuntivi da Kinesis (nella stessa chiamata get-records). Se `maxFetchRecordsPerShard` deve essere rigoroso, deve essere un multiplo di. `maxRecordPerRead` Il valore predefinito è `100000`.
- `"maxRecordPerRead"`: (facoltativo) utilizzato per la lettura. Il numero massimo di record da recuperare nel flusso di dati Kinesis in ciascuna operazione `getRecords`. Il valore predefinito è `10000`.
- `"addIdleTimeBetweenReads"`: (facoltativo) utilizzato per la lettura. Aggiunge un ritardo tra due operazioni consecutive `getRecords`. Il valore predefinito è `"False"`. Questa opzione è configurabile solo per Glue versione 2.0 e successive.
- `"idleTimeBetweenReadsInMs"`: (facoltativo) utilizzato per la lettura. Il ritardo minimo tra due operazioni consecutive `getRecords`, specificato in ms. Il valore predefinito è `1000`. Questa opzione è configurabile solo per Glue versione 2.0 e successive.
- `"describeShardInterval"`: (facoltativo) utilizzato per la lettura. L'intervallo di tempo minimo tra due chiamate API `ListShards` affinché lo script consideri il resharding. Per ulteriori informazioni, consulta Strategie per il resharding nella Guida per gli sviluppatori di Amazon Kinesis Data Streams. Il valore predefinito è `1s`.
- `"numRetries"`: (facoltativo) utilizzato per la lettura. Il numero massimo di tentativi per le richieste API Kinesis Data Streams. Il valore predefinito è `3`.
- `"retryIntervalMs"`: (facoltativo) utilizzato per la lettura. Il periodo di raffreddamento (specificato in ms) prima di riprovare la chiamata API Kinesis Data Streams. Il valore predefinito è `1000`.
- `"maxRetryIntervalMs"`: (facoltativo) utilizzato per la lettura. Il periodo di raffreddamento (specificato in ms) tra due tentativi di chiamata API Kinesis Data Streams. Il valore predefinito è `10000`.
- `"avoidEmptyBatches"`: (facoltativo) utilizzato per la lettura. Impedisce la creazione di un processo microbatch vuoto controllando la presenza di dati non letti nel flusso dei dati Kinesis prima che il batch venga avviato. Il valore predefinito è `"False"`.
- `"schema"`: (obbligatorio quando inferSchema è impostato su falso) utilizzato per la lettura. Lo schema da utilizzare per elaborare il payload. Se la classificazione è `avro`, lo schema fornito

dovrà essere nel formato dello schema Avro. Se la classificazione è `avro`, lo schema fornito dovrà essere nel formato dello schema DDL.

Di seguito sono riportati alcuni esempi di schema.

Example in DDL schema format

```
`column1` INT, `column2` STRING , `column3` FLOAT
```

Example in Avro schema format

```
{
  "type":"array",
  "items":
  {
    "type":"record",
    "name":"test",
    "fields":
    [
      {
        "name":"_id",
        "type":"string"
      },
      {
        "name":"index",
        "type":
        [
          "int",
          "string",
          "float"
        ]
      }
    ]
  }
}
```

- `"inferSchema"`: (facoltativo) utilizzato per la lettura. Il valore predefinito è "false". Se impostato su "true", lo schema verrà rilevato in fase di runtime dal payload all'interno di `foreachbatch`.
- `"avroSchema"`: (obsoleto) utilizzato per la lettura. Parametro utilizzato per specificare uno schema di dati Avro quando viene utilizzato il formato Avro. Questo parametro è obsoleto. Utilizzo del parametro `schema`.

- `"addRecordTimestamp"`: (facoltativo) utilizzato per la lettura. Quando questa opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "__src_timestamp" che indica l'ora in cui il record corrispondente è stato ricevuto dal flusso. Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.
- `"emitConsumerLagMetrics"`: (facoltativo) utilizzato per la lettura. Quando l'opzione è impostata su «true», per ogni batch emetterà le metriche relative alla durata compresa tra il record più vecchio ricevuto dallo stream e il momento in AWS Glue cui arriva. CloudWatch Il nome della metrica è «glue.driver.streaming. maxConsumerLagInMs». Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.
- `"fanoutConsumerARN"`: (facoltativo) utilizzato per la lettura. L'ARN di un consumatore di un flusso Kinesis per il flusso specificato in `streamARN`. Utilizzato per abilitare la modalità di fan-out avanzato per la connessione Kinesis. Per ulteriori informazioni sull'utilizzo di un flusso Kinesis con fan-out avanzato, consulta la pagina the section called “Utilizzo del fan-out avanzato nei processi di flussi di dati Kinesis”.
- `"recordMaxBufferedTime"`: (facoltativo) utilizzato per la scrittura. Predefinito: 1000 (ms). Tempo massimo di memorizzazione nel buffer di un record in attesa di essere scritto.
- `"aggregationEnabled"`: (facoltativo) utilizzato per la scrittura. Default: true (VERO). Speciifica se i record devono essere aggregati prima di inviarli a Kinesis.
- `"aggregationMaxSize"`: (facoltativo) utilizzato per la scrittura. Impostazione predefinita: 51200 (byte). Se un record è superiore a questo limite, ignorerà l'aggregatore. Ricorda che Kinesis impone un limite di 50 KB alla dimensione del record. Se imposti questo valore oltre i 50 KB, i record di grandi dimensioni verranno rifiutati da Kinesis.
- `"aggregationMaxCount"`: (facoltativo) utilizzato per la scrittura. Predefinito: 4294967295. Numero massimo di voci da inserire in un record aggregato.
- `"producerRateLimit"`: (facoltativo) utilizzato per la scrittura. Predefinito: 150 (%). Limita la velocità di trasmissione effettiva per partizione inviata da un singolo produttore (ad esempio, il tuo processo), come percentuale del limite di backend.
- `"collectionMaxCount"`: (facoltativo) utilizzato per la scrittura. Predefinito: 500. Numero massimo di articoli da imballare in una PutRecords richiesta.
- `"collectionMaxSize"`: (facoltativo) utilizzato per la scrittura. Impostazione predefinita: 5242880 (byte). Quantità massima di dati da inviare con una PutRecords richiesta.

Utilizzo del fan-out avanzato nei processi di flussi di dati Kinesis

Un consumatore con fan-out avanzato è in grado di ricevere i record da un flusso Kinesis con una velocità di trasmissione effettiva dedicata che può essere superiore a quella dei consumatori tipici. Questo viene fatto ottimizzando il protocollo di trasferimento utilizzato per fornire dati a un consumatore Kinesis, ad esempio il tuo processo. Per ulteriori informazioni sul fan-out avanzato di Kinesis, consulta la documentazione di Kinesis.

Nella modalità fan-out avanzato, le opzioni di connessione `maxRecordPerRead` e `idleTimeBetweenReadsInMs` non sono più valide, poiché tali parametri non sono configurabili quando si utilizza il fan-out avanzato. Le opzioni di configurazione per i nuovi tentativi funzionano come descritto.

Utilizza le seguenti procedure per abilitare e disabilitare il fan-out avanzato per il tuo processo di flussi di dati. Devi registrare un consumatore del flusso per ogni processo che consumerà i dati del flusso.

Per abilitare il consumo con fan-out avanzato nel processo:

1. Registra un consumatore del flusso per il tuo processo utilizzando l'API Kinesis. Segui le istruzioni per registrare un consumatore con fan-out avanzato utilizzando l'API Flusso di dati Kinesis riportate nella documentazione di Kinesis. Dovrai eseguire solo il primo passaggio: chiamare RegisterStreamConsumer. La tua richiesta dovrebbe restituire un ARN, *`consumerARN`*.
2. Imposta l'opzione di connessione `fanoutConsumerARN` su *`consumerARN`* negli argomenti del metodo di connessione.
3. Riavvia il processo.

Per disabilitare il consumo con fan-out avanzato nel processo:

1. Rimuovi l'opzione di connessione `fanoutConsumerARN` dalla chiamata al metodo.
2. Riavvia il processo.
3. Segui le istruzioni per annullare la registrazione di un consumatore riportate nella documentazione di Kinesis. Queste istruzioni si applicano alla console, ma possono essere utilizzate anche per l'API Kinesis. Per ulteriori informazioni sull'annullamento della registrazione dei consumatori dei flussi tramite l'API Kinesis, consulta la sezione DeregisterStreamConsumer nella documentazione di Kinesis.

Connessioni Amazon S3

È possibile utilizzare AWS Glue per Spark per leggere e scrivere file in Amazon S3. AWS Glue per Spark supporta molti formati di dati comuni archiviati in Amazon S3, tra cui CSV, Avro, JSON, Orc e Parquet. Per ulteriori informazioni sui formati di dati supportati, consulta la pagina the section called “Opzioni del formato dei dati”. Ogni formato di dati può supportare una diversa gamma di funzionalità di AWS Glue. Consulta la pagina relativa al formato dei dati per le specifiche del supporto delle funzionalità. Inoltre, puoi leggere e scrivere file con versioni diverse archiviati nei framework di data lake Hudi, Iceberg e Delta Lake. Per ulteriori informazioni sui framework di data lake, consulta la pagina the section called “Framework data lake”.

Con AWS Glue è possibile partizionare gli oggetti Amazon S3 in una struttura di cartelle durante la scrittura, quindi recuperarli in base alla partizione per migliorare le prestazioni con una configurazione semplice. È inoltre possibile impostare la configurazione per raggruppare file di piccole dimensioni durante la trasformazione dei dati per migliorare le prestazioni. In Amazon S3 è possibile leggere e scrivere archivi `bzip2` e `gzip`.

Argomenti

- Configurazione delle connessioni S3
- Indicazioni di riferimento alle opzioni di connessione ad Amazon S3
- Sintassi di connessione obsolete per i formati di dati
- Esclusione delle classi di storage Amazon S3
- Gestione delle partizioni per l'output ETL in AWS Glue
- Lettura di file di input in gruppi di grandi dimensioni
- Endpoint Amazon VPC per Amazon S3

Configurazione delle connessioni S3

Per connetterti ad Amazon S3 in AWS Glue con un processo Spark, dovrai soddisfare alcuni prerequisiti:

- Il processo AWS Glue deve disporre delle autorizzazioni IAM per i bucket Amazon S3 pertinenti.

In alcuni casi, è necessario configurare ulteriori prerequisiti:

- Quando configuri l'accesso multi-account, configura i controlli dell'accesso appropriati nel bucket Amazon S3.

- Per motivi di sicurezza, è possibile scegliere di instradare le richieste Amazon S3 tramite un Amazon VPC. Questo approccio può introdurre problematiche relative alla larghezza di banda e alla disponibilità. Per ulteriori informazioni, consulta the section called “Endpoint Amazon VPC per Amazon S3”.

Indicazioni di riferimento alle opzioni di connessione ad Amazon S3

Indica una connessione ad Amazon S3.

Poiché Amazon S3 gestisce i file anziché le tabelle, oltre a specificare le proprietà di connessione fornite in questo documento, dovrai specificare una configurazione aggiuntiva sul tipo di file. Queste informazioni vengono specificate tramite le opzioni di formato dei dati. Per ulteriori informazioni sulle opzioni di formato, consulta la pagina the section called “Opzioni del formato dei dati”. È inoltre possibile specificare queste informazioni mediante l'integrazione con Catalogo dati AWS Glue.

Per un esempio della distinzione tra opzioni di connessione e opzioni di formato, prendi in considerazione come il metodo the section called “create_dynamic_frame_from_options” recepisce `connection_type`, `connection_options`, `format` e `format_options`. Questa sezione illustra in modo specifico i parametri forniti a `connection_options`.

Utilizzare le seguenti opzioni di connessione con `"connectionType": "s3"`:

- `"paths"`: (Obbligatorio) un elenco dei percorsi Amazon S3 da cui leggere.
- `"exclusions"`: (facoltativo) una stringa contenente un elenco di JSON di modelli glob in stile Unix da escludere. Ad esempio `"[\"**.pdf\"]"` esclude tutti i file PDF. Per ulteriori informazioni sulla sintassi glob supportata da AWS Glue, vedere Modelli di inclusione e di esclusione.
- `"compressionType"`: o `"compression"`: (facoltativo) specifica il modo in cui i dati sono compressi. Utilizza `"compressionType"` per origini Amazon S3 e `"compression"` per destinazioni Amazon S3. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip2"`). È possibile che vengano supportati formati di compressione aggiuntivi per formati specifici. Per i dettagli sul supporto delle varie funzionalità, consulta la pagina relativa al formato dei dati.
- `"groupFiles"`: (facoltativo) il raggruppamento di file è attivato per impostazione predefinita quando l'input contiene più di 50.000 file. Per attivare il raggruppamento con meno di 50.000 file, imposta questo parametro su `"inPartition"`. Per disabilitare il raggruppamento in presenza di più di 50.000 file, imposta il parametro su `"none"`.

- `"groupSize"`: (facoltativo) dimensione del gruppo target in byte. Il valore di default viene calcolato in base alla dimensione dei dati di input e alle dimensioni del cluster. Quando sono presenti meno di 50.000 file di input, `"groupFiles"` deve essere impostato su `"inPartition"` per rendere effettiva la modifica.
- `"recurse"`: (facoltativo) se è impostato su "true", legge i file in modo ricorsivo in tutte le sottodirectory dei percorsi specificati.
- `"maxBand"`: (facoltativo, avanzato) questa opzione controlla la durata in millisecondi dopo la quale è probabile che l'elenco `s3` sia coerente. I file con timestamp di modifica che rientrano negli ultimi `maxBand` secondi, vengono tracciati in modo specifico quando si usano `JobBookmarks` per verificare la consistenza finale in Amazon S3. Per la maggior parte degli utenti non è necessario impostare questa opzione. Il valore di default è 900.000 millisecondi o 15 minuti.
- `"maxFilesInBand"`: (Facoltativo, avanzato) questa opzione specifica il numero massimo di file da salvare negli ultimi `maxBand` secondi. Se si supera questo valore, i file aggiuntivi vengono saltati e solo elaborati nella successiva esecuzione del processo. Per la maggior parte degli utenti non è necessario impostare questa opzione.
- `"isFailFast"`: (Facoltativo) questa opzione determina se un processo ETL AWS Glue generi eccezioni di analisi del lettore. Se impostato su `true`, i processi non riescono rapidamente se quattro tentativi del processo Spark non riescono a analizzare correttamente i dati.
- `"catalogPartitionPredicate"`: (facoltativo) utilizzato per la lettura. Il contenuto di una clausola `WHERE` in SQL. Utilizzato per la lettura dalle tabelle di Catalogo dati con una quantità molto elevata di partizioni. Recupera le partizioni corrispondenti dagli indici di Catalogo dati. Utilizzato con `push_down_predicate`, un'opzione sul metodo the section called “create_dynamic_frame_from_catalog” (e altri metodi simili). Per ulteriori informazioni, consulta the section called “Predicati di partizione di catalogo”.
- `"partitionKeys"`: (facoltativo) utilizzato per la scrittura. Un array di stringhe di etichette di colonne. AWS Glue partizionerà i dati come specificato da questa configurazione. Per ulteriori informazioni, consulta the section called “Scrittura delle partizioni”.
- `"excludeStorageClasses"`: (facoltativo) utilizzato per la lettura. Un array di stringhe che specificano le classi di archiviazione di Amazon S3. AWS Glue escluderà gli oggetti Amazon S3 in base a questa configurazione. Per ulteriori informazioni, consulta the section called “Esclusione delle classi di storage Amazon S3”.

Sintassi di connessione obsolete per i formati di dati

È possibile accedere a determinati formati di dati utilizzando una sintassi specifica per il tipo di connessione. Questa sintassi è obsoleta. Pertanto, si consiglia di specificare i formati utilizzando il tipo di connessione `s3` e le opzioni di formato fornite in the section called “Opzioni del formato dei dati”.

"connectionType": "Orc"

Indica una connessione a file archiviati in Amazon S3 nel formato Apache Hive Optimized Row Columnar (ORC).

Utilizzare le seguenti opzioni di connessione con `"connectionType": "orc"`:

- `paths`: (Obbligatorio) un elenco dei percorsi Amazon S3 da cui leggere.
- (altro nome opzione/coppie di valori): qualsiasi opzione aggiuntiva, incluso le opzioni di formattazione, vengono passate direttamente a SparkSQL `DataSource`.

"connectionType": "parquet"

Indica una connessione a file archiviati in Amazon S3 nel formato di file Apache Parquet.

Utilizzare le seguenti opzioni di connessione con `"connectionType": "parquet"`:

- `paths`: (Obbligatorio) un elenco dei percorsi Amazon S3 da cui leggere.
- (altro nome opzione/coppie di valori): qualsiasi opzione aggiuntiva, incluso le opzioni di formattazione, vengono passate direttamente a SparkSQL `DataSource`.

Esclusione delle classi di storage Amazon S3

Se esegui processi ETL AWS Glue che leggono file o partizioni da Amazon Simple Storage Service (Amazon S3), puoi escludere alcuni tipi di classe di archiviazione Amazon S3.

Le seguenti classi di storage sono disponibili in Amazon S3:

- `STANDARD`: per lo storage generico dei dati a cui si accede di frequente.
- `INTELLIGENT_TIERING`: per i dati di lunga durata con modelli di accesso sconosciuti o modificati.
- `STANDARD_IA` e `ONEZONE_IA`: per i dati esistenti da molto tempo a cui si accede meno frequentemente.

- `GLACIER`, `DEEP_ARCHIVE` e `REDUCED_REDUNDANCY`: per l'archiviazione a lungo termine e la conservazione digitale.

Per ulteriori informazioni, consulta Classi di storage Amazon S3 nella Guida per gli sviluppatori di Amazon S3.

Gli esempi in questa sezione mostrano come escludere le classi di storage `DEEP_ARCHIVE` e `GLACIER`. Queste classi consentono di elencare i file, ma non di leggere i file a meno che non vengano ripristinati. (Per ulteriori informazioni, consulta Ripristino di oggetti archiviati nella Guida per gli sviluppatori di Amazon S3).

Utilizzando le esclusioni della classe di storage, puoi assicurarti che i processi AWS Glue funzionino su tabelle che dispongono di partizioni tra questi livelli di classi di storage. Senza esclusioni, i processi che leggono i dati da questi livelli hanno esito negativo con il seguente errore: AmazonS3Exception: l'operazione non è valida per la classe di storage dell'oggetto.

Esistono diversi modi per filtrare le classi di archiviazione Amazon S3 in AWS Glue.

Argomenti

- Esclusione delle classi di storage Amazon S3 durante la creazione di un frame dinamico
- Esclusione delle classi di storage Amazon S3 in una tabella del catalogo dati

Esclusione delle classi di storage Amazon S3 durante la creazione di un frame dinamico

Per escludere le classi di archiviazione Amazon S3 durante la creazione di un frame dinamico, utilizza `excludeStorageClasses` in `additionalOptions`. AWS Glue utilizza automaticamente la propria implementazione Amazon S3 `Lister` per elencare ed escludere i file corrispondenti alle classi di archiviazione specificate.

I seguenti esempi Python e Scala mostrano come escludere le classi di storage `GLACIER` e `DEEP_ARCHIVE` durante la creazione di un frame dinamico.

Esempio di Python:

```
glueContext.create_dynamic_frame.from_catalog(
    database = "my_database",
    tableName = "my_table_name",
    redshift_tmp_dir = "",
    transformation_ctx = "my_transformation_context",
```

```
    additional_options = {
        "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
)
```

Esempio di Scala:

```
val* *df = glueContext.getCatalogSource(
    nameSpace, tableName, "", "my_transformation_context",
    additionalOptions = JsonOptions(
        Map("excludeStorageClasses" -> List("GLACIER", "DEEP_ARCHIVE"))
    )
).getDynamicFrame()
```

Esclusione delle classi di storage Amazon S3 in una tabella del catalogo dati

Puoi specificare le esclusioni della classe di archiviazione che devono essere utilizzate da un processo ETL AWS Glue come un parametro di tabella nel catalogo dati di AWS Glue. Puoi includere questo parametro nell'operazione `CreateTable` utilizzando AWS Command Line Interface (AWS CLI) o a livello di programmazione utilizzando l'API. Per ulteriori informazioni, consulta Struttura della tabella e CreateTable.

Puoi anche specificare classi di storage escluse nella console AWS Glue.

Per escludere classi di storage Amazon S3 (console)

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nel riquadro di navigazione a sinistra, scegliere Tables (Tabelle).
3. Scegliere il nome della tabella nell'elenco, quindi selezionare Edit table (Modifica tabella).
4. In Table properties (Proprietà tabella), aggiungere **`excludeStorageClasses`** come una chiave e **`[\"GLACIER\",\"DEEP_ARCHIVE\"]`** come un valore.
5. Seleziona Apply (Applica).

Gestione delle partizioni per l'output ETL in AWS Glue

Il partizionamento è una tecnica importante per organizzare i set di dati in modo da poterne eseguire query efficaci. I dati sono organizzati in una struttura gerarchica di directory basata su valori distinti di una o più colonne.

Ad esempio, puoi decidere di partizionare i log di un'applicazione in Amazon Simple Storage Service (Amazon S3) per data, suddivisi per anno, mese e giorno. I file che corrispondono ai dati di un solo giorno vengono quindi posti sotto un prefisso, ad esempio `s3://my_bucket/logs/year=2018/month=01/day=23/`. Sistemi come Amazon Athena, Amazon Redshift Spectrum e ora AWS Glue possono usare queste partizioni per filtrare i dati in base al valore di partizione senza dover leggere tutti i dati sottostanti da Amazon S3.

I crawler non deducono solo i tipi di file e gli schemi, ma identificano automaticamente anche la struttura della partizione del set di dati quando popolano AWS Glue Data Catalog. Le colonne di partizione risultanti sono disponibili per le query nei processi ETL AWS Glue o per motori di query come Amazon Athena.

Dopo aver eseguito il crawling di una tabella, puoi visualizzare le partizioni create dal crawler. Nella console AWS Glue, scegli Applications (Applicazioni) nel pannello di navigazione sinistro. Scegli la tabella creata dal crawler, quindi scegli View Partitions (Visualizza partizioni).

Per i percorsi partizionati di tipo Apache Hive nello stile `key=val`, i crawler popolano automaticamente il nome della colonna usando il nome della chiave. Altrimenti, utilizzano i nomi predefiniti, ad esempio `partition_0`, `partition_1` e così via. È possibile modificare i nomi predefiniti sulla console. A tale scopo, accedi alla tabella. Controlla se gli indici esistono nella scheda Indici. In tal caso, è necessario eliminarli per procedere; potrai ricrearli in seguito utilizzando i nuovi nomi di colonna. Quindi, scegli Modifica schema e modifica i nomi delle colonne delle partizioni.

Negli script ETL, puoi applicare un filtro alle colonne di partizione. Poiché le informazioni sulla partizione sono memorizzate nel catalogo dati, utilizza le chiamate API `from_catalog` per includere le colonne delle partizioni nel `DynamicFrame`. Ad esempio, utilizza `create_dynamic_frame.from_catalog` anziché `create_dynamic_frame.from_options`.

Il partizionamento è una tecnica di ottimizzazione che riduce la scansione dei dati. Per ulteriori informazioni sul processo di identificazione del momento in cui questa tecnica è appropriata, consulta Ridurre la quantità di scansione dei dati nella guida Best practice per l'ottimizzazione delle prestazioni dei processi AWS per Apache Spark nel Prontuario AWS.

Prefiltraggio con i predicati pushdown

In molti casi, puoi utilizzare un predicato pushdown per filtrare le partizioni senza dover elencare e leggere tutti i file del set di dati. Invece di leggere l'intero set di dati e quindi filtrarli in un oggetto DynamicFrame, puoi applicare il filtro direttamente sui metadati della partizione nel catalogo dati. Elenchi e leggi solo quello che effettivamente ti serve in un DynamicFrame.

Ad esempio, in Python puoi scrivere quanto segue.

```
glue_context.create_dynamic_frame.from_catalog(
    database = "my_S3_data_set",
    table_name = "catalog_data_table",
    push_down_predicate = my_partition_predicate)
```

Questo crea un oggetto DynamicFrame che carica solo le partizioni nel catalogo dati che soddisfano l'espressione del predicato. A seconda di quanto piccolo sia il sottoinsieme di dati che stai caricando, questo può far risparmiare moltissimo tempo nell'elaborazione.

L'espressione del predicato può essere qualsiasi espressione booleana supportata da Spark SQL. Funziona tutto ciò che puoi inserire in una clausola `WHERE` in una query Spark SQL. Ad esempio, l'espressione del predicato `pushDownPredicate = "(year=='2017' and month=='04')"` carica solo le partizioni in nel catalogo dati che hanno sia `year` uguale a 2017 che `month` uguale a 04. Per ulteriori informazioni, consulta la documentazione di Apache Spark SQL, in particolare il riferimento alle funzioni SQL Scala.

Filtraggio lato server utilizzando predicati delle partizioni di catalogo

L'opzione `push_down_predicate` viene applicata dopo aver elencato tutte le partizioni dal catalogo e prima di pubblicare i file da Amazon S3 per tali partizioni. Se per una tabella ci sono molte partizioni, l'elenco delle partizioni del catalogo può comunque essere soggetto a un sovraccarico di tempo aggiuntivo. Per risolvere questo sovraccarico, puoi eliminare le partizioni lato server con l'opzione `catalogPartitionPredicate` che utilizza indici delle partizioni in AWS Glue Data Catalog. Questo rende il filtraggio delle partizioni molto più veloce quando sono presenti milioni di partizioni in una tabella. Puoi utilizzare sia `push_down_predicate` che `catalogPartitionPredicate` insieme in `additional_options` se il `catalogPartitionPredicate` richiede sintassi del predicato che non è ancora supportata con gli indici delle partizioni del catalogo.

Python:

```
dynamic_frame = glueContext.create_dynamic_frame.from_catalog(
    database=dbname,
    table_name=tablename,
    transformation_ctx="datasource0",
    push_down_predicate="day>=10 and customer_id like '10%'",
    additional_options={"catalogPartitionPredicate":"year='2021' and month='06'"}
)
```

Scala:

```
val dynamicFrame = glueContext.getCatalogSource(
    database = dbname,
    tableName = tablename,
    transformationContext = "datasource0",
    pushDownPredicate="day>=10 and customer_id like '10%'",
    additionalOptions = JsonOptions("""{
        "catalogPartitionPredicate": "year='2021' and month='06'"}""")
    ).getDynamicFrame()
```

> **Note**
>
> `push_down_predicate` e `catalogPartitionPredicate` usano sintassi diverse. Il primo utilizza la sintassi standard Spark SQL e il secondo utilizza il parser JSQL.

Scrittura delle partizioni

Per impostazione predefinita, un DynamicFrame non è partizionato quando viene scritto. Tutti i file di output vengono scritti nel livello principale del percorso di output specificato. Fino a poco tempo fa, l'unico modo per scrivere un DynamicFrame nelle partizioni era convertirlo in un frame di dati SQL DataFrame prima della scrittura.

Ora però i DynamicFrames supportano il partizionamento nativo utilizzando una sequenza di chiavi, utilizzando l'opzione `partitionKeys` quando crei un sink. Ad esempio, il codice Python seguente scrive un set di dati in Amazon S3 in formato Parquet, in directory partizionate in base al campo type. Da qui puoi elaborare le partizioni usando altri sistemi, ad esempio Amazon Athena.

```
glue_context.write_dynamic_frame.from_options(
    frame = projectedEvents,
    connection_type = "s3",
    connection_options = {"path": "$outpath", "partitionKeys": ["type"]},
    format = "parquet")
```

Lettura di file di input in gruppi di grandi dimensioni

Puoi impostare le proprietà delle tabelle per permettere a un processo ETL AWS Glue di raggruppare i file quando vengono letti da un datastore Amazon S3. Queste proprietà permettono a ogni attività ETL di leggere un gruppo di file di input in una singola partizione in memoria. Ciò è

particolarmente utile quando è presente un numero elevato di file di piccole dimensioni nel datastore Amazon S3. Quando imposti determinate proprietà, indichi ad AWS Glue di raggruppare i file in una partizione dati Amazon S3 e di impostare la dimensione dei gruppi da leggere. Puoi anche impostare queste opzioni durante la lettura da un datastore Amazon S3 con il metodo `create_dynamic_frame.from_options`.

Per abilitare il raggruppamento di file in una tabella, devi impostare coppie di valore chiave nel campo parametri della struttura della tabella. Utilizza la notazione JSON per impostare un valore per il campo parametri della tabella. Per ulteriori informazioni sulle modifiche delle proprietà di una tabella, consulta Visualizzazione e modifica dei dettagli della tabella.

Puoi usare questo metodo per abilitare il raggruppamento per le tabelle nel catalogo dati con i datastore Amazon S3.

groupFiles

Imposta groupFiles su `inPartition` per abilitare il raggruppamento dei file all'interno di una partizione dati Amazon S3. AWS Glue abilita automaticamente il raggruppamento se ci sono più di 50.000 file di input, come nel seguente esempio.

```
'groupFiles': 'inPartition'
```

groupSize

Imposta groupSize (Dimensione gruppo) alla dimensione target dei gruppi in byte. La proprietà groupSize è facoltativa e se non è presente AWS Glue calcola una dimensione per usare tutti i core della CPU nel cluster, riducendo comunque il numero complessivo di attività ETL e di partizioni in memoria.

Ad esempio, di seguito viene impostata la dimensione del gruppo su 1 MB.

```
'groupSize': '1048576'
```

È necessario impostare `groupsize` con il risultato di un calcolo. Ad esempio 1024 * 1024 = 1048576.

recurse

Imposta recurse su `True` per leggere in modo ricorsivo i file in tutte le sottodirectory quando si specifica `paths` come una matrice di percorsi. Non è necessario impostare recurse se `paths` è una matrice di chiavi di oggetti in Amazon S3 o se il formato di input è parquet/orc, come nell'esempio seguente.

```
'recurse':True
```

Se leggi da Amazon S3 utilizzando direttamente il metodo `create_dynamic_frame.from_options`, aggiungi queste opzioni di connessione. Ad esempio, i seguenti tentativi di raggruppare file in gruppi da 1 MB.

```
df = glueContext.create_dynamic_frame.from_options("s3", {'paths': ["s3://s3path/"],
 'recurse':True, 'groupFiles': 'inPartition', 'groupSize': '1048576'}, format="json")
```

> **Note**
>
> `groupFiles` è supportato per DynamicFrames creati dai seguenti formati di dati: csv, ion, grokLog, json e xml. Questa opzione non è supportata per avro, parquet e orc.

Endpoint Amazon VPC per Amazon S3

Per motivi di sicurezza, molti clienti AWS eseguono le proprie applicazioni all'interno di un ambiente Amazon Virtual Private Cloud (Amazon VPC). Con Amazon VPC, è possibile avviare le istanze Amazon EC2 in un Virtual Private Cloud (VPC), che è isolato logicamente da altre reti, inclusa la rete Internet pubblica. Con un Amazon VPC, puoi controllarne l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete, nonché le impostazioni di sicurezza.

> **Note**
>
> Se hai creato il tuo account AWS dopo il 04-12-2013, disponi già di un VPC di default in ogni regione AWS. Puoi iniziare a utilizzare il tuo VPC predefinito subito, senza ulteriori configurazioni.

Per ulteriori informazioni, consulta VPC e sottoreti predefiniti nella Guida per l'utente di Amazon VPC.

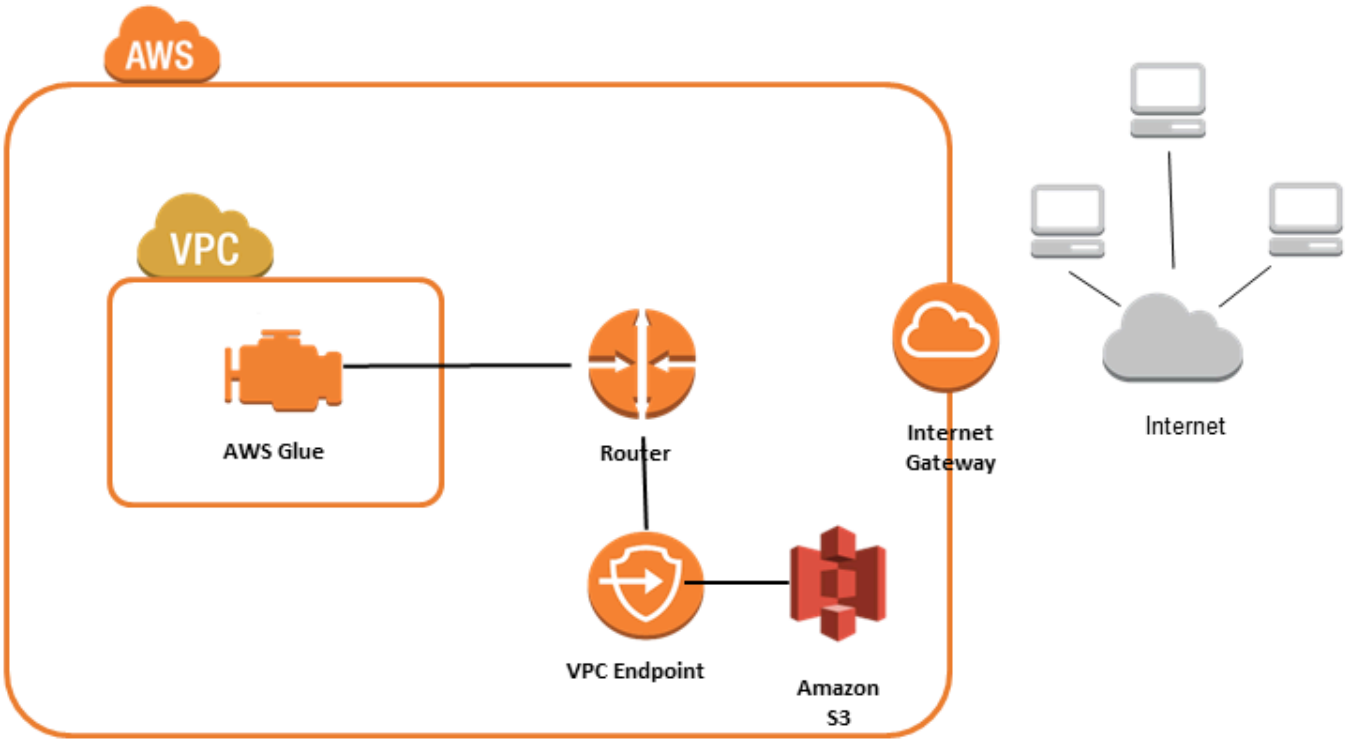
Molti clienti esprimono legittime preoccupazioni in materia di sicurezza e di privacy quando si tratta di inviare e ricevere dati tramite la rete Internet pubblica. I clienti possono risolvere questi problemi usando una rete privata virtuale (VPN) per instradare tutto il traffico di rete Amazon S3 tramite la propria infrastruttura di rete aziendale. Questo approccio tuttavia può introdurre problematiche relative alla larghezza di banda e alla disponibilità.

Gli endpoint VPC per Amazon S3 possono mitigare queste difficoltà. Un endpoint VPC per Amazon S3 permette a AWS Glue di usare indirizzi IP privati per accedere ad Amazon S3 senza esposizione nella rete Internet pubblica. AWS Glue non richiede indirizzi IP pubblici e non è necessario disporre di un gateway Internet, un dispositivo NAT o un gateway virtuale privato all'interno del VPC. Puoi utilizzare policy endpoint per controllare l'accesso ad Amazon S3. Il traffico tra il VPC e il servizio AWS non lascia la rete Amazon.

Quando crei un endpoint VPC per Amazon S3, qualsiasi richiesta a un endpoint Amazon S3 all'interno della regione (ad esempio s3.us-west-2.amazonaws.com) viene instradata verso un endpoint Amazon S3 privato all'interno della rete Amazon. Non è necessario modificare le applicazioni in esecuzione sulle istanze Amazon EC2 nel tuo VPC: il nome dell'endpoint rimane invariato, ma l'instradamento verso Amazon S3 rimane interamente all'interno della rete Amazon e non accede alla rete Internet pubblica.

Per ulteriori informazioni sugli endpoint VPC, consulta Endpoint VPC nella Guida per l'utente di Amazon VPC.

Il diagramma seguente mostra in che modo AWS Glue può usare un endpoint VPC per accedere ad Amazon S3.



Per configurare l'accesso ad Amazon S3

1. Accedere ad AWS Management Console e aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel riquadro di navigazione a sinistra, scegli Endpoints (Endpoint).
3. Scegli Create Endpoint (Crea endpoint) e segui la procedura per creare un endpoint VPC Amazon S3 di tipo Gateway.

Connessioni ad Amazon DocumentDB

Puoi utilizzare AWS Glue per Spark per la lettura e la scrittura su tabelle in Amazon DocumentDB. Puoi connetterti ad Amazon DocumentDB utilizzando le credenziali archiviate in AWS Secrets Manager tramite una connessione AWS Glue.

Per ulteriori informazioni su Amazon DocumentDB, consulta la documentazione di Amazon DocumentDB.

> **Note**
>
> I cluster elastici di Amazon DocumentDB non sono attualmente supportati quando si utilizza il connettore AWS Glue. Per ulteriori informazioni sui cluster elastici, consulta la pagina Using Amazon DocumentDB elastic clusters.

Lettura e scrittura nelle raccolte Amazon DocumentDB

> **Note**
>
> Quando crei un processo ETL a cui si connette Amazon DocumentDB, per la proprietà del processo `Connections`, devi designare un oggetto connessione che specifica il cloud privato virtuale (VPC) in cui Amazon DocumentDB è in esecuzione. Per l'oggetto connessione, il tipo di connessione deve essere `JDBC`, e `JDBC URL` deve essere `mongo://<DocumentDB_host>:27017`.

> **Note**
>
> Questi esempi di codice sono stati sviluppati per AWS Glue 3.0. Per effettuare la migrazione verso AWS Glue 4.0, consulta the section called “MongoDB”. Il parametro `uri` è cambiato.

> **Note**
>
> Quando si utilizza Amazon DocumentDB, `retryWrites` deve essere impostato su false in determinate situazioni, ad esempio quando il documento scritto specifica `_id`. Per ulteriori informazioni, consulta Differenze funzionali con MongoDB nella documentazione di Amazon DocumentDB.

Il seguente script Python dimostra l'utilizzo di tipi e opzioni di connessione per la lettura e la scrittura su Amazon DocumentDB.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext, SparkConf
from awsglue.context import GlueContext
from awsglue.job import Job
import time

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'], args)

output_path = "s3://some_bucket/output/" + str(time.time()) + "/"
documentdb_uri = "mongodb://<mongo-instanced-ip-address>:27017"
documentdb_write_uri = "mongodb://<mongo-instanced-ip-address>:27017"

read_docdb_options = {
    "uri": documentdb_uri,
    "database": "test",
    "collection": "coll",
    "username": "username",
    "password": "1234567890",
    "ssl": "true",
    "ssl.domain_match": "false",
    "partitioner": "MongoSamplePartitioner",
    "partitionerOptions.partitionSizeMB": "10",
    "partitionerOptions.partitionKey": "_id"
}

write_documentdb_options = {
    "retryWrites": "false",
    "uri": documentdb_write_uri,
    "database": "test",
    "collection": "coll",
    "username": "username",
    "password": "pwd"
}
```

```
# Get DynamicFrame from  DocumentDB
dynamic_frame2 =
 glueContext.create_dynamic_frame.from_options(connection_type="documentdb",

 connection_options=read_docdb_options)

# Write DynamicFrame to MongoDB and DocumentDB
glueContext.write_dynamic_frame.from_options(dynamic_frame2,
 connection_type="documentdb",

 connection_options=write_documentdb_options)

job.commit()
```

Il seguente script Scala dimostra l'utilizzo di tipi e opzioni di connessione per la lettura e la scrittura su Amazon DocumentDB.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  val DOC_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  val DOC_WRITE_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  lazy val documentDBJsonOption = jsonOptions(DOC_URI)
  lazy val writeDocumentDBJsonOption = jsonOptions(DOC_WRITE_URI)
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Get DynamicFrame from DocumentDB
    val resultFrame2: DynamicFrame = glueContext.getSource("documentdb",
 documentDBJsonOption).getDynamicFrame()
```

```
    // Write DynamicFrame to DocumentDB
    glueContext.getSink("documentdb", writeJsonOption).writeDynamicFrame(resultFrame2)

    Job.commit()
  }

  private def jsonOptions(uri: String): JsonOptions = {
    new JsonOptions(
      s"""{"uri": "${uri}",
         |"database":"test",
         |"collection":"coll",
         |"username": "username",
         |"password": "pwd",
         |"ssl":"true",
         |"ssl.domain_match":"false",
         |"partitioner": "MongoSamplePartitioner",
         |"partitionerOptions.partitionSizeMB": "10",
         |"partitionerOptions.partitionKey": "_id"}""".stripMargin)
  }
}
```

Indicazioni di riferimento alle opzioni di connessione ad Amazon DocumentDB

Indica una connessione ad Amazon DocumentDB (con compatibilità MongoDB).

Le opzioni di connessione differiscono per una connessione sorgente e una connessione sink.

"connectionType": "documentdb" come sorgente

Utilizzare le seguenti opzioni di connessione con `"connectionType": "documentdb"` come origine:

- `"uri"`: (obbligatorio) l'host Amazon DocumentDB da cui leggere, formattato come `mongodb://<host>:<port>`.
- `"database"`: (Obbligatorio) il database di Amazon DocumentDB da cui leggere.
- `"collection"`: (Obbligatorio) la raccolta di Amazon DocumentDB da cui leggere.
- `"username"`: (Obbligatorio) il nome utente di Amazon DocumentDB.
- `"password"`: (Obbligatorio) la password di Amazon DocumentDB.
- `"ssl"`: (Obbligatorio se si utilizza SSL) se la connessione utilizza SSL, è necessario includere questa opzione con il valore `"true"`.

- `"ssl.domain_match"`: (Obbligatorio se si utilizza SSL) se la connessione utilizza SSL, è necessario includere questa opzione con il valore `"false"`.
- `"batchSize"`: (Facoltativo): il numero di documenti da restituire per ogni batch, utilizzato all'interno del cursore dei batch interni.
- `"partitioner"`: (Facoltativo): il nome della classe del partizionatore per la lettura dei dati di input da Amazon DocumentDB. Il connettore fornisce i seguenti partizionatori:
  - `MongoDefaultPartitioner` (impostazione predefinita) (non supportato in AWS Glue 4.0)
  - `MongoSamplePartitioner` (non supportato in AWS Glue 4.0)
  - `MongoShardedPartitioner`
  - `MongoSplitVectorPartitioner`
  - `MongoPaginateByCountPartitioner`
  - `MongoPaginateBySizePartitioner` (non supportato in AWS Glue 4.0)
- `"partitionerOptions"` (Facoltativo): opzioni per il partizionatore designato. Per ogni partizionatore sono supportate le seguenti opzioni:
  - `MongoSamplePartitioner`: `partitionKey`, `partitionSizeMB`, `samplesPerPartition`
  - `MongoShardedPartitioner`: `shardkey`
  - `MongoSplitVectorPartitioner`: `partitionKey`, partitionSizeMB
  - `MongoPaginateByCountPartitioner`: `partitionKey`, `numberOfPartitions`
  - `MongoPaginateBySizePartitioner`: `partitionKey`, partitionSizeMB

  Per ulteriori informazioni su queste opzioni, vedere Partitioner Configuration (Configurazione partizionatore) nella documentazione di MongoDB.

"connectionType": "documentdb" come sink

Utilizzare le seguenti opzioni di connessione con `"connectionType": "documentdb"` come sink:

- `"uri"`: (Obbligatorio) l'host Amazon DocumentDB su cui scrivere, formattato come `mongodb://<host>:<port>`.
- `"database"`: (Obbligatorio) il database di Amazon DocumentDB su cui scrivere.
- `"collection"`: (Obbligatorio) la raccolta di Amazon DocumentDB su cui scrivere.
- `"username"`: (Obbligatorio) il nome utente di Amazon DocumentDB.
- `"password"`: (Obbligatorio) la password di Amazon DocumentDB.

- `"extendedBsonTypes"`: (Facoltativo) se il valore è `true`, abilita i tipi BSON estesi durante la scrittura dei dati su Amazon DocumentDB. Il valore predefinito è `true`.
- `"replaceDocument"`: (Facoltativo) Se il valore è `true`, sostituisce l'intero documento quando si salvano set di dati che contengono un campo `_id`. Se il valore è `false`, vengono aggiornati solo i campi del documento che corrispondono ai campi del set di dati. Il valore predefinito è `true`.
- `"maxBatchSize"`: (Facoltativo): la dimensione massima del batch per le operazioni in blocco durante il salvataggio dei dati. Il valore predefinito è 512.
- `"retryWrites"`: (Facoltativo): riprova automaticamente alcune operazioni di scrittura una sola volta se AWS Glue rileva un errore di rete.

OpenSearch Connessioni di servizio

Puoi usare AWS Glue for Spark per leggere e scrivere su tabelle in OpenSearch Service in AWS Glue 4.0 e versioni successive. È possibile definire cosa leggere dal OpenSearch servizio con una OpenSearch query. Ti connetti al OpenSearch servizio utilizzando le credenziali di autenticazione di base HTTP archiviate AWS Secrets Manager tramite una connessione AWS Glue. Questa funzionalità non è compatibile con OpenSearch Service serverless.

Per ulteriori informazioni su Amazon OpenSearch Service, consulta la documentazione OpenSearch di Amazon Service.

Configurazione delle connessioni OpenSearch al servizio

Per connetterti a OpenSearch Service da AWS Glue, dovrai creare e archiviare le tue credenziali OpenSearch Service in modo AWS Secrets Manager segreto, quindi associare quel segreto a una connessione OpenSearch Service AWS Glue.

Prerequisiti:

- Identifica l'endpoint del dominio, *`AOSEndpoint`* e la porta, *`AOSport`* da cui desideri leggere o crea la risorsa seguendo le istruzioni nella documentazione di Amazon Service. OpenSearch Per ulteriori informazioni sulla creazione di un dominio, consulta Creazione e gestione di domini Amazon OpenSearch Service nella documentazione di Amazon OpenSearch Service.

  Un endpoint OpenSearch di dominio Amazon Service avrà il seguente modulo predefinito, https:// search - *`DomainName`* -. *`unstructuredIdContent regione .es.amazonaws.com.`* Per ulteriori informazioni sull'identificazione dell'endpoint del tuo dominio, consulta Creazione e

gestione dei domini Amazon OpenSearch Service nella documentazione di Amazon OpenSearch Service.

Identifica o genera credenziali di autenticazione di base HTTP, aosUser e aosPassword per il tuo dominio.

Per configurare una connessione al OpenSearch servizio:

1. InAWS Secrets Manager, crea un segreto utilizzando le tue credenziali OpenSearch di servizio. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `opensearch.net.http.auth.user` con il valore *aosUser*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `opensearch.net.http.auth.pass` con il valore *aosPassword*.
2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per l'uso futuro in AWS Glue.
   - Quando selezioni un tipo di connessione, seleziona OpenSearch Servizio.
   - Quando selezioni un endpoint di dominio, fornisci *aosEndpoint*.
   - Quando selezioni una porta, fornisci *aosPort*.
   - Quando selezioni il Segreto AWS, fornisci *secretName*.

Dopo aver creato una connessione OpenSearch al servizio AWS Glue, dovrai eseguire i seguenti passaggi prima di eseguire il lavoro AWS Glue:

- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *secretName*.
- Nella configurazione del processo AWS Glue, fornisci *connectionName* come Connessione di rete aggiuntiva.

Lettura dagli indici OpenSearch dei servizi

Prerequisiti:

- Un indice OpenSearch di servizio da cui desideri leggere, *aosIndex*.
- Una connessione AWS Glue OpenSearch Service configurata per fornire informazioni di autenticazione e posizione di rete. Per acquisirla, completa i passaggi della procedura precedente, Per configurare una connessione al OpenSearch servizio. Sarà necessario il nome della connessione AWS Glue, *connectionName*.

Questo esempio legge un indice da Amazon OpenSearch Service. Dovrai fornire il parametro `pushdown`.

Per esempio:

```
opensearch_read = glueContext.create_dynamic_frame.from_options(
    connection_type="opensearch",
    connection_options={
        "connectionName": "connectionName",
        "opensearch.resource": "aosIndex",
        "pushdown": "true",
    }
)
```

Puoi anche fornire una stringa di query per filtrare i risultati restituiti nel tuo DynamicFrame. Sarà necessario configurare `opensearch.query`.

`opensearch.query` può accettare una stringa di parametri di query URL *queryString* o un oggetto di query DSL JSON *queryObject* Per ulteriori informazioni sulla query DSL, vedere Query DSL nella OpenSearch documentazione. Per fornire una stringa di parametri di query URL, anteponi `?q=` alla query, come faresti con un URL completo. Per fornire un oggetto di query DSL, la stringa evita l'oggetto JSON prima di fornirlo.

Per esempio:

```
            queryObject = "{ "query": { "multi_match": { "query": "Sample", "fields":
[ "sample" ] } } }"
            queryString = "?q=queryString"

            opensearch_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="opensearch",
    connection_options={
        "connectionName": "connectionName",
```

```
        "opensearch.resource": "aosIndex",
        "opensearch.query": queryString,
        "pushdown": "true",
    }
)
```

Per ulteriori informazioni su come creare una query al di fuori della relativa sintassi specifica, vedi Sintassi delle stringhe di query nella documentazione. OpenSearch

Quando si leggono da OpenSearch raccolte che contengono dati di tipo array, è necessario specificare quali campi sono di tipo array nella chiamata al metodo utilizzando il `opensearch.read.field.as.array.include` parametro.

Ad esempio, durante la lettura del documento seguente incontrerai i campi di array `genre` e `actor`:

```
{
    "_index": "movies",
    "_id": "2",
    "_version": 1,
    "_seq_no": 0,
    "_primary_term": 1,
    "found": true,
    "_source": {
        "director": "Frankenheimer, John",
        "genre": [
            "Drama",
            "Mystery",
            "Thriller",
            "Crime"
        ],
        "year": 1962,
        "actor": [
            "Lansbury, Angela",
            "Sinatra, Frank",
            "Leigh, Janet",
            "Harvey, Laurence",
            "Silva, Henry",
            "Frees, Paul",
            "Gregory, James",
            "Bissell, Whit",
            "McGiver, John",
            "Parrish, Leslie",
            "Edwards, James",
```

```
            "Flowers, Bess",
            "Dhiegh, Khigh",
            "Payne, Julie",
            "Kleeb, Helen",
            "Gray, Joe",
            "Nalder, Reggie",
            "Stevens, Bert",
            "Masters, Michael",
            "Lowell, Tom"
        ],
        "title": "The Manchurian Candidate"
    }
}
```

In questo caso, dovrai includere i nomi dei campi in questione nella chiamata al metodo. Per esempio:

```
"opensearch.read.field.as.array.include": "genre,actor"
```

Se il campo dell'array è annidato all'interno della struttura del documento, fai riferimento a esso utilizzando la notazione a punti: `"genre,actor,foo.bar.baz"`. Ciò specificherebbe un array `baz` incluso nel documento di origine tramite il documento incorporato `foo` contenente il documento incorporato `bar`.

Scrittura nelle tabelle OpenSearch dei servizi

Questo esempio scrive informazioni da un *`DynamicFrame`* to OpenSearch Service esistente DynamicFrame. Se l'indice contiene già informazioni, AWS Glue aggiungerà i dati dal tuo DynamicFrame. Dovrai fornire il parametro `pushdown`.

Prerequisiti:

- Una tabella dei OpenSearch servizi su cui scrivere. Avrai bisogno delle informazioni di identificazione per la tabella. Lo chiameremo *`tableName`*.
- Una connessione AWS Glue OpenSearch Service configurata per fornire informazioni di autenticazione e posizione di rete. Per acquisirla, completa i passaggi della procedura precedente, Per configurare una connessione al OpenSearch servizio. Sarà necessario il nome della connessione AWS Glue, *`connectionName`*.

Per esempio:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="opensearch",
    connection_options={
      "connectionName": "connectionName",
      "opensearch.resource": "aosIndex",
    },
)
```

OpenSearch Riferimento all'opzione di connessione al servizio

- `connectionName`: obbligatorio. Utilizzato per la lettura/scrittura. Il nome di una connessione al AWS Glue OpenSearch Service configurata per fornire informazioni di autenticazione e posizione di rete al metodo di connessione utilizzato.
- `opensearch.resource`: obbligatorio. Utilizzato per la lettura/scrittura. Valori validi: nomi degli OpenSearch indici. Il nome dell'indice con cui interagirà il metodo di connessione.
- `opensearch.query`: utilizzato per la lettura. Valori validi: stringa con escape JSON o, quando inizia con `?`, la parte di ricerca di un URL. Una OpenSearch query che filtra ciò che deve essere recuperato durante la lettura. Per ulteriori informazioni sull'utilizzo di questo parametro, consulta la sezione precedente the section called “Leggi dal servizio OpenSearch ”.
- `pushdown` — Richiesto se. Utilizzato per la lettura. Valori validi: booleani. Indica a Spark di passare le query di lettura in OpenSearch modo che il database restituisca solo i documenti pertinenti.
- `opensearch.read.field.as.array.include`: richiesto se si leggono dati di tipo array. Utilizzato per la lettura. Valori validi: elenchi di nomi di campi separati da virgole. Specifica i campi da leggere come matrici dai documenti. OpenSearch Per ulteriori informazioni sull'utilizzo di questo parametro, consulta la sezione precedente the section called “Leggi dal servizio OpenSearch ”.

Connessioni Redshift

Puoi usare AWS Glue for Spark per leggere e scrivere su tabelle nei database Amazon Redshift. Quando si connette ai database Amazon Redshift, AWS Glue sposta i dati tramite Amazon S3 per ottenere il massimo throughput, utilizzando SQL e comandi Amazon Redshift. `COPY` `UNLOAD` In AWS Glue 4.0 e versioni successive, puoi utilizzare l'integrazione Amazon Redshift per Apache Spark per leggere e scrivere con ottimizzazioni e funzionalità specifiche di Amazon Redshift oltre a quelle disponibili durante la connessione tramite versioni precedenti.

Scopri come AWS Glue sta semplificando più che mai per gli utenti di Amazon Redshift la migrazione a AWS Glue per l'integrazione dei dati senza server e l'ETL.

Configurazione delle connessioni Redshift

Per utilizzare i cluster Amazon Redshift in AWS Glue, sono necessari alcuni prerequisiti:

- Una directory Amazon S3 da utilizzare per l'archiviazione temporanea durante la lettura e la scrittura sul database.
- Un Amazon VPC che consente la comunicazione tra il cluster Amazon Redshift, il job AWS Glue e la directory Amazon S3.
- Autorizzazioni IAM appropriate sul job AWS Glue e sul cluster Amazon Redshift.

Configurazione dei ruoli IAM

Configurazione del ruolo per il cluster Amazon Redshift

Il tuo cluster Amazon Redshift deve essere in grado di leggere e scrivere su Amazon S3 per integrarsi con AWS Glue jobs. Per consentire ciò, puoi associare i ruoli IAM al cluster Amazon Redshift a cui desideri connetterti. Il tuo ruolo dovrebbe disporre di una policy che consenta la lettura e la scrittura nella tua directory temporanea di Amazon S3. Il tuo ruolo dovrebbe avere un rapporto di fiducia che consenta al servizio `redshift.amazonaws.com` di `AssumeRole`.

Associazione di un ruolo IAM ad Amazon Redshift

1. Prerequisiti: un bucket o una directory Amazon S3 utilizzato per l'archiviazione temporanea dei file.
2. Identifica le autorizzazioni Amazon S3 che occorreranno al cluster Amazon Redshift. Quando si spostano dati da e verso un cluster Amazon Redshift, i job AWS Glue emettono istruzioni COPY e UNLOAD su Amazon Redshift. Se il tuo job modifica una tabella in Amazon Redshift, AWS Glue emetterà anche istruzioni CREATE LIBRARY. Per informazioni sulle autorizzazioni specifiche di Amazon S3 necessarie ad Amazon Redshift per eseguire queste istruzioni, consulta la documentazione di Amazon Redshift: Amazon Redshift: Permissions to access other Resources. AWS
3. Nella console IAM, crea una policy IAM con le autorizzazioni necessarie. Per ulteriori informazioni sulla creazione di una policy, consulta la pagina Creazione di policy IAM.

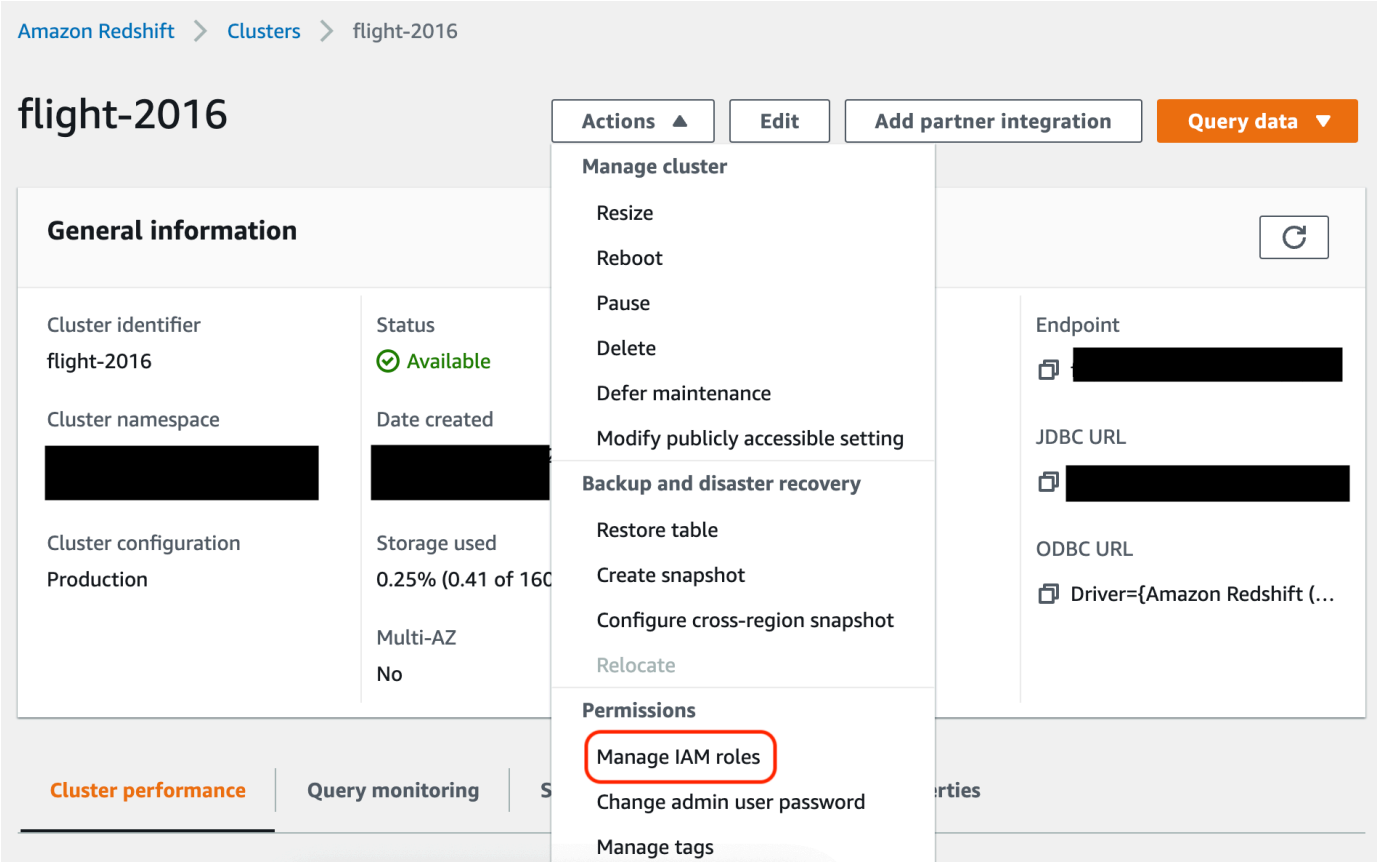
4. Da IAM, crea un ruolo e un rapporto di fiducia che consenta ad Amazon Redshift di assumere il ruolo. Segui le istruzioni nella documentazione IAM Per creare un ruolo per un servizio (console) AWS

   - Quando ti viene chiesto di scegliere un caso d'uso del AWS servizio, scegli «Redshift - Personalizzabile».
   - Quando ti viene chiesto di collegare una policy, scegli la policy che hai definito in precedenza.

   > **Note**
   >
   > Per ulteriori informazioni sulla configurazione dei ruoli per Amazon Redshift, consulta Autorizzazione di Amazon Redshift ad AWS accedere ad altri servizi per tuo conto nella documentazione di Amazon Redshift.

5. Da Amazon Redshift, associa il ruolo al tuo cluster Amazon Redshift. Segui le istruzioni nella documentazione di Amazon Redshift.

   Seleziona l'opzione evidenziata nella console Amazon Redshift per configurare questa impostazione:



> **Note**
>
> Per impostazione predefinita, i lavori AWS Glue passano le credenziali temporanee di Amazon Redshift create utilizzando il ruolo che hai specificato per eseguire il lavoro. Non è consigliabile utilizzare queste credenziali. Per motivi di sicurezza, queste credenziali scadono dopo 1 ora.

Imposta il ruolo per il lavoro AWS Glue

Il job AWS Glue richiede un ruolo per accedere al bucket Amazon S3. Non sono necessarie le autorizzazioni IAM per il cluster Amazon Redshift, il tuo accesso è controllato dalla connettività in Amazon VPC e dalle credenziali del database.

Configurazione di Amazon VPC

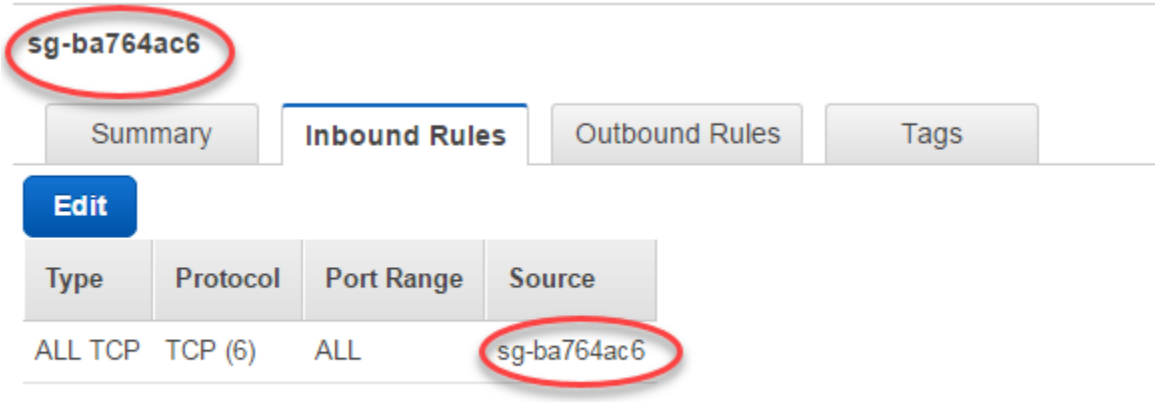
Per configurare l'accesso ai datastore Amazon Redshift

1. Accedi AWS Management Console e apri la console Amazon Redshift all'indirizzo https://console.aws.amazon.com/redshiftv2/.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Seleziona il nome del cluster al quale desideri accedere da AWS Glue.
4. Nella sezione Cluster Properties (Proprietà cluster) scegli un gruppo di sicurezza in VPC security groups (Gruppi di sicurezza VPC) per permettere l'uso a AWS Glue. Registra il nome del gruppo di sicurezza scelto per riferimenti futuri. La scelta del gruppo di sicurezza apre l'elenco Security Groups (Gruppi di sicurezza) della console Amazon EC2.
5. Scegli il gruppo di sicurezza da modificare e passa alla scheda Inbound (In entrata).
6. Aggiungi una regola autoreferenziale per permettere ai componenti di AWS Glue di comunicare tra loro. In particolare, aggiungi o verifica che sia presente una regola con Type (Tipo) `All TCP`, Protocol (Protocollo) `TCP`, Port Range (Intervallo porte) che include tutte le porte e Source (Origine) corrispondente al nome del gruppo di sicurezza indicato da Group ID (ID gruppo).

   La regola in entrata è simile alla seguente:

Type	Protocollo	Intervallo porte	Origine
Tutte le regole TCP	TCP	0–65535	database-security-group

   Per esempio:



7. Aggiungi una regola anche per il traffico in uscita. Quindi, apri il traffico in uscita per tutte le porte, ad esempio:

Type	Protocollo	Intervallo porte	Destinazione
All Traffic	ALL	ALL	0.0.0.0/0

In alternativa, crea una regola autoreferenziale in cui Type (Tipo) `All TCP`, Protocol (Protocollo) sta per `TCP` e Port Range (Intervallo porte) include tutte le porte, la cui Destination (Destinazione) ha lo stesso nome del gruppo di sicurezza del Group ID (ID gruppo). Se usi un endpoint VPC Amazon S3, aggiungi anche una regola HTTPS per l'accesso di Amazon S3. *`s3-prefix-list-id`* è richiesto nella regola del gruppo di sicurezza per consentire il traffico dal VPC all'endpoint VPC Amazon S3.

Per esempio:

Type	Protocollo	Intervallo porte	Destinazione
Tutte le regole TCP	TCP	0–65535	*`security-group`*
HTTPS	TCP	443	*`s3- prefix-list-id`*

Configura AWS Glue

Dovrai creare una connessione AWS Glue Data Catalog che fornisca informazioni sulla connessione Amazon VPC.

Per configurare la connettività Amazon Redshift (Amazon VPC) a AWS Glue nella console

1. Crea una connessione a Catalogo dati seguendo i passaggi indicati nella sezione the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per il passaggio successivo.

   - Quando selezioni un Tipo di connessione, seleziona Amazon Redshift.
   - Quando selezioni un Cluster Redshift, seleziona il tuo cluster in base al nome.
   - Fornisci informazioni di connessione predefinite per un utente Amazon Redshift sul tuo cluster.
   - Le impostazioni di Amazon VPC verranno configurate automaticamente.

   > **Note**
   >
   > Quando crei una connessione Amazon Redshift tramite l'SDK AWS , dovrai fornire manualmente il valore `PhysicalConnectionRequirements` per il tuo Amazon VPC.

2. Nella configurazione del lavoro AWS Glue, fornisci *ConnectionName* come connessione di rete aggiuntiva.

Esempio: lettura da tabelle Amazon Redshift

È possibile leggere da cluster Amazon Redshift e ambienti Amazon Redshift serverless.

Prerequisiti: una tabella Amazon Redshift da cui desideri leggere. Segui i passaggi della sezione precedente, the section called “Configurazione di Redshift” dopodiché dovresti avere l'URI Amazon S3 per una directory temporanea, *temp-s3-dir* e un ruolo IAM,, (nell'account). *rs-role-namerole-account-id*

Using the Data Catalog

Prerequisiti aggiuntivi: un database Catalogo dati e una tabella dai quali desideri che la tabella Amazon Redshift legga. Per ulteriori informazioni su Catalogo dati, consulta la pagina Catalogo

dati e crawler. Dopo aver creato una voce per la tabella Amazon Redshift, identificherai la tua connessione con un *redshift-dc-database-name*segno e. *redshift-table-name*

Configurazione: nelle opzioni della funzione identificherai la tabella di Catalogo dati con i parametri `database` e `table_name`. Identificherai la tua directory temporanea Amazon S3 con `redshift_tmp_dir`. Dovrai inoltre fornire *rs-role-name*l'utilizzo della `aws_iam_role` chiave nel `additional_options` parametro.

```
 glueContext.create_dynamic_frame.from_catalog(
    database = "redshift-dc-database-name",
    table_name = "redshift-table-name",
    redshift_tmp_dir = args["temp-s3-dir"],
    additional_options = {"aws_iam_role": "arn:aws:iam::role-account-id:role/rs-
role-name"})
```

Connecting directly

Prerequisiti aggiuntivi: è necessario il nome della tabella Amazon Redshift (. *redshift-table-name* Avrai bisogno delle informazioni di connessione JDBC per il cluster Amazon Redshift in cui è archiviata quella tabella. *Fornirai le informazioni di connessione con host, porta redshift-database-name, nome utente e password.*

Quando lavori con i cluster Amazon Redshift, puoi recuperare le informazioni di connessione dalla console Amazon Redshift. Se utilizzi Amazon Redshift serverless, consulta la sezione Connecting to Amazon Redshift Serverless nella documentazione di Amazon Redshift.

Configurazione: nelle opzioni della funzione identificherai i parametri di connessione con `url`, `dbtable`, `user` e `password`. Identificherai la tua directory temporanea Amazon S3 con `redshift_tmp_dir`. Quando utilizzi `from_options`, puoi specificare il tuo ruolo IAM utilizzando `aws_iam_role`. La sintassi è simile alla connessione tramite Catalogo dati, ma è necessario inserire i parametri nella mappa `connection_options`.

È una cattiva pratica codificare le password negli script AWS Glue. Valuta la possibilità di archiviare le password AWS Secrets Manager e recuperarle nello script con SDK for Python (Boto3).

```
my_conn_options = {
    "url": "jdbc:redshift://host:port/redshift-database-name",
    "dbtable": "redshift-table-name",
    "user": "username",
    "password": "password",
    "redshiftTmpDir": args["temp-s3-dir"],
    "aws_iam_role": "arn:aws:iam::account id:role/rs-role-name"
}

df = glueContext.create_dynamic_frame.from_options("redshift", my_conn_options)
```

Esempio: scrittura su tabelle Amazon Redshift

È possibile scrivere su cluster Amazon Redshift e ambienti Amazon Redshift serverless.

Prerequisiti: un cluster Amazon Redshift e segui i passaggi della the section called “Configurazione di Redshift” sezione precedente, dopodiché dovresti avere l'URI Amazon S3 per una directory temporanea*, temp-s3-dir e un ruolo IAM,,* (nell'account). *rs-role-namerole-account-id* Avrai anche bisogno di un `DynamicFrame` del quale desideri scrivere il contenuto nel database.

Using the Data Catalog

Prerequisiti aggiuntivi: un database Catalogo dati sul quale desideri che scrivano il cluster e la tabella Amazon Redshift. Per ulteriori informazioni su Catalogo dati, consulta la pagina Catalogo dati e crawler. Identificherai la tua connessione con e la tabella di destinazione con. *redshift-dc-database-nameredshift-table-name*

Configurazione: nelle opzioni della funzione identificherai il database di Catalogo dati con il parametro `database`, quindi fornirai la tabella con `table_name`. Identificherai la tua directory temporanea Amazon S3 con `redshift_tmp_dir`. Fornirai anche *rs-role-name*l'utilizzo della `aws_iam_role` chiave nel `additional_options` parametro.

```
glueContext.write_dynamic_frame.from_catalog(
   frame = input dynamic frame,
   database = "redshift-dc-database-name",
   table_name = "redshift-table-name",
   redshift_tmp_dir = args["temp-s3-dir"],
```

```
    additional_options = {"aws_iam_role": "arn:aws:iam::account-id:role/rs-role-
name"})
```

Connecting through a AWS Glue connection

È possibile connettersi ad Amazon Redshift direttamente utilizzando il metodo `write_dynamic_frame.from_options`. Tuttavia, anziché inserire i dettagli di connessione direttamente nello script, puoi fare riferimento ai dettagli di connessione archiviati in una connessione a Catalogo dati con il metodo `from_jdbc_conf`. È possibile eseguire questa operazione senza effettuare il crawling o creare tabelle di Catalogo dati per il database. Per ulteriori informazioni sulle connessioni a Catalogo dati, consulta la pagina Connessione ai dati.

Prerequisiti aggiuntivi: una connessione a Catalogo dati per il database, una tabella Amazon Redshift da cui desideri leggere

Configurazione: identificherai la tua connessione al Data Catalog con *dc-connection-name*. Identificherai il database e la tabella Amazon Redshift con *redshift-table-name*e. *redshift-database-name* Fornirai le informazioni di connessione a Catalogo dati con `catalog_connection` e le informazioni relative ad Amazon Redshift con `dbtable` e `database`. La sintassi è simile alla connessione tramite Catalogo dati, ma è necessario inserire i parametri nella mappa `connection_options`.

```
my_conn_options = {
    "dbtable": "redshift-table-name",
    "database": "redshift-database-name",
    "aws_iam_role": "arn:aws:iam::role-account-id:role/rs-role-name"
}

glueContext.write_dynamic_frame.from_jdbc_conf(
    frame = input dynamic frame,
    catalog_connection = "dc-connection-name",
    connection_options = my_conn_options,
    redshift_tmp_dir = args["temp-s3-dir"])
```

Indicazioni di riferimento alle opzione di connessione ad Amazon Redshift

Le opzioni di connessione di base utilizzate per tutte le connessioni JDBC AWS Glue per configurare informazioni come `url` `user` e `password` sono coerenti per tutti i tipi JDBC. Per ulteriori informazioni sui parametri JDBC standard, consulta la pagina the section called “Parametri di connessione JDBC”.

Il tipo di connessione Amazon Redshift richiede alcune opzioni di connessione aggiuntive:

- `"redshiftTmpDir"`: (obbligatorio) il percorso Amazon S3 in cui i dati temporanei possono essere caricati durante la copia dal database.
- `"aws_iam_role"`: (facoltativo) l'ARN di un ruolo IAM. Il job AWS Glue passerà questo ruolo al cluster Amazon Redshift per concedere al cluster le autorizzazioni necessarie per completare le istruzioni del job.

Opzioni di connessione aggiuntive disponibili in AWS Glue 4.0+

Puoi anche passare le opzioni per il nuovo connettore Amazon Redshift tramite le opzioni di connessione AWS Glue. Per un elenco completo delle opzioni di connettori supportate, consulta la sezione Parametri SQL Spark in Integrazione di Amazon Redshift per Apache Spark.

Per comodità, ribadiamo di seguito alcune nuove opzioni:

Nome	Obbligatorio	Predefinito	Descrizione
autopushdown	No	TRUE	Applica il pushdown di predicati e query acquisendo e analizzando i piani logici di Spark per le operazioni SQL. Le operazioni vengono tradotte in una query SQL e quindi eseguite in Amazon Redshift per migliorare le prestazioni.
autopushdown.s3_result_cache	No	FALSE	Memorizza nella cache la query SQL

Nome	Obbligatorio	Predefinito	Descrizione
			per scaricare i dati sulla mappatura dei percorsi di Amazon S3 in memoria, in modo che la stessa query non debba essere eseguita nuovamente nella stessa sessione di Spark. Supportato solo quando `autopushdown` è abilitato.
unload_s3_format	No	PARQUET	PARQUET: scarica i risultati della query in formato Parquet. TESTO: scarica i risultati della query in formato testo delimitato da barra verticale.
sse_kms_key	No	N/D	La chiave AWS SSE-KMS da utilizzare per la crittografia durante `UNLOAD` le operazioni anziché la crittografia predefinita per. AWS

Nome	Obbligatorio	Predefinito	Descrizione
extracopyoptions	No	N/D	Un elenco di opzioni ulteriori da aggiunger e al comando `COPY` di Amazon Redshift durante il caricamen to dei dati, come `TRUNCATECOLUMNS` o `MAXERROR n` (per altre opzioni, consulta COPY: parametri facoltativi). È importante notare che, poiché queste opzioni vengono aggiunte alla fine del comando `COPY`, è possibile utilizzar e solo le opzioni rilevanti alla fine del comando. Questo dovrebbe coprire la maggior parte dei casi d'uso possibili.
cvsnullstring (sperimentale)	No	NULL	Il valore di stringa da scrivere per i valori null quando si utilizza il `tempformat` CSV. Dovrebbe trattarsi di un valore che non è presente nei dati effettivi.

Questi nuovi parametri possono essere utilizzati nei seguenti modi.

Nuove opzioni per il miglioramento delle prestazioni

Il nuovo connettore introduce alcune nuove opzioni di miglioramento delle prestazioni:

- `autopushdown`: abilitato per impostazione predefinita.
- `autopushdown.s3_result_cache`: disabilitato per impostazione predefinita.
- `unload_s3_format`: `PARQUET` per impostazione predefinita.

Per informazioni sull'utilizzo di queste opzioni, consulta Integrazione di Amazon Redshift per Apache Spark. Si consiglia di non attivare `autopushdown.s3_result_cache` quando si eseguono operazioni di lettura e scrittura miste perché i risultati memorizzati nella cache potrebbero contenere informazioni obsolete. L'opzione `unload_s3_format` è impostata su `PARQUET` per impostazione predefinita per il comando `UNLOAD` per migliorare le prestazioni e ridurre i costi di archiviazione. Per utilizzare il comportamento predefinito del comando `UNLOAD`, reimposta l'opzione su `TEXT`.

Nuova opzione di crittografia per la lettura

Per impostazione predefinita, i dati nella cartella temporanea utilizzata da AWS Glue durante la lettura dei dati dalla tabella Amazon Redshift vengono crittografati tramite la crittografia `SSE-S3`. Per utilizzare le chiavi gestite dal cliente di AWS Key Management Service (AWS KMS) per crittografare i dati, puoi impostare da `("sse_kms_key" # kmsKey)` dove KSMKey proviene l'ID della chiave AWS KMS, anziché l'opzione di impostazione precedente nella versione 3.0. `("extraunloadoptions" # s"ENCRYPTED KMS_KEY_ID '$kmsKey'")` AWS Glue

```
datasource0 = glueContext.create_dynamic_frame.from_catalog(
  database = "database-name",
  table_name = "table-name",
  redshift_tmp_dir = args["TempDir"],
  additional_options = {"sse_kms_key":"<KMS_KEY_ID>"},
  transformation_ctx = "datasource0"
)
```

Supporto dell'URL JDBC basato su IAM

Il nuovo connettore supporta un URL JDBC basato su IAM, quindi non è necessario fornire le credenziali utente/password o un segreto. Con un URL JDBC basato su IAM, il connettore utilizza il ruolo di runtime del processo per accedere all'origine dati Amazon Redshift.

Fase 1: collegamento della seguente politica minima obbligatoria al ruolo di runtime del processo AWS Glue.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "redshift:GetClusterCredentials",
            "Resource": [
                "arn:aws:redshift:<region>:<account>:dbgroup:<cluster name>/*",
                "arn:aws:redshift:*:<account>:dbuser:*/*",
                "arn:aws:redshift:<region>:<account>:dbname:<cluster name>/<database
 name>"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "redshift:DescribeClusters",
            "Resource": "*"
        }
    ]
}
```

Fase 2: Uso dell'URL JDBC basato su IAM come segue. Specifica una nuova opzione `DbUser` con il nome utente Amazon Redshift con cui ti stai connettendo.

```
conn_options = {
     // IAM-based JDBC URL
    "url": "jdbc:redshift:iam://<cluster name>:<region>/<database name>",
    "dbtable": dbtable,
    "redshiftTmpDir": redshiftTmpDir,
    "aws_iam_role": aws_iam_role,
    "DbUser": "<Redshift User name>" // required for IAM-based JDBC URL
    }

redshift_write = glueContext.write_dynamic_frame.from_options(
    frame=dyf,
    connection_type="redshift",
    connection_options=conn_options
)
```

```
redshift_read = glueContext.create_dynamic_frame.from_options(
    connection_type="redshift",
    connection_options=conn_options
)
```

> **Note**
>
> Un `DynamicFrame` al momento supporta un URL JDBC basato su IAM solo con un `DbUser` nel flusso di lavoro `GlueContext.create_dynamic_frame.from_options`.

Migrazione da AWS Glue versione 3.0 alla versione 4.0

In AWS Glue 4.0, i job ETL hanno accesso a un nuovo connettore Amazon Redshift Spark e a un nuovo driver JDBC con diverse opzioni e configurazioni. Il nuovo connettore e driver Amazon Redshift sono stati progettati per le prestazioni e garantiscono la coerenza transazionale dei dati. Questi prodotti sono illustrati nella documentazione di Amazon Redshift. Per ulteriori informazioni, consultare:

- Integrazione di Amazon Redshift per Apache Spark
- Driver JDBC Amazon Redshift, versione 2.1

Restrizione dei nomi e degli identificatori di tabelle/colonne

Il nuovo connettore e il driver Amazon Redshift Spark hanno un requisito più limitato per il nome della tabella Redshift. Per ulteriori informazioni, consulta Nomi e identificatori per definire il nome della tabella Amazon Redshift. Il flusso di lavoro relativo ai segnalibri del processo potrebbe non funzionare con un nome di tabella che non corrisponde alle regole e con determinati caratteri, ad esempio uno spazio.

Se hai tabelle legacy con nomi non conformi alle regole dei nomi e degli identificatori e riscontri problemi con i segnalibri (processi che rielaborano i vecchi dati delle tabelle Amazon Redshift), ti consigliamo di rinominare le tabelle. Per ulteriori informazioni, consulta Esempi di ALTER TABLE.

Modifica del formato temporale predefinito in Dataframe

Il connettore Spark di AWS Glue versione 3.0 imposta automaticamente `tempformat` su CSV durante la scrittura su Amazon Redshift. Per continuità, in AWS Glue versione 3.0, `DynamicFrame` è ancora impostato su `tempformat` per l'uso di `CSV`. Se in precedenza hai utilizzato le API Spark

Dataframe direttamente con il connettore Spark di Amazon Redshift, puoi impostare `tempformat` in modo esplicito su CSV nelle opzioni `DataframeReader/Writer`. Altrimenti, `tempformat` è impostato su `AVRO` nel nuovo connettore Spark.

Modifica di comportamento: associazione del tipo di dati Amazon Redshift REAL al tipo di dati Spark FLOAT anziché DOUBLE

In AWS Glue versione 3.0, Amazon Redshift `REAL` viene convertito in un tipo `DOUBLE` Spark. Il nuovo connettore Amazon Redshift Spark ha aggiornato il comportamento in modo che il tipo `REAL` Amazon Redshift venga viene convertito e di nuovo dal tipo `FLOAT` Spark. Se hai un caso d'uso precedente in cui desideri ancora che il tipo `REAL` Amazon Redshift sia mappato a un tipo `DOUBLE` Spark, puoi utilizzare la seguente soluzione alternativa:

- Per un `DynamicFrame`, mappa il tipo `Float` a un tipo `Double` con `DynamicFrame.ApplyMapping`. Per un `Dataframe`, è necessario usare `cast`.

Esempio di codice:

```
dyf_cast = dyf.apply_mapping([('a', 'long', 'a', 'long'), ('b', 'float', 'b',
 'double')])
```

Connessioni Kafka

Indica una connessione a un cluster Kafka o a un cluster Amazon Managed Streaming for Apache Kafka.

Puoi utilizzare i metodi seguenti sotto l'oggetto `GlueContext` per utilizzare i registri da un'origine di streaming Kafka:

- `getCatalogSource`
- `getSource`
- `getSourceWithFormat`
- `createDataFrameFromOptions`

Se utilizzi `getCatalogSource`, il processo ha le informazioni sul nome della tabella e del database del catalogo dati e può utilizzarle per ottenere alcuni parametri di base per la lettura dal flusso Apache Kafka. Se utilizzi `getSource`, `getSourceWithFormat` o `createDataFrameFromOptions`, è necessario specificare esplicitamente questi parametri:

Puoi specificare queste opzioni utilizzando `connectionOptions` con `getSource` o `createDataFrameFromOptions`, `options` con `getSourceWithFormat` o `additionalOptions` con `getCatalogSource`.

Per osservazioni e restrizioni sui processi ETL dei flussi di dati, consulta la pagina the section called “Streaming di note e restrizioni ETL”.

Configurazione di Kafka

Non sussistono prerequisiti AWS per la connessione ai flussi di Kafka disponibili su Internet.

È possibile creare una connessione AWS Glue Kafka per gestire le proprie credenziali di connessione. Per ulteriori informazioni, consulta the section called “Creazione di una connessione per un flusso di dati Kafka”. Nella configurazione del processo AWS Glue, fornisci *`connectionName`* come Connessione di rete aggiuntiva, quindi, nella chiamata al metodo, fornisci *`connectionName`* al parametro `connectionName`.

In alcuni casi, è necessario configurare ulteriori prerequisiti:

- Se utilizzi Streaming gestito da Amazon per Apache Kafka con l'autenticazione IAM, avrai bisogno di una configurazione appropriata di IAM.
- Se utilizzi Streaming gestito da Amazon per Apache Kafka con un Amazon VPC, avrai bisogno di una configurazione appropriata di Amazon VPC. Dovrai creare una connessione ad AWS Glue che fornisca informazioni sulla connessione ad Amazon VPC. È necessario che la configurazione del processo includa la connessione AWS Glue come connessione di rete aggiuntiva.

Per ulteriori informazioni sui prerequisiti dei processi ETL dei flussi di dati, consulta la pagina the section called “Aggiunta di processi di streaming ETL”.

Esempio: lettura di flussi da Kafka

Usato in combinazione con the section called “forEachBatch”.

Esempio per l'origine di streaming Kafka:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
```

```
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

Indicazioni di riferimento alle opzioni di connessione a Kafka

Utilizzare le seguenti opzioni di connessione con `"connectionType": "kafka"`:

- `"bootstrap.servers"` (Obbligatorio) un elenco di URL del server bootstrap, ad esempio, come `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Questa opzione deve essere specificata nella chiamata API o definita nei metadati della tabella in catalogo dati.
- `"security.protocol"` (Obbligatorio) Il protocollo utilizzato per comunicare con i broker. I valori possibili sono `"SSL"` o `"PLAINTEXT"`.
- `"topicName"`: (obbligatorio) un elenco separato da virgole di argomenti a cui iscriversi. Devi specificare solo uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.
- `"assign"`: (obbligatorio) una stringa JSON che specifica il `TopicPartitions` specifico da utilizzare. Devi specificare solo uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.

  Esempio: '{"topicA":[0,1],"topicB":[2,4]}'
- `"subscribePattern"`: (Obbligatorio) una stringa regex Java che identifichi l'elenco degli argomenti a cui effettuare la sottoscrizione. Devi specificare solo uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.

  Esempio: 'topic.*'
- `"classification"` (obbligatorio): il formato di file utilizzato dai dati nel record. Obbligatorio, a meno che non sia fornito tramite Catalogo dati.
- `"delimiter"` (facoltativo): il separatore di valori utilizzato quando `classification` è CSV. Il valore predefinito è `","`.
- `"startingOffsets"`: (Facoltativo) la posizione di partenza nell'argomento Kafka da cui leggere i dati. I valori possibili sono `"earliest"` o `"latest"`. Il valore predefinito è `"latest"`.
- `"startingTimestamp"`: (facoltativo, supportato solo per AWS Glue versione 4.0 o successiva) il timestamp del record nell'argomento Kafka da cui leggere i dati. Il valore possibile è una stringa timestamp in formato UTC nel modello `yyyy-mm-ddTHH:MM:SSZ`, dove `Z` rappresenta un offset del fuso orario UTC con un segno +/- (ad esempio: "2023-04-04T08:00:00-04:00").

Nota: nell'elenco delle opzioni di connessione dello script di flussi di dati di AWS Glue può essere presente solo un valore tra "startingOffsets" o "startingTimestamp"; l'inclusione di entrambe queste proprietà comporterà un errore del processo.

- `"endingOffsets"`: (Facoltativo) il punto di fine di una query batch. I valori possibili sono `"latest"` o una stringa JSON che specifica un offset finale per ogni `TopicPartition`.

  Per la stringa JSON, il formato è `{"topicA":{"0":23,"1":-1},"topicB":{"0":-1}}`. Il valore `-1` come offset rappresenta `"latest"`.
- `"pollTimeoutMs"`: (Facoltativo) il timeout in millisecondi per il polling dei dati da Kafka negli executor del processo Spark. Il valore predefinito è `512`.
- `"numRetries"`: (Facoltativo) i numero di tentativi prima di non riuscire a recuperare gli offset Kafka. Il valore predefinito è `3`.
- `"retryIntervalMs"`: (Facoltativo) il tempo di attesa in millisecondi prima di riprovare a recuperare gli offset Kafka. Il valore predefinito è `10`.
- `"maxOffsetsPerTrigger"`: (Facoltativo) il limite di velocità sul numero massimo di offset elaborati per intervallo di trigger. Il numero totale di offset specificato viene suddiviso proporzionalmente tra `topicPartitions` di diversi volumi. Il valore di default è null, il che significa che il consumer legge tutti gli offset fino all'ultimo offset noto.
- `"minPartitions"`: (Facoltativo) il numero minimo desiderato di partizioni da leggere da Kafka. Il valore di default è null, il che significa che il numero di partizioni Spark è uguale al numero di partizioni Kafka.
- `"includeHeaders"`: (Facoltativo) indica se includere le intestazioni Kafka. Quando l'opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "glue_streaming_kafka_headers" con tipo `Array[Struct(key: String, value: String)]`. Il valore di default è "false". Questa opzione è disponibile in AWS Glue versione 3.0 o successive.
- `"schema"`: (obbligatorio quando inferSchema è impostato su false) lo schema da utilizzare per elaborare il payload. Se la classificazione è `avro`, lo schema fornito dovrà essere nel formato dello schema Avro. Se la classificazione è `avro`, lo schema fornito dovrà essere nel formato dello schema DDL.

  Di seguito sono riportati alcuni esempi di schema.

  Example in DDL schema format

  ```
  'column1' INT, 'column2' STRING , 'column3' FLOAT
  ```

Example in Avro schema format

```
{
"type":"array",
"items":
{
"type":"record",
"name":"test",
"fields":
[
  {
    "name":"_id",
    "type":"string"
  },
  {
    "name":"index",
    "type":
    [
      "int",
      "string",
      "float"
    ]
  }
]
}
}
```

- `"inferSchema"`: (facoltativo) il valore di default è "false". Se impostato su "true", lo schema verrà rilevato in fase di esecuzione dal payload all'interno di `foreachbatch`.
- `"avroSchema"`: (obsoleto) parametro utilizzato per specificare uno schema di dati Avro quando viene utilizzato il formato Avro. Questo parametro è obsoleto. Utilizzo del parametro `schema`.
- `"addRecordTimestamp"`: (Facoltativo) Quando questa opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "__src_timestamp" che indica l'ora in cui il record corrispondente è stato ricevuto dall'argomento. Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.
- `"emitConsumerLagMetrics"`: (Facoltativo) Quando questa opzione è impostata su "true", per ogni batch, emetterà i parametri relativi alla durata tra il record più vecchio ricevuto dall'argomento e l'ora in cui arriva in AWS Glue a CloudWatch. Il nome del parametro è "glue.driver.streaming.maxConsumerLaginMs". Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.

Connessioni Azure Cosmos DB

È possibile utilizzare AWS Glue per Spark per leggere e scrivere su contanier esistenti in Azure Cosmos DB utilizzando l'API NoSQL in AWS Glue 4.0 e versioni successive. È possibile definire cosa leggere da Azure Cosmos DB con una query SQL. Connettiti ad Azure Cosmos DB usando una chiave di Azure Cosmos DB archiviata in AWS Secrets Manager tramite una connessione AWS Glue.

Per altre informazioni su Azure Cosmos DB per NoSQL, consulta la documentazione di Azure.

Configurazione delle connessioni Azure Cosmos DB

Per connetterti ad Azure Cosmos DB da AWS Glue, dovrai creare e archiviare la tua chiave Azure Cosmos DB in un segreto AWS Secrets Manager, quindi associare quel segreto a una connessione Azure Cosmos DB AWS Glue.

Prerequisiti:

- In Azure, dovrai identificare o generare una chiave di Azure Cosmos DB da usare da AWS Glue, `cosmosKey`. Per altre informazioni, consulta Accesso sicuro ai dati in Azure Cosmos DB nella documentazione di Azure.

Per configurare una connessione ad Azure Cosmos DB:

1. In AWS Secrets Manager, crea un segreto utilizzando la tua chiave Azure Cosmos DB. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `spark.cosmos.accountKey` con il valore *`cosmosKey`*.
2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *`connectionName`*, per l'uso futuro in AWS Glue.
   - In Tipo di connessione, seleziona Azure Cosmos DB.
   - Quando selezioni il Segreto AWS, fornisci *`secretName`*.

Dopo aver creato una connessione AWS Glue Azure Cosmos DB, è necessario eseguire le seguenti operazioni prima di eseguire il processo AWS Glue:

- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *secretName*.
- Nella configurazione del processo AWS Glue, fornisci *connectionName* come Connessione di rete aggiuntiva.

Lettura da container Azure Cosmos DB per NoSQL

Prerequisiti:

- Un container Azure Cosmos DB per NoSQL da cui desideri leggere. Avrai bisogno delle informazioni di identificazione per il container.

  Un container Azure Cosmos per NoSQL è identificato dal database e dal container. È necessario fornire i nomi del database, *cosmosDBName*, e del container, *cosmosContainerName*, quando ci si connette all'API di Azure Cosmos per NoSQL.
- Una connessione ad Azure Cosmos DB AWS Glue configurata per fornire informazioni di autenticazione e posizione della rete. Per l'acquisizione, completa i passaggi della procedura precedente, Per configurare una connessione ad Azure Cosmos DB. Sarà necessario il nome della connessione AWS Glue, *connectionName*.

Ad esempio:

```
azurecosmos_read = glueContext.create_dynamic_frame.from_options(
    connection_type="azurecosmos",
    connection_options={
    "connectionName": connectionName,
    "spark.cosmos.database": cosmosDBName,
    "spark.cosmos.container": cosmosContainerName,
    }
)
```

È possibile anche fornire una query SELECT SQL per filtrare i risultati restituiti al DynamicFrame. Sarà necessario configurare `query`.

Ad esempio:

```
azurecosmos_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="azurecosmos",
    connection_options={
        "connectionName": "connectionName",
```

```
        "spark.cosmos.database": cosmosDBName,
        "spark.cosmos.container": cosmosContainerName,
        "spark.cosmos.read.customQuery": "query"
    }
)
```

Scrittura su container Azure Cosmos DB per NoSQL

Questo esempio scrive informazioni a partire da un DynamicFrame esistente, *dynamicFrame* ad Azure Cosmos DB. Se il contanier contiene già informazioni, AWS Glue aggiungerà i dati da DynamicFrame. Se le informazioni nel contanier hanno uno schema diverso da quello scritto, si verificheranno degli errori.

Prerequisiti:

- Una tabella di Azure Cosmos DB su cui scrivere. Avrai bisogno delle informazioni di identificazione per il container. È necessario creare il container prima di chiamare il metodo di connessione.

  Un container Azure Cosmos per NoSQL è identificato dal database e dal container. È necessario fornire i nomi del database, *cosmosDBName*, e del container, *cosmosContainerName*, quando ci si connette all'API di Azure Cosmos per NoSQL.
- Una connessione ad Azure Cosmos DB AWS Glue configurata per fornire informazioni di autenticazione e posizione della rete. Per l'acquisizione, completa i passaggi della procedura precedente, Per configurare una connessione ad Azure Cosmos DB. Sarà necessario il nome della connessione AWS Glue, *connectionName*.

Ad esempio:

```
azurecosmos_write = glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="azurecosmos",
    connection_options={
    "connectionName": connectionName,
    "spark.cosmos.database": cosmosDBName,
    "spark.cosmos.container": cosmosContainerName
)
```

Indicazioni di riferimento alle opzioni di connessione ad Azure Cosmos DB

- `connectionName`: obbligatorio. Utilizzato per la lettura/scrittura. Il nome di una connessione ad Azure Cosmos DB AWS Glue configurata per fornire informazioni di autenticazione e posizione della rete al metodo di connessione.
- `spark.cosmos.database`: obbligatorio. Utilizzato per la lettura/scrittura. Valori validi: nomi di database. Nome del database di Azure Cosmos DB per NoSQL.
- `spark.cosmos.container`: obbligatorio. Utilizzato per la lettura/scrittura. Valori validi: nomi dei contanier. Nome del contanier di Azure Cosmos DB per NoSQL.
- `spark.cosmos.read.customQuery`: utilizzato per la lettura. Valori validi: query SELECT SQL. Query personalizzata per selezionare i documenti da leggere.

Connessioni Azure SQL

È possibile utilizzare AWS Glue per Spark per leggere da e scrivere su tabelle sulle le istanze gestite di Azure SQL in AWS Glue 4.0 e versioni successive. È possibile definire cosa leggere da Azure SQL con una query SQL. Connettiti ad Azure SQL utilizzando le credenziali utente e password memorizzate in AWS Secrets Manager tramite una connessione AWS Glue.

Per altre informazioni su Azure SQL, consulta la documentazione di Azure SQL.

Configurazione delle connessioni Azure SQL

Per connetterti ad Azure SQL da AWS Glue, dovrai creare e archiviare le tue credenziali Azure SQL in un segreto AWS Secrets Manager, quindi associare quel segreto a una connessione Azure SQL AWS Glue.

Per configurare una connessione ad Azure SQL:

1. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali Azure SQL. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `user` con il valore *`azuresqlUsername`*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *`azuresqlPassword`*.

2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per l'uso futuro in AWS Glue.

   - In Tipo di connessione, seleziona Azure SQL.
   - Quando fornisci l'URL SQL di Azure, fornisci un URL di endpoint JDBC.

     L'elenco deve essere nel seguente formato: `jdbc:sqlserver://`*databaseServerName*`:`*databasePort*`;databaseName=`*azuresqlDBnam*

     AWS Glue richiede le seguenti proprietà URL:

     - `databaseName` - Un database predefinito in Azure SQL a cui connettersi.

     Per altre informazioni sugli URL JDBC per le istanze gestite di Azure SQL, consulta la documentazione di Microsoft.
   - Quando selezioni il Segreto AWS, fornisci *secretName*.

Dopo aver creato una connessione AWS Glue Azure SQL, è necessario eseguire le seguenti operazioni prima di eseguire il processo AWS Glue:

- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *secretName*.
- Nella configurazione del processo AWS Glue, fornisci *connectionName* come Connessione di rete aggiuntiva.

Lettura da tabelle SQL di Azure

Prerequisiti:

- Una tabella Azure SQL da cui si desidera leggere. Avrai bisogno di informazioni di identificazione per la tabella, *databaseName* e *tableIdentifier*.

  Una tabella SQL di Azure è identificata dal database, dallo schema e dal nome. È necessario fornire il nome del database e della tabella durante la connessione ad Azure SQL. È inoltre necessario fornire lo schema se diverso da quello predefinito, "pubblico". Il database viene fornito tramite una proprietà URL in *connectionName*, il nome dello schema e della tabella tramite `dbtable`.
- Una connessione Azure SQL AWS Glue configurata per fornire informazioni di autenticazione. Completa i passaggi della procedura precedente, Per configurare una connessione ad Azure SQL

per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *connectionName*.

Ad esempio:

```
azuresql_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier"
    }
)
```

È possibile anche fornire una query SELECT SQL per filtrare i risultati restituiti al DynamicFrame. Sarà necessario configurare `query`.

Ad esempio:

```
azuresql_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Scrittura su tabelle SQL di Azure

Questo esempio scrive informazioni a partire da un DynamicFrame esistente, *dynamicFrame* ad Azure SQL. Se la tabella contiene già informazioni, AWS Glue aggiungerà i dati da DynamicFrame.

Prerequisiti:

- Una tabella di Azure SQL su cui scrivere. Avrai bisogno di informazioni di identificazione per la tabella, *databaseName* e *tableIdentifier*.

  Una tabella SQL di Azure è identificata dal database, dallo schema e dal nome. È necessario fornire il nome del database e della tabella durante la connessione ad Azure SQL. È inoltre necessario fornire lo schema se diverso da quello predefinito, "pubblico". Il database viene fornito tramite una proprietà URL in *connectionName*, il nome dello schema e della tabella tramite `dbtable`.

- Informazioni di autenticazione SQL di Azure. Completa i passaggi della procedura precedente, Per configurare una connessione ad Azure SQL per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *connectionName*.

Ad esempio:

```
azuresql_write = glueContext.write_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier"
    }
)
```

Indicazioni di riferimento alle opzioni di connessione ad Azure SQL

- `connectionName`: obbligatorio. Utilizzato per la lettura/scrittura. Il nome di una connessione ad Azure SQL AWS Glue configurata per fornire informazioni di autenticazione al metodo di connessione.
- `databaseName`: utilizzato per la lettura/scrittura. Valori validi: nomi di database di Azure SQL. Il nome del database in Azure SQL a cui connettersi.
- `dbtable` — Richiesto per la scrittura, richiesto per la lettura a meno che non `query` sia fornito. Utilizzato per la lettura/scrittura. Valori validi: nomi delle tabelle di Azure SQL o combinazioni di nomi di tabella/schema separate da punti. Utilizzato per specificare la tabella e lo schema che identificano la tabella a cui connettersi. Lo schema predefinito è "pubblico". Se la tabella rientra in uno schema non predefinito, fornisci queste informazioni nel modulo *schemaName*.*tableName*.
- `query`: utilizzato per la lettura. Una query Transact-SQL SELECT che definisce cosa recuperare durante la lettura da Azure SQL. Per ulteriori informazioni, consulta la documentazione di Microsoft.

Connessioni BigQuery

È possibile utilizzare AWS Glue per Spark per leggere e scrivere su tabelle in Google BigQuery in AWS Glue 4.0 e versioni successive. È possibile leggere da BigQuery con una query SQL di Google. Ci si connette a BigQuery utilizzando le credenziali archiviate in AWS Secrets Manager tramite una connessione AWS Glue.

Per ulteriori informazioni su Google BigQuery, consulta il sito Web di Google Cloud BigQuery.

Configurazione delle connessioni BigQuery

Per connetterti a Google BigQuery da AWS Glue, dovrai creare e archiviare le tue credenziali di Google Cloud Platform in un segreto AWS Secrets Manager, quindi associare tale segreto a una connessione AWS Glue con Google BigQuery.

Per configurare una connessione a BigQuery:

1. In Google Cloud Platform, crea e identifica le risorse pertinenti:
   - Crea o identifica un progetto GCP contenente tabelle BigQuery a cui desideri connetterti.
   - Abilita l'API BigQuery. Per ulteriori informazioni, consulta la pagina Utilizzo dell'API di lettura dell'archiviazione BigQuery per leggere i dati delle tabelle .
2. In Google Cloud Platform, crea ed esporta le credenziali dell'account del servizio:

   Puoi utilizzare la procedura guidata per le credenziali di BigQuery per accelerare questo passaggio: Crea credenziali.

   Per creare un account di servizio in GCP, segui il tutorial disponibile in Creazione di account di servizio.
   - Quando selezioni il progetto, seleziona il progetto contenente la tua tabella BigQuery.
   - Quando selezioni i ruoli IAM di GCP per il tuo account di servizio, aggiungi o crea un ruolo che conceda le autorizzazioni appropriate per eseguire processi BigQuery per leggere, scrivere o creare tabelle BigQuery.

   Per creare le credenziali per il tuo account di servizio, segui il tutorial disponibile in Creazione della chiave di un account di servizio.
   - Quando selezioni il tipo di chiave, seleziona JSON.

   Ora dovresti avere scaricato un file JSON con le credenziali per il tuo account di servizio. La schermata visualizzata dovrebbe risultare simile a quella nell'immagine seguente:

```
{
  "type": "service_account",
  "project_id": "*****",
  "private_key_id": "*****",
  "private_key": "*****",
```

```
  "client_email": "*****",
  "client_id": "*****",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "*****",
  "universe_domain": "googleapis.com"
}
```

3. base64 codifica il tuo file di credenziali scaricato. In una sessione AWS CloudShell o simile, puoi farlo dalla riga di comando eseguendo `cat` *`credentialsFile.json`* `| base64 -w 0`. Conserva l'output di questo comando, *`credentialString`*.
4. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali di Google Cloud Platform. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `credentials` con il valore *`credentialString`*.
5. In Catalogo dati AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *`connectionName`*, per il passaggio successivo.
   - Quando selezioni un tipo di connessione, seleziona Google BigQuery.
   - Quando selezioni il Segreto AWS, fornisci *`secretName`*.
6. Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *`secretName`*.
7. Nella configurazione del processo AWS Glue, fornisci *`connectionName`* come Connessione di rete aggiuntiva.

Lettura da tabelle BigQuery

Prerequisiti:

- Una tabella BigQuery da cui si desidera leggere. Avrai bisogno della tabella di BigQuery e dei nomi dei set di dati, nel modulo `[dataset].[table]`. Lo chiameremo *`tableName`*.
- Il progetto di fatturazione per la tabella BigQuery. Avrai bisogno del nome del progetto, *`parentProject`*. Se non esiste un progetto padre di fatturazione, utilizza il progetto contenente la tabella.

- Informazioni di autenticazione di BigQuery. Completa i passaggi descritti in Gestione delle credenziali di connessione con AWS Glue per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *connectionName*.

Ad esempio:

```
bigquery_read = glueContext.create_dynamic_frame.from_options(
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "sourceType": "table",
        "table": "tableName",
    }
```

È possibile anche fornire una query per filtrare i risultati restituiti al DynamicFrame. Sarà necessario configurare `query`, `sourceType`, `viewsEnabled` e `materializationDataset`.

Ad esempio:

Prerequisiti aggiuntivi:

Sarà necessario creare o identificare un set di dati BigQuery, *materializationDataset*, in cui BigQuery può scrivere viste materializzate per le tue query.

Sarà necessario concedere le autorizzazioni GCP IAM appropriate all'account di servizio per creare tabelle in *materializationDataset*.

```
glueContext.create_dynamic_frame.from_options(
            connection_type="bigquery",
            connection_options={
                "connectionName": "connectionName",
                "materializationDataset": materializationDataset,
                "parentProject": "parentProject",
                "viewsEnabled": "true",
                "sourceType": "query",
                "query": "select * from bqtest.test"
            }
        )
```

Scrittura su tabelle BigQuery

Questo esempio scrive direttamente nel servizio BigQuery. BigQuery supporta anche il metodo di scrittura "indiretto". Per ulteriori informazioni sulla configurazione di scritture indirette, consulta la pagina the section called “Utilizzo della scrittura indiretta con Google BigQuery”.

Prerequisiti:

- Una tabella BigQuery in cui si desidera scrivere. Avrai bisogno della tabella di BigQuery e dei nomi dei set di dati, nel modulo `[dataset].[table]`. È possibile anche fornire un nuovo nome di tabella che verrà creato automaticamente. Lo chiameremo *`tableName`*.
- Il progetto di fatturazione per la tabella BigQuery. Avrai bisogno del nome del progetto, *`parentProject`*. Se non esiste un progetto padre di fatturazione, utilizza il progetto contenente la tabella.
- Informazioni di autenticazione di BigQuery. Completa i passaggi descritti in Gestione delle credenziali di connessione con AWS Glue per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *`connectionName`*.

Ad esempio:

```
bigquery_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "writeMethod": "direct",
        "table": "tableName",
    }
)
```

Indicazioni di riferimento alle opzioni di connessione a BigQuery

- `project`: Predefinita: impostazione predefinita dell'account del servizio Google Cloud. Utilizzato per la lettura/scrittura. Il nome di un progetto Google Cloud associato alla tua tabella.
- `table`: (obbligatorio) utilizzato per la lettura/scrittura. Il nome della tabella BigQuery nel formato `[[project:]dataset.]`.

- `dataset`: obbligatorio quando non è definito tramite l'opzione `table`. Utilizzato per la lettura/scrittura. Il nome del set di dati che contiene la tabella BigQuery.
- `parentProject`: Predefinita: impostazione predefinita dell'account del servizio Google Cloud. Utilizzato per la lettura/scrittura. Il nome di un progetto Google Cloud associato al `project` utilizzato per la fatturazione.
- `sourceType`: utilizzato per la lettura. Richiesto durante la lettura. Valori validi: `table`, `query` indica ad AWS Glue se la lettura avverrà per tabella o per query.
- `materializationDataset`: utilizzato per la lettura. Valori validi: stringhe. Il nome di un set di dati BigQuery utilizzato per archiviare le materializzazioni per le viste.
- `viewsEnabled`: utilizzato per la lettura. Valore predefinito: false. Valori validi: vero, falso. Configura se BigQuery utilizzerà le viste.
- `query`: utilizzato per la lettura. Usato quando `viewsEnabled` è vero. Una query DQL di GoogleSQL.
- `temporaryGcsBucket`: utilizzato per la scrittura. Obbligatorio quando `writeMethod` è impostato sull'impostazione predefinita (`indirect`). Nome di un bucket di Google Cloud Storage utilizzato per archiviare una forma intermedia dei dati durante la scrittura su BigQuery.
- `writeMethod`: valore predefinito: `indirect`. Valori validi: `direct`, `indirect`. Utilizzato per la scrittura. Specifica il metodo utilizzato per scrivere i dati.
  - Se impostato su `direct`, il connettore scriverà utilizzando l'API BigQuery Storage Write.
  - Se impostato su `indirect`, il connettore scriverà su Google Cloud Storage, quindi effettuerà il trasferimento su BigQuery utilizzando un'operazione di caricamento. L'account del servizio Google Cloud avrà bisogno delle autorizzazioni GCS appropriate.

Utilizzo della scrittura indiretta con Google BigQuery

Questo esempio utilizza la scrittura indiretta, che scrive i dati su Google Cloud Storage e li copia su Google BigQuery.

Prerequisiti:

Sarà necessario un bucket temporaneo di Google Cloud Storage, *`temporaryBucket`*.

Il ruolo IAM GCP per l'account di servizio GCP di AWS Glue richiederà le autorizzazioni GCS appropriate per accedere a *`temporaryBucket`*.

Configurazione aggiuntiva:

Per configurare la scrittura indiretta con BigQuery:

1. Valuta the section called “Configurazione di BigQuery” e individua o scarica nuovamente il file JSON delle credenziali GCP. Identifica *secretName*, il segreto AWS Secrets Manager per la connessione AWS Glue con Google BigQuery utilizzata nel tuo processo.
2. Carica il file JSON delle credenziali in una posizione Amazon S3 adeguatamente sicura. Mantieni il percorso del file, *s3secretpath*, per i passaggi futuri.
3. Modifica *secretName*, aggiungendo la chiave `spark.hadoop.google.cloud.auth.service.account.json.keyfile`. Imposta il valore su *s3secretpath*.
4. Concedi al tuo processo AWS Glue le autorizzazioni IAM Amazon S3 per accedere a *s3secretpath*.

Ora puoi fornire la posizione temporanea del bucket GCS al tuo metodo di scrittura. Non è necessario fornire il `writeMethod`, poiché `indirect` in passato è stata l'impostazione predefinita.

```
bigquery_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "temporaryGcsBucket": "temporaryBucket",
        "table": "tableName",
    }
)
```

Connessioni JDBC

Alcuni tipi di database, tipicamente relazionali, supportano la connessione tramite lo standard JDBC. Per ulteriori informazioni su JDBC, consulta la documentazione dell'API JDBC. AWS Glue supporta nativamente la connessione a determinati database tramite i relativi connettori JDBC; le librerie JDBC sono fornite nei processi AWS Glue Spark. Quando ci si connette a questi tipi di database utilizzando le librerie do AWS Glue, si ha accesso a un set standard di opzioni.

I valori JDBC connectionType includono quanto segue:

- `"connectionType": "sqlserver"`: designa una connessione a un database Microsoft SQL Server.
- `"connectionType": "mysql"`: designa una connessione al database MySQL.
- `"connectionType": "oracle"`: designa una connessione a un database Oracle.
- `"connectionType": "postgresql"`: designa una connessione al database PostgreSQL.
- `"connectionType": "redshift"`: designa una connessione a un database Amazon Redshift. Per ulteriori informazioni, consulta the section called “Connessioni Redshift”.

Nella tabella seguente sono elencate le versioni dei driver JDBC supportate da AWS Glue.

Prodotto	Versioni del driver JDBC per Glue 4.0	Versioni del driver JDBC per Glue 3.0	Versioni del driver JDBC per Glue 0.9, 1.0, 2.0
Microsoft SQL Server	9,40	7.x	6.x
MySQL	8.0.23	8.0.23	5.1
Oracle Database	21,7	21,1	11.2
PostgreSQL	42,36	4,2,18	42.1.x
MongoDB	4.7.2	4.0.0	2.0.0
Amazon Redshift*	redshift-jdbc42-2.1.0.16	redshift-jdbc41-1.2.12.1017	redshift-jdbc41-1.2.12.1017

* Per il tipo di connessione Amazon Redshift, tutte le altre coppie nome/valore incluse nelle opzioni di connessione per una connessione JDBC, incluse le opzioni di formattazione, vengono trasmesse direttamente all'origine dati SparkSQL sottostante. Nei processi AWS Glue con Spark in AWS Glue 4.0 e versioni successive, il connettore nativo di AWS Glue per Amazon Redshift utilizza l'integrazione Amazon Redshift per Apache Spark. Per ulteriori informazioni, consulta la pagina Integrazione di Amazon Redshift per Apache Spark. Nelle versioni precedenti, consulta la sezione Amazon Redshift data source for Spark.

Per configurare Amazon VPC per la connessione ai datastore Amazon RDS tramite JDBC, consulta la pagina the section called “Configurazione di Amazon VPC per la connessione agli archivi dati Amazon RDS”.

> **Note**
>
> I processi AWS Glue sono associati solo a una sottorete durante un'esecuzione. Ciò potrebbe influire sulla capacità di connettersi a più origini dati tramite lo stesso processo. Questo comportamento non è limitato alle origini JDBC.

Argomenti

- Indicazioni di riferimento alle opzioni di connessione a JDBC
- Utilizzo di sampleQuery
- Utilizzo di un driver JDBC personalizzato
- Lettura in parallelo dalle tabelle JDBC
- Configurazione di Amazon VPC per connessioni JDBC agli archivi dati Amazon RDS da AWS Glue

Indicazioni di riferimento alle opzioni di connessione a JDBC

Se hai già definito una connessione JDBC AWS Glue, puoi riutilizzare le proprietà di configurazione in essa definite, come url, utente e password; pertanto, non è necessario specificarle nel codice come opzioni di connessione. Questa funzionalità è disponibile solo in AWS Glue 3.0 e versioni successive. A tale scopo, utilizza le seguenti proprietà di connessione:

- `"useConnectionProperties"`: impostalo su "true" per indicare che desideri utilizzare la configurazione da una connessione.
- `"connectionName"`: inserisci il nome della connessione da cui recuperare la configurazione; la connessione deve essere definita nella stessa regione del processo.

Utilizzare queste opzioni di connessione con connessioni JDBC:

- `"url"`: (obbligatorio) l'URL JDBC per il database.
- `"dbtable"`: (obbligatorio) la tabella database da cui leggere. Per i archivi dati JDBC che supportano schemi all'interno di un database, specifica `schema.table-name`. Se non viene fornito alcuno schema, viene usato lo schema "pubblico" predefinito.

- `"user"`: (obbligatorio) Il nome utente da usare per la connessione.
- `"password"`: (obbligatorio) la password da usare per la connessione.
- (Facoltativo) Le seguenti opzioni consentono di fornire un driver JDBC personalizzato. Utilizzare queste opzioni se è necessario utilizzare un driver che AWS Glue non supporta in modo nativo.

  I processi ETL possono utilizzare versioni di driver JDBC diverse per l'origine dati e la destinazione, anche se l'origine e la destinazione sono lo stesso prodotto di database. In questo modo è possibile eseguire la migrazione dei dati tra database di origine e di destinazione con versioni diverse. Per utilizzare queste opzioni, è necessario innanzitutto caricare il file JAR del driver JDBC su Amazon S3.

  - `"customJdbcDriverS3Path"`: il percorso Amazon S3 del driver JDBC personalizzato.
  - `"customJdbcDriverClassName"`: il nome della classe del driver JDBC.
- `"bulkSize"`: (Facoltativo) Utilizzato per configurare inserti paralleli per accelerare i carichi di massa nelle destinazioni JDBC. Specifica un valore intero per il grado di parallelismo da utilizzare durante la scrittura o l'inserimento di dati. Questa opzione è utile per migliorare le prestazioni delle scritture in database come Arch User Repository (AUR).
- `"hashfield"`: (facoltativo) una stringa utilizzata per specificare il nome di una colonna nella tabella JDBC da utilizzare per dividere i dati in partizioni durante la lettura da tabelle JDBC in parallelo. Fornisci "hashfield" O "hashexpression". Per ulteriori informazioni, consulta the section called “Lettura in parallelo da JDBC”.
- `"hashexpression"`: (facoltativo) una clausola SQL select che restituisce un numero intero. Utilizzata per suddividere i dati presenti in una tabella JDBC in partizioni durante la lettura da tabelle JDBC in parallelo. Fornisci "hashfield" O "hashexpression". Per ulteriori informazioni, consulta the section called “Lettura in parallelo da JDBC”.
- `"hashpartitions"`: (facoltativo) un numero intero positivo. Utilizzato per specificare il numero di letture parallele della tabella JDBC durante la lettura da tabelle JDBC in parallelo. Impostazione predefinita: 7. Per ulteriori informazioni, consulta the section called “Lettura in parallelo da JDBC”.
- `"sampleQuery"`: (facoltativo) un'istruzione di query SQL personalizzata. Utilizzata per specificare un sottoinsieme di informazioni in una tabella per recuperare un campione del contenuto della tabella. Se configurato indipendentemente dai dati, può essere meno efficiente dei metodi DynamicFrame, causando timeout o errori di esaurimento della memoria. Per ulteriori informazioni, consulta the section called “Utilizzo di sampleQuery”.
- `"enablePartitioningForSampleQuery"`: (facoltativo) un valore booleano. Valore predefinito: false. Utilizzato per abilitare la lettura da tabelle JDBC in parallelo durante la specificazione della `sampleQuery`. Se impostato su true, **`sampleQuery`** deve terminare con "where" o "and" affinché

AWS Glue aggiunga le condizioni di partizionamento. Per ulteriori informazioni, consulta the section called “Utilizzo di sampleQuery”.

- `"sampleSize"`: (facoltativo) un numero intero positivo. Limita il numero di righe restituite dalla query di esempio. Funziona solo quando `enablePartitioningForSampleQuery` è true. Se il partizionamento non è abilitato, è invece necessario aggiungere direttamente `"limit x"` nella `sampleQuery` per limitare le dimensioni. Per ulteriori informazioni, consulta the section called “Utilizzo di sampleQuery”.

Utilizzo di sampleQuery

Questa sezione illustra come utilizzare `sampleQuery`, `sampleSize` e `enablePartitioningForSampleQuery`.

`sampleQuery` può essere un modo efficace per campionare alcune righe del set di dati. Per impostazione predefinita, la query viene eseguita da un singolo esecutore. Se configurata indipendentemente dai dati, può essere meno efficiente dei metodi DynamicFrame, causando timeout o errori di esaurimento della memoria. Nelle pipeline ETL, l'esecuzione diretta di query SQL sul database sottostante è in genere necessaria solo per ottimizzare le prestazioni. Se stai cercando di visualizzare in anteprima alcune righe del tuo set di dati, prendi in considerazione l'utilizzo di the section called “show”. Se stai cercando di trasformare il tuo set di dati utilizzando SQL, prendi in considerazione l'utilizzo di the section called “toDF” per definire una trasformazione SparkSQL relativamente ai tuoi dati in un modulo DataFrame.

Anche se la query può manipolare diverse tabelle, `dbtable` resta obbligatoria.

Utilizzo di SampleQuery per recuperare un campione della tabella

Quando si utilizza il comportamento sampleQuery predefinito per recuperare un campione dei dati, AWS Glue non prevede una velocità di trasmissione effettiva sostanziale, quindi esegue la query su un singolo esecutore. Per limitare i dati forniti e non causare problemi di prestazioni, consigliamo di fornire a SQL una clausola `LIMIT`.

Example Usare SampleQuery senza partizionamento

Il codice di esempio seguente mostra come utilizzare `sampleQuery` senza partizionamento.

```
//A full sql query statement.
val query = "select name from $tableName where age > 0 limit 1"
val connectionOptions = JsonOptions(Map(
    "url" -> url,
```

```
    "dbtable" -> tableName,
    "user" -> user,
    "password" -> password,
    "sampleQuery" -> query ))
val dyf = glueContext.getSource("mysql", connectionOptions)
          .getDynamicFrame()
```

Utilizzo di SampleQuery con set di dati più grandi

Se stai leggendo un set di dati di grandi dimensioni, potrebbe essere necessario abilitare il partizionamento JDBC per interrogare una tabella in parallelo. Per ulteriori informazioni, consulta the section called “Lettura in parallelo da JDBC”. Per utilizzare `sampleQuery` con partizionamento JDBC, imposta `enablePartitioningForSampleQuery` su true. L'attivazione di questa funzionalità richiede di apportare alcune modifiche alla `sampleQuery`.

Quando si utilizza il partizionamento JDBC con `sampleQuery`, la query deve terminare con "where" o "and" affinché AWS Glue aggiunga le condizioni di partizionamento.

Se desideri limitare i risultati di SampleQuery durante la lettura da tabelle JDBC in parallelo, imposta il parametro `"sampleSize"` anziché specificare una clausola `LIMIT`.

Example Usare SampleQuery con partizionamento JDBC

Il codice di esempio seguente mostra come utilizzare `sampleQuery` con partizionamento JDBC.

```
//note that the query should end with "where" or "and" if use with JDBC partitioning.
val query = "select name from $tableName where age > 0 and"

//Enable JDBC partitioning by setting hashfield.
//to use sampleQuery with partitioning, set enablePartitioningForSampleQuery.
//use sampleSize to limit the size of returned data.
val connectionOptions = JsonOptions(Map(
    "url" -> url,
    "dbtable" -> tableName,
    "user" -> user,
    "password" -> password,
    "hashfield" -> primaryKey,
    "sampleQuery" -> query,
    "enablePartitioningForSampleQuery" -> true,
    "sampleSize" -> "1" ))
val dyf = glueContext.getSource("mysql", connectionOptions)
          .getDynamicFrame()
```

Note e restrizioni:

Le query di esempio non possono essere utilizzate insieme ai segnalibri del processo. Lo stato del segnalibro verrà ignorato quando viene fornita la configurazione per entrambi.

Utilizzo di un driver JDBC personalizzato

Negli esempi di codice riportati di seguito viene illustrato come leggere e scrivere nei database JDBC con driver JDBC personalizzati. Essi dimostrano la lettura da una versione di un prodotto di database e la scrittura a una versione successiva dello stesso prodotto.

Python

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext, SparkConf
from awsglue.context import GlueContext
from awsglue.job import Job
import time
from pyspark.sql.types import StructType, StructField, IntegerType, StringType

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

# Construct JDBC connection options
connection_mysql5_options = {
    "url": "jdbc:mysql://<jdbc-host-name>:3306/db",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd"}

connection_mysql8_options = {
    "url": "jdbc:mysql://<jdbc-host-name>:3306/db",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd",
    "customJdbcDriverS3Path": "s3://DOC-EXAMPLE-BUCKET/mysql-connector-
java-8.0.17.jar",
    "customJdbcDriverClassName": "com.mysql.cj.jdbc.Driver"}

connection_oracle11_options = {
```

```
    "url": "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd"}

connection_oracle18_options = {
    "url": "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd",
    "customJdbcDriverS3Path": "s3://DOC-EXAMPLE-BUCKET/ojdbc10.jar",
    "customJdbcDriverClassName": "oracle.jdbc.OracleDriver"}

# Read from JDBC databases with custom driver
df_mysql8 = glueContext.create_dynamic_frame.from_options(connection_type="mysql",

 connection_options=connection_mysql8_options)

# Read DynamicFrame from MySQL 5 and write to MySQL 8
df_mysql5 = glueContext.create_dynamic_frame.from_options(connection_type="mysql",

 connection_options=connection_mysql5_options)
glueContext.write_from_options(frame_or_dfc=df_mysql5, connection_type="mysql",
                               connection_options=connection_mysql8_options)

# Read DynamicFrame from Oracle 11 and write to Oracle 18
df_oracle11 =
 glueContext.create_dynamic_frame.from_options(connection_type="oracle",

 connection_options=connection_oracle11_options)
glueContext.write_from_options(frame_or_dfc=df_oracle11, connection_type="oracle",
                               connection_options=connection_oracle18_options)
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
```

```
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {
  val MYSQL_5_URI: String = "jdbc:mysql://<jdbc-host-name>:3306/db"
  val MYSQL_8_URI: String = "jdbc:mysql://<jdbc-host-name>:3306/db"
  val ORACLE_11_URI: String = "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL"
  val ORACLE_18_URI: String = "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL"

  // Construct JDBC connection options
  lazy val mysql5JsonOption = jsonOptions(MYSQL_5_URI)
  lazy val mysql8JsonOption = customJDBCDriverJsonOptions(MYSQL_8_URI, "s3://DOC-
EXAMPLE-BUCKET/mysql-connector-java-8.0.17.jar", "com.mysql.cj.jdbc.Driver")
  lazy val oracle11JsonOption = jsonOptions(ORACLE_11_URI)
  lazy val oracle18JsonOption = customJDBCDriverJsonOptions(ORACLE_18_URI, "s3://
DOC-EXAMPLE-BUCKET/ojdbc10.jar", "oracle.jdbc.OracleDriver")

  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Read from JDBC database with custom driver
    val df_mysql8: DynamicFrame = glueContext.getSource("mysql",
 mysql8JsonOption).getDynamicFrame()

    // Read DynamicFrame from MySQL 5 and write to MySQL 8
    val df_mysql5: DynamicFrame = glueContext.getSource("mysql",
 mysql5JsonOption).getDynamicFrame()
    glueContext.getSink("mysql", mysql8JsonOption).writeDynamicFrame(df_mysql5)

    // Read DynamicFrame from Oracle 11 and write to Oracle 18
    val df_oracle11: DynamicFrame = glueContext.getSource("oracle",
 oracle11JsonOption).getDynamicFrame()
    glueContext.getSink("oracle", oracle18JsonOption).writeDynamicFrame(df_oracle11)

    Job.commit()
  }

  private def jsonOptions(url: String): JsonOptions = {
    new JsonOptions(
      s"""{"url": "${url}",
```

```
          |"dbtable":"test",
          |"user": "admin",
          |"password": "pwd"}""".stripMargin)
  }

  private def customJDBCDriverJsonOptions(url: String, customJdbcDriverS3Path:
 String, customJdbcDriverClassName: String): JsonOptions = {
    new JsonOptions(
      s"""{"url": "${url}",
          |"dbtable":"test",
          |"user": "admin",
          |"password": "pwd",
          |"customJdbcDriverS3Path": "${customJdbcDriverS3Path}",
          |"customJdbcDriverClassName" :
 "${customJdbcDriverClassName}"}""".stripMargin)
  }
}
```

Lettura in parallelo dalle tabelle JDBC

Puoi impostare le proprietà della tabella JDBC per permettere a AWS Glue di leggere i dati in parallelo. Quando imposti determinate proprietà, indichi a AWS Glue di eseguire query SQL parallele su partizioni logiche dei dati. Puoi controllare il partizionamento impostando un campo hash o un'espressione hash. Puoi anche controllare il numero di operazioni di lettura parallele usate per accedere ai dati.

La lettura delle tabelle JDBC in parallelo è una tecnica di ottimizzazione che può migliorare le prestazioni. Per ulteriori informazioni sul processo di identificazione del momento in cui questa tecnica è appropriata, consulta Ridurre la quantità di scansione dei dati nella guida Best practice per l'ottimizzazione delle prestazioni dei processi AWS Glue per Apache Spark nel Prontuario AWS.

Per abilitare le letture parallele, puoi impostare coppie chiave/valore nel campo dei parametri della struttura della tabella. Utilizza la notazione JSON per impostare un valore per il campo parametri della tabella. Per ulteriori informazioni sulle modifiche delle proprietà di una tabella, consulta Visualizzazione e modifica dei dettagli della tabella. Puoi anche abilitare le letture parallele chiamando i metodi ETL (Extract, Transform and Load, estrazione, trasformazione e caricamento) `create_dynamic_frame_from_options` e `create_dynamic_frame_from_catalog`. Per ulteriori informazioni sulla definizione delle opzioni in questi metodi, consulta from_options e from_catalog.

Puoi usare questo metodo per le tabelle JDBC, ovvero la maggior parte delle tabelle i cui dati dai base costituiscono un datastore JDBC. Queste proprietà vengono ignorate quando viene eseguita la lettura delle tabelle Amazon Redshift e Amazon S3.

hashfield

Imposta `hashfield` sul nome di una colonna nella tabella JDBC da usare per dividere i dati in partizioni. Per ottenere risultati ottimali, questa colonna deve avere una distribuzione uniforme dei valori per distribuire i dati tra le partizioni. Questa colonna può contenere qualsiasi tipo di dati. AWS Glue genera query non sovrapposte eseguite in parallelo per leggere i dati partizionati in base a questa colonna. Ad esempio, se i dati sono distribuiti in modo uniforme in base al mese, è possibile usare la colonna `month` per leggere ogni mese di dati in parallelo.

```
'hashfield': 'month'
```

AWS Glue crea una query per l'hashing del valore di campo a un numero di partizione ed esegue la query per tutte le partizioni in parallelo. Per usare la query personalizzata per partizionare la lettura di una tabella, fornisci un oggetto `hashexpression` al posto di un oggetto `hashfield`.

hashexpression

Imposta `hashexpression` su un'espressione SQL (conforme alla grammatica del motore di database JDBC) che restituisce un numero intero. Un'espressione semplice è il nome di qualsiasi colonna numerica nella tabella. AWS Glue genera query SQL per leggere i dati JDBC in parallelo usando `hashexpression` nella clausola `WHERE` per partizionare i dati.

Ad esempio, è possibile usare la colonna numerica `customerID` per leggere i dati partizionati in base a un numero cliente.

```
'hashexpression': 'customerID'
```

Per fare in modo che AWS Glue controlli il partizionamento, fornisci un oggetto `hashfield` al posto di un oggetto `hashexpression`.

hashpartitions

Imposta `hashpartitions` sul numero di letture parallele della tabella JDBC. Se questa proprietà non viene impostata, il valore predefinito è 7.

Imposta, ad esempio, il numero di letture parallele su `5`, in modo che AWS Glue legga i dati con cinque query (o meno).

```
'hashpartitions': '5'
```

Configurazione di Amazon VPC per connessioni JDBC agli archivi dati Amazon RDS da AWS Glue

> **Note**
>
> Per impostazione predefinita, le nuove istanze di database Amazon RDS utilizzeranno il nuovo certificato. `rds-ca-rsa2048-g1` AWS Gluejob e Test Connection si basano attualmente su. `certificate rds-ca-2019` Per connettere nuove istanze Amazon RDS con AWS Glue job o Test Connection, imposta l'istanza per utilizzare il certificato `rds-ca-2019` tramite la AWS console o. AWS CLI Per ulteriori informazioni, consulta la sezione Utilizzo di SSL/TLS per crittografare una connessione a un'istanza DB nella guida per l'utente di Amazon RDS per una guida dettagliata.

Quando usi JDBC per connetterti ai database in Amazon RDS, dovrai eseguire una configurazione aggiuntiva. Per consentire ai AWS Glue componenti di comunicare con Amazon RDS, devi configurare l'accesso ai tuoi archivi dati Amazon RDS in Amazon VPC. Per permettere la comunicazione dei componenti di AWS Glue, specifica un gruppo di sicurezza con una regola per il traffico in entrata autoreferenziale per tutte le porte TCP. Creando una regola di autoreferenziazione, puoi limitare l'origine allo stesso gruppo di sicurezza nel VPC. Una regola di autoreferenziazione non aprirà il VPC a tutte le reti. Il gruppo di sicurezza predefinito per il tuo VPC potrebbe già avere una regola autoreferenziata in entrata per ALL Traffic.

Per configurare l'accesso tra AWS Glue e gli archivi dati Amazon RDS

1. Accedi alla AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nella console Amazon RDS, identifica i gruppi di sicurezza utilizzati per controllare l'accesso al tuo database Amazon RDS.

   Nel riquadro di navigazione a sinistra, scegli Database, quindi seleziona l'istanza a cui desideri connetterti dall'elenco nel riquadro principale.

   Nella pagina dei dettagli del database, trova i gruppi di sicurezza VPC nella scheda Connettività e sicurezza.

3. In base all'architettura di rete, identificate quale gruppo di sicurezza associato è meglio modificare per consentire l'accesso al servizio AWS Glue. Salva il suo nome, *database-security-group*per riferimenti futuri. Se non esiste un gruppo di sicurezza appropriato, segui le istruzioni per Fornire l'accesso alla tua istanza DB nel tuo VPC creando un gruppo di sicurezza nella documentazione di Amazon RDS.

4. Accedere ad AWS Management Console e aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

5. Nella console Amazon VPC, identifica come eseguire l'aggiornamento. *database-security-group*

   Nel riquadro di navigazione a sinistra, scegli Gruppi di sicurezza, quindi seleziona *database-security-group*dall'elenco nel riquadro principale.

6. Identifica l'ID del gruppo di sicurezza per *database-security-group*, *database-sg-id*. Salvalo per future consultazioni.

   Nella pagina dei dettagli del gruppo di sicurezza, trova l'ID del gruppo di sicurezza.

7. Modifica le regole in entrata per *database-security-group*, aggiungi una regola di autoreferenziazione per consentire ai AWS Glue componenti di comunicare. In particolare, aggiungi o conferma l'esistenza di una regola in cui Type è**All TCP**, Protocol è**TCP**, Port Range include tutte le porte e Source è. ***database-sg-id*** Verifica che il gruppo di sicurezza che hai inserito per Source sia lo stesso del gruppo di sicurezza che stai modificando.

   Nella pagina dei dettagli del gruppo di sicurezza, seleziona Modifica regole in entrata.

   La regola in entrata è simile alla seguente:

Type	Protocollo	Intervallo porte	Origine
Tutte le regole TCP	TCP	0–65535	*database-sg-id*

8. Aggiungi regole per il traffico in uscita.

   Nella pagina dei dettagli del gruppo di sicurezza, seleziona Modifica regole in uscita.

   Se il gruppo di sicurezza consente tutto il traffico in uscita, non sono necessarie regole separate. Per esempio:

Type	Protocollo	Intervallo porte	Destinazione
All Traffic	ALL	ALL	0.0.0.0/0

   Se la tua architettura di rete è progettata per limitare il traffico in uscita, crea le seguenti regole in uscita:

   Crea una regola autoreferenziale in cui Type è**`All TCP`**, Protocol è**`TCP`**, Port Range include tutte le porte e Destinazione è. ***`database-sg-id`*** Verifica che il gruppo di sicurezza che hai inserito per Destinazione sia lo stesso del gruppo di sicurezza che stai modificando.

   Se utilizzi un endpoint VPC Amazon S3, aggiungi una regola HTTPS per consentire il traffico dal VPC ad Amazon S3. *`Crea una regola in cui` **`Type`** `èHTTPS,` **`Protocol`** `isTCP,` **`Port Range`** `è 443 e` **`Destination`** `è l'ID dell'elenco di prefissi gestiti per l'endpoint gateway Amazon S3, s3-. prefix-list-id`* Per ulteriori informazioni sugli elenchi di prefissi e sugli endpoint gateway Amazon S3, consulta Endpoint gateway per Amazon S3 nella documentazione di Amazon VPC.

   Per esempio:

Type	Protocollo	Intervallo porte	Destinazione
Tutte le regole TCP	TCP	0–65535	*`database-sg-id`*
HTTPS	TCP	443	*`s3- prefix-list-id`*

Connessioni MongoDB

È possibile utilizzare AWS Glue per Spark per leggere e scrivere su tabelle in MongoDB w MongoDB Atlas in AWS Glue 4.0 e versioni successive. È possibile connettersi a MongoDB utilizzando le credenziali di nome utente e password memorizzate in AWS Secrets Manager tramite una connessione AWS Glue.

Per ulteriori informazioni su MongoDB, consulta la documentazione di MongoDB.

Configurazione delle connessioni MongoDB

Per connetterti a MongoDB da AWS Glue, avrai bisogno delle tue credenziali MongoDB, *`mongodbUser`* e *`mongodbPass`*.

Per connettersi a MongoDB da AWS Glue, potrebbero essere necessari alcuni prerequisiti:

- Se la tua istanza MongoDB si trova in un Amazon VPC, configura Amazon VPC per consentire al processo AWS Glue di comunicare con l'istanza MongoDB senza che il traffico attraversi la rete Internet pubblica.

  In Amazon VPC, identifica o crea un VPC, una sottorete e un gruppo di sicurezza che AWS Glue utilizzerà durante l'esecuzione del processo. Inoltre, assicurati che Amazon VPC sia configurato per consentire il traffico di rete tra l'istanza MongoDB e questa posizione. In base al layout di rete, potrebbe richiedere modifiche alle regole dei gruppi di sicurezza, alle liste di controllo accessi di rete, ai gateway NAT e alle connessioni peering.

È quindi possibile procedere alla configurazione di AWS Glue per l'uso con MongoDB.

Per configurare una connessione a MongoDB:

1. Facoltativamente, in AWS Secrets Manager, crea un segreto utilizzando le tue credenziali MongoDB. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `username` con il valore *`mongodbUser`*.

     Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *`mongodbPass`*.

2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per l'uso futuro in AWS Glue.

   - Quando selezioni un tipo di connessione, seleziona MongoDB o MongoDB Atlas.
   - Quando selezioni l'URL MongoDB o URL MongoDB Atlas, fornisci il nome host dell'istanza MongoDB.

     Un URL MongoDB viene fornito nel formato `mongodb://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Un URL MongoDB Atlas viene fornito nel formato `mongodb+srv://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Fornendo il database predefinito per la connessione, *mongoDBname* è facoltativo.
   - Se hai scelto di creare un segreto di Secrets Manager, scegli il tipo di credenziali AWS Secrets Manager.

     Quindi, in Segreto di AWS, fornisci *secretName*.
   - Se scegli di fornire nome utente e password, fornisci *mongodbUser* e *mongodbPass*.
3. Nelle seguenti situazioni, potresti aver bisogno di una configurazione aggiuntiva:

   - Per le istanze MongoDB ospitate su AWS in un Amazon VPC
     - Dovrai fornire le informazioni di connessione Amazon VPC alla connessione AWS Glue che definisce le credenziali di sicurezza MongoDB. Durante la creazione o l'aggiornamento della connessione, imposta VPC, sottorete e Gruppi di sicurezza nelle opzioni di rete.

Dopo aver creato una connessione MongoDB AWS Glue, dovrai eseguire i seguenti passaggi prima di chiamare il metodo di connessione.

- Se hai scelto di creare un segreto di Secrets Manager, concedi al ruolo IAM associato al processo AWS Glue il permesso di leggere *secretName*.
- Nella configurazione del processo AWS Glue, fornisci *connectionName* come Connessione di rete aggiuntiva.

Per utilizzare la connessione MongoDB AWS Glue in AWS Glue per Spark, fornisci l'opzione `connectionName` nella chiamata al metodo di connessione. In alternativa, puoi seguire i passaggi in

the section called “Integrazione con MongoDB” per utilizzare la connessione insieme al Catalogo dati AWS Glue.

Lettura da MongoDB utilizzando una connessione AWS Glue

Prerequisiti:

- Una raccolta MongoDB da cui desideri leggere. Avrai bisogno delle informazioni di identificazione per la raccolta.

  Una raccolta MongoDB è identificata da un nome di database e di raccolta, *mongodbName*, *mongodbCollection*.
- Una connessione MongoDB AWS Glue configurata per fornire informazioni di autenticazione. Completa i passaggi della procedura precedente, Per configurare una connessione a MongoDB per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *connectionName*.

Ad esempio:

```
mongodb_read = glueContext.create_dynamic_frame.from_options(
    connection_type="mongodb",
    connection_options={
        "connectionName": "connectionName",
        "database": "mongodbName",
        "collection": "mongodbCollection",
        "partitioner":
 "com.mongodb.spark.sql.connector.read.partitioner.SinglePartitionPartitioner",
        "partitionerOptions.partitionSizeMB": "10",
        "partitionerOptions.partitionKey": "_id",
        "disableUpdateUri": "false",
    }
)
```

Scrittura su tabelle MongoDB

Questo esempio scrive informazioni a partire da un DynamicFrame esistente, *dynamicFrame* a MongoDB.

Prerequisiti:

- Una raccolta MongoDB su cui desideri scrivere. Avrai bisogno delle informazioni di identificazione per la raccolta.

  Una raccolta MongoDB è identificata da un nome di database e di raccolta, *mongodbName*, *mongodbCollection*.
- Una connessione MongoDB AWS Glue configurata per fornire informazioni di autenticazione. Completa i passaggi della procedura precedente, Per configurare una connessione a MongoDB per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *connectionName*.

Ad esempio:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="mongodb",
    connection_options={
        "connectionName": "connectionName",
        "database": "mongodbName",
        "collection": "mongodbCollection",
        "disableUpdateUri": "false",
        "retryWrites": "false",
    },
)
```

Lettura e scrittura in tabelle su tabelle MongoDB

Questo esempio scrive informazioni a partire da un DynamicFrame esistente, *dynamicFrame* a MongoDB.

Prerequisiti:

- Una raccolta MongoDB da cui desideri leggere. Avrai bisogno delle informazioni di identificazione per la raccolta.

  Una raccolta MongoDB su cui desideri scrivere. Avrai bisogno delle informazioni di identificazione per la raccolta.

  Una raccolta MongoDB è identificata da un nome di database e di raccolta, *mongodbName*, *mongodbCollection*.
- Informazioni di autenticazione MongoDB, *mongodbUser* e *mongodbPassword*.

Ad esempio:

Python

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext, SparkConf
from awsglue.context import GlueContext
from awsglue.job import Job
import time

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'], args)

output_path = "s3://some_bucket/output/" + str(time.time()) + "/"
mongo_uri = "mongodb://<mongo-instanced-ip-address>:27017"
mongo_ssl_uri = "mongodb://<mongo-instanced-ip-address>:27017"
write_uri = "mongodb://<mongo-instanced-ip-address>:27017"

read_mongo_options = {
    "uri": mongo_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "username": "mongodbUsername",
    "password": "mongodbPassword",
    "partitioner": "MongoSamplePartitioner",
    "partitionerOptions.partitionSizeMB": "10",
    "partitionerOptions.partitionKey": "_id"}

ssl_mongo_options = {
    "uri": mongo_ssl_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "ssl": "true",
    "ssl.domain_match": "false"
}
```

```
write_mongo_options = {
    "uri": write_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "username": "mongodbUsername",
    "password": "mongodbPassword",
}

# Get DynamicFrame from MongoDB
dynamic_frame =
 glueContext.create_dynamic_frame.from_options(connection_type="mongodb",

 connection_options=read_mongo_options)

# Write DynamicFrame to MongoDB
glueContext.write_dynamic_frame.from_options(dynamicFrame,
 connection_type="mongodb", connection_options=write_mongo_options)

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  val DEFAULT_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  val WRITE_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  lazy val defaultJsonOption = jsonOptions(DEFAULT_URI)
  lazy val writeJsonOption = jsonOptions(WRITE_URI)
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
```

```
    // Get DynamicFrame from MongoDB
    val dynamicFrame: DynamicFrame = glueContext.getSource("mongodb",
 defaultJsonOption).getDynamicFrame()

    // Write DynamicFrame to MongoDB
    glueContext.getSink("mongodb", writeJsonOption).writeDynamicFrame(dynamicFrame)

    Job.commit()
  }

  private def jsonOptions(uri: String): JsonOptions = {
    new JsonOptions(
      s"""{"uri": "${uri}",
         |"database":"mongodbName",
         |"collection":"mongodbCollection",
         |"username": "mongodbUsername",
         |"password": "mongodbPassword",
         |"ssl":"true",
         |"ssl.domain_match":"false",
         |"partitioner": "MongoSamplePartitioner",
         |"partitionerOptions.partitionSizeMB": "10",
         |"partitionerOptions.partitionKey": "_id"}""".stripMargin)
  }
}
```

Indicazioni di riferimento alle opzioni di connessione a MongoDB

Designa una connessione a MongoDB. Le opzioni di connessione differiscono per una connessione sorgente e una connessione sink.

Queste proprietà di connessione sono condivise tra le connessioni di origine e sink:

- `connectionName`: utilizzato per la lettura/scrittura. Il nome di una connessione a MongoDB AWS Glue configurata per fornire informazioni di autenticazione e sulla rete al metodo di connessione. Quando una connessione AWS Glue è configurata come descritto nella sezione precedente, the section called “Configurazione di MongoDB”, la fornitura di `connectionName` sostituirà la necessità di fornire le opzioni di connessione `"uri"`, `"username"` e `"password"`.
- `"uri"`: (Obbligatorio) L'host MongoDB da cui leggere, formattato come `mongodb://<host>:<port>`. Utilizzato nelle versioni AWS Glue precedenti alla versione di AWS Glue 4.0.

- "connection.uri": (Obbligatorio) L'host MongoDB da cui leggere, formattato come mongodb://<host>:<port>. Utilizzato nelle versioni AWS Glue 4.0 e successive.
- "username": (Obbligatorio) Il nome utente MongoDB.
- "password": (Obbligatorio) La password di MongoDB.
- "database": (Obbligatorio) Il database MongoDB da cui leggere. Questa opzione può anche essere trasmessa in additional_options quando si chiama glue_context.create_dynamic_frame_from_catalog nello script di processo.
- "collection": (Obbligatorio) La raccolta MongoDB da cui leggere. Questa opzione può anche essere trasmessa in additional_options quando si chiama glue_context.create_dynamic_frame_from_catalog nello script di processo.

"connectionType": "mongodb" come sorgente

Utilizzare le seguenti opzioni di connessione con "connectionType": "mongodb" come origine:

- "ssl": (Facoltativo) Se il valore è true, avvia una connessione SSL. Il valore predefinito è false.
- "ssl.domain_match": (Facoltativo) Se i valori true e ssl sono true, viene eseguito il controllo della corrispondenza del dominio. Il valore predefinito è true.
- "batchSize": (Facoltativo): il numero di documenti da restituire per ogni batch, utilizzato all'interno del cursore dei batch interni.
- "partitioner": (Facoltativo): il nome della classe del partizionatore per la lettura dei dati di input da MongoDB. Il connettore fornisce i seguenti partizionatori:
  - MongoDefaultPartitioner (impostazione predefinita) (non supportato in AWS Glue 4.0)
  - MongoSamplePartitioner (Richiede MongoDB 3.2 o versioni successive) (non supportato in AWS Glue 4.0)
  - MongoShardedPartitioner (non supportato in AWS Glue 4.0)
  - MongoSplitVectorPartitioner (non supportato in AWS Glue 4.0)
  - MongoPaginateByCountPartitioner (non supportato in AWS Glue 4.0)
  - MongoPaginateBySizePartitioner (non supportato in AWS Glue 4.0)
  - com.mongodb.spark.sql.connector.read.partitioner.SinglePartitionPartitioner
  - com.mongodb.spark.sql.connector.read.partitioner.ShardedPartitioner
  - com.mongodb.spark.sql.connector.read.partitioner.PaginateIntoPartitionsPartit

- `"partitionerOptions"` (Facoltativo): opzioni per il partizionatore designato. Per ogni partizionatore sono supportate le seguenti opzioni:
  - `MongoSamplePartitioner`: `partitionKey`, `partitionSizeMB`, `samplesPerPartition`
  - `MongoShardedPartitioner`: `shardkey`
  - `MongoSplitVectorPartitioner`: `partitionKey`, `partitionSizeMB`
  - `MongoPaginateByCountPartitioner`: `partitionKey`, `numberOfPartitions`
  - `MongoPaginateBySizePartitioner`: `partitionKey`, `partitionSizeMB`

  Per ulteriori informazioni su queste opzioni, vedere Partitioner Configuration (Configurazione partizionatore) nella documentazione di MongoDB.

"connectionType": "mongodb" come sink

Utilizzare le seguenti opzioni di connessione con `"connectionType": "mongodb"` come sink:

- `"ssl"`: (Facoltativo) Se il valore è `true`, avvia una connessione SSL. Il valore predefinito è `false`.
- `"ssl.domain_match"`: (Facoltativo) Se i valori `true` e `ssl` sono `true`, viene eseguito il controllo della corrispondenza del dominio. Il valore predefinito è `true`.
- `"extendedBsonTypes"`: (Facoltativo) Se il valore è`true`, permette i tipi BSON estesi durante la scrittura di dati su MongoDB. Il valore predefinito è `true`.
- `"replaceDocument"`: (Facoltativo) Se il valore è `true`, sostituisce l'intero documento quando si salvano set di dati che contengono un campo `_id`. Se il valore è `false`, vengono aggiornati solo i campi del documento che corrispondono ai campi del set di dati. Il valore predefinito è `true`.
- `"maxBatchSize"`: (Facoltativo): la dimensione massima del batch per le operazioni in blocco durante il salvataggio dei dati. Il valore predefinito è 512.
- `"retryWrites"`: (Facoltativo): riprova automaticamente alcune operazioni di scrittura una sola volta se AWS Glue rileva un errore di rete.

Connessioni SAP HANA

È possibile utilizzare AWS Glue per Spark per leggere e scrivere su tabelle in SAP HANA in AWS Glue 4.0 e versioni successive. È possibile definire cosa leggere da SAP HANA con una query SQL. Ci si connette a SAP HANA utilizzando le credenziali JDBC archiviate in AWS Secrets Manager tramite una connessione SAP HANA AWS Glue.

Per ulteriori informazioni sulle porte SAP HANA JDBC, consulta la documentazione SAP HANA.

Configurazione delle connessioni SAP HANA

Per connetterti a SAP HANA da AWS Glue, dovrai creare e archiviare le tue credenziali SAP HANA in un segreto AWS Secrets Manager, quindi associare tale segreto a una connessione SAP HANA AWS Glue. Dovrai configurare la connettività di rete tra il tuo servizio SAP HANA e AWS Glue.

Per connetterti a SAP HANA, potrebbero essere necessari alcuni prerequisiti:

- Se il tuo servizio SAP HANA si trova in un Amazon VPC, configura Amazon VPC per consentire al processo AWS Glue di comunicare con il servizio SAP HANA senza che il traffico attraversi la rete Internet pubblica.

  In Amazon VPC, identifica o crea un VPC, una sottorete e un gruppo di sicurezza che AWS Glue utilizzerà durante l'esecuzione del processo. Inoltre, assicurati che Amazon VPC sia configurato per consentire il traffico di rete tra l'endpoint SAP HANA e questa posizione. Il tuo processo dovrà stabilire una connessione TCP con la tua porta SAP HANA JDBC. Per ulteriori informazioni sulle porte SAP HANA, consulta la documentazione SAP HANA. In base al layout di rete, potrebbe richiedere modifiche alle regole dei gruppi di sicurezza, alle liste di controllo accessi di rete, ai gateway NAT e alle connessioni peering.
- Non ci sono prerequisiti aggiuntivi se l'endpoint SAP HANA è accessibile a Internet.

Per configurare una connessione a SAP HANA:

1. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali SAP HANA. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *`secretName`*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `user` con il valore *`saphanaUsername`*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *`saphanaPassword`*.
2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *`connectionName`*, per l'uso futuro in AWS Glue.

   - In Tipo di connessione, seleziona SAP HANA.

- Quando fornisci l'URL SAP HANA, fornisci l'URL per la tua istanza.

  Gli URL SAP HANA JDBC sono nel modulo `jdbc:sap://saphanaHostname:saphanaPort/?databaseName=saphanaDBname,Paramete`

  AWS Glue richiede i seguenti parametri URL JDBC:

  - `databaseName` - Un database predefinito in SAP HANA a cui connettersi.
- Quando selezioni il Segreto AWS, fornisci *`secretName`*.

Dopo aver creato una connessione AWS Glue SAP HANA, è necessario eseguire le seguenti operazioni prima di eseguire il processo AWS Glue:

- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *`secretName`*.
- Nella configurazione del processo AWS Glue, fornisci *`connectionName`* come Connessione di rete aggiuntiva.

Lettura da tabelle SAP HANA

Prerequisiti:

- Una tabella SAP HANA da cui si desidera leggere. Avrai bisogno delle informazioni di identificazione per la tabella.

  Una tabella può essere specificata con un nome di tabella SAP HANA e di schema, nel modulo *`schemaName`*`.`*`tableName`*. Il nome dello schema e il separatore "." non sono necessari se la tabella si trova nello schema predefinito, "pubblico". Chiamalo *`tableIdentifier`*. Il database viene fornito come parametro URL JDBC in `connectionName`.
- Una connessione AWS Glue SAP HANA configurata per fornire informazioni di autenticazione. Completa i passaggi della procedura precedente, Per configurare una connessione a SAP HANA per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *`connectionName`*.

Ad esempio:

```
saphana_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="saphana",
    connection_options={
```

```
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier",
    }
)
```

È possibile anche fornire una query SELECT SQL per filtrare i risultati restituiti al DynamicFrame. Sarà necessario configurare `query`.

Ad esempio:

```
saphana_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="saphana",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Scrittura su tabelle SAP HANA

Questo esempio scrive informazioni a partire da un DynamicFrame esistente, *`dynamicFrame`* a SAP HANA. Se la tabella contiene già informazioni, AWS Glue genererà un errore.

Prerequisiti:

- Una tabella SAP HANA su cui scrivere.

  Una tabella può essere specificata con un nome di tabella SAP HANA e di schema, nel modulo *`schemaName`*`.`*`tableName`*. Il nome dello schema e il separatore "." non sono necessari se la tabella si trova nello schema predefinito, "pubblico". Chiamalo *`tableIdentifier`*. Il database viene fornito come parametro URL JDBC in `connectionName`.
- Informazioni di autenticazione SAP HANA. Completa i passaggi della procedura precedente, Per configurare una connessione a SAP HANA per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *`connectionName`*.

Ad esempio:

```
options = {
    "connectionName": "connectionName",
    "dbtable": 'tableIdentifier'
```

```
}

    saphana_write = glueContext.write_dynamic_frame.from_options(
        frame=dynamicFrame,
        connection_type="saphana",
        connection_options=options
)
```

Indicazioni di riferimento alle opzioni di connessione a SAP HANA

- `connectionName`: obbligatorio. Utilizzato per la lettura/scrittura. Il nome di una connessione a SAP HANA AWS Glue configurata per fornire informazioni di autenticazione e sulla rete al metodo di connessione.
- `databaseName`: utilizzato per la lettura/scrittura. Valori validi: nomi dei database in SAP HANA. Nome del database a cui connettersi.
- `dbtable` — Richiesto per la scrittura, richiesto per la lettura a meno che non `query` sia fornito. Utilizzato per la lettura/scrittura. Valori validi: contenuto di una clausola SAP HANA SQL FROM. Identifica una tabella in SAP HANA a cui connettersi. È inoltre possibile fornire un codice SQL diverso dal nome della tabella, ad esempio una sottoquery. Per ulteriori informazioni, consulta la clausola From nella documentazione di SAP HANA.
- `query`: utilizzato per la lettura. Una query SAP HANA SQL SELECT che definisce cosa recuperare durante la lettura da SAP HANA.

Connessioni Snowflake

È possibile utilizzare AWS Glue per Spark per leggere e scrivere su tabelle in Snowflake in AWS Glue 4.0 e versioni successive. È possibile leggere da Snowflake con una query SQL. È possibile connettersi a Snowflake utilizzando un utente e una password. È possibile fare riferimento alle credenziali Snowflake archiviate in AWS Secrets Manager attraverso Catalogo dati AWS Glue. Le credenziali Catalogo dati Snowflake per AWS Glue per Spark vengono archiviate separatamente dalle credenziali Catalogo dati Snowflake per i crawler. È necessario scegliere un tipo di connessione `SNOWFLAKE` e non un tipo di connessione `JDBC`configurato per la connessione a Snowflake.

Per ulteriori informazioni su Snowflake, consulta il sito Web di Snowflake. Per ulteriori informazioni su Snowflake su AWS, consulta la pagina Snowflake Data Warehouse on Amazon Web Services.

Configurazione delle connessioni Snowflake

Non sussistono prerequisiti AWS per la connessione ai database Snowflake disponibili su Internet.

Facoltativamente, è possibile applicare la seguente configurazione per gestire le credenziali di connessione con AWS Glue.

Gestione delle credenziali di connessione con AWS Glue

1. In Snowflake, genera un utente, *sowflakeUser*, e una password, *snowflakePassword*.
2. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali Snowflake. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.
   - Quando selezioni le coppie chiave/valore, crea una coppia per *snowflakeUser* con la chiave `sfUser`.
   - Quando selezioni le coppie chiave/valore, crea una coppia per *snowflakePassword* con la chiave `sfPassword`.
   - Quando selezioni le coppie chiave/valore, puoi fornire la chiave `sfWarehouse` al tuo warehouse Snowflake.
3. In Catalogo dati AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per il passaggio successivo.
   - In Tipo di connessione, seleziona Snowflake.
   - Quando selezioni l'URL Snowflake, fornisci l'URL della tua istanza Snowflake. L'URL utilizzerà un nome host nel modulo *account_identifier*`.snowflakecomputing.com`.
   - Quando selezioni il Segreto AWS, fornisci *secretName*.
4. Nella configurazione del processo AWS Glue, fornisci *connectionName* come Connessione di rete aggiuntiva.

Nelle seguenti situazioni, potresti aver bisogno di quanto segue:

- Per Snowflake ospitato su AWS in un Amazon VPC
  - Avrai bisogno di una configurazione Amazon VPC appropriata per Snowflake. Per ulteriori informazioni su come configurare il tuo Amazon VPC, consulta la sezione AWS PrivateLink & Snowflake nella documentazione di Snowflake.
  - Avrai bisogno di una configurazione Amazon VPC appropriata per AWS Glue. the section called “Endpoint VPC (AWS PrivateLink)”.

- Dovrai creare una connessione a Catalogo dati AWS Glue che fornisca le informazioni di connessione Amazon VPC (oltre all'ID di un segreto AWS Secrets Manager che definisce le tue credenziali di sicurezza Snowflake). L'URL cambierà durante l'utilizzo di AWS PrivateLink, come descritto nella documentazione di Snowflake, un collegamento alla quale è stato fornito in precedenza.
- È necessario che la configurazione del processo includa la connessione a Catalogo dati come Connessione di rete aggiuntiva.

Lettura dalle tabelle Snowflake

Prerequisiti: una tabella Snowflake da cui desideri leggere. Avrai bisogno del nome della tabella Snowflake, *tableName*. Avrai bisogno del tuo URL Snowflake, *snowflakeUrl*, del nome utente, *snowflakeUser*, e della password, *snowflakePassword*. Se il tuo utente Snowflake non dispone di uno spazio dei nomi predefinito, avrai bisogno del nome del database Snowflake, *databaseName* e del nome dello schema *SchemaName*. Inoltre, se il tuo utente Snowflake non dispone di un set di warehouse predefinito, avrai bisogno di un nome di warehouse *WarehouseName*.

Per esempio:

Prerequisiti aggiuntivi: completa la procedura Gestione delle credenziali di connessione con AWS Glue per configurare *snowflakeUrl*, *snowflakeUsername* e *snowflakePassword*. Per esaminare questi passaggi, consulta the section called “Configurazione di Snowflake”, la sezione precedente. Per selezionare la connessione di rete aggiuntiva con la quale connettersi, utilizzeremo il parametro `connectionName`.

```
snowflake_read = glueContext.create_dynamic_frame.from_options(
  connection_type="snowflake",
  connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    }
)
```

Inoltre, puoi utilizzare i parametri `autopushdown` e `query` per leggere una parte di una tabella Snowflake. Questo può essere molto più efficiente rispetto al filtraggio dei risultati dopo che sono stati caricati in Spark. Prendiamo in esame un esempio in cui tutte le vendite sono archiviate nella stessa

tabella, ma è necessario analizzare solo le vendite di un determinato negozio nei giorni festivi. Se tali informazioni sono archiviate nella tabella, è possibile utilizzare il predicato pushdown per recuperare i risultati come segue:

```
snowflake_node = glueContext.create_dynamic_frame.from_options(
    connection_type="snowflake",
    connection_options={
        "autopushdown": "on",
        "query": "select * from sales where store='1' and IsHoliday='TRUE'",
        "connectionName": "snowflake-glue-conn",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    }
)
```

Scrittura su tabelle Snowflake

Prerequisiti: un database Snowflake su cui scrivere. Avrai bisogno di un nome di tabella attuale o desiderato, *tableName*. Avrai bisogno del tuo URL Snowflake, *snowflakeUrl*, del nome utente, *snowflakeUser*, e della password, *snowflakePassword*. Se il tuo utente Snowflake non dispone di uno spazio dei nomi predefinito, avrai bisogno del nome del database Snowflake, *databaseName* e del nome dello schema *SchemaName*. Inoltre, se il tuo utente Snowflake non dispone di un set di warehouse predefinito, avrai bisogno di un nome di warehouse *WarehouseName*.

Per esempio:

Prerequisiti aggiuntivi: completa la procedura Gestione delle credenziali di connessione con AWS Glue per configurare *snowflakeUrl*, *snowflakeUsername* e *snowflakePassword*. Per esaminare questi passaggi, consulta the section called “Configurazione di Snowflake”, la sezione precedente. Per selezionare la connessione di rete aggiuntiva con la quale connettersi, utilizzeremo il parametro `connectionName`.

```
glueContext.write_dynamic_frame.from_options(
    connection_type="snowflake",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
```

```
    },
)
```

Indicazioni di riferimento alle opzioni di connessione a Snowflake

Il tipo di connessione Snowflake accetta le seguenti opzioni di connessione:

È possibile recuperare alcuni dei parametri di questa sezione da una connessione a Catalogo dati (`sfUrl`, `sfUser` e `sfPassword`), nel qual caso non è necessario fornirli. È possibile farlo fornendo il parametro `connectionName`.

È possibile recuperare alcuni dei parametri di questa sezione da un segreto AWS Secrets Manager (`sfUser`, `sfPassword`), nel qual caso non è necessario fornirli. Il segreto deve fornire il contenuto sotto le chiavi `sfUser` e `sfPassword`. È possibile farlo fornendo il parametro `secretId`.

Per la connessione a Snowflake generalmente vengono utilizzati i seguenti parametri.

- `sfDatabase`: obbligatorio se in Snowflake non è impostato un valore predefinito per l'utente. Utilizzato per la lettura/scrittura. Il database da utilizzare per la sessione dopo la connessione.
- `sfSchema`: obbligatorio se in Snowflake non è impostato un valore predefinito per l'utente. Utilizzato per la lettura/scrittura. Lo schema da utilizzare per la sessione dopo la connessione.
- `sfWarehouse`: obbligatorio se in Snowflake non è impostato un valore predefinito per l'utente. Utilizzato per la lettura/scrittura. Il warehouse virtuale predefinito da utilizzare per la sessione dopo la connessione.
- `sfRole`: obbligatorio se in Snowflake non è impostato un valore predefinito per l'utente. Utilizzato per la lettura/scrittura. Il ruolo di sicurezza predefinito da utilizzare per la sessione dopo la connessione.
- `sfUrl`: (obbligatorio) utilizzato per la lettura/scrittura. Specifica il nome host del tuo account nel seguente formato: *`account_identifier`*`.snowflakecomputing.com`. Per ulteriori informazioni sugli identificatori di account, consulta la pagina Account Identifiers nella documentazione di Snowflake.
- `sfUser`: (obbligatorio) utilizzato per la lettura/scrittura. Il nome di accesso per l'utente Snowflake.
- `sfPassword` (obbligatorio se non viene fornito `pem_private_key`). Utilizzato per lettura/scrittura. La password per l'utente Snowflake.
- `dbtable`: obbligatorio quando si lavora con tabelle complete. Utilizzato per la lettura/scrittura. Il nome della tabella da leggere o la tabella in cui vengono scritti i dati. Durante la lettura, vengono recuperate tutte le colonne e i record.

- `pem_private_key`: utilizzato per la lettura/scrittura. Una stringa di chiave privata non crittografata con codifica b64. La chiave privata per l'utente Snowflake. È comune copiare tale chiave da un file PEM. Per ulteriori informazioni, consulta Autenticazione e rotazione delle coppie di chiavi nella documentazione di Snowflake.
- `query`: obbligatorio durante la lettura con una query. Utilizzato per la lettura. La query esatta (istruzione `SELECT`) da eseguire

Le seguenti opzioni vengono utilizzate per configurare comportamenti specifici durante il processo di connessione a Snowflake.

- `preactions`: utilizzato per la lettura/scrittura. Valori validi: elenco di istruzioni SQL separato da punto e virgola in formato stringa. Le istruzioni SQL vengono eseguite prima del trasferimento dei dati tra AWS Glue e Snowflake. Se un'istruzione contiene `%s`, `%s` viene sostituito con il nome della tabella a cui si fa riferimento per l'operazione.
- `postactions`: utilizzato per la lettura/scrittura. Le istruzioni SQL vengono eseguite dopo il trasferimento dei dati tra AWS Glue e Snowflake. Se un'istruzione contiene `%s`, `%s` viene sostituito con il nome della tabella a cui si fa riferimento per l'operazione.
- `autopushdown`: valore predefinito: `"on"`. Valori validi: `"on"`, `"off"`. Questo parametro controlla se il pushdown automatico delle query è abilitato. Se il pushdown è abilitato, quando su Spark viene eseguita una query, se una parte di essa può essere "trasferita" al server Snowflake, viene sottoposta a pushdown. Ciò migliora le prestazioni di alcune query. Per sapere se la tua query può essere spostata verso il basso, consulta la sezione Pushdown nella documentazione di Snowflake.

Inoltre, alcune delle opzioni disponibili sul connettore Snowflake Spark potrebbero essere supportate in AWS Glue. Per ulteriori informazioni sulle opzioni disponibili sul connettore Snowflake Spark, consulta la sezione Setting Configuration Options for the Connector nella documentazione di Snowflake.

Limitazioni del connettore Snowflake

La connessione a Snowflake con AWS Glue per Spark è soggetta alle seguenti limitazioni.

- Questo connettore non supporta i segnalibri di processo. Per ulteriori informazioni sui segnalibri di processo, consultare the section called “Monitoraggio dei dati elaborati mediante segnalibri di processo”.

- Questo connettore non supporta la lettura e la scrittura di Snowflake tramite le tabelle in Catalogo dati AWS Glue utilizzando i metodi `create_dynamic_frame.from_catalog` e `write_dynamic_frame.from_catalog`.
- Questo connettore non supporta la connessione a Snowflake con credenziali diverse da utente e password.
- Questo connettore non è supportato nei processi di flussi di dati.
- Questo connettore supporta le query basate su istruzioni `SELECT` per il recupero di informazioni, ad esempio con il parametro `query`. Altri tipi di query (ad esempio istruzioni DML, `SHOW` o `DESC`) non sono supportati.
- Snowflake limita la dimensione del testo della query (ad esempio istruzioni SQL) inviato tramite i client Snowflake a 1 MB per istruzione. Per ulteriori informazioni, consulta la pagina Limits on Query Text Size.

Connessioni Teradata Vantage

È possibile utilizzare AWS Glue for Spark per leggere e scrivere su tabelle esistenti in Teradata Vantage in AWS Glue 4.0 e versioni successive. È possibile definire cosa leggere da Teradata con una query SQL. È possibile connettersi a Teradata utilizzando le credenziali di nome utente e password memorizzate tramite AWS Secrets Manager una connessione AWS Glue.

Per ulteriori informazioni su Teradata, consulta la documentazione di Teradata.

Configurazione delle connessioni Teradata

Per connetterti a Teradata da AWS Glue, dovrai creare e archiviare le tue credenziali Teradata in un luogo AWS Secrets Manager segreto, quindi associare quel segreto a una connessione Glue Teradata. AWS Se la tua istanza Teradata si trova in un Amazon VPC, dovrai anche fornire opzioni di rete alla tua connessione AWS Glue Teradata.

Per connettersi a Teradata da AWS Glue, potrebbero essere necessari alcuni prerequisiti:

- Se accedi al tuo ambiente Teradata tramite Amazon VPC, configura Amazon VPC per consentire al tuo job AWS Glue di comunicare con l'ambiente Teradata. Sconsigliamo l'accesso all'ambiente Teradata tramite la rete Internet pubblica.

  In Amazon VPC, identifica o crea un VPC, una sottorete e un gruppo di sicurezza che AWS Glue utilizzerà durante l'esecuzione del lavoro. Inoltre, assicurati che Amazon VPC sia configurato per consentire il traffico di rete tra l'istanza Teradata e questa posizione. Il tuo processo dovrà stabilire

una connessione TCP con la tua porta del client Teradata. Per ulteriori informazioni sulle porte Teradata, consulta la documentazione di Teradata.

In base al layout di rete, la connettività VPC sicura potrebbe richiedere modifiche ad Amazon VPC e ad altri servizi di rete. Per ulteriori informazioni sulla AWS connettività, consulta le opzioni di AWS connettività nella documentazione di Teradata.

Per configurare una connessione AWS Glue Teradata:

1. *Nella configurazione Teradata, identifica o crea un utente e una password con cui AWS Glue si connetterà, TeraDataUser e TeraDataPassword.* Per ulteriori informazioni, consulta Vantage Security Overview nella documentazione di Teradata.
2. Nel, crea un segreto utilizzando le AWS Secrets Manager tue credenziali Teradata. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Crea un AWS Secrets Manager segreto nella AWS Secrets Manager documentazione. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `user` con il valore *teradataUsername*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *teradataPassword*.
3. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati di seguitothe section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per il passaggio successivo.

   - In Tipo di connessione, seleziona Snowflake.
   - Quando fornisci JDBC URL, fornisci l'URL per la tua istanza. Puoi anche codificare determinati parametri di connessione, separati da virgole, nel tuo URL JDBC. L'URL deve rispettare il seguente formato: `jdbc:teradata://`*teradataHostname*`/`*ParameterName*`=`*ParameterValue*`,`*ParameterName*

     I parametri URL supportati includono:

     - `DATABASE`: nome del database sull'host a cui accedere per impostazione predefinita.
     - `DBS_PORT`: la porta del database, utilizzata con una porta non standard.

- *Quando selezioni un* ***tipo di credenziali****, seleziona* ***AWS Secrets Manager****, quindi imposta* ***Segreto di AWS*** *su secretName*.

4. Nelle seguenti situazioni, potresti aver bisogno di una configurazione aggiuntiva:

   - Per le istanze Teradata ospitate su AWS un Amazon VPC
     - Dovrai fornire le informazioni di connessione Amazon VPC alla connessione AWS Glue che definisce le tue credenziali di sicurezza Teradata. Durante la creazione o l'aggiornamento della connessione, imposta VPC, sottorete e Gruppi di sicurezza nelle opzioni di rete.

Dopo aver creato una connessione AWS Glue Teradata, dovrai eseguire i seguenti passaggi prima di chiamare il metodo di connessione.

- Concedi al ruolo IAM associato al tuo lavoro AWS Glue il permesso di leggere *SecretName*.
- Nella configurazione del lavoro AWS Glue, fornisci *ConnectionName* come connessione di rete aggiuntiva.

Lettura da Teradata

Prerequisiti:

- Una tabella Teradata da cui si desidera leggere. Avrai bisogno del nome della tabella, *tableName*.
- Una connessione AWS Glue Teradata configurata per fornire informazioni di autenticazione. Completa i passaggi Per configurare una connessione a Teradata per configurare le informazioni di autenticazione. È necessario il nome della connessione AWS Glue, *ConnectionName*.

Per esempio:

```
teradata_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName"
    }
)
```

Puoi anche fornire una query SQL SELECT per filtrare i risultati restituiti al tuo. DynamicFrame Sarà necessario configurare `query`.

Per esempio:

```
teradata_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Scrittura su tabelle Teradata

Prerequisiti: una tabella Teradata su cui scrivere, *tableName*. È necessario creare la tabella prima di chiamare il metodo di connessione.

Per esempio:

```
teradata_write = glueContext.write_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName"
    }
)
```

Indicazioni di riferimento alle opzioni di connessione a Teradata

- `connectionName`: obbligatorio. Utilizzato per la lettura/scrittura. Il nome di una connessione AWS Glue Teradata configurata per fornire informazioni di autenticazione e di rete al metodo di connessione utilizzato.
- `dbtable` — Richiesto per la scrittura, richiesto per la lettura a meno che non `query` sia fornito. Utilizzato per la lettura/scrittura. Il nome di una tabella con cui interagirà il metodo di connessione.
- `query`: utilizzato per la lettura. Una query SELECT SQL che definisce cosa recuperare durante la lettura da Teradata.

Connessioni Vertica

È possibile utilizzare AWS Glue per Spark per leggere e scrivere su tabelle in Vertica in AWS Glue 4.0 e versioni successive. È possibile definire cosa leggere da Vertica con una query SQL. Connettiti

a Vertica utilizzando le credenziali di nome utente e password memorizzate in AWS Secrets Manager tramite una connessione AWS Glue.

Per ulteriori informazioni su Vertica, consulta la documentazione di Vertica.

Configurazione delle connessioni Vertica

Per connetterti a Vertica da AWS Glue, dovrai creare e archiviare le tue credenziali Vertica in un segreto AWS Secrets Manager, quindi associare tale segreto a una connessione Vertica AWS Glue. Se la tua istanza Vertica si trova in un Amazon VPC, dovrai anche fornire opzioni di rete alla tua connessione AWS Glue Vertica. Ti servirà un bucket o una cartella Amazon S3 da utilizzare per l'archiviazione temporanea durante la lettura e la scrittura sul database.

Per connettersi a Vertica da AWS Glue, potrebbero essere necessari alcuni prerequisiti:

- Un bucket o una cartella Amazon S3 da utilizzare per l'archiviazione temporanea durante la lettura e la scrittura sul database, a cui fa riferimento *tempS3Path*.

  > **Note**
  >
  > Quando utilizzi Vertica nelle anteprime dei dati del processo di AWS Glue, i file temporanei potrebbero non essere rimossi automaticamente da *tempS3Path*. Per garantire la rimozione dei file temporanei, interrompi direttamente la sessione di anteprima dei dati scegliendo Termina sessione nel riquadro Anteprima dei dati.
  > Se non sei in grado di terminare direttamente la sessione di anteprima dei dati, valuta la possibilità di impostare la configurazione del ciclo di vita di Amazon S3 per rimuovere i dati obsoleti. Consigliamo di rimuovere i dati più vecchi di 49 ore, in base al runtime massimo del processo in aggiunta a un margine. Per ulteriori informazioni sulla configurazione del ciclo di vita di Amazon S3, consulta Gestione del ciclo di vita dello storage nella documentazione di Amazon S3.

- Una policy IAM con autorizzazioni appropriate per il percorso Amazon S3 che puoi associare al ruolo di processo AWS Glue.
- Se la tua istanza VPC si trova in un Amazon VPC, configura Amazon VPC per consentire al processo AWS Glue di comunicare con l'istanza Vertica senza che il traffico attraversi la rete Internet pubblica.

  In Amazon VPC, identifica o crea un VPC, una sottorete e un gruppo di sicurezza che AWS Glue utilizzerà durante l'esecuzione del processo. Inoltre, assicurati che Amazon VPC sia configurato

per consentire il traffico di rete tra l'istanza Vertica e questa posizione. Il tuo processo dovrà stabilire una connessione TCP con la tua porta del client Vertica, (per impostazione predefinita, 5433). In base al layout di rete, potrebbe richiedere modifiche alle regole dei gruppi di sicurezza, alle liste di controllo accessi di rete, ai gateway NAT e alle connessioni peering.

È quindi possibile procedere alla configurazione di AWS Glue per l'uso con Vertica.

Per configurare una connessione a Vertica:

1. In AWS Secrets Manager, crea un segreto utilizzando le tue credenziali Vertica, *verticaUsername* e *verticaPassword*. Per creare un segreto in Secrets Manager, segui il tutorial disponibile in Create an AWS Secrets Manager secret nella documentazione di AWS Secrets Manager. Dopo aver creato il segreto, prendi nota del nome, *secretName*, per il passaggio successivo.

   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `user` con il valore *verticaUsername*.
   - Quando selezioni le coppie chiave/valore, crea una coppia per la chiave `password` con il valore *verticaPassword*.

2. Nella console AWS Glue, crea una connessione seguendo i passaggi riportati in the section called “Aggiunta di una connessione AWS Glue”. Dopo aver creato la connessione, prendi nota del nome, *connectionName*, per il passaggio successivo.

   - In Tipo di connessione, seleziona Vertica.
   - In Host Vertica, fornisci il nome host dell'installazione Vertica.
   - In Porta Vertica, indica la porta tramite cui è disponibile l'installazione di Vertica.
   - Quando selezioni il Segreto AWS, fornisci *secretName*.

3. Nelle seguenti situazioni, potresti aver bisogno di una configurazione aggiuntiva:

   - Per le istanze Vertica ospitate su AWS in un Amazon VPC
     - Fornisci le informazioni di connessione Amazon VPC alla connessione AWS Glue che definisce le credenziali di sicurezza Vertica. Durante la creazione o l'aggiornamento della connessione, imposta VPC, sottorete e Gruppi di sicurezza nelle opzioni di rete.

Dopo aver creato una connessione AWS Glue Vertica, dovrai eseguire i seguenti passaggi prima di chiamare il metodo di connessione.

- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso per *tempS3Path*.
- Concedi al ruolo IAM associato al tuo processo AWS Glue il permesso di leggere *secretName*.
- Nella configurazione del processo AWS Glue, fornisci *connectionName* come Connessione di rete aggiuntiva.

Lettura da Vertica

Prerequisiti:

- Una tabella Vertica da cui si desidera leggere. Avrai bisogno del nome del database Vertica, *dbName*, e della tabella, *tableName*.
- Una connessione AWS Glue Vertica configurata per fornire informazioni di autenticazione. Completa i passaggi della procedura precedente, Per configurare una connessione a Vertica per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *connectionName*.
- Un bucket o una cartella Amazon S3 da utilizzare per lo storage temporaneo, menzionato in precedenza. Avrai bisogno del nome, *tempS3Path*. Dovrai connetterti a questa posizione utilizzando il protocollo `s3a`.

Ad esempio:

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "table": "tableName",
    }
)
```

È possibile anche fornire una query SELECT SQL per filtrare i risultati restituiti al DynamicFrame o accedere a un set di dati da più tabelle.

Ad esempio:

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="vertica",
```

```
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "query": "select * FROM tableName",
    },
)
```

Scrittura su tabelle Vertica

Questo esempio scrive informazioni a partire da un DynamicFrame esistente, *dynamicFrame* a Vertica. Se la tabella contiene già informazioni, AWS Glue aggiungerà i dati da DynamicFrame.

Prerequisiti:

- Un nome di tabella corrente o desiderato, *tableName*, su cui scrivere. Avrai anche bisogno del nome del database Vertica corrispondente, *dbName*.
- Una connessione AWS Glue Vertica configurata per fornire informazioni di autenticazione. Completa i passaggi della procedura precedente, Per configurare una connessione a Vertica per configurare le informazioni di autenticazione. Sarà necessario il nome della connessione AWS Glue, *connectionName*.
- Un bucket o una cartella Amazon S3 da utilizzare per lo storage temporaneo, menzionato in precedenza. Avrai bisogno del nome, *tempS3Path*. Dovrai connetterti a questa posizione utilizzando il protocollo `s3a`.

Ad esempio:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "table": "tableName",
    }
)
```

Indicazioni di riferimento alle opzioni di connessione a Vertica

- `connectionName`: obbligatorio. Utilizzato per la lettura/scrittura. Il nome di una connessione a Vertica AWS Glue configurata per fornire informazioni di autenticazione e sulla rete al metodo di connessione.
- `db`: obbligatorio. Utilizzato per la lettura/scrittura. Il nome dell'indice in Vertica con cui interagirà il metodo di connessione.
- `dbSchema` — Obbligatorio se necessario per identificare la tabella. Utilizzato per la lettura/scrittura. Default: `public`. Il nome di uno schema con cui interagirà il metodo di connessione.
- `table` — Richiesto per la scrittura, richiesto per la lettura a meno che non `query` sia fornito. Utilizzato per la lettura/scrittura. Il nome di una tabella con cui interagirà il metodo di connessione.
- `query`: utilizzato per la lettura. Una query SELECT SQL che definisce cosa recuperare durante la lettura da Teradata.
- `staging_fs_url`: obbligatorio. Utilizzato per la lettura/scrittura. Valori validi: URL `s3a`. L'URL di un bucket o di una cartella Amazon S3 da utilizzare per l'archiviazione temporanea.

Valori di personalizzazione e connectionType Marketplace AWS

Questi sono i seguenti:

- `"connectionType": "marketplace.athena"`: designa una connessione a un archivio dati Amazon Athena. La connessione utilizza un connettore di Marketplace AWS.
- `"connectionType": "marketplace.spark"`: designa una connessione a un archivio dati Apache Spark. La connessione utilizza un connettore di Marketplace AWS.
- `"connectionType": "marketplace.jdbc"`: designa una connessione a un archivio dati JDBC. La connessione utilizza un connettore di Marketplace AWS.
- `"connectionType": "custom.athena"`: designa una connessione a un archivio dati Amazon Athena. La connessione utilizza un connettore personalizzato che va caricato in AWS Glue Studio.
- `"connectionType": "custom.spark"`: designa una connessione a un archivio dati Apache Spark. La connessione utilizza un connettore personalizzato che va caricato in AWS Glue Studio.
- `"connectionType": "custom.jdbc"`: designa una connessione a un archivio dati JDBC. La connessione utilizza un connettore personalizzato che va caricato in AWS Glue Studio.

Opzioni di connessione per il tipo custom.jdbc o marketplace.jdbc

- `className`: stringa, obbligatorio, nome della classe driver.
- `connectionName`: stringa, obbligatorio, nome della connessione associata al connettore.
- `url`: stringa, obbligatorio, URL JDBC con segnaposto (`${}`) che vengono utilizzati per creare la connessione all'origine dati. Il segnaposto `${secretKey}` viene sostituito con il segreto con lo stesso nome in AWS Secrets Manager. Per ulteriori informazioni sulla creazione dell'URL, fare riferimento alla documentazione dell'archivio dati.
- `secretId` o `user/password`: stringa, obbligatorio, utilizzato per recuperare le credenziali per l'URL.
- `dbTable` o `query`: stringa, obbligatorio, la tabella o la query SQL da cui ottenere i dati. Puoi specificare `dbTable` o `query`, ma non entrambi.
- `partitionColumn`: stringa, facoltativo, il nome di una colonna intera utilizzata per il partizionamento. Questa opzione funziona solo quando è inclusa con `lowerBound`, `upperBound` e `numPartitions`. Questa opzione funziona allo stesso modo del lettore Spark SQL JDBC. Per ulteriori informazioni, consulta Da JDBC ad altri database nel manuale Apache Spark SQL, DataFrames and Datasets Guide.

  I valori `lowerBound` e `upperBound` vengono utilizzati per decidere lo stride della partizione, non per filtrare le righe nella tabella. Tutte le righe della tabella vengono partizionate e restituite.

  > **Note**
  >
  > Quando si utilizza una query anziché un nome di tabella, è necessario verificare che la query funzioni con la condizione di partizionamento specificata. Ad esempio:
  >
  > - Se il formato della query è `"SELECT col1 FROM table1"`, testa la query aggiungendo una clausola `WHERE` alla fine della query che utilizza la colonna della partizione.
  > - Se il formato della query è `"SELECT col1 FROM table1 WHERE col2=val"`, testa la query estendendo la clausola `WHERE` con `AND` e un'espressione che utilizza la colonna della partizione.

- `lowerBound`: intero, facoltativo, il valore minimo di `partitionColumn` che viene utilizzato per decidere lo stride della partizione.
- `upperBound`: intero, facoltativo, il valore massimo di `partitionColumn` che viene utilizzato per decidere lo stride della partizione.

- `numPartitions`: intero, facoltativo, il numero di partizioni. Questo valore, insieme a `lowerBound` (incluso) e `upperBound` (escluso), forma stride di partizione per espressioni con le clausole `WHERE` generate che vengono utilizzate per dividere la `partitionColumn`.

  > **Important**
  >
  > Presta attenzione al numero di partizioni perché troppe partizioni potrebbero causare problemi nei sistemi di database esterni.

- `filterPredicate`: stringa, opzionale, clausola condizione extra per filtrare i dati dall'origine. Ad esempio:

  ```
  BillingCity='Mountain View'
  ```

  Quando si utilizza una query anziché un nome di table, è necessario verificare che la query funzioni con il `filterPredicate` specificato. Ad esempio:

  - Se il formato della query è `"SELECT col1 FROM table1"`, testa la query aggiungendo una clausola `WHERE` alla fine della query che utilizza il predicato filtro.
  - Se il formato della query è `"SELECT col1 FROM table1 WHERE col2=val"`, testa la query estendendo la clausola `WHERE` con `AND` e un'espressione che utilizza il predicato filtro.

- `dataTypeMapping`: dizionario, opzionale, mappatura del tipo di dati personalizzata che crea una mappatura da un tipo di dati JDBC a un tipo di dati Glue. Ad esempio, l'opzione `"dataTypeMapping":{"FLOAT":"STRING"}` mappa i campi di dati di tipo JDBC `FLOAT` nel tipo Java `String` chiamando il metodo `ResultSet.getString()` del driver e lo usa per costruire registri di AWS Glue. L'oggetto `ResultSet` viene implementato da ciascun driver, quindi il comportamento è specifico del driver utilizzato. Consulta la documentazione relativa al driver JDBC per capire come il driver esegue le conversioni.
- I tipi di dati AWS Glue correntemente supportati sono:
  - DATE
  - STRING
  - TIMESTAMP
  - INT
  - FLOAT
  - LONG
  - BIGDECIMAL

- BYTE
- SHORT
- DOUBLE

I tipi di dati JDBC supportati sono Java8 java.sql.types.

Le mappature di default dei tipi di dati (da JDBC a AWS Glue) sono:

- DATE -> DATE
- VARCHAR -> STRING
- CHAR -> STRING
- LONGNVARCHAR -> STRING
- TIMESTAMP -> TIMESTAMP
- INTEGER -> INT
- FLOAT -> FLOAT
- REAL -> FLOAT
- BIT -> BOOLEAN
- BOOLEAN -> BOOLEAN
- BIGINT -> LONG
- DECIMAL -> BIGDECIMAL
- NUMERIC -> BIGDECIMAL
- TINYINT -> SHORT
- SMALLINT -> SHORT
- DOUBLE -> DOUBLE

Se si utilizza un mapping del tipo di dati personalizzato con l'opzione `dataTypeMapping`, è possibile sovrascrivere una mappatura di default del tipo di dati. Sono interessati solo i tipi di dati JDBC elencati nell'opzione `dataTypeMapping`; per tutti gli altri tipi di dati JDBC viene utilizzata la mappatura di default. Se necessario, è possibile aggiungere mappature per tipi di dati JDBC aggiuntivi. Se un tipo di dati JDBC non è incluso nella mappatura di default o in una mappatura personalizzata, per impostazione predefinita viene convertito nel tipo di dati `STRING` AWS Glue.

Negli esempi di codice Python riportati di seguito viene illustrato come leggere dai database JDBC con driver JDBC Marketplace AWS. Mostra la lettura da un database e la scrittura in una posizione S3.

```
    import sys
    from awsglue.transforms import *
    from awsglue.utils import getResolvedOptions
    from pyspark.context import SparkContext
    from awsglue.context import GlueContext
    from awsglue.job import Job

    ## @params: [JOB_NAME]
    args = getResolvedOptions(sys.argv, ['JOB_NAME'])

    sc = SparkContext()
    glueContext = GlueContext(sc)
    spark = glueContext.spark_session
    job = Job(glueContext)
    job.init(args['JOB_NAME'], args)
    ## @type: DataSource
    ## @args: [connection_type = "marketplace.jdbc", connection_options =
     {"dataTypeMapping":{"INTEGER":"STRING"},"upperBound":"200","query":"select id,
       name, department from department where id < 200","numPartitions":"4",
       "partitionColumn":"id","lowerBound":"0","connectionName":"test-connection-
jdbc"},
        transformation_ctx = "DataSource0"]
    ## @return: DataSource0
    ## @inputs: []
    DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
      "marketplace.jdbc", connection_options = {"dataTypeMapping":{"INTEGER":"STRING"},
      "upperBound":"200","query":"select id, name, department from department where
       id < 200","numPartitions":"4","partitionColumn":"id","lowerBound":"0",
       "connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
    ## @type: ApplyMapping
    ## @args: [mappings = [("department", "string", "department", "string"), ("name",
 "string",
      "name", "string"), ("id", "int", "id", "int")], transformation_ctx =
 "Transform0"]
    ## @return: Transform0
    ## @inputs: [frame = DataSource0]
    Transform0 = ApplyMapping.apply(frame = DataSource0, mappings = [("department",
 "string",
```

```
      "department", "string"), ("name", "string", "name", "string"), ("id", "int",
"id", "int")],
       transformation_ctx = "Transform0")
    ## @type: DataSink
    ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
     "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
    ## @return: DataSink0
    ## @inputs: [frame = Transform0]
    DataSink0 = glueContext.write_dynamic_frame.from_options(frame = Transform0,
      connection_type = "s3", format = "json", connection_options = {"path":
      "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
    job.commit()
```

Opzioni di connessione per il tipo custom.athena o marketplace.athena

- `className` – Stringa, obbligatorio, nome della classe driver. Quando si utilizza il connettore Athena-CloudWatch, questo valore di parametro è il prefisso della classe Name (ad esempio, `"com.amazonaws.athena.connectors"`). Il connettore Athena-CloudWatch è composto da due classi: un gestore di metadati e un gestore di registri. Se si fornisce qui il prefisso comune, l'API carica le classi corrette in base a tale prefisso.
- `tableName`: stringa, obbligatorio, il nome del flusso di log CloudWatch da leggere. In questo frammento di codice viene utilizzato il nome della vista speciale `all_log_streams`, il che significa che il frame di dati dinamico restituito conterrà i dati di tutti i flussi di log nel gruppo di log.
- `schemaName`: stringa, obbligatorio, il nome del gruppo di log CloudWatch da cui leggere. Ad esempio, `/aws-glue/jobs/output`.
- `connectionName` – Stringa, obbligatorio, nome della connessione associata al connettore.

Per ulteriori opzioni per questo connettore, consultare il README del connettore Amazon Athena CloudWatch su GitHub.

Il seguente esempio di codice Python mostra come leggere da un archivio dati Athena utilizzando un connettore Marketplace AWS. Mostra la lettura da Athena e la scrittura in una posizione S3.

```
    import sys
    from awsglue.transforms import *
    from awsglue.utils import getResolvedOptions
    from pyspark.context import SparkContext
    from awsglue.context import GlueContext
    from awsglue.job import Job
```

```
    ## @params: [JOB_NAME]
    args = getResolvedOptions(sys.argv, ['JOB_NAME'])

    sc = SparkContext()
    glueContext = GlueContext(sc)
    spark = glueContext.spark_session
    job = Job(glueContext)
    job.init(args['JOB_NAME'], args)
    ## @type: DataSource
    ## @args: [connection_type = "marketplace.athena", connection_options =
     {"tableName":"all_log_streams","schemaName":"/aws-glue/jobs/output",
      "connectionName":"test-connection-athena"}, transformation_ctx = "DataSource0"]
    ## @return: DataSource0
    ## @inputs: []
    DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
      "marketplace.athena", connection_options = {"tableName":"all_log_streams",,
      "schemaName":"/aws-glue/jobs/output","connectionName":
      "test-connection-athena"}, transformation_ctx = "DataSource0")
    ## @type: ApplyMapping
    ## @args: [mappings = [("department", "string", "department", "string"), ("name",
"string",
     "name", "string"), ("id", "int", "id", "int")], transformation_ctx =
"Transform0"]
    ## @return: Transform0
    ## @inputs: [frame = DataSource0]
    Transform0 = ApplyMapping.apply(frame = DataSource0, mappings = [("department",
"string",
     "department", "string"), ("name", "string", "name", "string"), ("id", "int",
"id", "int")],
      transformation_ctx = "Transform0")
    ## @type: DataSink
    ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
     "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
    ## @return: DataSink0
    ## @inputs: [frame = Transform0]
    DataSink0 = glueContext.write_dynamic_frame.from_options(frame = Transform0,
      connection_type = "s3", format = "json", connection_options = {"path":
      "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
    job.commit()
```

Opzioni di connessione per il tipo custom.spark o marketplace.spark

- `className`: stringa, obbligatorio, nome della classe del connettore.

- `secretId`: stringa, facoltativo, utilizzato per recuperare le credenziali per la connessione del connettore.
- `connectionName` – Stringa, obbligatorio, nome della connessione associata al connettore.
- Altre opzioni dipendono dall'archivio dati. Ad esempio, le opzioni di configurazione di OpenSearch iniziano con il prefisso `es`, come descritto nella documentazione di Elasticsearch per Apache Hadoop. Le connessioni Spark a Snowflake utilizzano opzioni come `sfUser` e `sfPassword`, come descritto in Using the Spark Connector nella guida Connecting to Snowflake.

Il seguente esempio di codice Python mostra come leggere da un archivio dati OpenSearch utilizzando una connessione `marketplace.spark`.

```
   import sys
   from awsglue.transforms import *
   from awsglue.utils import getResolvedOptions
   from pyspark.context import SparkContext
   from awsglue.context import GlueContext
   from awsglue.job import Job

   ## @params: [JOB_NAME]
   args = getResolvedOptions(sys.argv, ['JOB_NAME'])

   sc = SparkContext()
   glueContext = GlueContext(sc)
   spark = glueContext.spark_session
   job = Job(glueContext)
   job.init(args['JOB_NAME'], args)
   ## @type: DataSource
   ## @args: [connection_type = "marketplace.spark", connection_options =
{"path":"test",
     "es.nodes.wan.only":"true","es.nodes":"https://<AWS endpoint>",
     "connectionName":"test-spark-es","es.port":"443"}, transformation_ctx =
"DataSource0"]
   ## @return: DataSource0
   ## @inputs: []
   DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
     "marketplace.spark", connection_options = {"path":"test","es.nodes.wan.only":
     "true","es.nodes":"https://<AWS endpoint>","connectionName":
     "test-spark-es","es.port":"443"}, transformation_ctx = "DataSource0")
   ## @type: DataSink
   ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
        "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
```

```
    ## @return: DataSink0
    ## @inputs: [frame = DataSource0]
    DataSink0 = glueContext.write_dynamic_frame.from_options(frame = DataSource0,
       connection_type = "s3", format = "json", connection_options = {"path":
       "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
    job.commit()
```

Opzioni generali

Le opzioni in questa sezione sono fornite come connettore `connection_options`, ma non si applicano specificamente a tale connettore.

I seguenti parametri vengono generalmente utilizzati per la configurazione dei segnalibri. Possono applicarsi ai flussi di lavoro Amazon S3 o JDBC. Per ulteriori informazioni, consulta the section called “Utilizzo di segnalibri di processo”.

- `jobBookmarkKeys`: un array di nomi di colonna.
- `jobBookmarkKeysSortOrder`: una stringa che definisce come confrontare i valori in base all'ordinamento. Valori validi: `"asc"`, `"desc"`.
- `useS3ListImplementation`: utilizzato per gestire le prestazioni della memoria quando si elencano i contenuti dei bucket Amazon S3. Per ulteriori informazioni, consulta la pagina Optimize memory management in AWS Glue.

Opzioni del formato dati per input e output in AWS Glue per Spark

Queste pagine offrono informazioni sul supporto delle funzionalità e sui parametri di configurazione per i formati di dati supportati da AWS Glue per Spark. Consulta quanto riportato di seguito per una descrizione dell'uso e dell'applicabilità di queste informazioni.

Supporto delle funzioni per tutti i formati di dati in AWS Glue

Ogni formato dati può supportare diverse funzioni di AWS Glue. Le funzioni comuni indicate di seguito possono essere supportate o meno in base al tipo di formato. Consulta la documentazione relativa al formato dati per capire come sfruttare le nostre funzioni per soddisfare i tuoi requisiti.

Lettura	AWS Glue può riconoscere e interpretare questo formato dati senza risorse aggiuntive, ad esempio i connettori.

Scrittura	AWS Glue può scrivere dati in questo formato senza risorse aggiuntive. Puoi includere librerie di terzi nel tuo processo e utilizzare funzioni standard di Apache Spark per scrivere i dati, come con altri ambienti Spark. Per ulteriori informazioni sull'inclusione di librerie, consulta the section called "Librerie Python".
Lettura in streamin	AWS Glue può riconoscere e interpret are questo formato dati da un flusso di messaggi Apache Kafka, Amazon Managed Streaming for Apache Kafka o Amazon Kinesis. Prevediamo che i flussi presentino i dati in un formato coerente, quindi vengano letti come `DataFrames` .
Gruppo di file piccoli	AWS Glue può raggruppare i file per il lavoro in batch inviato a ogni nodo durante l'esecuzi one di trasformazioni di AWS Glue. Ciò può migliorare significativamente le prestazioni per carichi di lavoro che implicano grandi quantità di file piccoli. Per ulteriori informazioni, consulta the section called "Raggruppamento dei file di input".
Segnalib i di processc	AWS Glue può tracciare l'avanzamento delle trasformazioni che eseguono lo stesso lavoro sullo stesso set di dati tra esecuzion i di processi con segnalibri di processo. Ciò può migliorare le prestazioni per carichi di lavoro che implicano set di dati in cui occorre operare solo su nuovi dati dall'ultima esecuzion e del processo. Per ulteriori informazioni, consulta the section called "Monitoraggio dei dati elaborati mediante segnalibri di processo".

Parametri utilizzati per l'interazione con i formati di dati in AWS Glue

Determinati tipi di connessione di AWS Glue supportano più tipi di `format`, che richiedono di specificare informazioni sul formato dati con un oggetto `format_options` quando si utilizzano metodi come `GlueContext.write_dynamic_frame.from_options`.

- `s3`: per ulteriori informazioni, consulta Tipi di connessione e opzioni per ETL in AWS Glue: Parametri di connessione di S3 Puoi anche visualizzare la documentazione relativa ai metodi che facilitano questo tipo di connessione: the section called "create_dynamic_frame_from_options" e the section called "write_dynamic_frame_from_options" in Python e i metodi Scala corrispondenti the section called "getSourceWithFormat" e the section called "getSinkWithFormat".
- `kinesis`: per ulteriori informazioni, consulta Tipi di connessione e opzioni per ETL in AWS Glue: Parametri di connessione Kinesis Puoi anche visualizzare la documentazione relativa ai metodi che facilitano questo tipo di connessione: the section called "create_data_frame_from_options" e il metodo Scala corrispondente the section called "createDataFrameFromOptions".
- `kafka`: per ulteriori informazioni, consulta Tipi di connessione e opzioni per ETL in AWS Glue: Parametri di connessione Kafka Puoi anche visualizzare la documentazione relativa ai metodi che facilitano questo tipo di connessione: the section called "create_data_frame_from_options" e il metodo Scala corrispondente the section called "createDataFrameFromOptions".

Alcuni tipi di connessione non richiedono `format_options`. Ad esempio, nell'utilizzo normale, una connessione JDBC a un database relazionale recupera i dati in un formato dati tabulare coerente. Pertanto, la lettura da una connessione JDBC non richiedere `format_options`.

Alcuni metodi per la lettura e la scrittura di dati in Glue non richiedono `format_options`. Ad esempio, utilizzando `GlueContext.create_dynamic_frame.from_catalog` con crawler di AWS Glue. I crawler determinano la forma dei dati. Quando si utilizzano i crawler, un classificatore AWS Glue esamina i tuoi dati per prendere decisioni intelligenti sulla modalità di rappresentazione del formato dati. A questo punto, archivia una rappresentazione dei tuoi dati nel catalogo dati di AWS Glue, che può essere utilizzato in uno script ETL di AWS Glue per il recupero dei tuoi dati con il metodo `GlueContext.create_dynamic_frame.from_catalog`. I crawler eliminano la necessità di specificare manualmente informazioni sul formato dati.

Per i processi che accedono alle tabelle governate AWS Lake Formation, AWS Gluesupporta la lettura e la scrittura di tutti i formati supportati da tabelle governate da Lake Formation. Per l'elenco corrente dei formati supportati per tabelle governate da AWS Lake Formation, consulta Note e restrizioni per le tabelle governate nella Guida per gli sviluppatori di AWS Lake Formation.

> **Note**
>
> Per scrivere Apache Parquet, AWS Glue ETL supporta solo la scrittura su una tabella governata, specificando un'opzione per un tipo di scritture Parquet personalizzata ottimizzata per Dynamic Frames. Quando scrivi su una tabella governata con il formato `parquet`, è necessario aggiungere la chiave `useGlueParquetWriter` con un valore di `true` nei parametri della tabella.

Argomenti

Utilizzo del formato CSV in AWS Glue

AWS Glue recupera i dati dalle origini e scrive i dati sulle destinazioni archiviati e trasportati in vari formati di dati. Se i tuoi dati sono archiviati o trasportati nel formato di dati CSV, questo documento descrive le funzionalità disponibili per l'utilizzo dei tuoi dati in AWS Glue.

AWS Glue supporta solo il formato di file CSV (valori separati da virgole). Questo formato è un formato dati minimo basato su righe. I CSV spesso non sono strettamente conformi a uno standard, ma puoi fare riferimento a RFC 4180 e RFC 7111 per ulteriori informazioni.

Puoi utilizzare AWS Glue per leggere i CSV da Amazon S3 e da fonti di streaming e scrivere i CSV su Amazon S3. Puoi leggere e scrivere gli archivi `bzip` e `gzip` contenenti file CSV da S3. Puoi configurare il comportamento di compressione sul Parametri di connessione di S3 invece che nella configurazione discussa in questa pagina.

La tabella seguente mostra quali funzionalità comuni di AWS Glue supportano l'opzione del formato CSV.

Lettura	Scrittura	Lettura in streaming	Gruppo di file piccoli	Segnalibri di processo
Supportato	Supportato	Supportato	Supportato	Supportato

Esempio: lettura di file CSV o cartelle da S3

Prerequisiti: avrai bisogno dei percorsi S3 (`s3path`) nei file CSV o nelle cartelle che desideri leggere.

Configurazione : nelle opzioni della funzione, specifica `format="csv"`. In `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi configurare il modo in cui il reader interagisce con S3 in `connection_options`. Per maggiori dettagli, consulta Tipi di connessione e opzioni per ETL in AWS Glue: Parametri di connessione di S3. Puoi configurare il modo in cui il reader interpreta i file CSV nel tuo `format_options`. Per maggiori dettagli, consulta .Riferimento alla configurazione CSV.

Il seguente script ETL di AWS Glue mostra il processo di lettura di file o cartelle CSV da S3.

Forniamo un reader CSV personalizzato con ottimizzazioni delle prestazioni per i flussi di lavoro comuni attraverso la chiave di configurazione `optimizePerformance`. Per determinare se questo reader è adatto al tuo carico di lavoro, consulta the section called “Utilizzo di un reader CSV ottimizzato”.

Python

Per questo esempio, utilizza il metodo create_dynamic_frame.from_options.

```
# Example: Read CSV from S3
# For show, we handle a CSV with a header row.  Set the withHeader option.
# Consider whether optimizePerformance is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
```

```
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="csv",
    format_options={
        "withHeader": True,
        # "optimizePerformance": True,
    },
)
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .format("csv")\
    .option("header", "true")\
    .load("s3://s3path")
```

Scala

Per questo esempio, utilizza l'operazione getSourceWithFormat.

```
// Example: Read CSV from S3
// For show, we handle a CSV with a header row.  Set the withHeader option.
// Consider whether optimizePerformance is right for your workflow.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"withHeader": true}"""),
      connectionType="s3",
      format="csv",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
```

```
}
```

Puoi utilizzare DataFrames anche in uno script (`org.apache.spark.sql.DataFrame`).

```
val dataFrame = spark.read
  .option("header","true")
  .format("csv")
  .load("s3://s3path“)
```

Esempio: scrittura di file e cartelle CSV su S3

Prerequisiti: avrai bisogno di un DataFrame inizializzato (`dataFrame`) o di un DynamicFrame (`dynamicFrame`). Avrai bisogno anche del tuo percorso di output S3 previsto, `s3path`.

Configurazione: nelle opzioni della funzione, specifica `format="csv"`. In `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi configurare il modo in cui il writer interagisce con S3 in `connection_options`. Per maggiori dettagli, consulta Tipi di connessione e opzioni per ETL in AWS Glue: Parametri di connessione di S3. Puoi configurare il modo in cui l'operazione scrive il contenuto dei file in `format_options`. Per maggiori dettagli, consulta .Riferimento alla configurazione CSV. Il seguente script ETL di AWS Glue mostra il processo di scrittura di file o cartelle CSV da S3.

Python

Per questo esempio, utilizza il metodo write_dynamic_frame.from_options.

```
# Example: Write CSV to S3
# For show, customize how we write string type values.  Set quoteChar to -1 so our
 values are not quoted.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
```

```
    connection_options={"path": "s3://s3path"},
    format="csv",
    format_options={
        "quoteChar": -1,
    },
)
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
dataFrame.write\
    .format("csv")\
    .option("quote", None)\
    .mode("append")\
    .save("s3://s3path")
```

Scala

Per questo esempio, utilizza l'operazione getSinkWithFormat.

```
// Example: Write CSV to S3
// For show, customize how we write string type values. Set quoteChar to -1 so our
 values are not quoted.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
        format="csv"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

Puoi utilizzare DataFrames anche in uno script (`org.apache.spark.sql.DataFrame`).

```
dataFrame.write
    .format("csv")
    .option("quote", null)
    .mode("Append")
    .save("s3://s3path")
```

Riferimento alla configurazione CSV

Puoi utilizzare le seguenti `format_options` ovunque le librerie di AWS Glue specifichino `format="csv"`:

- `separator`: specifica il carattere delimitatore. L'impostazione predefinita è una virgola, ma è possibile specificare qualsiasi altro carattere.
  - Tipo: testo, Valore predefinito: `","`
- `escaper`: specifica un carattere da utilizzare per l'escape. Questa opzione viene utilizzata solo durante la lettura di file CSV e non durante la scrittura. Se questa opzione è abilitata, il carattere immediatamente seguente viene usato così come è, ad eccezione di un piccolo set di caratteri escape ben noti (`\n`, `\r`, `\t` e `\0`).
  - Tipo: testo, Valore predefinito: nessuno
- `quoteChar`: specifica il carattere da usare per le virgolette. Per impostazione predefinita, vengono usate le virgolette doppie. Imposta questo valore su `-1` per disattivare completamente le virgolette.
  - Tipo: testo, Valore predefinito: `'"'`
- `multiLine`: specifica se un singolo record può estendersi su più righe. Ciò può accadere quando un campo contiene un carattere di nuova riga tra virgolette. Imposta questa opzione su `True` se i registri si estendono su più righe. L'abilitazione di `multiLine` potrebbe ridurre le prestazioni perché richiede una divisione dei file più cauta durante l'analisi.
  - Tipo: booleano, Valore predefinito: `false`
- `withHeader`: specifica se trattare la prima riga come intestazione. Questa opzione può essere usata nella classe `DynamicFrameReader`.
  - Tipo: booleano, Valore predefinito: `false`
- `writeHeader`: specifica se scrivere l'intestazione nell'output. Questa opzione può essere usata nella classe `DynamicFrameWriter`.
  - Tipo: booleano, Valore predefinito: `true`
- `skipFirst`: specifica se ignorare la prima riga di dati.

  - Tipo: booleano, Valore predefinito: `false`
- `optimizePerformance`: specifica se utilizzare il reader CSV SIMD avanzato insieme ai formati di memoria colonnare basati su Apache Arrow. Disponibile solo in AWS Glue versione 3.0 o successiva.
  - Tipo: booleano, Valore predefinito: `false`
- `strictCheckForQuoting`: durante la scrittura di CSV, Glue può aggiungere virgolette ai valori che interpreta come stringhe. Questo viene fatto per evitare ambiguità in ciò che viene scritto. Per risparmiare tempo nella decisione su cosa scrivere, Glue può utilizzare le virgolette in determinate situazioni in cui in realtà non sono necessarie. L'attivazione di un controllo rigoroso eseguirà un calcolo più intensivo e ricorrerà alle virgolette solo quando strettamente necessario. Disponibile solo in AWS Glue versione 3.0 o successiva.
  - Tipo: booleano, Valore predefinito: `false`

Ottimizza le prestazioni di lettura con il reader CSV SIMD vettorizzato

AWS Glue versione 3.0 aggiunge un lettore CSV ottimizzato che può velocizzare significativamente le prestazioni complessive del lavoro rispetto ai reader CSV basati su righe.

Il reader ottimizzato:

- Usa le istruzioni SIMD della CPU per leggere dal disco
- Scrive immediatamente i record in memoria in un formato colonnare (Apache Arrow)
- Divide i record in batch

Ciò consente di risparmiare tempo di elaborazione quando i record verranno raggruppati in batch o convertiti in un formato colonnare in un secondo momento. Alcuni esempi sono quando si modificano schemi o si recuperano i dati in base alla colonna.

Per utilizzare il reader ottimizzato, imposta `"optimizePerformance"` su `true` nelle `format_options` o nella proprietà della tabella.

```
glueContext.create_dynamic_frame.from_options(
    frame = datasource1,
    connection_type = "s3",
    connection_options = {"paths": ["s3://s3path"]},
    format = "csv",
    format_options={
```

```
        "optimizePerformance": True,
        "separator": ","
        },
    transformation_ctx = "datasink2")
```

Limitazioni per il reader CSV vettorizzato

Tieni presente le seguenti limitazioni del reader CSV vettorizzato:

- Non supporta il le opzioni di formato `multiLine` e `escaper`. Utilizza l'`escaper` predefinito del carattere doppia virgoletta `'"'`. Quando queste opzioni sono impostate, AWS Glue torna automaticamente a utilizzare il reader CSV basato su righe.
- Non supporta la creazione di un DynamicFrame con ChoiceType.
- Non supporta la creazione di un DynamicFrame con record di errore.
- Non supporta la lettura di file CSV con caratteri multibyte, come i caratteri giapponesi o cinesi.

Utilizzo del formato Parquet in AWS Glue

AWS Glue recupera i dati dalle origini e scrive i dati sulle destinazioni archiviati e trasportati in vari formati di dati. Se i tuoi dati sono archiviati o trasportati nel formato di dati Parquet, questo documento descrive le funzionalità disponibili per l'utilizzo dei tuoi dati in AWS Glue.

AWS Glue supporta l'uso del formato Parquet. Questo formato è un formato dati basato su colonne orientato alle prestazioni. Per un'introduzione al formato da parte dell'autorità standard, consulta Panoramica della documentazione di Apache Parquet.

Puoi utilizzare AWS Glue per leggere i Parquet da Amazon S3 e da fonti di streaming e scrivere i Parquet su Amazon S3. Puoi leggere e scrivere gli archivi `bzip` e `gzip` contenenti file Parquet da S3. Puoi configurare il comportamento di compressione sul Parametri di connessione di S3 invece che nella configurazione discussa in questa pagina.

La tabella seguente mostra quali funzionalità comuni di AWS Glue supportano l'opzione del formato Parquet.

Lettura	Scrittura	Lettura in streaming	Gruppo di file piccoli	Segnalibri di processo
Supportato	Supportato	Supportato	Non supportato.	Supportato*

* Supportato in AWS Glue versione 1.0 e successive

Esempio: lettura di file Parquet o cartelle da S3

Prerequisiti: avrai bisogno dei percorsi S3 (`s3path`) nei file Parquet o nelle cartelle che desideri leggere.

Configurazione: nelle opzioni della funzione, specifica `format="parquet"`. Nelle tue `connection_options`, utilizza la chiave `paths` per specificare `s3path`.

Puoi configurare il modo in cui il reader interagisce con S3 in `connection_options`. Per maggiori dettagli, consulta Tipi di connessione e opzioni per ETL in AWS Glue: Parametri di connessione di S3.

Puoi configurare il modo in cui il reader interpreta i file Parquet nel tuo `format_options`. Per maggiori dettagli, consulta Riferimento alla configurazione Parquet.

Il seguente script ETL di AWS Glue mostra il processo di lettura di file o cartelle Parquet da S3:

Python

Per questo esempio, utilizza il metodo create_dynamic_frame.from_options.

```
# Example: Read Parquet from S3

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type = "s3",
    connection_options = {"paths": ["s3://s3path/"]},
    format = "parquet"
)
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read.parquet("s3://s3path/")
```

Scala

Per questo esempio, utilizza il metodo getSourceWithFormat.

```
// Example: Read Parquet from S3

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="parquet",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
}
```

Puoi utilizzare DataFrames anche in uno script (`org.apache.spark.sql.DataFrame`).

```
spark.read.parquet("s3://s3path/")
```

Esempio: scrittura di file e cartelle Parquet su S3

Prerequisiti: avrai bisogno di un DataFrame inizializzato (`dataFrame`) o di un DynamicFrame (`dynamicFrame`). Avrai bisogno anche del tuo percorso di output S3 previsto, `s3path`.

Configurazione: nelle opzioni della funzione, specifica `format="parquet"`. In `connection_options`, utilizza la chiave `paths` per specificare `s3path`.

Puoi modificare ulteriormente il modo in cui il writer interagisce con S3 nelle `connection_options`. Per maggiori dettagli, consulta Tipi di connessione e opzioni per ETL in AWS Glue: Parametri di connessione di S3. Puoi configurare il modo in cui l'operazione scrive il contenuto dei file in `format_options`. Per maggiori dettagli, consulta .Riferimento alla configurazione Parquet.

Il seguente script ETL di AWS Glue mostra il processo di scrittura di file o cartelle Parquet da S3.

Forniamo un writer Parquet personalizzato con ottimizzazioni delle prestazioni per DynamicFrame, tramite la chiave di configurazione `useGlueParquetWriter`. Per determinare se questo writer è adatto al tuo carico di lavoro, consulta Scrittore Glue Parquet.

Python

Per questo esempio, utilizza il metodo write_dynamic_frame.from_options.

```
# Example: Write Parquet to S3
# Consider whether useGlueParquetWriter is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="parquet",
    connection_options={
        "path": "s3://s3path",
    },
    format_options={
        # "useGlueParquetWriter": True,
    },
)
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
df.write.parquet("s3://s3path/")
```

Scala

Per questo esempio, utilizza l'operazione getSinkWithFormat.

```
// Example: Write Parquet to S3
// Consider whether useGlueParquetWriter is right for your workflow.

import com.amazonaws.services.glue.util.JsonOptions
```

```
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
        format="parquet"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

Puoi utilizzare DataFrames anche in uno script (`org.apache.spark.sql.DataFrame`).

```
df.write.parquet("s3://s3path/")
```

Riferimento alla configurazione Parquet

Puoi utilizzare le seguenti `format_options` ovunque le librerie di AWS Glue specifichino `format="parquet"`:

- `useGlueParquetWriter`: specifica l'utilizzo di un writer Parquet personalizzato con ottimizzazioni delle prestazioni per i flussi di lavoro di DynamicFrame. Per informazioni dettagliate sull'utilizzo, consulta Writer Glue Parquet.
  - Tipo: booleano, Valore predefinito: `false`
- `compression`: specifica il codec di compressione utilizzato. I valori sono pienamente compatibili con `org.apache.parquet.hadoop.metadata.CompressionCodecName`.
  - Tipo: testo enumerato, Valore predefinito: `"snappy"`
  - Valori: `"uncompressed"`, `"snappy"`, `"gzip"` e `"lzo"`
- `blockSize`: specifica la dimensione in byte di un gruppo di righe memorizzate nel buffer in memoria. Utilizzi questo valore per ottimizzare le prestazioni. Le dimensioni dovrebbero dividersi esattamente in un numero di megabyte.
  - Tipo: numerico, Valore predefinito: `134217728`
  - Il valore predefinito è 128 MB.

- `pageSize`: specifica le dimensioni in byte di una pagina. Utilizzi questo valore per ottimizzare le prestazioni. Una pagina è l'unità più piccola che deve essere letta interamente per accedere a un singolo record.
  - Tipo: numerico, Valore predefinito: `1048576`
  - Il valore predefinito è 1 MB.

> **Note**
>
> Inoltre, tutte le opzioni accettate dal codice SparkSQL sottostante possono essere passate tramite il parametro mappa `connection_options`. Ad esempio, puoi impostare una configurazione Spark come mergeSchema per il reader Spark di AWS Glue per unire lo schema di tutti i file.

Ottimizzazione delle prestazioni di scrittura con il writer Parquet di AWS Glue

> **Note**
>
> Il writer Parquet di AWS Glue è stato storicamente accessibile tramite il tipo di formato `glueparquet`. Questo schema di accesso non è più raccomandato. Utilizza invece il tipo `parquet` con `useGlueParquetWriter` abilitato.

Il writer Parquet di AWS Glue offre miglioramenti delle prestazioni che consentono scritture di file Parquet più veloci. Il writer tradizionale calcola uno schema prima della scrittura. Il formato Parquet non memorizza lo schema in un modo recuperabile rapidamente, quindi questa operazione potrebbe richiedere del tempo. Con il writer Parquet di AWS Glue non è richiesto uno schema pre-calcolato. Quando arrivano i dati, il writer calcola e modifica lo schema in modo dinamico.

Quando specifichi `useGlueParquetWriter`, tieni presente le seguenti limitazioni:

- Il writer supporta solo l'evoluzione dello schema, come l'aggiunta o la rimozione di colonne, ma non la modifica dei tipi di colonna, ad esempio con `ResolveChoice`.
- Il writer non è in grado di memorizzare un DataFrame vuoto, ad esempio per scrivere un file solo su schema. Durante l'integrazione con Catalogo dati AWS Glue tramite l'impostazione `enableUpdateCatalog=True`, il tentativo di scrivere un DataFrame vuoto non aggiornerà il Catalogo dati. Il nome di una tabella nel Catalogo dati.

Se la trasformazione non richiede queste limitazioni, l'attivazione del writer Parquet di AWS Glue dovrebbe aumentare le prestazioni.

Utilizzo del formato XML in AWS Glue

AWS Glue recupera i dati dalle origini e scrive i dati sulle destinazioni archiviati e trasportati in vari formati di dati. Se i tuoi dati sono archiviati o trasportati nel formato di dati XML, questo documento descrive le funzionalità disponibili per l'utilizzo dei tuoi dati in AWS Glue.

AWS Glue supporta l'uso del formato XML. Questo formato rappresenta strutture dati altamente configurabili e rigidamente definite che non sono basate su righe o colonne. XML è altamente standardizzato. Per un'introduzione al formato da parte dell'autorità di standard, consulta Informazioni essenziali su XML.

Puoi utilizzare AWS Glue per leggere file XML da Amazon S3 e gli archivi `bzip` e `gzip` contenenti file XML. Puoi configurare il comportamento di compressione sul Parametri di connessione di S3 invece che nella configurazione discussa in questa pagina.

La tabella seguente mostra quali funzionalità comuni di AWS Glue supportano l'opzione del formato XML.

Lettura	Scrittura	Lettura in streaming	Gruppo di file piccoli	Segnalibri di processo
Supportato	Non supportato.	Non supportato.	Supportato	Supportato

Esempio: lettura di XML da S3

Il reader XML assume un nome di tag XML. Esamina gli elementi con quel tag all'interno del suo input per dedurre uno schema e popola un DynamicFrame con i valori corrispondenti. La XML di AWS Glue si comporta in modo simile all'origine dei dati XML per Apache Spark. Potresti essere in grado di ottenere informazioni sul comportamento di base confrontando questo reader con la documentazione di quel progetto.

Prerequisiti: avrai bisogno dei percorsi S3 (`s3path`) nei file XML o nelle cartelle che desideri leggere e di alcune informazioni sul file XML. Avrai bisogno anche del tag per l'elemento XML che desideri leggere, `xmlTag`.

Configurazione: nelle opzioni della funzione, specifica `format="xml"`. In `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi configurare il modo in cui il reader interagisce

con S3 in `connection_options`. Per maggiori dettagli, consulta Tipi di connessione e opzioni per ETL in AWS Glue: Parametri di connessione di S3. In `format_options`, utilizza la chiave `rowTag` per specificare `xmlTag`. Puoi configurare il modo in cui il reader interpreta i file XML nel tuo `format_options`. Per maggiori dettagli, consulta Riferimento alla configurazione XML.

Il seguente script ETL di AWS Glue mostra il processo di lettura di file o cartelle XML da S3.

Python

Per questo esempio, utilizza il metodo create_dynamic_frame.from_options.

```
# Example: Read XML from S3
# Set the rowTag option to configure the reader.

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="xml",
    format_options={"rowTag": "xmlTag"},
)
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .format("xml")\
    .option("rowTag", "xmlTag")\
    .load("s3://s3path")
```

Scala

Per questo esempio, utilizza l'operazione getSourceWithFormat.

```
// Example: Read XML from S3
// Set the rowTag option to configure the reader.
```

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkSession

val glueContext = new GlueContext(SparkContext.getOrCreate())
val sparkSession: SparkSession = glueContext.getSparkSession

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"rowTag": "xmlTag"}"""),
      connectionType="s3",
      format="xml",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
}
```

Puoi utilizzare DataFrames anche in uno script (`org.apache.spark.sql.DataFrame`).

```
val dataFrame = spark.read
  .option("rowTag", "xmlTag")
  .format("xml")
  .load("s3://s3path“)
```

Riferimento alla configurazione XML

Puoi utilizzare le seguenti `format_options` ovunque le librerie di AWS Glue specifichino `format="xml"`:

- `rowTag`: specifica il tag XML nel file da trattare come riga. I tag di riga non possono essere con chiusura automatica.
  - Tipo: testo, obbligatorio
- `encoding`: specifica la codifica dei caratteri. Può essere il nome o l'alias di un Charset supportato dal nostro ambiente di runtime. Non forniamo garanzie specifiche sul supporto della codifica, ma le codifiche principali dovrebbero funzionare.
  - Tipo: testo, Valore predefinito: `"UTF-8"`
- `excludeAttribute`: specifica se escludere o meno gli attributi negli elementi.
  - Tipo: booleano, Valore predefinito: `false`

- `treatEmptyValuesAsNulls`: specifica se trattare uno spazio vuoto come valore null.
  - Tipo: booleano, Valore predefinito: `false`
- `attributePrefix`: un prefisso per gli attributi, per differenziarli dal testo degli elementi figlio. Questo prefisso viene usato per i nomi di campi.
  - Tipo: testo, Valore predefinito: `"_"`
- `valueTag`: il tag usato per un valore quando ci sono attributi nell'elemento che non hanno elementi figlio.
  - Tipo: testo, Valore predefinito: `"_VALUE"`
- `ignoreSurroundingSpaces`: specifica se lo spazio vuoto intorno ai valori deve essere ignorato.
  - Tipo: booleano, Valore predefinito: `false`
- `withSchema`: contiene lo schema previsto, in situazioni in cui si desidera sovrascrivere lo schema dedotto. Se non utilizzi questa opzione, AWS Glue deduce lo schema dai dati XML.
  - Tipo: testo, Valore predefinito: non applicabile
  - Il valore deve essere un oggetto JSON che rappresenta un `StructType`.

Specifica manuale dello schema XML

Esempio di schema XML manuale

Questo è un esempio dell'utilizzo di `withSchema` per specificare lo schema per i dati XML.

```
from awsglue.gluetypes import *

schema = StructType([
  Field("id", IntegerType()),
  Field("name", StringType()),
  Field("nested", StructType([
    Field("x", IntegerType()),
    Field("y", StringType()),
    Field("z", ChoiceType([IntegerType(), StringType()]))
  ]))
])

datasource0 = create_dynamic_frame_from_options(
    connection_type,
    connection_options={"paths": ["s3://xml_bucket/someprefix"]},
    format="xml",
```

```
    format_options={"withSchema": json.dumps(schema.jsonValue())},
    transformation_ctx = ""
)
```

Utilizzo del formato Avro in AWS Glue

AWS Glue recupera i dati dalle origini e scrive i dati sulle destinazioni archiviati e trasportati in vari formati di dati. Se i tuoi dati sono archiviati o trasportati nel formato dati Avro, questo documento descrive le funzioni disponibili per l'utilizzo dei tuoi dati in AWS Glue.

AWS Glue supporta l'uso del formato Avro. Questo formato è un formato dati basato su righe orientato alle prestazioni. Per un'introduzione al formato da parte dell'autorità degli standard, consulta la Documentazione di Apache Avro 1.8.2.

Puoi utilizzare AWS Glue per leggere file Avro da Amazon S3 e da origini di streaming, nonché scrivere file Avro in Amazon S3. Puoi leggere e scrivere archivi `bzip2` e `gzip` contenenti file Avro da S3. Inoltre, è possibile scrivere archivi `deflate`, `snappy` e `xz` contenenti file Avro. Puoi configurare il comportamento di compressione sul Parametri di connessione di S3 invece che nella configurazione discussa in questa pagina.

La tabella seguente mostra le operazioni comuni di AWS Glue che supportano l'opzione del formato Avro.

Lettura	Scrittura	Lettura in streaming	Gruppo di file piccoli	Segnalibri di processo
Supportato	Supportato	Supportato*	Non supportato.	Supportato

* Supportato con restrizioni. Per ulteriori informazioni, consulta the section called “Note e restrizioni per le origini di streaming Avro”.

Esempio: lettura di cartelle o file Avro da S3

Prerequisiti: occorrono i percorsi S3 (`s3path`) nelle cartelle o nei file Avro da leggere.

Configurazione: nelle opzioni della funzione, specifica `format="avro"`. In `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi configurare il modo in cui il reader interagisce con S3 in `connection_options`. Per i dettagli, consulta le Opzioni del formato dati per input e

output ETL in AWS Glue: the section called “Parametri di connessione di S3”. Puoi configurare la modalità con cui il reader interpreta i file Avro CSV in `format_options`. Per i dettagli, consulta Documentazione di riferimento della configurazione Avro.

Lo script ETL di AWS Glue riportato di seguito mostra il processo di lettura di cartelle o file Avro da S3:

Python

Per questo esempio, utilizza il metodo create_dynamic_frame.from_options.

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="avro"
)
```

Scala

Per questo esempio, utilizza l'operazione getSourceWithFormat.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="avro",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
```

Esempio: scrittura di file e cartelle Avro in S3

Prerequisiti: avrai bisogno di un DataFrame inizializzato (`dataFrame`) o di un DynamicFrame (`dynamicFrame`). Avrai bisogno anche del tuo percorso di output S3 previsto, `s3path`.

Configurazione: nelle opzioni della funzione, specifica `format="avro"`. Nelle tue `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi modificare ulteriormente il modo in cui il writer interagisce con S3 nelle `connection_options`. Per i dettagli, consulta le Opzioni del formato dati per input e output ETL in AWS Glue: the section called "Parametri di connessione di S3". Puoi modificare la modalità con cui il writer interpreta i file Avro in `format_options`. Per i dettagli, consulta Documentazione di riferimento della configurazione Avro.

Lo script ETL di AWS Glue riportato di seguito mostra il processo di scrittura di cartelle o file Avro in S3.

Python

Per questo esempio, utilizza il metodo write_dynamic_frame.from_options.

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="avro",
    connection_options={
        "path": "s3://s3path"
    }
)
```

Scala

Per questo esempio, utilizza l'operazione getSinkWithFormat.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext
```

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
      connectionType="s3",
      options=JsonOptions("""{"path": "s3://s3path"}"""),
      format="avro"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

Documentazione di riferimento della configurazione Avro

Puoi utilizzare i seguenti valori `format_options` dove le librerie di AWS Glue specificano `format="avro"`:

- `version`: specifica la versione del formato di lettura/scrittura Apache Avro da supportare. Il valore predefinito è 1.7. Puoi specificare `format_options={"version": “1.8”}` per abilitare la lettura e la scrittura del tipo logico Avro. Per ulteriori informazioni, consulta Specifiche di Apache Avro 1.7.7 e Specifiche di Apache Avro 1.8.2.

  Il connettore Apache Avro 1.8 supporta le seguenti conversioni di tipo logico:

Per le operazioni di lettura: questa tabella mostra la conversione tra il tipo di dati Avro (tipo logico e tipo primitivo Avro) e il tipo di dati AWS Glue `DynamicFrame` per le operazioni di lettura Avro 1.7 e 1.8.

Tipo di dati Avro: Tipo logico	Tipo di dati Avro: Tipi primitivi .NET	Tipo di dati GluedyNamicFrame: Avro Reader 1.7	Tipo di dati GluedyNamicFrame: Avro Reader 1.8
Decimale	byte	BINARY	Decimale
Decimale	fisso	BINARY	Decimale
Data	int	INT	Data

Tipo di dati Avro: Tipo logico	Tipo di dati Avro: Tipi primitivi .NET	Tipo di dati GluedyNamicFrame: Avro Reader 1.7	Tipo di dati GluedyNamicFrame: Avro Reader 1.8
Tempo (millisecondi)	int	INT	INT
Tempo (microsecondi)	Long	LONG	LONG
Timestamp (millisec ondi)	Long	LONG	Time stamp
Timestamp (microsec ondi)	Long	LONG	LONG
Durata (non un tipo logico)	fisso di 12	BINARY	BINARY

Per le operazioni di scrittura: questa tabella mostra la conversione tra il tipo di dati AWS Glue `DynamicFrame` e il tipo di dati Avro per le operazioni di scrittura Avro 1.7 e 1.8.

Tipo di dati AWS Glue **`DynamicFrame`**	Tipo di dati Avro: Avro Writer 1.7	Tipo di dati Avro: Avro Writer 1.8
Decimale	Stringa	decimal
Data	Stringa	date
Timestamp	Stringa	timestamp-micros

Supporto Avro di Spark DataFrame

Per utilizzare Avro dall'API di Spark DataFrame, devi installare il plugin Spark Avro per la versione di Spark corrispondente. La versione di Spark disponibile nel processo è determinata dalla versione di AWS Glue. Per ulteriori informazioni sulle versioni di Spark, consulta the section called “Versioni AWS Glue”. Questo plugin è gestito da Apache; non forniamo garanzie specifiche sul supporto.

In AWS Glue 2.0: utilizza la versione 2.4.3 del plugin Spark Avro. Puoi trovare questo JAR su Maven Central, consulta org.apache.spark:spark-avro_2.12:2.4.3.

In AWS Glue 3.0: utilizza la versione 3.1.1 del plugin Spark Avro. Puoi trovare questo JAR su Maven Central, consulta org.apache.spark:spark-avro_2.12:3.1.1.

Per includere JAR aggiuntivi in un processo ETL di AWS Glue, usa il parametro di processo `--extra-jars`. Per ulteriori informazioni sui parametri di processo, consulta the section called "Parametri del processo". Puoi configurare questo parametro anche nella AWS Management Console.

Utilizzo del formato grokLog in AWS Glue

AWS Glue recupera i dati dalle origini e scrive i dati sulle destinazioni archiviati e trasportati in vari formati di dati. Se i tuoi dati vengono archiviati o trasportati in un formato di testo semplice poco strutturato, questo documento descrive le funzioni disponibili per l'utilizzo dei tuoi dati in AWS Glue tramite modelli Grok.

AWS Glue supporta l'uso di modelli Grok. I modelli Grok sono simili a gruppi di acquisizione di espressioni regolari. Riconoscono modelli di sequenze di caratteri in un file di testo semplice e forniscono loro un tipo e uno scopo. In AWS Glue, il loro scopo principale è leggere i registri. Per un'introduzione a Grok da parte degli autori, consulta Documentazione di riferimento di Logstash: plugin per filtri Grok.

Lettura	Scrittura	Lettura in streaming	Gruppo di file piccoli	Segnalibri di processo
Supportato	Non applicabile	Supportato	Supportato	Non supportato.

Documentazione di riferimento sulla configurazione grokLog

Puoi usare i valori di `format_options` seguenti con `format="grokLog"`:

- `logFormat`: specifica il pattern grok corrispondente al formato del log.
- `customPatterns`: specifica ulteriori pattern Grok usati.
- `MISSING`: specifica il segnale da usare per identificare i valori mancanti. Il valore predefinito è `'-'`.
- `LineCount`: specifica il numero di righe in ogni record di log. Il valore predefinito è `'1'` e attualmente sono supportati solo record a riga singola.

- `StrictMode`: valore booleano che specifica se la modalità strict è abilitata. In modalità strict, il lettore non esegue la conversione o il ripristino automatico dei tipi. Il valore di default è `"false"`.

Utilizzo del formato Ion in AWS Glue

AWS Glue recupera i dati dalle origini e scrive i dati sulle destinazioni archiviati e trasportati in vari formati di dati. Se i tuoi dati vengono archiviati o trasportati nel formato di dati Ion, questo documento descrive le funzioni disponibili per l'utilizzo dei tuoi dati in AWS Glue.

AWS Glue supporta l'uso del formato Ion. Questo formato rappresenta strutture di dati (che non sono basate su righe o colonne) in rappresentazioni binarie e di testo semplice intercambiabili. Per un'introduzione al formato da parte degli autori, consulta Amazon Ion. Per ulteriori informazioni consulta la specifica Amazon Ion.

Puoi utilizzare AWS Glue per leggere file Ion da Amazon S3. Puoi leggere e scrivere archivi `bzip` e `gzip` contenenti file Ion da S3. Puoi configurare il comportamento di compressione sul Parametri di connessione di S3 invece che nella configurazione discussa in questa pagina.

La tabella seguente mostra le operazioni comuni di AWS Glue che supportano l'opzione del formato Ion.

Lettura	Scrittura	Lettura in streaming	Gruppo di file piccoli	Segnalibri di processo
Supportato	Non supportato.	Non supportato.	Supportato	Non supportato.

Esempio: lettura di cartelle e file Ion da S3

Prerequisiti: occorreranno i percorsi S3 (`s3path`) nelle cartelle o nei file Ion da leggere.

Configurazione: nelle opzioni della funzione, specifica `format="json"`. Nelle tue `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi configurare il modo in cui il reader interagisce con S3 in `connection_options`. Per maggiori dettagli, consulta Tipi di connessione e opzioni per ETL in AWS Glue: the section called “Parametri di connessione di S3”.

Il seguente script ETL di AWS Glue mostra il processo di lettura di cartelle o file Ion da S3:

Python

Per questo esempio, utilizza il metodo create_dynamic_frame.from_options.

```
# Example: Read ION from S3

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="ion"
)
```

Scala

Per questo esempio, utilizza l'operazione getSourceWithFormat.

```
// Example: Read ION from S3

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="ion",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
}
```

Documentazione di riferimento della configurazione Ion

Non ci sono valori di `format_options` per `format="ion"`.

Utilizzo del formato JSON in AWS Glue

AWS Glue recupera i dati dalle origini e scrive i dati sulle destinazioni archiviati e trasportati in vari formati di dati. Se i tuoi dati vengono archiviati o trasportati nel formato dati JSON, questo documento descrive le funzionalità disponibili per l'utilizzo dei dati in AWS Glue.

AWS Glue supporta l'uso del formato JSON. Questo formato rappresenta strutture di dati con forma coerente ma contenuti flessibili, che non sono basate su righe o colonne. JSON è definito tramite standard paralleli emessi da diverse autorità, una delle quali è ECMA-404. Per un'introduzione al formato da una fonte di riferimento comune, consulta Introduzione a JSON.

Puoi utilizzare AWS Glue per leggere file JSON da Amazon S3, nonché file KJSON compressi `bzip` e `gzip`. Puoi configurare il comportamento di compressione sul Parametri di connessione di S3 invece che nella configurazione discussa in questa pagina.

Lettura	Scrittura	Lettura in streaming	Gruppo di file piccoli	Segnalibri di processo
Supportato	Supportato	Supportato	Supportato	Supportato

Esempio: lettura di cartelle o file JSON da S3

Prerequisiti: occorrono i percorsi S3 (`s3path`) nelle cartelle o nei file JSON da leggere.

Configurazione: nelle opzioni della funzione, specifica `format="json"`. Nelle tue `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi modificare ulteriormente la modalità con cui l'operazione di lettura attraversa s3 nelle opzioni di connessione; consulta the section called “Parametri di connessione di S3” per dettagli. Puoi configurare la modalità con cui il reader interpreta i file JSON in `format_options`. Per i dettagli, consulta la Documentazione di riferimento della configurazione JSON.

Lo script ETL di AWS Glue riportato di seguito mostra il processo di lettura di cartelle o file JSON da S3:

Python

Per questo esempio, utilizza il metodo create_dynamic_frame.from_options.

```
# Example: Read JSON from S3
```

```
# For show, we handle a nested JSON file that we can limit with the JsonPath
 parameter
# For show, we also handle a JSON where a single entry spans multiple lines
# Consider whether optimizePerformance is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="json",
    format_options={
        "jsonPath": "$.id",
        "multiline": True,
        # "optimizePerformance": True, -> not compatible with jsonPath, multiline
    }
)
```

Puoi anche usare DataFrames in uno script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .option("multiline", "true")\
    .json("s3://s3path")
```

Scala

Per questo esempio, utilizzate l'operazione getSourceWithFormat.

```
// Example: Read JSON from S3
// For show, we handle a nested JSON file that we can limit with the JsonPath
 parameter
// For show, we also handle a JSON where a single entry spans multiple lines
// Consider whether optimizePerformance is right for your workflow.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext
```

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"jsonPath": "$.id", "multiline": true,
 "optimizePerformance":false}"""),
      connectionType="s3",
      format="json",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
}
```

È inoltre possibile utilizzare DataFrames in uno script (`pyspark.sql.DataFrame`).

```
val dataFrame = spark.read
    .option("multiline", "true")
    .json("s3://s3path")
```

Esempio: scrittura di file e cartelle JSON su S3

Prerequisiti: è necessario un DataFrame (`dataFrame`) o DynamicFrame (`dynamicFrame`) inizializzato. Avrai bisogno anche del tuo percorso di output S3 previsto, `s3path`.

Configurazione: nelle opzioni della funzione, specifica `format="json"`. In `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi modificare ulteriormente il modo in cui il writer interagisce con S3 nelle `connection_options`. Per i dettagli, consulta le Opzioni del formato dati per input e output ETL in AWS Glue: the section called “Parametri di connessione di S3”. Puoi configurare la modalità con cui il writer interpreta i file JSON in `format_options`. Per i dettagli, consulta la Documentazione di riferimento della configurazione JSON.

Lo script ETL di AWS Glue riportato di seguito mostra il processo di scrittura di cartelle o file JSON da S3:

Python

Per questo esempio, utilizza il metodo write_dynamic_frame.from_options.

```
# Example: Write JSON to S3
```

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    connection_options={"path": "s3://s3path"},
    format="json"
)
```

Puoi anche usare DataFrames in uno script ()`pyspark.sql.DataFrame`.

```
df.write.json("s3://s3path/")
```

Scala

Per questo esempio, utilizzate il metodo getSinkWithFormat.

```
// Example: Write JSON to S3

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
        format="json"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

È inoltre possibile utilizzare DataFrames in uno script (`pyspark.sql.DataFrame`).

```
df.write.json("s3://s3path")
```

Documentazione di riferimento della configurazione JSON

Puoi usare i valori di `format_options` seguenti con `format="json"`:

- `jsonPath`— Un'JsonPathespressione che identifica un oggetto da leggere nei record. È particolarmente utile quando un file contiene registri annidati in una matrice esterna. Ad esempio, l' JsonPath espressione seguente si rivolge al `id` campo di un oggetto JSON.

  ```
  format="json", format_options={"jsonPath": "$.id"}
  ```

- `multiLine`: un valore booleano che specifica se un singolo registro può estendersi su più righe. Ciò può accadere quando un campo contiene un carattere di nuova riga tra virgolette. Imposta questa opzione su `"true"` se i registri si estendono su più righe. Il valore di default è `"false"`, che consente una divisione dei file più netta durante l'analisi.
- `optimizePerformance`: valore booleano che specifica se utilizzare il lettore JSON SIMD avanzato insieme ai formati di memoria colonnare basati su Apache Arrow. Disponibile solo in AWS Glue 3.0. Non compatibile con `multiLine` o `jsonPath`. Fornire una di queste opzioni istruirà AWS Glue per tornare al lettore standard.
- `withSchema`: un valore di stringa che specifica uno schema di tabella nel formato descritto in the section called “Specifica dello schema XML”. Utilizzato solo con `optimizePerformance` durante la lettura da connessioni non di catalogo.

Utilizzo del lettore JSON SIMD vettorizzato con formato colonnare Apache Arrow

La versione AWS Glue 3.0 aggiunge un lettore vettorizzato per i dati JSON. Funziona 2 volte più velocemente in determinate condizioni, rispetto al lettore standard. Questo lettore presenta alcune limitazioni di cui gli utenti dovrebbero essere consapevoli prima dell'uso, documentate in questa sezione.

Per utilizzare il lettore ottimizzato, imposta `"optimizePerformance"` a True nella `format_options` o nella proprietà della tabella. Dovrai anche fornire `withSchema` a meno che non venga letto dal catalogo. `withSchema` prevede un input come descritto nel the section called “Specifica dello schema XML”

```
// Read from S3 data source
glueContext.create_dynamic_frame.from_options(
    connection_type = "s3",
    connection_options = {"paths": ["s3://s3path"]},
    format = "json",
    format_options={
        "optimizePerformance": True,
        "withSchema": SchemaString
        })

// Read from catalog table
glueContext.create_dynamic_frame.from_catalog(
    database = database,
    table_name = table,
    additional_options = {
    // The vectorized reader for JSON can read your schema from a catalog table
 property.
        "optimizePerformance": True,
        })
```

Per ulteriori informazioni sull'edificio a *SchemaString*nella libreria AWS Glue, vederethe section called “Tipi”.

Limitazioni per il lettore CSV vettorizzato

Nota i seguenti limiti:

- Gli elementi JSON con oggetti nidificati o valori di array non sono supportati. Se previsto, AWS Glue tornerà al lettore standard.
- È necessario fornire uno schema, dal catalogo o con `withSchema`.
- Non compatibile con `multiLine` o `jsonPath`. Fornire una di queste opzioni istruirà AWS Glue per tornare al lettore standard.
- Fornire registri di input che non corrispondono allo schema di input provocherà il fallimento del lettore.
- I registri di errori non verranno creati.
- Non supporta file JSON con caratteri multibyte (come caratteri giapponesi o cinesi).

Utilizzo del formato ORC in AWS Glue

AWS Glue recupera i dati dalle origini e scrive i dati sulle destinazioni archiviati e trasportati in vari formati di dati. Se i tuoi dati vengono archiviati o trasportati nel formato dati ORC, questo documento descrive le funzioni disponibili per l'utilizzo dei tuoi dati in AWS Glue.

AWS Glue supporta l'uso del formato ORC. Questo formato è un formato dati basato su colonne orientato alle prestazioni. Per un'introduzione al formato da parte dell'autorità degli standard, consulta Apache Orc.

Puoi utilizzare AWS Glue per leggere i file ORC da Amazon S3 e da origini di streaming, nonché scrivere file ORC in Amazon S3. Puoi leggere e scrivere archivi `bzip` e `gzip` contenenti file ORC da S3. Puoi configurare il comportamento di compressione sul Parametri di connessione di S3 invece che nella configurazione discussa in questa pagina.

La tabella seguente mostra le operazioni comuni di AWS Glue che supportano l'opzione del formato ORC.

Lettura	Scrittura	Lettura in streaming	Gruppo di file piccoli	Segnalibri di processo
Supportato	Supportato	Supportato	Non supportato.	Supportato*

*Supportato in AWS Glue versione 1.0 e successive

Esempio: lettura di cartelle o file ORC da S3

Prerequisiti: occorreranno i percorsi S3 (`s3path`) nelle cartelle o nei file ORC da leggere.

Configurazione: nelle opzioni della funzione, specifica `format="orc"`. Nelle tue `connection_options`, utilizza la chiave `paths` per specificare `s3path`. Puoi configurare il modo in cui il reader interagisce con S3 in `connection_options`. Per maggiori dettagli, consulta Tipi di connessione e opzioni per ETL in AWS Glue: the section called “Parametri di connessione di S3”.

Il seguente script ETL di AWS Glue mostra il processo di lettura di cartelle o file ORC da S3:

Python

Per questo esempio, utilizza il metodo create_dynamic_frame.from_options.

```
from pyspark.context import SparkContext
```

```
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="orc"
)
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .orc("s3://s3path")
```

Scala

Per questo esempio, utilizza l'operazione getSourceWithFormat.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="orc",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
}
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
val dataFrame = spark.read
    .orc("s3://s3path")
```

Esempio: scrittura di cartelle e file ORC in S3

Prerequisiti: avrai bisogno di un DataFrame inizializzato (`dataFrame`) o di un DynamicFrame (`dynamicFrame`). Avrai bisogno anche del tuo percorso di output S3 previsto, `s3path`.

Configurazione: nelle opzioni della funzione, specifica `format="orc"`. Nelle opzioni di connessione, usa la chiave `paths` per specificare `s3path`. Puoi modificare ulteriormente il modo in cui il writer interagisce con S3 nelle `connection_options`. Per i dettagli, consulta le Opzioni del formato dati per input e output ETL in AWS Glue: the section called “Parametri di connessione di S3”. L'esempio di codice seguente mostra il processo:

Python

Per questo esempio, utilizza il metodo write_dynamic_frame.from_options.

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="orc",
    connection_options={
        "path": "s3://s3path"
    }
)
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
df.write.orc("s3://s3path/")
```

Scala

Per questo esempio, utilizza l'operazione getSinkWithFormat.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext
```

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
      connectionType="s3",
      options=JsonOptions("""{"path": "s3://s3path"}"""),
      format="orc"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

Puoi utilizzare DataFrames anche in uno script (`pyspark.sql.DataFrame`).

```
df.write.orc("s3://s3path/")
```

Riferimento alla configurazione XML

Non ci sono valori di `format_options` per `format="orc"`. Tutte le opzioni accettate dal codice SparkSQL sottostante possono tuttavia essere passate tramite il parametro mappa `connection_options`.

Utilizzo di framework data lake con processi ETL di AWS Glue

I framework data lake open source semplificano l'elaborazione incrementale dei dati per i file archiviati in data lake basati su Amazon S3. AWS Glue 3.0 e versioni successive supportano i seguenti framework data lake open source:

- Apache Hudi
- Linux Foundation Delta Lake
- Apache Iceberg

Forniamo supporto nativo per questi framework in modo che sia possibile leggere e scrivere i dati archiviati in Amazon S3 in modo coerente dal punto di vista transazionale. Non è necessario installare un connettore separato o completare passaggi di configurazione aggiuntivi per utilizzare questi framework nei processi ETL di AWS Glue.

Quando gestisci i set di dati tramite AWS Glue Data Catalog, puoi utilizzare i metodi AWS Glue per leggere e scrivere tabelle di data lake con Spark DataFrames. È possibile leggere e scrivere dati Amazon S3 anche utilizzando l'API Spark DataFrame.

Questo video illustra le basi del funzionamento di Apache Hudi, Apache Iceberg e Delta Lake. Scoprirai come inserire, aggiornare ed eliminare i dati nel tuo data lake e come funziona ciascuno di questi framework.

Argomenti

Limitazioni

Considera le seguenti limitazioni prima di utilizzare i framework data lake con AWS Glue.

- I seguenti AWS Glue `GlueContext` metodi DynamicFrame non supportano la lettura e la scrittura di tabelle del framework Data Lake. Utilizza invece i `GlueContext` metodi DataFrame per l' DataFrame API Spark.
  - I seguenti `GlueContext` metodi per non DynamicFrame sono supportati con il controllo delle autorizzazioni di Lake Formation:
    - `create_dynamic_frame.from_catalog`
    - `write_dynamic_frame.from_catalog`
    - `getDynamicFrame`
    - `writeDynamicFrame`
  - I seguenti `GlueContext` metodi DataFrame sono supportati con il controllo dei permessi di Lake Formation:
    - `create_data_frame.from_catalog`
    - `write_data_frame.from_catalog`
    - `getDataFrame`
    - `writeDataFrame`
- Il raggruppamento di file di piccole dimensioni non è supportato.

- I segnalibri dei processi non sono supportati.
- Apache Hudi 0.10.1 per AWS Glue 3.0 non supporta le tabelle Hudi Merge on Read (MoR).
- `ALTER TABLE … RENAME TO` non è disponibile per Apache Iceberg 0.13.1 per AWS Glue 3.0.

Limitazioni per le tabelle in formato data lake gestite dalle autorizzazioni di Lake Formation

I formati di data lake sono integrati con AWS Glue ETL tramite le autorizzazioni di Lake Formation. La creazione di un DynamicFrame utilizzo non `create_dynamic_frame` è supportata. Per maggiori informazioni, consulta i seguenti esempi:

- Esempio: lettura e scrittura della tabella Iceberg con il controllo delle autorizzazioni di Lake Formation
- Esempio: lettura e scrittura della tabella Hudi con il controllo delle autorizzazioni di Lake Formation
- Esempio: lettura e scrittura della tabella Delta Lake con il controllo delle autorizzazioni di Lake Formation

> **Note**
>
> L'integrazione con AWS Glue ETL tramite le autorizzazioni Lake Formation per Apache Hudi, Apache Iceberg e Delta Lake è supportata solo in AWS Glue versione 4.0.

Apache Iceberg ha la migliore integrazione con AWS Glue ETL tramite le autorizzazioni di Lake Formation. Supporta quasi tutte le operazioni e include il supporto per SQL.

Hudi supporta la maggior parte delle operazioni di base, ad eccezione di quelle amministrative. Queste opzioni generalmente vengono eseguite tramite la scrittura di dataframe e specificate tramite `additional_options`. È necessario utilizzare le AWS Glue API DataFrames per creare le proprie operazioni poiché SparkSQL non è supportato.

Delta Lake supporta solo la lettura, l'aggiunta e la sovrascrittura dei dati delle tabelle. Delta Lake richiede l'uso delle proprie librerie per poter eseguire varie attività come gli aggiornamenti.

Le seguenti funzionalità non sono disponibili per le tabelle Iceberg gestite dai permessi di Lake Formation.

- Compattazione tramite AWS Glue ETL

- Supporto Spark SQL tramite AWS Glue ETL

Di seguito, sono riportate le limitazioni delle tabelle Hudi gestite dai permessi di Lake Formation:

- Rimozione di file orfani

Di seguito, sono riportate le limitazioni delle tabelle Delta Lake gestite dai permessi di Lake Formation:

- Tutte le funzionalità diverse dall'inserimento e dalla lettura dalle tabelle Delta Lake.

Utilizzo del framework Hudi in AWS Glue

AWS Glue 3.0 e versioni successive supportano il framework Apache Hudi per i data lake. Hudi è un framework di archiviazione di data lake open source che semplifica l'elaborazione incrementale dei dati e lo sviluppo di pipeline di dati. Questo argomento descrive le funzionalità disponibili per l'utilizzo dei dati in AWS Glue durante il trasporto o l'archiviazione dei dati in una tabella Hudi. Per ulteriori informazioni su Hudi, consulta la documentazione ufficiale di Apache Hudi.

È possibile usare AWS Glue per eseguire operazioni di lettura e scrittura sulle tabelle Hudi in Amazon S3 o lavorare con le tabelle Hudi utilizzando il catalogo dati AWS Glue. Sono supportate anche operazioni aggiuntive, tra cui inserimento, aggiornamento e tutte le operazioni di Apache Spark.

> **Note**
>
> Apache Hudi 0.10.1 per AWS Glue 3.0 non supporta le tabelle Hudi Merge on Read (MoR).

La tabella seguente elenca la versione di Hudi inclusa in ogni versione di AWS Glue.

Versione di AWS Glue	Versione Hudi supportata
4.0	0.12.1
3.0	0,10,1

Per ulteriori informazioni sui framework di data lake supportati da AWS Glue, consulta Utilizzo di framework data lake con processi ETL di AWS Glue.

Abilitazione di Hudi

Per abilitare Hudi per AWS Glue, completa le seguenti attività:

- Specifica `hudi` come valore per i parametri del processo `--datalake-formats`. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.
- Crea una chiave denominata `--conf` per il tuo processo AWS Glue e impostala sul seguente valore. In alternativa, puoi impostare la seguente configurazione usando `SparkConf` nel tuo script. Queste impostazioni consentono ad Apache Spark di gestire correttamente le tabelle Hudi.

```
spark.serializer=org.apache.spark.serializer.KryoSerializer --conf
 spark.sql.hive.convertMetastoreParquet=false
```

- Il supporto delle autorizzazioni di Lake Formation per Hudi è abilitato per impostazione predefinita per AWS Glue 4.0. Non è necessaria alcuna configurazione aggiuntiva per la lettura/scrittura su tabelle Hudi registrate da Lake Formation. Per leggere una tabella Hudi registrata, il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SELECT. Per scrivere su una tabella Hudi registrata, il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SUPER. Per ulteriori informazioni sulla gestione delle autorizzazioni di Lake Formation, consulta Concessione e revoca delle autorizzazioni del catalogo dati.

Utilizzo di una versione differente di Hudi

Per utilizzare una versione di Hudi non supportata da AWS Glue, specifica i tuoi file JAR Hudi utilizzando il parametro del processo `--extra-jars`. Non includere `hudi` come valore per il parametro del processo `--datalake-formats`.

Esempio: scrittura di una tabella Hudi su Amazon S3 e registrazione nel catalogo dati AWS Glue

Il seguente script di esempio mostra come scrivere una tabella Hudi su Amazon S3 e registrarla nel catalogo dati AWS Glue. Per registrare la tabella, viene utilizzato lo strumento Hive Sync di Hudi.

> **Note**
>
> Questo esempio consente di impostare il parametro del processo `--enable-glue-datacatalog` in modo da utilizzare il catalogo dati AWS Glue come metastore Apache Spark Hive. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.

Python

```
# Example: Create a Hudi table from a DataFrame
# and register the table to Glue Data Catalog

additional_options={
    "hoodie.table.name": "<your_table_name>",
    "hoodie.datasource.write.storage.type": "COPY_ON_WRITE",
    "hoodie.datasource.write.operation": "upsert",
    "hoodie.datasource.write.recordkey.field": "<your_recordkey_field>",
    "hoodie.datasource.write.precombine.field": "<your_precombine_field>",
    "hoodie.datasource.write.partitionpath.field": "<your_partitionkey_field>",
    "hoodie.datasource.write.hive_style_partitioning": "true",
    "hoodie.datasource.hive_sync.enable": "true",
    "hoodie.datasource.hive_sync.database": "<your_database_name>",
    "hoodie.datasource.hive_sync.table": "<your_table_name>",
    "hoodie.datasource.hive_sync.partition_fields": "<your_partitionkey_field>",
    "hoodie.datasource.hive_sync.partition_extractor_class":
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
    "hoodie.datasource.hive_sync.use_jdbc": "false",
    "hoodie.datasource.hive_sync.mode": "hms",
    "path": "s3://<s3Path/>"
}

dataFrame.write.format("hudi") \
    .options(**additional_options) \
    .mode("overwrite") \
    .save()
```

Scala

```
// Example: Example: Create a Hudi table from a DataFrame
// and register the table to Glue Data Catalog

val additionalOptions = Map(
```

```
  "hoodie.table.name" -> "<your_table_name>",
  "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
  "hoodie.datasource.write.operation" -> "upsert",
  "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
  "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
  "hoodie.datasource.write.partitionpath.field" -> "<your_partitionkey_field>",
  "hoodie.datasource.write.hive_style_partitioning" -> "true",
  "hoodie.datasource.hive_sync.enable" -> "true",
  "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
  "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
  "hoodie.datasource.hive_sync.partition_fields" -> "<your_partitionkey_field>",
  "hoodie.datasource.hive_sync.partition_extractor_class" ->
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
  "hoodie.datasource.hive_sync.use_jdbc" -> "false",
  "hoodie.datasource.hive_sync.mode" -> "hms",
  "path" -> "s3://<s3Path/>")

dataFrame.write.format("hudi")
  .options(additionalOptions)
  .mode("append")
  .save()
```

Esempio: lettura di una tabella Hudi da Amazon S3 tramite il catalogo dati AWS Glue

In questo esempio viene letta la tabella Hudi che hai creato in Esempio: scrittura di una tabella Hudi su Amazon S3 e registrazione nel catalogo dati AWS Glue da Amazon S3.

> **Note**
>
> Questo esempio consente di impostare il parametro del processo `--enable-glue-datacatalog` in modo da utilizzare il catalogo dati AWS Glue come metastore Apache Spark Hive. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.

Python

Per questo esempio, usa il metodo `GlueContext.create_data_frame.from_catalog()`.

```
# Example: Read a Hudi table from Glue Data Catalog

from awsglue.context import GlueContext
```

```
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

dataFrame = glueContext.create_data_frame.from_catalog(
    database = "<your_database_name>",
    table_name = "<your_table_name>"
)
```

Scala

Per questo esempio, utilizza il metodo getCatalogSource.

```
// Example: Read a Hudi table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dataFrame = glueContext.getCatalogSource(
      database = "<your_database_name>",
      tableName = "<your_table_name>"
    ).getDataFrame()
  }
}
```

Esempio: aggiornamento e inserimento di un **`DataFrame`** in una tabella Hudi in Amazon S3

In questo esempio viene utilizzato il catalogo dati AWS Glue per inserire un DataFrame nella tabella Hudi creata in Esempio: scrittura di una tabella Hudi su Amazon S3 e registrazione nel catalogo dati AWS Glue.

> **Note**
>
> Questo esempio consente di impostare il parametro del processo `--enable-glue-datacatalog` in modo da utilizzare il catalogo dati AWS Glue come metastore Apache Spark Hive. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.

Python

Per questo esempio, usa il metodo `GlueContext.write_data_frame.from_catalog()`.

```
# Example: Upsert a Hudi table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame = dataFrame,
    database = "<your_database_name>",
    table_name = "<your_table_name>",
    additional_options={
        "hoodie.table.name": "<your_table_name>",
        "hoodie.datasource.write.storage.type": "COPY_ON_WRITE",
        "hoodie.datasource.write.operation": "upsert",
        "hoodie.datasource.write.recordkey.field": "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field": "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field": "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning": "true",
        "hoodie.datasource.hive_sync.enable": "true",
        "hoodie.datasource.hive_sync.database": "<your_database_name>",
        "hoodie.datasource.hive_sync.table": "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields": "<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class":
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc": "false",
        "hoodie.datasource.hive_sync.mode": "hms"
    }
)
```

Scala

Per questo esempio, utilizza il metodo getCatalogSink.

```
// Example: Upsert a Hudi table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.JsonOptions
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = JsonOptions(Map(
        "hoodie.table.name" -> "<your_table_name>",
        "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
        "hoodie.datasource.write.operation" -> "upsert",
        "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field" ->
 "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning" -> "true",
        "hoodie.datasource.hive_sync.enable" -> "true",
        "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
        "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields" ->
 "<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class" ->
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc" -> "false",
        "hoodie.datasource.hive_sync.mode" -> "hms"
      )))
      .writeDataFrame(dataFrame, glueContext)
  }
}
```

Esempio: lettura di una tabella Hudi da Amazon S3 tramite Spark

In questo esempio viene letta una tabella Hudi da Amazon S3 tramite l'API Spark DataFrame.

Python

```
# Example: Read a Hudi table from S3 using a Spark DataFrame

dataFrame = spark.read.format("hudi").load("s3://<s3path/>")
```

Scala

```
// Example: Read a Hudi table from S3 using a Spark DataFrame

val dataFrame = spark.read.format("hudi").load("s3://<s3path/>")
```

Esempio: scrittura di una tabella Hudi su Amazon S3 tramite Spark

In questo esempio viene scritta una tabella Hudi su Amazon S3 tramite Spark.

Python

```
# Example: Write a Hudi table to S3 using a Spark DataFrame

dataFrame.write.format("hudi") \
    .options(**additional_options) \
    .mode("overwrite") \
    .save("s3://<s3Path/>")
```

Scala

```
// Example: Write a Hudi table to S3 using a Spark DataFrame

dataFrame.write.format("hudi")
  .options(additionalOptions)
  .mode("overwrite")
  .save("s3://<s3path/>")
```

Esempio: lettura e scrittura della tabella Hudi con il controllo delle autorizzazioni di Lake Formation

Questo esempio legge da e scrive su una tabella Hudi con il controllo delle autorizzazioni di Lake Formation.

1. Crea una tabella Hudi e registrala in Lake Formation.

a. Per abilitare il controllo delle autorizzazioni di Lake Formation, devi prima registrare il percorso della tabella Amazon S3 su Lake Formation. Per ulteriori informazioni, consulta la pagina Registrazione di una posizione Amazon S3. Puoi registrarlo dalla console di Lake Formation o utilizzando la CLI AWS:

```
aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-
folder> --use-service-linked-role --region <REGION>
```

Una volta registrata una posizione Amazon S3, qualsiasi tabella AWS Glue che punta alla posizione, o a una delle sue sedi secondarie, restituirà il valore del parametro `IsRegisteredWithLakeFormation` come true nella chiamata `GetTable`.

b. Crea una tabella Hudi che punti al percorso registrato di Amazon S3 tramite l'API Spark DataFrame:

```
hudi_options = {
    'hoodie.table.name': table_name,
    'hoodie.datasource.write.storage.type': 'COPY_ON_WRITE',
    'hoodie.datasource.write.recordkey.field': 'product_id',
    'hoodie.datasource.write.table.name': table_name,
    'hoodie.datasource.write.operation': 'upsert',
    'hoodie.datasource.write.precombine.field': 'updated_at',
    'hoodie.datasource.write.hive_style_partitioning': 'true',
    'hoodie.upsert.shuffle.parallelism': 2,
    'hoodie.insert.shuffle.parallelism': 2,
    'path': <S3_TABLE_LOCATION>,
    'hoodie.datasource.hive_sync.enable': 'true',
    'hoodie.datasource.hive_sync.database': database_name,
    'hoodie.datasource.hive_sync.table': table_name,
    'hoodie.datasource.hive_sync.use_jdbc': 'false',
    'hoodie.datasource.hive_sync.mode': 'hms'
}

df_products.write.format("hudi")  \
    .options(**hudi_options)  \
    .mode("overwrite")  \
    .save()
```

2. Concedi a Lake Formation l'autorizzazione per il ruolo IAM del processo AWS Glue. Puoi concedere le autorizzazioni dalla console di Lake Formation o utilizzando la CLI AWS. Per ulteriori

informazioni, consulta la pagina Concessione delle autorizzazioni alla tabella tramite la console di Lake Formation e il metodo delle risorse denominate

3. Leggi la tabella Hudi registrata in Lake Formation. Il codice equivale a leggere una tabella Hudi non registrata. Il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SELECT affinché la lettura abbia esito positivo.

```
val dataFrame = glueContext.getCatalogSource(
     database = "<your_database_name>",
     tableName = "<your_table_name>"
   ).getDataFrame()
```

4. Scrivi sulla tabella Hudi registrata in Lake Formation. Il codice equivale a scrivere su una tabella Hudi non registrata. Il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SUPER affinché la scrittura abbia esito positivo.

```
glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = JsonOptions(Map(
        "hoodie.table.name" -> "<your_table_name>",
        "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
        "hoodie.datasource.write.operation" -> "<write_operation>",
        "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field" -> "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning" -> "true",
        "hoodie.datasource.hive_sync.enable" -> "true",
        "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
        "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields" ->
 "<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class" ->
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc" -> "false",
        "hoodie.datasource.hive_sync.mode" -> "hms"
      )))
      .writeDataFrame(dataFrame, glueContext)
```

Utilizzo del framework Delta Lake in AWS Glue

AWS Glue 3.0 e versioni successive supportano il framework Linux Foundation Delta Lake. Delta Lake è un framework di archiviazione di data lake open source che consente di eseguire transazioni

ACID, scalare la gestione dei metadati e unificare lo streaming e l'elaborazione dei dati in batch. Questo argomento descrive le funzionalità disponibili per l'utilizzo dei dati in AWS Glue durante il trasporto o l'archiviazione dei dati in una tabella Delta Lake. Per saperne di più su Delta Lake, consulta la documentazione ufficiale di Delta Lake.

È possibile usare AWS Glue per eseguire operazioni di lettura e scrittura sulle tabelle Delta Lake in Amazon S3 o lavorare con le tabelle Delta Lake utilizzando il catalogo di dati AWS Glue. Sono supportate anche operazioni aggiuntive come inserimento, aggiornamento e letture e scritture in batch di tabelle. Quando usi le tabelle Delta Lake, hai anche la possibilità di utilizzare metodi della libreria Python di Delta Lake come `DeltaTable.forPath`. Per ulteriori informazioni sulla libreria Python di Delta Lake, consulta la documentazione Python di Delta Lake.

La tabella seguente elenca la versione di Delta Lake inclusa in ogni versione di AWS Glue.

Versione di AWS Glue	Versione Delta Lake supportata
4.0	2.1.0
3.0	1.0.0

Per ulteriori informazioni sui framework data lake supportati da AWS Glue, consulta Utilizzo di framework data lake con processi ETL di AWS Glue.

Abilitazione di Delta Lake per AWS Glue

Per abilitare Delta Lake per AWS Glue, completa le seguenti attività:

- Specifica `delta` come valore per i parametri del processo `--datalake-formats`. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.
- Crea una chiave denominata `--conf` per il tuo processo AWS Glue e impostala sul seguente valore. In alternativa, puoi impostare la seguente configurazione usando `SparkConf` nel tuo script. Queste impostazioni consentono ad Apache Spark di gestire correttamente le tabelle Delta Lake.

```
spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog --
conf
 spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDriverLogStore
```

- Il supporto delle autorizzazioni di Lake Formation per le tabelle Delta è abilitato per impostazione predefinita per AWS Glue 4.0. Non è necessaria alcuna configurazione aggiuntiva per la lettura/scrittura su tabelle Delta registrate da Lake Formation. Per leggere una tabella Delta registrata, il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SELECT. Per scrivere su una tabella Delta registrata, il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SUPER. Per ulteriori informazioni sulla gestione delle autorizzazioni di Lake Formation, consulta Concessione e revoca delle autorizzazioni del catalogo dati.

Utilizzo di una versione differente di Delta Lake

Per utilizzare una versione di Delta lake non supportata da AWS Glue, specifica i tuoi file JAR Delta Lake utilizzando il parametro del processo `--extra-jars`. Non includere `delta` come valore per il parametro del processo `--datalake-formats`. Per utilizzare la libreria Python Delta Lake in questo caso, è necessario specificare i file JAR della libreria utilizzando il parametro del processo `--extra-py-files`. La libreria Python è contenuta nei file JAR di Delta Lake.

Esempio: scrittura di una tabella Delta Lake su Amazon S3 e registrazione nel catalogo dati AWS Glue

Il seguente script ETL di AWS Glue dimostra come scrivere una tabella Delta Lake su Amazon S3 e registrarla nel catalogo dati AWS Glue.

Python

```
# Example: Create a Delta Lake table from a DataFrame
# and register the table to Glue Data Catalog

additional_options = {
    "path": "s3://<s3Path>"
}
dataFrame.write \
    .format("delta") \
    .options(**additional_options) \
    .mode("append") \
    .partitionBy("<your_partitionkey_field>") \
    .saveAsTable("<your_database_name>.<your_table_name>")
```

Scala

```
// Example: Example: Create a Delta Lake table from a DataFrame
```

```
// and register the table to Glue Data Catalog

val additional_options = Map(
  "path" -> "s3://<s3Path>"
)
dataFrame.write.format("delta")
  .options(additional_options)
  .mode("append")
  .partitionBy("<your_partitionkey_field>")
  .saveAsTable("<your_database_name>.<your_table_name>")
```

Esempio: lettura di una tabella Delta Lake da Amazon S3 tramite il catalogo dati AWS Glue

Il seguente script ETL di AWS Glue legge la tabella Delta Lake creata in Esempio: scrittura di una tabella Delta Lake su Amazon S3 e registrazione nel catalogo dati AWS Glue.

Python

Per questo esempio, utilizza il metodo create_data_frame.from_catalog.

```
# Example: Read a Delta Lake table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

Per questo esempio, usa il metodo getCatalogSource.

```
// Example: Read a Delta Lake table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext
```

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val df = glueContext.getCatalogSource("<your_database_name>",
 "<your_table_name>",
      additionalOptions = additionalOptions)
      .getDataFrame()
  }
}
```

Esempio: inserimento di un **`DataFrame`** in una tabella Delta Lake in Amazon S3 tramite il catalogo dati AWS Glue

Questo esempio inserisce i dati nella tabella Delta Lake creata in Esempio: scrittura di una tabella Delta Lake su Amazon S3 e registrazione nel catalogo dati AWS Glue.

> **Note**
>
> Questo esempio consente di impostare il parametro del processo `--enable-glue-datacatalog` in modo da utilizzare il catalogo dati AWS Glue come metastore Apache Spark Hive. Per ulteriori informazioni, vedi Parametri del processo AWS Glue.

Python

Per questo esempio, utilizza il metodo write_data_frame.from_catalog.

```
# Example: Insert into a Delta Lake table in S3 using Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
```

```
    additional_options=additional_options
)
```

Scala

Per questo esempio, usa il metodo getCatalogSink.

```
// Example: Insert into a Delta Lake table in S3 using Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = additionalOptions)
      .writeDataFrame(dataFrame, glueContext)
  }
}
```

Esempio: lettura di una tabella Delta Lake da Amazon S3 tramite l'API Spark

In questo esempio viene letta una tabella Delta Lake da Amazon S3 tramite l'API Spark.

Python

```
# Example: Read a Delta Lake table from S3 using a Spark DataFrame

dataFrame = spark.read.format("delta").load("s3://<s3path/>")
```

Scala

```
// Example: Read a Delta Lake table from S3 using a Spark DataFrame

val dataFrame = spark.read.format("delta").load("s3://<s3path/>")
```

Esempio: scrittura di una tabella Delta Lake su Amazon S3 tramite Spark

In questo esempio viene scritta una tabella Delta Lake su Amazon S3 tramite Spark.

Python

```
# Example: Write a Delta Lake table to S3 using a Spark DataFrame

dataFrame.write.format("delta") \
    .options(**additional_options) \
    .mode("overwrite") \
    .partitionBy("<your_partitionkey_field>") 
    .save("s3://<s3Path>")
```

Scala

```
// Example: Write a Delta Lake table to S3 using a Spark DataFrame

dataFrame.write.format("delta")
  .options(additionalOptions)
  .mode("overwrite")
  .partitionBy("<your_partitionkey_field>")
  .save("s3://<s3path/>")
```

Esempio: lettura e scrittura della tabella Delta Lake con il controllo delle autorizzazioni di Lake Formation

Questo esempio legge da e scrive su una tabella Delta Lake con il controllo delle autorizzazioni di Lake Formation.

1. Crea una tabella Delta e registrala in Lake Formation
   a. Per abilitare il controllo delle autorizzazioni di Lake Formation, devi prima registrare il percorso della tabella Amazon S3 su Lake Formation. Per ulteriori informazioni, consulta la pagina Registrazione di una posizione Amazon S3. Puoi registrarlo dalla console di Lake Formation o utilizzando la CLI AWS:

      ```
      aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-folder> --use-service-linked-role --region <REGION>
      ```

      Una volta registrata una posizione Amazon S3, qualsiasi tabella AWS Glue che punta alla posizione, o a una delle sue sedi secondarie, restituirà il valore del parametro `IsRegisteredWithLakeFormation` come true nella chiamata `GetTable`.
   b. Crea una tabella Delta che punti al percorso registrato di Amazon S3 tramite Spark:

> **Note**
>
> Di seguito vengono mostrati gli esempi Python.

```
dataFrame.write \
 .format("delta") \
 .mode("overwrite") \
 .partitionBy("<your_partitionkey_field>") \
 .save("s3://<the_s3_path>")
```

Dopo aver scritto i dati su Amazon S3, usa il crawler AWS Glue per creare una nuova tabella del catalogo Delta. Per ulteriori informazioni, consulta Introduzione al supporto nativo delle tabelle Delta Lake con i crawler AWS Glue.

Puoi anche creare la tabella manualmente tramite l'API `CreateTable` di AWS Glue.

2. Concedi a Lake Formation l'autorizzazione per il ruolo IAM del processo AWS Glue. Puoi concedere le autorizzazioni dalla console di Lake Formation o utilizzando la CLI AWS. Per ulteriori informazioni, consulta la pagina Concessione delle autorizzazioni alla tabella tramite la console di Lake Formation e il metodo delle risorse denominate
3. Leggi la tabella Delta registrata in Lake Formation. Il codice equivale a leggere una tabella Delta non registrata. Il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SELECT affinché la lettura abbia esito positivo.

```
# Example: Read a Delta Lake table from Glue Data Catalog

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

4. Scrivi sulla tabella Delta registrata in Lake Formation. Il codice equivale a scrivere su una tabella Delta non registrata. Il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SUPER affinché la scrittura abbia esito positivo.

Per impostazione predefinita, AWS Glue utilizza `Append` come saveMode. È possibile modificarlo impostando l'opzione saveMode in `additional_options`. Per informazioni sul supporto saveMode nelle tabelle Delta, consulta Scrivi su una tabella.

```
glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Utilizzo del framework Iceberg in AWS Glue

AWS Glue 3.0 e versioni successive supportano il framework Apache Iceberg per i data lake. Iceberg fornisce un formato di tabella ad alte prestazioni che funziona proprio come una tabella SQL. Questo argomento descrive le funzionalità disponibili per l'utilizzo dei dati in AWS Glue durante il trasporto o l'archiviazione dei dati in una tabella Iceberg. Per ulteriori informazioni su Iceberg, consulta la documentazione ufficiale di Apache Iceberg.

È possibile usare AWS Glue per eseguire operazioni di lettura e scrittura sulle tabelle Iceberg in Amazon S3 o lavorare con le tabelle Iceberg utilizzando il catalogo dati AWS Glue. Sono supportate anche operazioni aggiuntive, tra cui inserimento, aggiornamento e tutte le scritture Spark delle query Spark.

> **Note**
>
> `ALTER TABLE … RENAME TO` non è disponibile per Apache Iceberg 0.13.1 per AWS Glue 3.0.

La tabella seguente elenca la versione di Iceberg inclusa in ogni versione di AWS Glue.

Versione di AWS Glue	Versione Iceberg supportata
4.0	1.0.0
3.0	0.13.1

Per ulteriori informazioni sui framework di data lake supportati da AWS Glue, consulta Utilizzo di framework data lake con processi ETL di AWS Glue.

Abilitazione del framework Iceberg

Per abilitare Iceberg per AWS Glue, completa le seguenti attività:

- Specifica `iceberg` come valore per i parametri del processo `--datalake-formats`. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.
- Crea una chiave denominata `--conf` per il tuo processo AWS Glue e impostala sul seguente valore. In alternativa, puoi impostare la seguente configurazione usando `SparkConf` nel tuo script. Queste impostazioni consentono ad Apache Spark di gestire correttamente le tabelle Iceberg.

```
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
--conf spark.sql.catalog.glue_catalog=org.apache.iceberg.spark.SparkCatalog
--conf spark.sql.catalog.glue_catalog.warehouse=s3://<your-warehouse-dir>/
--conf spark.sql.catalog.glue_catalog.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog
--conf spark.sql.catalog.glue_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO
```

  Se stai leggendo da o scrivendo su tabelle Iceberg registrate con Lake Formation, aggiungi la seguente configurazione per abilitare il supporto di Lake Formation. Solo AWS Glue 4.0 supporta le tabelle Iceberg registrate con Lake Formation:

```
--conf spark.sql.catalog.glue_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.glue_catalog.glue.id=<table-catalog-id>
```

  Se utilizzi AWS Glue 3.0 con Iceberg 0.13.1, devi impostare le seguenti configurazioni aggiuntive per utilizzare il gestore blocchi di Amazon DynamoDB per garantire la transazione atomica. AWS Glue 4.0 utilizza il blocco ottimistico per impostazione predefinita. Per ulteriori informazioni, consulta Integrazioni di AWS Iceberg nella documentazione ufficiale di Apache Iceberg.

```
--conf spark.sql.catalog.glue_catalog.lock-
impl=org.apache.iceberg.aws.glue.DynamoLockManager
--conf spark.sql.catalog.glue_catalog.lock.table=<your-dynamodb-table-name>
```

Utilizzo di una versione differente di Iceberg

Per utilizzare una versione di Iceberg non supportata da AWS Glue, specifica i tuoi file JAR Iceberg utilizzando il parametro del processo `--extra-jars`. Non includere `iceberg` come valore per il parametro `--datalake-formats`.

Abilitazione della crittografia per le tabelle Iceberg

> **Note**
>
> Le tabelle Iceberg dispongono di meccanismi propri per abilitare la crittografia lato server. È necessario abilitare questa configurazione oltre alla configurazione di sicurezza di AWS Glue.

Per abilitare la crittografia lato server sulle tabelle Iceberg, consulta le indicazioni contenute nella documentazione di Iceberg.

Esempio: scrittura di una tabella Iceberg su Amazon S3 e registrazione nel catalogo dati AWS Glue

Questo script di esempio dimostra come scrivere una tabella Iceberg su Amazon S3. L'esempio utilizza Integrazioni di AWS Iceberg per registrare la tabella nel catalogo dati AWS.

Python

```
# Example: Create an Iceberg table from a DataFrame
# and register the table to Glue Data Catalog

dataFrame.createOrReplaceTempView("tmp_<your_table_name>")

query = f"""
CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
USING iceberg
TBLPROPERTIES ("format-version"="2")
AS SELECT * FROM tmp_<your_table_name>
"""
spark.sql(query)
```

Scala

```
// Example: Example: Create an Iceberg table from a DataFrame
// and register the table to Glue Data Catalog

dataFrame.createOrReplaceTempView("tmp_<your_table_name>")
```

```
val query = """CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
USING iceberg
TBLPROPERTIES ("format-version"="2")
AS SELECT * FROM tmp_<your_table_name>
"""
spark.sql(query)
```

In alternativa, è possibile scrivere una tabella Iceberg su Amazon S3 e catalogo dati tramite metodi Spark.

Prerequisiti: è necessario fornire un catalogo per l'utilizzo della libreria Iceberg. AWS Glue facilita l'utilizzo di Catalogo dati AWS Glue. Catalogo dati AWS Glue è preconfigurato per essere utilizzato dalle librerie Spark come `glue_catalog`. Le tabelle di Catalogo dati sono identificate da un *`databaseName`* e un *`tableName`*. Per ulteriori informazioni su Catalogo dati AWS Glue, consulta la pagina Catalogo dati e crawler.

Se non utilizzi Catalogo dati AWS Glue, dovrai fornire un catalogo tramite le API Spark. Per ulteriori informazioni, consulta la pagina Spark Configuration nella documentazione di Spark.

In questo esempio viene scritta una tabella Iceberg in Amazon S3 e il catalogo dati tramite Spark.

Python

```
# Example: Write an Iceberg table to S3 on the Glue Data Catalog

# Create (equivalent to CREATE TABLE AS SELECT)
dataFrame.writeTo("glue_catalog.databaseName.tableName") \
    .tableProperty("format-version", "2") \
    .create()

# Append (equivalent to INSERT INTO)
dataFrame.writeTo("glue_catalog.databaseName.tableName") \
    .tableProperty("format-version", "2") \
    .append()
```

Scala

```
// Example: Write an Iceberg table to S3 on the Glue Data Catalog

// Create (equivalent to CREATE TABLE AS SELECT)
dataFrame.writeTo("glue_catalog.databaseName.tableName")
```

```
    .tableProperty("format-version", "2")
    .create()

// Append (equivalent to INSERT INTO)
dataFrame.writeTo("glue_catalog.databaseName.tableName")
    .tableProperty("format-version", "2")
    .append()
```

Esempio: lettura di una tabella Iceberg da Amazon S3 tramite il catalogo dati AWS Glue

Questo esempio legge la tabella Iceberg che creata in Esempio: scrittura di una tabella Iceberg su Amazon S3 e registrazione nel catalogo dati AWS Glue.

Python

Per questo esempio, usa il metodo `GlueContext.create_data_frame.from_catalog()`.

```
# Example: Read an Iceberg table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

Per questo esempio, utilizza il metodo getCatalogSource.

```
// Example: Read an Iceberg table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
```

```
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val df = glueContext.getCatalogSource("<your_database_name>",
 "<your_table_name>",
      additionalOptions = additionalOptions)
      .getDataFrame()
  }
}
```

Esempio: inserimento di un **`DataFrame`** in una tabella Iceberg in Amazon S3 tramite il catalogo dati AWS Glue

Questo esempio inserisce i dati nella tabella Iceberg creata in Esempio: scrittura di una tabella Iceberg su Amazon S3 e registrazione nel catalogo dati AWS Glue.

> **Note**
>
> Questo esempio consente di impostare il parametro del processo `--enable-glue-datacatalog` in modo da utilizzare il catalogo dati AWS Glue come metastore Apache Spark Hive. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.

Python

Per questo esempio, usa il metodo `GlueContext.write_data_frame.from_catalog()`.

```
# Example: Insert into an Iceberg table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

Per questo esempio, utilizza il metodo getCatalogSink.

```
// Example: Insert into an Iceberg table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = additionalOptions)
      .writeDataFrame(dataFrame, glueContext)
  }
}
```

Esempio: lettura di una tabella Iceberg da Amazon S3 tramite Spark

Prerequisiti: è necessario fornire un catalogo per l'utilizzo della libreria Iceberg. AWS Glue facilita l'utilizzo di Catalogo dati AWS Glue. Catalogo dati AWS Glue è preconfigurato per essere utilizzato dalle librerie Spark come `glue_catalog`. Le tabelle di Catalogo dati sono identificate da un *`databaseName`* e un *`tableName`*. Per ulteriori informazioni su Catalogo dati AWS Glue, consulta la pagina Catalogo dati e crawler.

Se non utilizzi Catalogo dati AWS Glue, dovrai fornire un catalogo tramite le API Spark. Per ulteriori informazioni, consulta la pagina Spark Configuration nella documentazione di Spark.

In questo esempio viene letta una tabella Iceberg in Amazon S3 da Catalogo dati tramite Spark.

Python

```
# Example: Read an Iceberg table on S3 as a DataFrame from the Glue Data Catalog

dataFrame = spark.read.format("iceberg").load("glue_catalog.databaseName.tableName")
```

Scala

```
// Example: Read an Iceberg table on S3 as a DataFrame from the Glue Data Catalog
```

```
val dataFrame =
 spark.read.format("iceberg").load("glue_catalog.databaseName.tableName")
```

Esempio: lettura e scrittura della tabella Iceberg con il controllo delle autorizzazioni di Lake Formation

Questo esempio legge da e scrive su una tabella Iceberg con il controllo delle autorizzazioni di Lake Formation.

1. Crea una tabella Iceberg e registrala in Lake Formation:

   a. Per abilitare il controllo delle autorizzazioni di Lake Formation, devi prima registrare il percorso della tabella Amazon S3 su Lake Formation. Per ulteriori informazioni, consulta la pagina Registrazione di una posizione Amazon S3. Puoi registrarlo dalla console di Lake Formation o utilizzando la CLI AWS:

   ```
   aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-folder> --use-service-linked-role --region <REGION>
   ```

   Una volta registrata una posizione Amazon S3, qualsiasi tabella AWS Glue che punta alla posizione, o a una delle sue sedi secondarie, restituirà il valore del parametro `IsRegisteredWithLakeFormation` come true nella chiamata `GetTable`.

   b. Crea una tabella Iceberg che punti al percorso registrato di Amazon S3 tramite Spark SQL:

   > **Note**
   >
   > Di seguito vengono mostrati gli esempi Python.

   ```
   dataFrame.createOrReplaceTempView("tmp_<your_table_name>")

   query = f"""
   CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
   USING iceberg
   AS SELECT * FROM tmp_<your_table_name>
   """
   spark.sql(query)
   ```

Puoi anche creare la tabella manualmente tramite l'API `CreateTable` di AWS Glue. Per ulteriori informazioni, consulta Creazione di tabelle Apache Iceberg.

2. Concedi a Lake Formation l'autorizzazione per il ruolo IAM del processo. Puoi concedere le autorizzazioni dalla console di Lake Formation o utilizzando la CLI AWS. Per ulteriori informazioni, consulta: https://docs.aws.amazon.com/lake-formation/latest/dg/granting-table-permissions.html
3. Leggi una tabella Iceberg registrata con Lake Formation. Il codice equivale a leggere una tabella Iceberg non registrata. Il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SELECT affinché la lettura abbia esito positivo.

```
# Example: Read an Iceberg table from the AWS Glue Data Catalog
from awsglue.context import GlueContextfrom pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

4. Scrivi su una tabella Iceberg registrata con Lake Formation. Il codice equivale a scrivere su una tabella Iceberg non registrata. Il ruolo IAM del processo AWS Glue deve disporre dell'autorizzazione SUPER affinché la scrittura abbia esito positivo.

```
glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Riferimento alla configurazione condivisa

È possibile utilizzare i seguenti valori di `format_options` con ogni tipo di formato.

- `attachFilename`: una stringa nel formato appropriato da utilizzare come nome di colonna. Se si fornisce questa opzione, il nome del file di origine del record verrà aggiunto al record. Il valore del parametro verrà utilizzato come nome della colonna.

- `attachTimestamp`: una stringa nel formato appropriato da utilizzare come nome di colonna. Se si fornisce questa opzione, l'ora di modifica del file di origine del record verrà aggiunta al record. Il valore del parametro verrà utilizzato come nome della colonna.

Supporto del catalogo dati di AWS Glue per i processi Spark SQL

AWS Glue Data Catalog è un catalogo compatibile con il database metastore Apache Hive. Puoi configurare gli endpoint di sviluppo e i processi AWS Glue affinché usino catalogo dati come metastore Apache Hive esterno. Quindi, puoi eseguire le query Apache Spark SQL direttamente sulle tabelle archiviate nel catalogo dati. I frame dinamici AWS Glue si integrano con catalogo dati per impostazione predefinita. Tuttavia con questa caratteristica i processi Spark SQL possono iniziare a usare il catalogo dati come metastore Hive esterno.

Questa caratteristica richiede l'accesso di rete all' endpoint API AWS Glue. Per processi AWS Glue con connessioni situate in sottoreti private, è necessario configurare un endpoint VPC o un gateway NAT, per fornire l'accesso alla rete. Per informazioni sulla configurazione degli endpoint VPC, consulta Impostazione dell'accesso di rete agli archivi di dati. Per creare un gateway NAT, consulta Gateway NAT nella Guida per l'utente di Amazon VPC.

È possibile configurare gli endpoint di sviluppo e i processi AWS Glue aggiungendo l'argomento `"--enable-glue-datacatalog": ""` agli argomenti degli endpoint di sviluppo e del processo rispettivamente. Questo argomento imposta determinate configurazioni in Spark che gli consentono di accedere al catalogo dati come metastore Hive esterno. Inoltre abilita il supporto Hive nell'oggetto `SparkSession` creato nell'endpoint di sviluppo o nel processo AWS Glue.

Per abilitare l'accesso del catalogo dati, seleziona la casella di controllo Use AWS Glue Data Catalog as the Hive metastore (Usa AWS Glue Data Catalog come metastore Hive) nel gruppo Catalog options (Opzioni catalogo) nella pagina Add job (Aggiungi processo) o Add endpoint (Aggiungi endpoint) della console. Tieni presente che il ruolo IAM utilizzato per il processo o per l'endpoint di sviluppo deve disporre delle autorizzazioni `glue:CreateDatabase`. Viene creato un database chiamato "`default`" nel catalogo dati, nel caso non fosse già presente.

Osserviamo un esempio per utilizzare questa caratteristica nei processi Spark SQL. L'esempio seguente presuppone che hai sottoposto a crawling il set di dati dei legislatori degli Stati Uniti disponibile in `s3://awsglue-datasets/examples/us-legislators`.

Per serializzare/deserializzare i dati provenienti da tabelle definite in AWS Glue Data Catalog, Spark SQL richiede la classe Hive SerDe per il formato definito nel catalogo dati diAWS Glue nel classpath del processo Spark.

Le classi SerDe per determinati formati comuni sono distribuite da AWS Glue. Di seguito sono elencati i relativi collegamenti Amazon S3:

- JSON
- XML
- Grok

Aggiungi la classe JSON SerDe come un ulteriore JAR all'endpoint di sviluppo. Per i processi, puoi aggiungere la classe SerDe utilizzando l'argomento `--extra-jars` nel campo degli argomenti. Per ulteriori informazioni, consulta Parametri del processo AWS Glue.

Ecco un esempio di JSON di input per creare un endpoint di sviluppo con il catalogo dati abilitato per Spark SQL.

```
{
    "EndpointName": "Name",
    "RoleArn": "role_ARN",
    "PublicKey": "public_key_contents",
    "NumberOfNodes": 2,
    "Arguments": {
      "--enable-glue-datacatalog": ""
    },
    "ExtraJarsS3Path": "s3://crawler-public/json/serde/json-serde.jar"
}
```

Ora esegui una query sulle tabelle create dal set di dati dei legislatori degli Stati Uniti utilizzando Spark SQL.

```
>>> spark.sql("use legislators")
DataFrame[]
>>> spark.sql("show tables").show()
+-----------+-----------------+-----------+
|   database|        tableName|isTemporary|
+-----------+-----------------+-----------+
|legislators|       areas_json|      false|
|legislators|   countries_json|      false|
|legislators|      events_json|      false|
|legislators| memberships_json|      false|
|legislators|organizations_json|      false|
```

```
|legislators|      persons_json|      false|
+-----------+------------------+-----------+
>>> spark.sql("describe memberships_json").show()
+--------------------+---------+-----------------+
|            col_name|data_type|          comment|
+--------------------+---------+-----------------+
|             area_id|   string|from deserializer|
|     on_behalf_of_id|   string|from deserializer|
|     organization_id|   string|from deserializer|
|                role|   string|from deserializer|
|           person_id|   string|from deserializer|
|legislative_perio...|   string|from deserializer|
|          start_date|   string|from deserializer|
|            end_date|   string|from deserializer|
+--------------------+---------+-----------------+
```

Se la classe SerDe per il formato non è disponibile nel classpath del processo, verrà visualizzato un messaggio di errore simile al seguente.

```
>>> spark.sql("describe memberships_json").show()

Caused by: MetaException(message:java.lang.ClassNotFoundException Class
 org.openx.data.jsonserde.JsonSerDe not found)
    at
 org.apache.hadoop.hive.metastore.MetaStoreUtils.getDeserializer(MetaStoreUtils.java:399)
    at
 org.apache.hadoop.hive.ql.metadata.Table.getDeserializerFromMetaStore(Table.java:276)
    ... 64 more
```

Per visualizzare solo gli `organization_id` distinti della tabella `memberships`, esegui la seguente query SQL.

```
>>> spark.sql("select distinct organization_id from memberships_json").show()
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

Se è necessaria la stessa operazione per i frame dinamici, esegui la query seguente.

```
>>> memberships = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="memberships_json")
>>> memberships.toDF().createOrReplaceTempView("memberships")
>>> spark.sql("select distinct organization_id from memberships").show()
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

I DynamicFrame oltre a essere ottimizzati per le operazioni ETL, consentendo a Spark SQL di accedere al catalogo dati direttamente, forniscono un modo conciso per eseguire istruzioni SQL complesse o trasferire applicazioni esistenti.

Utilizzo di segnalibri di processo

AWS Glue per Spark monitora i dati già elaborati attraverso i segnalibri dei processi. Per un riepilogo della funzionalità dei segnalibri di processo e di ciò che supportano, consulta la pagina the section called “Monitoraggio dei dati elaborati mediante segnalibri di processo”. Quando si programma un processo AWS Glue con segnalibri, si ha accesso a una flessibilità non disponibile nei processi visivi.

- Durante la lettura da JDBC, è possibile specificare le colonne da utilizzare come chiavi di segnalibro nello script AWS Glue.
- È possibile scegliere quale `transformation_ctx` applicare a ciascuna chiamata al metodo.

Chiama sempre *`job.init`* all'inizio dello script e alla *`job.commit`* fine dello script con parametri configurati in modo appropriato. Queste due funzioni inizializzano il servizio di segnalibri e aggiornano la modifica dello stato al servizio. I segnalibri non funzioneranno senza che vengano richiamati.

Specifica delle chiavi dei segnalibri

Per i flussi di lavoro JDBC, il segnalibro tiene traccia delle righe lette dal processo confrontando i valori dei campi chiave con un valore aggiunto ai segnalibri. Questa operazione non è necessaria o applicabile per i flussi di processo Amazon S3. Quando scrivi uno script AWS Glue senza l'editor visivo, puoi specificare la colonna di cui tenere traccia mediante i segnalibri. È possibile specificare anche più colonne. Sono consentite lacune nella sequenza di valori quando si specificano chiavi di segnalibro definite dall'utente.

> **Warning**
>
> Se vengono utilizzate le chiavi di segnalibro definite dall'utente, devono essere fornite rigorosamente e monotonicamente in ordine crescente o decrescente. Quando si selezionano campi aggiuntivi per una chiave composta, i campi relativi a concetti come "versioni secondarie" o "numeri di revisione" non soddisfano questi criteri, poiché i loro valori vengono riutilizzati in tutto il set di dati.

È possibile specificare `jobBookmarkKeys` e `jobBookmarkKeysSortOrder` nei seguenti modi:

- `create_dynamic_frame.from_catalog`: utilizza `additional_options`.
- `create_dynamic_frame.from_options`: utilizza `connection_options`.

Contesto di trasformazione

Molti dei metodi di frame AWS Glue PySpark dinamici includono un parametro opzionale denominato`transformation_ctx`, che è un identificatore univoco per l'istanza dell'operatore ETL. Il parametro `transformation_ctx` viene utilizzato per identificare le informazioni sullo stato all'interno di un segnalibro di processo per un determinato operatore. Nello specifico, AWS Glue utilizza `transformation_ctx` per indicizzare la chiave per lo stato del segnalibro.

> **Warning**
>
> Il parametro `transformation_ctx` serve come chiave per cercare nello stato del segnalibro una fonte specifica nello script. Affinché il segnalibro funzioni correttamente, è necessario mantenere sempre la fonte e il parametro `transformation_ctx` associato coerenti. Modificare la proprietà di origine o rinominare il parametro `transformation_ctx` potrebbe rendere il segnalibro precedente non valido e il filtro basato sulla marca temporale potrebbe non produrre il risultato corretto.

Per fare in modo che i segnalibri del processo funzionino correttamente, abilita il parametro del segnalibro del processo e imposta il parametro `transformation_ctx`. Se non passi il parametro `transformation_ctx`, i segnalibri del processo non sono abilitati per un frame dinamico oppure nel metodo viene utilizzata una tabella. Ad esempio, in presenza di un processo ETL che legge e unisce due origini Amazon S3, puoi scegliere di passare il parametro `transformation_ctx` solo ai

metodi per cui vuoi abilitare i segnalibri. Reimpostando il segnalibro per un processo, si reimpostano tutte le trasformazioni associate al processo, indipendentemente dal `transformation_ctx` utilizzato.

Per ulteriori informazioni sulla classe `DynamicFrameReader`, consulta DynamicFrameReader classe. Per ulteriori informazioni sulle PySpark estensioni, vedere. Informazioni di riferimento sulle estensioni PySpark AWS Glue

Esempi

Example

Di seguito è riportato un esempio di script generato per un'origine dati Amazon S3. Le parti dello script necessarie per l'utilizzo dei segnalibri di processo sono visualizzate in corsivo. Per ulteriori informazioni su questi elementi, consulta le API Classe GlueContext e le API Classe DynamicFrameWriter.

```
# Sample Script
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ['JOB_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

datasource0 = glueContext.create_dynamic_frame.from_catalog(
    database = "database",
    table_name = "relatedqueries_csv",
    transformation_ctx = "datasource0"
)

applymapping1 = ApplyMapping.apply(
    frame = datasource0,
    mappings = [("col0", "string", "name", "string"), ("col1", "string", "number",
 "string")],
```

```
    transformation_ctx = "applymapping1"
)

datasink2 = glueContext.write_dynamic_frame.from_options(
    frame = applymapping1,
    connection_type = "s3",
    connection_options = {"path": "s3://input_path"},
    format = "json",
    transformation_ctx = "datasink2"
)


job.commit()
```

Example

Di seguito è riportato un esempio di script generato per un'origine JDBC. La tabella di origine è una tabella dipendente con la colonna empno come chiave primaria. Sebbene per impostazione predefinita il processo utilizzi una chiave primaria sequenziale come chiave di segnalibro se non viene specificata alcuna chiave di segnalibro, poiché empno non è necessariamente sequenziale (potrebbero esserci delle lacune nei valori), non si qualifica come chiave segnalibro predefinita. Di conseguenza, lo script designa esplicitamente empno come chiave di segnalibro. Quella parte del codice è mostrata in corsivo.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

datasource0 = glueContext.create_dynamic_frame.from_catalog(
    database = "hr",
    table_name = "emp",
```

```
    transformation_ctx = "datasource0",
    additional_options = {"jobBookmarkKeys":["empno"],"jobBookmarkKeysSortOrder":"asc"}
)

applymapping1 = ApplyMapping.apply(
    frame = datasource0,
    mappings = [("ename", "string", "ename", "string"), ("hrly_rate", "decimal(38,0)",
 "hrly_rate", "decimal(38,0)"), ("comm", "decimal(7,2)", "comm", "decimal(7,2)"),
 ("hiredate", "timestamp", "hiredate", "timestamp"), ("empno", "decimal(5,0)", "empno",
 "decimal(5,0)"), ("mgr", "decimal(5,0)", "mgr", "decimal(5,0)"), ("photo", "string",
 "photo", "string"), ("job", "string", "job", "string"), ("deptno", "decimal(3,0)",
 "deptno", "decimal(3,0)"), ("ssn", "decimal(9,0)", "ssn", "decimal(9,0)"), ("sal",
 "decimal(7,2)", "sal", "decimal(7,2)")],
    transformation_ctx = "applymapping1"
)

datasink2 = glueContext.write_dynamic_frame.from_options(
    frame = applymapping1,
    connection_type = "s3",
    connection_options = {"path": "s3://hr/employees"},
    format = "csv",
    transformation_ctx = "datasink2"
)

job.commit()
```

Utilizzo del rilevamento dei dati sensibili fuori da AWS Glue Studio

AWS Glue Studio consente di rilevare dati sensibili, tuttavia, è possibile utilizzare la funzionalità di rilevamento dei dati sensibili anche esternamente a AWS Glue Studio.

Per un elenco completo dei tipi di dati sensibili gestiti, consulta la pagina Managed Sensitive Data Types.

Individuazione del Rilevamento di dati sensibili con i tipi PII di AWS Managed

AWS Glue fornisce due API in un processo ETL AWS Glue. Questi sono `detect()` e `classifyColumns()`:

```
  detect(frame: DynamicFrame,
      entityTypesToDetect: Seq[String],
      outputColumnName: String = "DetectedEntities",
```

```
      detectionSensitivity: String = "LOW"): DynamicFrame

detect(frame: DynamicFrame,
      detectionParameters: JsonOptions,
      outputColumnName: String = "DetectedEntities",
      detectionSensitivity: String = "LOW"): DynamicFrame

 classifyColumns(frame: DynamicFrame,
      entityTypesToDetect: Seq[String],
      sampleFraction: Double = 0.1,
      thresholdFraction: Double = 0.1,
      detectionSensitivity: String = "LOW")
```

Puoi utilizzare l'API `detect()` per identificare i tipi di PII AWS gestite e i tipi di entità personalizzate. Una nuova colonna viene creata automaticamente con il risultato del rilevamento. L'API `classifyColumns()` restituisce una mappa in cui le chiavi sono i nomi delle colonne e i valori sono un elenco di tipi di entità rilevati. `SampleFraction` indica la frazione dei dati da campionare durante la scansione di ricerca delle entità PII, mentre `ThresholdFraction` indica la frazione dei dati che devono essere soddisfatti per identificare una colonna come dati PII.

Rilevamento a livello di riga

Nell'esempio, il processo sta eseguendo le seguenti azioni utilizzando le API `detect()` e `classifyColumns()`:

- leggere i dati da un bucket Amazon S3 e trasformarli in un dynamicFrame
- rilevamento di istanze di “e-mail” e “carta di credito” in dynamicFrame
- restituzione di un dynamicFrame con valori originali più una colonna che include il risultato del rilevamento per ogni riga
- scrivere il dynamicFrame restituito in un altro percorso Amazon S3

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
```

```
 import scala.collection.JavaConverters._
 import com.amazonaws.services.glue.ml.EntityDetector

 object GlueApp {
   def main(sysArgs: Array[String]) {
     val spark: SparkContext = new SparkContext()
     val glueContext: GlueContext = new GlueContext(spark)
     val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
     Job.init(args("JOB_NAME"), glueContext, args.asJava)
     val frame=
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
"withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
transformationContext="AmazonS3_node1650160158526").getDynamicFrame()

     val frameWithDetectedPII = EntityDetector.detect(frame, Seq("EMAIL",
"CREDIT_CARD"))

     glueContext.getSinkWithFormat(connectionType="s3",
options=JsonOptions("""{"path": "s3://pathToOutput/", "partitionKeys": []}"""),
transformationContext="someCtx",
format="json").writeDynamicFrame(frameWithDetectedPII)

     Job.commit()
   }
 }
```

Rilevamento a livello di riga con operazioni granulari

Nell'esempio, il processo sta eseguendo le seguenti azioni utilizzando le API `detect()`:

- leggere i dati da un bucket di Amazon S3 e trasformarli in un dynamicFrame
- rilevamento dei tipi di dati sensibili per “USA_PTIN”, “ BANK_ACCOUNT”, “USA_SSN”, “USA_PASSPORT_NUMBER” e “PHONE_NUMBER” nel dynamicFrame
- restituzione di un dynamicFrame con valori mascherati modificati più una colonna che include il risultato del rilevamento per ogni riga
- scrittura del dynamicFrame restituito in un altro percorso di Amazon S3

A differenza dell'API `detect()` di cui sopra, questa utilizza operazioni granulari per rilevare i tipi di entità. Per ulteriori informazioni, consulta Parametri di rilevamento per l'utilizzo di `detect()`.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.ml.EntityDetector

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    val frame =
 glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
 "withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
 options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
 transformationContext="AmazonS3_node_source").getDynamicFrame()

    val detectionParameters = JsonOptions(
      """
        {
          "USA_DRIVING_LICENSE": [{
            "action": "PARTIAL_REDACT",
            "sourceColumns": ["Driving License"],
            "actionOptions": {
              "matchPattern": "[0-9]",
              "redactChar": "*"
            }
          }],
          "BANK_ACCOUNT": [{
            "action": "DETECT",
            "sourceColumns": ["*"]
          }],
          "USA_SSN": [{
            "action": "SHA256_HASH",
            "sourceColumns": ["SSN"]
          }],
          "IP_ADDRESS": [{
            "action": "REDACT",
```

```
            "sourceColumns": ["IP Address"],
            "actionOptions": {"redactText": "*****"}
          }],
          "PHONE_NUMBER": [{
            "action": "PARTIAL_REDACT",
            "sourceColumns": ["Phone Number"],
            "actionOptions": {
              "numLeftCharsToExclude": 1,
              "numRightCharsToExclude": 0,
              "redactChar": "*"
            }
          }]
        }
      """
    )

    val frameWithDetectedPII = EntityDetector.detect(frame, detectionParameters,
 "DetectedEntities", "HIGH")

    glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path":
 "s3://pathToOutput/", "partitionKeys": []}"""),
 transformationContext="AmazonS3_node_target",
 format="json").writeDynamicFrame(frameWithDetectedPII)

    Job.commit()
  }
}
```

Rilevamento a livello di colonna

Nell'esempio, il processo sta eseguendo le seguenti azioni utilizzando le API `classifyColumns()`:

- leggere i dati da un bucket di Amazon S3 e trasformarli in un dynamicFrame
- rilevamento di istanze di “e-mail” e “carta di credito” in dynamicFrame
- imposta i parametri per campionare il 100% della colonna, contrassegna un'entità come rilevata se si trova nel 10% delle celle e ha una sensibilità "LOW"
- restituisce una mappa in cui le chiavi sono i nomi delle colonne e i valori sono l'elenco dei tipi di entità rilevati
- scrittura del dynamicFrame restituito in un altro percorso di Amazon S3

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.DynamicFrame
import com.amazonaws.services.glue.ml.EntityDetector

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    val frame =
 glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar":
 "\"", "withHeader": true, "separator": ",", "optimizePerformance": false}"""),
 connectionType="s3", format="csv", options=JsonOptions("""{"paths": ["s3://
pathToSource"], "recurse": true}"""), transformationContext="frame").getDynamicFrame()

    import glueContext.sparkSession.implicits._

    val detectedDataFrame = EntityDetector.classifyColumns(
        frame,
        entityTypesToDetect = Seq("CREDIT_CARD", "PHONE_NUMBER"),
        sampleFraction = 1.0,
        thresholdFraction = 0.1,
        detectionSensitivity = "LOW"
    )
    val detectedDF = (detectedDataFrame).toSeq.toDF("columnName", "entityTypes")
    val DetectSensitiveData_node = DynamicFrame(detectedDF, glueContext)

    glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path":
 "s3://pathToOutput", "partitionKeys": []}"""), transformationContext="someCtx",
 format="json").writeDynamicFrame(DetectSensitiveData_node)

    Job.commit()
  }
}
```

Rilevamento di dati sensibili Rilevamento mediante tipi di AWS CustomEntityType PII

È possibile definire entità personalizzate tramite AWS Studio. Tuttavia, per utilizzare questa funzionalità fuori da AWS Studio, devi prima definire i tipi di entità personalizzati e quindi aggiungere i tipi di entità personalizzati definiti all'elenco di `entityTypesToDetect`.

Se hai tipi di dati sensibili specifici nei tuoi dati (come “ID dipendente”), puoi creare entità personalizzate chiamando l'API `CreateCustomEntityType()`. L'esempio seguente definisce il tipo di entità personalizzato 'EMPLOYEE_ID' per l'API `CreateCustomEntityType()` con i parametri della richiesta:

```
{
    "name": "EMPLOYEE_ID",
    "regexString": "\d{4}-\d{3}",
    "contextWords": ["employee"]
}
```

Quindi, modifica il lavoro per utilizzare il nuovo tipo di dati sensibili personalizzato aggiungendo il tipo di entità personalizzato (EMPLOYEE_ID) all'API `EntityDetector()`:

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.ml.EntityDetector

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
```

```
     val frame=
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
"withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
transformationContext="AmazonS3_node1650160158526").getDynamicFrame()

     val frameWithDetectedPII = EntityDetector.detect(frame, Seq("EMAIL",
"CREDIT_CARD", "EMPLOYEE_ID"))

     glueContext.getSinkWithFormat(connectionType="s3",
options=JsonOptions("""{"path": "s3://pathToOutput/", "partitionKeys": []}"""),
transformationContext="someCtx",
format="json").writeDynamicFrame(frameWithDetectedPII)

     Job.commit()
   }
 }
```

> **Note**
>
> Se un tipo di dati sensibili personalizzato è definito con lo stesso nome di un tipo di entità gestita esistente, il tipo di dati sensibili personalizzato avrà la precedenza e sovrascriverà la logica del tipo di entità gestita.

Parametri di rilevamento per l'utilizzo di **`detect()`**

Questo metodo viene utilizzato per rilevare le entità in a. DynamicFrame Ne restituisce una nuova DataFrame con valori originali e una colonna aggiuntiva con outputColumnName metadati di rilevamento PII. Il mascheramento personalizzato può essere eseguito dopo che questo DynamicFrame è stato restituito all'interno dello AWS Glue script, oppure è possibile utilizzare l'API detect () con azioni granulari.

```
detect(frame: DynamicFrame,
       entityTypesToDetect: Seq[String],
       outputColumnName: String = "DetectedEntities",
       detectionSensitivity: String = "LOW"): DynamicFrame
```

Parametri:

- frame — (type:`DynamicFrame`) L'input DynamicFrame contenente i dati da elaborare.
- entityTypesToRileva — (tipo:`[Seq[String]`) Elenco dei tipi di entità da rilevare. Possono essere tipi di entità gestiti o personalizzati.
- outputColumnName— (tipo:`String`, default: "DetectedEntities«) Il nome della colonna in cui verranno memorizzate le entità rilevate. Se non viene fornito, il nome di colonna predefinito è "DetectedEntities».
- detectionSensivity — (tipo: `String`, opzioni: "BASSA" oppure "ELEVATA", impostazione predefinita: "BASSA") specifica la distinzione del processo di rilevamento. Le opzioni valide sono "BASSA" oppure "ELEVATA". Se non viene fornita, la distinzione predefinita è impostata su "LOW".

Impostazioni di `outputColumnName`:

Il nome della colonna in cui verranno memorizzate le entità rilevate. Se non viene fornito, il nome di colonna predefinito è "DetectedEntities». Per ogni riga della colonna di output, la colonna supplementare include una mappa del nome della colonna ai metadati dell'entità rilevata, con le seguenti coppie chiave-valore:

- entityType: il tipo di entità rilevato.
- start — la posizione iniziale dell'entità rilevata nei dati originali.
- end — la posizione finale dell'entità rilevata nei dati originali.
- actionUsed — l'azione eseguita sull'entità rilevata (ad esempio, "DETECT," "REDACT," "PARTIAL_REDACT," "SHA256_HASH").

Esempio:

```
{
   "DetectedEntities":{
      "SSN Col":[
         {
            "entityType":"USA_SSN",
            "actionUsed":"DETECT",
            "start":4,
            "end":15
         }
      ],
      "Random Data col":[
         {
            "entityType":"BANK_ACCOUNT",
```

```
            "actionUsed":"PARTIAL_REDACT",
            "start":4,
            "end":13
         },
         {
            "entityType":"IP_ADDRESS",
            "actionUsed":"REDACT",
            "start":4,
            "end":13
         }
      ]
   }
}
```

Parametri di rilevamento per **`detect()`** con operazioni granulari

Questo metodo viene utilizzato per rilevare le entità in un DynamicFrame utilizzando parametri specificati. Ne restituisce una nuova DataFrame con valori originali sostituiti con dati sensibili mascherati e una colonna aggiuntiva con metadati `outputColumnName` di rilevamento PII.

```
detect(frame: DynamicFrame,
       detectionParameters: JsonOptions,
       outputColumnName: String = "DetectedEntities",
       detectionSensitivity: String = "LOW"): DynamicFrame
```

Parametri:

- frame — (type:`DynamicFrame`): L'input DynamicFrame contenente i dati da elaborare.
- detectionParameters — (tipo: `JsonOptions`): opzioni JSON che specificano i parametri per il processo di rilevamento.
- outputColumnName— (type:`String`, default: "DetectedEntities«): Il nome della colonna in cui verranno memorizzate le entità rilevate. Se non viene fornito, il nome di colonna predefinito è "DetectedEntities».
- detectionSensivity — (tipo: `String`, opzioni: "BASSA" oppure "ELEVATA", impostazione predefinita: "BASSA"): specifica la distinzione del processo di rilevamento. Le opzioni valide sono "BASSA" oppure "ELEVATA". Se non viene fornita, la distinzione predefinita è impostata su "LOW".

Impostazioni di `detectionParameters`

Se non è inclusa alcuna impostazione, verranno utilizzati i valori predefiniti.

- action — (tipo: `String`, opzioni: "DETECT", "REDACT", "PARTIAL_REDACT", "SHA256_HASH") specifica l'operazione da eseguire sull'entità. Obbligatorio. Le operazioni che eseguono il mascheramento (ad eccezione di "DETECT") possono eseguire solo un'operazione per colonna. Si tratta di una misura preventiva per mascherare le entità unite.
- sourceColumns — (tipo: `List[String]`, impostazione predefinita: ["*"]) elenco dei nomi delle colonne di origine su cui eseguire il rilevamento dell'entità. L'impostazione predefinita è ["*"], se non presente. Viene generato `IllegalArgumentException` se viene utilizzato un nome di colonna non valido.
- sourceColumnsToEscludi — (tipo:`List[String]`) Elenco dei nomi delle colonne di origine su cui eseguire il rilevamento dell'entità. Usa `sourceColumns` o `sourceColumnsToExclude`. Viene generato `IllegalArgumentException` se viene utilizzato un nome di colonna non valido.
- actionOptions — opzioni aggiuntive basate sull'operazione specificata:
  - Per "DETECT" e "SHA256_HASH", non sono consentite opzioni.
  - Per "REDACT":
    - redactText — (tipo: `String`, impostazione predefinita: "*****") testo per sostituire l'entità rilevata.
  - Per "PARTIAL_REDACT":
    - redactChar — (tipo: `String`, impostazione predefinita: "*") carattere per sostituire ogni carattere rilevato nell'entità.
    - matchPattern — (tipo: `String`) modello Regex per la redazione parziale. Non può essere combinato con numLeftChars ToExclude o`numRightCharsToExclude`.
    - numLeftCharsToExclude— (tipo:`String, integer`) Numero di caratteri a sinistra da escludere. Non può essere unito a matchPattern, ma può essere utilizzato con `numRightCharsToExclude`.
    - numRightCharsToExclude— (digitare:`String, integer`) Numero di caratteri a destra da escludere. Non può essere unito a matchPattern, ma può essere utilizzato con `numRightCharsToExclude`.

Impostazioni di `outputColumnName`

Vedi outputColumnName le impostazioni

Parametri di rilevamento per **`classifyColumns()`**

Questo metodo viene utilizzato per rilevare le entità in a DynamicFrame. Restituisce una mappa in cui le chiavi sono i nomi delle colonne e i valori sono l'elenco dei tipi di entità rilevati. Il mascheramento personalizzato può essere eseguito dopo che questo è stato restituito all'interno dello script AWS Glue.

```
classifyColumns(frame: DynamicFrame,
                entityTypesToDetect: Seq[String],
                sampleFraction: Double = 0.1,
                thresholdFraction: Double = 0.1,
                detectionSensitivity: String = "LOW")
```

Parametri:

- frame — (tipo:`DynamicFrame`) L'input DynamicFrame contenente i dati da elaborare.
- entityTypesToRileva — (tipo:`Seq[String]`) Elenco dei tipi di entità da rilevare. Possono essere tipi di entità gestiti o personalizzati.
- sampleFraction — (tipo: `Double`, impostazione predefinita: 10%) la frazione dei dati da campionare durante la scansione di entità PII.
- thresholdFraction — (tipo: `Double`, impostazione predefinita: 10%): la frazione dei dati che devono essere soddisfatti per identificare una colonna come dati PII.
- detectionSensivity — (tipo: `String`, opzioni: "BASSA" oppure "ELEVATA", impostazione predefinita: "BASSA") specifica la distinzione del processo di rilevamento. Le opzioni valide sono "BASSA" oppure "ELEVATA". Se non viene fornita, la distinzione predefinita è impostata su "LOW".

Tipi di dati sensibili gestiti

Entità globali

Tipo di dati	Categoria	Descrizione
PERSON_NAME	Universal	Il nome della persona.
EMAIL	Personale	L'indirizzo e-mail.
IP_ADDRESS	Computer	L’indirizzo IP

Tipo di dati	Categoria	Descrizione
MAC_ADDRESS	Personale	L'indirizzo MAC.

Tipi di dati negli Stati Uniti

Tipo di dati	Descrizione
BANK_ACCOUNT	Il numero di conto bancario. Non è specifico per un paese o una regione, tuttavia vengono rilevati solo i formati di conto statunitensi e canadesi.
CREDIT_CARD	Il numero di carta di credito.
PHONE_NUMBER	Il numero di telefono. Non è specifico per un paese o una regione, tuttavia, al momento vengono rilevati solo i numeri di telefono statunitensi e canadesi.
USA_ATIN	Il numero identificativo del contribuente per l'adozione negli Stati Uniti rilasciato dall'Internal Revenue Service.
USA_CPT_CODE	Il codice CPT (specifico per gli Stati Uniti).
USA_DEA_NUMBER	Il numero DEA (specifico per gli Stati Uniti).
USA_DRIVING_LICENSE	Il numero della patente di guida (specifico per gli Stati Uniti).
USA_HCPCS_CODE	Il codice HCPCS (specifico per gli Stati Uniti).
USA_HEALTH_INSURANCE_CLAIM_NUMBER	Il numero di attestazione dell'assicurazione sanitaria (specifico per gli Stati Uniti).
USA_ITIN	L'ITIN (per persone o entità statunitensi).

Tipo di dati	Descrizione
USA_MEDICARE_BENEFICIARY_IDENTIFIER	Il numero identificativo del beneficiario di Medicare (specifico per gli Stati Uniti).
USA_NATIONAL_DRUG_CODE	Il codice NDC (specifico per gli Stati Uniti).
USA_NATIONAL_PROVIDER_IDENTIFIER	Il numero identificativo del fornitore nazionale (specifico per gli Stati Uniti).
USA_PASSPORT_NUMBER	Il numero del passaporto (per i cittadini statunitensi).
USA_PTIN	Il codice di identificazione fiscale US Preparer TIN rilasciato dall'Internal Revenue Service.
USA_SSN	Il numero di previdenza sociale (per i cittadini statunitensi).

Tipi di dati in Argentina

Tipo di dati	Descrizione
ARGENTINA_TAX_IDENTIFICATIO N_NUMBER	Il codice di identificazione fiscale dell'Argentina. Conosciuto anche come CUIT o CUIL.

Tipi di dati in Australia

Tipo di dati	Descrizione
AUSTRALIA_BUSINESS_NUMBER	Australia Business Number (ABN). Un codice identificativo univoco rilasciato dall'Australian Business Register (ABR) per identificare le aziende a livello amministrativo e pubblico.

Tipo di dati	Descrizione
AUSTRALIA_COMPANY_NUMBER	Australia Company Number (ACN). Un codice identificativo univoco rilasciato dalla Australian Securities and Investments Commission.
AUSTRALIA_DRIVING_LICENSE	Il numero della patente di guida per l'Australia.
AUSTRALIA_MEDICARE_NUMBER	Il numero Medicare australiano. Il numero identificativo personale rilasciato dalla Australia n Health Insurance Commission.
AUSTRALIA_PASSPORT_NUMBER	Il numero del passaporto australiano.
AUSTRALIA_TAX_FILE_NUMBER	Il codice fiscale australiano (TFN). Rilasciat o dall'Australian Taxation Office (ATO) ai contribuenti (persone fisiche, società, ecc.) per le finalità fiscali.

Tipi di dati in Austria

Tipo di dati	Descrizione
AUSTRIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per l'Austria).
AUSTRIA_PASSPORT_NUMBER	Il numero del passaporto (specifico per l'Austria).
AUSTRIA_SSN	Il numero di previdenza sociale (per i cittadini austriaci).
AUSTRIA_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per l'Austria).
AUSTRIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per l'Austria).

Tipi di dati nell'area balcanica

Tipo di dati	Descrizione
BOSNIA_UNIQUE_MASTER_CITIZEN_NUMBER	Il codice identificativo unico (JMBG) per i cittadini della Bosnia-Erzegovina.
KOSOVO_UNIQUE_MASTER_CITIZEN_NUMBER	Il codice identificativo unico (JMBG) per il Kosovo.
MACEDONIA_UNIQUE_MASTER_CITIZEN_NUMBER	Il codice identificativo unico per la Macedonia.
MONTENEGRO_UNIQUE_MASTER_CITIZEN_NUMBER	Il codice identificativo unico (JMBG) per il Montenegro.
SERBIA_UNIQUE_MASTER_CITIZEN_NUMBER	Il codice identificativo unico (JMBG) per la Serbia.
SERBIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Serbia).
VOJVODINA_UNIQUE_MASTER_CITIZEN_NUMBER	Il codice identificativo unico (JMBG) per la Vojvodina.

Tipi di dati in Belgio

Tipo di dati	Descrizione
BELGIUM_DRIVING_LICENSE	Il numero della patente di guida (specifico per il Belgio).
BELGIUM_NATIONAL_IDENTIFICATION_NUMBER	Il numero nazionale belga (BNN).
BELGIUM_PASSPORT_NUMBER	Il numero del passaporto (specifico per il Belgio).

Tipo di dati	Descrizione
BELGIUM_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per il Belgio).
BELGIUM_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per il Belgio).

Tipi di dati in Brasile

Tipo di dati	Descrizione
BRAZIL_BANK_ACCOUNT	Il numero di conto bancario (specifico per il Brasile).
BRAZIL_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per il Brasile).
BRAZIL_NATIONAL_REGISTRY_OF_LEGAL_ENTITIES_NUMBER	Il codice di identificazione rilasciato alle società (specifico per il Brasile), noto anche come CNPJ.
BRAZIL_NATURAL_PERSON_REGISTRY_NUMBER	Numero di registro delle persone fisiche, noto anche come CPF.

Tipi di dati in Bulgaria

Tipo di dati	Descrizione
BULGARIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Bulgaria).
BULGARIA_UNIFORM_CIVIL_NUMBER	Il codice di identificazione unificato (EGN) che funge da numero di identificazione nazionale.
BULGARIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Bulgaria).

Tipi di dati in Canada

Tipo di dati	Descrizione
CANADA_DRIVING_LICENSE	Il numero della patente di guida (specifico per il Canada).
CANADA_GOVERNMENT_IDENTIFICATION_CARD_NUMBER	Il numero identificativo nazionale (specifico per il Canada).
CANADA_PASSPORT_NUMBER	Il numero del passaporto (specifico per il Canada).
CANADA_PERMANENT_RESIDENCE_NUMBER	Il numero di residenza permanente (numero di PR Card).
CANADA_PERSONAL_HEALTH_NUMBER	Il codice identificativo univoco per l'assistenza sanitaria (numero PHN).
CANADA_SOCIAL_INSURANCE_NUMBER	Il numero di previdenza sociale (SIN) in Canada.

Tipi di dati in Cile

Tipo di dati	Descrizione
CHILE_DRIVING_LICENSE	Il numero della patente di guida (specifico per il Cile).
CHILE_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale per il Cile, noto anche come RUT o RUN.

Tipi di dati in Cina, Hong Kong, Macao e Taiwan

Tipo di dati	Descrizione
CHINA_IDENTIFICATION	L'identificatore cinese.

Tipo di dati	Descrizione
CHINA_LICENSE_PLATE_NUMBER	Il numero della patente di guida (specifico per la Cina).
CHINA_MAINLAND_TRAVEL_PERMIT_ID_HONG_KONG_MACAU	Il permesso di viaggio sulla terraferma per i residenti di Hong Kong e Macao.
CHINA_MAINLAND_TRAVEL_PERMIT_ID_TAIWAN	Il permesso di viaggio sulla terraferma per i residenti di Taiwan rilasciato dal governo della Repubblica Popolare Cinese (RPC).
CHINA_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Cina).
CHINA_PHONE_NUMBER	Il numero di telefono (specifico per la Cina).
HONG_KONG_IDENTITY_CARD	Il documento di identità ufficiale rilasciato dal Dipartimento dell'Immigrazione di Hong Kong.
MACAU_RESIDENT_IDENTITY_CARD	La Macau Resident Identity Card o BIR è una carta d'identità ufficiale emessa dall'Identification Services Bureau di Macao.
TAIWAN_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per Taiwan).
TAIWAN_PASSPORT_NUMBER	Il numero del passaporto (specifico per Taiwan).

Tipi di dati in Colombia

Tipo di dati	Descrizione
COLOMBIA_PERSONAL_IDENTIFICATION_NUMBER	Il codice identificativo univoco assegnato ai colombiani alla nascita.

Tipo di dati	Descrizione
COLOMBIA_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per la Colombia).

Tipi di dati in Croazia

Tipo di dati	Descrizione
CROATIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Croazia).
CROATIA_IDENTITY_NUMBER	Il numero identificativo nazionale (specifico per la Croazia).
CROATIA_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Croazia).
CROATIA_PERSONAL_IDENTIFICATION_NUMBER	Il codice di identificazione personale (OIB).

Tipi di dati a Cipro

Tipo di dati	Descrizione
CYPRUS_DRIVING_LICENSE	Il numero della patente di guida (specifico per Cipro).
CYPRUS_NATIONAL_IDENTIFICATION_NUMBER	La carta d'identità cipriota.
CYPRUS_PASSPORT_NUMBER	Il numero del passaporto (specifico per Cipro).
CYPRUS_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per Cipro).

Tipo di dati	Descrizione
CYPRUS_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per Cipro).

Tipi di dati in Repubblica Ceca

Tipo di dati	Descrizione
CZECHIA_DRIVING_LICENSE	Numero della patente di guida (specifico per la Repubblica Ceca).
CZECHIA_PERSONAL_IDENTIFICATION_NUMBER	Il codice di identificazione personale (specifico per la Repubblica Ceca).
CZECHIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Repubblica Ceca).

Tipi di dati in Danimarca

Tipo di dati	Descrizione
DENMARK_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Danimarca).
DENMARK_PERSONAL_IDENTIFICATION_NUMBER	Il codice di identificazione personale (specifico per la Danimarca).
DENMARK_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per la Danimarca).
DENMARK_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Danimarca).

Tipi di dati in Estonia

Tipo di dati	Descrizione
ESTONIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per l'Estonia).
ESTONIA_PASSPORT_NUMBER	Il numero del passaporto (specifico per l'Estonia).
ESTONIA_PERSONAL_IDENTIFICATION_CODE	Il codice di identificazione personale (specifico per l'Estonia).
ESTONIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per l'Estonia).

Tipi di dati in Finlandia

Tipo di dati	Descrizione
FINLAND_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Finlandia).
FINLAND_HEALTH_INSURANCE_NUMBER	Il numero dell'assicurazione sanitaria (specifico per la Finlandia).
FINLAND_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per la Finlandia).
FINLAND_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Finlandia).
FINLAND_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Finlandia).

Tipi di dati in Francia

Tipo di dati	Descrizione
FRANCE_BANK_ACCOUNT	Il numero di conto bancario (specifico per la Francia).
FRANCE_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Francia).
FRANCE_HEALTH_INSURANCE_NUMBER	Il numero dell'assicurazione sanitaria in Francia.
FRANCE_INSEE_CODE	Il codice di previdenza sociale, SSN o NIR in Francia.
FRANCE_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (CNI) per la Francia.
FRANCE_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Francia).
FRANCE_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per la Francia).
FRANCE_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Francia).

Tipi di dati in Germania

Tipo di dati	Descrizione
GERMANY_BANK_ACCOUNT	Il numero di conto bancario (specifico per la Germania).
GERMANY_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Germania).
GERMANY_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Germania).

Tipo di dati	Descrizione
GERMANY_PERSONAL_IDENTIFICATION_NUMBER	Il codice di identificazione personale (specifico per la Germania).
GERMANY_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per la Germania).
GERMANY_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Germania).

Tipi di dati in Grecia

Tipo di dati	Descrizione
GREECE_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Grecia).
GREECE_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Grecia).
GREECE_SSN	Il numero di previdenza sociale (per i cittadini greci).
GREECE_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per la Grecia).
GREECE_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Grecia).

Tipi di dati in Ungheria

Tipo di dati	Descrizione
HUNGARY_DRIVING_LICENSE	Il numero della patente di guida (specifico per l'Ungheria).

Tipo di dati	Descrizione
HUNGARY_PASSPORT_NUMBER	Il numero del passaporto (specifico per l'Ungheria).
HUNGARY_SSN	Il numero di previdenza sociale (per i cittadini ungheresi).
HUNGARY_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per l'Ungheria).
HUNGARY_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per l'Ungheria).

Tipi di dati in Islanda

Tipo di dati	Descrizione
ICELAND_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per l'Islanda).
ICELAND_PASSPORT_NUMBER	Il numero del passaporto (specifico per l'Islanda).
ICELAND_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per l'Islanda).

Tipi di dati in India

Tipo di dati	Descrizione
INDIA_AADHAAR_NUMBER	Il codice di identificazione Aadhaar rilasciato dalla Unique Identification Authority of India.
INDIA_PERMANENT_ACCOUNT_NUMBER	il Permanent Account Number (PAN) indiano.

Tipi di dati in Indonesia

Tipo di dati	Descrizione
INDONESIA_IDENTITY_CARD_NUMBER	Il numero identificativo nazionale (specifico per l'Indonesia).

Tipi di dati in Irlanda

Tipo di dati	Descrizione
IRELAND_DRIVING_LICENSE	Il numero della patente di guida (specifico per l'Irlanda).
IRELAND_PASSPORT_NUMBER	Il numero del passaporto (specifico per l'Irlanda).
IRELAND_PERSONAL_PUBLIC_SER VICE_NUMBER	Il numero di servizio pubblico personale (PPS) irlandese.
IRELAND_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per l'Irlanda).
IRELAND_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per l'Irlanda).

Tipi di dati in Israele

Tipo di dati	Descrizione
ISRAEL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per Israele).

Tipi di dati in Italia

Tipo di dati	Descrizione
ITALY_BANK_ACCOUNT	Il numero di conto bancario (specifico per l'Italia).
ITALY_DRIVING_LICENSE	Il numero della patente di guida (specifico per l'Italia).
ITALY_FISCAL_CODE	Il codice identificativo, noto anche come codice fiscale italiano.
ITALY_PASSPORT_NUMBER	Il numero del passaporto (specifico per l'Italia).
ITALY_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per l'Italia).

Tipi di dati in Giappone

Tipo di dati	Descrizione
JAPAN_BANK_ACCOUNT	Il numero di conto bancario in Giappone.
JAPAN_DRIVING_LICENSE	Il numero di patente di guida per il Giappone.
JAPAN_MY_NUMBER	L'identificativo univoco per i cittadini o le società giapponesi utilizzato per l'amministrazione fiscale, l'amministrazione della sicurezza sociale e la risposta alle catastrofi
JAPAN_PASSPORT_NUMBER	Il numero del passaporto giapponese.

Tipi di dati in Corea

Tipo di dati	Descrizione
KOREA_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Corea).

Tipo di dati	Descrizione
KOREA_RESIDENCE_REGISTRATION_NUMBER_FOR_CITIZENS	Il numero di registrazione di residenza coreano per i cittadini.
KOREA_RESIDENCE_REGISTRATION_NUMBER_FOR_FOREIGNERS	Il numero di registrazione di residenza coreano per gli stranieri.

Tipi di dati in Lettonia

Tipo di dati	Descrizione
LATVIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Lettonia).
LATVIA_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Lettonia).
LATVIA_PERSONAL_IDENTIFICATION_NUMBER	Il codice di identificazione personale (specifico per la Lettonia).
LATVIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Lettonia).

Tipi di dati nel Liechtenstein

Tipo di dati	Descrizione
LIECHTENSTEIN_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per il Liechtenstein).
LIECHTENSTEIN_PASSPORT_NUMBER	Il numero del passaporto (specifico per il Liechtenstein).
LIECHTENSTEIN_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico del Liechtenstein).

Tipi di dati in Lituania

Tipo di dati	Descrizione
LITHUANIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Lituania).
LITHUANIA_PERSONAL_IDENTIFICATION_NUMBER	Il codice di identificazione personale (specifico per la Lituania).
LITHUANIA_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per la Lituania).
LITHUANIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Lituania).

Tipi di dati in Lussemburgo

Tipo di dati	Descrizione
LUXEMBOURG_DRIVING_LICENSE	Il numero della patente di guida (specifico per il Lussemburgo).
LUXEMBOURG_NATIONAL_INDIVIDUAL_NUMBER	Il numero identificativo nazionale (specifico per il Lussemburgo).
LUXEMBOURG_PASSPORT_NUMBER	Il numero del passaporto (specifico per il Lussemburgo).
LUXEMBOURG_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per il Lussemburgo).
LUXEMBOURG_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per il Lussemburgo).

Tipi di dati in Malesia

Tipo di dati	Descrizione
MALAYSIA_MYKAD_NUMBER	Il numero identificativo nazionale (specifico per la Malesia).
MALAYSIA_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Malesia).

Tipi di dati a Malta

Tipo di dati	Descrizione
MALTA_DRIVING_LICENSE	Il numero della patente di guida (specifico per Malta).
MALTA_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per Malta).
MALTA_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per Malta).
MALTA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per Malta).

Tipi di dati in Messico

Tipo di dati	Descrizione
MEXICO_CLABE_NUMBER	Codice bancario Mexico CLABE (Clave Bancaria Estandarizada).
MEXICO_DRIVING_LICENSE	Numero della patente di guida (specifico per il Messico).
MEXICO_PASSPORT_NUMBER	Il numero del passaporto (specifico per il Messico).

Tipo di dati	Descrizione
MEXICO_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per il Messico).
MEXICO_UNIQUE_POPULATION_REGISTRY_CODE	La Clave Única de Registro de Población (CURP) o codice di identità del Messico.

Tipi di dati nei Paesi Bassi

Tipo di dati	Descrizione
NETHERLANDS_CITIZEN_SERVICE_NUMBER	Il codice identificativo olandese (BSN, burgerservicenummer).
NETHERLANDS_DRIVING_LICENSE	Il numero della patente di guida (specifico per i Paesi Bassi).
NETHERLANDS_PASSPORT_NUMBER	Il numero del passaporto (specifico per i Paesi Bassi).
NETHERLANDS_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per i Paesi Bassi).
NETHERLANDS_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per i Paesi Bassi).
NETHERLANDS_BANK_ACCOUNT	Il numero di conto bancario (specifico per i Paesi Bassi).

Tipi di dati in Nuova Zelanda

Tipo di dati	Descrizione
NEW_ZEALAND_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Nuova Zelanda).

Tipo di dati	Descrizione
NEW_ZEALAND_NATIONAL_HEALTH_INDEX_NUMBER	Il numero del sistema sanitario nazionale della Nuova Zelanda.
NEW_ZEALAND_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale, noto anche come codice fiscale interno (specifico per la Nuova Zelanda).

Tipi di dati in Norvegia

Tipo di dati	Descrizione
NORWAY_BIRTH_NUMBER	Il numero di identità nazionale norvegese.
NORWAY_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Norvegia).
NORWAY_HEALTH_INSURANCE_NUMBER	Il numero dell'assicurazione sanitaria norvegese.
NORWAY_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per la Norvegia).
NORWAY_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Norvegia).

Tipi di dati nelle Filippine

Tipo di dati	Descrizione
PHILIPPINES_DRIVING_LICENSE	Il numero della patente di guida (specifico per le Filippine).
PHILIPPINES_PASSPORT_NUMBER	Il numero del passaporto (specifico per le Filippine).

Tipi di dati in Polonia

Tipo di dati	Descrizione
POLAND_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Polonia).
POLAND_IDENTIFICATION_NUMBER	Il codice identificativo della Polonia.
POLAND_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Polonia).
POLAND_REGON_NUMBER	Il codice di identificazione REGON, noto anche come numero di identificazione statistica.
POLAND_SSN	Il numero di previdenza sociale (per i cittadini polacchi).
POLAND_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per la Polonia).
POLAND_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Polonia).

Tipi di dati in Portogallo

Tipo di dati	Descrizione
PORTUGAL_DRIVING_LICENSE	Il numero della patente di guida (specifico per il Portogallo).
PORTUGAL_NATIONAL_IDENTIFIC ATION_NUMBER	Il numero identificativo nazionale (specifico per il Portogallo).
PORTUGAL_PASSPORT_NUMBER	Il numero del passaporto (specifico per il Portogallo).
PORTUGAL_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per il Portogallo).

Tipo di dati	Descrizione
PORTUGAL_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per il Portogallo).

Tipi di dati in Romania

Tipo di dati	Descrizione
ROMANIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Romania).
ROMANIA_NUMERICAL_PERSONAL_CODE	Il codice di identificazione personale (specifico per la Romania).
ROMANIA_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Romania).
ROMANIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Romania).

Tipi di dati a Singapore

Tipo di dati	Descrizione
SINGAPORE_DRIVING_LICENSE	Il numero della patente di guida (specifico per Singapore).
SINGAPORE_NATIONAL_REGISTRY _IDENTIFICATION_NUMBER	La carta d'identità nazionale di Singapore.
SINGAPORE_PASSPORT_NUMBER	Il numero del passaporto (specifico per Singapore).
SINGAPORE_UNIQUE_ENTITY_NUMBER	Il numero di entità univoco (UEN) per Singapore.

Tipi di dati in Slovacchia

Tipo di dati	Descrizione
SLOVAKIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Slovacchia).
SLOVAKIA_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per la Slovacchia).
SLOVAKIA_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Slovacchia).
SLOVAKIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Slovacchia).

Tipi di dati in Slovenia

Tipo di dati	Descrizione
SLOVENIA_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Slovenia).
SLOVENIA_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Slovenia).
SLOVENIA_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale (specifico per la Slovenia).
SLOVENIA_UNIQUE_MASTER_CITIZEN_NUMBER	Il codice identificativo unico (JMBG) per i cittadini sloveni.
SLOVENIA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Slovenia).

Tipi di dati in Sudafrica

Tipo di dati	Descrizione
SOUTH_AFRICA_PERSONAL_IDENTIFICATION_NUMBER	Il codice di identificazione personale (specifico per il Sud Africa).

Tipi di dati in Spagna

Tipo di dati	Descrizione
SPAIN_BANK_ACCOUNT	Il numero di conto bancario (specifico per la Spagna).
SPAIN_DNI	La carta d'identità nazionale (Documento Nacional de Identidad) spagnola.
SPAIN_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Spagna).
SPAIN_NIE	Il numero identificativo degli stranieri (specifico per la Spagna), noto anche come NIE.
SPAIN_NIF	Il codice di identificazione fiscale (specifico per la Spagna), noto anche come NIF.
SPAIN_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Spagna).
SPAIN_SSN	Il numero di previdenza sociale (per i cittadini spagnoli).
SPAIN_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Spagna).

Tipi di dati nello Sri Lanka

Tipo di dati	Descrizione
SRI_LANKA_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per lo Sri Lanka).

Tipi di dati in Svezia

Tipo di dati	Descrizione
SWEDEN_DRIVING_LICENSE	Il numero della patente di guida (specifico per la Svezia).
SWEDEN_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Svezia).
SWEDEN_PERSONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per la Svezia).
SWEDEN_TAX_IDENTIFICATION_NUMBER	Il codice di identificazione fiscale per la Svezia (personnummer).
SWEDEN_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Svezia).

Tipi di dati in Svizzera

Tipo di dati	Descrizione
SWITZERLAND_AHV	Il numero di previdenza sociale per i cittadini svizzeri (AHV).
SWITZERLAND_HEALTH_INSURANCE_NUMBER	Il numero dell'assicurazione sanitaria svizzera.
SWITZERLAND_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Svizzera).

Tipo di dati	Descrizione
SWITZERLAND_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Svizzera).

Tipi di dati in Thailandia

Tipo di dati	Descrizione
THAILAND_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Thailandia).
THAILAND_PERSONAL_IDENTIFICATION_NUMBER	Il codice di identificazione personale (specifico per la Thailandia).

Tipi di dati in Turchia

Tipo di dati	Descrizione
TURKEY_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per la Turchia).
TURKEY_PASSPORT_NUMBER	Il numero del passaporto (specifico per la Turchia).
TURKEY_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per la Turchia).

Tipi di dati in Ucraina

Tipo di dati	Descrizione
UKRAINE_INDIVIDUAL_IDENTIFICATION_NUMBER	Il codice identificativo univoco (specifico per l'Ucraina).

Tipo di dati	Descrizione
UKRAINE_PASSPORT_NUMBER_DOMESTIC	Il numero del passaporto nazionale (specifico per l'Ucraina).
UKRAINE_PASSPORT_NUMBER_INTERNATIONAL	Il numero del passaporto internazionale (specifico per l'Ucraina).

Tipi di dati negli Emirati Arabi Uniti (EAU)

Tipo di dati	Descrizione
UNITED_ARAB_EMIRATES_PERSONAL_NUMBER	Il codice di identificazione personale (specifico per gli Emirati Arabi Uniti).

Tipi di dati nel Regno Unito

Tipo di dati	Descrizione
UK_BANK_ACCOUNT	Il numero di conto bancario del Regno Unito (UK).
UK_BANK_SORT_CODE	Il codice di ordinamento bancario del Regno Unito (UK). I codici di ordinamento sono codici bancari utilizzati per indirizzare i trasferimenti di denaro tra le banche nei rispettivi paesi tramite le rispettive organizzazioni di liquidazione.
UK_DRIVING_LICENSE	Il numero della patente di guida per il Regno Unito di Gran Bretagna e Irlanda del Nord (specifico per il Regno Unito)
UK_ELECTORAL_ROLL_NUMBER	Il numero di iscrizione alle liste elettorali (ERN) è il numero identificativo rilasciato a una persona per la registrazione delle elezioni nel Regno Unito. Il formato di questo numero

Tipo di dati	Descrizione
	è specificato dagli standard governativi del Regno Unito del Gabinetto del Regno Unito.
UK_NATIONAL_HEALTH_SERVICE_NUMBER	Il numero del National Health Service (NHS) è il numero unico assegnato a un utente registrato di servizi sanitari pubblici nel Regno Unito.
UK_NATIONAL_INSURANCE_NUMBER	Il numero di previdenza nazionale (NINO) è un numero utilizzato nel Regno Unito (Regno Unito) per identificare una persona per il programma assicurativo nazionale o il sistema di sicurezza sociale. Talvolta è denominata NO NI o NINO.
UK_PASSPORT_NUMBER	Il numero di passaporto del Regno Unito (UK).
UK_UNIQUE_TAXPAYER_REFERENC E_NUMBER	Il numero di riferimento unico del contribue nte (UTR) del Regno Unito (UK). Un identific atore utilizzato dal governo del Regno Unito per gestire il sistema fiscale.
UK_VALUE_ADDED_TAX	L'IVA è un'imposta sui consumi a carico del consumatore finale. L'IVA viene pagata per ogni transazione nel processo di produzione e distribuzione. Per il Regno Unito, il numero di partita IVA è rilasciato dall'ufficio IVA della regione in cui è stabilita l'azienda.
UK_PHONE_NUMBER	Il numero di telefono del Regno Unito (UK).

Tipi di dati in Venezuela

Tipo di dati	Descrizione
VENEZUELA_DRIVING_LICENSE	Il numero della patente di guida (specifico per il Venezuela).

Tipo di dati	Descrizione
VENEZUELA_NATIONAL_IDENTIFICATION_NUMBER	Il numero identificativo nazionale (specifico per il Venezuela).
VENEZUELA_VALUE_ADDED_TAX	L'imposta sul valore aggiunto (specifica per il Venezuela).

Utilizzo del rilevamento dei dati sensibili granulari

> **Note**
>
> Le azioni granulari sono disponibili solo in AWS Glue 3.0 e 4.0. Ciò include l'esperienza AWS Glue Studio. Inoltre, le modifiche persistenti del log di audit non sono disponibili nella versione 2.0.
> Per tutti i processi visivi AWS Glue Studio 3.0 e 4.0, verrà creato uno script che utilizza automaticamente API di azioni granulari.

La trasformazione relativa al rilevamento dei dati sensibili fornisce la possibilità di rilevare, mascherare o rimuovere le entità definite o che sono predefinite da AWS Glue. Le azioni granulari consentono inoltre di applicare un'azione specifica per entità. I vantaggi aggiuntivi includono:

- Prestazioni migliorate in quanto le azioni vengono applicate non appena vengono rilevati i dati.
- Possibilità di includere o escludere colonne specifiche.
- La possibilità di utilizzare il mascheramento parziale. Questo consente di mascherare parzialmente le entità di dati sensibili rilevate, anziché mascherare l'intera stringa. Sono supportati sia i parametri semplici con offset che regex.

Di seguito, sono riportati frammenti di codice di API di rilevamento dei dati sensibili e delle azioni granulari utilizzate nei processo di esempio a cui si fa riferimento nella sezione successiva.

Rilevare l'API: le azioni granulari utilizzano il nuovo parametro `detectionParameters`:

```
def detect(
    frame: DynamicFrame,
    detectionParameters: JsonOptions,
```

```
    outputColumnName: String = "DetectedEntities",
    detectionSensitivity: String = "LOW"
): DynamicFrame = {}
```

Utilizzo di API di rilevamento dei dati sensibili con azioni granulari

Le API di rilevamento dei dati sensibili che utilizzano rilevamento analizzano i dati forniti, determinano se le righe o le colonne sono tipi di entità di dati sensibili ed eseguono le azioni specificate dall'utente per ogni tipo di entità.

Utilizzo di API di rilevamento con azioni granulari

Utilizza l'API rilevamento e specifica e `outputColumnName` e `detectionParameters`.

```
   object GlueApp {
     def main(sysArgs: Array[String]) {

       val spark: SparkContext = new SparkContext()
       val glueContext: GlueContext = new GlueContext(spark)

       // @params: [JOB_NAME]
       val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
       Job.init(args("JOB_NAME"), glueContext, args.asJava)

       // Script generated for node S3 bucket. Creates DataFrame from data stored in
S3.
       val S3bucket_node1 =
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar":
"\"", "withHeader": true, "separator": ",", "optimizePerformance": false}"""),
connectionType="s3", format="csv", options=JsonOptions("""{"paths":
["s3://189657479688-ddevansh-pii-test-bucket/tiny_pii.csv"], "recurse": true}"""),
transformationContext="S3bucket_node1").getDynamicFrame()

       // Script generated for node Detect Sensitive Data. Will run detect API for the
DataFrame
       // detectionParameter contains information on which EntityType are being
detected
       // and what actions are being applied to them when detected.
       val DetectSensitiveData_node2 = EntityDetector.detect(
           frame = S3bucket_node1,
           detectionParameters = JsonOptions(
```

```
            """
               {
                   "PHONE_NUMBER": [
                       {
                           "action": "PARTIAL_REDACT",
                           "actionOptions": {
                               "numLeftCharsToExclude": "3",
                               "numRightCharsToExclude": "4",
                               "redactChar": "#"
                           },
                           "sourceColumnsToExclude": [ "Passport No", "DL NO#" ]
                       }
                   ],
                   "USA_PASSPORT_NUMBER": [
                       {
                           "action": "SHA256_HASH",
                           "sourceColumns": [ "Passport No" ]
                       }
                   ],
                   "USA_DRIVING_LICENSE": [
                       {
                           "action": "REDACT",
                           "actionOptions": {
                               "redactText": "USA_DL"
                           },
                           "sourceColumns": [ "DL NO#" ]
                       }
                   ]

               }
            """
            ),
            outputColumnName = "DetectedEntities"
        )

        // Script generated for node S3 bucket. Store Results of detect to S3 location
        val S3bucket_node3 = glueContext.getSinkWithFormat(connectionType="s3",
 options=JsonOptions("""{"path": "s3://189657479688-ddevansh-pii-test-bucket/
test-output/", "partitionKeys": []}"""), transformationContext="S3bucket_node3",
 format="json").writeDynamicFrame(DetectSensitiveData_node2)

        Job.commit()
      }
```

Lo script precedente creerà un DataFrame da una posizione in Amazon S3 e quindi eseguirà l'API `detect`. Poiché l'API `detect`richiede che il campo `detectionParameters` (una mappa del nome dell'entità su un elenco di tutte le impostazioni di azione da utilizzare per quell'entità) sia rappresentato dall'oggetto `JsonOptions` di AWS Glue, ci consentirà anche di estendere la funzionalità dell'API.

Per ogni azione specificata per entità, inserisci un elenco di tutti i nomi di colonna a cui applicare la combinazione entità/azione. Questo consente di personalizzare le entità da rilevare per ogni colonna del set di dati e ignorare le entità che non si trovano in una colonna specifica. Questo consente inoltre di aumentare le prestazioni dei processi, evitando di eseguire chiamate di rilevamento non necessarie a tali entità, e consente di eseguire azioni uniche per ogni combinazione di colonne ed entità.

Esaminando più da vicino `detectionParameters`, ci sono tre tipi di entità nel processo di esempio. Questi sono `Phone Number`, `USA_PASSPORT_NUMBER` e `USA_DRIVING_LICENSE`. Per ognuno di questi tipi di entità, AWS Glue eseguirà diverse azioni che sono `PARTIAL_REDACT`, `SHA256_HASH` ,`REDACT` e `DETECT`. A ciascuno dei tipi di entità deve essere applicato il valore `sourceColumns` e/o `sourceColumnsToExclude` se rilevato.

> **Note**
>
> È possibile utilizzare una sola azione di modifica sul posto (`PARTIAL_REDACT`, `SHA256_HASH` o `REDACT`) per colonna, ma l'azione `DETECT` può essere utilizzata con ognuna di queste azioni.

Il campo `detectionParameters` ha il seguente layout:

```
ENTITY_NAME -> List[Actions]
{
 "ENTITY_NAME": [{
  Action, // required
  ColumnSpecs,
  ActionOptionsMap
    }],
    "ENTITY_NAME2": [{
  ...
    }]
```

```
    }
```

I tipi di `actions` e `actionOptions` sono elencati di seguito:

```
DETECT
{
    # Required
    "action": "DETECT",
    # Optional, depending on action chosen
    "actionOptions": {
        // There are no actionOptions for DETECT
    },
    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}

SHA256_HASH
{
    # Required
    "action": "SHA256_HASH",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // There are no actionOptions for SHA256_HASH
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}

REDACT
{
    # Required
```

```
    "action": "REDACT",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // The text that is being replaced
        "redactText": "USA_DL"
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}

PARTIAL_REDACT
{
    # Required
    "action": "PARTIAL_REDACT",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // number of characters to not redact from the left side
        "numLeftCharsToExclude": "3",
        // number of characters to not redact from the right side
        "numRightCharsToExclude": "4",
        // the partial redact will be made with this redacted character
        "redactChar": "#",
        // regex pattern for partial redaction
        "matchPattern": "[0-9]"
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}
```

Una volta eseguito lo script, i risultati vengono inviati alla posizione Amazon S3 specificata. Puoi visualizzare i dati in Amazon S3 ma con i tipi di entità selezionati che vengono sensibilizzati in base all'azione selezionata. Nel caso, avremmo una riga simile a questa:


```
{
    "Name": "Colby Schuster",
    "Address": "39041 Antonietta Vista, South Rodgerside, Nebraska 24151",
    "Car Owned": "Fiat",
    "Email": "Kitty46@gmail.com",
    "Company": "O'Reilly Group",
    "Job Title": "Dynamic Functionality Facilitator",
    "ITIN": "991-22-2906",
    "Username": "Cassandre.Kub43",
    "SSN": "914-22-2906",
    "DOB": "2020-08-27",
    "Phone Number": "1-2#######1718",
    "Bank Account No": "69741187",
    "Credit Card Number": "6441-6289-6867-2162-2711",
    "Passport No": "94f311e93a623c72ccb6fc46cf5f5b0265ccb42c517498a0f27fd4c43b47111e",
    "DL NO#": "USA_DL"
}
```


Nello script precedente, `Phone Number` è stato parzialmente redatto con #. `Passport No` è stato modificato in un hash SHA256. `DL NO#` è stato rilevato come numero di patente di guida USA ed è stato redatto in "USA_DL" proprio come indicato in `detectionParameters`.

> **Note**
>
> L'API classifyColumns non è disponibile per l'uso con azioni dettagliate a causa della natura dell'API. L'API esegue il campionamento delle colonne (regolabile dall'utente ma con valori predefiniti) per eseguire il rilevamento più rapidamente. Per questo motivo, le azioni dettagliate richiedono l'iterazione su ogni valore.

Log di audit persistente

Una nuova funzionalità introdotta con azioni granulari (ma disponibile anche quando si utilizzano le normali API) è la presenza di un log di audit persistente. Attualmente, l'esecuzione dell'API di rilevamento aggiunge una colonna aggiuntiva (impostazione predefinita su `DetectedEntities`

ma personalizzabile tramite `outputColumnName`) con i metadati di rilevamento PII. Questo ora ha una chiave di metadati "actionUsed", che è una tra `DETECT`, `PARTIAL_REDACT`, `SHA256_HASH` e `REDACT`.

```
"DetectedEntities": {
    "Credit Card Number": [
        {
            "entityType": "CREDIT_CARD",
            "actionUsed": "DETECT",
            "start": 0,
            "end": 19
        }
    ],
    "Phone Number": [
        {
            "entityType": "PHONE_NUMBER",
            "actionUsed": "REDACT",
            "start": 0,
            "end": 14
        }
    ]
}
```

Anche i clienti che utilizzano le API senza azioni granulari, ad esempio `detect(entityTypesToDetect, outputColumnName)`, vedranno questo log di audit persistente nel dataframe risultante.

I clienti che utilizzano API con azioni granulari vedranno tutte le azioni, indipendentemente dal fatto che siano state oscurate o meno. Esempio:

```
+--------------------+---------------
+-------------------------------------------------------------------------------------------------
+
| Credit Card Number  |  Phone Number  |
                                              DetectedEntities
                                                                  |
+--------------------+---------------
+-------------------------------------------------------------------------------------------------
+
| 622126741306XXXX    | +12#####7890   | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":16}],"Phone
```

```
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":12}]}} |
| 6221 2674 1306 XXXX | +12#######7890 | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":19}],"Phone
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":14}]}} |
| 6221-2674-1306-XXXX | 22#######7890  | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":19}],"Phone
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":14}]}} |
+--------------------+---------------
+-------------------------------------------------------------------------------------------------
+
```

Se non desideri visualizzare la colonna DetectedEntities, puoi semplicemente inserire la colonna aggiuntiva in uno script personalizzato.

API Visual Job di AWS Glue

AWS Glue fornisce un'API che permette ai clienti di creare processi di integrazione dei dati utilizzando l'API AWS Glue di un oggetto JSON che rappresenta un DAG. I clienti possono quindi utilizzare l'editor visivo in AWS Glue Studio per lavorare con questi processi.

Per ulteriori informazioni sui tipi di dati di Visual Job API, consulta API Visual Job.

Argomenti

- API di progettazione e API CRUD
- Nozioni di base
- Limitazioni Visual job

API di progettazione e API CRUD

Le API CreateJob e UpdateJob ora supportano un parametro opzionale aggiuntivo, codeGenConfigurationNodes. Fornire una struttura json non vuota per questo campo comporterà la registrazione del gruppo di disponibilità del database in AWS Glue Studio per il processo creato e per il codice associato generato. Un valore nullo o una stringa vuota per questo campo durante la creazione di processi verrà ignorato.

Gli aggiornamenti del campo codeGenConfigurationNodes verranno eseguiti tramite l'API UpdateJob AWS Glue in modo simile a quello di CreateJob. L'intero campo deve essere specificato in UpdateJob in cui il DAG è stato modificato come desiderato. Un valore nullo fornito verrà ignorato e non verrà eseguito alcun aggiornamento del DAG. Una struttura o una stringa vuota causerà l'impostazione di codeGenConfigurationNodes come vuota e la rimozione di qualsiasi DAG precedente. L'API GetJob restituirà un gruppo di disponibilità dei database se esistente. L'API DeleteJob eliminerà anche qualsiasi DAG associato.

Nozioni di base

Per creare un processo, utilizza l'operazione CreateJob. La richiesta `CreateJob` avrà un campo aggiuntivo "CodeGenConfigurationNodes" in cui sarà possibile specificare l'oggetto DAG in JSON.

Cose da tenere a mente:

- Il campo "codegenConfigurationNodes" è una mappa di nodeID al nodo.
- Ciascun nodo inizia con una chiave che ne identifica il tipo.
- È possibile specificare una sola chiave, poiché un nodo può essere di un solo tipo.
- Il campo di input contiene i nodi padre del nodo corrente.

Di seguito è riportata una rappresentazione JSON di un input CreateJob.

```
{
  "node-1": {
    "S3CatalogSource": {
      "Table": "csvFormattedTable",
      "PartitionPredicate": "",
      "Name": "S3 bucket",
      "AdditionalOptions": {},
      "Database": "myDatabase"
    }
  },
  "node-3": {
    "S3DirectTarget": {
      "Inputs": ["node-2"],
      "PartitionKeys": [],
      "Compression": "none",
      "Format": "json",
      "SchemaChangePolicy": { "EnableUpdateCatalog": false },
      "Path": "",
```

```
      "Name": "S3 bucket"
    }
  },
  "node-2": {
    "ApplyMapping": {
      "Inputs": ["node-1"],
      "Name": "ApplyMapping",
      "Mapping": [
        {
          "FromType": "long",
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader1",
          "FromPath": ["myheader1"]
        },
        {
          "FromType": "long",
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader2",
          "FromPath": ["myheader2"]
        },
        {
          "FromType": "long",
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader3",
          "FromPath": ["myheader3"]
        }
      ]
    }
  }
}
```

Aggiornamento e acquisizione di processi

Poiché UpdateJob avrà anche un campo 'codeGenConfigurationNodes', il formato di input sarà lo stesso. Consulta l'operazione UpdateJob.

L'operazione GetJob restituirà anche un campo "codeGenConfigurationNodes" nello stesso formato. Consulta l'operazione GetJob.

Limitazioni Visual job

Poiché il parametro "codeGenConfigurationNodes" è stato aggiunto alle API esistenti, tutte le limitazioni di tali API verranno ereditate. Inoltre, il codeGenConfigurationNodes e alcuni nodi saranno di dimensioni limitate. Per ulteriori informazioni, consulta Struttura processo.

Script di programmazione Ray

AWS Glue semplifica la scrittura e l'esecuzione di script Ray. Questa sezione descrive le funzionalità Ray supportate disponibili in AWS Glue per Ray. Gli script Ray vengono programmati in Python.

Lo script personalizzato deve essere compatibile con la versione di Ray definita dal campo `Runtime` nella definizione del processo. Per ulteriori informazioni sui `Runtime` nell'API Processi, consulta la pagina the section called “Processi”. Per ulteriori informazioni su ciascun ambiente di runtime, consulta la pagina the section called “Ambienti di runtime Ray supportati”.

Argomenti

- Tutorial: scrittura di uno script ETL in AWS Glue per Ray
- Utilizzo di Ray Core e Ray Data in AWS Glue per Ray
- Fornitura di file e librerie Python ai processi Ray
- Connessione ai dati nei processi Ray

Tutorial: scrittura di uno script ETL in AWS Glue per Ray

Ray ti dà la possibilità di scrivere e dimensionare attività distribuite in modo nativo in Python. AWS Glue per Ray offre ambienti Ray serverless a cui è possibile accedere sia dai processo sia dalle sessioni interattive. Le sessioni interattive di Ray sono disponibili in anteprima. Il sistema di processo AWS Glue fornisce un modo ottimizzato per gestire ed eseguire le tue attività attraverso una pianificazione, da un trigger o dalla console AWS Glue.

La combinazione di questi strumenti di AWS Glue crea una potente toolchain che puoi utilizzare per i carichi di lavoro di estrazione, trasformazione e caricamento (ETL), un caso d'uso molto comune per AWS Glue. Questo tutorial ti illustrerà le basi per creare questa soluzione.

Supportiamo anche l'utilizzo di AWS Glue per Spark per i tuoi carichi di lavoro ETL. Per un tutorial sulla scrittura di script AWS Glue per Spark, consulta la pagina the section called “Tutorial: scrittura di

uno script Spark". Per ulteriori informazioni sui motori disponibili, consulta la pagina the section called "AWS Glue per Spark e AWS Glue per Ray". Ray è in grado di affrontare vari tipi di attività nell'ambito dell'analisi, del machine learning (ML) e dello sviluppo di applicazioni.

In questo tutorial, estrarrai, trasformerai e caricherai un set di dati CSV ospitato in Amazon Simple Storage Service (Amazon S3). Inizierai con il set di dati dei dati di record di viaggio della New York City Taxi and Limousine Commission (TLC), archiviato in un bucket Amazon S3 pubblico. Per ulteriori informazioni su questo set di dati, consulta il Registry of Open Data su AWS.

Trasformerai i tuoi dati con le trasformazioni predefinite disponibili nella libreria Ray Data. Ray Data è una libreria per la preparazione di set di dati progettata da Ray e inclusa per impostazione predefinita negli ambienti AWS Glue per Ray. Per ulteriori informazioni sulle librerie incluse in modo predefinito, consulta la pagina the section called "Moduli disponibili con i processi Ray". Potrai quindi scrivere i dati trasformati in un bucket Amazon S3 da te controllato.

Prerequisiti: per questo tutorial, è necessario disporre di un account AWS con accesso ad AWS Glue e Amazon S3.

Passaggio 1: creazione di un bucket in Amazon S3 per contenere i dati di output

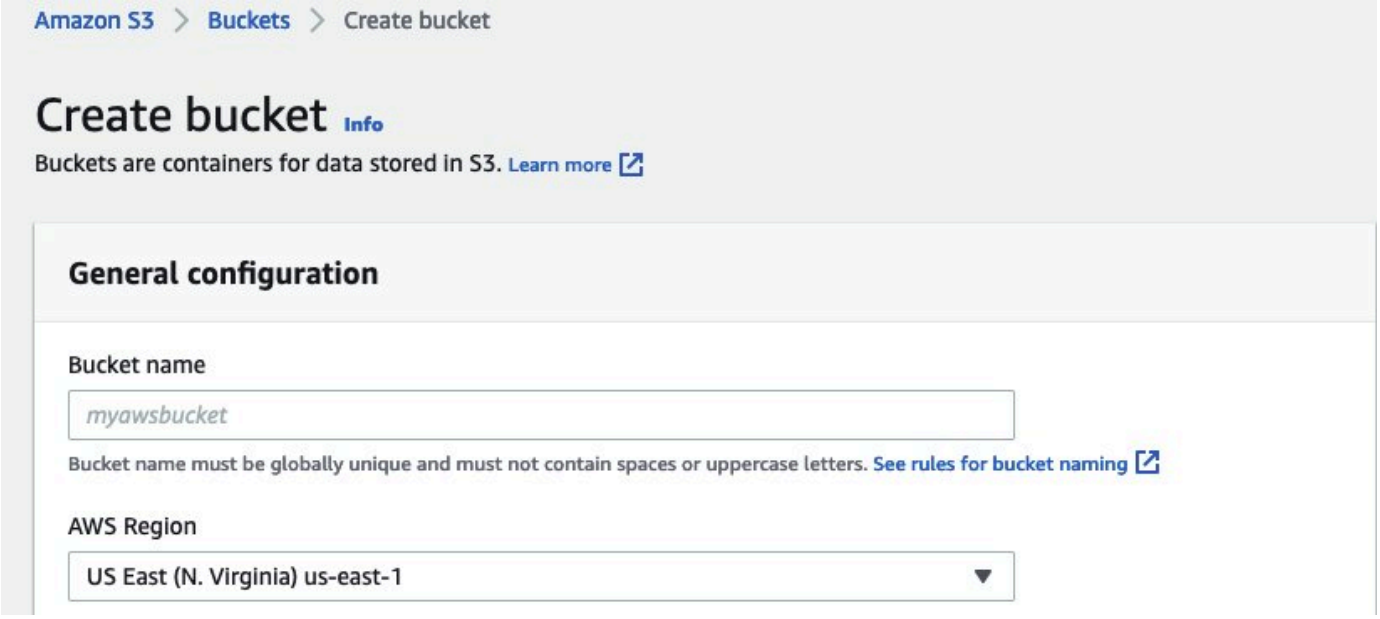
Avrai bisogno di un bucket Amazon S3 da te controllato che funga da sink per i dati creati in questo tutorial. È possibile creare questo bucket con la procedura seguente.

> **Note**
>
> Se desideri scrivere i tuoi dati in un bucket esistente sotto il tuo controllo, puoi saltare questo passaggio. Prendi nota di *yourBucketName*, il nome del bucket esistente, da utilizzare nei passaggi successivi.

Creazione di un bucket per l'output del processo Ray

- Crea un bucket seguendo i passaggi descritti in Creating a bucket nella Guida per l'utente di Amazon S3.
  - Quando scegli un nome per il bucket, prendi nota di *yourBucketName*, al quale farai riferimento nei passaggi successivi.
  - Per altre configurazioni, le impostazioni suggerite fornite nella console Amazon S3 dovrebbero funzionare correttamente in questo tutorial.

Ad esempio, la finestra di dialogo per la creazione del bucket potrebbe avere questo aspetto nella console Amazon S3.



Passaggio 2: creazione di un ruolo e una policy IAM per il processo Ray

Il tuo processo richiederà un ruolo AWS Identity and Access Management (IAM) dotato di quanto segue:

- Autorizzazioni concesse dalla policy gestita da `AWSGlueServiceRole`. Queste sono le autorizzazioni di base necessarie per eseguire un processo AWS Glue.
- Autorizzazioni `Read` a livello di accesso per la risorsa `nyc-tlc/*` Amazon S3.
- Autorizzazioni `Write` a livello di accesso per la risorsa *`yourBucketName`*`/*` Amazon S3.
- Una relazione di fiducia che consente al principale `glue.amazonaws.com` di assumere il ruolo.

È possibile creare questo ruolo con la procedura seguente.

Creazione di un ruolo IAM per il processo AWS Glue per Ray

> **Note**
>
> È possibile creare un ruolo IAM seguendo molte procedure diverse. Per ulteriori informazioni o opzioni su come effettuare il provisioning delle risorse IAM, consulta la documentazione di AWS Identity and Access Management.

1. Crea una policy che definisca le autorizzazioni Amazon S3 precedentemente delineate seguendo i passaggi descritti in Creating IAM policies (console) with the visual editor nella Guida per l'utente di IAM.

   - Quando selezioni un servizio, scegli Amazon S3.
   - Quando selezioni le autorizzazioni per la tua policy, collega i seguenti set di operazioni per le seguenti risorse (menzionate in precedenza):
     - Autorizzazioni con livello di accesso di lettura per la risorsa `nyc-tlc/*` Amazon S3.
     - Autorizzazioni con livello di accesso di scrittura per la risorsa *`yourBucketName`*`/*` di Amazon S3.
   - Quando scegli un nome per la policy, prendi nota di *`YourPolicyName`*, al quale farai riferimento nei passaggi successivi.
2. Crea un ruolo per il processo AWS Glue per Ray seguendo i passaggi descritti nella sezione Creating a role for an AWS service (console) nella Guida per l'utente di IAM.

   - Quando selezioni un'entità di servizio AWS attendibile, scegli `Glue`. Questo creerà automaticamente la relazione di attendibilità necessaria per il processo.
   - Quando selezioni le policy per la policy delle autorizzazioni, collega le seguenti policy:
     - `AWSGlueServiceRole`
     - *`YourPolicyName`*
   - Quando scegli il nome del ruolo, prendi nota di *`YourRoleName`*, al quale farai riferimento nei passaggi successivi.

Passaggio 3: creazione ed esecuzione di un processo AWS Glue per Ray

In questo passaggio, crei un processo AWS Glue utilizzando la AWS Management Console, fornisci uno script di esempio ed esegui il processo. Quando crei un processo, nella console viene creato

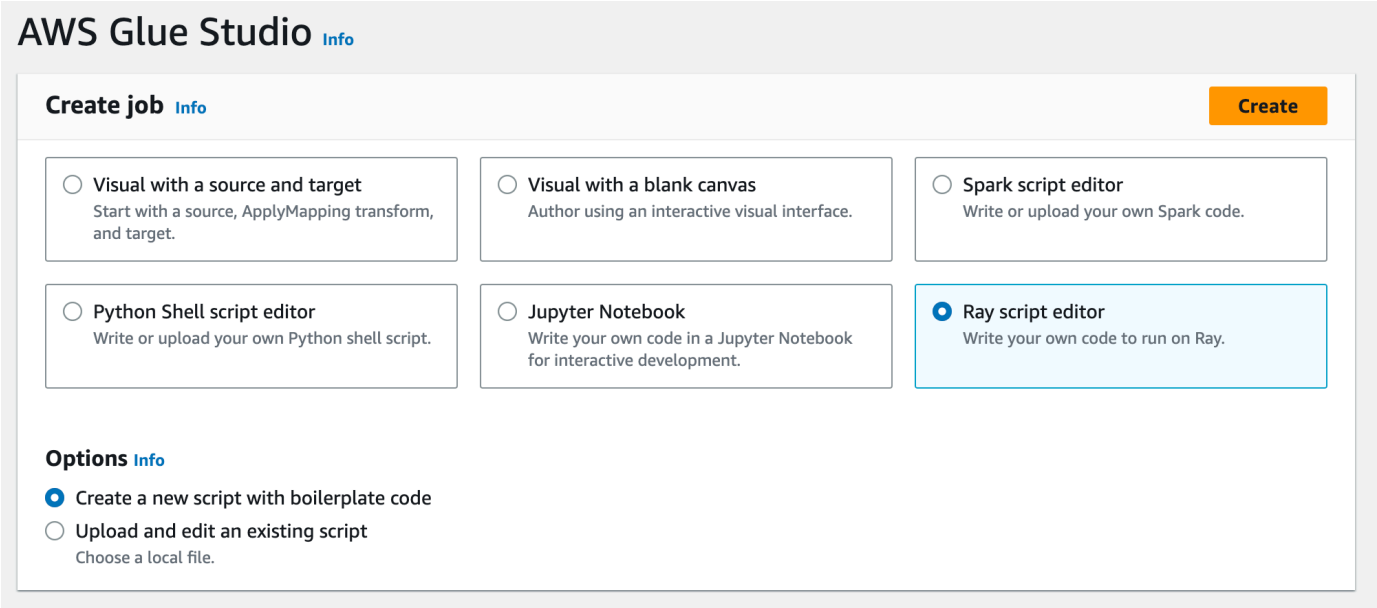
uno spazio in cui archiviare, configurare e modificare lo script Ray. Per informazioni su come creare i processi, consulta the section called “Accesso alla console ”.

In questo tutorial, affrontiamo il seguente scenario ETL: hai la necessità di leggere i record di Gennaio 2022 dal set di dati di New York City TLC Trip Record, aggiungere una nuova colonna (`tip_rate`) al set di dati combinando i dati nelle colonne esistenti, rimuovere una certa quantità di colonne che non sono rilevanti per la tua analisi corrente e quindi scrivere i risultati su *`yourBucketName`*. Il seguente script Ray esegue questi passaggi:

```
import ray
import pandas
from ray import data

ray.init('auto')

ds = ray.data.read_csv("s3://nyc-tlc/opendata_repo/opendata_webconvert/yellow/
yellow_tripdata_2022-01.csv")

# Add the given new column to the dataset and show the sample record after adding a new
 column
ds = ds.add_column( "tip_rate", lambda df: df["tip_amount"] / df["total_amount"])

# Dropping few columns from the underlying Dataset
ds = ds.drop_columns(["payment_type", "fare_amount", "extra", "tolls_amount",
 "improvement_surcharge"])

ds.write_parquet("s3://yourBucketName/ray/tutorial/output/")
```

Creazione ed esecuzione di un processo AWS Glue per Ray

1. Nella AWS Management Console, accedi alla pagina di destinazione di AWS Glue.
2. Nel riquadro di navigazione laterale, scegli Processi ETL.
3. In Crea processo, scegli Ray script editor, quindi scegli Crea, come nella figura seguente.



4. Incolla il testo completo dello script nel riquadro Script e sostituisci l'eventuale testo presente.
5. Vai ai Dettagli del processo e imposta la proprietà Ruolo IAM su *`YourRoleName`*.
6. Seleziona Salva, quindi scegli Esegui.

Passaggio 4: ispezione dell'output

Dopo aver eseguito il processo AWS Glue, è necessario verificare che l'output corrisponda alle aspettative di questo scenario. È possibile farlo con la seguente procedura.

Verifica della corretta esecuzione del processo Ray

1. Nella pagina dei dettagli del processo, vai a Esecuzioni.
2. Dopo alcuni minuti, dovresti vedere un'esecuzione con lo Stato di esecuzione impostato su Operazione riuscita.
3. Accedi alla console Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/ e controlla *`yourBucketName`*. Dovresti visualizzare i file scritti nel tuo bucket di output.
4. Leggi i file Parquet e verificane il contenuto. Puoi farlo utilizzando gli strumenti esistenti. Se non disponi di un processo per la convalida dei file Parquet, puoi farlo nella console AWS Glue con una sessione interattiva AWS Glue, utilizzando Spark o Ray (in anteprima).

   In una sessione interattiva, hai accesso alle librerie Ray Data, Spark o pandas, fornite per impostazione predefinita (in base al motore scelto). Per verificare i contenuti del tuo file, è possibile utilizzare i metodi di ispezione comuni disponibili in tali librerie, ad esempio `count`,

`schema` e `show`. Per ulteriori informazioni sulle sessioni interattive nella console, consulta la pagina Using notebooks with AWS Glue Studio and AWS Glue.

Poiché hai confermato che i file sono stati scritti nel bucket, puoi affermare con relativa certezza che se l'output presenta problemi, non sono correlati alla configurazione IAM. Configura la sessione con *`yourRoleName`* per avere accesso ai file pertinenti.

Se non vedi i risultati previsti, esamina i contenuti per la risoluzione dei problemi in questa guida per identificare e correggere l'origine dell'errore. Per interpretare gli stati di errore durante l'esecuzione del processo, consulta la pagina the section called “Stati di esecuzione dei processi”. Puoi trovare i contenuti relativi alla risoluzione dei problemi nel capitolo Risoluzione dei problemi relativi a AWS Glue. Per errori specifici relativi ai processi Ray, consulta la sezione the section called “Risoluzione degli errori relativi ai processi Ray” nel capitolo sulla risoluzione dei problemi.

Passaggi successivi

Ora hai visto ed eseguito un processo ETL utilizzando AWS Glue per Ray dall'inizio alla fine. Puoi utilizzare le seguenti risorse per capire quali strumenti AWS Glue per Ray mette a disposizione per trasformare e interpretare i tuoi dati su larga scala.

- Per ulteriori informazioni sul modello di attività di Ray, consulta la pagina the section called “Utilizzo di Ray Core e Ray Data in AWS Glue per Ray”. Per una maggiore esperienza nell'uso delle attività di Ray, segui gli esempi nella documentazione di Ray Core. Consulta la pagina Ray Core: Ray Tutorials and Examples (2.4.0) nella documentazione di Ray.
- Per indicazioni sulle librerie di gestione dei dati disponibili in AWS Glue per Ray, consulta la pagina the section called “Connessione ai dati”. Per ulteriori esperienze con Ray Data per trasformare e scrivere set di dati, segui gli esempi nella documentazione di Ray Data. Consulta la sezione Ray Data: Examples (2.4.0).
- Per ulteriori informazioni sulla configurazione dei processi AWS Glue per Ray, consulta la pagina the section called “Utilizzo dei processi Ray”.
- Per ulteriori informazioni sulla scrittura di script AWS Glue per Ray, continua a leggere la documentazione in questa sezione.

Utilizzo di Ray Core e Ray Data in AWS Glue per Ray

Ray è un framework per dimensionare gli script Python distribuendo il processo su un cluster. Ray fornisce librerie per ottimizzare determinate attività e puoi usarlo come soluzione a molti tipi

di problemi. In AWS Glue, ci concentriamo sull'uso di Ray per trasformare set di dati di grandi dimensioni. AWS Glue offre supporto per Ray Data e parti di Ray Core per facilitare questo compito.

Cos'è Ray Core?

Il primo passaggio per creare un'applicazione distribuita consiste nell'identificare e definire i processi che possono essere eseguiti in simultanea. Ray Core contiene le parti di Ray che si utilizzano per definire le attività che possono essere eseguite contemporaneamente. Ray fornisce informazioni di riferimento e di avvio rapido che è possibile utilizzare per apprendere gli strumenti forniti. Per ulteriori informazioni, consulta le pagine What is Ray Core? e Ray Core Quick Start. Per ulteriori informazioni sulla definizione efficace delle attività simultanee in Ray, consulta la pagina Tips for first-time users.

> **Attività e attori di Ray**
>
> Nella documentazione di AWS Glue per Ray, potremmo fare riferimento ad attività e attori, che sono concetti fondamentali di Ray.
> Ray utilizza le funzioni e le classi Python come elementi costitutivi di un sistema di calcolo distribuito. Analogamente a quanto accade con le funzioni e le variabili di Python, che diventano "metodi" e "attributi" quando vengono utilizzate in una classe, le funzioni diventano "attività" e le classi diventano "attori" quando vengono utilizzate in Ray per inviare codice ai worker. È possibile identificare le funzioni e le classi che potrebbero essere utilizzate da Ray tramite l'annotazione `@ray.remote`.
> Le attività e gli attori sono configurabili, hanno un ciclo di vita e occupano risorse di elaborazione per tutto il ciclo di vita. Il codice che genera errori può essere ricondotto a un'attività o a un attore quando si individua la causa principale dei problemi. Pertanto, potresti imbatterti in questi termini mentre impari a configurare, eseguire il debug o monitorare i processi AWS Glue per Ray.
> Per imparare a utilizzare in modo efficace le attività e gli attori per creare un'applicazione distribuita, consulta la pagina Key Concepts nella documentazione di Ray.

Ray Core in AWS Glue per Ray

Gli ambienti AWS Glue per Ray gestiscono la formazione e la scalabilità dei cluster, nonché la raccolta e la visualizzazione dei log. Poiché gestiamo questi problemi, limitiamo di conseguenza l'accesso e il supporto alle API di Ray Core che verrebbero utilizzate per risolvere questi problemi in un cluster open source.

Nell'ambiente di runtime gestito `Ray2.4`, non supportiamo:

- CLI Ray Core
- CLI Ray State
- Metodi di utilizzo del parametro `ray.util.metrics` di Prometheus:
  - Contatore
  - Gauge
  - Istogramma
- Altri strumenti di debug:
  - ray.util.pdb.set_trace
  - ray.util.inspect_serializability
  - ray.timeline

Cos'è Ray Data?

Quando ti connetti a origini e destinazioni dati, gestisci set di dati e avvii trasformazioni comuni, Ray Data è una metodologia semplice per utilizzare Ray per risolvere i problemi di trasformazione dei set di dati Ray. Per ulteriori informazioni sull'utilizzo di Ray Data, consulta la pagina Ray Datasets: Distributed Data Preprocessing.

Puoi utilizzare Ray Data o altri strumenti per accedere ai tuoi dati. Per ulteriori informazioni sull'accesso ai dati in Ray, consulta la pagina the section called “Connessione ai dati”.

Ray Data in AWS Glue per Ray

Ray Data è supportato e fornito per impostazione predefinita nell'ambiente di runtime gestito `Ray2.4`. Per ulteriori informazioni sui moduli disponibili, consulta the section called “Moduli disponibili con i processi Ray”.

Fornitura di file e librerie Python ai processi Ray

Questa sezione fornisce le informazioni necessarie per utilizzare le librerie Python con i processi di AWS Glue Ray. In tutti i processi Ray è possibile utilizzare alcune librerie comuni incluse per impostazione predefinita. Puoi anche fornire le tue librerie Python al tuo processo Ray.

Moduli disponibili con i processi Ray

È possibile eseguire flussi di lavoro di integrazione dei dati in un processo Ray con i seguenti pacchetti. Per impostazione predefinita, questi pacchetti sono disponibili nei processi Ray.

AWS Glue version 4.0

In AWS Glue 4.0, l'ambiente Ray (runtime `Ray2.4`) fornisce i seguenti pacchetti:

- boto3 == 1.26.133
- ray == 2.4.0
- pyarrow == 11.0.0
- pandas == 1.5.3
- numpy == 1.24.3
- fsspec == 2023.4.0

Questo elenco include tutti i pacchetti che verrebbero installati con `ray[data] == 2.4.0`. Ray Data è supportato immediatamente.

Fornitura di file al processo Ray

Puoi fornire file al tuo processo Ray con il parametro `--working-dir`. Fornisci a questo parametro un percorso per un file .zip ospitato su Amazon S3. All'interno del file .zip, i file devono essere contenuti in un'unica directory di primo livello. Nessun altro file deve trovarsi al livello superiore.

I file verranno distribuiti su ogni nodo Ray prima dell'inizio dell'esecuzione dello script. Considera come ciò potrebbe influire sullo spazio su disco disponibile per ogni nodo Ray. Lo spazio disponibile su disco è determinato dal WorkerType impostato nella configurazione del processo. Se desideri fornire i tuoi dati del processo su larga scala, questo meccanismo non è la soluzione giusta.
Per ulteriori informazioni sulla fornitura di dati al processo, consulta la pagina the section called “Connessione ai dati”.

I tuoi file saranno accessibili come se la directory fosse stata fornita a Ray tramite il parametro `working_dir`. Ad esempio, per leggere un file denominato `sample.txt` nella directory di primo livello del file .zip, puoi chiamare:

```
@ray.remote
def do_work():
    f = open("sample.txt", "r")
    print(f.read())
```

Per ulteriori informazioni su `working_dir`, consulta la documentazione di Ray. Questa funzionalità si comporta in modo simile alle funzionalità native di Ray.

Moduli Python aggiuntivi per i processi Ray

Moduli aggiuntivi di PyPI

I processi Ray utilizzano Python Package Installer (pip3) per installare moduli aggiuntivi da utilizzare con uno script Ray. Puoi utilizzare il parametro `--pip-install` con un elenco di moduli Python separati da virgole per aggiungere un nuovo modulo o modificare la versione di un modulo esistente.

Ad esempio, per aggiornare o aggiungere un nuovo modulo `scikit-learn`, utilizza la seguente coppia chiave-valore:

```
"--pip-install", "scikit-learn==0.21.3"
```

Se disponi di moduli o patch personalizzati, puoi distribuire le tue librerie da Amazon S3 con il parametro `--s3-py-modules`. Prima di caricare la tua distribuzione, potrebbe essere necessario riconfezionarla e ricompilarla. Segui le linee guida riportate nella sezione the section called “Inclusione del codice Python nei processi Ray”.

Distribuzioni personalizzate da Amazon S3

Le distribuzioni personalizzate devono rispettare le linee guida di Ray sulla creazione di pacchetti per le dipendenze. Le istruzioni per creare queste distribuzioni sono riportate nella sezione successiva. Per ulteriori informazioni su come Ray imposta le dipendenze, consulta Dipendenze dell'ambiente nella documentazione di Ray.

Per includere un distribuibile personalizzato dopo averne valutato il contenuto, caricalo in un bucket accessibile dal ruolo IAM del processo. Nella configurazione dei parametri, specifica il percorso Amazon S3 a un archivio zip Python. Se fornisci più distribuibili, separali con una virgola. Ad esempio:

```
"--s3-py-modules", "s3://s3bucket/pythonPackage.zip"
```

Limitazioni

I processi Ray non supportano la compilazione di codice nativo nell'ambiente del processo. Questo può essere un limite se le tue dipendenze da Python dipendono in modo transitorio dal codice nativo compilato. I processi Ray possono eseguire i binari forniti, ma devono essere compilati per Linux su ARM64. Ciò significa che potresti essere in grado di utilizzare il contenuto dei wheel `aarch64manylinux`. Puoi fornire le tue dipendenze native in un formato compilato reimpacchettando una ruota secondo gli standard Ray. In genere, ciò significa rimuovere le cartelle `dist-info` in modo che vi sia una sola cartella alla radice dell'archivio.

Non è possibile aggiornare la versione di `ray` o `ray[data]` utilizzando questo parametro. Per utilizzare una nuova versione di Ray, dovrai modificare il campo di runtime del tuo processo dopo che avremo rilasciato il supporto. Per ulteriori informazioni sulle versioni supportate di Ray, consulta la pagina the section called “Versioni AWS Glue”.

Inclusione del codice Python nei processi Ray

La Python Software Foundation offre comportamenti standardizzati per la creazione di pacchetti di file Python da utilizzare in diversi runtime. Ray introduce delle limitazioni agli standard di confezionamento di cui dovresti essere a conoscenza. AWS Glue non specifica standard di confezionamento oltre a quelli specificati da Ray. Le seguenti istruzioni forniscono una guida standard sulla creazione di pacchetti Python semplici.

Crea un pacchetto dei file in un archivio `.zip`. Alla root dell'archivio dovrebbe essere presente una directory. Non dovrebbero esserci altri file al livello root dell'archivio, altrimenti si verificherà un comportamento imprevisto. La directory root è il pacchetto e il suo nome viene utilizzato per fare riferimento al codice Python durante l'importazione.

Se fornisci una distribuzione in questo formato a un processo Ray con `--s3-py-modules`, sarai in grado di importare il codice Python dal tuo pacchetto nello script Ray.

Il tuo pacchetto può fornire un singolo modulo Python con alcuni file Python, oppure puoi confezionare insieme diversi moduli. Quando riconfezioni le dipendenze, come le librerie da PyPI, controlla i file nascosti e le directory di metadati all'interno di quei pacchetti.

> **Warning**
>
> Alcuni comportamenti del sistema operativo rendono difficile seguire correttamente queste istruzioni di confezionamento.
>
> - OSX può aggiungere file nascosti, ad esempio `__MACOSX`, al file zip di livello superiore.
> - Windows può aggiungere automaticamente i file a una cartella all'interno del file zip, creando involontariamente una cartella annidata.

Le seguenti procedure presuppongono che tu stia interagendo con i tuoi file in Amazon Linux 2 o in un sistema operativo simile che fornisce una distribuzione delle utility `zip` e `zipinfo` di Info-ZIP. Ti consigliamo di utilizzare questi strumenti per prevenire comportamenti imprevisti.

Confezionamento di file Python da utilizzare in Ray

1. Crea una directory temporanea con il nome del tuo pacchetto, quindi verifica che la directory di lavoro sia quella padre. A questo scopo, puoi eseguire il comando seguente:

```
cd parent_directory
mkdir temp_dir
```

2. Copia i file nella directory temporanea, quindi verifica la struttura della directory. Il contenuto di questa directory sarà accessibile direttamente come modulo Python. A questo scopo, puoi eseguire il comando seguente:

```
ls -AR temp_dir
# my_file_1.py
# my_file_2.py
```

3. Comprimi la cartella temporanea utilizzando zip. A questo scopo, puoi eseguire il comando seguente:

```
zip -r zip_file.zip temp_dir
```

4. Verifica che il file sia correttamente confezionato. Ora il *zip_file*`.zip` dovrebbe essere disponibile nella tua directory di lavoro. È possibile verificarla con il seguente comando:

```
zipinfo -1 zip_file.zip
# temp_dir/
# temp_dir/my_file_1.py
# temp_dir/my_file_2.py
```

Riconfezionamento di un pacchetto Python da utilizzare in Ray.

1. Crea una directory temporanea con il nome del tuo pacchetto, quindi verifica che la directory di lavoro sia quella padre. A questo scopo, puoi eseguire il comando seguente:

```
cd parent_directory
mkdir temp_dir
```

2. Decomprimi il pacchetto e copia il contenuto nella directory temporanea. Rimuovi i file relativi allo standard di confezionamento precedente, lasciando solo il contenuto del modulo. Conferma che la struttura del file sia corretta con il seguente comando:

```
ls -AR temp_dir
# my_module
# my_module/__init__.py
# my_module/my_file_1.py
# my_module/my_submodule/__init__.py
# my_module/my_submodule/my_file_2.py
# my_module/my_submodule/my_file_3.py
```

3. Comprimi la cartella temporanea utilizzando zip. A questo scopo, puoi eseguire il comando seguente:

```
zip -r zip_file.zip temp_dir
```

4. Verifica che il file sia correttamente confezionato. Ora il *zip_file*`.zip` dovrebbe essere disponibile nella tua directory di lavoro. È possibile verificarla con il seguente comando:

```
zipinfo -1 zip_file.zip
# temp_dir/my_module/
# temp_dir/my_module/__init__.py
# temp_dir/my_module/my_file_1.py
# temp_dir/my_module/my_submodule/
# temp_dir/my_module/my_submodule/__init__.py
# temp_dir/my_module/my_submodule/my_file_2.py
# temp_dir/my_module/my_submodule/my_file_3.py
```

Connessione ai dati nei processi Ray

I processi AWS Glue Ray possono utilizzare un'ampia gamma di pacchetti Python progettati per integrare rapidamente i dati. Forniamo un set minimo di dipendenze per non appesantire l'ambiente. Per ulteriori informazioni sui componenti inclusi in modo predefinito, consulta la pagina the section called “Moduli disponibili con i processi Ray”.

> **Note**
>
> AWS Glueextract, transform, and load (ETL) fornisce l' DynamicFrame astrazione per semplificare i flussi di lavoro ETL in cui risolvi le differenze di schema tra le righe del set di dati. AWS Glue ETL offre funzionalità aggiuntive, segnalibri di processo e raggruppamento dei file di input. Al momento non forniamo funzionalità corrispondenti nei processi Ray.

AWS Glue per Spark fornisce supporto diretto per la connessione a determinati formati di dati, origini e sink. In Ray, l'SDK AWS per pandas e le attuali librerie di terze parti soddisfano sostanzialmente questa esigenza. Dovrai consultare tali librerie per capire quali funzionalità sono disponibili.

L'integrazione di AWS Glue per Ray con Amazon VPC non è attualmente disponibile. Le risorse in Amazon VPC non saranno accessibili senza un percorso pubblico. Per ulteriori informazioni sull'utilizzo di AWS Glue con i VPC di Amazon, consulta la pagina the section called “Endpoint VPC (AWS PrivateLink)”.

Librerie comuni per lavorare con i dati in Ray

Ray Data: Ray Data fornisce metodi per gestire formati di dati, origini e sink comuni. Per ulteriori informazioni sui formati e le origini supportati in Ray Data, consulta la sezione Input/Output nella documentazione di Ray Data. Ray Data è una libreria prescrittiva anziché generica per la gestione di set di dati.

Ray fornisce alcune indicazioni sui casi d'uso in cui Ray Data potrebbe essere la soluzione migliore per il processo. Per ulteriori informazioni, consulta Casi d'uso di Ray nella documentazione di Ray.

AWSSDK for pandas (awswrangler) — AWS SDK for pandas è un AWS prodotto che offre soluzioni pulite e testate per la lettura e la scrittura da servizi quando le trasformazioni gestiscono i dati con pandas. AWS DataFrames Per ulteriori informazioni sui formati e le origini supportati nell'SDK AWS per pandas, consulta la Documentazione di riferimento all'API nella documentazione dell'SDK AWS per pandas.

Per esempi di come leggere e scrivere dati con l'SDK AWS per pandas, consulta la sezione Quick Start sul sito Web dell'SDK AWS per pandas. L'SDK AWS per pandas non fornisce trasformazioni per i dati. Fornisce supporto solo per la lettura e la scrittura dalle origini.

Modin: Modin è una libreria Python che implementa le comuni operazioni pandas in modo distribuibile. Per ulteriori informazioni su Modin, consulta la documentazione di Modin. Modin non fornisce supporto per la lettura e la scrittura dalle origini. Fornisce implementazioni distribuite di trasformazioni comuni. Modin è supportato dall'SDK AWS per pandas.

Quando esegui Modin e l'SDK AWS per pandas in combinazione in un ambiente Ray, puoi eseguire attività ETL comuni con risultati performanti. Per ulteriori informazioni sull'utilizzo di Modin con l'SDK AWS per pandas, consulta la sezione At scale nella documentazione dell'SDK AWS per pandas.

Altri framework: per ulteriori informazioni sui framework supportati da Ray, consulta The Ray Ecosystem nella documentazione di Ray. Non forniamo supporto per altri framework in AWS Glue per Ray.

Connessione ai dati tramite Catalogo dati

La gestione dei dati tramite Catalogo dati in combinazione con i processi Ray è supportata dall'SDK AWS per pandas. Per ulteriori informazioni, consulta Catalogo Glue sul sito Web dell'SDK AWS per pandas.

Utilizzo di questo servizio con un AWS SDK

AWS I kit di sviluppo software (SDK) sono disponibili per molti linguaggi di programmazione più diffusi. Ogni SDK fornisce un'API, esempi di codice, e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Documentazione sugli SDK	Esempi di codice
AWS SDK for C++	AWS SDK for C++ esempi di codice
AWS SDK for Go	AWS SDK for Go esempi di codice
AWS SDK for Java	AWS SDK for Java esempi di codice
AWS SDK for JavaScript	AWS SDK for JavaScript esempi di codice
SDK AWS for Kotlin	SDK AWS for Kotlin esempi di codice
AWS SDK for .NET	AWS SDK for .NET esempi di codice
AWS SDK for PHP	AWS SDK for PHP esempi di codice
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) esempi di codice
AWS SDK for Ruby	AWS SDK for Ruby esempi di codice
AWS SDK for Rust	AWS SDK for Rust esempi di codice
SDK AWS per SAP ABAP	SDK AWS per SAP ABAP esempi di codice
SDK AWS per Swift	SDK AWS per Swift esempi di codice

Per esempi specifici del servizio, consulta AWS Glue Esempi di codice API con AWS SDK.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Provide feedback (Fornisci un feedback) nella parte inferiore di questa pagina.

API AWS Glue

In questa sezione sono descritti i tipi di dati e le primitive utilizzati da SDK AWS Glue e dagli strumenti. Ci sono tre modi generali per interagire con AWS Glue a livello di programmazione al di fuori della AWS Management Console, ognuno descritto nella relativa documentazione:

- Le librerie di linguaggi SDK consentono di accedere a risorse AWS provenienti da linguaggi di programmazione comuni. Per ulteriori informazioni, consulta Strumenti per costruire in AWS.
- La AWS CLI consente di accedere a risorse AWS dalla riga di comando. Per ulteriori informazioni, consulta Riferimento ai comandi della AWS CLI.
- AWS CloudFormation consente di definire un set di risorse AWS di cui eseguire il provisioning in modo coerente. Per ulteriori informazioni, consulta AWS CloudFormation: riferimento ai tipi di risorse AWS Glue.

In questa sezione sono descritte le primitive condivise indipendentemente da questi SDK e strumenti. Gli strumenti utilizzano Informazioni di riferimento sull'API Web AWS Glue per comunicare con AWS.

Indice

- Struttura SerDeInfo
- Struttura dell'ordine
- Struttura SkewedInfo
- Struttura TableVersion
- Struttura TableError
- Struttura TableVersionError
- Struttura SortCriterion
- Struttura TableIdentifier
- Struttura KeySchemaElement
- Struttura PartitionIndex
- Struttura PartitionIndexDescriptor
- Struttura BackfillError
- Struttura IcebergInput
- Struttura OpenTableFormatInput
- Operazioni
- Operazione CreateTable (Python: create_table)
- Operazione UpdateTable (Python: update_table)
- Operazione DeleteTable (Python: delete_table)
- Operazione BatchDeleteTable (Python: batch_delete_table)
- Operazione GetTable (Python: get_table)
- Operazione GetTables (Python: get_tables)
- Operazione GetTableVersion (Python: get_table_version)
- Operazione GetTableVersions (Python: get_table_versions)
- Operazione DeleteTableVersion (Python: delete_table_version)
- Operazione BatchDeleteTableVersion (Python: batch_delete_table_version)
- Operazione SearchTables (Python: search_tables)
- Operazione GetPartitionIndexes (Python: get_partition_indexes)
- Operazione CreatePartitionIndex (Python: create_partition_index)
- Operazione DeletePartitionIndex (Python: delete_partition_index)
- Operazione GetColumnStatisticsForTable (Python: get_column_statistics_for_table)

- CatalogKinesisSource struttura
- DirectKinesisSource struttura
- KinesisStreamingSourceOptions struttura
- CatalogKafkaSource struttura
- DirectKafkaSource struttura
- KafkaStreamingSourceOptions struttura
- RedshiftSource struttura
- AmazonRedshiftSource struttura
- AmazonRedshiftNodeData struttura
- AmazonRedshiftAdvancedOption struttura
- Struttura Option
- struttura S3 CatalogSource
- Struttura S3 SourceAdditionalOptions
- Struttura S3 CsvSource
- Struttura DirectJDBCSource
- Struttura S3 DirectSourceAdditionalOptions
- Struttura S3 JsonSource
- Struttura S3 ParquetSource
- Struttura S3 DeltaSource
- Struttura S3 CatalogDeltaSource
- CatalogDeltaSource struttura
- Struttura S3 HudiSource
- Struttura S3 CatalogHudiSource
- CatalogHudiSource struttura
- Struttura DynamoDB CatalogSource
- RelationalCatalogSource struttura
- struttura JDBC ConnectorTarget
- SparkConnectorTarget struttura
- BasicCatalogTarget struttura
- Struttura MySQL CatalogTarget

- Struttura PostgreSQL CatalogTarget
- Struttura OracleSQL CatalogTarget
- Struttura Microsoft SQL ServerCatalogTarget
- RedshiftTarget struttura
- AmazonRedshiftTarget struttura
- UpsertRedshiftTargetOptions struttura
- struttura S3 CatalogTarget
- Struttura S3 GlueParquetTarget
- CatalogSchemaChangePolicy struttura
- struttura S3 DirectTarget
- Struttura S3 HudiCatalogTarget
- Struttura S3 HudiDirectTarget
- Struttura S3 DeltaCatalogTarget
- Struttura S3 DeltaDirectTarget
- DirectSchemaChangePolicy struttura
- ApplyMapping struttura
- Struttura mappatura
- SelectFields struttura
- DropFields struttura
- RenameField struttura
- Struttura Spigot
- Struttura join
- JoinColumn struttura
- SplitFields struttura
- SelectFromCollection struttura
- FillMissingValues struttura
- Struttura filtro
- FilterExpression struttura
- FilterValue struttura
- CustomCode struttura

- Struttura SparkSQL
- SqlAlias struttura
- DropNullFields struttura
- NullCheckBoxList struttura
- NullValueField struttura
- Struttura Datatype
- Struttura Merge
- Struttura unione
- Struttura PIIDetection
- Struttura aggregata
- DropDuplicates struttura
- GovernedCatalogTarget struttura
- GovernedCatalogSource struttura
- AggregateOperation struttura
- GlueSchema struttura
- GlueStudioSchemaColumn struttura
- GlueStudioColumn struttura
- DynamicTransform struttura
- TransformConfigParameter struttura
- EvaluateDataQuality struttura
- struttura DQ ResultsPublishingOptions
- Struttura DQ StopJobOnFailureOptions
- EvaluateDataQualityMultiFrame struttura
- Struttura Recipe
- RecipeReference struttura
- SnowflakeNodeData struttura
- SnowflakeSource struttura
- SnowflakeTarget struttura
- ConnectorDataSource struttura
- ConnectorDataTarget struttura

- MetadataKeyValuePair struttura
- SchemaVersionErrorItem struttura
- ErrorDetails struttura
- SchemaVersionNumber struttura
- SchemaId struttura
- Operazioni
- CreateRegistry azione (Python: create_registry)
- CreateSchema azione (Python: create_schema)
- GetSchema azione (Python: get_schema)
- ListSchemaVersions azione (Python: list_schema_versions)
- GetSchemaVersion azione (Python: get_schema_version)
- GetSchemaVersionsDiff azione (Python: get_schema_versions_diff)
- ListRegistries azione (Python: list_registries)
- ListSchemas azione (Python: list_schemas)
- RegisterSchemaVersion azione (Python: register_schema_version)
- UpdateSchema azione (Python: update_schema)
- CheckSchemaVersionValidity azione (Python: check_schema_version_idity)
- UpdateRegistry azione (Python: update_registry)
- GetSchemaByDefinition azione (Python: get_schema_by_definition)
- GetRegistry azione (Python: get_registry)
- PutSchemaVersionMetadata azione (Python: put_schema_version_metadata)
- QuerySchemaVersionMetadata azione (Python: query_schema_version_metadata)
- RemoveSchemaVersionMetadata azione (Python: remove_schema_version_metadata)
- DeleteRegistry azione (Python: delete_registry)
- DeleteSchema azione (Python: delete_schema)
- DeleteSchemaVersions azione (Python: delete_schema_versions)

- Flussi di lavoro
  - Tipi di dati
  - JobNodeDetails struttura
  - CrawlerNodeDetails struttura

- TriggerNodeDetails struttura
- Struttura crawl
- Struttura nodo
- Struttura edge
- Struttura flusso di lavoro
- WorkflowGraph struttura
- WorkflowRun struttura
- WorkflowRunStatistics struttura
- StartingEventBatchCondition struttura
- Struttura schema
- BlueprintDetails struttura
- LastActiveDefinition struttura
- BlueprintRun struttura
- Operazioni
- CreateWorkflow azione (Python: create_workflow)
- UpdateWorkflow azione (Python: update_workflow)
- DeleteWorkflow azione (Python: delete_workflow)
- GetWorkflow azione (Python: get_workflow)
- ListWorkflows azione (Python: list_workflows)
- BatchGetWorkflows azione (Python: batch_get_workflows)
- GetWorkflowRun azione (Python: get_workflow_run)
- GetWorkflowRuns azione (Python: get_workflow_runs)
- GetWorkflowRunProperties azione (Python: get_workflow_run_properties)
- PutWorkflowRunProperties azione (Python: put_workflow_run_properties)
- CreateBlueprint azione (Python: create_blueprint)
- UpdateBlueprint azione (Python: update_blueprint)
- DeleteBlueprint azione (Python: delete_blueprint)
- ListBlueprints azione (Python: list_blueprints)
- BatchGetBlueprints azione (Python: batch_get_blueprints)
- StartBlueprintRun azione (Python: start_blueprint_run)

- Operazione UpdateMLTransform (Python: update_ml_transform)
- Operazione DeleteMLTransform (Python: delete_ml_transform)
- Operazione GetMLTransform (Python: get_ml_transform)
- Operazione GetMLTransforms (Python: get_ml_transforms)
- Operazione ListMLTransforms (Python: list_ml_transforms)
- Operazione StartMLEvaluationTaskRun (Python: start_ml_evaluation_task_run)
- Operazione StartMLLabelingSetGenerationTaskRun (Python: start_ml_labeling_set_generation_task_run)
- Operazione GetMLTaskRun (Python: get_ml_task_run)
- Operazione GetMLTaskRuns (Python: get_ml_task_runs)
- Operazione CancelMLTaskRun (Python: cancel_ml_task_run)
- Operazione StartExportLabelsTaskRun (Python: start_export_labels_task_run)
- Operazione StartImportLabelsTaskRun (Python: start_import_labels_task_run)

- API di qualità dei dati
  - Tipi di dati
  - DataSource struttura
  - DataQualityRulesetListDetails struttura
  - DataQualityTargetTable struttura
  - DataQualityRulesetEvaluationRunDescription struttura
  - DataQualityRulesetEvaluationRunFilter struttura
  - DataQualityEvaluationRunAdditionalRunOptions struttura
  - DataQualityRuleRecommendationRunDescription struttura
  - DataQualityRuleRecommendationRunFilter struttura
  - DataQualityResult struttura
  - DataQualityAnalyzerResult struttura
  - DataQualityObservation struttura
  - MetricBasedObservation struttura
  - DataQualityMetricValues struttura
  - DataQualityRuleResult struttura
  - DataQualityResultDescription struttura

- DataQualityResultFilterCriteria struttura
- DataQualityRulesetFilterCriteria struttura
- Operazioni
- StartDataQualityRulesetEvaluationRun azione (Python: start_data_quality_ruleset_evaluation_run)
- CancelDataQualityRulesetEvaluationRun azione (Python: cancel_data_quality_ruleset_evaluation_run)
- GetDataQualityRulesetEvaluationRun azione (Python: get_data_quality_ruleset_evaluation_run)
- ListDataQualityRulesetEvaluationRuns azione (Python: list_data_quality_ruleset_evaluation_runs)
- StartDataQualityRuleRecommendationRun azione (Python: start_data_quality_rule_recommendation_run)
- CancelDataQualityRuleRecommendationRun azione (Python: cancel_data_quality_rule_recommendation_run)
- GetDataQualityRuleRecommendationRun azione (Python: get_data_quality_rule_recommendation_run)
- ListDataQualityRuleRecommendationRuns azione (Python: list_data_quality_rule_recommendation_runs)
- GetDataQualityResult azione (Python: get_data_quality_result)
- BatchGetDataQualityResult azione (Python: batch_get_data_quality_result)
- ListDataQualityResults azione (Python: list_data_quality_results)
- CreateDataQualityRuleset azione (Python: create_data_quality_ruleset)
- DeleteDataQualityRuleset azione (Python: delete_data_quality_ruleset)
- GetDataQualityRuleset azione (Python: get_data_quality_ruleset)
- ListDataQualityRulesets azione (Python: list_data_quality_rulesets)
- UpdateDataQualityRuleset azione (Python: update_data_quality_ruleset)

- API di rilevamento dati sensibili
  - Tipi di dati
  - Struttura CustomEntityType
  - Operazioni
  - Operazione CreateCustomEntityType (Python: create_custom_entity_type)

- Struttura AccessDeniedException
- Struttura AlreadyExistsException
- Struttura ConcurrentModificationException
- Struttura ConcurrentRunsExceededException
- Struttura CrawlerNotRunningException
- Struttura CrawlerRunningException
- Struttura CrawlerStoppingException
- Struttura EntityNotFoundException
- Struttura FederationSourceException
- Struttura FederationSourceRetryableException
- Struttura GlueEncryptionException
- Struttura IdempotentParameterMismatchException
- Struttura IllegalWorkflowStateException
- Struttura InternalServiceException
- Struttura InvalidExecutionEngineException
- Struttura InvalidInputException
- Struttura InvalidStateException
- Struttura InvalidTaskStatusTransitionException
- Struttura JobDefinitionErrorException
- Struttura JobRunInTerminalStateException
- Struttura JobRunInvalidStateTransitionException
- Struttura JobRunNotInTerminalStateException
- Struttura LateRunnerException
- Struttura NoScheduleException
- Struttura OperationTimeoutException
- Struttura ResourceNotReadyException
- Struttura ResourceNumberLimitExceededException
- Struttura SchedulerNotRunningException
- Struttura SchedulerRunningException
- Struttura SchedulerTransitioningException

- Struttura UnrecognizedRunnerException
- Struttura ValidationException
- Struttura VersionMismatchException

API di sicurezza in AWS Glue

L'API di sicurezza descrive i tipi di dati di sicurezza e l'API relativa alla sicurezza in AWS Glue.

Tipi di dati

- DataCatalogEncryptionSettings struttura
- EncryptionAtRest struttura
- ConnectionPasswordEncryption struttura
- EncryptionConfiguration struttura
- Struttura S3Encryption
- CloudWatchEncryption struttura
- JobBookmarksEncryption struttura
- SecurityConfiguration struttura
- GluePolicy struttura

DataCatalogEncryptionSettings struttura

Contiene le informazioni di configurazione per mantenere la sicurezza del catalogo dati.

Campi

- `EncryptionAtRest`: un oggetto EncryptionAtRest.

  Speciifica la encryption-at-rest configurazione per il Data Catalog.
- `ConnectionPasswordEncryption`: un oggetto ConnectionPasswordEncryption.

  Quando è abilitata la protezione della password di connessione, il catalogo dati utilizza una chiave fornita dal cliente per crittografare la password come parte di `CreateConnection` o `UpdateConnection` e memorizzarla nel campo `ENCRYPTED_PASSWORD` nelle proprietà di connessione. È possibile abilitare la crittografia del catalogo o solo la crittografia delle password.

EncryptionAtRest struttura

Speciifica la encryption-at-rest configurazione per il Data Catalog.

Campi

- `CatalogEncryptionMode`: obbligatorio: stringa UTF-8 (valori validi: `DISABLED` | `SSE-KMS="SSEKMS"` | `SSE-KMS-WITH-SERVICE-ROLE="SSEKMSWITHSERVICEROLE"`).

  La encryption-at-rest modalità per crittografare i dati del Data Catalog.

- `SseAwsKmsKeyId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID della AWS KMS chiave da utilizzare per la crittografia a riposo.

- `CatalogEncryptionServiceRole`: stringa UTF-8, corrispondente a Custom string pattern #19.

  Il ruolo che AWS Glue assume di crittografare e decrittografare gli oggetti del Data Catalog per conto del chiamante.

ConnectionPasswordEncryption struttura

La struttura di dati utilizzata dal catalogo dati per crittografare la password come parte di `CreateConnection` o `UpdateConnection` e memorizzarla nel campo `ENCRYPTED_PASSWORD` nelle proprietà di connessione. È possibile abilitare la crittografia del catalogo o solo la crittografia delle password.

Quando arriva una `CreationConnection` richiesta contenente una password, il Data Catalog crittografa innanzitutto la password utilizzando la AWS KMS chiave dell'utente. Successivamente crittografa l'intero oggetto di connessione di nuovo se anche la crittografia del catalogo è abilitata.

Questa crittografia richiede l'impostazione delle autorizzazioni AWS KMS chiave per abilitare o limitare l'accesso alla chiave della password in base ai requisiti di sicurezza. Ad esempio, è possibile che si voglia concedere solo agli amministratori l'autorizzazione di decrittografare la chiave della password.

Campi

- `ReturnConnectionPasswordEncrypted`: obbligatorio: booleano.

Quando il flag `ReturnConnectionPasswordEncrypted` è impostato su "true", le password rimangono crittografate nelle risposte di `GetConnection` e `GetConnections`. Questa crittografia è effettiva indipendentemente dalla crittografia del catalogo.

- `AwsKmsKeyId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Una AWS KMS chiave utilizzata per crittografare la password di connessione.

  Se la protezione tramite password di connessione è abilitata, il chiamante `CreateConnection` e `UpdateConnection` necessita almeno dell'`kms:Encrypt`autorizzazione sulla AWS KMS chiave specificata per crittografare le password prima di archiviarle nel Data Catalog.

  È possibile impostare l'autorizzazione di decrittografia per consentire o limitare l'accesso alla chiave della password in base ai requisiti di sicurezza.

EncryptionConfiguration struttura

Specifica una configurazione di crittografia.

Campi

- `S3Encryption`: una matrice di oggetti S3Encryption.

  La configurazione di crittografia per i dati Amazon Simple Storage Service (Amazon S3).

- `CloudWatchEncryption`: un oggetto CloudWatchEncryption.

  La configurazione di crittografia per Amazon CloudWatch.

- `JobBookmarksEncryption`: un oggetto JobBookmarksEncryption.

  Configurazione di crittografia per i segnalibri dei processi.

Struttura S3Encryption

Specifica come devono essere crittografati i dati Amazon Simple Storage Service (Amazon S3).

Campi

- `S3EncryptionMode`: stringa UTF-8 (valori validi: `DISABLED` | `SSE-KMS="SSEKMS"` | `SSE-S3="SSES3"`).

  Modalità di crittografia da usare per i dati Amazon S3.
- `KmsKeyArn`: stringa UTF-8, corrispondente a Custom string pattern #20.

  ARN (Amazon Resource Name) della chiave KMS da usare per crittografare i dati.

CloudWatchEncryption struttura

Speciifica in che modo CloudWatch i dati di Amazon devono essere crittografati.

Campi

- `CloudWatchEncryptionMode`: stringa UTF-8 (valori validi: `DISABLED` | `SSE-KMS="SSEKMS"`).

  La modalità di crittografia da utilizzare per CloudWatch i dati.
- `KmsKeyArn`: stringa UTF-8, corrispondente a Custom string pattern #20.

  ARN (Amazon Resource Name) della chiave KMS da usare per crittografare i dati.

JobBookmarksEncryption struttura

Specifica come devono essere crittografati i dati dei segnalibri dei processi.

Campi

- `JobBookmarksEncryptionMode`: stringa UTF-8 (valori validi: `DISABLED` | `CSE-KMS="CSEKMS"`).

  Modalità di crittografia da usare per i dati dei segnalibri dei processi.
- `KmsKeyArn`: stringa UTF-8, corrispondente a Custom string pattern #20.

  ARN (Amazon Resource Name) della chiave KMS da usare per crittografare i dati.

SecurityConfiguration struttura

Specifica una configurazione di sicurezza.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della configurazione di sicurezza.

- `CreatedTimeStamp`: timestamp.

  Data e ora in cui questa configurazione di sicurezza è stata creata.

- `EncryptionConfiguration`: un oggetto EncryptionConfiguration.

  Configurazione di crittografia associata a questa configurazione di sicurezza.

GluePolicy struttura

Una struttura per la restituzione di una policy delle risorse.

Campi

- `PolicyInJson`: stringa UTF-8, almeno 2 byte di lunghezza.

  Contiene il documento di policy richiesto, in formato JSON.

- `PolicyHash`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Contiene il valore hash associato a questa policy.

- `CreateTime`: timestamp.

  La data e l'ora di creazione della policy.

- `UpdateTime`: timestamp.

  La data e l'ora dell'ultimo aggiornamento della policy.

Operazioni

- GetDataCatalogEncryptionSettings azione (Python: get_data_catalog_encryption_settings)
- PutDataCatalogEncryptionSettings azione (Python: put_data_catalog_encryption_settings)
- PutResourcePolicy azione (Python: put_resource_policy)
- GetResourcePolicy azione (Python: get_resource_policy)
- DeleteResourcePolicy azione (Python: delete_resource_policy)
- CreateSecurityConfiguration azione (Python: create_security_configuration)
- DeleteSecurityConfiguration azione (Python: delete_security_configuration)
- GetSecurityConfiguration azione (Python: get_security_configuration)
- GetSecurityConfigurations azione (Python: get_security_configurations)
- GetResourcePolicies azione (Python: get_resource_policies)

GetDataCatalogEncryptionSettings azione (Python: get_data_catalog_encryption_settings)

Recupera la configurazione di sicurezza per un catalogo specificato.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  ID del catalogo dati per cui recuperare la configurazione di sicurezza. Se non ne viene fornito nessuno, per impostazione predefinita viene utilizzato l'ID dell'account. AWS

Risposta

- `DataCatalogEncryptionSettings`: un oggetto DataCatalogEncryptionSettings.

  Configurazione di sicurezza richiesta.

Errori

- `InternalServiceException`

- `InvalidInputException`
- `OperationTimeoutException`

PutDataCatalogEncryptionSettings azione (Python: put_data_catalog_encryption_settings)

Imposta la configurazione di sicurezza per un catalogo specificato. Dopo aver impostato la configurazione, la crittografia specificata viene applicata a ogni scrittura successiva nel catalogo.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  ID del catalogo dati per cui impostare la configurazione di sicurezza. Se non ne viene fornito nessuno, per impostazione predefinita viene utilizzato l'ID dell'account. AWS
- `DataCatalogEncryptionSettings`: obbligatorio: un oggetto DataCatalogEncryptionSettings.

  Configurazione di sicurezza da impostare.

Risposta

- Nessun parametro di risposta.

Errori

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

PutResourcePolicy azione (Python: put_resource_policy)

Imposta la policy per la risorsa del catalogo dati per il controllo accessi.

Richiesta

- `PolicyInJson`: obbligatorio: stringa UTF-8, almeno 2 byte di lunghezza.

Contiene il documento di policy da impostare, in formato JSON.

- `ResourceArn`: stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  Non usare. Solo per uso interno.

- `PolicyHashCondition`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il valore hash restituito quando la policy precedente è stata impostata utilizzando `PutResourcePolicy`. Il suo scopo è prevenire modifiche simultanee di una policy. Non utilizzare questo parametro se non è stata impostata alcuna policy precedente.

- `PolicyExistsCondition`: stringa UTF-8 (valori validi: `MUST_EXIST` | `NOT_EXIST` | `NONE`).

  Il valore `MUST_EXIST` viene utilizzato per aggiornare una policy. Il valore `NOT_EXIST` viene utilizzato per creare una nuova policy. Se viene utilizzato il valore `NONE` o un valore null, la chiamata non dipende dalla presenza di una policy.

- `EnableHybrid`: stringa UTF-8 (valori validi: `TRUE` | `FALSE`).

  Se `'TRUE'`, vuol dire che stai usando entrambi i metodi per concedere l'accesso tra account alle risorse del catalogo dati:

  - Aggiornando direttamente la policy delle risorse con `PutResourePolicy`
  - Utilizzando il comando Concessione di autorizzazioni sulla AWS Management Console.

  Se è già stata utilizzata la Console di gestione per concedere l'accesso tra account, deve essere impostato su `'TRUE'`, altrimenti la chiamata non riesce. Il valore di default è 'FALSE'.

Risposta

- `PolicyHash`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un hash della policy appena impostata. Questo deve essere incluso in una chiamata successiva che sovrascrive o aggiorna questa policy.

Errori

- `EntityNotFoundException`

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ConditionCheckFailureException`

GetResourcePolicy azione (Python: get_resource_policy)

Recupera una policy di risorse specificata.

Richiesta

- `ResourceArn`: stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN della AWS Glue risorsa per cui recuperare la politica delle risorse. Se non viene fornito, viene restituita la policy delle risorse del catalogo dati. Utilizza `GetResourcePolicies` per visualizzare tutte le policy delle risorse esistenti. Per ulteriori informazioni, consulta Specificazione degli ARN delle risorse AWS Glue.

Risposta

- `PolicyInJson`: stringa UTF-8, almeno 2 byte di lunghezza.

  Contiene il documento di policy richiesto, in formato JSON.
- `PolicyHash`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Contiene il valore hash associato a questa policy.
- `CreateTime`: timestamp.

  La data e l'ora di creazione della policy.
- `UpdateTime`: timestamp.

  La data e l'ora dell'ultimo aggiornamento della policy.

Errori

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

DeleteResourcePolicy azione (Python: delete_resource_policy)

Elimina una policy specificata.

Richiesta

- `PolicyHashCondition`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il valore hash restituito quando è stata impostata questa policy.
- `ResourceArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN della AWS Glue risorsa per la politica delle risorse da eliminare.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ConditionCheckFailureException`

CreateSecurityConfiguration azione (Python: create_security_configuration)

Crea una nuova configurazione di sicurezza. Una configurazione della sicurezza è un set di proprietà di sicurezza che AWS Glue può usare. Puoi usare una configurazione della sicurezza per crittografare i dati inattivi. Per informazioni sull'utilizzo delle configurazioni di sicurezza in AWS Glue, consulta Encrypting Data Written by Crawlers, Jobs and Development Endpoints.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome per la nuova configurazione di sicurezza.
- `EncryptionConfiguration`: obbligatorio: oggetto EncryptionConfiguration.

  Configurazione di crittografia per la nuova configurazione di sicurezza.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome assegnato alla nuova configurazione di sicurezza.
- `CreatedTimestamp`: timestamp.

  Data e ora in cui la nuova configurazione di sicurezza è stata creata.

Errori

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

DeleteSecurityConfiguration azione (Python: delete_security_configuration)

Elimina una configurazione di sicurezza specificata.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della configurazione di sicurezza da eliminare.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetSecurityConfiguration azione (Python: get_security_configuration)

Recupera una configurazione di sicurezza specificata.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della configurazione di sicurezza da recuperare.

Risposta

- `SecurityConfiguration`: un oggetto SecurityConfiguration.

  Configurazione di sicurezza richiesta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetSecurityConfigurations azione (Python: get_security_configurations)

Recupera un elenco di tutte le configurazioni di sicurezza.

Richiesta

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Numero massimo di risultati da restituire.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `SecurityConfigurations`: una matrice di oggetti SecurityConfiguration.

  Elenco di configurazioni di sicurezza.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se devono essere restituite più configurazioni di sicurezza.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetResourcePolicies azione (Python: get_resource_policies)

Recupera le politiche relative alle risorse impostate sulle singole risorse durante le concessioni di autorizzazioni tra account. AWS Resource Access Manager Recupera anche la policy per la risorsa del catalogo dati.

Se hai abilitato la crittografia dei metadati nelle impostazioni di Data Catalog e non disponi dell'autorizzazione sulla AWS KMS chiave, l'operazione non può restituire la politica delle risorse del Data Catalog.

Richiesta

- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima di un elenco da restituire.

Risposta

- `GetResourcePoliciesResponseList`: una matrice di oggetti GluePolicy.

  Elenco delle singole policy delle risorse e delle policy delle risorse a livello di account.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se l'elenco restituito non contiene l'ultima policy delle risorse disponibile.

Errori

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

Catalogo API

L'API Catalog descrive i tipi di dati e l'API relativa all'utilizzo dei cataloghi in AWS Glue.

Argomenti

API database

L'API Database descrive i tipi di dati del database e include l'API per creare, eliminare, localizzare, aggiornare ed elencare i database.

Tipi di dati

Struttura dei database

L'oggetto `Database` rappresenta un raggruppamento logico di tabelle che potrebbero trovarsi in un metastore Hive o in un RDBMS.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del database. Per la compatibilità Hive, questo viene scritto in minuscolo durante la memorizzazione.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

Descrizione del database.

- `LocationUri`: uniform resource identifier (uri), non inferiore a 1 e non superiore a 1024 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La posizione del database (per esempio, un percorso HDFS).

- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono parametri e proprietà del database.

- `CreateTime`: timestamp.

  L'ora in cui è stato creato il database di metadati nel catalogo.

- `CreateTableDefaultPermissions`: una matrice di oggetti PrincipalPermissions.

  Crea un set di autorizzazioni predefinite per la tabella dei principal. Utilizzato da AWS Lake Formation. Non utilizzato nel corso delle normali operazioni di AWS Glue.

- `TargetDatabase`: un oggetto DatabaseIdentifier.

  Una struttura `DatabaseIdentifier` che descrive un database di destinazione per il collegamento delle risorse.

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede il database.

- `FederatedDatabase`: un oggetto FederatedDatabase.

  Una struttura `FederatedDatabase` che fa riferimento a un'entità esterna al AWS Glue Data Catalog.

Struttura DatabaseInput

Struttura utilizzata per la creazione o per l'aggiornamento di un database.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del database. Per la compatibilità Hive, questo viene scritto in minuscolo durante la memorizzazione.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Descrizione del database.

- `LocationUri`: uniform resource identifier (uri), non inferiore a 1 e non superiore a 1024 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La posizione del database (per esempio, un percorso HDFS).

- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono parametri e proprietà del database.

  Queste coppie chiave-valore definiscono parametri e proprietà del database.

- `CreateTableDefaultPermissions`: una matrice di oggetti PrincipalPermissions.

  Crea un set di autorizzazioni predefinite per la tabella dei principal. Utilizzato da AWS Lake Formation. Non utilizzato nel corso delle normali operazioni di AWS Glue.

- `TargetDatabase`: un oggetto DatabaseIdentifier.

  Una struttura `DatabaseIdentifier` che descrive un database di destinazione per il collegamento delle risorse.

- `FederatedDatabase`: un oggetto FederatedDatabase.

  Una struttura `FederatedDatabase` che fa riferimento a un'entità esterna al AWS Glue Data Catalog.

Struttura PrincipalPermissions

Autorizzazioni concesse a un principal.

Campi

- `Principal`: un oggetto DataLakePrincipal.

  Il principal a cui vengono concesse le autorizzazioni.
- `Permissions`: una matrice di stringhe UTF-8.

  Le autorizzazioni concesse al principal.

Struttura DataLakePrincipal

Il principale AWS Lake Formation.

Campi

- `DataLakePrincipalIdentifier`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza.

  Un identificatore del principal AWS Lake Formation.

Struttura DatabaseIdentifier

Una struttura che descrive un database di destinazione per il collegamento delle risorse.

Campi

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede il database.
- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo.
- `Region`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

La regione della tabella di destinazione.

Struttura FederatedDatabase

Un database che punta a un'entità esterna al AWS Glue Data Catalog.

Campi

- `Identifier`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Single-line string pattern.

  Un identificatore univoco per la tabella federata.
- `ConnectionName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della connessione al metastore esterno.

Operazioni

- Operazione CreateDatabase (Python: create_database)
- Operazione UpdateDatabase (Python: update_database)
- Operazione DeleteDatabase (Python: delete_database)
- Operazione GetDatabase (Python: get_database)
- Operazione GetDatabase (Python: get_database)

Operazione CreateDatabase (Python: create_database)

Crea un nuovo database in un catalogo di dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui creare il database. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseInput`: obbligatorio: un oggetto DatabaseInput.

  I metadati per il database.

- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag assegnati al database.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`
- `FederatedResourceAlreadyExistsException`

Operazione UpdateDatabase (Python: update_database)

Aggiorna una definizione di database esistente in un catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede il database dei metadati. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database da caricare nel catalogo. Per la compatibilità Hive, questo è scritto in caratteri minuscoli.

- `DatabaseInput`: obbligatorio: un oggetto DatabaseInput.

  Un oggetto `DatabaseInput` che specifica la nuova definizione del database di metadati nel catalogo.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`

Operazione DeleteDatabase (Python: delete_database)

Rimuove un database specificato da un catalogo dati.

> **Note**
>
> Una volta completata questa operazione, non potrai più accedere alle tabelle (e a tutte le versioni e le partizioni delle tabelle che potrebbero appartenere alle tabelle stesse) e alle funzioni definite dall'utente del database eliminato. AWS Glue elimina tempestivamente queste risorse "orfane" in modo asincrono, a discrezione del servizio.
> Per garantire l'eliminazione immediata di tutte le risorse correlate, prima di chiamare `DeleteDatabase`, utilizza `DeleteTableVersion` o `BatchDeleteTableVersion`, `DeletePartition` o `BatchDeletePartition`, `DeleteUserDefinedFunction` e

`DeleteTable` o `BatchDeleteTable` per eliminare eventuali risorse che appartengono al database.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede il database. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database da eliminare. Per la compatibilità Hive, deve essere interamente in caratteri minuscoli.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Operazione GetDatabase (Python: get_database)

Recupera la definizione di un database specificato.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID del catalogo dati in cui risiede il database. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del database da ripristinare. Per la compatibilità Hive, deve essere interamente in caratteri minuscoli.

Risposta

- `Database`: un oggetto Database.

  La definizione del database specificato nel catalogo dati.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `FederationSourceException`

Operazione GetDatabase (Python: get_database)

Recupera tutti i database definiti in un determinato catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati da cui recuperare `Databases`. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `NextToken`: stringa UTF-8.

Un token di continuazione, se si tratta di una chiamata di continuazione.

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 100.

  Il numero massimo di database da restituire in una risposta.

- `ResourceShareType`: stringa UTF-8 (valori validi: `FOREIGN` | `ALL` | `FEDERATED`).

  Consente di specificare che si desidera elencare i database condivisi con l'account. I valori consentiti sono `FEDERATED`, `FOREIGN` o `ALL`.

  - Se impostato su `FEDERATED`, elencherà i database federati (con riferimento a un'entità esterna) condivisi con l'account.
  - Se impostato su `FOREIGN`, elencherà i database condivisi con l'account.
  - Se impostato su `ALL`, elencherà i database condivisi con l'account, così come i database nell'account locale.

Risposta

- `DatabaseList`: obbligatorio: una matrice di oggetti Database.

  Un elenco di oggetti `Database` dal catalogo specificato.

- `NextToken`: stringa UTF-8.

  Un token di continuazione per impaginare l'elenco restituito di token, restituiti se il segmento corrente dell'elenco non è l'ultimo.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Tabella API

L'API Table descrive i tipi di dati e le operazioni associate alle tabelle.

Tipi di dati

Struttura della tabella

Rappresenta una raccolta di dati correlati organizzati in colonne e righe.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella. Per la compatibilità Hive, deve essere interamente in caratteri minuscoli.

- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del database dei metadati in cui risiedono i metadati della tabella. Per la compatibilità Hive, deve essere interamente in caratteri minuscoli.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Descrizione della tabella.

- `Owner`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il proprietario della tabella.

- `CreateTime`: timestamp.

  Ora della creazione della definizione della tabella nel catalogo dati.

- `UpdateTime`: timestamp.

  L'ultima volta che la tabella è stata aggiornata.

- `LastAccessTime`: timestamp.

  L'ultima volta che la tabella è stata consultata. Questo dato in genere viene fornito dall'HDFS e potrebbe non essere affidabile.

- `LastAnalyzedTime`: timestamp.

  L'ultima volta in cui sono state calcolate le statistiche di colonna per questa tabella.

- `Retention`: numero (intero), non superiore a Nessuno.

  Tempo di conservazione per questa tabella.

- `StorageDescriptor`: un oggetto StorageDescriptor.

  Un descrittore di archiviazione contenente informazioni sull'archiviazione fisica di questa tabella.

- `PartitionKeys`: una matrice di oggetti Colonna.

  Un elenco di colonne in base al quale la tabella è partizionata. Solo i tipi primitivi sono supportati come chiavi di partizione.

Quando crei una tabella utilizzata da Amazon Athena e non specifichi una `partitionKeys`, è necessario almeno impostare il valore di `partitionKeys` su un elenco vuoto. Ad esempio:

`"PartitionKeys": []`

- `ViewOriginalText`: stringa UTF-8, non superiore a 409600 byte di lunghezza.

  Incluso per la compatibilità con Apache Hive. Non utilizzato nel corso delle normali operazioni di AWS Glue. Se la tabella è una `VIRTUAL_VIEW`, alcune configurazioni di Athena sono codificate in formato base64.
- `ViewExpandedText`: stringa UTF-8, non superiore a 409600 byte di lunghezza.

  Incluso per la compatibilità con Apache Hive. Non utilizzato nel corso delle normali operazioni di AWS Glue.
- `TableType`: stringa UTF-8, non superiore a 255 byte di lunghezza.

  Il tipo di questa tabella. AWS Glue creerà tabelle con il tipo `EXTERNAL_TABLE`. Altri servizi, come Athena, possono creare tabelle con tipi di tabella aggiuntivi.

  Tipi di tabella correlati ad AWS Glue:

  EXTERNAL_TABLE

  Attributo compatibile con Hive. Indica una tabella non gestita da Hive.

  GOVERNED

  Utilizzato da AWS Lake Formation. Il Catalogo dati AWS Glue comprende `GOVERNED`.
- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono proprietà associate a questa tabella.
- `CreatedBy`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Persona o entità che ha creato la tabella.
- `IsRegisteredWithLakeFormation`: booleano.

Indica se la tabella è stata registrata su AWS Lake Formation.

- `TargetTable`: un oggetto TableIdentifier.

  Una struttura `TableIdentifier` che descrive una tabella di destinazione per il collegamento delle risorse.

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella.

- `VersionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID della versione della tabella.

- `FederatedTable`: un oggetto FederatedTable.

  Una struttura `FederatedTable` che fa riferimento a un'entità esterna al AWS Glue Data Catalog.

Struttura TableInput

Una struttura utilizzata per definire una tabella.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella. Per la compatibilità Hive, questo viene scritto in minuscolo durante la memorizzazione.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Descrizione della tabella.

- `Owner`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il proprietario della tabella. Incluso per la compatibilità con Apache Hive. Non utilizzato nel corso delle normali operazioni di AWS Glue.

- `LastAccessTime`: timestamp.

  L'ultima volta che la tabella è stata consultata.

- `LastAnalyzedTime`: timestamp.

  L'ultima volta in cui sono state calcolate le statistiche di colonna per questa tabella.

- `Retention`: numero (intero), non superiore a Nessuno.

  Tempo di conservazione per questa tabella.

- `StorageDescriptor`: un oggetto StorageDescriptor.

  Un descrittore di archiviazione contenente informazioni sull'archiviazione fisica di questa tabella.

- `PartitionKeys`: una matrice di oggetti Colonna.

  Un elenco di colonne in base al quale la tabella è partizionata. Solo i tipi primitivi sono supportati come chiavi di partizione.

  Quando crei una tabella utilizzata da Amazon Athena e non specifichi una `partitionKeys`, è necessario almeno impostare il valore di `partitionKeys` su un elenco vuoto. Ad esempio:

  `"PartitionKeys": []`

- `ViewOriginalText`: stringa UTF-8, non superiore a 409600 byte di lunghezza.

  Incluso per la compatibilità con Apache Hive. Non utilizzato nel corso delle normali operazioni di AWS Glue. Se la tabella è una `VIRTUAL_VIEW`, alcune configurazioni di Athena sono codificate in formato base64.

- `ViewExpandedText`: stringa UTF-8, non superiore a 409600 byte di lunghezza.

  Incluso per la compatibilità con Apache Hive. Non utilizzato nel corso delle normali operazioni di AWS Glue.

- `TableType`: stringa UTF-8, non superiore a 255 byte di lunghezza.

  Il tipo di questa tabella. AWS Glue creerà tabelle con il tipo `EXTERNAL_TABLE`. Altri servizi, come Athena, possono creare tabelle con tipi di tabella aggiuntivi.

  Tipi di tabella correlati ad AWS Glue:

  EXTERNAL_TABLE

  Attributo compatibile con Hive. Indica una tabella non gestita da Hive.

GOVERNED

Utilizzato da AWS Lake Formation. Il Catalogo dati AWS Glue comprende `GOVERNED`.

- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono proprietà associate a questa tabella.

- `TargetTable`: un oggetto TableIdentifier.

  Una struttura `TableIdentifier` che descrive una tabella di destinazione per il collegamento delle risorse.

Struttura FederatedTable

Una tabella che punta a un'entità esterna al AWS Glue Data Catalog.

Campi

- `Identifier`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Single-line string pattern.

  Un identificatore univoco per la tabella federata.

- `DatabaseIdentifier`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Single-line string pattern.

  Un identificatore univoco per la tabella federata.

- `ConnectionName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della connessione al metastore esterno.

Struttura delle colonne

Una colonna in una `Table`.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della `Column`.
- `Type`: stringa UTF-8, non superiore a 131072 byte di lunghezza, corrispondente a Single-line string pattern.

  Il tipo di dati di `Column`.
- `Comment`: stringa di commento, non superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Commento con testo in formato libero.
- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono proprietà associate alla colonna.

Struttura StorageDescriptor

Descrive lo storage fisico dei dati della tabella.

Campi

- `Columns`: una matrice di oggetti Colonna.

  Un elenco delle `Columns` nella tabella.
- `Location`: stringa di posizione, non superiore a 2056 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La posizione fisica della tabella. Per impostazione predefinita questo utilizza il modulo della posizione del warehouse, seguito dalla posizione del database nel warehouse, seguito dal nome della tabella.
- `AdditionalLocations`: una matrice di stringhe UTF-8.

Un elenco di posizioni che puntano al percorso in cui si trova una tabella Delta.

- `InputFormat`: stringa di formato, non superiore a 128 byte di lunghezza, corrispondente a Single-line string pattern.

  Il formato di input: `SequenceFileInputFormat` (binario) o `TextInputFormat` o un formato personalizzato.
- `OutputFormat`: stringa di formato, non superiore a 128 byte di lunghezza, corrispondente a Single-line string pattern.

  Il formato di output: `SequenceFileOutputFormat` (binario) o `IgnoreKeyTextOutputFormat` o un formato personalizzato.
- `Compressed`: booleano.

  `True` se i dati nella tabella sono compressi, in caso contrario `False`.
- `NumberOfBuckets`: numero (intero).

  Deve essere specificato se la tabella contiene una qualsiasi colonna di dimensione.
- `SerdeInfo`: un oggetto SerDeInfo.

  Informazioni di serializzazione/deserializzazione (SerDe).
- `BucketColumns`: una matrice di stringhe UTF-8.

  Un elenco di colonne per il raggruppamento del reducer, colonne di clustering, colonne di bucketing nella tabella.
- `SortColumns`: una matrice di oggetti Order.

  Un elenco specificando l'ordine di ciascuna bucket nella tabella.
- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Le proprietà fornite dall'utente nel modulo chiave-valore.
- `SkewedInfo`: un oggetto SkewedInfo.

  Informazioni sui valori che appaiono di frequente in una colonna (valori disallineati).

- `StoredAsSubDirectories`: booleano.

  `True` se i dati nella tabella sono archiviati nelle sottodirectory, in caso contrario `False`.

- `SchemaReference`: un oggetto SchemaReference.

  Un oggetto che fa riferimento a uno schema memorizzato nel Registro degli schemi di AWS Glue.

  Quando crei una tabella, puoi passare un elenco vuoto di colonne per lo schema e utilizzare invece un riferimento allo schema.

Struttura SchemaReference

Un oggetto che fa riferimento a uno schema memorizzato nel Registro degli schemi di AWS Glue.

Campi

- `SchemaId`: un oggetto SchemaId.

  Una struttura che contiene campi di identità dello schema. Deve essere fornito questo o `SchemaVersionId`.

- `SchemaVersionId`: stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID univoco assegnato a una versione dello schema. Deve essere fornito questo o `SchemaId`.

- `SchemaVersionNumber`: numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione dello schema.

Struttura SerDeInfo

Informazioni su un programma di serializzazione/deserializzazione (SerDe) che viene utilizzato come estrattore e loader.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del SerDe.

- `SerializationLibrary`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  In genere la classe che implementa il SerDe. Un esempio è `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe`.
- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono i parametri di inizializzazione per SerDe.

Struttura dell'ordine

Specifica l'ordine di una colonna ordinata.

Campi

- `Column`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della colonna.
- `SortOrder`: obbligatorio: : numero (intero), non superiore a 1.

  Indica che la colonna è in ordine crescente (`==  1`) o in ordine decrescente (`==0`).

Struttura SkewedInfo

Specifica i valori disallineati in una tabella. I valori disallineati sono quelli che si verificano con una frequenza molto elevata.

Campi

- `SkewedColumnNames`: una matrice di stringhe UTF-8.

  Un elenco di nomi delle colonne contenenti i valori disallineati.
- `SkewedColumnValues`: una matrice di stringhe UTF-8.

Un elenco di valori che appaiono così frequentemente da poter essere considerati disallineati.

- `SkewedColumnValueLocationMaps`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Una mappatura di valori disallineati per le colonne che li contengono.

Struttura TableVersion

Specifica una versione di una tabella.

Campi

- `Table`: un oggetto Tabella.

  La tabella in questione

- `VersionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il valore ID identificativo di questa versione della tabella. `VersionId` è una rappresentazione di stringa di un numero intero. Ogni versione viene incrementata di 1.

Struttura TableError

Un record di errore per le operazioni di tabella.

Campi

- `TableName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella. Per la compatibilità Hive, deve essere interamente in caratteri minuscoli.

- `ErrorDetail`: un oggetto ErrorDetail.

  I dettagli sull'errore.

Struttura TableVersionError

Un record di errore per le operazioni table-version.

Campi

- `TableName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella in questione.
- `VersionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il valore ID della versione in questione. `VersionID` è una rappresentazione di stringa di un numero intero. Ogni versione viene incrementata di 1.
- `ErrorDetail`: un oggetto ErrorDetail.

  I dettagli sull'errore.

Struttura SortCriterion

Specifica un campo in base al quale ordinare e un ordinamento.

Campi

- `FieldName`: stringa Value, non superiore a 1024 byte di lunghezza.

  Il nome del campo in base al quale eseguire l'ordinamento.
- `Sort`: stringa UTF-8 (valori validi: `ASC="ASCENDING"` | `DESC="DESCENDING"`).

  Un ordinamento crescente o decrescente.

Struttura TableIdentifier

Una struttura che descrive una tabella di destinazione per il collegamento delle risorse.

Campi

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID del catalogo dati in cui risiede la tabella.

- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database di catalogo che contiene la tabella di destinazione.

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella di destinazione.

- `Region`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  La regione della tabella di destinazione.

Struttura KeySchemaElement

Una coppia di chiavi di partizione costituita da un nome e un tipo.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di una chiave di partizione.

- `Type`: obbligatorio: stringa UTF-8, non superiore a 131072 byte di lunghezza, corrispondente a Single-line string pattern.

  Il tipo di una chiave di partizione.

Struttura PartitionIndex

Una struttura per un indice della partizione.

Campi

- `Keys`: obbligatorio: una matrice di stringhe UTF-8, almeno 1 stringa.

  Le chiavi per l'indice della partizione.

- `IndexName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome dell'indice della partizione.

Struttura PartitionIndexDescriptor

Un descrittore per un indice della partizione in una tabella.

Campi

- `IndexName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome dell'indice della partizione.
- `Keys`: obbligatorio: una matrice di oggetti KeySchemaElement, almeno 1 struttura.

  Un elenco di una o più chiavi, come strutture `KeySchemaElement`, per l'indice della partizione.
- `IndexStatus`: obbligatorio: stringa UTF-8 (valori validi: `CREATING` | `ACTIVE` | `DELETING` | `FAILED`).

  Lo stato dell'indice della partizione.

  I possibili stati sono:
  - CREATING: l'indice è in fase di creazione. Quando un indice è in uno stato CREATING, l'indice o la relativa tabella non possono essere eliminati.
  - ACTIVE: la creazione dell'indice ha esito positivo.
  - FAILED: la creazione dell'indice non riesce.
  - DELETING: l'indice viene eliminato dall'elenco degli indici.
- `BackfillErrors`: una matrice di oggetti BackfillError.

  Un elenco degli errori che possono verificarsi durante la registrazione degli indici delle partizioni per una tabella esistente.

Struttura BackfillError

Un elenco degli errori che possono verificarsi durante la registrazione degli indici delle partizioni per una tabella esistente.

Questi errori forniscono i dettagli sul motivo per cui una registrazione dell'indice non è riuscita e forniscono un numero limitato di partizioni nella risposta, in modo da poter correggere le partizioni in errore e provare a registrare nuovamente l'indice. La serie più comune di errori che possono verificarsi sono classificati come segue:

- EncryptedPartitionError: le partizioni sono crittografate.
- InvalidPartitionTypeDataError: il valore della partizione non corrisponde al tipo di dati per la colonna della partizione.
- MissingPartitionValueError: le partizioni sono crittografate.
- UnsupportedPartitionCharacterError: i caratteri all'interno del valore della partizione non sono supportati. Ad esempio: U+0000, U+0001, U+0002.
- InternalError: qualsiasi errore che non appartiene ad altri codici di errore.

Campi

- `Code`: stringa UTF-8 (valori validi: `ENCRYPTED_PARTITION_ERROR` | `INTERNAL_ERROR` | `INVALID_PARTITION_TYPE_DATA_ERROR` | `MISSING_PARTITION_VALUE_ERROR` | `UNSUPPORTED_PARTITION_CHARACTER_ERROR`).

  Il codice di errore per un errore che si è verificato durante la registrazione degli indici delle partizioni per una tabella esistente.
- `Partitions`: una matrice di oggetti PartitionValueList.

  Un elenco di un numero limitato di partizioni nella risposta.

Struttura IcebergInput

Una struttura che definisce una tabella di metadati di Apache Iceberg da creare nel catalogo.

Campi

- `MetadataOperation`: obbligatorio: stringa UTF-8 (valori validi: `CREATE`).

  Un'operazione sui metadati richiesta. Questa opzione può essere impostata solo su `CREATE`.
- `Version`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  La versione della tabella per le tabelle Iceberg. L'impostazione predefinita è 2.

Struttura OpenTableFormatInput

Una struttura che rappresenta una tabella in formato aperto.

Campi

- `IcebergInput`: un oggetto IcebergInput.

  Specifica una struttura `IcebergInput` che definisce una tabella di metadati di Apache Iceberg.

Operazioni

- Operazione CreateTable (Python: create_table)
- Operazione UpdateTable (Python: update_table)
- Operazione DeleteTable (Python: delete_table)
- Operazione BatchDeleteTable (Python: batch_delete_table)
- Operazione GetTable (Python: get_table)
- Operazione GetTables (Python: get_tables)
- Operazione GetTableVersion (Python: get_table_version)
- Operazione GetTableVersions (Python: get_table_versions)
- Operazione DeleteTableVersion (Python: delete_table_version)
- Operazione BatchDeleteTableVersion (Python: batch_delete_table_version)
- Operazione SearchTables (Python: search_tables)
- Operazione GetPartitionIndexes (Python: get_partition_indexes)
- Operazione CreatePartitionIndex (Python: create_partition_index)
- Operazione DeletePartitionIndex (Python: delete_partition_index)
- Operazione GetColumnStatisticsForTable (Python: get_column_statistics_for_table)
- Operazione UpdateColumnStatisticsForTable Action (Python: update_column_statistics_for_table)
- Operazione DeleteColumnStatisticsForTable (Python: delete_column_statistics_for_table)

Operazione CreateTable (Python: create_table)

Crea una nuova definizione di tabella nel catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui creare la `Table`. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il database del catalogo in cui creare la nuova tabella. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `TableInput`: obbligatorio: un oggetto TableInput.

  L'oggetto `TableInput` che definisce la tabella di metadati da creare nel catalogo.
- `PartitionIndexes`: una matrice di oggetti PartitionIndex, non superiore a 3 strutture.

  Un elenco di indici delle partizioni, `PartitionIndex` strutture, da creare nella tabella.
- `TransactionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #11.

  L'ID della transazione.
- `OpenTableFormatInput`: un oggetto OpenTableFormatInput.

  Specifica una struttura `OpenTableFormatInput` durante la creazione di una tabella in formato aperto.

Risposta

- Nessun parametro di risposta.

Errori

- `AlreadyExistsException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`

- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`
- `ResourceNotReadyException`

Operazione UpdateTable (Python: update_table)

Aggiorna una tabella di metadati nel catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiede la tabella. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `TableInput`: obbligatorio: un oggetto TableInput.

  Un'oggetto `TableInput` avanzato per la definizione della tabella di metadati nel catalogo.
- `SkipArchive`: booleano.

  Per impostazione predefinita, `UpdateTable` crea sempre una versione archiviata della tabella prima di aggiornarla. Se tuttavia `skipArchive` è impostato su true, `UpdateTable` non crea la versione archiviata.
- `TransactionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #11.

  ID transazione in cui aggiornare il contenuto della tabella.
- `VersionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

ID della versione in cui aggiornare il contenuto della tabella.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`
- `ResourceNotReadyException`

Operazione DeleteTable (Python: delete_table)

Rimuove una definizione di tabella dal catalogo dati.

> **Note**
>
> Una volta completata questa operazione, non potrai più accedere alle versioni e alle partizioni delle tabelle che appartengono alle tabelle eliminate. AWS Glue elimina tempestivamente queste risorse "orfane" in modo asincrono, a discrezione del servizio.
> Per garantire l'eliminazione immediata di tutte le risorse correlate, prima di chiamare `DeleteTable`, utilizza `DeleteTableVersion` o `BatchDeleteTableVersion` e `DeletePartition` o `BatchDeletePartition` per eliminare eventuali risorse che appartengono alla tabella.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiede la tabella. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella da eliminare. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.

- `TransactionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #11.

  ID transazione in cui eliminare il contenuto della tabella.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `ResourceNotReadyException`

Operazione BatchDeleteTable (Python: batch_delete_table)

Elimina più tabelle contemporaneamente.

> **Note**
>
> Una volta completata questa operazione, non potrai più accedere alle versioni e alle partizioni delle tabelle che appartengono alle tabelle eliminate. AWS Glue elimina tempestivamente queste risorse "orfane" in modo asincrono, a discrezione del servizio.
> Per garantire l'eliminazione immediata di tutte le risorse correlate, prima di chiamare `BatchDeleteTable`, utilizza `DeleteTableVersion` o `BatchDeleteTableVersion` e `DeletePartition` o `BatchDeletePartition` per eliminare eventuali risorse che appartengono alla tabella.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiede la tabella da eliminare. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `TablesToDelete`: obbligatorio: una matrice di stringhe UTF-8, non superiore a 100 stringhe.

  Un elenco della tabella da eliminare.
- `TransactionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #11.

  ID transazione in cui eliminare il contenuto della tabella.

Risposta

- `Errors`: una matrice di oggetti TableError.

Un elenco di errori riscontrati nel tentativo di eliminazione delle tabelle specificate.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ResourceNotReadyException`

Operazione GetTable (Python: get_table)

Consente di recuperare la definizione `Table` in un catalogo dati per una tabella specificata.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database nel catalogo in cui risiede la tabella. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella per cui recuperare la definizione. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `TransactionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #11.

  ID transazione in cui leggere il contenuto della tabella.

- `QueryAsOfTime`: timestamp.

  Il momento a partire dal quale leggere il contenuto della tabella. Se non è impostato, verrà utilizzato l'orario di esecuzione del commit della transazione più recente. Non può essere specificato insieme a `TransactionId`.

Risposta

- `Table`: un oggetto Tabella.

  L'oggetto `Table` che definisce la tabella specificata.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Operazione GetTables (Python: get_tables)

Consente di recuperare le definizioni di alcune o di tutte le tabelle in un determinato `Database`.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il database nel catalogo delle tabelle da elencare. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.

- `Expression`: stringa UTF-8, non superiore a 2048 byte di lunghezza, corrispondente a Single-line string pattern.

  Un modello di espressione regolare. Se presente, vengono restituite solo le tabelle con i nomi corrispondenti al modello.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, incluso se si tratta di una chiamata di continuazione.

- `MaxResults` – Numero (intero), non inferiore a 1 o superiore a 100.

  Il numero massimo di tabelle da restituire in una risposta singola.

- `TransactionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #11.

  ID transazione in cui leggere il contenuto della tabella.

- `QueryAsOfTime`: timestamp.

  Il momento a partire dal quale leggere il contenuto della tabella. Se non è impostato, verrà utilizzato l'orario di esecuzione del commit della transazione più recente. Non può essere specificato insieme a `TransactionId`.

Risposta

- `TableList`: una matrice di oggetti Tabella.

  Un elenco di tutti gli oggetti `Table` richiesti.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, presente se il segmento dell'elenco corrente non è l'ultimo.

Errori

- `EntityNotFoundException`
- `InvalidInputException`

- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Operazione GetTableVersion (Python: get_table_version)

Consente di recuperare una versione specificata di una tabella.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui la tabella risiede. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `VersionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il valore ID della versione della tabella da recuperare. `VersionID` è una rappresentazione di stringa di un numero intero. Ogni versione viene incrementata di 1.

Risposta

- `TableVersion`: un oggetto TableVersion.

La versione richiesta della tabella.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione GetTableVersions (Python: get_table_versions)

Consente di recuperare un elenco di stringhe identificativo delle versioni disponibili di una tabella specificata.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui la tabella risiede. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non è la prima chiamata.
- `MaxResults` – Numero (intero), non inferiore a 1 o superiore a 100.

Il numero massimo di versioni della tabella da restituire in una risposta.

Risposta

- `TableVersions`: una matrice di oggetti TableVersion.

  Un elenco di stringhe identificativo delle versioni disponibili della tabella specificata.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se l'elenco delle versioni disponibili non comprende l'ultima.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione DeleteTableVersion (Python: delete_table_version)

Elimina una versione specificata di una tabella.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui la tabella risiede. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.

- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.

- `VersionId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il valore ID della versione della tabella da eliminare. `VersionID` è una rappresentazione di stringa di un numero intero. Ogni versione viene incrementata di 1.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Operazione BatchDeleteTableVersion (Python: batch_delete_table_version)

Elimina un batch di versioni specificato di una tabella.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome del database del catalogo in cui la tabella risiede. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.

- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella. Per la compatibilità Hive, questo nome è scritto interamente in caratteri minuscoli.

- `VersionIds`. Obbligatorio: una serie di stringhe UTF-8, non superiore a 100 stringhe.

  Un elenco degli ID delle versioni da eliminare. `VersionId` è una rappresentazione di stringa di un numero intero. Ogni versione viene incrementata di 1.

Risposta

- `Errors`: una matrice di oggetti TableVersionError.

  Un elenco di errori riscontrati nel tentativo di eliminazione delle versioni della tabella specificata.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Operazione SearchTables (Python: search_tables)

Cerca un set di tabelle in base alle proprietà nei metadati della tabella nonché nel database padre. Puoi eseguire ricerche su condizioni di testo o filtro.

Puoi ottenere solo tabelle a cui hai accesso in base alle policy di sicurezza definite in Lake Formation. È necessario almeno un accesso in sola lettura alla tabella affinché venga restituita. Se non disponi dell'accesso a tutte le colonne della tabella, non verranno eseguite ricerche in queste colonne quando l'elenco delle tabelle viene restituito. Se disponi dell'accesso alle colonne ma non ai dati nelle colonne, tali colonne e i metadati associati a tali colonne saranno inclusi nella ricerca.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un identificatore univoco, costituito da *`account_id`*.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, incluso se si tratta di una chiamata di continuazione.

- `Filters`: una matrice di oggetti PropertyPredicate.

  Un elenco di coppie chiave-valore e un comparatore utilizzato per filtrare i risultati della ricerca. Restituisce tutte le entità che corrispondono al predicato.

  Il membro `Comparator` della struttura `PropertyPredicate` viene utilizzata solo per i campi ora e può essere omesso per altri tipi di campo. Inoltre, quando si confrontano i valori di stringa, ad esempio quando `Key=Name`, viene utilizzato un algoritmo di corrispondenza parziale. Il campo `Key` (ad esempio, il valore del campo `Name`) è diviso su determinati caratteri di punteggiatura, ad esempio -, :, #, ecc. in token. Quindi ogni token è una corrispondenza esatta rispetto al membro `Value` di `PropertyPredicate`. Ad esempio se `Key=Name` e `Value=link`, le tabelle denominate `customer-link` e `xx-link-yy` vengono restituite, ma`xxlinkyy` non viene restituita.

- `SearchText` – Stringa Value, non superiore a 1024 byte di lunghezza.

  Una stringa utilizzata per una ricerca di testo.

  Specifica un valore in filtri tra virgolette basato su una corrispondenza esatta con il valore.

- `SortCriteria`: una matrice di oggetti SortCriterion, non superiore a 1 struttura.

  Un elenco di criteri per ordinare i risultati in base a un nome di campo, in ordine crescente o decrescente.

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Il numero massimo di tabelle da restituire in una risposta singola.

- `ResourceShareType`: stringa UTF-8 (valori validi: `FOREIGN` | `ALL` | `FEDERATED`).

  Consente di specificare che si desidera eseguire la ricerca nelle tabelle condivise con l'account. I valori consentiti sono `FOREIGN` o `ALL`.

  - Se impostato su `FOREIGN`, cercherà le tabelle condivise con l'account.

- Se impostato su `ALL`, cercherà le tabelle condivise con l'account, così come le tabelle nell'account locale.

Risposta

- `NextToken`: stringa UTF-8.

  Un token di continuazione, presente se il segmento dell'elenco corrente non è l'ultimo.
- `TableList`: una matrice di oggetti Tabella.

  Un elenco di tutti gli oggetti `Table` richiesti. La risposta `SearchTables` restituisce solo le tabelle a cui hai accesso.

Errori

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Operazione GetPartitionIndexes (Python: get_partition_indexes)

Recupera gli indici delle partizioni associati a una tabella.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo in cui si trova la tabella.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Specifica il nome di un database da cui si desidera recuperare gli indici delle partizioni.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Specifica il nome di una tabella da cui si desidera recuperare gli indici delle partizioni.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, incluso se si tratta di una chiamata di continuazione.

Risposta

- `PartitionIndexDescriptorList`: una matrice di oggetti PartitionIndexDescriptor.

  Un elenco di descrittori di indice.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, presente se il segmento dell'elenco corrente non è l'ultimo.

Errori

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ConflictException`

Operazione CreatePartitionIndex (Python: create_partition_index)

Crea un indice della partizione specificato in una tabella esistente.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo in cui si trova la tabella.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Specifica il nome di un database in cui si desidera creare un indice della partizione.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Specifica il nome di una tabella in cui si desidera creare un indice della partizione.

- `PartitionIndex`: obbligatorio: un oggetto PartitionIndex.

  Specifica una struttura `PartitionIndex` per creare un indice della partizione in una tabella esistente.

Risposta

- Nessun parametro di risposta.

Errori

- `AlreadyExistsException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione DeletePartitionIndex (Python: delete_partition_index)

Elimina un indice della partizione specificato da una tabella esistente.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo in cui si trova la tabella.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Specifica il nome di un database da cui si desidera eliminare un indice della partizione.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Specifica il nome di una tabella da cui si desidera eliminare un indice della partizione.

- `IndexName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome dell'indice della partizione da eliminare.

Risposta

- Nessun parametro di risposta.

Errori

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ConflictException`
- `GlueEncryptionException`

Operazione GetColumnStatisticsForTable (Python: get_column_statistics_for_table)

Recupera le statistiche delle colonne della tabella.

L'autorizzazione Identity and Access Management (IAM) necessaria per questa operazione è `GetTable`.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le partizioni. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiedono le partizioni.

- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.
- `ColumnNames`. Obbligatorio: una serie di stringhe UTF-8, non superiore a 100 stringhe.

  Un elenco dei nomi delle colonne.

Risposta

- `ColumnStatisticsList`: una matrice di oggetti ColumnStatistics.

  Elenco di ColumnStatistics.
- `Errors`: una matrice di oggetti ColumnError.

  Elenco di ColumnStatistics che non è stato possibile recuperare.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione UpdateColumnStatisticsForTable Action (Python: update_column_statistics_for_table)

Crea o aggiorna le statistiche delle tabelle della colonna.

L'autorizzazione Identity and Access Management (IAM) necessaria per questa operazione è `UpdateTable`.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID del catalogo dati in cui si trovano le partizioni. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiedono le partizioni.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.
- `ColumnStatisticsList`: obbligatorio: una matrice di oggetti ColumnStatistics, non superiore a 25 strutture.

  Un elenco delle statistiche delle colonne.

Risposta

- `Errors`: una matrice di oggetti ColumnStatisticsError.

  Elenco di ColumnStatisticsErrors.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione DeleteColumnStatisticsForTable (Python: delete_column_statistics_for_table)

Recupera le statistiche delle colonne della tabella.

L'autorizzazione Identity and Access Management (IAM) necessaria per questa operazione è `DeleteTable`.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le partizioni. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiedono le partizioni.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.
- `ColumnName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della colonna.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

API della partizione

L'API Partition descrive i tipi di dati e le operazioni utilizzate per lavorare con le partizioni.

Tipi di dati

Struttura della partizione

Rappresenta una porzione dei dati della tabella.

Campi

- `Values`: una matrice di stringhe UTF-8.

  I valori della partizione.
- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui creare la partizione.
- `TableName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella di database in cui creare la partizione.
- `CreationTime`: timestamp.

L'ora in cui è stata creata la partizione.

- `LastAccessTime`: timestamp.

  L'ora in cui è stato effettuato l'ultimo accesso alla partizione.

- `StorageDescriptor`: un oggetto StorageDescriptor.

  Fornisce informazioni sulla posizione fisica in cui è memorizzata la partizione.

- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono i parametri per la partizione.

- `LastAnalyzedTime`: timestamp.

  L'ultima volta in cui sono state calcolate le statistiche di colonna per questa partizione.

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le partizioni.

Struttura PartitionInput

Struttura utilizzata per la creazione e per l'aggiornamento di una partizione.

Campi

- `Values`: una matrice di stringhe UTF-8.

  I valori della partizione. Sebbene questo parametro non sia richiesto dall'SDK, è necessario specificarlo per un input valido.

  I valori delle chiavi per la nuova partizione devono essere passati come una matrice di oggetti String che devono essere sistemati seguendo lo stesso ordine delle chiavi di partizione che appaiono nel prefisso Amazon S3. In caso contrario, AWS Glue aggiungerà i valori alle chiavi errate.

- `LastAccessTime`: timestamp.

  L'ora in cui è stato effettuato l'ultimo accesso alla partizione.
- `StorageDescriptor`: un oggetto StorageDescriptor.

  Fornisce informazioni sulla posizione fisica in cui è memorizzata la partizione.
- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono i parametri per la partizione.
- `LastAnalyzedTime`: timestamp.

  L'ultima volta in cui sono state calcolate le statistiche di colonna per questa partizione.

Struttura PartitionSpecWithSharedStorageDescriptor

Una specifica per le partizioni che condividono una posizione fisica.

Campi

- `StorageDescriptor`: un oggetto StorageDescriptor.

  Le informazioni condivise sullo storage fisico.
- `Partitions`: una matrice di oggetti Partizione.

  Un elenco di partizioni che condividono questa posizione fisica.

Struttura PartitionListComposingSpec

Elenca le partizioni correlate.

Campi

- `Partitions`: una matrice di oggetti Partizione.

  Un elenco di partizioni nella specifica di composizione.

Struttura PartitionSpecProxy

Fornisce un percorso radice per partizioni specifiche.

Campi

- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il database del catalogo in cui risiedono le partizioni.
- `TableName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella che contiene le partizioni.
- `RootPath`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il percorso radice del proxy per gestire le partizioni.
- `PartitionSpecWithSharedSD`: un oggetto PartitionSpecWithSharedStorageDescriptor.

  Una specifica delle partizioni che condividono la stessa posizione dello storage fisico.
- `PartitionListComposingSpec`: un oggetto PartitionListComposingSpec.

  Specifica un elenco di partizioni.

Struttura PartitionValueList

Contiene un elenco di valori che definiscono le partizioni.

Campi

- `Values`: obbligatorio: una matrice di stringhe UTF-8.

  L'elenco dei valori.

Struttura del segmento

Definisce una regione non sovrapposta delle partizioni di una tabella, consentendo l'esecuzione di più richieste in parallelo.

Campi

- `SegmentNumber`: obbligatorio: numero (intero), non superiore a Nessuno.

  Il numero dell'indice a base zero del segmento. Ad esempio, se il numero totale di segmenti è 4, i alori di `SegmentNumber` vanno da 0 a 3.
- `TotalSegments`: obbligatorio: numero (intero), non inferiore a 1 o superiore a 10.

  Il numero totale dei segmenti.

Struttura PartitionError

Contiene informazioni sull'errore di una partizione.

Campi

- `PartitionValues`: una matrice di stringhe UTF-8.

  I valori che definiscono la partizione.
- `ErrorDetail`: un oggetto ErrorDetail.

  Dettagli sull'errore della partizione.

Struttura BatchUpdatePartitionFailureEntry

Contiene informazioni sull'errore di una partizione di aggiornamento in batch.

Campi

- `PartitionValueList`: una matrice di stringhe UTF-8, non superiore a 100.

  Un elenco di valori che definiscono le partizioni.
- `ErrorDetail`: un oggetto ErrorDetail.

  Dettagli sull'errore della partizione di aggiornamento in batch.

Struttura BatchUpdatePartitionRequestEntry

Una struttura che contiene i valori e la struttura utilizzati per aggiornare una partizione.

Campi

- `PartitionValueList`. Obbligatorio: una serie di stringhe UTF-8, non superiore a 100 stringhe.

  Un elenco di valori che definiscono le partizioni.
- `PartitionInput`: obbligatorio: un oggetto PartitionInput.

  Struttura utilizzata per l'aggiornamento di una partizione.

Struttura StorageDescriptor

Descrive lo storage fisico dei dati della tabella.

Campi

- `Columns`: una matrice di oggetti Colonna.

  Un elenco delle `Columns` nella tabella.
- `Location`: stringa di posizione, non superiore a 2056 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La posizione fisica della tabella. Per impostazione predefinita questo utilizza il modulo della posizione del warehouse, seguito dalla posizione del database nel warehouse, seguito dal nome della tabella.
- `AdditionalLocations`: una matrice di stringhe UTF-8.

  Un elenco di posizioni che puntano al percorso in cui si trova una tabella Delta.
- `InputFormat`: stringa di formato, non superiore a 128 byte di lunghezza, corrispondente a Single-line string pattern.

  Il formato di input: `SequenceFileInputFormat` (binario) o `TextInputFormat` o un formato personalizzato.
- `OutputFormat`: stringa di formato, non superiore a 128 byte di lunghezza, corrispondente a Single-line string pattern.

  Il formato di output: `SequenceFileOutputFormat` (binario) o `IgnoreKeyTextOutputFormat` o un formato personalizzato.
- `Compressed`: booleano.

`True` se i dati nella tabella sono compressi, in caso contrario `False`.

- `NumberOfBuckets`: numero (intero).

  Deve essere specificato se la tabella contiene una qualsiasi colonna di dimensione.

- `SerdeInfo`: un oggetto SerDeInfo.

  Informazioni di serializzazione/deserializzazione (SerDe).

- `BucketColumns`: una matrice di stringhe UTF-8.

  Un elenco di colonne per il raggruppamento del reducer, colonne di clustering, colonne di bucketing nella tabella.

- `SortColumns`: una matrice di oggetti Order.

  Un elenco specificando l'ordine di ciascuna bucket nella tabella.

- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Le proprietà fornite dall'utente nel modulo chiave-valore.

- `SkewedInfo`: un oggetto SkewedInfo.

  Informazioni sui valori che appaiono di frequente in una colonna (valori disallineati).

- `StoredAsSubDirectories`: booleano.

  `True` se i dati nella tabella sono archiviati nelle sottodirectory, in caso contrario `False`.

- `SchemaReference`: un oggetto SchemaReference.

  Un oggetto che fa riferimento a uno schema memorizzato nel Registro degli schemi di AWS Glue.

  Quando crei una tabella, puoi passare un elenco vuoto di colonne per lo schema e utilizzare invece un riferimento allo schema.

Struttura SchemaReference

Un oggetto che fa riferimento a uno schema memorizzato nel Registro degli schemi di AWS Glue.

Campi

- `SchemaId`: un oggetto SchemaId.

  Una struttura che contiene campi di identità dello schema. Deve essere fornito questo o `SchemaVersionId`.

- `SchemaVersionId`: stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID univoco assegnato a una versione dello schema. Deve essere fornito questo o `SchemaId`.

- `SchemaVersionNumber`: numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione dello schema.

Struttura SerDeInfo

Informazioni su un programma di serializzazione/deserializzazione (SerDe) che viene utilizzato come estrattore e loader.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del SerDe.

- `SerializationLibrary`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  In genere la classe che implementa il SerDe. Un esempio è `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe`.

- `Parameters`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa chiave, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8, lunga non più di 512000 byte.

  Queste coppie chiave-valore definiscono i parametri di inizializzazione per SerDe.

Struttura SkewedInfo

Specifica i valori disallineati in una tabella. I valori disallineati sono quelli che si verificano con una frequenza molto elevata.

Campi

- `SkewedColumnNames`: una matrice di stringhe UTF-8.

  Un elenco di nomi delle colonne contenenti i valori disallineati.
- `SkewedColumnValues`: una matrice di stringhe UTF-8.

  Un elenco di valori che appaiono così frequentemente da poter essere considerati disallineati.
- `SkewedColumnValueLocationMaps`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Una mappatura di valori disallineati per le colonne che li contengono.

Operazioni

- Operazione CreatePartition (Python: create_partition)
- Operazione BatchCreatePartition (Python: batch_create_partition)
- Operazione UpdatePartition (Python: update_partition)
- Operazione DeletePartition (Python: delete_partition)
- Operazione BatchDeletePartition (Python: batch_delete_partition)
- Operazione GetPartition (Python: get_partition)
- Operazione GetPartitions (Python: get_partitions)
- Operazione BatchGetPartition (Python: batch_get_partition)
- Operazione BatchUpdatePartition (Python: batch_update_partition)
- Operazione GetColumnStatisticsForPartition (Python: get_column_statistics_for_partition)
- Operazione UpdateColumnStatisticsForPartition (Python: update_column_statistics_for_partition)
- Operazione DeleteColumnStatisticsForPartition (Python: delete_column_statistics_for_partition)

Operazione CreatePartition (Python: create_partition)

Crea una nuova partizione.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID dell'account AWS del catalogo in cui la partizione deve essere creata.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database dei metadata in cui deve essere creata la partizione.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella dei metadata in cui deve essere creata la partizione.
- `PartitionInput`: obbligatorio: un oggetto PartitionInput.

  Una struttura `PartitionInput` che definisce la partizione da creare.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione BatchCreatePartition (Python: batch_create_partition)

Crea una o più partizioni in un'operazione in batch.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo in cui deve essere creata la partizione. Al momento, dovrebbe essere l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database dei metadata in cui deve essere creata la partizione.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella dei metadata in cui deve essere creata la partizione.
- `PartitionInputList`: obbligatorio: una matrice di oggetti PartitionInput, non superiore a 100 strutture.

  Un elenco di strutture `PartitionInput` che definiscono le partizioni da creare.

Risposta

- `Errors`: una matrice di oggetti PartitionError.

  Errori rilevati durante il tentativo di creare le partizioni richieste.

Errori

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`

- `GlueEncryptionException`

Operazione UpdatePartition (Python: update_partition)

Aggiorna una partizione.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trova la partizione da aggiornare. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella in cui si trova la partizione da aggiornare.
- `PartitionValueList`. Obbligatorio: una serie di stringhe UTF-8, non superiore a 100 stringhe.

  Elenco dei valori della chiave della partizione che definiscono la partizione da aggiornare.
- `PartitionInput`: obbligatorio: un oggetto PartitionInput.

  Il nuovo oggetto della partizione a cui aggiornare la partizione.

  La proprietà `Values` non può essere modificata. Se si desidera modificare i valori della chiave per una partizione, è necessario eliminare e creare nuovamente la partizione.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione DeletePartition (Python: delete_partition)

Elimina una partizione specificata.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trova la partizione da eliminare. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella che contiene la partizione da eliminare.
- `PartitionValues`. Obbligatorio: una serie di stringhe UTF-8.

  I valori che definiscono la partizione.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`

- `OperationTimeoutException`

Operazione BatchDeletePartition (Python: batch_delete_partition)

Cancella una o più partizioni in un'operazione in batch.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trova la partizione da eliminare. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella che contiene le partizioni da eliminare.
- `PartitionsToDelete`. Obbligatorio: una serie di oggetti PartitionValueList, non superiore a 25 strutture.

  Un elenco di strutture `PartitionInput` che definiscono le partizioni da eliminare.

Risposta

- `Errors`: una matrice di oggetti PartitionError.

  Errori rilevati durante il tentativo di cancellare le partizioni richieste.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Operazione GetPartition (Python: get_partition)

Consente di recuperare informazioni su una partizione specificata.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trova la partizione. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiede la partizione.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.
- `PartitionValues`. Obbligatorio: una serie di stringhe UTF-8.

  I valori che definiscono la partizione.

Risposta

- `Partition`: un oggetto Partizione.

  Le informazioni richieste, sotto forma di un oggetto della `Partition`.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `FederationSourceException`

- `FederationSourceRetryableException`

Operazione GetPartitions (Python: get_partitions)

Recupera informazioni sulle partizioni in una tabella.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le partizioni. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiedono le partizioni.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.
- `Expression`: stringa predicato, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Espressione che filtra le partizioni da restituire.

  L'espressione usa la sintassi SQL simile alla clausola di filtro SQL `WHERE`. Il parser di istruzioni SQL JSQLParser analizza l'espressione.

  Operatori: di seguito sono elencati gli operatori che puoi usare nella chiamata API `Expression`:

  =

  Verifica se i valori dei due operandi sono uguali. In caso affermativo, la condizione diventa true.

  Esempio: presupponiamo che "variable a" contenga 10 e "variable b" contenga 20.

  (a = b) non è true.

  < >

  Verifica se i valori dei due operandi sono uguali. In caso negativo, la condizione diventa true.

Esempio: (a < > b) è true.

>

Verifica se il valore dell'operando di sinistra è superiore al valore dell'operando di destra. In caso affermativo, la condizione diventa true.

Esempio: (a > b) non è true.

<

Verifica se il valore dell'operando di sinistra è inferiore al valore dell'operando di destra. In caso affermativo, la condizione diventa true.

Esempio: (a < b) è true.

>=

Verifica se il valore dell'operando di sinistra è superiore o uguale al valore dell'operando di destra. In caso affermativo, la condizione diventa true.

Esempio: (a >= b) non è true.

<=

Verifica se il valore dell'operando di sinistra è inferiore o uguale al valore dell'operando di destra. In caso affermativo, la condizione diventa true.

Esempio: (a <= b) è true.

AND, OR, IN, BETWEEN, LIKE NOT, IS NULL

Operatori logici.

Tipi di chiave di partizione supportati: di seguito sono indicate le chiavi di partizione supportate.

- `string`
- `date`
- `timestamp`
- `int`
- `bigint`
- `long`
- `tinyint`

- `smallint`
- `decimal`

Se viene riscontrato un tipo non valido, viene generata un'eccezione.

L'elenco seguente mostra gli operatori validi per ogni tipo. Quando definisci un crawler, il tipo `partitionKey` viene creato come `STRING`, perché sia compatibile con le partizioni del catalogo.

Chiamata API di esempio:

Example

La tabella `twitter_partition` contiene tre partizioni:

```
year = 2015
        year = 2016
        year = 2017
```

Example

Get Partition `year` equivale a 2015

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year*=*'2015'"
```

Example

Get partition `year` tra 2016 e 2018 (esclusi)

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year>'2016' AND year<'2018'"
```

Example

Get partition `year` tra 2015 e 2018 (inclusi). Le chiamate API seguenti si equivalgono tra loro

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year>='2015' AND year<='2018'"
```

```
        aws glue get-partitions --database-name dbname --table-name
 twitter_partition
        --expression "year BETWEEN 2015 AND 2018"

        aws glue get-partitions --database-name dbname --table-name
 twitter_partition
        --expression "year IN (2015,2016,2017,2018)"
```

Example

Un filtro di partizione con caratteri jolly, in cui l'output di chiamata seguente è l'anno di partizione = 2017. Un'espressione regolare non è supportata in `LIKE`.

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year LIKE '%7'"
```

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non è la prima chiamata per recuperare le partizioni.

- `Segment`: un oggetto Segment.

  Il segmento delle partizioni della tabella per analizzare questa richiesta.

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Il numero massimo di partizioni da restituire in una risposta singola.

- `ExcludeColumnSchema`: booleano.

  Se vero, specifica di non restituire lo schema della colonna di partizione. Utile quando sei interessato solo ad altri attributi di partizione, come i valori o la posizione delle partizioni. Questo approccio evita il problema di una risposta ampia non restituendo dati duplicati.

- `TransactionId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #11.

  ID transazione in cui leggere il contenuto della partizione.

- `QueryAsOfTime`: timestamp.

  Il momento a partire dal quale leggere il contenuto della partizione. Se non è impostato, verrà utilizzato l'orario di esecuzione del commit della transazione più recente. Non può essere specificato insieme a `TransactionId`.

Risposta

- `Partitions`: una matrice di oggetti Partizione.

  Un elenco di partizioni richieste.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se l'elenco restituito di partizioni non include l'ultima.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `InvalidStateException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Operazione BatchGetPartition (Python: batch_get_partition)

Recupera le partizioni in una richiesta di batch.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le partizioni. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiedono le partizioni.

- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.

- `PartitionsToGet`: obbligatorio: una matrice di oggetti PartitionValueList, non superiore a 1000 strutture.

  Un elenco di valori della partizione che identificano le partizioni da recuperare.

Risposta

- `Partitions`: una matrice di oggetti Partizione.

  Un elenco di tutte le partizioni richieste.

- `UnprocessedKeys`: una matrice di oggetti PartitionValueList, non superiore a 1000 strutture.

  Un elenco dei valori della partizione nella richiesta per la quale le partizioni non sono state restituite.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `InvalidStateException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Operazione BatchUpdatePartition (Python: batch_update_partition)

Aggiorna una o più partizioni in un'operazione in batch.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo in cui deve essere aggiornata la partizione. Al momento, dovrebbe essere l'ID dell'account AWS.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database di metadati in cui deve essere aggiornata la partizione.

- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella di metadati in cui deve essere aggiornata la partizione.

- `Entries`: obbligatorio: una matrice di oggetti BatchUpdatePartitionRequestEntry, non meno di 1 o più di 100 strutture.

  Un elenco fino a 100 oggetti `BatchUpdatePartitionRequestEntry` da aggiornare.

Risposta

- `Errors`: una matrice di oggetti BatchUpdatePartitionFailureEntry.

  Errori rilevati durante il tentativo di aggiornamento delle partizioni richieste. Elenco di oggetti `BatchUpdatePartitionFailureEntry`.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`

Operazione GetColumnStatisticsForPartition (Python: get_column_statistics_for_partition)

Recupera le statistiche della partizione delle colonne.

L'autorizzazione Identity and Access Management (IAM) necessaria per questa operazione è `GetPartition`.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le partizioni. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiedono le partizioni.

- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.

- `PartitionValues`. Obbligatorio: una serie di stringhe UTF-8.

  Un elenco di valori della partizione che identificano la partizione.

- `ColumnNames`. Obbligatorio: una serie di stringhe UTF-8, non superiore a 100 stringhe.

  Un elenco dei nomi delle colonne.

Risposta

- `ColumnStatisticsList`: una matrice di oggetti ColumnStatistics.

  Elenco di ColumnStatistics che non è stato possibile recuperare.

- `Errors`: una matrice di oggetti ColumnError.

  Errore durante il recupero dei dati delle statistiche delle colonne.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione UpdateColumnStatisticsForPartition (Python: update_column_statistics_for_partition)

Crea o aggiorna le statistiche delle partizioni delle colonne.

L'autorizzazione Identity and Access Management (IAM) necessaria per questa operazione è `UpdatePartition`.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le partizioni. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiedono le partizioni.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.
- `PartitionValues`. Obbligatorio: una serie di stringhe UTF-8.

  Un elenco di valori della partizione che identificano la partizione.
- `ColumnStatisticsList`. Obbligatorio: una serie di oggetti ColumnStatistics, non superiore a 25 strutture.

  Un elenco delle statistiche delle colonne.

Risposta

- `Errors`: una matrice di oggetti ColumnStatisticsError.

  Errore durante l'aggiornamento dei dati delle statistiche delle colonne.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione DeleteColumnStatisticsForPartition (Python: delete_column_statistics_for_partition)

Elimina le statistiche della colonna della partizione di una colonna.

L'autorizzazione Identity and Access Management (IAM) necessaria per questa operazione è `DeletePartition`.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le partizioni. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui risiedono le partizioni.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella della partizione.

- `PartitionValues`. Obbligatorio: una serie di stringhe UTF-8.

  Un elenco di valori della partizione che identificano la partizione.

- `ColumnName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della colonna.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

API di connessione

L'API Connection descrive i tipi di dati di AWS Glue connessione e l'API per creare, eliminare, aggiornare ed elencare le connessioni.

Tipi di dati

- Struttura di connessione
- ConnectionInput struttura
- PhysicalConnectionRequirements struttura
- GetConnectionsFilter struttura

Struttura di connessione

Definisce una connessione a un'origine dati.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della definizione di connessione.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La descrizione della connessione.

- `ConnectionType`: stringa UTF-8 (valori validi: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM`).

  Il tipo di connessione. Attualmente, SFTP non è supportato.

- `MatchCriteria`: una matrice di stringhe UTF-8, non superiore a 10 stringhe.

  Un elenco di criteri che possono essere utilizzati nella selezione di questa connessione.

- `ConnectionProperties`: una matrice di mappe con coppie chiave-valore, non superiore alle 100 coppie.

  Ogni chiave è una stringa UTF-8 (valori validi: `HOST` | | `PORT` | `USERNAME="USER_NAME"` | `PASSWORD` | `ENCRYPTED_PASSWORD` | `JDBC_DRIVER_JAR_URI` | `JDBC_DRIVER_CLASS_NAME` | `JDBC_ENGINE` | `JDBC_ENGINE_VERSION` | `CONFIG_FILES` | | `INSTANCE_ID` | `JDBC_CONNECTION_URL` | `JDBC_ENFORCE_SSL` | `CUSTOM_JDBC_CERT` | `SKIP_CUSTOM_JDBC_CERT_VALIDATION` | `CUSTOM_JDBC_CERT_STRING` | `CONNECTION_URL` | `KAFKA_BOOTSTRAP_SERVERS` | `KAFKA_SSL_ENABLED` | | `KAFKA_CUSTOM_CERT` | `KAFKA_SKIP_CUSTOM_CERT_VALIDATION` | `KAFKA_CLIENT_KEYSTORE` | `KAFKA_CLIENT_KEYSTORE_PASSWORD` | `KAFKA_CLIENT_KEY_PASSWORD` | `ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD` | `ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD` | `SECRET_ID` | `CONNECTOR_URL` | | `CONNECTOR_TYPE` | `CONNECTOR_CLASS_NAME` | `KAFKA_SASL_MECHANISM` | `KAFKA_SASL_PLAIN_USERNAME` | `KAFKA_SASL_PLAIN_PASSWORD` `ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD` `KAFKA_SASL_SCRAM_USERNAME` `KAFKA_SASL_SCRAM_PASSWORD` | `KAFKA_SASL_SCRAM_SECRETS_ARN` | `ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD` | | `KAFKA_SASL_GSSAPI_KEYTAB` `KAFKA_SASL_GSSAPI_KRB5_CONF` `KAFKA_SASL_GSSAPI_SERVICE` |`KAFKA_SASL_GSSAPI_PRINCIPAL`).

Ogni valore è una stringa Value, lunga non più di 1024 byte.

Queste coppie chiave-valore definiscono i parametri per la connessione:

- `HOST`: URI host: il nome di dominio completo (FQDN) o l'indirizzo IPv4 dell'host del database.
- `PORT`: il numero della porta, compreso tra 1024 e 65535, della porta su cui l'host del database ascolta le connessioni del database.
- `USER_NAME`: il nome sotto il quale accedere al database. La stringa del valore per `USER_NAME` è `USERNAME`.
- `PASSWORD`: una password, se usata, per il nome utente.
- `ENCRYPTED_PASSWORD`: quando si abilita la protezione della password di connessione impostando `ConnectionPasswordEncryption` nelle impostazioni di crittografia del catalogo dati, questo campo memorizza la password crittografata.
- `JDBC_DRIVER_JAR_URI`: il percorso Amazon Simple Storage Service (Amazon S3) del file JAR che contiene il driver JDBC da utilizzare.
- `JDBC_DRIVER_CLASS_NAME`: il nome classe del driver JDBC da utilizzare.
- `JDBC_ENGINE`: il nome del motore JDBC da utilizzare.
- `JDBC_ENGINE_VERSION`: la versione del motore JDBC da utilizzare.
- `CONFIG_FILES`: (Riservato per uso futuro).
- `INSTANCE_ID`: l'ID istanza da utilizzare.
- `JDBC_CONNECTION_URL`: l'URL per la connessione a un'origine dati JDBC.
- `JDBC_ENFORCE_SSL`: una stringa booleana (true, false) che specifica se il protocollo Secure Sockets Layer (SSL) con nome host corrispondente viene applicato per la connessione JDBC al client. Il valore predefinito è false.
- `CUSTOM_JDBC_CERT`- Una posizione Amazon S3 che specifica il certificato principale del cliente. AWS Glue utilizza questo certificato radice per convalidare il certificato del cliente durante la connessione al database dei clienti. AWS Glue gestisce solo certificati X.509. Il certificato fornito deve essere codificato DER e fornito in formato PEM con codifica Base64.
- `SKIP_CUSTOM_JDBC_CERT_VALIDATION`- Per impostazione predefinita, questo è. `false` AWS Glue convalida l'algoritmo Signature e Subject Public Key Algorithm per il certificato del cliente. Gli unici algoritmi consentiti per l'algoritmo di firma sono SHA256withRSA, SHA384withRSA o SHA512withRSA. Per l'algoritmo della chiave pubblica oggetto, la lunghezza della chiave deve essere almeno 2048. Puoi impostare il valore di questa proprietà su `true` per ignorare la convalida di AWS Glue del certificato del cliente.

- `CUSTOM_JDBC_CERT_STRING`- Una stringa di certificato JDBC personalizzata che viene utilizzata per la corrispondenza del dominio o della corrispondenza dei nomi distinti per prevenire un attacco. man-in-the-middle Nel database Oracle, viene utilizzato come `SSL_SERVER_CERT_DN`; in Microsoft SQL Server, viene utilizzato come `hostNameInCertificate`.
- `CONNECTION_URL`: l'URL per la connessione a un'origine dati generale (non JDBC).
- `SECRET_ID`: l'ID segreto utilizzato per il Secret Manager delle credenziali.
- `CONNECTOR_URL`: l'URL del connettore per una connessione MARKETPLACE o CUSTOM.
- `CONNECTOR_TYPE`: il tipo di connettore per una connessione MARKETPLACE o CUSTOM.
- `CONNECTOR_CLASS_NAME`: il nome di classe del connettore per una connessione MARKETPLACE o CUSTOM.
- `KAFKA_BOOTSTRAP_SERVERS`: un elenco separato da virgole di coppie host e porte che sono gli indirizzi dei broker Apache Kafka in un cluster Kafka a cui un client Kafka si connetterà e si avvierà.
- `KAFKA_SSL_ENABLED`: indica se abilitare o disabilitare SSL su una connessione Apache Kafka. Il valore di default è "true".
- `KAFKA_CUSTOM_CERT`: l'URL Amazon S3 per il file di certificazione CA privato (formato .pem). L'impostazione predefinita è una stringa vuota.
- `KAFKA_SKIP_CUSTOM_CERT_VALIDATION`- Se saltare o meno la convalida del file di certificato CA. AWS Glue esegue la convalida per tre algoritmi: SHA256WithRSA, SHA384WithRSA e SHA512WithRSA. Il valore predefinito è "false".
- `KAFKA_CLIENT_KEYSTORE`: la posizione Amazon S3 del file keystore del client per l'autenticazione lato client Kafka (facoltativo).
- `KAFKA_CLIENT_KEYSTORE_PASSWORD`: la password per accedere al keystore fornito (facoltativo).
- `KAFKA_CLIENT_KEY_PASSWORD`: un keystore può essere costituito da più chiavi, quindi questa è la password per accedere alla chiave client da utilizzare con la chiave lato server Kafka (facoltativo).
- `ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD`- La versione crittografata della password del keystore del client Kafka (se l'utente ha selezionato l'impostazione di crittografia delle password). AWS Glue

- `ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD`- La versione crittografata della password della chiave del client Kafka (se l'utente ha selezionato l'impostazione di crittografia delle password). AWS Glue
- `KAFKA_SASL_MECHANISM`-`"SCRAM-SHA-512"`,, o`"GSSAPI"`. `"AWS_MSK_IAM"` `"PLAIN"` Queste sono i due meccanismi SASL supportati.
- `KAFKA_SASL_PLAIN_USERNAME`- Un nome utente in testo semplice utilizzato per l'autenticazione con il meccanismo «PLAIN».
- `KAFKA_SASL_PLAIN_PASSWORD`- Una password in testo semplice utilizzata per l'autenticazione con il meccanismo «PLAIN».
- `ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD`- La versione crittografata della password Kafka SASL PLAIN (se l'utente ha selezionato l'impostazione di crittografia delle AWS Glue password).
- `KAFKA_SASL_SCRAM_USERNAME`: un nome utente in testo semplice utilizzato per autenticarsi con il meccanismo "SCRAM-SHA-512".
- `KAFKA_SASL_SCRAM_PASSWORD`: una password in testo semplice utilizzata per autenticarsi con il meccanismo "SCRAM-SHA-512".
- `ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD`- La versione crittografata della password Kafka SASL SCRAM (se l'utente ha selezionato l'impostazione di crittografia delle password). AWS Glue
- `KAFKA_SASL_SCRAM_SECRETS_ARN`- Il nome di risorsa Amazon di un segreto in AWS Secrets Manager.
- `KAFKA_SASL_GSSAPI_KEYTAB`: la posizione S3 di un file `keytab` Kerberos. Un keytab memorizza le chiavi a lungo termine per uno o più principali. Per ulteriori informazioni, consulta la Documentazione di MIT Kerberos: keytab.
- `KAFKA_SASL_GSSAPI_KRB5_CONF`: la posizione S3 di un file `krb5.conf` Kerberos. Un krb5.conf memorizza le informazioni di configurazione Kerberos, ad esempio la posizione del server KDC. Per ulteriori informazioni, consulta la Documentazione di MIT Kerberos: krb5.conf.
- `KAFKA_SASL_GSSAPI_SERVICE`: il nome del servizio Kerberos, come impostato con `sasl.kerberos.service.name` nella Configurazione Kafka.
- `KAFKA_SASL_GSSAPI_PRINCIPAL`- Il nome del principale Kerberos utilizzato da. AWS Glue Per ulteriori informazioni, consulta la Documentazione di Kafka: configurazione dei broker Kafka.

- `PhysicalConnectionRequirements`: un oggetto PhysicalConnectionRequirements.

  Una mappa di requisiti di connessione fisici, come un cloud privato virtuale (VPC) e `SecurityGroup`, necessari per effettuare questa connessione.

- `CreationTime`: timestamp.

  L'ora in cui è stata creata questa definizione di connessione.

- `LastUpdatedTime`: timestamp.

  L'ultima volta che è stata aggiornata questa definizione di connessione.

- `LastUpdatedBy`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'utente, gruppo o ruolo che ha aggiornato per ultimo questa definizione di connessione.

ConnectionInput struttura

Una struttura utilizzata per specificare una connessione da creare o aggiornare.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della connessione. La connessione non funzionerà come previsto senza un nome.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La descrizione della connessione.

- `ConnectionType`: obbligatorio: stringa UTF-8 (valori validi: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM`).

  Il tipo di connessione. Attualmente, sono supportati questi tipi:

  - `JDBC`: designa una connessione a un database tramite Java Database Connectivity (JDBC).

    `JDBC`Le connessioni utilizzano quanto segue ConnectionParameters.

    - Obbligatorio: tutti (`HOST`, `PORT`, `JDBC_ENGINE`) o `JDBC_CONNECTION_URL`.
    - Obbligatorio: tutti (`USERNAME`, `PASSWORD`) o `SECRET_ID`.
    - Facoltativo: `JDBC_ENFORCE_SSL`, `CUSTOM_JDBC_CERT`, `CUSTOM_JDBC_CERT_STRING`, `SKIP_CUSTOM_JDBC_CERT_VALIDATION`. Questi parametri vengono utilizzati per configurare SSL con JDBC.

  - `KAFKA`: indica una connessione a una piattaforma di streaming Apache Kafka.

KAFKALe connessioni utilizzano quanto segue ConnectionParameters.

- Obbligatorio: KAFKA_BOOTSTRAP_SERVERS.
- Facoltativo: KAFKA_SSL_ENABLED, KAFKA_CUSTOM_CERT, KAFKA_SKIP_CUSTOM_CERT_VALIDATION. Questi parametri vengono utilizzati per configurare SSL con KAFKA.
- Facoltativo: KAFKA_CLIENT_KEYSTORE, KAFKA_CLIENT_KEYSTORE_PASSWORD, KAFKA_CLIENT_KEY_PASSWORD, ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD, ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD. Questi parametri vengono utilizzati per impostare la configurazione del client TLS con SSL in KAFKA.
- Facoltativo: KAFKA_SASL_MECHANISM. Può essere specificato come SCRAM-SHA-512, GSSAPI o AWS_MSK_IAM.
- Facoltativo: KAFKA_SASL_SCRAM_USERNAME, KAFKA_SASL_SCRAM_PASSWORD, ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD. Questi parametri vengono utilizzati per configurare l'autenticazione SASL/SCRAM-SHA-512 con KAFKA.
- Facoltativo: KAFKA_SASL_GSSAPI_KEYTAB, KAFKA_SASL_GSSAPI_KRB5_CONF, KAFKA_SASL_GSSAPI_SERVICE, KAFKA_SASL_GSSAPI_PRINCIPAL. Questi parametri vengono utilizzati per configurare l'autenticazione SASL/GSSAPI con KAFKA.

- MONGODB: designa una connessione a un database di documenti MongoDB.

  MONGODBLe connessioni utilizzano quanto segue ConnectionParameters.

  - Obbligatorio: CONNECTION_URL.
  - Obbligatorio: tutti (USERNAME, PASSWORD) o SECRET_ID.

- NETWORK: designa una connessione di rete a un'origine dati all'interno di un ambiente Amazon Virtual Private Cloud (Amazon VPC).

  NETWORKLe connessioni non richiedono ConnectionParameters. Fornisci invece un PhysicalConnectionRequirements.

- MARKETPLACE- Utilizza le impostazioni di configurazione contenute in un connettore acquistato Marketplace AWS per leggere e scrivere su archivi dati che non sono supportati nativamente da AWS Glue.

  MARKETPLACELe connessioni utilizzano quanto segue ConnectionParameters.

  - Obbligatorio: CONNECTOR_TYPE, CONNECTOR_URL, CONNECTOR_CLASS_NAME, CONNECTION_URL.

- Obbligatorio per le connessioni JDBC CONNECTOR_TYPE: tutti (USERNAME, PASSWORD) o SECRET_ID.
- CUSTOM: utilizza le impostazioni di configurazione contenute in un connettore per leggere e scrivere in archivi dati non supportati nativamente da AWS Glue.

SFTP non è supportato.

Per ulteriori informazioni su come ConnectionProperties vengono utilizzate le opzioni opzionali per configurare le funzionalità in AWS Glue, consulta le proprietà di AWS Glue connessione.

Per ulteriori informazioni su come ConnectionProperties vengono utilizzate le funzionalità opzionali per configurare le funzionalità in AWS Glue Studio, consulta Utilizzo di connettori e connessioni.

- MatchCriteria: una matrice di stringhe UTF-8, non superiore a 10 stringhe.

  Un elenco di criteri che possono essere utilizzati nella selezione di questa connessione.

- ConnectionProperties: obbligatorio: una matrice di mappe di coppie chiave-valore, non superiore a 100 coppie.

  Ogni chiave è una stringa UTF-8 (valori validi: HOST | | PORT | USERNAME="USER_NAME" | PASSWORD | ENCRYPTED_PASSWORD | JDBC_DRIVER_JAR_URI | JDBC_DRIVER_CLASS_NAME | JDBC_ENGINE | JDBC_ENGINE_VERSION | CONFIG_FILES | INSTANCE_ID | | JDBC_CONNECTION_URL | JDBC_ENFORCE_SSL | CUSTOM_JDBC_CERT | SKIP_CUSTOM_JDBC_CERT_VALIDATION | CUSTOM_JDBC_CERT_STRING | CONNECTION_URL | KAFKA_BOOTSTRAP_SERVERS | KAFKA_SSL_ENABLED | | KAFKA_CUSTOM_CERT | KAFKA_SKIP_CUSTOM_CERT_VALIDATION | KAFKA_CLIENT_KEYSTORE | KAFKA_CLIENT_KEYSTORE_PASSWORD | KAFKA_CLIENT_KEY_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD | SECRET_ID | CONNECTOR_URL | CONNECTOR_TYPE | | CONNECTOR_CLASS_NAME | KAFKA_SASL_MECHANISM | KAFKA_SASL_PLAIN_USERNAME | KAFKA_SASL_PLAIN_PASSWORD | ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD KAFKA_SASL_SCRAM_USERNAME KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_SCRAM_SECRETS_ARN | ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD | | KAFKA_SASL_GSSAPI_KEYTAB KAFKA_SASL_GSSAPI_KRB5_CONF KAFKA_SASL_GSSAPI_SERVICE |KAFKA_SASL_GSSAPI_PRINCIPAL).

  Ogni valore è una stringa Value, lunga non più di 1024 byte.

Queste coppie chiave-valore definiscono i parametri per la connessione.

- `PhysicalConnectionRequirements`: un oggetto PhysicalConnectionRequirements.

  Una mappa di requisiti di connessione fisici, come un cloud privato virtuale (VPC) e `SecurityGroup`, necessari per effettuare questa connessione.

PhysicalConnectionRequirements struttura

Specifica i requisiti fisici per una connessione.

Campi

- `SubnetId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID della sottorete utilizzato dalla connessione.
- `SecurityGroupIdList`: una matrice di stringhe UTF-8, non superiore a 50 stringhe.

  L'elenco di ID del gruppo di sicurezza utilizzato dalla connessione.
- `AvailabilityZone`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  La zona di disponibilità della connessione. Il campo è ridondante perché la sottorete specificata implica la zona di disponibilità da utilizzare. Al momento il campo deve essere popolato, ma diventerà obsoleto in futuro.

GetConnectionsFilter struttura

Filtra le definizioni di connessione restituite dall'operazione API `GetConnections`.

Campi

- `MatchCriteria`: una matrice di stringhe UTF-8, non superiore a 10 stringhe.

  Una stringa di criteri che deve corrispondere ai criteri registrati nella definizione di connessione affinché venga restituita quella definizione di connessione.
- `ConnectionType`: stringa UTF-8 (valori validi: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM`).

Il tipo di connessioni da restituire. Attualmente, SFTP non è supportato.

Operazioni

- CreateConnection azione (Python: create_connection)
- DeleteConnection azione (Python: delete_connection)
- GetConnection azione (Python: get_connection)
- GetConnections azione (Python: get_connections)
- UpdateConnection azione (Python: update_connection)
- BatchDeleteConnection azione (Python: batch_delete_connection)

CreateConnection azione (Python: create_connection)

Crea una definizione di connessione nel catalogo dati.

Le connessioni utilizzate per creare risorse federate richiedono l'autorizzazione IAM `glue:PassConnection`.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui creare la connessione. Se non ne viene fornito nessuno, per impostazione predefinita viene utilizzato l'ID AWS dell'account.
- `ConnectionInput`: obbligatorio: un oggetto ConnectionInput.

  Un oggetto `ConnectionInput` che definisce la connessione da creare.
- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag assegnati alla connessione.

Risposta

- Nessun parametro di risposta.

Errori

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`

DeleteConnection azione (Python: delete_connection)

Elimina una connessione dal catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la connessione. Se non ne viene fornito nessuno, per impostazione predefinita viene utilizzato l'ID AWS dell'account.
- `ConnectionName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della connessione da eliminare.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `OperationTimeoutException`

GetConnection azione (Python: get_connection)

Recupera una definizione di connessione dal catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la connessione. Se non ne viene fornito nessuno, per impostazione predefinita viene utilizzato l'ID AWS dell'account.

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della definizione di connessione da recuperare.

- `HidePassword`: booleano.

  Consente di recuperare i metadati di connessione senza restituire la password. Ad esempio, la AWS Glue console utilizza questo flag per recuperare la connessione e non visualizza la password. Imposta questo parametro quando il chiamante potrebbe non avere l'autorizzazione per utilizzare la AWS KMS chiave per decrittografare la password, ma dispone dell'autorizzazione per accedere al resto delle proprietà della connessione.

Risposta

- `Connection`: un oggetto Connessione.

  La definizione di connessione richiesta.

Errori

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

GetConnections azione (Python: get_connections)

Recupera un elenco di definizioni di connessione dal catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiedono le connessioni. Se non ne viene fornito nessuno, per impostazione predefinita viene utilizzato l'ID AWS dell'account.

- `Filter`: un oggetto GetConnectionsFilter.

  Un filtro che controlla quali connessioni vengono restituite.

- `HidePassword`: booleano.

  Consente di recuperare i metadati di connessione senza restituire la password. Ad esempio, la AWS Glue console utilizza questo flag per recuperare la connessione e non visualizza la password. Imposta questo parametro quando il chiamante potrebbe non avere l'autorizzazione per utilizzare la AWS KMS chiave per decrittografare la password, ma dispone dell'autorizzazione per accedere al resto delle proprietà della connessione.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Il numero massimo di connessioni da restituire in una risposta.

Risposta

- `ConnectionList`: una matrice di oggetti Connessione.

  Un elenco di definizioni di connessione richieste.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se l'elenco di connessioni restituite non include l'ultima delle connessioni filtrate.

Errori

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

UpdateConnection azione (Python: update_connection)

Aggiorna una definizione di connessione nel catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la connessione. Se non ne viene fornito nessuno, per impostazione predefinita viene utilizzato l'ID AWS dell'account.
- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della definizione di connessione da aggiornare.
- `ConnectionInput`: obbligatorio: un oggetto ConnectionInput.

  Un oggetto `ConnectionInput` che ridefinisce la connessione in questione.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

BatchDeleteConnection azione (Python: batch_delete_connection)

Elimina un elenco di definizioni di connessione dal catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiedono le connessioni. Se non ne viene fornito nessuno, per impostazione predefinita viene utilizzato l'ID AWS dell'account.
- `ConnectionNameList`: obbligatorio: matrice di stringhe UTF-8, non superiore a 25 stringhe.

  Un elenco di nomi delle connessioni da eliminare.

Risposta

- `Succeeded`: una matrice di stringhe UTF-8.

  Un elenco di nomi delle definizioni di connessione la cui eliminazione è riuscita.
- `Errors`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è un oggetto ErrorDetail.

  Una mappa dei nomi delle connessioni la cui eliminazione non è riuscita per i dettagli di errore.

Errori

- `InternalServiceException`
- `OperationTimeoutException`

API della funzione definita dall'utente

L'API della funzione definita dall'utente descrive i tipi di dati e le operazioni AWS Glue utilizzate per lavorare con le funzioni.

Tipi di dati

- Struttura UserDefinedFunction
- Struttura UserDefinedFunctionInput

Struttura UserDefinedFunction

Rappresenta l'equivalente di una definizione di funzione Hive definita dall'utente (UDF).

Campi

- `FunctionName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della funzione .
- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo che contiene la funzione.
- `ClassName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  La classe Java che contiene il codice della funzione.
- `OwnerName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il proprietario della funzione.
- `OwnerType`: stringa UTF-8 (valori validi: `USER` | `ROLE` | `GROUP`).

  Il tipo di proprietario.
- `CreateTime`: timestamp.

  L'ora in cui è stata creata la funzione.
- `ResourceUris`: una matrice di oggetti ResourceUri, non superiore a 1000 strutture.

  Gli URI della risorsa per la funzione.
- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID del catalogo dati in cui si trova la funzione.

Struttura UserDefinedFunctionInput

Una struttura utilizzata per creare o aggiornare una funzione definita dall'utente.

Campi

- `FunctionName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della funzione .

- `ClassName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  La classe Java che contiene il codice della funzione.

- `OwnerName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il proprietario della funzione.

- `OwnerType`: stringa UTF-8 (valori validi: `USER` | `ROLE` | `GROUP`).

  Il tipo di proprietario.

- `ResourceUris`: una matrice di oggetti ResourceUri, non superiore a 1000 strutture.

  Gli URI della risorsa per la funzione.

Operazioni

- Operazione CreateUserDefinedFunction (Python: create_user_defined_function)
- Operazione UpdateUserDefinedFunction (Python: update_user_defined_function)
- Operazione DeleteUserDefinedFunction (Python: delete_user_defined_function)
- Operazione GetUserDefinedFunction (Python: get_user_defined_function)
- Operazione GetUserDefinedFunctions (Python: get_user_defined_functions)

Operazione CreateUserDefinedFunction (Python: create_user_defined_function)

Crea una nuova definizione di funzione nel catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui creare la funzione. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui creare la funzione.
- `FunctionInput`: obbligatorio: oggetto UserDefinedFunctionInput.

  Un oggetto `FunctionInput` che definisce la funzione da creare nel catalogo dati.

Risposta

- Nessun parametro di risposta.

Errori

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`

Operazione UpdateUserDefinedFunction (Python: update_user_defined_function)

Aggiorna una definizione di funzione esistente nel catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trova la funzione da aggiornare. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui si trova la funzione da aggiornare.
- `FunctionName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della funzione .
- `FunctionInput`: obbligatorio: oggetto UserDefinedFunctionInput.

  Un oggetto `FunctionInput` che ridefinisce la funzione nel catalogo dati.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione DeleteUserDefinedFunction (Python: delete_user_defined_function)

Elimina una definizione di funzione esistente dal catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trova la funzione da eliminare. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui si trova la funzione.
- `FunctionName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della definizione della funzione da eliminare.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Operazione GetUserDefinedFunction (Python: get_user_defined_function)

Richiama una definizione di funzione specificata dal catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trova la funzione da richiamare. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database del catalogo in cui si trova la funzione.

- `FunctionName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della funzione .

Risposta

- `UserDefinedFunction`: un oggetto UserDefinedFunction.

  La definizione di funzione richiesta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Operazione GetUserDefinedFunctions (Python: get_user_defined_functions)

Richiama definizioni di funzione multiple dal catalogo dati.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui si trovano le funzioni da recuperare. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.

- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome del database del catalogo in cui si trovano le funzioni. Se non ne viene fornito nessuno, verranno restituite le funzioni di tutti i database del catalogo.

- `Pattern`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Una stringa di modello nome-funzione facoltativa che filtra le definizioni di funzione restituite.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

- `MaxResults` – Numero (intero), non inferiore a 1 o superiore a 100.

  Il numero massimo di funzioni da restituire in una risposta.

Risposta

- `UserDefinedFunctions`: una matrice di oggetti UserDefinedFunction.

  Un elenco di definizioni di funzione richieste.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se l'elenco di funzioni restituite non include l'ultima funzione richiesta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`

Importazione di un catalogo Athena a AWS Glue

L'API di migrazione descrive i tipi di dati e le operazioni di AWS Glue relative alla migrazione di un catalogo dati Athena su AWS Glue.

Tipi di dati

- Struttura CatalogImportStatus

Struttura CatalogImportStatus

Una struttura che contiene informazioni sullo stato della migrazione.

Campi

- `ImportCompleted`: booleano.

  `True` se la migrazione è stata completata, in caso contrario `False`.

- `ImportTime`: timestamp.

  L'ora in cui la migrazione è stata avviata.

- `ImportedBy`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della persona che ha avviato la migrazione.

Operazioni

- Azione ImportCatalogToGlue (Python: import_catalog_to_glue)
- Azione GetCatalogImportStatus (Python: get_catalog_import_status)

Azione ImportCatalogToGlue (Python: import_catalog_to_glue)

Importa un catalogo dati Amazon Athena esistente in AWS Glue.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo da importare. Al momento, dovrebbe essere l'ID dell'account AWS.

Risposta

- Nessun parametro di risposta.

Errori

- `InternalServiceException`
- `OperationTimeoutException`

Azione GetCatalogImportStatus (Python: get_catalog_import_status)

Recupera lo stato di un'operazione di migrazione.

Richiesta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo da migrare. Al momento, dovrebbe essere l'ID dell'account AWS.

Risposta

- `ImportStatus`: un oggetto CatalogImportStatus.

  Lo stato della migrazione del catalogo specificata.

Errori

- `InternalServiceException`
- `OperationTimeoutException`

API dell'ottimizzatore di tabelle

L'API di ottimizzazione delle tabelle descrive l' AWS Glue API per abilitare la compattazione per migliorare le prestazioni di lettura.

Tipi di dati

- TableOptimizer struttura
- TableOptimizerConfiguration struttura
- TableOptimizerRun struttura
- RunMetrics struttura
- BatchGetTableOptimizerEntry struttura
- BatchTableOptimizer struttura
- BatchGetTableOptimizerError struttura

TableOptimizer struttura

Contiene dettagli su un ottimizzatore associato a una tabella.

Campi

- `type`: stringa UTF-8 (valori validi: `compaction="COMPACTION"`).

  Il tipo di ottimizzatore di tabelle. Attualmente, l'unico valore valido è `compaction`.
- `configuration`: un oggetto TableOptimizerConfiguration.

  Un oggetto `TableOptimizerConfiguration` specificato durante la creazione o l'aggiornamento di un ottimizzatore di tabelle.
- `lastRun`: un oggetto TableOptimizerRun.

  Un oggetto `TableOptimizerRun` che rappresenta l'ultima esecuzione dell'ottimizzatore di tabelle.

TableOptimizerConfiguration struttura

Contiene dettagli sulla configurazione di un ottimizzatore di tabelle. Questa configurazione viene passata quando si crea o si aggiorna un ottimizzatore di tabelle.

Campi

- `roleArn`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Single-line string pattern.

Un ruolo passato dal chiamante che autorizza il servizio ad aggiornare le risorse associate all'ottimizzatore per suo conto.

- `enabled`: booleano.

  Se l'ottimizzazione delle tabelle è abilitata.

TableOptimizerRun struttura

Contiene i dettagli per l'esecuzione di un ottimizzatore di tabelle.

Campi

- `eventType`: stringa UTF-8 (valori validi: `starting="STARTING"` | `completed="COMPLETED"` | `failed="FAILED"` | `in_progress="IN_PROGRESS"`).

  Un tipo di evento che rappresenta lo stato dell'esecuzione dell'ottimizzatore di tabella.

- `startTimestamp`: timestamp.

  Rappresenta il timestamp di epoca in cui è stato avviato il processo di compattazione all'interno di Lake Formation.

- `endTimestamp`: timestamp.

  Rappresenta il timestamp di epoca in cui è terminato il processo di compattazione.

- `metrics`: un oggetto RunMetrics.

  Un oggetto `RunMetrics` contenente i parametri per l'esecuzione dell'ottimizzatore.

- `error`: stringa UTF-8.

  Un errore che si è verificato durante l'esecuzione dell'ottimizzatore.

RunMetrics struttura

Parametri per l'esecuzione dell'ottimizzatore.

Campi

- `NumberOfBytesCompacted`: stringa UTF-8.

Il numero di byte rimossi dall'esecuzione del processo di compattazione.

- `NumberOfFilesCompacted`: stringa UTF-8.

  Il numero di file rimossi dall'esecuzione del processo di compattazione.

- `NumberOfDpus`: stringa UTF-8.

  Il numero di ore DPU utilizzate dal processo.

- `JobDurationInHour`: stringa UTF-8.

  La durata del processo in ore.

BatchGetTableOptimizerEntry struttura

Rappresenta un ottimizzatore di tabella da recuperare durante l'operazione `BatchGetTableOptimizer`.

Campi

- `catalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo della tabella.

- `databaseName`: stringa UTF-8, almeno 1 byte di lunghezza.

  Il nome del database nel catalogo in cui risiede la tabella.

- `tableName`: stringa UTF-8, almeno 1 byte di lunghezza.

  Nome della tabella.

- `type`: stringa UTF-8 (valori validi: `compaction="COMPACTION"`).

  Il tipo di ottimizzatore di tabelle.

BatchTableOptimizer struttura

Contiene i dettagli per uno degli ottimizzatori di tabella restituiti dall'operazione `BatchGetTableOptimizer`.

Campi

- `catalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo della tabella.
- `databaseName`: stringa UTF-8, almeno 1 byte di lunghezza.

  Il nome del database nel catalogo in cui risiede la tabella.
- `tableName`: stringa UTF-8, almeno 1 byte di lunghezza.

  Nome della tabella.
- `tableOptimizer`: un oggetto TableOptimizer.

  Un oggetto `TableOptimizer` che contiene i dettagli sulla configurazione e l'ultima esecuzione di un ottimizzatore di tabella.

BatchGetTableOptimizerError struttura

Contiene dettagli su uno degli errori nell'elenco degli errori restituito dall'operazione `BatchGetTableOptimizer`.

Campi

- `error`: un oggetto ErrorDetail.

  Un oggetto `ErrorDetail` contenente i dettagli del codice e del messaggio di errore.
- `catalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo della tabella.
- `databaseName`: stringa UTF-8, almeno 1 byte di lunghezza.

  Il nome del database nel catalogo in cui risiede la tabella.
- `tableName`: stringa UTF-8, almeno 1 byte di lunghezza.

  Nome della tabella.
- `type`: stringa UTF-8 (valori validi: `compaction="COMPACTION"`).

Il tipo di ottimizzatore di tabelle.

Operazioni

- GetTableOptimizer azione (Python: get_table_optimizer)
- BatchGetTableOptimizer azione (Python: batch_get_table_optimizer)
- ListTableOptimizerRuns azione (Python: list_table_optimizer_runs)
- CreateTableOptimizer azione (Python: create_table_optimizer)
- DeleteTableOptimizer azione (Python: delete_table_optimizer)
- UpdateTableOptimizer azione (Python: update_table_optimizer)

GetTableOptimizer azione (Python: get_table_optimizer)

Restituisce la configurazione di tutti gli ottimizzatori associati a una tabella specificata.

Richiesta

- `CatalogId` - Obbligatorio:: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern

  L'ID del catalogo della tabella.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database nel catalogo in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.
- `Type`: obbligatorio: stringa UTF-8 (valori validi: `compaction="COMPACTION"`).

  Il tipo di ottimizzatore di tabelle.

Risposta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo della tabella.

- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database nel catalogo in cui risiede la tabella.

- `TableName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.

- `TableOptimizer`: un oggetto TableOptimizer.

  L'ottimizzatore associato alla tabella specificata.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`
- `ThrottlingException`

BatchGetTableOptimizer azione (Python: batch_get_table_optimizer)

Restituisce la configurazione per gli ottimizzatori di tabella specificati.

Richiesta

- `Entries`: obbligatorio: una matrice di oggetti BatchGetTableOptimizerEntry.

  Un elenco di oggetti `BatchGetTableOptimizerEntry` che specificano gli ottimizzatori di tabella da recuperare.

Risposta

- `TableOptimizers`: una matrice di oggetti BatchTableOptimizer.

  Elenco di oggetti `BatchTableOptimizer`.
- `Failures`: una matrice di oggetti BatchGetTableOptimizerError.

  Un elenco di errori derivanti dall'operazione.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`
- `ThrottlingException`

ListTableOptimizerRuns azione (Python: list_table_optimizer_runs)

Elenca la cronologia delle esecuzioni dell'ottimizzatore precedenti per una tabella specifica.

Richiesta

- `CatalogId` - Obbligatorio:: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern

  L'ID del catalogo della tabella.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database nel catalogo in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.
- `Type`: obbligatorio: stringa UTF-8 (valori validi: `compaction="COMPACTION"`).

  Il tipo di ottimizzatore di tabelle. Attualmente, l'unico valore valido è `compaction`.

- `MaxResults`: numero (intero).

  Il numero massimo di esecuzioni dell'ottimizzatore da restituire per ogni chiamata.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `CatalogId`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo della tabella.
- `DatabaseName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database nel catalogo in cui risiede la tabella.
- `TableName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.
- `NextToken`: stringa UTF-8.

  Un token di continuazione per impaginare l'elenco restituito di esecuzioni dell'ottimizzatore, restituite se il segmento corrente dell'elenco non è l'ultimo.
- `TableOptimizerRuns`: una matrice di oggetti TableOptimizerRun.

  Un elenco delle esecuzioni di ottimizzazione associate a una tabella.

Errori

- `EntityNotFoundException`
- `AccessDeniedException`
- `InvalidInputException`
- `ValidationException`
- `InternalServiceException`
- `ThrottlingException`

CreateTableOptimizer azione (Python: create_table_optimizer)

Crea un nuovo ottimizzatore di tabella per una funzione specifica. `compaction` è l'unico tipo di ottimizzatore attualmente supportato.

Richiesta

- `CatalogId` - Obbligatorio:: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern

  L'ID del catalogo della tabella.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database nel catalogo in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.
- `Type`: obbligatorio: stringa UTF-8 (valori validi: `compaction="COMPACTION"`).

  Il tipo di ottimizzatore di tabelle. Attualmente, l'unico valore valido è `compaction`.
- `TableOptimizerConfiguration`: obbligatorio: un oggetto TableOptimizerConfiguration.

  Un oggetto `TableOptimizerConfiguration` che rappresenta la configurazione dell'ottimizzatore di tabelle.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `ValidationException`
- `InvalidInputException`
- `AccessDeniedException`

- `AlreadyExistsException`
- `InternalServiceException`
- `ThrottlingException`

DeleteTableOptimizer azione (Python: delete_table_optimizer)

Elimina un ottimizzatore e tutti i metadati associati per una tabella. L'ottimizzazione non verrà più eseguita sulla tabella.

Richiesta

- `CatalogId` - Obbligatorio:: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern

  L'ID del catalogo della tabella.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database nel catalogo in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.
- `Type`: obbligatorio: stringa UTF-8 (valori validi: `compaction="COMPACTION"`).

  Il tipo di ottimizzatore di tabelle.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

- `ThrottlingException`

UpdateTableOptimizer azione (Python: update_table_optimizer)

Aggiorna la configurazione per un ottimizzatore di tabelle esistente.

Richiesta

- `CatalogId` - Obbligatorio:: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern

  L'ID del catalogo della tabella.
- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database nel catalogo in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.
- `Type`: obbligatorio: stringa UTF-8 (valori validi: `compaction="COMPACTION"`).

  Il tipo di ottimizzatore di tabelle. Attualmente, l'unico valore valido è `compaction`.
- `TableOptimizerConfiguration`: obbligatorio: un oggetto TableOptimizerConfiguration.

  Un oggetto `TableOptimizerConfiguration` che rappresenta la configurazione dell'ottimizzatore di tabelle.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`

- `ValidationException`
- `InternalServiceException`
- `ThrottlingException`

API crawler e classificatori

L'API crawler e classificatori descrive i tipi di dati di crawler e classificatori di AWS Glue e include l'API per la creazione, l'eliminazione, l'aggiornamento e l'elencazione dei crawler o dei classificatori.

Argomenti

- API classificatore
- API crawler
- API delle statistiche delle colonne
- API del pianificatore del crawler

API classificatore

L'API Classifier descrive i tipi di dati classificatori di AWS Glue e include l'API per la creazione, l'eliminazione, l'aggiornamento e l'elenco dei classificatori.

Tipi di dati

- Struttura classificatore
- Struttura GrokClassifier
- Struttura XMLClassifier
- Struttura JsonClassifier
- Struttura CsvClassifier
- Struttura CreateGrokClassifierRequest
- Struttura UpdateGrokClassifierRequest
- Struttura CreateXMLClassifierRequest
- Struttura UpdateXMLClassifierRequest
- Struttura CreateJsonClassifierRequest
- Struttura UpdateJsonClassifierRequest

- Struttura CreateCsvClassifierRequest
- Struttura UpdateCsvClassifierRequest

Struttura classificatore

I classificatori vengono attivati durante un'attività di crawling. Un classificatore verifica se un determinato file è in un formato che è in grado di gestire. In questo caso il classificatore crea uno schema nel formato di un oggetto `StructType` che corrisponde a quel formato di dati.

È possibile usare i classificatori standard forniti da AWS Glue Glue oppure scrivere i propri classificatori per suddividere al meglio le origini dati e specificare gli schemi appropriati per l'utilizzo. Un classificatore può essere di tipo `grok`, `XML`, `JSON` o `CSV` personalizzato come specificato in uno dei campi dell'oggetto `Classifier`.

Campi

- `GrokClassifier`: un oggetto GrokClassifier.

  Un classificatore che utilizza `grok`.
- `XMLClassifier`: un oggetto XMLClassifier.

  Classificatore per contenuto XML.
- `JsonClassifier`: un oggetto JsonClassifier.

  Classificatore per contenuto JSON.
- `CsvClassifier`: un oggetto CsvClassifier.

  Un classificatore per i valori separati da virgole (CSV).

Struttura GrokClassifier

Un classificatore che utilizza i pattern `grok`.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del classificatore.

- `Classification`: obbligatorio: stringa UTF-8.

  Identificatore del formato di dati corrisposto dal classificatore, ad esempio log Twitter, JSON, Omniture e così via.

- `CreationTime`: timestamp.

  L'ultima volta in cui è stato registrato il classificatore.

- `LastUpdated`: timestamp.

  L'ultima volta in cui è stato aggiornato il classificatore.

- `Version`: numero (lungo).

  La versione del classificatore.

- `GrokPattern`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 2048 byte di lunghezza, corrispondente a A Logstash Grok string pattern.

  Il pattern grok applicato a un datastore da questo classificatore. Per ulteriori informazioni, consulta i pattern integrati in Scrittura di classificatori personalizzati.

- `CustomPatterns`: stringa UTF-8, non superiore a 16000 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Pattern grok personalizzati opzionali definiti da questo classificatore. Per ulteriori informazioni, consulta i pattern personalizzati in Scrittura di classificatori personalizzati.

Struttura XMLClassifier

Classificatore per contenuto `XML`.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del classificatore.

- `Classification`. Obbligatorio: stringa UTF-8.

  Identificatore del formato di dati corrisposto dal classificatore.

- `CreationTime`: timestamp.

L'ultima volta in cui è stato registrato il classificatore.

- `LastUpdated`: timestamp.

  L'ultima volta in cui è stato aggiornato il classificatore.

- `Version`: numero (lungo).

  La versione del classificatore.

- `RowTag`: stringa UTF-8.

  Il tag XML che designa l'elemento contenente ogni record in un documento XML da analizzare. Non è in grado di identificare un elemento con chiusura automatica (chiuso da `/>`). Un elemento riga vuota contenente solo attributi può essere analizzato fintantoché termina con un tag di chiusura (ad esempio, `<row item_a="A" item_b="B"></row>` è corretto, mentre `<row item_a="A" item_b="B" />` non lo è).

Struttura JsonClassifier

Classificatore per contenuto `JSON`.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del classificatore.

- `CreationTime`: timestamp.

  L'ultima volta in cui è stato registrato il classificatore.

- `LastUpdated`: timestamp.

  L'ultima volta in cui è stato aggiornato il classificatore.

- `Version`: numero (lungo).

  La versione del classificatore.

- `JsonPath`. Obbligatorio: stringa UTF-8.

Una stringa `JsonPath` che definisce i dati JSON per il classificatore da classificare. AWS Glue supporta un sottoinsieme di JsonPath, come descritto in Scrittura di classificatori personalizzati JsonPath.

Struttura CsvClassifier

Classificatore per contenuto `CSV` personalizzato.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del classificatore.
- `CreationTime`: timestamp.

  L'ultima volta in cui è stato registrato il classificatore.
- `LastUpdated`: timestamp.

  L'ultima volta in cui è stato aggiornato il classificatore.
- `Version`: numero (lungo).

  La versione del classificatore.
- `Delimiter`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 1 byte di lunghezza, corrispondente a Custom string pattern #10.

  Un simbolo personalizzato per indicare il separatore di ogni voce di colonna nella riga.
- `QuoteSymbol`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 1 byte di lunghezza, corrispondente a Custom string pattern #10.

  Un simbolo personalizzato per indicare la combinazione dei contenuti in un singolo valore di colonna. Deve essere diverso dal delimitatore di colonna.
- `ContainsHeader`: stringa UTF-8 (valori validi: `UNKNOWN` | `PRESENT` | `ABSENT`).

  Indica se il file CSV contiene un'intestazione.
- `Header`: una matrice di stringhe UTF-8.

  Un elenco di stringhe che rappresenta i nomi delle colonne.

- `DisableValueTrimming`: booleano.

  Specifica di non tagliare i valori prima di individuare il tipo di valori di colonna. Il valore predefinito è `true`.

- `AllowSingleColumn`: booleano.

  Abilita l'elaborazione dei file che contengono una sola colonna.

- `CustomDatatypeConfigured`: booleano.

  Consente di configurare il tipo di dati personalizzato.

- `CustomDatatypes`: una matrice di stringhe UTF-8.

  Un elenco di tipi di dati personalizzati tra cui "BINARY", "BOOLEAN", "DATE", "DECIMAL", "DOUBLE", "FLOAT", "INT", "LONG", "SHORT", "STRING", "TIMESTAMP".

- `Serde`: stringa UTF-8 (valori validi: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Imposta il SerDe per l'elaborazione del CSV nel classificatore, che verrà applicato in Catalogo dati. I valori validi sono `OpenCSVSerDe`, `LazySimpleSerDe` e `None`. È possibile specificare il valore `None` quando si desidera che il crawler esegua il rilevamento.

Struttura CreateGrokClassifierRequest

Specifica un classificatore `grok` per `CreateClassifier`.

Campi

- `Classification`. Obbligatorio: stringa UTF-8.

  Identificatore del formato di dati corrisposto dal classificatore, ad esempio log Twitter, JSON, Omniture, Amazon CloudWatch Logs e così via.

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del nuovo classificatore.

- `GrokPattern`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 2048 byte di lunghezza, corrispondente a A Logstash Grok string pattern.

  Il pattern grok utilizzato da questo classificatore.

- `CustomPatterns`: stringa UTF-8, non superiore a 16000 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Pattern grok personalizzati opzionali utilizzati da questo classificatore.

Struttura UpdateGrokClassifierRequest

Specifica un classificatore grok da aggiornare quando viene passato a `UpdateClassifier`.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della `GrokClassifier`.
- `Classification`: stringa UTF-8.

  Identificatore del formato di dati corrisposto dal classificatore, ad esempio log Twitter, JSON, Omniture, Amazon CloudWatch Logs e così via.
- `GrokPattern`: stringa UTF-8, non inferiore a 1 o superiore a 2048 byte di lunghezza, corrispondente a A Logstash Grok string pattern.

  Il pattern grok utilizzato da questo classificatore.
- `CustomPatterns`: stringa UTF-8, non superiore a 16000 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Pattern grok personalizzati opzionali utilizzati da questo classificatore.

Struttura CreateXMLClassifierRequest

Specifica un classificatore XML per `CreateClassifier`.

Campi

- `Classification`. Obbligatorio: stringa UTF-8.

  Identificatore del formato di dati corrisposto dal classificatore.
- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome del classificatore.

- `RowTag`: stringa UTF-8.

  Il tag XML che designa l'elemento contenente ogni record in un documento XML da analizzare. Non è in grado di identificare un elemento con chiusura automatica (chiuso da `/>`). Un elemento riga vuota contenente solo attributi può essere analizzato fintantoché termina con un tag di chiusura (ad esempio, `<row item_a="A" item_b="B"></row>` è corretto, mentre `<row item_a="A" item_b="B" />` non lo è).

Struttura UpdateXMLClassifierRequest

Specifica un classificatore XML da aggiornare.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del classificatore.

- `Classification`: stringa UTF-8.

  Identificatore del formato di dati corrisposto dal classificatore.

- `RowTag`: stringa UTF-8.

  Il tag XML che designa l'elemento contenente ogni record in un documento XML da analizzare. Non è in grado di identificare un elemento con chiusura automatica (chiuso da `/>`). Un elemento riga vuota contenente solo attributi può essere analizzato fintantoché termina con un tag di chiusura (ad esempio, `<row item_a="A" item_b="B"></row>` è corretto, mentre `<row item_a="A" item_b="B" />` non lo è).

Struttura CreateJsonClassifierRequest

Specifica un classificatore JSON per `CreateClassifier`.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome del classificatore.

- `JsonPath`. Obbligatorio: stringa UTF-8.

  Una stringa `JsonPath` che definisce i dati JSON per il classificatore da classificare. AWS Glue supporta un sottoinsieme di JsonPath, come descritto in Scrittura di classificatori personalizzati JsonPath.

Struttura UpdateJsonClassifierRequest

Specifica un classificatore JSON da aggiornare.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del classificatore.

- `JsonPath`: stringa UTF-8.

  Una stringa `JsonPath` che definisce i dati JSON per il classificatore da classificare. AWS Glue supporta un sottoinsieme di JsonPath, come descritto in Scrittura di classificatori personalizzati JsonPath.

Struttura CreateCsvClassifierRequest

Specifica un classificatore CSV personalizzato per `CreateClassifier`.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del classificatore.

- `Delimiter`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 1 byte di lunghezza, corrispondente a Custom string pattern #10.

  Un simbolo personalizzato per indicare il separatore di ogni voce di colonna nella riga.

- `QuoteSymbol`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 1 byte di lunghezza, corrispondente a Custom string pattern #10.

  Un simbolo personalizzato per indicare la combinazione dei contenuti in un singolo valore di colonna. Deve essere diverso dal delimitatore di colonna.

- `ContainsHeader`: stringa UTF-8 (valori validi: `UNKNOWN` | `PRESENT` | `ABSENT`).

  Indica se il file CSV contiene un'intestazione.

- `Header`: una matrice di stringhe UTF-8.

  Un elenco di stringhe che rappresenta i nomi delle colonne.

- `DisableValueTrimming`: booleano.

  Specifica di non tagliare i valori prima di individuare il tipo di valori di colonna. Il valore di default è true.

- `AllowSingleColumn`: booleano.

  Abilita l'elaborazione dei file che contengono una sola colonna.

- `CustomDatatypeConfigured`: booleano.

  Consente di configurare tipi di dati personalizzati.

- `CustomDatatypes`: una matrice di stringhe UTF-8.

  Crea un elenco di tipi di dati personalizzati supportati.

- `Serde`: stringa UTF-8 (valori validi: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Imposta il SerDe per l'elaborazione del CSV nel classificatore, che verrà applicato in Catalogo dati. I valori validi sono `OpenCSVSerDe`, `LazySimpleSerDe` e `None`. È possibile specificare il valore `None` quando si desidera che il crawler esegua il rilevamento.

Struttura UpdateCsvClassifierRequest

Specifica un classificatore CSV personalizzato da aggiornare.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome del classificatore.

- `Delimiter`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 1 byte di lunghezza, corrispondente a Custom string pattern #10.

  Un simbolo personalizzato per indicare il separatore di ogni voce di colonna nella riga.

- `QuoteSymbol`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 1 byte di lunghezza, corrispondente a Custom string pattern #10.

  Un simbolo personalizzato per indicare la combinazione dei contenuti in un singolo valore di colonna. Deve essere diverso dal delimitatore di colonna.

- `ContainsHeader`: stringa UTF-8 (valori validi: `UNKNOWN` | `PRESENT` | `ABSENT`).

  Indica se il file CSV contiene un'intestazione.

- `Header`: una matrice di stringhe UTF-8.

  Un elenco di stringhe che rappresenta i nomi delle colonne.

- `DisableValueTrimming`: booleano.

  Specifica di non tagliare i valori prima di individuare il tipo di valori di colonna. Il valore di default è true.

- `AllowSingleColumn`: booleano.

  Abilita l'elaborazione dei file che contengono una sola colonna.

- `CustomDatatypeConfigured`: booleano.

  Specifica la configurazione di tipi di dati personalizzati.

- `CustomDatatypes`: una matrice di stringhe UTF-8.

  Specifica un elenco di tipi di dati personalizzati supportati.

- `Serde`: stringa UTF-8 (valori validi: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Imposta il SerDe per l'elaborazione del CSV nel classificatore, che verrà applicato in Catalogo dati. I valori validi sono `OpenCSVSerDe`, `LazySimpleSerDe` e `None`. È possibile specificare il valore `None` quando si desidera che il crawler esegua il rilevamento.

Operazioni

- Operazione CreateClassifier (Python: create_classifier)
- Operazione DeleteClassifier (Python: delete_classifier)
- Operazione GetClassifier (Python: get_classifier)
- Operazione GetClassifiers (Python: get_classifiers)
- Operazione UpdateClassifier (Python: update_classifier)

Operazione CreateClassifier (Python: create_classifier)

Crea un classificatore nell'account utente. L'operazione può essere un `GrokClassifier`, un `XMLClassifier`, un `JsonClassifier` o un `CsvClassifier` a seconda del campo in cui è presente la richiesta.

Richiesta

- `GrokClassifier`: un oggetto CreateGrokClassifierRequest.

  Oggetto `GrokClassifier` che specifica il classificatore da creare.
- `XMLClassifier`: un oggetto CreateXMLClassifierRequest.

  Oggetto `XMLClassifier` che specifica il classificatore da creare.
- `JsonClassifier`: un oggetto CreateJsonClassifierRequest.

  Oggetto `JsonClassifier` che specifica il classificatore da creare.
- `CsvClassifier`: un oggetto CreateCsvClassifierRequest.

  Oggetto `CsvClassifier` che specifica il classificatore da creare.

Risposta

- Nessun parametro di risposta.

Errori

- `AlreadyExistsException`
- `InvalidInputException`

- `OperationTimeoutException`

Operazione DeleteClassifier (Python: delete_classifier)

Rimuove un classificatore dal catalogo dati.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del classificatore da rimuovere.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `OperationTimeoutException`

Operazione GetClassifier (Python: get_classifier)

Recupera un classificatore per nome.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del classificatore da recuperare.

Risposta

- `Classifier`: un oggetto Classificatore.

  Il classificatore richiesto.

Errori

- `EntityNotFoundException`
- `OperationTimeoutException`

Operazione GetClassifiers (Python: get_classifiers)

Visualizza l'elenco di tutti gli oggetti classificatore nel catalogo dati.

Richiesta

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Dimensione dell'elenco da restituire (opzionale).
- `NextToken`: stringa UTF-8.

  Token di continuazione opzionale.

Risposta

- `Classifiers`: una matrice di oggetti Classificatore.

  L'elenco richiesto di tutti gli oggetti classificatore.
- `NextToken`: stringa UTF-8.

  Token di continuazione.

Errori

- `OperationTimeoutException`

Operazione UpdateClassifier (Python: update_classifier)

Modifica un classificatore esistente (`GrokClassifier`, `XMLClassifier`, `JsonClassifier` o `CsvClassifier` a seconda del campo in cui è presente).

Richiesta

- `GrokClassifier`: un oggetto UpdateGrokClassifierRequest.

Oggetto `GrokClassifier` con i campi aggiornati.

- `XMLClassifier`: un oggetto UpdateXMLClassifierRequest.

  Oggetto `XMLClassifier` con i campi aggiornati.
- `JsonClassifier`: un oggetto UpdateJsonClassifierRequest.

  Oggetto `JsonClassifier` con i campi aggiornati.
- `CsvClassifier`: un oggetto UpdateCsvClassifierRequest.

  Oggetto `CsvClassifier` con i campi aggiornati.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `VersionMismatchException`
- `EntityNotFoundException`
- `OperationTimeoutException`

API crawler

L'API Crawler descrive i tipi di dati dei AWS Glue crawler, oltre all'API per la creazione, l'eliminazione, l'aggiornamento e l'elenco dei crawler.

Tipi di dati

- Struttura dei crawler
- Struttura della pianificazione
- CrawlerTargets struttura
- Struttura S3Target
- Struttura S3 DeltaCatalogTarget
- Struttura S3 DeltaDirectTarget

- JdbcTarget struttura
- Struttura MongoDBTarget
- Struttura DynamoDBTarget
- DeltaTarget struttura
- IcebergTarget struttura
- HudiTarget struttura
- CatalogTarget struttura
- CrawlerMetrics struttura
- CrawlerHistory struttura
- CrawlsFilter struttura
- SchemaChangePolicy struttura
- LastCrawlInfo struttura
- RecrawlPolicy struttura
- LineageConfiguration struttura
- LakeFormationConfiguration struttura

Struttura dei crawler

Specifica un programma crawler che esamina un'origine dati e usa i classificatori per cercare di determinarne lo schema. Se l'esito è positivo, il crawler registra i metadati riguardanti l'origine dati in AWS Glue Data Catalog.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del crawler.
- `Role`: stringa UTF-8.

  Il nome della risorsa Amazon (ARN) di un ruolo IAM utilizzato per accedere alle risorse del cliente, ad esempio i dati di Amazon Simple Storage Service (Amazon S3).
- `Targets`: un oggetto CrawlerTargets.

  Raccolta di destinazioni da sottoporre al crawling.

- `DatabaseName`: stringa UTF-8.

  Il nome del database di catalogo in cui viene archiviato l'output del crawler.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Descrizione del crawler.

- `Classifiers`: una matrice di stringhe UTF-8.

  Elenco di stringhe UTF-8 che specificano i classificatori personalizzati associati al crawler.

- `RecrawlPolicy`: un oggetto RecrawlPolicy.

  Una policy che specifica se eseguire nuovamente il crawling dell'intero set di dati o solo delle cartelle aggiunte dall'ultima esecuzione del crawler.

- `SchemaChangePolicy`: un oggetto SchemaChangePolicy.

  La policy che specifica i comportamenti di aggiornamento ed eliminazione per il crawler.

- `LineageConfiguration`: un oggetto LineageConfiguration.

  Una configurazione che specifica se la derivazione dei dati è abilitata per il crawler.

- `State`: stringa UTF-8 (valori validi: `READY` | `RUNNING` | `STOPPING`).

  Indica se il crawler è in esecuzione o se una sessione è in sospeso.

- `TablePrefix`: stringa UTF-8, non superiore a 128 byte di lunghezza.

  Il prefisso aggiunto ai nomi delle tabelle create.

- `Schedule`: un oggetto Pianificazione.

  Per i crawler pianificati, la pianificazione dell'esecuzione del crawler.

- `CrawlElapsedTime`: numero (lungo).

  Se il crawler è in esecuzione, contiene il tempo totale trascorso dall'inizio dell'ultimo crawling.

- `CreationTime`: timestamp.

  L'ora di creazione del crawler.

- `LastUpdated`: timestamp.

  L'ora dell'ultimo aggiornamento del crawler.

- `LastCrawl`: un oggetto LastCrawlInfo.

  Lo stato dell'ultimo crawling ed eventualmente le informazioni sull'errore, se presente.

- `Version`: numero (lungo).

  La versione del crawler.

- `Configuration`: stringa UTF-8.

  Le informazioni di configurazione del crawler. Questa stringa JSON con versione consente agli utenti di specificare gli aspetti del comportamento di un crawler. Per ulteriori informazioni, consulta la pagina Impostazione delle opzioni di configurazione del crawler.

- `CrawlerSecurityConfiguration`: stringa UTF-8, non superiore a 128 byte di lunghezza.

  Il nome della struttura `SecurityConfiguration` che questo crawler deve utilizzare.

- `LakeFormationConfiguration`: un oggetto LakeFormationConfiguration.

  Speciifica se il crawler deve utilizzare le credenziali per il crawler anziché AWS Lake Formation le credenziali del ruolo IAM.

Struttura della pianificazione

Oggetto di pianificazione che utilizza una dichiarazione `cron` per pianificare un evento.

Campi

- `ScheduleExpression`: stringa UTF-8.

  Espressione `cron` usata per specificare la pianificazione (consulta Pianificazioni basate sul tempo per processi e crawler. Ad esempio, per eseguire un processo ogni giorno alle 12:15 UTC, devi specificare: `cron(15 12 * * ? *)`.

- `State`: stringa UTF-8 (valori validi: `SCHEDULED` | `NOT_SCHEDULED` | `TRANSITIONING`).

  Lo stato della pianificazione.

CrawlerTargets struttura

Specifica gli archivi dati da sottoporre al crawling.

Campi

- `S3Targets`: una matrice di oggetti S3Target.

  Specifica le destinazioni di Amazon Simple Storage Service (Amazon S3).

- `JdbcTargets`: una matrice di oggetti JdbcTarget.

  Specifica le destinazioni JDBC.

- `MongoDBTargets`: una matrice di oggetti MongoDBTarget.

  Specifica destinazioni Amazon DocumentDB o MongoDB.

- `DynamoDBTargets`: una matrice di oggetti DynamoDBTarget.

  Specifica le destinazioni di Amazon DynamoDB.

- `CatalogTargets`: una matrice di oggetti CatalogTarget.

  Speciifica gli AWS Glue Data Catalog obiettivi.

- `DeltaTargets`: una matrice di oggetti DeltaTarget.

  Specifica le destinazioni dell'archivio dati Delta.

- `IcebergTargets`: una matrice di oggetti IcebergTarget.

  Specifica le destinazioni del datastore Apache Iceberg.

- `HudiTargets`: una matrice di oggetti HudiTarget.

  Specifica le destinazioni del datastore Apache Hudi.

Struttura S3Target

Specifica un archivio dati in Amazon Simple Storage Service (Amazon S3).

Campi

- `Path`: stringa UTF-8.

  Il percorso della destinazione Amazon S3.

- `Exclusions`: una matrice di stringhe UTF-8.

Elenco di modelli globali utilizzati per l'esclusione dal crawling. Per ulteriori informazioni, consulta la sezione relativa alla catalogazione delle tabelle con un crawler.

- `ConnectionName`: stringa UTF-8.

  Il nome di una connessione che consente a un processo o a un crawler di accedere ai dati in Amazon S3 all'interno di un ambiente Amazon Virtual Private Cloud (Amazon VPC).

- `SampleSize`: numero (intero).

  Imposta il numero di file in ogni cartella foglia da sottoporre al crawling durante il crawling di file di esempio in un set di dati. Se non è impostato, tutti i file vengono sottoposti al crawling. Un valore valido è un numero intero compreso tra 1 e 249.

- `EventQueueArn`: stringa UTF-8.

  Un ARN Amazon SQS valido. Ad esempio, `arn:aws:sqs:region:account:sqs`.

- `DlqEventQueueArn`: stringa UTF-8.

  Un ARN Amazon SQS di messaggi non recapitabili valido. Ad esempio, `arn:aws:sqs:region:account:deadLetterQueue`.

Struttura S3 DeltaCatalogTarget

Speciifica una destinazione che scrive su un'origine dati Delta Lake nel AWS Glue Data Catalog.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.
- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica le opzioni di connessione aggiuntive per il connettore.
- `SchemaChangePolicy`: un oggetto CatalogSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

Struttura S3 DeltaDirectTarget

Speciifica una destinazione che scrive su un'origine dati Delta Lake in. Amazon S3

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.
- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.
- `Path`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il percorso Amazon S3 dell'origine dati Delta Lake su cui scrivere.
- `Compression`: obbligatorio: stringa UTF-8 (valori validi: `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  Specifica il modo in cui i dati sono compressi. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip"`).
- `Format`: obbligatorio: stringa UTF-8 (valori validi: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

Specifica il formato di output dei dati per la destinazione.

- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica le opzioni di connessione aggiuntive per il connettore.

- `SchemaChangePolicy`: un oggetto DirectSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

JdbcTarget struttura

Specifica un archivio dati JDBC da sottoporre al crawling.

Campi

- `ConnectionName`: stringa UTF-8.

  Il nome della connessione da usare per connettersi alla destinazione JDBC.

- `Path`: stringa UTF-8.

  Il percorso della destinazione JDBC.

- `Exclusions`: una matrice di stringhe UTF-8.

  Elenco di modelli globali utilizzati per l'esclusione dal crawling. Per ulteriori informazioni, consulta la sezione relativa alla catalogazione delle tabelle con un crawler.

- `EnableAdditionalMetadata`: una matrice di stringhe UTF-8.

  Specifica un valore di `RAWTYPES` o `COMMENTS` per abilitare metadati aggiuntivi nelle risposte della tabella. `RAWTYPES` fornisce il tipo di dati a livello nativo. `COMMENTS` fornisce commenti associati a una colonna o a una tabella del database.

  Se non hai bisogno di metadati aggiuntivi, lascia il campo vuoto.

Struttura MongoDBTarget

Specifica un archivio dati Amazon DocumentDB o MongoDB da sottoporre al crawling.

Campi

- `ConnectionName`: stringa UTF-8.

  Il nome della connessione da usare per connettersi alla destinazione Amazon DocumentDB o MongoDB.

- `Path`: stringa UTF-8.

  Il percorso della destinazione Amazon DocumentDB o MongoDB (database/raccolta).

- `ScanAll`: booleano.

  Indica se eseguire la scansione di tutti i registri o campionare le righe della tabella. La scansione di tutti i registri può richiedere molto tempo quando la tabella non è una tabella di throughput elevato.

  Un valore di `true` significa analizzare tutti i registri, mentre un valore di `false` significa campionare i registri. Se non viene specificato alcun valore, il valore di default è `true`.

Struttura DynamoDBTarget

Specifica una tabella Amazon DynamoDB per eseguire il crawling.

Campi

- `Path`: stringa UTF-8.

  Nome della tabella DynamoDB di cui eseguire il crawling.

- `scanAll`: booleano.

  Indica se eseguire la scansione di tutti i registri o campionare le righe della tabella. La scansione di tutti i registri può richiedere molto tempo quando la tabella non è una tabella di throughput elevato.

  Un valore di `true` significa analizzare tutti i registri, mentre un valore di `false` significa campionare i registri. Se non viene specificato alcun valore, il valore di default è `true`.

- `scanRate`: numero (doppio).

La percentuale di unità di capacità di lettura configurate da utilizzare dal AWS Glue crawler. L'unità di capacità di lettura è un termine definito da DynamoDB ed è un valore numerico che funge da limitatore di velocità per il numero di letture che possono essere eseguite su tale tabella al secondo.

I valori validi sono null o un valore compreso tra 0,1 e 1,5. Un valore null viene utilizzato quando l'utente non fornisce un valore e il valore predefinito è 0,5 dell'unità di capacità di lettura massima configurata (per le tabelle con provisioning) o 0,25 dell'unità di capacità di lettura massima configurata (per le tabelle che utilizzano la modalità on demand).

DeltaTarget struttura

Specifica un archivio dati Delta per eseguire la scansione di una o più tabelle Delta.

Campi

- `DeltaTables`: una matrice di stringhe UTF-8.

  Un elenco dei percorsi Amazon S3 alle tabelle Delta.
- `ConnectionName`: stringa UTF-8.

  Il nome della connessione da usare per connettersi alla destinazione della tabella Delta.
- `WriteManifest`: booleano.

  Specifica se scrivere i file manifest sul percorso della tabella Delta.
- `CreateNativeDeltaTable`: booleano.

  Specifica se il crawler creerà tabelle native per consentire l'integrazione con i motori di query che supportano l'interrogazione diretta del log delle transazioni Delta.

IcebergTarget struttura

Specifica un'origine dati Apache Iceberg in cui sono archiviate le tabelle Iceberg all'interno di Amazon S3.

Campi

- `Paths`: una matrice di stringhe UTF-8.

Uno o più Amazon S3 percorsi che contengono le cartelle di metadati Iceberg come. `s3://bucket/prefix`

- `ConnectionName`: stringa UTF-8.

  Il nome della connessione da utilizzare per connettersi alla destinazione Iceberg.

- `Exclusions`: una matrice di stringhe UTF-8.

  Elenco di modelli globali utilizzati per l'esclusione dal crawling. Per ulteriori informazioni, consulta la sezione relativa alla catalogazione delle tabelle con un crawler.

- `MaximumTraversalDepth`: numero (intero).

  La profondità massima dei Amazon S3 percorsi che il crawler può attraversare per scoprire la cartella di metadati Iceberg nel percorso. Amazon S3 Viene utilizzata per limitare il tempo di esecuzione del crawler.

HudiTarget struttura

Specifica un'origine dati Apache Hudi.

Campi

- `Paths`: una matrice di stringhe UTF-8.

  Una serie di stringhe di Amazon S3 posizione per Hudi, ognuna delle quali indica la cartella principale in cui risiedono i file di metadati per una tabella Hudi. La cartella Hudi può trovarsi in una cartella figlia della principale.

  Il crawler scansionerà tutte le cartelle al di sotto del percorso di una cartella Hudi.

- `ConnectionName`: stringa UTF-8.

  Il nome della connessione da utilizzare per connettersi alla destinazione Hudi. Se i tuoi file Hudi sono archiviati in bucket che richiedono l'autorizzazione VPC, puoi impostarne le proprietà di connessione qui.

- `Exclusions`: una matrice di stringhe UTF-8.

  Elenco di modelli globali utilizzati per l'esclusione dal crawling. Per ulteriori informazioni, consulta la sezione relativa alla catalogazione delle tabelle con un crawler.

- `MaximumTraversalDepth`: numero (intero).

La profondità massima dei Amazon S3 percorsi che il crawler può attraversare per scoprire la cartella di metadati Hudi nel percorso. Amazon S3 Viene utilizzata per limitare il tempo di esecuzione del crawler.

CatalogTarget struttura

Speciifica un AWS Glue Data Catalog obiettivo.

Campi

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database da sincronizzare.
- `Tables`: obbligatorio: una matrice di stringhe UTF-8, almeno 1 stringa.

  Elenco di tabelle da sincronizzare.
- `ConnectionName`: stringa UTF-8.

  Il nome della connessione per una tabella di Catalogo dati supportata da Amazon S3 come destinazione del crawling quando si utilizza un tipo di connessione `Catalog` abbinato a un tipo di connessione `NETWORK`.
- `EventQueueArn`: stringa UTF-8.

  Un ARN Amazon SQS valido. Ad esempio, `arn:aws:sqs:region:account:sqs`.
- `DlqEventQueueArn`: stringa UTF-8.

  Un ARN Amazon SQS di messaggi non recapitabili valido. Ad esempio, `arn:aws:sqs:region:account:deadLetterQueue`.

CrawlerMetrics struttura

I parametri di un determinato crawler.

Campi

- `CrawlerName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome del crawler.

- `TimeLeftSeconds`: numero (doppio), non superiore a None (Nessuno).

  Il tempo stimato che rimane per completare un crawling in esecuzione.

- `StillEstimating`: booleano.

  True se il crawler sta ancora valutando il tempo necessario per completare la sessione.

- `LastRuntimeSeconds`: numero (doppio), non superiore a None (Nessuno).

  La durata in secondi della sessione più recente del crawler.

- `MedianRuntimeSeconds`: numero (doppio), non superiore a None (Nessuno).

  La durata media in secondi delle sessioni del crawler.

- `TablesCreated`: numero (intero), non superiore a Nessuno.

  Il numero di tabelle create dal crawler.

- `TablesUpdated`: numero (intero), non superiore a Nessuno.

  Il numero di tabelle aggiornate dal crawler.

- `TablesDeleted`: numero (intero), non superiore a Nessuno.

  Il numero di tabelle eliminate dal crawler.

CrawlerHistory struttura

Contiene le informazioni per l'esecuzione di un crawler.

Campi

- `CrawlId`: stringa UTF-8.

  Un identificatore UUID per ogni crawling.

- `State`: stringa UTF-8 (valori validi: `RUNNING` | `COMPLETED` | `FAILED` | `STOPPED`).

  Lo stato del crawling.

- `StartTime`: timestamp.

  La data e l'ora in cui è stata avviata l'esecuzione del crawler.

- `EndTime`: timestamp.

  La data e l'ora in cui è terminato il crawling.

- `Summary`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un riepilogo dell'esecuzione per il crawling in JSON. Contiene le tabelle e le partizioni del catalogo che sono state aggiunte, aggiornate o eliminate.

- `ErrorMessage`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Se si è verificato un errore, il messaggio di errore è associato al crawling.

- `LogGroup`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Log group string pattern.

  Il gruppo di log associato al crawler.

- `LogStream`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Log-stream string pattern.

  Il flusso di log associato all'esecuzione del crawler.

- `MessagePrefix`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il prefisso per un CloudWatch messaggio relativo a questo crawl.

- `DPUHour`: numero (doppio), non superiore a None (Nessuno).

  Il numero di unità di elaborazione dati (DPU) utilizzate in ore per il crawling.

CrawlsFilter struttura

Un elenco di campi, comparatori e valori che puoi utilizzare per filtrare le esecuzioni del crawler per un crawler specificato.

Campi

- `FieldName`: stringa UTF-8 (valori validi: `CRAWL_ID` | `STATE` | `START_TIME` | `END_TIME` | `DPU_HOUR`).

Una chiave utilizzata per filtrare le esecuzioni del crawler per un crawler specificato. I valori validi per ciascuno dei nomi di campo sono:

- `CRAWL_ID`: una stringa che rappresenta l'identificatore UUID per un crawling.
- `STATE`: una stringa che rappresenta lo stato del crawling.
- `START_TIME` e `END_TIME`: il timestamp epoch in millisecondi.
- `DPU_HOUR`: il numero di unità di elaborazione dati (DPU) utilizzate in ore per il crawling.

- `FilterOperator`: stringa UTF-8 (valori validi: `GT` | `GE` | `LT` | `LE` | `EQ` | `NE`).

  Un comparatore definito che opera sul valore. Gli operatori disponibili sono:

  - `GT`: maggiore di.
  - `GE`: maggiore o uguale a.
  - `LT`: minore di.
  - `LE`: minore o uguale a.
  - `EQ`: uguale a.
  - `NE`: non uguale a.
- `FieldValue`: stringa UTF-8.

  Il valore fornito per il confronto nel campo del crawling.

SchemaChangePolicy struttura

Una policy che specifica i comportamenti di aggiornamento ed eliminazione per il crawler.

Campi

- `UpdateBehavior`: stringa UTF-8 (valori validi: `LOG` | `UPDATE_IN_DATABASE`).

  Il comportamento di aggiornamento quando il crawler riscontra una variazione dello schema.
- `DeleteBehavior`: stringa UTF-8 (valori validi: `LOG` | `DELETE_FROM_DATABASE` | `DEPRECATE_IN_DATABASE`).

  Il comportamento di eliminazione quando il crawler riscontra un oggetto eliminato.

LastCrawlInfo struttura

Informazioni sullo stato e sull'errore relative al crawling più recente.

Campi

- `Status`: stringa UTF-8 (valori validi: `SUCCEEDED` | `CANCELLED` | `FAILED`).

  Stato dell'ultimo crawling.

- `ErrorMessage`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Le informazioni sull'errore dell'ultimo crawling, se presente.

- `LogGroup`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Log group string pattern.

  Il gruppo di log per l'ultimo crawling.

- `LogStream`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Log-stream string pattern.

  Il flusso di log per l'ultimo crawling.

- `MessagePrefix`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il prefisso per un messaggio sul crawling.

- `StartTime`: timestamp.

  L'ora di inizio del crawling.

RecrawlPolicy struttura

Quando si esegue il crawling di un'origine dati Amazon S3 dopo il completamento del primo crawling, specifica se eseguire nuovamente il crawling dell'intero set di dati o solo delle cartelle aggiunte dopo l'ultima esecuzione del crawler. Per ulteriori informazioni, consulta Crawling incrementali in AWS Glue nella guida per sviluppatori.

Campi

- `RecrawlBehavior`: stringa UTF-8 (valori validi: `CRAWL_EVERYTHING` | `CRAWL_NEW_FOLDERS_ONLY` | `CRAWL_EVENT_MODE`).

  Specifica se eseguire nuovamente il crawling dell'intero set di dati o solo delle cartelle aggiunte dall'ultima esecuzione del crawler.

  Un valore di `CRAWL_EVERYTHING` specifica nuovamente il crawling dell'intero set di dati.

  Un valore di `CRAWL_NEW_FOLDERS_ONLY` specifica il crawling solo delle cartelle che sono state aggiunte dopo l'ultima esecuzione del crawler.

  Un valore di `CRAWL_EVENT_MODE` specifica il crawling solo delle modifiche identificate dagli eventi Amazon S3.

LineageConfiguration struttura

Specifica le impostazioni di configurazione della derivazione dei dati per il crawler.

Campi

- `CrawlerLineageSettings`: stringa UTF-8 (valori validi: `ENABLE` | `DISABLE`).

  Specifica se la derivazione dei dati è abilitata per il crawler. I valori validi sono:

  - ENABLE: abilita la derivazione dei dati per il crawler
  - DISABLE: disabilita la derivazione dei dati per il crawler

LakeFormationConfiguration struttura

Specifica le impostazioni AWS Lake Formation di configurazione per il crawler.

Campi

- `UseLakeFormationCredentials`: booleano.

  Speciifica se utilizzare le AWS Lake Formation credenziali per il crawler anziché le credenziali del ruolo IAM.

- `AccountId`: stringa UTF-8, non superiore a 12 byte di lunghezza.

Obbligatorio per i crawling tra più account. Per il crawling degli stessi account dei dati di destinazione, può essere lasciato come null.

Operazioni

- CreateCrawler azione (Python: create_crawler)
- DeleteCrawler azione (Python: delete_crawler)
- GetCrawler azione (Python: get_crawler)
- GetCrawlers azione (Python: get_crawlers)
- GetCrawlerMetrics azione (Python: get_crawler_metrics)
- UpdateCrawler azione (Python: update_crawler)
- StartCrawler azione (Python: start_crawler)
- StopCrawler azione (Python: stop_crawler)
- BatchGetCrawlers azione (Python: batch_get_crawlers)
- ListCrawlers azione (Python: list_crawlers)
- ListCrawls azione (Python: list_crawls)

CreateCrawler azione (Python: create_crawler)

Crea un nuovo crawler con destinazioni, ruolo, configurazione specifici e pianificazione opzionale. Deve essere specificata almeno una destinazione di crawling nel campo `s3Targets`, nel campo, `jdbcTargets` o nel campo `DynamoDBTargets`.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del nuovo crawler.

- `Role`. Obbligatorio: stringa UTF-8.

  Il ruolo IAM o il nome della risorsa Amazon (ARN) di un ruolo IAM utilizzato dal nuovo crawler per accedere alle risorse dei clienti.

- `DatabaseName`: stringa UTF-8.

Il AWS Glue database in cui vengono scritti i risultati, ad esempio:. `arn:aws:daylight:us-east-1::database/sometable/*`

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Descrizione del nuovo crawler.

- `Targets`: obbligatorio: un oggetto CrawlerTargets.

  Elenco della raccolta di destinazioni da sottoporre al crawling.

- `Schedule`: stringa UTF-8.

  Espressione `cron` usata per specificare la pianificazione (consulta Pianificazioni basate sul tempo per processi e crawler. Ad esempio, per eseguire un processo ogni giorno alle 12:15 UTC, devi specificare: `cron(15 12 * * ? *)`.

- `Classifiers`: una matrice di stringhe UTF-8.

  Elenco di classificatori personalizzati registrati dall'utente. Per impostazione predefinita, tutti i classificatori integrati sono inclusi in un crawling, ma i classificatori personalizzati sovrascrivono sempre i classificatori predefiniti per una determinata classificazione.

- `TablePrefix`: stringa UTF-8, non superiore a 128 byte di lunghezza.

  Il prefisso di tabella utilizzato per le tabelle di catalogo create.

- `SchemaChangePolicy`: un oggetto SchemaChangePolicy.

  Policy per il comportamento di aggiornamento ed eliminazione del crawler.

- `RecrawlPolicy`: un oggetto RecrawlPolicy.

  Una policy che specifica se eseguire nuovamente il crawling dell'intero set di dati o solo delle cartelle aggiunte dall'ultima esecuzione del crawler.

- `LineageConfiguration`: un oggetto LineageConfiguration.

  Specifica le impostazioni di configurazione della derivazione dei dati per il crawler.

- `LakeFormationConfiguration`: un oggetto LakeFormationConfiguration.

  Specifica le impostazioni AWS Lake Formation di configurazione per il crawler.

- `Configuration`: stringa UTF-8.

Le informazioni di configurazione del crawler. Questa stringa JSON con versione consente agli utenti di specificare gli aspetti del comportamento di un crawler. Per ulteriori informazioni, consulta la pagina Impostazione delle opzioni di configurazione del crawler.

- `CrawlerSecurityConfiguration`: stringa UTF-8, non superiore a 128 byte di lunghezza.

  Il nome della struttura `SecurityConfiguration` che questo crawler deve utilizzare.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag da usare con questa richiesta crawler. Puoi usare i tag per limitare l'accesso al crawler. Per ulteriori informazioni sui tag in AWS Glue, consulta AWS Tags in AWS Glue nella guida per sviluppatori.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

DeleteCrawler azione (Python: delete_crawler)

Rimuove un crawler specificato da, a meno che lo stato del crawler non lo sia AWS Glue Data Catalog. `RUNNING`

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Nome del crawler da rimuovere.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `CrawlerRunningException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

GetCrawler azione (Python: get_crawler)

Recupera i metadati per un determinato crawler.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del crawler per cui recuperare i metadati.

Risposta

- `Crawler`: un oggetto Crawler.

  I metadati per il crawler specificato.

Errori

- `EntityNotFoundException`
- `OperationTimeoutException`

GetCrawlers azione (Python: get_crawlers)

Recupera i metadati per tutti i crawler definiti nell'account del cliente.

Richiesta

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Il numero di crawler da restituire per ciascuna chiamata.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.

Risposta

- `Crawlers`: una matrice di oggetti Crawler.

  Elenco di metadati di crawler.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se l'elenco restituito non ha raggiunto la fine delle voci definite in questo account del cliente.

Errori

- `OperationTimeoutException`

GetCrawlerMetrics azione (Python: get_crawler_metrics)

Recupera i parametri sul crawler specificato.

Richiesta

- `CrawlerNameList`: una matrice di stringhe UTF-8, non superiore a 100.

  Elenco di nomi di crawler su cui recuperare i parametri.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima di un elenco da restituire.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `CrawlerMetricsList`: una matrice di oggetti CrawlerMetrics.

  Elenco di parametri per il crawler specificato.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se l'elenco restituito non contiene l'ultimo parametro disponibile.

Errori

- `OperationTimeoutException`

UpdateCrawler azione (Python: update_crawler)

Aggiorna un crawler. Se un crawler è in esecuzione, è necessario arrestarlo utilizzando `StopCrawler` prima dell'aggiornamento.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del nuovo crawler.
- `Role`: stringa UTF-8.

  Il ruolo IAM o il nome della risorsa Amazon (ARN) di un ruolo IAM utilizzato dal nuovo crawler per accedere alle risorse dei clienti.
- `DatabaseName`: stringa UTF-8.

  Il AWS Glue database in cui sono archiviati i risultati, ad esempio:. `arn:aws:daylight:us-east-1::database/sometable/*`
- `Description`: stringa UTF-8, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

Descrizione del nuovo crawler.

- `Targets`: un oggetto CrawlerTargets.

  Elenco di destinazioni da sottoporre al crawling.

- `Schedule`: stringa UTF-8.

  Espressione `cron` usata per specificare la pianificazione (consulta Pianificazioni basate sul tempo per processi e crawler. Ad esempio, per eseguire un processo ogni giorno alle 12:15 UTC, devi specificare: `cron(15 12 * * ? *)`.

- `Classifiers`: una matrice di stringhe UTF-8.

  Elenco di classificatori personalizzati registrati dall'utente. Per impostazione predefinita, tutti i classificatori integrati sono inclusi in un crawling, ma i classificatori personalizzati sovrascrivono sempre i classificatori predefiniti per una determinata classificazione.

- `TablePrefix`: stringa UTF-8, non superiore a 128 byte di lunghezza.

  Il prefisso di tabella utilizzato per le tabelle di catalogo create.

- `SchemaChangePolicy`: un oggetto SchemaChangePolicy.

  Policy per il comportamento di aggiornamento ed eliminazione del crawler.

- `RecrawlPolicy`: un oggetto RecrawlPolicy.

  Una policy che specifica se eseguire nuovamente il crawling dell'intero set di dati o solo delle cartelle aggiunte dall'ultima esecuzione del crawler.

- `LineageConfiguration`: un oggetto LineageConfiguration.

  Specifica le impostazioni di configurazione della derivazione dei dati per il crawler.

- `LakeFormationConfiguration`: un oggetto LakeFormationConfiguration.

  Specifica le impostazioni AWS Lake Formation di configurazione per il crawler.

- `Configuration`: stringa UTF-8.

  Le informazioni di configurazione del crawler. Questa stringa JSON con versione consente agli utenti di specificare gli aspetti del comportamento di un crawler. Per ulteriori informazioni, consulta la pagina Impostazione delle opzioni di configurazione del crawler.

- `CrawlerSecurityConfiguration`: stringa UTF-8, non superiore a 128 byte di lunghezza.

Il nome della struttura `SecurityConfiguration` che questo crawler deve utilizzare.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `VersionMismatchException`
- `EntityNotFoundException`
- `CrawlerRunningException`
- `OperationTimeoutException`

StartCrawler azione (Python: start_crawler)

Avvia un crawling utilizzando il crawler specificato, indipendentemente dalla pianificazione. Se il crawler è già in esecuzione, restituisce un. CrawlerRunningException

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del crawler da avviare.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `CrawlerRunningException`
- `OperationTimeoutException`

StopCrawler azione (Python: stop_crawler)

Se il crawler specificato è in esecuzione, arresta il crawling.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del crawler da arrestare.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `CrawlerNotRunningException`
- `CrawlerStoppingException`
- `OperationTimeoutException`

BatchGetCrawlers azione (Python: batch_get_crawlers)

Restituisce un elenco di metadati di risorse per un elenco di nomi di crawler. Dopo aver chiamato l'operazione `ListCrawlers`, puoi chiamare questa operazione per accedere ai dati a cui sono state concesse le autorizzazioni. Questa operazione supporta tutte le autorizzazioni IAM, tra cui le condizioni di autorizzazione che utilizzano i tag.

Richiesta

- `CrawlerNames`. Obbligatorio: una serie di stringhe UTF-8, non superiore a 100 stringhe.

  L'elenco dei nomi di crawler che potrebbero essere i nomi restituiti dall'operazione `ListCrawlers`.

Risposta

- `Crawlers`: una matrice di oggetti Crawler.

Un elenco di definizioni di crawler.

- `CrawlersNotFound`: una matrice di stringhe UTF-8, non superiore a 100.

  Un elenco di nomi di crawler non trovati.

Errori

- `InvalidInputException`
- `OperationTimeoutException`

ListCrawlers azione (Python: list_crawlers)

Recupera i nomi di tutte le risorse del crawler in questo AWS account o delle risorse con il tag specificato. Questa operazione consente di vedere quali risorse sono disponibili nel proprio account e i relativi nomi.

L'operazione accetta il campo facoltativo `Tags` che si può utilizzare come filtro per la risposta in modo che le risorse con tag possano essere recuperate come gruppo. Se si sceglie di utilizzare il filtro dei tag, potranno essere recuperate solo le risorse con tag.

Richiesta

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima di un elenco da restituire.

- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Specifica che vengono restituite solo le risorse con tag.

Risposta

- `CrawlerNames`: una matrice di stringhe UTF-8, non superiore a 100.

  I nomi di tutti i crawler nell'account oppure i crawler con i tag specificati.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se l'elenco restituito non contiene l'ultimo parametro disponibile.

Errori

- `OperationTimeoutException`

ListCrawls azione (Python: list_crawls)

Restituisce tutti i crawling di un determinato crawler. Restituisce solo i crawling che si sono verificati dalla data di avvio della funzione cronologia del crawler e conserva solo fino a 12 mesi di crawling. I crawling più vecchi non verranno restituiti.

È possibile utilizzare questa API per:

- Recuperare tutti i crawling di un determinato crawler.
- Recuperare tutti i crawling di un crawler specificato entro un conteggio limitato.
- Recuperare tutti i crawling di un crawler specificato in un intervallo di tempo specifico.
- Recuperare tutti i crawling di un crawler specificato con uno stato particolare, un ID di crawling o un valore orario della DPU.

Richiesta

- `CrawlerName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del crawler di cui vuoi recuperare le esecuzioni.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Numero massimo di risultati da restituire. Il valore predefinito è 20 e il valore massimo è 100.
- `Filters`: una matrice di oggetti CrawlsFilter.

Filtra i crawling in base ai criteri specificati in un elenco di oggetti `CrawlsFilter`.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `Crawls`: una matrice di oggetti CrawlerHistory.

  Un elenco di oggetti `CrawlerHistory` che rappresentano le esecuzioni del crawling che soddisfano i criteri specificati.

- `NextToken`: stringa UTF-8.

  Un token di continuazione per impaginare l'elenco restituito di token, restituiti se il segmento corrente dell'elenco non è l'ultimo.

Errori

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`

API delle statistiche delle colonne

L'API delle statistiche delle colonne descrive le API AWS Glue per la restituzione di statistiche delle colonne di una tabella.

Tipi di dati

- Struttura ColumnStatisticsTaskRun
- Struttura ColumnStatisticsTaskRunningException
- Struttura ColumnStatisticsTaskNotRunningException
- Struttura ColumnStatisticsTaskStoppingException

Struttura ColumnStatisticsTaskRun

L'oggetto che mostra i dettagli dell'esecuzione delle statistiche delle colonne.

Campi

- `CustomerId`: stringa UTF-8, non superiore a 12 byte di lunghezza.

  ID dell'account AWS.
- `ColumnStatisticsTaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore per l'esecuzione dell'attività delle statistiche delle colonne specifica.
- `DatabaseName`: stringa UTF-8.

  Il database in cui risiede la tabella.
- `TableName`: stringa UTF-8.

  Il nome della tabella per cui vengono generate le statistiche delle colonne.
- `ColumnNameList`: una matrice di stringhe UTF-8.

  Un elenco dei nomi delle colonne. Se non viene fornito, per impostazione predefinita verranno utilizzati tutti i nomi delle colonne della tabella.
- `CatalogID`: stringa ID catalogo, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `Role`: stringa UTF-8.

  Il ruolo IAM che assume il servizio per generare statistiche.
- `SampleSize`: numero (doppio), non superiore a 100.

  La percentuale di righe utilizzate per generare statistiche. Se non viene fornita, per generare statistiche verrà utilizzata l'intera tabella.
- `SecurityConfiguration`: stringa UTF-8, non superiore a 128 byte di lunghezza.

  Nome della configurazione di sicurezza utilizzata per crittografare i log di CloudWatch per l'esecuzione dell'attività delle statistiche delle colonne.

- `NumberOfWorkers`: numero (intero), almeno 1.

  Il numero di worker utilizzati per generare statistiche delle colonne. Il processo è preconfigurato per scalare automaticamente fino a 25 istanze.
- `WorkerType`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il tipo di worker utilizzati per generare statistiche. Il valore predefinito è `g.1x`.
- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `SUCCEEDED` | `FAILED` | `STOPPED`).

  Lo stato dell'esecuzione dell'attività.
- `CreationTime`: timestamp.

  L'ora di creazione di questa attività.
- `LastUpdated`: timestamp.

  Il momento dell'ultima modifica di questa attività.
- `StartTime`: timestamp.

  L'orario di inizio dell'attività.
- `EndTime`: timestamp.

  L'orario di fine dell'attività.
- `ErrorMessage`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Il messaggio di errore per il processo.
- `DPUSeconds`: numero (doppio), non superiore a None (Nessuno).

  L'utilizzo della DPU calcolato in secondi per tutti i worker con scalabilità automatica.

Struttura ColumnStatisticsTaskRunningException

Un'eccezione generata quando si cerca di avviare un altro processo durante l'esecuzione di un processo di generazione di statistiche delle colonne.

Campi

- `Message`: stringa UTF-8.

Messaggio che descrive il problema.

Struttura ColumnStatisticsTaskNotRunningException

Un'eccezione generata quando si tenta di interrompere l'esecuzione di un'attività quando non è in esecuzione alcuna attività.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura ColumnStatisticsTaskStoppingException

Un'eccezione generata quando si tenta di interrompere l'esecuzione di un'attività.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Operazioni

- Operazione StartColumnStatisticsTaskRun (Python: start_column_statistics_task_run)
- Operazione GetColumnStatisticsTaskRun (Python: get_column_statistics_task_run)
- Operazione GetColumnStatisticsTaskRuns (Python: get_column_statistics_task_runs)
- Operazione ListColumnStatisticsTaskRuns (Python: list_column_statistics_task_runs)
- Operazione StopColumnStatisticsTaskRun (Python: stop_column_statistics_task_run)

Operazione StartColumnStatisticsTaskRun (Python: start_column_statistics_task_run)

Avvia l'esecuzione di un'attività di statistica delle colonne, per una tabella e delle colonne specificate.

Richiesta

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del database in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della tabella per generare statistiche.
- `ColumnNameList`: una matrice di stringhe UTF-8.

  Un elenco dei nomi delle colonne per generare statistiche. Se non viene fornito, per impostazione predefinita verranno utilizzati tutti i nomi delle colonne della tabella.
- `Role`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il ruolo IAM che assume il servizio per generare statistiche.
- `SampleSize`: numero (doppio), non superiore a 100.

  La percentuale di righe utilizzate per generare statistiche. Se non viene fornita, per generare statistiche verrà utilizzata l'intera tabella.
- `CatalogID`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del catalogo dati in cui risiede la tabella. Se non viene fornito, per impostazione predefinita viene utilizzato l'ID dell'account AWS.
- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della configurazione di sicurezza utilizzata per crittografare i log di CloudWatch per l'esecuzione dell'attività delle statistiche delle colonne.

Risposta

- `ColumnStatisticsTaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'identificatore per l'esecuzione dell'attività delle statistiche delle colonne.

Errori

- `AccessDeniedException`
- `EntityNotFoundException`
- `ColumnStatisticsTaskRunningException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `InvalidInputException`

Operazione GetColumnStatisticsTaskRun (Python: get_column_statistics_task_run)

Ottieni i metadati/le informazioni associati per l'esecuzione di un'attività, con un ID di esecuzione attività.

Richiesta

- `ColumnStatisticsTaskRunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore per l'esecuzione dell'attività delle statistiche delle colonne specifica.

Risposta

- `ColumnStatisticsTaskRun`: un oggetto ColumnStatisticsTaskRun.

  Un oggetto `ColumnStatisticsTaskRun` che rappresenta i dettagli dell'esecuzione delle statistiche delle colonne.

Errori

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`

Operazione GetColumnStatisticsTaskRuns (Python: get_column_statistics_task_runs)

Recupera le informazioni su tutte le esecuzioni associate alla tabella specificata.

Richiesta

- `DatabaseName`: obbligatorio: stringa UTF-8.

  Nome del database in cui risiede la tabella.

- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima della risposta.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `ColumnStatisticsTaskRuns`: una matrice di oggetti ColumnStatisticsTaskRun.

  Un elenco delle esecuzioni dell'attività delle statistiche delle colonne.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non sono ancora stati restituite tutte le esecuzioni dell'attività.

Errori

- `OperationTimeoutException`

Operazione ListColumnStatisticsTaskRuns (Python: list_column_statistics_task_runs)

Elenca tutte le attività eseguite per un determinato account.

Richiesta

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima della risposta.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `ColumnStatisticsTaskRunIds`: una matrice di stringhe UTF-8, non superiore a 100.

  Un elenco degli ID delle esecuzioni dell'attività delle statistiche delle colonne.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non sono ancora stati restituiti tutti gli ID delle esecuzioni dell'attività.

Errori

- `OperationTimeoutException`

Operazione StopColumnStatisticsTaskRun (Python: stop_column_statistics_task_run)

Interrompe l'esecuzione di un'operazione per la tabella specificata.

Richiesta

- `DatabaseName`: obbligatorio: stringa UTF-8.

  Nome del database in cui risiede la tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `ColumnStatisticsTaskNotRunningException`
- `ColumnStatisticsTaskStoppingException`
- `OperationTimeoutException`

API del pianificatore del crawler

L'API del pianificatore del crawler descrive i tipi di dati delle ricerche per indicizzazione AWS Glue e l'API per la loro creazione, eliminazione, aggiornamento ed elenco.

Tipi di dati

- Struttura della pianificazione

Struttura della pianificazione

Oggetto di pianificazione che utilizza una dichiarazione `cron` per pianificare un evento.

Campi

- `ScheduleExpression`: stringa UTF-8.

  Espressione `cron` usata per specificare la pianificazione (consulta Pianificazioni basate sul tempo per processi e crawler. Ad esempio, per eseguire un processo ogni giorno alle 12:15 UTC, devi specificare: `cron(15 12 * * ? *)`.
- `State`: stringa UTF-8 (valori validi: `SCHEDULED` | `NOT_SCHEDULED` | `TRANSITIONING`).

  Lo stato della pianificazione.

Operazioni

- Operazione UpdateCrawlerSchedule (Python: update_crawler_schedule)
- Operazione StartCrawlerSchedule (Python: start_crawler_schedule)
- Operazione StopCrawlerSchedule (Python: stop_crawler_schedule)

Operazione UpdateCrawlerSchedule (Python: update_crawler_schedule)

Aggiorna la pianificazione di un crawler utilizzando un'espressione `cron`.

Richiesta

- `CrawlerName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del crawler la cui programmazione è da aggiornare.
- `Schedule`: stringa UTF-8.

  Espressione `cron` aggiornata usata per specificare la pianificazione, consulta Pianificazioni basate sul tempo per processi e crawler. Ad esempio, per eseguire un processo ogni giorno alle 12:15 UTC, devi specificare: `cron(15 12 * * ? *)`.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `VersionMismatchException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

Operazione StartCrawlerSchedule (Python: start_crawler_schedule)

Cambia lo stato della pianificazione del crawler specificato su `SCHEDULED`, a meno che il crawler non sia già in esecuzione o lo stato della pianificazione sia già impostata su `SCHEDULED`.

Richiesta

- `CrawlerName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Nome del crawler da pianificare.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `SchedulerRunningException`
- `SchedulerTransitioningException`
- `NoScheduleException`
- `OperationTimeoutException`

Operazione StopCrawlerSchedule (Python: stop_crawler_schedule)

Imposta lo stato della pianificazione del crawler specificato su `NOT_SCHEDULED`, ma non arresta il crawler se è già in esecuzione.

Richiesta

- `CrawlerName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del crawler il cui stato della programmazione è da impostare.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `SchedulerNotRunningException`
- `SchedulerTransitioningException`

- `OperationTimeoutException`

API script ETL auto-generanti

L'API di generazione degli script ETL descrive i tipi di dati e l'API per la generazione di script ETL in AWS Glue.

Tipi di dati

- Struttura CodeGenNode
- Struttura CodeGenNodeArg
- Struttura CodeGenEdge
- Struttura della posizione
- Struttura CatalogEntry
- Struttura MappingEntry

Struttura CodeGenNode

Rappresenta un nodo in un grafo aciclico orientato (DAG)

Campi

- `Id`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Identifier string pattern.

  Un identificatore del nodo univoco all'interno del grafo del nodo.
- `NodeType`. Obbligatorio: stringa UTF-8.

  Il tipo di nodo.
- `Args`: obbligatorio: una matrice di oggetti CodeGenNodeArg, non superiore a 50 strutture.

  Proprietà del nodo sotto forma di coppie nome-valore.
- `LineNumber`: numero (intero).

  Il numero di riga del nodo.

Struttura CodeGenNodeArg

Un argomento o una proprietà di un nodo.

Campi

- `Name`. Obbligatorio: stringa UTF-8.

  Il nome dell'argomento o della proprietà.
- `Value`. Obbligatorio: stringa UTF-8.

  Il valore dell'argomento o della proprietà.
- `Param`: booleano.

  True se il valore viene utilizzato come parametro.

Struttura CodeGenEdge

Rappresenta un edge direzionale in un grafo aciclico orientato (DAG).

Campi

- `Source`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Identifier string pattern.

  L'ID del nodo in cui inizia l'edge.
- `Target`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Identifier string pattern.

  L'ID del nodo in cui termina l'edge.
- `TargetParameter`: stringa UTF-8.

  La destinazione dell'edge.

Struttura della posizione

La posizione delle risorse.

Campi

- `Jdbc`: una matrice di oggetti CodeGenNodeArg, non superiore a 50 strutture.

  Una posizione JDBC.
- `S3`: una matrice di oggetti CodeGenNodeArg, non superiore a 50 strutture.

  Posizione Amazon Simple Storage Service (Amazon S3).
- `DynamoDB`: una matrice di oggetti CodeGenNodeArg, non superiore a 50 strutture.

  Posizione di una tabella Amazon DynamoDB.

Struttura CatalogEntry

Specifica una definizione di tabella in AWS Glue Data Catalog.

Campi

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il database in cui risiedono i metadata della tabella.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della tabella in questione.

Struttura MappingEntry

Definisce una mappatura.

Campi

- `SourceTable`: stringa UTF-8.

  Il nome della tabella di origine.
- `SourcePath`: stringa UTF-8.

  Il percorso di origine .

- `SourceType`: stringa UTF-8.

  Il tipo di sorgente.

- `TargetTable`: stringa UTF-8.

  La tabella di destinazione.

- `TargetPath`: stringa UTF-8.

  Il percorso di destinazione.

- `TargetType`: stringa UTF-8.

  Il tipo di destinazione.

Operazioni

- Operazione CreateScript (Python: create_script)
- Operazione GetDataflowGraph (Python: get_dataflow_graph)
- Operazione GetMapping (Python: get_mapping)
- Operazione GetPlan (Python: get_plan)

Operazione CreateScript (Python: create_script)

Trasforma un grafo aciclico orientato (DAG) in codice.

Richiesta

- `DagNodes`: una matrice di oggetti CodeGenNode.

  Un elenco dei nodi del DAG.

- `DagEdges`: una matrice di oggetti CodeGenEdge.

  Un elenco dei confini del DAG.

- `Language`: stringa UTF-8 (valori validi: `PYTHON` | `SCALA`).

  Il linguaggio di programmazione del codice derivante dal DAG.

Risposta

- `PythonScript`: stringa UTF-8.

  Lo script in Python generato dal DAG.
- `ScalaCode`: stringa UTF-8.

  Il codice Scala generato dal DAG.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Operazione GetDataflowGraph (Python: get_dataflow_graph)

Trasforma uno script in Python in un grafo aciclico orientato (DAG).

Richiesta

- `PythonScript`: stringa UTF-8.

  Lo script in Python da trasformare.

Risposta

- `DagNodes`: una matrice di oggetti CodeGenNode.

  Un elenco dei nodi del DAG risultante.
- `DagEdges`: una matrice di oggetti CodeGenEdge.

  Un elenco dei confini del DAG risultante.

Errori

- `InvalidInputException`
- `InternalServiceException`

- `OperationTimeoutException`

Operazione GetMapping (Python: get_mapping)

Crea mappature.

Richiesta

- `Source`: obbligatorio: un oggetto CatalogEntry.

  Specifica la tabella di origine.
- `Sinks`: una matrice di oggetti CatalogEntry.

  Un elenco di tabelle di destinazione.
- `Location`: un oggetto Ubicazione.

  Parametri per la mappatura.

Risposta

- `Mapping`: obbligatorio: una matrice di oggetti MappingEntry.

  Un elenco delle mappature per le destinazioni specificate.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Operazione GetPlan (Python: get_plan)

Ottiene il codice per eseguire una mappatura specificata.

Richiesta

- `Mapping`: obbligatorio: una matrice di oggetti MappingEntry.

L'elenco delle mappature da una tabella di origine per le tabelle di destinazione.

- `Source`: obbligatorio: un oggetto CatalogEntry.

  La tabella di origine.

- `Sinks`: una matrice di oggetti CatalogEntry.

  Le tabelle di destinazione.

- `Location`: un oggetto Ubicazione.

  Parametri per la mappatura.

- `Language`: stringa UTF-8 (valori validi: `PYTHON` | `SCALA`).

  Il linguaggio di programmazione del codice per eseguire la mappatura.

- `AdditionalPlanOptionsMap`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Una mappa per contenere parametri facoltativi chiave-valore aggiuntivi.

  Attualmente, vengono supportate queste coppie chiave-valore:

  - `inferSchema` : specifica se impostare `inferSchema` su true o false per lo script predefinito generato da un processo AWS Glue. Ad esempio, per impostare `inferSchema` su true, bisogna fornire la seguente coppia di chiave-valore:

    ```
    --additional-plan-options-map '{"inferSchema":"true"}'
    ```

Risposta

- `PythonScript`: stringa UTF-8.

  Uno script in Python per eseguire la mappatura.

- `ScalaCode`: stringa UTF-8.

  Codice Scala per eseguire la mappatura.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

API processo visuale

L'API Visual Job consente di creare processi di integrazione dei dati utilizzando l' AWS Glue API di un oggetto JSON che rappresenta una configurazione visiva di un AWS Glue lavoro.

Viene fornito un elenco di `CodeGenConfigurationNodes` API di creazione o aggiornamento del lavoro per registrare un DAG in AWS Glue Studio per il lavoro creato e generare il codice associato.

Tipi di dati

- CodeGenConfigurationNode struttura
- Struttura JDBC ConnectorOptions
- StreamingDataPreviewOptions struttura
- AthenaConnectorSource struttura
- Struttura JDBC ConnectorSource
- SparkConnectorSource struttura
- CatalogSource struttura
- Struttura MySQL CatalogSource
- Struttura PostgreSQL CatalogSource
- CatalogSource Struttura OracleSQL
- Struttura Microsoft SQL ServerCatalogSource
- CatalogKinesisSource struttura
- DirectKinesisSource struttura
- KinesisStreamingSourceOptions struttura
- CatalogKafkaSource struttura
- DirectKafkaSource struttura
- KafkaStreamingSourceOptions struttura

- RedshiftSource struttura
- AmazonRedshiftSource struttura
- AmazonRedshiftNodeData struttura
- AmazonRedshiftAdvancedOption struttura
- Struttura Option
- struttura S3 CatalogSource
- Struttura S3 SourceAdditionalOptions
- Struttura S3 CsvSource
- Struttura DirectJDBCSource
- Struttura S3 DirectSourceAdditionalOptions
- Struttura S3 JsonSource
- Struttura S3 ParquetSource
- Struttura S3 DeltaSource
- Struttura S3 CatalogDeltaSource
- CatalogDeltaSource struttura
- Struttura S3 HudiSource
- Struttura S3 CatalogHudiSource
- CatalogHudiSource struttura
- Struttura DynamoDB CatalogSource
- RelationalCatalogSource struttura
- struttura JDBC ConnectorTarget
- SparkConnectorTarget struttura
- BasicCatalogTarget struttura
- Struttura MySQL CatalogTarget
- Struttura PostgreSQL CatalogTarget
- Struttura OracleSQL CatalogTarget
- Struttura Microsoft SQL ServerCatalogTarget
- RedshiftTarget struttura
- AmazonRedshiftTarget struttura

- UpsertRedshiftTargetOptions struttura
- struttura S3 CatalogTarget
- Struttura S3 GlueParquetTarget
- CatalogSchemaChangePolicy struttura
- struttura S3 DirectTarget
- Struttura S3 HudiCatalogTarget
- Struttura S3 HudiDirectTarget
- Struttura S3 DeltaCatalogTarget
- Struttura S3 DeltaDirectTarget
- DirectSchemaChangePolicy struttura
- ApplyMapping struttura
- Struttura mappatura
- SelectFields struttura
- DropFields struttura
- RenameField struttura
- Struttura Spigot
- Struttura join
- JoinColumn struttura
- SplitFields struttura
- SelectFromCollection struttura
- FillMissingValues struttura
- Struttura filtro
- FilterExpression struttura
- FilterValue struttura
- CustomCode struttura
- Struttura SparkSQL
- SqlAlias struttura
- DropNullFields struttura
- NullCheckBoxList struttura
- NullValueField struttura

- Struttura Datatype
- Struttura Merge
- Struttura unione
- Struttura PIIDetection
- Struttura aggregata
- DropDuplicates struttura
- GovernedCatalogTarget struttura
- GovernedCatalogSource struttura
- AggregateOperation struttura
- GlueSchema struttura
- GlueStudioSchemaColumn struttura
- GlueStudioColumn struttura
- DynamicTransform struttura
- TransformConfigParameter struttura
- EvaluateDataQuality struttura
- struttura DQ ResultsPublishingOptions
- Struttura DQ StopJobOnFailureOptions
- EvaluateDataQualityMultiFrame struttura
- Struttura Recipe
- RecipeReference struttura
- SnowflakeNodeData struttura
- SnowflakeSource struttura
- SnowflakeTarget struttura
- ConnectorDataSource struttura
- ConnectorDataTarget struttura

CodeGenConfigurationNode struttura

`CodeGenConfigurationNode` enumera tutti i tipi di nodo validi. È possibile compilare una e solo una delle variabili membro.

Campi

- `AthenaConnectorSource`: un oggetto AthenaConnectorSource.

  Specifica un connettore per un'origine dati Amazon Athena.
- `JDBCConnectorSource`: un oggetto JDBC ConnectorSource.

  Specifica un connettore per un'origine dati JDBC.
- `SparkConnectorSource`: un oggetto SparkConnectorSource.

  Specifica un connettore per un'origine dati Apache Spark.
- `CatalogSource`: un oggetto CatalogSource.

  Speciifica un data store nel AWS Glue Data Catalog.
- `RedshiftSource`: un oggetto RedshiftSource.

  Specifica un archivio dati Amazon Redshift.
- `S3CatalogSource`: un oggetto S3 CatalogSource.

  Speciifica un data store Amazon S3 nel Data Catalog AWS Glue .
- `S3CsvSource`: un oggetto S3 CsvSource.

  Specifica un archivio dati CSV (valori delimitati da comandi) archiviati in Amazon S3.
- `S3JsonSource`: un oggetto S3 JsonSource.

  Specifica un archivio dati JSON in Amazon S3.
- `S3ParquetSource`: un oggetto S3 ParquetSource.

  Specifica un archivio dati di Apache Parquet archiviato in Amazon S3.
- `RelationalCatalogSource`: un oggetto RelationalCatalogSource.

  Speciifica un data store di catalogo relazionale nel Data Catalog. AWS Glue
- `DynamoDBCatalogSource`: un oggetto DynamoDB CatalogSource.

  Specifica un data store DynamoDBC Catalog nel Data Catalog. AWS Glue
- `JDBCConnectorTarget`: un oggetto JDBC ConnectorTarget.

  Specifica una destinazioni di dati che scrive su Amazon S3 nell'archiviazione colonnare di Apache Parquet.

- `SparkConnectorTarget`: un oggetto SparkConnectorTarget.

  Specifica una destinazione che utilizza un connettore Apache Spark.

- `CatalogTarget`: un oggetto BasicCatalogTarget.

  Specifica una destinazione che utilizza una AWS Glue tabella Data Catalog.

- `RedshiftTarget`: un oggetto RedshiftTarget.

  Specifica una destinazione che utilizza Amazon Redshift.

- `S3CatalogTarget`: un oggetto S3 CatalogTarget.

  Speciifica un target di dati che scrive su Amazon S3 utilizzando AWS Glue il Data Catalog.

- `S3GlueParquetTarget`: un oggetto S3 GlueParquetTarget.

  Specifica una destinazioni di dati che scrive su Amazon S3 nell'archiviazione colonnare di Apache Parquet.

- `S3DirectTarget`: un oggetto S3 DirectTarget.

  Specifica una destinazione di dati che scrive su Amazon S3.

- `ApplyMapping`: un oggetto ApplyMapping.

  Specifica una trasformazione che mappa le chiavi delle proprietà dei dati nell'origine dei dati alle chiavi delle proprietà dei dati nella destinazione. È possibile rinominare le chiavi, modificare i tipi di dati per le chiavi e scegliere le chiavi da eliminare dal set di dati.

- `SelectFields`: un oggetto SelectFields.

  Specifica una trasformazione che sceglie le chiavi della proprietà dati che si desidera conservare.

- `DropFields`: un oggetto DropFields.

  Specifica una trasformazione che sceglie le chiavi della proprietà dati che si desidera eliminare.

- `RenameField`: un oggetto RenameField.

  Specifica una trasformazione che rinominerà una singola chiave di proprietà dati.

- `Spigot`: un oggetto Spigot.

  Specifica una trasformazione che scrive campioni dei dati in un bucket Amazon S3.

- `Join`: un oggetto Join.

Specifica una trasformazione che unisce due set di dati in un unico set di dati utilizzando una frase di confronto sulle chiavi di proprietà dei dati specificate. È possibile utilizzare inner, outer, left, right, left semi e left anti join.

- `SplitFields`: un oggetto SplitFields.

  Specifica una trasformazione che divide le chiavi della proprietà dati in due `DynamicFrames`. L'output è una raccolta di `DynamicFrames`: uno con le chiavi di proprietà dei dati selezionate e uno con le chiavi di proprietà dei dati rimanenti.

- `SelectFromCollection`: un oggetto SelectFromCollection.

  Specifica una trasformazione che sceglie un `DynamicFrame` da una raccolta di `DynamicFrames`. L'output è il `DynamicFrame` selezionato

- `FillMissingValues`: un oggetto FillMissingValues.

  Specifica una trasformazione che individua i registri nel set di dati che hanno valori mancanti e aggiunge un nuovo campo con un valore determinato dall'imputazione. Il set di dati di input viene utilizzato per addestrare il modello di machine learning che determina quale dovrebbe essere il valore mancante.

- `Filter`: un oggetto Filtro.

  Specifica una trasformazione che divide un set di dati in due, in base a una condizione di filtro.

- `CustomCode`: un oggetto CustomCode.

  Specifica una trasformazione che utilizza il codice personalizzato fornito per eseguire la trasformazione dei dati. L'output è una raccolta di. DynamicFrames

- `SparkSQL`: un oggetto SparkSQL.

  Specifica una trasformazione in cui si inserisce una query SQL utilizzando la sintassi Spark SQL per trasformare i dati. L'output è un singolo `DynamicFrame`.

- `DirectKinesisSource`: un oggetto DirectKinesisSource.

  Specifica un'origine dati Amazon Kinesis diretta.

- `DirectKafkaSource`: un oggetto DirectKafkaSource.

  Specifica un archivio dati Apache Kafka.

- `CatalogKinesisSource`: un oggetto CatalogKinesisSource.

Specifica un'origine dati Kinesis nel Data Catalog AWS Glue .

- `CatalogKafkaSource`: un oggetto CatalogKafkaSource.

  Specifica un archivio dati Apache Kafka nel catalogo dati.

- `DropNullFields`: un oggetto DropNullFields.

  Specifica una trasformazione che rimuove le colonne dal set di dati se tutti i valori nella colonna sono “null”. Per impostazione predefinita, AWS Glue Studio riconosce gli oggetti nulli, ma alcuni valori come stringhe vuote, stringhe «nulle», numeri interi -1 o altri segnaposto come zeri, non vengono riconosciuti automaticamente come nulli.

- `Merge`: un oggetto Unione.

  Specifica una trasformazione che unisce `DynamicFrame` a con un `DynamicFrame` di staging basato sulle chiavi primarie specificate per identificare i registri. I registri duplicati (registri con le stesse chiavi primarie) non vengono deduplicati.

- `Union`: un oggetto Union.

  Specifica una trasformazione che combina le righe di due o più set di dati in un unico risultato.

- `PIIDetection`: un oggetto PIIDetection.

  Specifica una trasformazione che identifica, rimuove o maschera i dati PII.

- `Aggregate`: un oggetto Aggregazione.

  Specifica una trasformazione che raggruppa le righe in base ai campi scelti e calcola il valore aggregato in base alla funzione specificata.

- `DropDuplicates`: un oggetto DropDuplicates.

  Specifica una trasformazione che rimuove le righe di dati ripetuti da un set di dati.

- `GovernedCatalogTarget`: un oggetto GovernedCatalogTarget.

  Specifica una destinazione di dati che scrive su un catalogo governato.

- `GovernedCatalogSource`: un oggetto GovernedCatalogSource.

  Specifica un'origine dei dati in un catalogo dati governato.

- `MicrosoftSQLServerCatalogSource`: un oggetto Microsoft SQL ServerCatalogSource.

  Specifica un'origine dei dati di Microsoft SQL Server nel Catalogo dati di AWS Glue .

- `MySQLCatalogSource`: un oggetto MySQL CatalogSource.

  Specifica un'origine dati MySQL nel Data Catalog. AWS Glue
- `OracleSQLCatalogSource`: un oggetto OracleSQL CatalogSource.

  Specifica un'origine dati Oracle nel Data Catalog. AWS Glue
- `PostgreSQLCatalogSource`: un oggetto PostgreSQL CatalogSource.

  Specifica un'origine dati PostgressQL nel Data Catalog. AWS Glue
- `MicrosoftSQLServerCatalogTarget`: un oggetto Microsoft SQL ServerCatalogTarget.

  Specifica una destinazione che utilizza Microsoft SQL.
- `MySQLCatalogTarget`: un oggetto MySQL CatalogTarget.

  Specifica una destinazione che utilizza MySQL.
- `OracleSQLCatalogTarget`: un oggetto OracleSQL CatalogTarget.

  Specifica una destinazione che utilizza Oracle SQL.
- `PostgreSQLCatalogTarget`: un oggetto PostgreSQL CatalogTarget.

  Specifica una destinazione che utilizza Postgres SQL.
- `DynamicTransform`: un oggetto DynamicTransform.

  Specifica una trasformazione visiva personalizzata creata da un utente.
- `EvaluateDataQuality`: un oggetto EvaluateDataQuality.

  Specifica i criteri di valutazione della qualità dei dati.
- `S3CatalogHudiSource`: un oggetto S3 CatalogHudiSource.

  Specifica un'origine dati Hudi registrata nel Data Catalog. AWS Glue L'origine dati deve essere archiviata in. Amazon S3
- `CatalogHudiSource`: un oggetto CatalogHudiSource.

  Specifica una fonte di dati Hudi registrata nel AWS Glue Data Catalog.
- `S3HudiSource`: un oggetto S3 HudiSource.

  Specifica un'origine dati Hudi memorizzata in. Amazon S3
- `S3HudiCatalogTarget`: un oggetto S3 HudiCatalogTarget.

Specifica una destinazione che scrive su un'origine dati Hudi nel Data Catalog. AWS Glue

- `S3HudiDirectTarget`: un oggetto S3 HudiDirectTarget.

Speciifica una destinazione che scrive su una fonte di dati Hudi in. Amazon S3

- `S3CatalogDeltaSource`: un oggetto S3 CatalogDeltaSource.

Specifica un'origine dati Delta Lake registrata nel Data Catalog. AWS Glue L'origine dati deve essere archiviata in Amazon S3.

- `CatalogDeltaSource`: un oggetto CatalogDeltaSource.

Specifica un'origine dati Delta Lake registrata nel AWS Glue Data Catalog.

- `S3DeltaSource`: un oggetto S3 DeltaSource.

Specifica un'origine dati Delta Lake memorizzata in. Amazon S3

- `S3DeltaCatalogTarget`: un oggetto S3 DeltaCatalogTarget.

Speciifica una destinazione che scrive su un'origine dati Delta Lake nel AWS Glue Data Catalog.

- `S3DeltaDirectTarget`: un oggetto S3 DeltaDirectTarget.

Speciifica una destinazione che esegue la scrittura su un'origine dati Delta Lake in. Amazon S3

- `AmazonRedshiftSource`: un oggetto AmazonRedshiftSource.

Specifica una destinazione che scrive su un'origine dati in Amazon Redshift.

- `AmazonRedshiftTarget`: un oggetto AmazonRedshiftTarget.

Specifica una destinazione che scrive su una destinazione dati in Amazon Redshift.

- `EvaluateDataQualityMultiFrame`: un oggetto EvaluateDataQualityMultiFrame.

Specifica i criteri di valutazione della qualità dei dati. Consente più dati di input e restituisce una raccolta di frame dinamici.

- `Recipe`: un oggetto Recipe.

Specifica un nodo di AWS Glue DataBrew ricetta.

- `SnowflakeSource`: un oggetto SnowflakeSource.

Specifica un'origine dati Snowflake.

- `SnowflakeTarget`: un oggetto SnowflakeTarget.

Specifica una destinazione che scrive su un'origine dati Snowflake.

- `ConnectorDataSource`: un oggetto ConnectorDataSource.

  Specifica un'origine generata con opzioni di connessione standard.

- `ConnectorDataTarget`: un oggetto ConnectorDataTarget.

  Specifica un a destinazione generata con opzioni di connessione standard.

Struttura JDBC ConnectorOptions

Opzioni di connessione aggiuntive per il connettore.

Campi

- `FilterPredicate`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Clausola condizione extra per filtrare i dati dall'origine. Ad esempio:

  ```
  BillingCity='Mountain View'
  ```

  Quando si utilizza una query anziché un nome di tabella, è necessario verificare che la query funzioni con il `filterPredicate` specificato.

- `PartitionColumn`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome di una colonna intera utilizzata per il partizionamento. Questa opzione funziona solo quando è inclusa con `lowerBound`, `upperBound` e `numPartitions`. Questa opzione funziona allo stesso modo del lettore Spark SQL JDBC.

- `LowerBound`: numero (long), non superiore a Nessuno.

  Il valore minimo di `partitionColumn` che viene utilizzato per decidere lo stride della partizione.

- `UpperBound`: numero (long), non superiore a Nessuno.

  Il valore massimo di `partitionColumn` che viene utilizzato per decidere lo stride della partizione.

- `NumPartitions`: numero (long), non superiore a Nessuno.

  Il numero di partizioni. Questo valore, insieme a `lowerBound` (incluso) e `upperBound` (escluso), forma stride di partizione per espressioni con le clausole `WHERE` generate che vengono utilizzate per dividere la `partitionColumn`.

- `JobBookmarkKeys`: una matrice di stringhe UTF-8.

  Il nome delle chiavi dei segnalibri di processo su cui eseguire l'ordinamento.

- `JobBookmarkKeysSortOrder`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il criterio di ordinamento crescente o decrescente.

- `DataTypeMapping`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 (valori validi:`ARRAY` | `BIGINT` | `BINARY` | `BIT` | `BLOB` | `BOOLEAN` | `CHAR` | `CLOB` | `DATALINK` | `DATE` | `DECIMAL` | `DISTINCT` | `DOUBLE` | `FLOAT` | `INTEGER` | `JAVA_OBJECT` | `LONGNVARCHAR` | `LONGVARBINARY` | `LONGVARCHAR` | `NCHAR` | `NCLOB` | `NULL` | `NUMERIC` | `NVARCHAR` | `OTHER` | `REAL` | `REF` | `REF_CURSOR` | `ROWID` | `SMALLINT` | `SQLXML` | `STRUCT` | `TIME` | `TIME_WITH_TIMEZONE` | `TIMESTAMP` | `TIMESTAMP_WITH_TIMEZONE` | `TINYINT` | `VARBINARY` | `VARCHAR`).

  Ogni valore è una stringa UTF-8 (valori validi: `DATE` | `STRING` | `TIMESTAMP` | `INT` | `FLOAT` | `LONG` | `BIGDECIMAL` | `BYTE` | `SHORT` | `DOUBLE`).

  Mappatura del tipo di dati personalizzata che crea una mappatura da un tipo di dati JDBC a un tipo di dati AWS Glue . Ad esempio, l'opzione `"dataTypeMapping":{"FLOAT":"STRING"}` mappa i campi di dati di tipo JDBC `FLOAT` nel `String` tipo Java chiamando il `ResultSet.getString()` metodo del driver e lo utilizza per creare il record. AWS Glue L'oggetto `ResultSet` viene implementato da ciascun driver, quindi il comportamento è specifico del driver utilizzato. Consulta la documentazione relativa al driver JDBC per capire come il driver esegue le conversioni.

StreamingDataPreviewOptions struttura

Specifica le opzioni relative all'anteprima dei dati per la visualizzazione di un campione dei dati.

Campi

- `PollingTime`: numero (lungo), almeno 10.

  Il tempo di polling in millisecondi.

- `RecordPollingLimit`: numero (lungo), almeno 1.

  Il limite al numero di registri per cui è stato fatto il polling.

AthenaConnectorSource struttura

Specifica un connettore per un'origine dati Amazon Athena.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.
- `ConnectionName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della connessione associata al connettore.
- `ConnectorName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome di un connettore che consente di accedere all'archivio dati in AWS Glue Studio.
- `ConnectionType`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il tipo di connessione, come marketplace.athena o custom.athena, che designa una connessione a un archivio dati Amazon Athena.
- `ConnectionTable`: stringa UTF-8, corrispondente a Custom string pattern #35.

  Il nome della tabella nell'origine dati.
- `SchemaName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del gruppo di log CloudWatch da cui leggere. Ad esempio, `/aws-glue/jobs/output`.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine Athena personalizzata.

Struttura JDBC ConnectorSource

Specifica un connettore per un'origine dati JDBC.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.
- `ConnectionName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

Il nome della connessione associata al connettore.

- `ConnectorName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome di un connettore che consente di accedere all'archivio dati in Studio. AWS Glue

- `ConnectionType`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il tipo di connessione, come marketplace.jdbc o custom.jdbc, che designa una connessione a un archivio dati JDBC.

- `AdditionalOptions`: un oggetto JDBC ConnectorOptions.

  Opzioni di connessione aggiuntive per il connettore.

- `ConnectionTable`: stringa UTF-8, corrispondente a Custom string pattern #35.

  Il nome della tabella nell'origine dati.

- `Query`: stringa UTF-8, corrispondente a Custom string pattern #36.

  La tabella o la query SQL da cui ottenere i dati. Puoi specificare `ConnectionTable` o `query`, ma non entrambi.

- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine JDBC personalizzata.

SparkConnectorSource struttura

Specifica un connettore per un'origine dati Apache Spark.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.

- `ConnectionName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della connessione associata al connettore.

- `ConnectorName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome di un connettore che consente di accedere all'archivio dati in AWS Glue Studio.

- `ConnectionType`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il tipo di connessione, come marketplace.spark o custom.spark, che designa una connessione a un archivio dati di Apache Spark.
- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Opzioni di connessione aggiuntive per il connettore.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine Spark personalizzata.

CatalogSource struttura

Speciifica un data store nel AWS Glue Data Catalog.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del archivio dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.

Struttura MySQL CatalogSource

Specifica un'origine dati MySQL nel Data Catalog. AWS Glue

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.

Struttura PostgreSQL CatalogSource

Specifica un'origine dati PostgressQL nel Data Catalog. AWS Glue

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.

CatalogSource Struttura OracleSQL

Speciifica un'origine dati Oracle nel Data Catalog. AWS Glue

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.

Struttura Microsoft SQL ServerCatalogSource

Specifica un'origine dei dati di Microsoft SQL Server nel Catalogo dati di AWS Glue .

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.

CatalogKinesisSource struttura

Specifica un'origine dati Kinesis nel Data Catalog AWS Glue .

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.
- `WindowSize`: numero (intero), non superiore a Nessuno.

  La quantità di tempo da dedicare all'elaborazione di ciascun micro batch.
- `DetectSchema`: booleano.

  Se determinare automaticamente o meno lo schema dai dati in entrata.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `StreamingOptions`: un oggetto KinesisStreamingSourceOptions.

  Opzioni aggiuntive per l'origine dati di streaming Kinesis.

- `DataPreviewOptions`: un oggetto StreamingDataPreviewOptions.

  Opzioni aggiuntive per l'anteprima dei dati.

DirectKinesisSource struttura

Specifica un'origine dati Amazon Kinesis diretta.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.
- `WindowSize`: numero (intero), non superiore a Nessuno.

  La quantità di tempo da dedicare all'elaborazione di ciascun micro batch.
- `DetectSchema`: booleano.

  Se determinare automaticamente o meno lo schema dai dati in entrata.
- `StreamingOptions`: un oggetto KinesisStreamingSourceOptions.

  Opzioni aggiuntive per l'origine dati di streaming Kinesis.
- `DataPreviewOptions`: un oggetto StreamingDataPreviewOptions.

  Opzioni aggiuntive per l'anteprima dei dati.

KinesisStreamingSourceOptions struttura

Opzioni aggiuntive per l'origine dati di streaming Amazon Kinesis.

Campi

- `EndpointUrl`: stringa UTF-8, corrispondente a Custom string pattern #34.

  L'URL dell'endpoint di Kinesis.
- `StreamName`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del flusso di dati Kinesis.
- `Classification`: stringa UTF-8, corrispondente a Custom string pattern #34.

Una classificazione facoltativa.

- `Delimiter`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il carattere delimitatore.

- `StartingPosition`: stringa UTF-8 (valori validi: `latest="LATEST"` | `trim_horizon="TRIM_HORIZON"` | `earliest="EARLIEST"` | `timestamp="TIMESTAMP"`).

  La posizione di partenza nel flusso dei dati Kinesis da cui leggere i dati. I valori possibili sono `"latest"`, `"trim_horizon"`, `"earliest"` o una stringa di timestamp in formato UTC con il modello `yyyy-mm-ddTHH:MM:SSZ`, dove Z rappresenta uno scostamento del fuso orario UTC con un segno +/- (ad esempio: "2023-04-04T08:00:00-04:00"). Il valore predefinito è `"latest"`.

  Nota: l'utilizzo di un valore che è una stringa di timestamp in formato UTC per «startingPosition» è supportato solo per AWS Glue la versione 4.0 o successiva.

- `MaxFetchTimeInMs`: numero (long), non superiore a Nessuno.

  Il tempo massimo impiegato nell'esecutore del processo per recuperare un registro dal flusso dei dati Kinesis per partizione, specificato in millisecondi (ms). Il valore di default è `1000`.

- `MaxFetchRecordsPerShard`: numero (long), non superiore a Nessuno.

  Il numero massimo di record da recuperare per shard nel flusso di dati Kinesis per microbatch. Nota: il client può superare questo limite se il job di streaming ha già letto record aggiuntivi da Kinesis (nella stessa chiamata get-records). Se `MaxFetchRecordsPerShard` deve essere rigoroso, deve essere un multiplo di. `MaxRecordPerRead` Il valore di default è `100000`.

- `MaxRecordPerRead`: numero (long), non superiore a Nessuno.

  Il numero massimo di registri da recuperare nel flusso dei dati Kinesis in ciascuna operazione getRecords. Il valore predefinito è `10000`.

- `AddIdleTimeBetweenReads`: booleano.

  Aggiunge un ritardo tra due operazioni consecutive getRecords. Il valore predefinito è `"False"`. Questa opzione è configurabile solo per Glue versione 2.0 e successive.

- `IdleTimeBetweenReadsInMs`: numero (long), non superiore a Nessuno.

  Il ritardo minimo tra due operazioni consecutive getRecords, specificato in ms. Il valore predefinito è `1000`. Questa opzione è configurabile solo per Glue versione 2.0 e successive.

- `DescribeShardInterval`: numero (long), non superiore a Nessuno.

L'intervallo di tempo minimo tra due chiamate ListShards API entro il quale lo script deve prendere in considerazione il resharding. Il valore predefinito è `1s`.

- `NumRetries`: numero (intero), non superiore a Nessuno.

  Il numero massimo di tentativi per le richieste API Kinesis Data Streams. Il valore di default è `3`.

- `RetryIntervalMs`: numero (long), non superiore a Nessuno.

  Il periodo di raffreddamento (specificato in ms) prima di riprovare la chiamata API Kinesis Data Streams. Il valore di default è `1000`.

- `MaxRetryIntervalMs`: numero (long), non superiore a Nessuno.

  Il periodo di raffreddamento (specificato in ms) tra due tentativi di chiamata API Kinesis Data Streams. Il valore predefinito è `10000`.

- `AvoidEmptyBatches`: booleano.

  Impedisce la creazione di un processo microbatch vuoto controllando la presenza di dati non letti nel flusso dei dati Kinesis prima che il batch venga avviato. Il valore predefinito è `"False"`.

- `StreamArn`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della risorsa Amazon (ARN) del flusso di dati Kinesis.

- `RoleArn`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della risorsa Amazon (ARN) del ruolo da assumere tramite il servizio di token di sicurezza AWS (AWS STS). Questo ruolo deve disporre delle autorizzazioni per descrivere o leggere le operazioni dei registri per il flusso di dati Kinesis. Quando si accede a un flusso di dati in un altro account, è necessario utilizzare questo parametro. Usato in combinazione con `"awsSTSSessionName"`.

- `RoleSessionName`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Un identificatore della sessione che assume il ruolo tramite AWS STS. Quando si accede a un flusso di dati in un altro account, è necessario utilizzare questo parametro. Usato in combinazione con `"awsSTSRoleARN"`.

- `AddRecordTimestamp`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Quando questa opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "__src_timestamp" che indica l'ora in cui il record corrispondente è stato ricevuto dal

flusso. Il valore predefinito è "false". Questa opzione è supportata nella AWS Glue versione 4.0 o successiva.

- `EmitConsumerLagMetrics`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Quando questa opzione è impostata su «true», per ogni batch emette le metriche relative alla durata compresa tra il record più vecchio ricevuto dallo stream e l'ora in AWS Glue cui arriva. CloudWatch Il nome della metrica è «glue.driver.streaming. maxConsumerLagInMs». Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.

- `StartingTimestamp`: stringa UTF-8.

  Il timestamp del record nel flusso di dati Kinesis da cui iniziare la lettura dei dati. I valori possibili sono una stringa di timestamp in formato UTC del modello `yyyy-mm-ddTHH:MM:SSZ`, dove Z rappresenta uno scostamento del fuso orario UTC con un segno +/- (ad esempio: "2023-04-04T 08:00:00+08:00").

CatalogKafkaSource struttura

Specifica un archivio dati Apache Kafka nel catalogo dati.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del archivio dati.

- `WindowSize`: numero (intero), non superiore a Nessuno.

  La quantità di tempo da dedicare all'elaborazione di ciascun micro batch.

- `DetectSchema`: booleano.

  Se determinare automaticamente o meno lo schema dai dati in entrata.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.

- `StreamingOptions`: un oggetto KafkaStreamingSourceOptions.

Specifica le opzioni di streaming.

- `DataPreviewOptions`: un oggetto StreamingDataPreviewOptions.

  Specifica le opzioni relative all'anteprima dei dati per la visualizzazione di un campione dei dati.

DirectKafkaSource struttura

Specifica un archivio dati Apache Kafka.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del archivio dati.
- `StreamingOptions`: un oggetto KafkaStreamingSourceOptions.

  Specifica le opzioni di streaming.
- `WindowSize`: numero (intero), non superiore a Nessuno.

  La quantità di tempo da dedicare all'elaborazione di ciascun micro batch.
- `DetectSchema`: booleano.

  Se determinare automaticamente o meno lo schema dai dati in entrata.
- `DataPreviewOptions`: un oggetto StreamingDataPreviewOptions.

  Specifica le opzioni relative all'anteprima dei dati per la visualizzazione di un campione dei dati.

KafkaStreamingSourceOptions struttura

Opzioni aggiuntive per lo streaming.

Campi

- `BootstrapServers`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Un elenco di URL del server bootstrap, ad esempio, come `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Questa opzione deve essere specificata nella chiamata API o definita nei metadati della tabella in catalogo dati.

- `SecurityProtocol`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il protocollo utilizzato per comunicare con i broker. I valori possibili sono `"SSL"` o `"PLAINTEXT"`.

- `ConnectionName`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della connessione.

- `TopicName`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome dell'argomento come specificato in Apache Kafka. Devi specificare almeno uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.

- `Assign`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Lo specifico `TopicPartitions` per consumare. Devi specificare almeno uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.

- `SubscribePattern`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Uuna stringa regex Java che identifichi l'elenco degli argomenti a cui effettuare la sottoscrizione. Devi specificare almeno uno tra `"topicName"`, `"assign"` o `"subscribePattern"`.

- `Classification`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Una classificazione facoltativa.

- `Delimiter`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il carattere delimitatore.

- `StartingOffsets`: stringa UTF-8, corrispondente a Custom string pattern #34.

  La posizione di partenza nell'argomento Kafka da cui leggere i dati. I valori possibili sono `"earliest"` o `"latest"`. Il valore predefinito è `"latest"`.

- `EndingOffsets`: stringa UTF-8, corrispondente a Custom string pattern #34.

  L'endpoint al quale viene terminata una query batch. I valori possibili sono `"latest"` o una stringa JSON che specifica un offset finale per ogni `TopicPartition`.

- `PollTimeoutMs`: numero (long), non superiore a Nessuno.

  Il timeout in millisecondi per il polling dei dati da Kafka negli esecutori del processo Spark. Il valore predefinito è `512`.

- `NumRetries`: numero (intero), non superiore a Nessuno.

Il numero di tentativi prima di non riuscire a recuperare gli offset Kafka. Il valore di default è 3.

- `RetryIntervalMs`: numero (long), non superiore a Nessuno.

  Il tempo di attesa in millisecondi prima di riprovare a recuperare gli offset Kafka. Il valore di default è 10.

- `MaxOffsetsPerTrigger`: numero (long), non superiore a Nessuno.

  Il limite di velocità sul numero massimo di offset elaborati per intervallo di attivazione. Il numero totale di offset specificato viene suddiviso proporzionalmente tra `topicPartitions` di diversi volumi. Il valore di default è null, il che significa che il consumer legge tutti gli offset fino all'ultimo offset noto.

- `MinPartitions`: numero (intero), non superiore a Nessuno.

  Il numero minimo desiderato di partizioni da leggere da Kafka. Il valore di default è null, il che significa che il numero di partizioni Spark è uguale al numero di partizioni Kafka.

- `IncludeHeaders`: booleano.

  Se includere le intestazioni di Kafka. Quando l'opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "glue_streaming_kafka_headers" con tipo `Array[Struct(key: String, value: String)]`. Il valore di default è "false". Questa opzione è disponibile solo nella AWS Glue versione 3.0 o successiva.

- `AddRecordTimestamp`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Quando questa opzione è impostata su "true", l'output dei dati conterrà una colonna aggiuntiva denominata "__src_timestamp" che indica l'ora in cui il record corrispondente è stato ricevuto dall'argomento. Il valore predefinito è "false". Questa opzione è supportata nella AWS Glue versione 4.0 o successiva.

- `EmitConsumerLagMetrics`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Quando questa opzione è impostata su «true», per ogni batch emette le metriche relative alla durata compresa tra il record più vecchio ricevuto dall'argomento e il momento in AWS Glue cui arriva. CloudWatch Il nome della metrica è «glue.driver.streaming. maxConsumerLagInMs». Il valore predefinito è "false". Questa opzione è supportata in AWS Glue versione 4.0 o successive.

- `StartingTimestamp`: stringa UTF-8.

  Il timestamp del record nell'argomento Kinesis da cui iniziare la lettura dei dati. I valori possibili sono una stringa di timestamp in formato UTC del modello `yyyy-mm-ddTHH:MM:SSZ`, dove Z

rappresenta uno scostamento del fuso orario UTC con un segno +/- (ad esempio: "2023-04-04T 08:00:00+08:00").

Deve essere impostato solo un valore tra `StartingTimestamp` e `StartingOffsets`.

RedshiftSource struttura

Specifica un archivio dati Amazon Redshift.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'archivio dati Amazon Redshift.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  La tabella del database da cui leggere.
- `RedshiftTmpDir`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il percorso Amazon S3 in cui i dati temporanei possono essere caricati durante la copia dal database.
- `TmpDirIAMRole`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il ruolo IAM con autorizzazioni.

AmazonRedshiftSource struttura

Il nome della connessione è l'origine Amazon Redshift.

Campi

- `Name`: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine Amazon Redshift.
- `Data`: un oggetto AmazonRedshiftNodeData.

Specifica i dati del nodo di origine Amazon Redshift.

AmazonRedshiftNodeData struttura

Specifica un nodo Amazon Redshift.

Campi

- `AccessType`: stringa UTF-8, corrispondente a Custom string pattern #33.

  Il tipo di accesso per la connessione Redshift. Può essere una connessione diretta o una connessione al catalogo.
- `SourceType`: stringa UTF-8, corrispondente a Custom string pattern #33.

  Il tipo di origine per specificare se una tabella specifica è l'origine o una query personalizzata.
- `Connection`: un oggetto Opzione.

  La AWS Glue connessione al cluster Redshift.
- `Schema`: un oggetto Opzione.

  Il nome dello schema Redshift quando si lavora con una connessione diretta.
- `Table`: un oggetto Opzione.

  Il nome della tabella Redshift quando si lavora con una connessione diretta.
- `CatalogDatabase`: un oggetto Opzione.

  Il nome del database AWS Glue Data Catalog quando si lavora con un catalogo di dati.
- `CatalogTable`: un oggetto Opzione.

  Il nome della tabella AWS Glue Data Catalog quando si lavora con un catalogo di dati.
- `CatalogRedshiftSchema`: stringa UTF-8.

  Il nome dello schema Redshift quando si lavora con un catalogo dati.
- `CatalogRedshiftTable`: stringa UTF-8.

  La tabella del database da cui leggere.
- `TempDir`: stringa UTF-8, corrispondente a Custom string pattern #34.

Il percorso Amazon S3 in cui i dati temporanei possono essere caricati durante la copia dal database.

- `IamRole`: un oggetto Opzione.

  Facoltativo. Il nome del ruolo utilizzato durante la connessione a S3. Se lasciato vuoto, il ruolo IAM assumerà per impostazione predefinita il ruolo nel processo.

- `AdvancedOptions`: una matrice di oggetti AmazonRedshiftAdvancedOption.

  Valori facoltativi durante la connessione al cluster Redshift.

- `SampleQuery`: stringa UTF-8.

  L'SQL utilizzato per recuperare i dati da una fonte Redshift quando SourceType è 'query'.

- `PreAction`: stringa UTF-8.

  L'SQL utilizzato prima di un'esecuzione di MERGE o APPEND con upsert.

- `PostAction`: stringa UTF-8.

  L'SQL utilizzato prima di un'esecuzione di MERGE o APPEND con upsert.

- `Action`: stringa UTF-8.

  Specifica come verrà eseguita la scrittura su un cluster Redshift.

- `TablePrefix`: stringa UTF-8, corrispondente a Custom string pattern #33.

  Specifica il prefisso di una tabella.

- `Upsert`: booleano.

  L'operazione utilizzata in un sink Redshift quando si esegue un APPEND.

- `MergeAction`: stringa UTF-8, corrispondente a Custom string pattern #33.

  L'operazione utilizzata per determinare come verrà gestito un MERGE in un sink Redshift.

- `MergeWhenMatched`: stringa UTF-8, corrispondente a Custom string pattern #33.

  L'operazione utilizzata per determinare come verrà gestito un MERGE in un sink Redshift quando un record esistente corrisponde a un nuovo record.

- `MergeWhenNotMatched`: stringa UTF-8, corrispondente a Custom string pattern #33.

L'operazione utilizzata per determinare come verrà gestito un MERGE in un sink Redshift quando un record esistente non corrisponde a un nuovo record.

- `MergeClause`: stringa UTF-8.

  L'SQL utilizzato in un merge personalizzato per gestire i record corrispondenti.

- `CrawlerConnection`: stringa UTF-8.

  Specifica il nome della connessione associata alla tabella del catalogo utilizzata.

- `TableSchema`: una matrice di oggetti Opzione.

  L'array di output dello schema per un determinato nodo.

- `StagingTable`: stringa UTF-8.

  Il nome della tabella intermedia temporanea utilizzata quando si esegue un MERGE o un APPEND con upsert.

- `SelectedColumns`: una matrice di oggetti Opzione.

  L'elenco dei nomi di colonna utilizzati per determinare un record corrispondente quando si esegue un MERGE o un APPEND con upsert.

AmazonRedshiftAdvancedOption struttura

Specifica un valore facoltativo per la connessione al cluster Redshift.

Campi

- `Key`: stringa UTF-8.

  La chiave dell'opzione di connessione aggiuntiva.

- `Value`: stringa UTF-8.

  Il valore dell'opzione di connessione aggiuntiva.

Struttura Option

Specifica il valore di un'opzione.

Campi

- `Value`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il valore dell'opzione.
- `Label`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica l'etichetta dell'opzione.
- `Description`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica la descrizione dell'opzione.

struttura S3 CatalogSource

Speciifica un data store Amazon S3 nel Data Catalog AWS Glue .

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del archivio dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  La tabella del database da cui leggere.
- `PartitionPredicate`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Le partizioni che soddisfano questo predicato vengono eliminate. I file all'interno del periodo di conservazione in queste partizioni non vengono eliminati. Impostato su `""`: vuoto per impostazione predefinita.
- `AdditionalOptions`: un oggetto S3 SourceAdditionalOptions.

  Specifica opzioni di connessione aggiuntive.

Struttura S3 SourceAdditionalOptions

Specifica opzioni di connessione aggiuntive per l'archivio dati Amazon S3.

Campi

- `BoundedSize`: numero (lungo).

  Imposta il limite superiore per la dimensione di destinazione del set di dati in byte che verranno elaborati.

- `BoundedFiles`: numero (lungo).

  Imposta il limite superiore per il numero di file di destinazione che verranno elaborati.

Struttura S3 CsvSource

Specifica un archivio dati CSV (valori delimitati da comandi) archiviati in Amazon S3.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del archivio dati.

- `Paths`: obbligatorio: una matrice di stringhe UTF-8.

  Un elenco dei percorsi Amazon S3 da cui leggere.

- `CompressionType`: stringa UTF-8 (valori validi: `gzip="GZIP"` | `bzip2="BZIP2"`).

  Specifica il modo in cui i dati sono compressi. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip"`).

- `Exclusions`: una matrice di stringhe UTF-8.

  Una stringa contenente un elenco di JSON di modelli glob in stile Unix da escludere. Ad esempio "[\"**.pdf \"]" esclude tutti i file PDF.

- `GroupSize`: stringa UTF-8, corrispondente a Custom string pattern #34.

  La dimensione del gruppo di destinazione in byte. Il valore di default viene calcolato in base alla dimensione dei dati di input e alle dimensioni del cluster. Quando sono presenti meno di 50.000 file di input, `"groupFiles"` deve essere impostato su `"inPartition"` per rendere effettiva la modifica.

- `GroupFiles`: stringa UTF-8, corrispondente a Custom string pattern #34.

Quando l'input contiene più di 50.000 file, il raggruppamento di file è attivato per impostazione predefinita. Per attivare il raggruppamento con meno di 50.000 file, imposta questo parametro su "inPartition". Per disabilitare il raggruppamento in presenza di più di 50.000 file, imposta il parametro su `''none''`.

- `Recurse`: booleano.

  Se è impostato su "vero", legge i file in modo ricorsivo in tutte le sottodirectory dei percorsi specificati.

- `MaxBand`: numero (intero), non superiore a Nessuno.

  Questa opzione controlla la durata in millisecondi dopo la quale è probabile che l'elenco s3 sia coerente. I file con timestamp di modifica che rientrano negli ultimi millisecondi MaxBand vengono tracciati appositamente quando vengono utilizzati per tenere conto JobBookmarks della coerenza finale di Amazon S3. Per la maggior parte degli utenti non è necessario impostare questa opzione. Il valore di default è 900.000 millisecondi o 15 minuti.

- `MaxFilesInBand`: numero (intero), non superiore a Nessuno.

  Questa opzione specifica il numero massimo di file da salvare negli ultimi secondi maxBand. Se si supera questo valore, i file aggiuntivi vengono saltati e solo elaborati nella successiva esecuzione del processo.

- `AdditionalOptions`: un oggetto S3 DirectSourceAdditionalOptions.

  Specifica opzioni di connessione aggiuntive.

- `Separator`: obbligatorio: stringa UTF-8 (valori validi: `comma="COMMA"` | `ctrla="CTRLA"` | `pipe="PIPE"` | `semicolon="SEMICOLON"` | `tab="TAB"`).

  Specifica il carattere delimitatore. Il valore di default è una virgola: ",", ma è possibile specificare qualsiasi altro carattere.

- `Escaper`: stringa UTF-8, corrispondente a Custom string pattern #35.

  Specifica un carattere di escape. Questa opzione viene utilizzata solo durante la lettura di file CSV. Il valore predefinito è `none`. Se questa opzione è abilitata, il carattere immediatamente seguente viene usato così come è, ad eccezione di un piccolo set di caratteri di escape ben noti (`\n`, `\r`, `\t` e `\0`).

- `QuoteChar`: obbligatorio: stringa UTF-8 (valori validi: `quote="QUOTE"` | `quillemet="QUILLEMET"` | `single_quote="SINGLE_QUOTE"` | `disabled="DISABLED"`).

Specifica il carattere da usare per le virgolette. Per impostazione predefinita vengono usate le virgolette doppie: `'"'`. Imposta questo valore su `-1` per disattivare completamente le virgolette.

- `Multiline`: booleano.

  Un valore booleano che specifica se un singolo registro può estendersi su più righe. Ciò può accadere quando un campo contiene un carattere di nuova riga tra virgolette. Imposta questa opzione su "Vero" se un qualsiasi registro si estende su più righe. Il valore di default è `False`, che consente una divisione dei file più netta durante l'analisi.

- `WithHeader`: booleano.

  Un valore booleano che specifica se trattare la prima riga come intestazione. Il valore predefinito è `False`.

- `WriteHeader`: booleano.

  Un valore booleano che specifica se scrivere l'intestazione nell'output. Il valore predefinito è `True`.

- `SkipFirst`: booleano.

  Un valore booleano che specifica se ignorare la prima riga di dati. Il valore predefinito è `False`.

- `OptimizePerformance`: booleano.

  Un valore booleano che specifica se utilizzare il lettore SIMD CSV avanzato insieme ai formati di memoria colonnare basati su Apache Arrow. AWS Glue Disponibile solo nella versione 3.0.

- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine CSV S3 personalizzata.

Struttura DirectJDBCSource

Specifica la connessione diretta all'origine JDBC.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome della connessione di origine JDBC.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il database della connessione di origine JDBC.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  La tabella della connessione di origine JDBC.
- `ConnectionName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della connessione dell'origine JDBC.
- `ConnectionType`: obbligatorio: stringa UTF-8 (valori validi: `sqlserver` | `mysql` | `oracle` | `postgresql` | `redshift`).

  Il tipo di connessione dell'origine JDBC.
- `RedshiftTmpDir`: stringa UTF-8, corrispondente a Custom string pattern #34.

  La directory temporanea dell'origine JDBC Redshift.

Struttura S3 DirectSourceAdditionalOptions

Specifica opzioni di connessione aggiuntive per l'archivio dati Amazon S3.

Campi

- `BoundedSize`: numero (lungo).

  Imposta il limite superiore per la dimensione di destinazione del set di dati in byte che verranno elaborati.
- `BoundedFiles`: numero (lungo).

  Imposta il limite superiore per il numero di file di destinazione che verranno elaborati.
- `EnableSamplePath`: booleano.

  Imposta l'opzione per abilitare un percorso di esempio.
- `SamplePath`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Se abilitato, specifica il percorso di esempio.

Struttura S3 JsonSource

Specifica un archivio dati JSON in Amazon S3.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del archivio dati.

- `Paths`: obbligatorio: una matrice di stringhe UTF-8.

  Un elenco dei percorsi Amazon S3 da cui leggere.

- `CompressionType`: stringa UTF-8 (valori validi: `gzip="GZIP"` | `bzip2="BZIP2"`).

  Specifica il modo in cui i dati sono compressi. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip"`).

- `Exclusions`: una matrice di stringhe UTF-8.

  Una stringa contenente un elenco di JSON di modelli glob in stile Unix da escludere. Ad esempio "[\"**.pdf \"]" esclude tutti i file PDF.

- `GroupSize`: stringa UTF-8, corrispondente a Custom string pattern #34.

  La dimensione del gruppo di destinazione in byte. Il valore di default viene calcolato in base alla dimensione dei dati di input e alle dimensioni del cluster. Quando sono presenti meno di 50.000 file di input, `"groupFiles"` deve essere impostato su `"inPartition"` per rendere effettiva la modifica.

- `GroupFiles`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Quando l'input contiene più di 50.000 file, il raggruppamento di file è attivato per impostazione predefinita. Per attivare il raggruppamento con meno di 50.000 file, imposta questo parametro su "inPartition". Per disabilitare il raggruppamento in presenza di più di 50.000 file, imposta il parametro su `"none"`.

- `Recurse`: booleano.

  Se è impostato su "vero", legge i file in modo ricorsivo in tutte le sottodirectory dei percorsi specificati.

- `MaxBand`: numero (intero), non superiore a Nessuno.

  Questa opzione controlla la durata in millisecondi dopo la quale è probabile che l'elenco s3 sia coerente. I file con timestamp di modifica che rientrano negli ultimi millisecondi MaxBand vengono tracciati appositamente quando vengono utilizzati per tenere conto JobBookmarks della coerenza

finale di Amazon S3. Per la maggior parte degli utenti non è necessario impostare questa opzione. Il valore di default è 900.000 millisecondi o 15 minuti.

- `MaxFilesInBand`: numero (intero), non superiore a Nessuno.

  Questa opzione specifica il numero massimo di file da salvare negli ultimi secondi maxBand. Se si supera questo valore, i file aggiuntivi vengono saltati e solo elaborati nella successiva esecuzione del processo.

- `AdditionalOptions`: un oggetto S3 DirectSourceAdditionalOptions.

  Specifica opzioni di connessione aggiuntive.

- `JsonPath`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Una stringa che definisce i dati JSON. JsonPath

- `Multiline`: booleano.

  Un valore booleano che specifica se un singolo registro può estendersi su più righe. Ciò può accadere quando un campo contiene un carattere di nuova riga tra virgolette. Imposta questa opzione su "Vero" se un qualsiasi registro si estende su più righe. Il valore di default è `False`, che consente una divisione dei file più netta durante l'analisi.

- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine JSON S3 personalizzata.

Struttura S3 ParquetSource

Specifica un archivio dati di Apache Parquet archiviato in Amazon S3.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del archivio dati.

- `Paths`: obbligatorio: una matrice di stringhe UTF-8.

  Un elenco dei percorsi Amazon S3 da cui leggere.

- `CompressionType`: stringa UTF-8 (valori validi: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

Specifica il modo in cui i dati sono compressi. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip"`).

- `Exclusions`: una matrice di stringhe UTF-8.

  Una stringa contenente un elenco di JSON di modelli glob in stile Unix da escludere. Ad esempio "[\"**.pdf \"]" esclude tutti i file PDF.

- `GroupSize`: stringa UTF-8, corrispondente a Custom string pattern #34.

  La dimensione del gruppo di destinazione in byte. Il valore di default viene calcolato in base alla dimensione dei dati di input e alle dimensioni del cluster. Quando sono presenti meno di 50.000 file di input, `"groupFiles"` deve essere impostato su `"inPartition"` per rendere effettiva la modifica.

- `GroupFiles`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Quando l'input contiene più di 50.000 file, il raggruppamento di file è attivato per impostazione predefinita. Per attivare il raggruppamento con meno di 50.000 file, imposta questo parametro su "inPartition". Per disabilitare il raggruppamento in presenza di più di 50.000 file, imposta il parametro su `"none"`.

- `Recurse`: booleano.

  Se è impostato su "vero", legge i file in modo ricorsivo in tutte le sottodirectory dei percorsi specificati.

- `MaxBand`: numero (intero), non superiore a Nessuno.

  Questa opzione controlla la durata in millisecondi dopo la quale è probabile che l'elenco s3 sia coerente. I file con timestamp di modifica che rientrano negli ultimi millisecondi MaxBand vengono tracciati appositamente quando vengono utilizzati per tenere conto JobBookmarks della coerenza finale di Amazon S3. Per la maggior parte degli utenti non è necessario impostare questa opzione. Il valore di default è 900.000 millisecondi o 15 minuti.

- `MaxFilesInBand`: numero (intero), non superiore a Nessuno.

  Questa opzione specifica il numero massimo di file da salvare negli ultimi secondi maxBand. Se si supera questo valore, i file aggiuntivi vengono saltati e solo elaborati nella successiva esecuzione del processo.

- `AdditionalOptions`: un oggetto S3 DirectSourceAdditionalOptions.

  Specifica opzioni di connessione aggiuntive.

- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine Parquet S3 personalizzata.

Struttura S3 DeltaSource

Specifica un'origine dati Delta Lake archiviata in. Amazon S3

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine del Delta Lake.
- `Paths`: obbligatorio: una matrice di stringhe UTF-8.

  Un elenco dei percorsi Amazon S3 da cui leggere.
- `AdditionalDeltaOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica opzioni di connessione aggiuntive.
- `AdditionalOptions`: un oggetto S3 DirectSourceAdditionalOptions.

  Specifica opzioni aggiuntive per il connettore.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine Delta Lake.

Struttura S3 CatalogDeltaSource

Speciifica un'origine dati Delta Lake registrata nel AWS Glue Data Catalog. L'origine dati deve essere archiviata in Amazon S3.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati Delta Lake.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.
- `AdditionalDeltaOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica opzioni di connessione aggiuntive.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine Delta Lake.

CatalogDeltaSource struttura

Specifica un'origine dati Delta Lake registrata nel AWS Glue Data Catalog.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati Delta Lake.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.
- `AdditionalDeltaOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica opzioni di connessione aggiuntive.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

Specifica lo schema di dati per l'origine Delta Lake.

Struttura S3 HudiSource

Specifica una fonte di dati Hudi memorizzata in. Amazon S3

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine Hudi.
- `Paths`: obbligatorio: una matrice di stringhe UTF-8.

  Un elenco dei percorsi Amazon S3 da cui leggere.
- `AdditionalHudiOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica opzioni di connessione aggiuntive.
- `AdditionalOptions`: un oggetto S3 DirectSourceAdditionalOptions.

  Specifica opzioni aggiuntive per il connettore.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine Hudi.

Struttura S3 CatalogHudiSource

Speciifica una fonte di dati Hudi registrata nel Data Catalog. AWS Glue L'origine dati Hudi deve essere archiviata in. Amazon S3

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati Hudi.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.
- `AdditionalHudiOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica opzioni di connessione aggiuntive.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per l'origine Hudi.

CatalogHudiSource struttura

Specifica un'origine dati Hudi registrata nel AWS Glue Data Catalog.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati Hudi.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.
- `AdditionalHudiOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica opzioni di connessione aggiuntive.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

Specifica lo schema di dati per l'origine Hudi.

Struttura DynamoDB CatalogSource

Specifica un'origine dati DynamoDB nel Data Catalog. AWS Glue

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.

RelationalCatalogSource struttura

Specifica un'origine dei dati del database relazionale nel Catalogo dati di AWS Glue .

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella nel database da cui leggere.

struttura JDBC ConnectorTarget

Specifica una destinazioni di dati che scrive su Amazon S3 nell'archiviazione colonnare di Apache Parquet.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.
- `ConnectionName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della connessione associata al connettore.
- `ConnectionTable`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #35.

  Il nome della tabella nella destinazione di dati.
- `ConnectorName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome di un connettore che verrà utilizzato.
- `ConnectionType`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il tipo di connessione, come marketplace.jdbc o custom.jdbc, che designa una connessione a una destinazione di dati JDBC.
- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Opzioni di connessione aggiuntive per il connettore.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema dati per la destinazione JDBC.

SparkConnectorTarget struttura

Specifica una destinazione che utilizza un connettore Apache Spark.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `ConnectionName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome di una connessione per un connettore Apache Spark.

- `ConnectorName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome di un connettore Apache Spark.

- `ConnectionType`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il tipo di connessione, come marketplace.spark o custom.spark, che designa una connessione a un archivio dati di Apache Spark.

- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Opzioni di connessione aggiuntive per il connettore.

- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per la destinazione Spark personalizzata.

BasicCatalogTarget struttura

Speciifica una destinazione che utilizza una tabella del catalogo AWS Glue dati.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome della destinazione di dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

Il database che contiene la tabella da utilizzare come destinazione. Questo database deve esistere già nel catalogo dati.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  La tabella che definisce lo schema dei dati di output. Questa tabella deve esistere già nel Data Catalog.

Struttura MySQL CatalogTarget

Specifica una destinazione che utilizza MySQL.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

Struttura PostgreSQL CatalogTarget

Specifica una destinazione che utilizza Postgres SQL.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

Struttura OracleSQL CatalogTarget

Specifica una destinazione che utilizza Oracle SQL.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

Struttura Microsoft SQL ServerCatalogTarget

Specifica una destinazione che utilizza Microsoft SQL.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

Il nome del database in cui scrivere.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

RedshiftTarget struttura

Specifica una destinazione che utilizza Amazon Redshift.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

- `RedshiftTmpDir`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il percorso Amazon S3 in cui i dati temporanei possono essere caricati durante la copia dal database.

- `TmpDirIAMRole`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il ruolo IAM con autorizzazioni.

- `UpsertRedshiftOptions`: un oggetto UpsertRedshiftTargetOptions.

  Il set di opzioni per configurare un'operazione di upsert durante la scrittura su una destinazione Redshift.

AmazonRedshiftTarget struttura

Specifica una destinazione Amazon Redshift.

Campi

- `Name`: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome della tabella Amazon Redshift.
- `Data`: un oggetto AmazonRedshiftNodeData.

  Specifica i dati del nodo di destinazione Amazon Redshift.
- `Inputs`: un array di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

UpsertRedshiftTargetOptions struttura

Le opzioni per configurare un'operazione di upsert durante la scrittura su una destinazione Redshift.

Campi

- `TableLocation`: stringa UTF-8, corrispondente a Custom string pattern #34.

  La posizione fisica della tabella Redshift.
- `ConnectionName`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della connessione da usare per scrivere su Redshift.
- `UpsertKeys`: una matrice di stringhe UTF-8.

  Le chiavi utilizzate per determinare se eseguire un aggiornamento o un inserimento.

struttura S3 CatalogTarget

Speciifica un target di dati che scrive su Amazon S3 utilizzando AWS Glue il Data Catalog.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.

- `SchemaChangePolicy`: un oggetto CatalogSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

Struttura S3 GlueParquetTarget

Specifica una destinazioni di dati che scrive su Amazon S3 nell'archiviazione colonnare di Apache Parquet.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.

- `Path`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Un singolo percorso Amazon S3 su cui scrivere.

- `Compression`: stringa UTF-8 (valori validi: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

Specifica il modo in cui i dati sono compressi. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip"`).

- `SchemaChangePolicy`: un oggetto DirectSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

CatalogSchemaChangePolicy struttura

Una policy che specifica i comportamenti di aggiornamento per il crawler.

Campi

- `EnableUpdateCatalog`: booleano.

  Stabilisce se usare il comportamento di aggiornamento quando il crawler riscontra una variazione dello schema.
- `UpdateBehavior`: stringa UTF-8 (valori validi: `UPDATE_IN_DATABASE` | `LOG`).

  Il comportamento di aggiornamento quando il crawler riscontra una variazione dello schema.

struttura S3 DirectTarget

Specifica una destinazione di dati che scrive su Amazon S3.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.
- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.
- `Path`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Un singolo percorso Amazon S3 su cui scrivere.

- `Compression`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il modo in cui i dati sono compressi. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip"`).

- `Format`: obbligatorio: stringa UTF-8 (valori validi: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Specifica il formato di output dei dati per la destinazione.

- `SchemaChangePolicy`: un oggetto DirectSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

Struttura S3 HudiCatalogTarget

Specifica una destinazione che scrive su un'origine dati Hudi nel Data Catalog. AWS Glue

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.

- `AdditionalOptions`: obbligatorio: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

Specifica le opzioni di connessione aggiuntive per il connettore.

- `SchemaChangePolicy`: un oggetto CatalogSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

Struttura S3 HudiDirectTarget

Speciifica una destinazione che scrive su una fonte di dati Hudi in. Amazon S3

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `Path`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il percorso Amazon S3 dell'origine dati Hudi su cui scrivere.

- `Compression`: obbligatorio: stringa UTF-8 (valori validi: `gzip="GZIP"` | `lzo="LZO"` | `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  Specifica il modo in cui i dati sono compressi. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip"`).

- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.

- `Format`: obbligatorio: stringa UTF-8 (valori validi: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Specifica il formato di output dei dati per la destinazione.

- `AdditionalOptions`: obbligatorio: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

Specifica le opzioni di connessione aggiuntive per il connettore.

- `SchemaChangePolicy`: un oggetto DirectSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

Struttura S3 DeltaCatalogTarget

Speciifica una destinazione che scrive su un'origine dati Delta Lake nel AWS Glue Data Catalog.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.

- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica le opzioni di connessione aggiuntive per il connettore.

- `SchemaChangePolicy`: un oggetto CatalogSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

Struttura S3 DeltaDirectTarget

Speciifica una destinazione che scrive su un'origine dati Delta Lake in. Amazon S3

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.

- `Path`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il percorso Amazon S3 dell'origine dati Delta Lake su cui scrivere.

- `Compression`: obbligatorio: stringa UTF-8 (valori validi: `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  Specifica il modo in cui i dati sono compressi. In genere questo non è necessario se i dati hanno un'estensione del file standard. I valori possibili sono `"gzip"` e `"bzip"`).

- `Format`: obbligatorio: stringa UTF-8 (valori validi: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Specifica il formato di output dei dati per la destinazione.

- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica le opzioni di connessione aggiuntive per il connettore.

- `SchemaChangePolicy`: un oggetto DirectSchemaChangePolicy.

  Una policy che specifica i comportamenti di aggiornamento per il crawler.

DirectSchemaChangePolicy struttura

Una policy che specifica i comportamenti di aggiornamento per il crawler.

Campi

- `EnableUpdateCatalog`: booleano.

  Stabilisce se usare il comportamento di aggiornamento quando il crawler riscontra una variazione dello schema.
- `UpdateBehavior`: stringa UTF-8 (valori validi: `UPDATE_IN_DATABASE` | `LOG`).

  Il comportamento di aggiornamento quando il crawler riscontra una variazione dello schema.
- `Table`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica la tabella nel database a cui si applica la policy di modifica dello schema.
- `Database`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il database a cui si applica la policy di modifica dello schema.

ApplyMapping struttura

Specifica una trasformazione che mappa le chiavi delle proprietà dei dati nell'origine dei dati alle chiavi delle proprietà dei dati nella destinazione. È possibile rinominare le chiavi, modificare i tipi di dati per le chiavi e scegliere le chiavi da eliminare dal set di dati.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.
- `Mapping`: obbligatorio: una matrice di oggetti Mapping.

  Specifica la mappatura delle chiavi delle proprietà dei dati nell'origine dei dati alle chiavi delle proprietà dei dati nella destinazione.

Struttura mappatura

Specifica la mappatura delle chiavi della proprietà dati.

Campi

- `ToKey`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Dopo l'applicazione della mappatura, quale dovrebbe essere il nome della colonna. Può coincidere con `FromPath`.

- `FromPath`: una matrice di stringhe UTF-8.

  La tabella o la colonna da modificare.

- `FromType`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il tipo di dati da modificare.

- `ToType`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Tipo di dati che devono essere modificati.

- `Dropped`: booleano.

  Se è true, la colonna viene rimossa.

- `Children`: una matrice di oggetti Mapping.

  Applicabile solo alle strutture dati nidificate. Se si desidera modificare la struttura padre, ma anche uno dei suoi figli, è possibile compilare questa struttura di dati. È anche `Mapping`, ma il suo `FromPath` sarà la struttura padre `FromPath` più il `FromPath` da questa struttura.

  Per la parte dei figli, supponiamo di avere la struttura:

```
{ "FromPath": "OuterStructure", "ToKey": "OuterStructure", "ToType":
"Struct", "Dropped": false, "Chidlren": [{ "FromPath": "inner", "ToKey":
"inner", "ToType": "Double", "Dropped": false, }] }
```

  Puoi specificare un `Mapping` con l'aspetto:

```
{ "FromPath": "OuterStructure", "ToKey": "OuterStructure", "ToType":
"Struct", "Dropped": false, "Chidlren": [{ "FromPath": "inner", "ToKey":
"inner", "ToType": "Double", "Dropped": false, }] }
```

SelectFields struttura

Specifica una trasformazione che sceglie le chiavi della proprietà dati che si desidera conservare.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.
- `Paths`: obbligatorio: una matrice di stringhe UTF-8.

  Un percorso JSON a una variabile nella struttura dati.

DropFields struttura

Specifica una trasformazione che sceglie le chiavi della proprietà dati che si desidera eliminare.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.
- `Paths`: obbligatorio: una matrice di stringhe UTF-8.

  Un percorso JSON a una variabile nella struttura dati.

RenameField struttura

Specifica una trasformazione che rinominerà una singola chiave di proprietà dati.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.

- `SourcePath`: obbligatorio: una matrice di stringhe UTF-8.

  Un percorso JSON a una variabile nella struttura dati per i dati di origine.

- `TargetPath`: obbligatorio: una matrice di stringhe UTF-8.

  Un percorso JSON a una variabile nella struttura dati per i dati di destinazione.

Struttura Spigot

Specifica una trasformazione che scrive campioni dei dati in un bucket Amazon S3.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.

- `Path`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Un percorso in Amazon S3 dove la trasformazione scriverà un sottoinsieme di registri dal set di dati in un file JSON in un bucket Amazon S3.

- `Topk`: numero (intero), non superiore a 100.

  Specifica un numero di registri da scrivere a partire dall'inizio del set di dati.

- `Prob`: numero (doppio), non superiore a 1.

  La probabilità (un valore decimale con un valore massimo di 1) di scegliere un determinato registro. Il valore 1 indica che ogni riga letta dal set di dati deve essere inclusa nell'output del campione.

Struttura join

Specifica una trasformazione che unisce due set di dati in un unico set di dati utilizzando una frase di confronto sulle chiavi di proprietà dei dati specificate. È possibile utilizzare inner, outer, left, right, left semi e left anti join.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore a o superiore a 2 stringhe.

  Gli input di dati identificati dai nomi dei nodi.

- `JoinType`: obbligatorio:: stringa UTF-8 (valori validi: `equijoin="EQUIJOIN"` | `left="LEFT"` | `right="RIGHT"` | `outer="OUTER"` | `leftsemi="LEFT_SEMI"` | `leftanti="LEFT_ANTI"`).

  Specifica il tipo di join da eseguire sui set di dati.

- `Columns`: obbligatorio: una matrice di oggetti JoinColumn, non inferiore a o superiore a 2 strutture.

  Un elenco delle due colonne da unire.

JoinColumn struttura

Specifica una colonna da unire.

Campi

- `From`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  La colonna da unire.

- `Keys`: obbligatorio: una matrice di stringhe UTF-8.

  La chiave della colonna da unire.

SplitFields struttura

Specifica una trasformazione che divide le chiavi della proprietà dati in due `DynamicFrames`. L'output è una raccolta di `DynamicFrames`: uno con le chiavi di proprietà dei dati selezionate e uno con le chiavi di proprietà dei dati rimanenti.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.
- `Paths`: obbligatorio: una matrice di stringhe UTF-8.

  Un percorso JSON a una variabile nella struttura dati.

SelectFromCollection struttura

Specifica una trasformazione che sceglie un `DynamicFrame` da una raccolta di `DynamicFrames`. L'output è il `DynamicFrame` selezionato

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.
- `Index`. Obbligatorio: numero (intero), non superiore a Nessuno.

  L'indice per il DynamicFrame da selezionare.

FillMissingValues struttura

Specifica una trasformazione che individua i registri nel set di dati che hanno valori mancanti e aggiunge un nuovo campo con un valore determinato dall'imputazione. Il set di dati di input viene

utilizzato per addestrare il modello di machine learning che determina quale dovrebbe essere il valore mancante.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.

- `ImputedPath`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Un percorso JSON a una variabile nella struttura dati per il set di dati imputato.

- `FilledPath`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Un percorso JSON a una variabile nella struttura dati per il set di dati compilato.

Struttura filtro

Specifica una trasformazione che divide un set di dati in due, in base a una condizione di filtro.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.

- `LogicalOperator`: obbligatorio: stringa UTF-8 (valori validi: `AND` | `OR`).

  L'operatore utilizzato per filtrare le righe confrontando il valore chiave con un valore specificato.

- `Filters`: obbligatorio: una matrice di oggetti FilterExpression.

  Specifica un'espressione di filtro.

FilterExpression struttura

Specifica un'espressione di filtro.

Campi

- `Operation`: obbligatorio: stringa UTF-8 (valori validi: `EQ` | `LT` | `GT` | `LTE` | `GTE` | `REGEX` | `ISNULL`).

  Tipo di operazione da eseguire nell'espressione.
- `Negated`: booleano.

  Se l'espressione deve essere negata.
- `Values`: obbligatorio: una matrice di oggetti FilterValue.

  Un elenco di valori di filtro.

FilterValue struttura

Rappresenta un'unica voce nell'elenco di valori di un `FilterExpression`.

Campi

- `Type`: obbligatorio: stringa UTF-8 (valori validi: `COLUMNEXTRACTED` | `CONSTANT`).

  Il tipo di valore del filtro.
- `Value`: obbligatorio: una matrice di stringhe UTF-8.

  Il valore da associare.

CustomCode struttura

Specifica una trasformazione che utilizza il codice personalizzato fornito per eseguire la trasformazione dei dati. L'output è una raccolta di DynamicFrames.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, almeno 1 stringa.

Gli input di dati identificati dai nomi dei nodi.

- `Code`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #29.

  Il codice personalizzato utilizzato per eseguire la trasformazione dei dati.

- `ClassName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome definito per la classe del nodo di codice personalizzato.

- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per la trasformazione del codice personalizzata.

Struttura SparkSQL

Specifica una trasformazione in cui si inserisce una query SQL utilizzando la sintassi Spark SQL per trasformare i dati. L'output è un singolo `DynamicFrame`.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, almeno 1 stringa.

  Gli input di dati identificati dai nomi dei nodi. È possibile associare un nome di tabella a ciascun nodo di input da utilizzare nella query SQL. Il nome scelto deve soddisfare le restrizioni sui nomi di Spark SQL.

- `SqlQuery`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #36.

  Query SQL che deve utilizzare la sintassi Spark SQL e restituire un singolo set di dati.

- `SqlAliases`: obbligatorio: una matrice di oggetti SqlAlias.

  Un elenco di alias. Un alias permette di specificare il nome da utilizzare nell'SQL per un determinato input. Ad esempio, hai una fonte di dati denominata "»MyDataSource. Se specifichi `From` as MyDataSource e `Alias` as SqlName, nel tuo SQL puoi fare:

  ```
  select * from SqlName
  ```

  e che ottiene dati da MyDataSource.

- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per la trasformazione SparkSQL.

SqlAlias struttura

Rappresenta un'unica voce nell'elenco di valori per `SqlAliases`.

Campi

- `From`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #33.

  Una tabella o una colonna in una tabella.
- `Alias`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #35.

  Un nome temporaneo dato a una tabella o a una colonna in una tabella.

DropNullFields struttura

Specifica una trasformazione che rimuove le colonne dal set di dati se tutti i valori nella colonna sono “null”. Per impostazione predefinita, AWS Glue Studio riconosce gli oggetti nulli, ma alcuni valori come stringhe vuote, stringhe «nulle», numeri interi -1 o altri segnaposto come zeri, non vengono riconosciuti automaticamente come nulli.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.
- `NullCheckBoxList`: un oggetto NullCheckBoxList.

  Struttura che indica se determinati valori siano riconosciuti come valori nulli per la rimozione.
- `NullTextList`: una matrice di oggetti NullValueField, non superiore a 50 strutture.

  Una struttura che specifica un elenco di NullValueField strutture che rappresentano un valore nullo personalizzato come zero o un altro valore utilizzato come segnaposto nullo unico per il set di dati.

La trasformazione `DropNullFields` rimuove i valori nulli personalizzati solo se sia il valore del segnaposto null che il tipo di dati corrispondono ai dati.

NullCheckBoxList struttura

Indica se alcuni valori siano riconosciuti come valori nulli per la rimozione.

Campi

- `IsEmpty`: booleano.

  Specifica che una stringa vuota è considerata un valore nullo.

- `IsNullString`: booleano.

  Specifica che un valore che indica la parola “null” è considerato un valore nullo.

- `IsNegOne`: booleano.

  Specifica che un valore intero di -1 è considerato un valore nullo.

NullValueField struttura

Rappresenta un valore nullo personalizzato, ad esempio uno zero o un altro valore utilizzato come segnaposto nullo univoco per il set di dati.

Campi

- `Value`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il valore del segnaposto nullo.

- `Datatype`: obbligatorio: un oggetto DataType.

  Il tipo di dati del valore.

Struttura Datatype

Struttura che rappresenta il tipo di dati del valore.

Campi

- `Id`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #33.

  Il tipo di dati del valore.
- `Label`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #33.

  Etichetta assegnata al tipo di dati.

Struttura Merge

Specifica una trasformazione che unisce `DynamicFrame` a con un `DynamicFrame` di staging basato sulle chiavi primarie specificate per identificare i registri. I registri duplicati (registri con le stesse chiavi primarie) non vengono deduplicati.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore a o superiore a 2 stringhe.

  Gli input di dati identificati dai nomi dei nodi.
- `Source`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #33.

  L'origine `DynamicFrame` che sarà unita a `DynamicFrame` di staging.
- `PrimaryKeys`: obbligatorio: una matrice di stringhe UTF-8.

  L'elenco dei campi chiave primaria per abbinare i registri dall'origine e dai frame dinamici di staging.

Struttura unione

Specifica una trasformazione che combina le righe di due o più set di dati in un unico risultato.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore a o superiore a 2 stringhe.

  L'ID del nodo immette la trasformazione.

- `UnionType`: obbligatorio: stringa UTF-8 (valori validi: `ALL` | `DISTINCT`).

  Indica il tipo di trasformazione Union.

  `ALL`Specificare di unire tutte le righe dalle fonti di dati a quelle risultanti DynamicFrame. L'unione risultante non rimuove le righe duplicate.

  `DISTINCT`Specificare di rimuovere le righe duplicate nel risultato DynamicFrame.

Struttura PIIDetection

Specifica una trasformazione che identifica, rimuove o maschera i dati PII.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  L'ID del nodo immette la trasformazione.

- `PiiType`: obbligatorio: stringa UTF-8 (valori validi: `RowAudit` | `RowMasking` | `ColumnAudit` | `ColumnMasking`).

  Indica il tipo di trasformazione PIIDetection.

- `EntityTypesToDetect`: obbligatorio: una matrice di stringhe UTF-8.

  Indica i tipi di entità che la trasformazione PIIDetection identificherà come dati PII.

  Le entità di tipo PII includono: PERSON_NAME, DATE, USA_SNN, EMAIL, USA_ITIN, USA_PASSPORT_NUMBER, PHONE_NUMBER, BANK_ACCOUNT, IP_ADDRESS, MAC_ADDRESS, USA_CPT_CODE, USA_HCPCS_CODE, USA_NATIONAL_DRUG_CODE, USA_MEDICARE_BENEFICIARY_IDENTIFIER, USA_HEALTH_INSURANCE_CLAIM_NUMBER,CREDIT_CARD,USA_NATIONAL_PROVIDER_IDENTIFIE

- `OutputColumnName`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Indica il nome della colonna di output che conterrà qualsiasi tipo di entità rilevato in quella riga.
- `SampleFraction`: numero (doppio), non superiore a 1.

  Indica la frazione dei dati da campionare durante la scansione di entità PII.
- `ThresholdFraction`: numero (doppio), non superiore a 1.

  Indica la frazione dei dati che devono essere soddisfatti per identificare una colonna come dati PII.
- `MaskValue`: stringa UTF-8, non superiore a 256 byte di lunghezza, corrispondente a Custom string pattern #31.

  Indica il valore che sostituirà l'entità rilevata.

Struttura aggregata

Specifica una trasformazione che raggruppa le righe in base ai campi scelti e calcola il valore aggregato in base alla funzione specificata.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Specifica i campi e le righe da utilizzare come input per la trasformazione aggregata.
- `Groups`: obbligatorio: una matrice di stringhe UTF-8.

  Specifica i campi in base ai quali raggruppare.
- `Aggs` – Obbligatorio: una matrice di oggetti AggregateOperation, non meno di 1 o più di 30 strutture.

  Specifica le funzioni di aggregazione da eseguire su campi specificati.

DropDuplicates struttura

Specifica una trasformazione che rimuove le righe di dati ripetuti da un set di dati.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di trasformazione.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input di dati identificati dai nomi dei nodi.

- `Columns`: una matrice di stringhe UTF-8.

  Il nome delle colonne da unire o rimuovere in caso di ripetizione.

GovernedCatalogTarget struttura

Speciifica un target di dati che scrive su Amazon S3 utilizzando AWS Glue il Data Catalog.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome di destinazione dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

- `PartitionKeys`: una matrice di stringhe UTF-8.

  Specifica il partizionamento nativo utilizzando una sequenza di chiavi.

- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome della tabella del database in cui scrivere.

- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il nome del database in cui scrivere.

- `SchemaChangePolicy`: un oggetto CatalogSchemaChangePolicy.

  Una policy che specifica il comportamento di aggiornamento per il catalogo governato.

GovernedCatalogSource struttura

Speciifica l'archivio dati nel AWS Glue Data Catalog governato.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del archivio dati.
- `Database`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il database da cui leggere.
- `Table`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  La tabella del database da cui leggere.
- `PartitionPredicate`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Le partizioni che soddisfano questo predicato vengono eliminate. I file all'interno del periodo di conservazione in queste partizioni non vengono eliminati. Impostato su `""`: vuoto per impostazione predefinita.
- `AdditionalOptions`: un oggetto S3 SourceAdditionalOptions.

  Specifica opzioni di connessione aggiuntive.

AggregateOperation struttura

Specifica il set di parametri necessari per eseguire l'aggregazione nella trasformazione di aggregazione.

Campi

- `Column`: obbligatorio: una matrice di stringhe UTF-8.

  Specifica la colonna sul set di dati su cui verrà applicata la funzione di aggregazione.
- `AggFunc` – Obbligatorio: stringa UTF-8 (valori validi:`avg` | `countDistinct` | `count` | `first` | `last` | `kurtosis` | `max` | `min` | `skewness` | `stddev_samp` | `stddev_pop` | `sum` | `sumDistinct` | `var_samp` | `var_pop`).

  Specifica la funzione di aggregazione da applicare.

Le possibili funzioni di aggregazione includono: avg countDistinct, count, first, last, kurtosis, max, min, skewness, stddev_samp, stddev_pop, sum, sumDistinct, var_samp, var_pop

GlueSchema struttura

Specifica uno schema definito dall'utente quando uno schema non può essere determinato da AWS Glue.

Campi

- `Columns`: una matrice di oggetti GlueStudioSchemaColumn.

  Specifica le definizioni delle colonne che compongono uno AWS Glue schema.

GlueStudioSchemaColumn struttura

Specifica una singola colonna in una definizione AWS Glue dello schema.

Campi

- `Name` – Obbligatorio: stringa UTF-8, non più lunga di 1024 byte, corrispondente al Single-line string pattern.

  Il nome della colonna nello schema di AWS Glue Studio.
- `Type`: stringa UTF-8, non superiore a 131072 byte di lunghezza, corrispondente a Single-line string pattern.

  Il tipo di hive per questa colonna nello schema di AWS Glue Studio.

GlueStudioColumn struttura

Specifica una singola colonna in AWS GlueStudio.

Campi

- `Key`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #35.

  La chiave della colonna in AWS Glue Studio.

- `FullPath`: obbligatorio: una matrice di stringhe UTF-8.

  L'URL completo della colonna in AWS Glue Studio.

- `Type` – Obbligatorio: stringa UTF-8 (valori validi: `array="ARRAY"` | `bigint="BIGINT"` | `bigint array="BIGINT_ARRAY"` | `binary="BINARY"` | `binary array="BINARY_ARRAY"` | `boolean="BOOLEAN"` | `boolean array="BOOLEAN_ARRAY"` | `byte="BYTE"` | `byte array="BYTE_ARRAY"` | `char="CHAR"` | `char array="CHAR_ARRAY"` | `choice="CHOICE"` | `choice array="CHOICE_ARRAY"` | `date="DATE"` | `date array="DATE_ARRAY"` | `decimal="DECIMAL"` | `decimal array="DECIMAL_ARRAY"` | `double="DOUBLE"` | `double array="DOUBLE_ARRAY"` | `enum="ENUM"` | `enum array="ENUM_ARRAY"` | `float="FLOAT"` | `float array="FLOAT_ARRAY"` | `int="INT"` | `int array="INT_ARRAY"` | `interval="INTERVAL"` | `interval array="INTERVAL_ARRAY"` | `long="LONG"` | `long array="LONG_ARRAY"` | `object="OBJECT"` | `short="SHORT"` | `short array="SHORT_ARRAY"` | `smallint="SMALLINT"` | `smallint array="SMALLINT_ARRAY"` | `string="STRING"` | `string array="STRING_ARRAY"` | `timestamp="TIMESTAMP"` | `timestamp array="TIMESTAMP_ARRAY"` | `tinyint="TINYINT"` | `tinyint array="TINYINT_ARRAY"` | `varchar="VARCHAR"` | `varchar array="VARCHAR_ARRAY"` | `null="NULL"` | `unknown="UNKNOWN"` | `unknown array="UNKNOWN_ARRAY"`).

  Il tipo di colonna in AWS Glue Studio.

- `Children`: un array di strutture.

  I figli della colonna principale in AWS Glue Studio.

DynamicTransform struttura

Specifica il set di parametri necessari per eseguire la trasformazione dinamica.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il nome della trasformazione dinamica.

- `TransformName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Speciifica il nome della trasformazione dinamica così come appare nell'editor visivo di AWS Glue Studio.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

Specifica gli input necessari per la trasformazione dinamica.

- `Parameters`: una matrice di oggetti TransformConfigParameter.

  Specifica i parametri della trasformazione dinamica.

- `FunctionName`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il nome della funzione della trasformazione dinamica.

- `Path`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il percorso dei file sorgente e di configurazione della trasformazione dinamica.

- `Version`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Questo campo non è utilizzato e verrà dichiarato obsoleto in una versione futura.

- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per la trasformazione dinamica.

TransformConfigParameter struttura

Specifica i parametri nel file di configurazione della trasformazione dinamica.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il nome del parametro nel file di configurazione della trasformazione dinamica.

- `Type`: obbligatorio: stringa UTF-8 (valori validi: `str="STR"` | `int="INT"` | `float="FLOAT"` | `complex="COMPLEX"` | `bool="BOOL"` | `list="LIST"` | `null="NULL"`).

  Specifica il tipo di parametro nel file di configurazione della trasformazione dinamica.

- `ValidationRule`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica la regola di convalida nel file di configurazione della trasformazione dinamica.

- `ValidationMessage`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Specifica il messaggio di convalida nel file di configurazione della trasformazione dinamica.

- `Value`: una matrice di stringhe UTF-8.

Specifica il valore del parametro nel file di configurazione della trasformazione dinamica.

- `ListType`: stringa UTF-8 (valori validi: `str="STR"` | `int="INT"` | `float="FLOAT"` | `complex="COMPLEX"` | `bool="BOOL"` | `list="LIST"` | `null="NULL"`).

  Specifica il tipo di elenco del parametro nel file di configurazione della trasformazione dinamica.

- `IsOptional`: booleano.

  Specifica se il parametro è facoltativo o meno nel file di configurazione della trasformazione dinamica.

EvaluateDataQuality struttura

Specifica i criteri di valutazione della qualità dei dati.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome della valutazione della qualità dei dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  Gli input della valutazione della qualità dei dati.

- `Ruleset`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 65.536 byte di lunghezza, corrispondente a Custom string pattern #32.

  Il set di regole per la valutazione della qualità dei dati.

- `Output`: stringa UTF-8 (valori validi: `PrimaryInput` | `EvaluationResults`).

  L'output della valutazione della qualità dei dati.

- `PublishingOptions`: un oggetto DQ ResultsPublishingOptions.

  Opzioni per configurare la modalità di pubblicazione dei risultati.

- `StopJobOnFailureOptions`: un oggetto DQ StopJobOnFailureOptions.

  Opzioni per configurare come si interromperà il processo se la valutazione della qualità dei dati fallisce.

struttura DQ ResultsPublishingOptions

Opzioni per configurare la modalità di pubblicazione dei risultati della valutazione della qualità dei dati.

Campi

- `EvaluationContext`: stringa UTF-8, corrispondente a Custom string pattern #33.

  Il contesto della valutazione.
- `ResultsS3Prefix`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Il prefisso Amazon S3 aggiunto all'inizio dei risultati.
- `CloudWatchMetricsEnabled`: booleano.

  Abilita i parametri per i risultati della qualità dei dati.
- `ResultsPublishingEnabled`: booleano.

  Abilita la pubblicazione per i risultati della qualità dei dati.

Struttura DQ StopJobOnFailureOptions

Opzioni per configurare come si interromperà il processo se la valutazione della qualità dei dati fallisce.

Campi

- `StopJobOnFailureTiming`: stringa UTF-8 (valori validi: `Immediate` | `AfterDataLoad`).

  Quando interrompere il processo se la valutazione della qualità dei dati fallisce. Le opzioni sono Immediate o AfterDataLoad.

EvaluateDataQualityMultiFrame struttura

Specifica i criteri di valutazione della qualità dei dati.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

Il nome della valutazione della qualità dei dati.

- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, almeno 1 stringa.

  Gli input della valutazione della qualità dei dati. Il primo input in questo elenco è l'origine dati primaria.

- `AdditionalDataSources`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #37.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Gli alias di tutte le origini dati, tranne quella primaria.

- `Ruleset`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 65.536 byte di lunghezza, corrispondente a Custom string pattern #32.

  Il set di regole per la valutazione della qualità dei dati.

- `PublishingOptions`: un oggetto DQ ResultsPublishingOptions.

  Opzioni per configurare la modalità di pubblicazione dei risultati.

- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 (valori validi: `performanceTuning.caching="CacheOption"` | `observations.scope="ObservationsOption"`).

  Ogni valore è una stringa UTF-8.

  Opzioni per configurare il comportamento di runtime della trasformazione.

- `StopJobOnFailureOptions`: un oggetto DQ StopJobOnFailureOptions.

  Opzioni per configurare come si interromperà il processo se la valutazione della qualità dei dati fallisce.

Struttura Recipe

Un nodo AWS Glue Studio che utilizza una AWS Glue DataBrew ricetta nei AWS Glue lavori.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo AWS Glue Studio.
- `Inputs`: obbligatorio: una matrice di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che costituiscono gli input del nodo della ricetta, identificati dal rispettivo ID.
- `RecipeReference`: obbligatorio: un oggetto RecipeReference.

  Un riferimento alla DataBrew ricetta usata dal nodo.

RecipeReference struttura

Un riferimento a una AWS Glue DataBrew ricetta.

Campi

- `RecipeArn`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  L'ARN della ricetta. DataBrew
- `RecipeVersion`: obbligatorio: stringa UTF-8, lunghezza non inferiore a 1 o non superiore a 16 byte.

  L' RecipeVersion origine della DataBrew ricetta.

SnowflakeNodeData struttura

Specifica la configurazione per i nodi Snowflake in Studio. AWS Glue

Campi

- `SourceType`: stringa UTF-8, corrispondente a Custom string pattern #33.

  Specifica come vengono specificati i dati recuperati. Valori validi: `"table"`, `"query"`.
- `Connection`: un oggetto Opzione.

  Specifica una connessione al catalogo AWS Glue dati a un endpoint Snowflake.
- `Schema`: stringa UTF-8.

Specifica uno schema di database Snowflake da utilizzare per il nodo.

- `Table`: stringa UTF-8.

  Specifica una tabella Snowflake da utilizzare per il nodo.

- `Database`: stringa UTF-8.

  Specifica un database Snowflake da utilizzare per il nodo.

- `TempDir`: stringa UTF-8, corrispondente a Custom string pattern #34.

  Attualmente non utilizzato.

- `IamRole`: un oggetto Opzione.

  Attualmente non utilizzato.

- `AdditionalOptions`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Ogni valore è una stringa UTF-8 corrispondente al Custom string pattern #34.

  Specifica le opzioni aggiuntive trasmesse al connettore Snowflake. Se altre opzioni sono specificate altrove in questo nodo, esse avranno la precedenza.

- `SampleQuery`: stringa UTF-8.

  Una stringa SQL utilizzata per recuperare i dati con il tipo di origine `query`.

- `PreAction`: stringa UTF-8.

  Una stringa SQL eseguita prima che il connettore Snowflake esegua le operazioni standard.

- `PostAction`: stringa UTF-8.

  Una stringa SQL eseguita dopo che il connettore Snowflake esegua le operazioni standard.

- `Action`: stringa UTF-8.

  Specifica l'operazione da intraprendere quando si scrive su una tabella con dati preesistenti. Valori validi: `append`, `merge`, `truncate`, `drop`.

- `Upsert`: booleano.

Utilizzato quando Operazione è `append`. Specifica il comportamento di risoluzione quando esiste già una riga. Se impostato su true, le righe preesistenti verranno aggiornate. Se false, verranno inserite quelle righe.

- `MergeAction`: stringa UTF-8, corrispondente a Custom string pattern #33.

  Specifica un'operazione di unione. Valori validi: `simple`, `custom`. Se semplice, il comportamento di unione è definito da `MergeWhenMatched` e `MergeWhenNotMatched`. Se personalizzato, definito da `MergeClause`.

- `MergeWhenMatched`: stringa UTF-8, corrispondente a Custom string pattern #33.

  Specifica come risolvere i record che corrispondono a dati preesistenti durante l'unione. Valori validi: `update`, `delete`.

- `MergeWhenNotMatched`: stringa UTF-8, corrispondente a Custom string pattern #33.

  Specifica come elaborare i record che non corrispondono a dati preesistenti durante l'unione. Valori validi: `insert`, `none`.

- `MergeClause`: stringa UTF-8.

  Un'istruzione SQL che specifica un comportamento di merge personalizzato.

- `StagingTable`: stringa UTF-8.

  Il nome di una tabella intermedia utilizzata durante le operazioni `merge` o `append` con upsert. I dati vengono scritti in questa tabella, quindi spostati in `table` da un'azione successiva (PostAction) generata.

- `SelectedColumns`: una matrice di oggetti Opzione.

  Specifica le colonne combinate per identificare un record quando vengono rilevate corrispondenze per i merge e gli upsert. Un elenco di strutture con chiavi `value`, `label` e `description`. Ogni struttura descrive una colonna.

- `AutoPushdown`: booleano.

  Specifica se il pushdown automatico delle query è abilitato. Se il pushdown è abilitato, quando su Spark viene eseguita una query, se una parte di essa può essere "trasferita" al server Snowflake, viene sottoposta a pushdown. Ciò migliora le prestazioni di alcune query.

- `TableSchema`: una matrice di oggetti Opzione.

Definisce manualmente lo schema di destinazione per il nodo. Un elenco di strutture con chiavi `value`, `label` e `description`. Ogni struttura definisce una colonna.

SnowflakeSource struttura

Specifica un'origine dati Snowflake.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome dell'origine dati Snowflake.
- `Data`: obbligatorio: un oggetto SnowflakeNodeData.

  Configurazione per l'origine dati Snowflake.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica gli schemi definiti dall'utente per i dati di output.

SnowflakeTarget struttura

Specifica una destinazione Snowflake.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome della destinazione Snowflake.
- `Data`: obbligatorio: un oggetto SnowflakeNodeData.

  Specifica i dati del nodo di destinazione Snowflake.
- `Inputs`: un array di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

ConnectorDataSource struttura

Specifica un'origine generata con opzioni di connessione standard.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di origine.
- `ConnectionType`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  IlconnectionType, come fornito alla AWS Glue libreria sottostante. Il tipo di nodo supporta i tipi di connessione seguenti:
  - `opensearch`
  - `azuresql`
  - `azurecosmos`
  - `bigquery`
  - `saphana`
  - `teradata`
  - `vertica`
- `Data`: obbligatorio: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Una mappa che specifica le opzioni di connessione per il nodo. È possibile trovare le opzioni di connessione standard per il tipo di connessione corrispondente nella sezione Parametri di connessione della AWS Glue documentazione.
- `OutputSchemas`: una matrice di oggetti GlueSchema.

  Specifica lo schema di dati per questa origine.

ConnectorDataTarget struttura

Specifica un a destinazione generata con opzioni di connessione standard.

Campi

- `Name`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #37.

  Il nome del nodo di destinazione.

- `ConnectionType`: obbligatorio: stringa UTF-8, corrispondente a Custom string pattern #34.

  IlconnectionType, come fornito alla AWS Glue libreria sottostante. Il tipo di nodo supporta i tipi di connessione seguenti:

  - `opensearch`
  - `azuresql`
  - `azurecosmos`
  - `bigquery`
  - `saphana`
  - `teradata`
  - `vertica`
- `Data`: obbligatorio: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Una mappa che specifica le opzioni di connessione per il nodo. È possibile trovare le opzioni di connessione standard per il tipo di connessione corrispondente nella sezione Parametri di connessione della AWS Glue documentazione.
- `Inputs`: un array di stringhe UTF-8, non inferiore o superiore a 1 stringa.

  I nodi che sono input per la destinazione di dati.

API dei processi

L'API dei processi descrive i tipi di dati dei processi e contiene API per gestire processi, esecuzioni di processi e trigger in AWS Glue.

Argomenti

- Processi
- Esecuzioni di processi
- Trigger

Processi

L'API Jobs descrive i tipi di dati e l'API relativi alla creazione, all'aggiornamento, all'eliminazione o alla visualizzazione di lavori in AWS Glue.

Tipi di dati

Struttura del processo

Specifica una definizione del processo.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome assegnato alla definizione del processo.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Descrizione del processo.
- `LogUri`: stringa UTF-8.

  Questo campo è riservato per uso futuro.
- `Role`: stringa UTF-8.

  Il nome o ARN (Amazon Resource Name) del ruolo IAM associato a questo processo.
- `CreatedOn`: timestamp.

  La data e l'ora in cui è stata creata la specifica della definizione del processo.

- `LastModifiedOn`: timestamp.

  L'ultimo point-in-time in cui è stata modificata la definizione del processo.

- `ExecutionProperty`: un oggetto ExecutionProperty.

  `ExecutionProperty` che specifica il numero massimo di esecuzioni simultanee consentite per il processo.

- `Command`: un oggetto JobCommand.

  Il `JobCommand` che esegue questo lavoro.

- `DefaultArguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Gli argomenti predefiniti per ciascuna esecuzione del processo, specificati come coppie nome-valore.

  Qui puoi specificare gli argomenti utilizzati dal tuo script di esecuzione del lavoro, nonché gli argomenti che utilizza da solo. AWS Glue

  Gli argomenti del processo potrebbero essere registrati. Non passare segreti in testo chiaro come argomenti. Recupera i segreti da una AWS Glue connessione AWS Secrets Manager o da un altro meccanismo di gestione dei segreti se intendi mantenerli all'interno del Job.

  Per informazioni su come specificare e utilizzare gli argomenti del proprio processo, fai riferimento a Chiamare le API AWS Glue in Python nella guida per gli sviluppatori.

  Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Spark, consulta la pagina Special Parameters Used by AWS Glue nella Guida per gli sviluppatori.

  Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Ray, consulta la pagina Using job parameters in Ray jobs nella Guida per gli sviluppatori.

- `NonOverridableArguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

Gli argomenti per questo processo che non vengono sovrascritti quando si forniscono argomenti di processo in un'esecuzione di processo, specificati come coppie nome-valore.

- `Connections`: un oggetto ConnectionsList.

  Le connessioni utilizzate per questo processo.

- `MaxRetries`: numero (intero).

  Il numero massimo di volte in cui riprovare questo lavoro dopo un JobRun errore.

- `AllocatedCapacity`: numero (intero).

  in quanto obsoleto. Usare invece `MaxCapacity`.

  Il numero di unità di elaborazione AWS Glue dati (DPU) assegnate alle esecuzioni di questo processo. È possibile allocare un minimo di 2 DPU; l'impostazione di default è 10. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

- `Timeout`: numero (intero), almeno 1.

  Timeout del processo in minuti. Indica il tempo massimo durante cui l'esecuzione di un processo può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).

- `MaxCapacity`: numero (doppio).

  Per i lavori di Glue versione 1.0 o precedente, utilizzando il tipo di worker standard, il numero di unità di elaborazione AWS Glue dati (DPU) che possono essere allocate durante l'esecuzione di questo lavoro. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  Per i processi Glue versione 2.0 e successive, non è possibile specificare il valore `Maximum capacity`. Si deve invece specificare un `Worker type` e un `Number of workers`.

  Non impostare `MaxCapacity` se usi `WorkerType` e `NumberOfWorkers`.

  Il valore che è possibile allocare per `MaxCapacity` varia a seconda che si esegua un processo shell di Python, un processo ETL di Apache Spark o un processo ETL di streaming di Apache Spark:

- Quando si specifica un processo shell di Python (`JobCommand.Name`="pythonshell"), è possibile allocare 0,0625 o 1 DPU. Il valore di default è 0,0625 DPU.
- Quando si specifica un processo ETL Apache Spark (`JobCommand.Name`="glueetl") o un processo ETL di streaming Apache Spark (`JobCommand.Name`="gluestreaming"), è possibile allocare da 2 a 100 DPU. Il valore di default è 10 DPU. Questo tipo di processo non può avere un'allocazione DPU frazionata.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard`="" | `G.1X`="" | `G.2X`="" | `G.025X`="" | `G.4X`="" | `G.8X`="" | `Z.2X`="").

  Il tipo di worker predefinito allocato quando viene eseguito un processo. Accetta un valore di G.1X, G.2X, G.4X, G.8X o G.025X per i processi Spark. Accetta il valore Z.2X per i processi Ray.

  - Per il tipo di worker `G.1X`, ciascun worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.2X`, ciascun worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria) con disco da 128 GB (circa 77 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.4X`, ciascun worker esegue la mappatura su 4 DPU (16 vCPU, 64 GB di memoria) con disco da 256 GB (circa 235 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di lavoratore è disponibile solo per i lavori Spark ETL AWS Glue versione 3.0 o successiva AWS nelle seguenti regioni: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda) ed Europa (Stoccolma).
  - Per il tipo di worker `G.8X`, ciascun worker esegue la mappatura su 8 DPU (32 vCPU, 128 GB di memoria) con disco da 512 GB (circa 487 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i job Spark ETL AWS Glue versione 3.0 o successiva, nelle stesse AWS regioni supportate per il tipo di `G.4X` lavoratore.
  - Per il tipo di worker `G.025X`, ciascun worker esegue la mappatura su 0,25 DPU (2 vCPU, 4 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Consigliamo questo

tipo di worker per i processi di streaming a basso volume. Questo tipo di worker è disponibile solo per i lavori di streaming della AWS Glue versione 3.0.

- Per il tipo di worker `Z.2X`, ciascun worker esegue la mappatura su 2 M-DPU (8 vCPU, 64 GB di memoria) con disco da 128 GB (circa 120 GB liberi) e fornisce un massimo di 8 worker Ray in base all'autoscaler.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` specifico allocati quando viene eseguito un processo.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da usare con questo processo.

- `NotificationProperty`: un oggetto NotificationProperty.

  Specifica le proprietà di configurazione di una notifica di processo.

- `Running`: booleano.

  Questo campo è riservato per uso futuro.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Nei job Spark, `GlueVersion` determina le versioni di Apache Spark e Python disponibili in un job. AWS Glue La versione Python indica la versione supportata per i processi di tipo Spark.

  I processi Ray devono impostare il valore di `GlueVersion` su `4.0` o superiore. Tuttavia, le versioni di Ray, Python e le librerie aggiuntive disponibili nel processo Ray sono determinate dal parametro `Runtime` del comando del processo.

  Per ulteriori informazioni sulle AWS Glue versioni disponibili e sulle versioni corrispondenti di Spark e Python, consulta la versione Glue nella guida per sviluppatori.

  Processi creati senza specificare una versione Glue utilizzano Glue 0.9 per impostazione predefinita.

- `CodeGenConfigurationNodes`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #33.

  Ogni valore è un oggetto CodeGenConfigurationNode.

La rappresentazione di un grafico aciclico diretto su cui si basano sia il componente visivo che la generazione di codice di Glue Studio.

- `ExecutionClass`: una stringa UTF-8, non superiore a 16 byte di lunghezza (valori validi: `FLEX=""` | `STANDARD=""`).

  Indica se il processo viene eseguito con una classe di esecuzione standard o flessibile. La classe di esecuzione standard è ideale per carichi di lavoro sensibili al tempo che richiedono un avvio rapido dei processi e risorse dedicate.

  La classe di esecuzione flessibile è appropriata per i processi non sensibili al tempo i cui tempi di inizio e completamento possono variare.

  Potranno essere `ExecutionClass` impostati solo i lavori con la AWS Glue versione 3.0 e successive e `glueetl` il tipo di comando. `FLEX` La classe di esecuzione flessibile è disponibile per i processi Spark.

- `SourceControlDetails`: un oggetto SourceControlDetails.

  I dettagli per una configurazione di controllo di origine per un processo, che consente la sincronizzazione degli artefatti del processo da o verso un repository remoto.

- `ProfileName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di un profilo di AWS Glue utilizzo associato al job.

ExecutionProperty struttura

Una proprietà di esecuzione di un processo.

Campi

- `MaxConcurrentRuns`: numero (intero).

  Il numero massimo di esecuzioni simultanee consentite per il processo. Il valore di default è 1. Viene restituito un errore al raggiungimento della soglia. Il valore massimo che è possibile specificare è controllato da un limite di servizio.

NotificationProperty struttura

Specifica le proprietà di configurazione di una notifica.

Campi

- `NotifyDelayAfter`: numero (intero), almeno 1.

  Dopo l'inizio dell'esecuzione di un processo, la quantità di minuti da attendere prima di inviare una notifica di ritardo dell'esecuzione di un processo.

JobCommand struttura

Specifica il codice eseguito quando viene eseguito un processo.

Campi

- `Name`: stringa UTF-8.

  Il nome del comando del processo. Per un processo ETL Apache Spark, deve essere `glueetl`. Per un processo shell Python, deve essere `pythonshell`. Per un processo ETL di streaming Apache Spark, deve essere `gluestreaming`. Per un processo Ray, questo deve essere `glueray`.
- `ScriptLocation`: stringa UTF-8, non superiore a 400000 byte di lunghezza.

  Specifica il percorso di Amazon Simple Storage Service (Amazon S3) per uno script che esegue un processo.
- `PythonVersion`: stringa UTF-8, corrispondente a Custom string pattern #16.

  La versione Python utilizzata per eseguire un processo shell Python. I valori consentiti sono 2 o 3.
- `Runtime`: stringa UTF-8, non superiore a 64 byte di lunghezza, corrispondente a Custom string pattern #24.

  Nei processi Ray, Runtime viene utilizzato per specificare le versioni di Ray, Python e librerie aggiuntive disponibili nell'ambiente. Questo campo non viene utilizzato in altri tipi di processo. Per i valori dell'ambiente di runtime supportati, consultate Supported Ray runtime Environments nella AWS Glue Developer Guide.

ConnectionsList struttura

Specifica le connessioni utilizzate da un processo.

Campi

- `Connections`: una matrice di stringhe UTF-8.

  Un elenco di connessioni utilizzate dal processo.

JobUpdate struttura

Specifica le informazioni utilizzate per aggiornare una definizione del processo esistente. La precedente definizione di processo viene completamente sovrascritta da questa informazione.

Campi

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Descrizione del processo da definire.
- `LogUri`: stringa UTF-8.

  Questo campo è riservato per uso futuro.
- `Role`: stringa UTF-8.

  Il nome o ARN (Amazon Resource Name) del ruolo IAM associato a questo processo (richiesto).
- `ExecutionProperty`: un oggetto ExecutionProperty.

  `ExecutionProperty` che specifica il numero massimo di esecuzioni simultanee consentite per il processo.
- `Command`: un oggetto JobCommand.

  `JobCommand` che esegue il processo (richiesto).
- `DefaultArguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

Gli argomenti predefiniti per ciascuna esecuzione del processo, specificati come coppie nome-valore.

Qui è possibile specificare gli argomenti utilizzati dal proprio script di esecuzione del lavoro, nonché gli argomenti utilizzati a sua volta. AWS Glue

Gli argomenti del processo potrebbero essere registrati. Non passare segreti in testo chiaro come argomenti. Recupera i segreti da una AWS Glue connessione AWS Secrets Manager o da un altro meccanismo di gestione dei segreti se intendi mantenerli all'interno del Job.

Per informazioni su come specificare e utilizzare gli argomenti del proprio processo, fai riferimento a Chiamare le API AWS Glue in Python nella guida per gli sviluppatori.

Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Spark, consulta la pagina Special Parameters Used by AWS Glue nella Guida per gli sviluppatori.

Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Ray, consulta la pagina Using job parameters in Ray jobs nella Guida per gli sviluppatori.

- `NonOverridableArguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Gli argomenti per questo processo che non vengono sovrascritti quando si forniscono argomenti di processo in un'esecuzione di processo, specificati come coppie nome-valore.

- `Connections`: un oggetto ConnectionsList.

  Le connessioni utilizzate per questo processo.

- `MaxRetries`: numero (intero).

  Il numero massimo di tentativi per riprovare il processo se ha esito negativo.

- `AllocatedCapacity`: numero (intero).

  in quanto obsoleto. Usare invece `MaxCapacity`.

  Il numero di unità di elaborazione AWS Glue dati (DPU) da allocare a questo lavoro. È possibile allocare un minimo di 2 DPU; l'impostazione di default è 10. Una DPU è una misura relativa della

potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

- `Timeout`: numero (intero), almeno 1.

  Timeout del processo in minuti. Indica il tempo massimo durante cui l'esecuzione di un processo può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).

- `MaxCapacity`: numero (doppio).

  Per i lavori di Glue versione 1.0 o precedente, utilizzando il tipo di worker standard, il numero di unità di elaborazione AWS Glue dati (DPU) che possono essere allocate durante l'esecuzione di questo lavoro. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  Per i processi Glue versione 2.0 e successive, non è possibile specificare il valore `Maximum capacity`. Si deve invece specificare un `Worker type` e un `Number of workers`.

  Non impostare `MaxCapacity` se usi `WorkerType` e `NumberOfWorkers`.

  Il valore che è possibile allocare per `MaxCapacity` varia a seconda che si esegua un processo shell di Python, un processo ETL di Apache Spark o un processo ETL di streaming di Apache Spark:

  - Quando si specifica un processo shell di Python (`JobCommand.Name`="pythonshell"), è possibile allocare 0,0625 o 1 DPU. Il valore di default è 0,0625 DPU.
  - Quando si specifica un processo ETL Apache Spark (`JobCommand.Name`="glueetl”) o un processo ETL di streaming Apache Spark (`JobCommand.Name`="gluestreaming”), è possibile allocare da 2 a 100 DPU. Il valore di default è 10 DPU. Questo tipo di processo non può avere un'allocazione DPU frazionata.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato quando viene eseguito un processo. Accetta un valore di G.1X, G.2X, G.4X, G.8X o G.025X per i processi Spark. Accetta il valore Z.2X per i processi Ray.

  - Per il tipo di worker `G.1X`, ciascun worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Questi tipi di worker

sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.

- Per il tipo di worker `G.2X`, ciascun worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria) con disco da 128 GB (circa 77 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
- Per il tipo di worker `G.4X`, ciascun worker esegue la mappatura su 4 DPU (16 vCPU, 64 GB di memoria) con disco da 256 GB (circa 235 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di lavoratore è disponibile solo per i lavori Spark ETL AWS Glue versione 3.0 o successiva AWS nelle seguenti regioni: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda) ed Europa (Stoccolma).
- Per il tipo di worker `G.8X`, ciascun worker esegue la mappatura su 8 DPU (32 vCPU, 128 GB di memoria) con disco da 512 GB (circa 487 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i job Spark ETL AWS Glue versione 3.0 o successiva, nelle stesse AWS regioni supportate per il tipo di `G.4X` lavoratore.
- Per il tipo di worker `G.025X`, ciascun worker esegue la mappatura su 0,25 DPU (2 vCPU, 4 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Consigliamo questo tipo di worker per i processi di streaming a basso volume. Questo tipo di worker è disponibile solo per i lavori di streaming della AWS Glue versione 3.0.
- Per il tipo di worker `Z.2X`, ciascun worker esegue la mappatura su 2 M-DPU (8 vCPU, 64 GB di memoria) con disco da 128 GB (circa 120 GB liberi) e fornisce un massimo di 8 worker Ray in base all'autoscaler.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` specifico allocati quando viene eseguito un processo.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da usare con questo processo.

- `NotificationProperty`: un oggetto NotificationProperty.

Specifica le proprietà di configurazione di una notifica di un processo.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Nei job Spark, `GlueVersion` determina le versioni di Apache Spark e Python disponibili in un job. AWS Glue La versione Python indica la versione supportata per i processi di tipo Spark.

  I processi Ray devono impostare il valore di `GlueVersion` su `4.0` o superiore. Tuttavia, le versioni di Ray, Python e le librerie aggiuntive disponibili nel processo Ray sono determinate dal parametro `Runtime` del comando del processo.

  Per ulteriori informazioni sulle AWS Glue versioni disponibili e sulle versioni corrispondenti di Spark e Python, consulta la versione Glue nella guida per sviluppatori.

  Processi creati senza specificare una versione Glue utilizzano Glue 0.9 per impostazione predefinita.

- `CodeGenConfigurationNodes`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #33.

  Ogni valore è un oggetto CodeGenConfigurationNode.

  La rappresentazione di un grafico aciclico diretto su cui si basano sia il componente visivo che la generazione di codice di Glue Studio.

- `ExecutionClass`: una stringa UTF-8, non superiore a 16 byte di lunghezza (valori validi: `FLEX=""` | `STANDARD=""`).

  Indica se il processo viene eseguito con una classe di esecuzione standard o flessibile. La classe di esecuzione standard è ideale per carichi di lavoro sensibili al tempo che richiedono un avvio rapido dei processi e risorse dedicate.

  La classe di esecuzione flessibile è appropriata per i processi non sensibili al tempo i cui tempi di inizio e completamento possono variare.

  Potranno essere `ExecutionClass` impostati solo i lavori con la AWS Glue versione 3.0 e successive e `glueetl` il tipo di comando. `FLEX` La classe di esecuzione flessibile è disponibile per i processi Spark.

- `SourceControlDetails`: un oggetto SourceControlDetails.

I dettagli per una configurazione di controllo di origine per un processo, che consente la sincronizzazione degli artefatti del processo da o verso un repository remoto.

- `ProfileName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di un profilo di AWS Glue utilizzo associato al job.

SourceControlDetails struttura

I dettagli per una configurazione di controllo di origine per un processo, che consente la sincronizzazione degli artefatti del processo da o verso un repository remoto.

Campi

- `Provider`: stringa UTF-8.

  Il provider per il repository remoto.

- `Repository`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  Il nome del repository remoto che contiene gli artefatti del processo.

- `Owner`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  Il proprietario del repository remoto che contiene gli artefatti del processo.

- `Branch`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  Un ramo opzionale nel repository remoto.

- `Folder`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  Una cartella opzionale nel repository remoto.

- `LastCommitId`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  L'ultimo ID di commit per un commit nel repository remoto.

- `LastSyncTimestamp`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  La data e l'ora in cui è stata eseguita l'ultima sincronizzazione di processo.

- `AuthStrategy`: stringa UTF-8.

Il tipo di autenticazione, che può essere un token di autenticazione memorizzato in AWS Secrets Manager o un token di accesso personale.

- `AuthToken`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  Il valore di un token di autorizzazione.

Operazioni

- CreateJob azione (Python: create_job)
- UpdateJob azione (Python: update_job)
- GetJob azione (Python: get_job)
- GetJobs azione (Python: get_jobs)
- DeleteJob azione (Python: delete_job)
- ListJobs azione (Python: list_jobs)
- BatchGetJobs azione (Python: batch_get_jobs)
- UpdateSourceControlFromJob azione (Python: update_source_control_from_job)
- UpdateJobFromSourceControl azione (Python: update_job_from_source_control)

CreateJob azione (Python: create_job)

Crea una nuova definizione del processo.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome assegnato alla definizione del processo. Deve essere univoco all'interno dell'account .
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Descrizione del processo da definire.
- `LogUri`: stringa UTF-8.

  Questo campo è riservato per uso futuro.

- `Role`. Obbligatorio: stringa UTF-8.

  Il nome o ARN (Amazon Resource Name) del ruolo IAM associato a questo processo.

- `ExecutionProperty`: un oggetto ExecutionProperty.

  `ExecutionProperty` che specifica il numero massimo di esecuzioni simultanee consentite per il processo.

- `Command`: obbligatorio: un oggetto JobCommand.

  Il `JobCommand` che esegue questo lavoro.

- `DefaultArguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Gli argomenti predefiniti per ciascuna esecuzione del processo, specificati come coppie nome-valore.

  Qui puoi specificare gli argomenti utilizzati dal tuo script di esecuzione del lavoro, nonché gli argomenti che utilizza esso stesso. AWS Glue

  Gli argomenti del processo potrebbero essere registrati. Non passare segreti in testo chiaro come argomenti. Recupera i segreti da una AWS Glue connessione AWS Secrets Manager o da un altro meccanismo di gestione dei segreti se intendi mantenerli all'interno del Job.

  Per informazioni su come specificare e utilizzare gli argomenti del proprio processo, fai riferimento a Chiamare le API AWS Glue in Python nella guida per gli sviluppatori.

  Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Spark, consulta la pagina Special Parameters Used by AWS Glue nella Guida per gli sviluppatori.

  Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Ray, consulta la pagina Using job parameters in Ray jobs nella Guida per gli sviluppatori.

- `NonOverridableArguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

Gli argomenti per questo processo che non vengono sovrascritti quando si forniscono argomenti di processo in un'esecuzione di processo, specificati come coppie nome-valore.

- `Connections`: un oggetto ConnectionsList.

  Le connessioni utilizzate per questo processo.

- `MaxRetries`: numero (intero).

  Il numero massimo di tentativi per riprovare il processo se ha esito negativo.

- `AllocatedCapacity`: numero (intero).

  Questo parametro è obsoleto. Usare invece `MaxCapacity`.

  Il numero di unità di elaborazione AWS Glue dati (DPU) da allocare a questo Job. È possibile allocare un minimo di 2 DPU; l'impostazione di default è 10. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

- `Timeout`: numero (intero), almeno 1.

  Timeout del processo in minuti. Indica il tempo massimo durante cui l'esecuzione di un processo può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).

- `MaxCapacity`: numero (doppio).

  Per i lavori di Glue versione 1.0 o precedente, utilizzando il tipo di worker standard, il numero di unità di elaborazione AWS Glue dati (DPU) che possono essere allocate durante l'esecuzione di questo lavoro. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  Per i processi Glue versione 2.0 e successive, non è possibile specificare il valore `Maximum capacity`. Si deve invece specificare un `Worker type` e un `Number of workers`.

  Non impostare `MaxCapacity` se usi `WorkerType` e `NumberOfWorkers`.

  Il valore che è possibile allocare per `MaxCapacity` varia a seconda che si esegua un processo shell di Python, un processo ETL di Apache Spark o un processo ETL di streaming di Apache Spark:

  - Quando si specifica un processo shell di Python (`JobCommand.Name`="pythonshell"), è possibile allocare 0,0625 o 1 DPU. Il valore di default è 0,0625 DPU.
  - Quando si specifica un processo ETL Apache Spark (`JobCommand.Name`="glueetl") o un processo ETL di streaming Apache Spark (`JobCommand.Name`="gluestreaming"), è possibile allocare da 2 a 100 DPU. Il valore di default è 10 DPU. Questo tipo di processo non può avere un'allocazione DPU frazionata.
- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da usare con questo processo.
- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag da usare con questo processo. Puoi usare i tag per limitare l'accesso al processo. Per ulteriori informazioni sui tag in AWS Glue, consulta AWS Tags in AWS Glue nella guida per sviluppatori.
- `NotificationProperty`: un oggetto NotificationProperty.

  Specifica le proprietà di configurazione di una notifica di processo.
- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Nei job Spark, `GlueVersion` determina le versioni di Apache Spark e Python disponibili in un job. AWS Glue La versione Python indica la versione supportata per i processi di tipo Spark.

  I processi Ray devono impostare il valore di `GlueVersion` su `4.0` o superiore. Tuttavia, le versioni di Ray, Python e le librerie aggiuntive disponibili nel processo Ray sono determinate dal parametro `Runtime` del comando del processo.

  Per ulteriori informazioni sulle AWS Glue versioni disponibili e sulle versioni corrispondenti di Spark e Python, consulta la versione Glue nella guida per sviluppatori.

  Processi creati senza specificare una versione Glue utilizzano Glue 0.9 per impostazione predefinita.
- `NumberOfWorkers`: numero (intero).

Il numero di worker di un `workerType` specifico allocati quando viene eseguito un processo.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato quando viene eseguito un processo. Accetta un valore di G.1X, G.2X, G.4X, G.8X o G.025X per i processi Spark. Accetta il valore Z.2X per i processi Ray.

  - Per il tipo di worker `G.1X`, ciascun worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.2X`, ciascun worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria) con disco da 128 GB (circa 77 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.4X`, ciascun worker esegue la mappatura su 4 DPU (16 vCPU, 64 GB di memoria) con disco da 256 GB (circa 235 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di lavoratore è disponibile solo per i lavori Spark ETL AWS Glue versione 3.0 o successiva AWS nelle seguenti regioni: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda) ed Europa (Stoccolma).
  - Per il tipo di worker `G.8X`, ciascun worker esegue la mappatura su 8 DPU (32 vCPU, 128 GB di memoria) con disco da 512 GB (circa 487 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i job Spark ETL AWS Glue versione 3.0 o successiva, nelle stesse AWS regioni supportate per il tipo di `G.4X` lavoratore.
  - Per il tipo di worker `G.025X`, ciascun worker esegue la mappatura su 0,25 DPU (2 vCPU, 4 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Consigliamo questo tipo di worker per i processi di streaming a basso volume. Questo tipo di worker è disponibile solo per i lavori di streaming della AWS Glue versione 3.0.
  - Per il tipo di worker `Z.2X`, ciascun worker esegue la mappatura su 2 M-DPU (8 vCPU, 64 GB di memoria) con disco da 128 GB (circa 120 GB liberi) e fornisce un massimo di 8 worker Ray in base all'autoscaler.

- `CodeGenConfigurationNodes`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8 corrispondente al Custom string pattern #33.

  Ogni valore è un oggetto CodeGenConfigurationNode.

  La rappresentazione di un grafico aciclico diretto su cui si basano sia il componente visivo che la generazione di codice di Glue Studio.

- `ExecutionClass`: una stringa UTF-8, non superiore a 16 byte di lunghezza (valori validi: `FLEX=""` | `STANDARD=""`).

  Indica se il processo viene eseguito con una classe di esecuzione standard o flessibile. La classe di esecuzione standard è ideale per carichi di lavoro sensibili al tempo che richiedono un avvio rapido dei processi e risorse dedicate.

  La classe di esecuzione flessibile è appropriata per i processi non sensibili al tempo i cui tempi di inizio e completamento possono variare.

  Solo i lavori con AWS Glue versione 3.0 e successive e il tipo di comando `glueetl` potranno essere `ExecutionClass` impostati su`FLEX`. La classe di esecuzione flessibile è disponibile per i processi Spark.

- `SourceControlDetails`: un oggetto SourceControlDetails.

  I dettagli per una configurazione di controllo di origine per un processo, che consente la sincronizzazione degli artefatti del processo da o verso un repository remoto.

- `ProfileName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di un profilo di AWS Glue utilizzo associato al job.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome univoco assegnato alla definizione del processo.

Errori

- `InvalidInputException`
- `IdempotentParameterMismatchException`
- `AlreadyExistsException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

UpdateJob azione (Python: update_job)

Aggiorna la definizione di un processo esistente. La precedente definizione di processo viene completamente sovrascritta da questa informazione.

Richiesta

- `JobName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione del processo da aggiornare.
- `JobUpdate`: obbligatorio: un oggetto JobUpdate.

  Specifica i valori con cui aggiornare la definizione del processo. La configurazione non specificata viene rimossa o ripristinata ai valori predefiniti.
- `ProfileName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di un profilo di AWS Glue utilizzo associato al lavoro.

Risposta

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Restituisce il nome della definizione aggiornata del processo.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

GetJob azione (Python: get_job)

Recupera la definizione di un processo esistente.

Richiesta

- `JobName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione del processo da recuperare.

Risposta

- `Job`: un oggetto Processo.

  La definizione del processo richiesta.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobs azione (Python: get_jobs)

Recupera tutte le attuali definizioni del processo.

Richiesta

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima della risposta.

Risposta

- `Jobs`: una matrice di oggetti Processo.

  Un elenco di definizioni del processo.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non sono ancora state restituite tutte le definizioni del processo.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

DeleteJob azione (Python: delete_job)

Elimina una specifica definizione del processo. Se la definizione del processo non viene trovata, non viene generata alcuna eccezione.

Richiesta

- `JobName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione del processo da eliminare.

Risposta

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della definizione del processo eliminata.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

ListJobs azione (Python: list_jobs)

Recupera i nomi di tutte le risorse lavorative in questo AWS account o le risorse con il tag specificato. Questa operazione consente di vedere quali risorse sono disponibili nel proprio account e i relativi nomi.

L'operazione accetta il campo facoltativo `Tags` che si può utilizzare come filtro per la risposta in modo che le risorse con tag possano essere recuperate come gruppo. Se si sceglie di utilizzare il filtro dei tag, potranno essere recuperate solo le risorse con tag.

Richiesta

- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima di un elenco da restituire.
- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Specifica che vengono restituite solo le risorse con tag.

Risposta

- `JobNames`: una matrice di stringhe UTF-8.

  I nomi di tutti i processi nell'account oppure i processi con i tag specificati.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se l'elenco restituito non contiene l'ultimo parametro disponibile.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchGetJobs azione (Python: batch_get_jobs)

Restituisce un elenco di metadati di risorse per un determinato elenco di nomi di processi. Dopo aver chiamato l'operazione `ListJobs`, puoi chiamare questa operazione per accedere ai dati a cui sono state concesse le autorizzazioni. Questa operazione supporta tutte le autorizzazioni IAM, tra cui le condizioni di autorizzazione che utilizzano i tag.

Richiesta

- `JobNames`. Obbligatorio: matrice di stringhe UTF-8.

  L'elenco dei nomi di processo, che potrebbero essere i nomi restituiti dall'operazione `ListJobs`.

Risposta

- `Jobs`: una matrice di oggetti Processo.

  Un elenco di definizioni del processo.
- `JobsNotFound`: una matrice di stringhe UTF-8.

  Un elenco di nomi di processi non trovati.

Errori

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

UpdateSourceControlFromJob azione (Python: update_source_control_from_job)

Sincronizza un processo con il repository di controllo del codice sorgente. Questa operazione prende gli artefatti del lavoro dagli archivi AWS Glue interni ed effettua un commit nell'archivio remoto configurato sul job.

Questa API supporta parametri opzionali che acquisiscono le informazioni del repository.

Richiesta

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del AWS Glue lavoro da sincronizzare verso o dal repository remoto.
- `Provider`: stringa UTF-8.

  Il provider per il repository remoto. Valori possibili: GITHUB, AWS_CODE_COMMIT, GITLAB, BITBUCKET.
- `RepositoryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del repository remoto che contiene gli artefatti del processo. Per i BitBucket provider, `RepositoryName` dovrebbe includere. `WorkspaceName` Utilizzare il formato `<WorkspaceName>/<RepositoryName>`.
- `RepositoryOwner`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il proprietario del repository remoto che contiene gli artefatti del processo.
- `BranchName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un ramo opzionale nel repository remoto.

- `Folder`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Una cartella opzionale nel repository remoto.

- `CommitId`: stringa UTF-8, non inferiore a 1 o superiore a 40 byte di lunghezza, corrispondente a Single-line string pattern.

  Un ID di commit per un commit nel repository remoto.

- `AuthStrategy`: stringa UTF-8.

  Il tipo di autenticazione, che può essere un token di autenticazione memorizzato in AWS Secrets Manager o un token di accesso personale.

- `AuthToken`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il valore del token di autorizzazione.

Risposta

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del AWS Glue lavoro.

Errori

- `AccessDeniedException`
- `AlreadyExistsException`
- `InvalidInputException`
- `ValidationException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

UpdateJobFromSourceControl azione (Python: update_job_from_source_control)

Sincronizza un processo dal repository di controllo del codice sorgente. Questa operazione prende gli artefatti del lavoro che si trovano nell'archivio remoto e aggiorna gli archivi interni con questi artefatti. AWS Glue

Questa API supporta parametri opzionali che acquisiscono le informazioni del repository.

Richiesta

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del AWS Glue lavoro da sincronizzare verso o dal repository remoto.
- `Provider`: stringa UTF-8.

  Il provider per il repository remoto. Valori possibili: GITHUB, AWS_CODE_COMMIT, GITLAB, BITBUCKET.
- `RepositoryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del repository remoto che contiene gli artefatti del processo. Per i BitBucket provider, `RepositoryName` dovrebbe includere. `WorkspaceName` Utilizzare il formato `<WorkspaceName>/<RepositoryName>`.
- `RepositoryOwner`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il proprietario del repository remoto che contiene gli artefatti del processo.
- `BranchName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un ramo opzionale nel repository remoto.
- `Folder`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Una cartella opzionale nel repository remoto.
- `CommitId`: stringa UTF-8, non inferiore a 1 o superiore a 40 byte di lunghezza, corrispondente a Single-line string pattern.

Un ID di commit per un commit nel repository remoto.

- `AuthStrategy`: stringa UTF-8.

  Il tipo di autenticazione, che può essere un token di autenticazione memorizzato in AWS Secrets Manager o un token di accesso personale.

- `AuthToken`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il valore del token di autorizzazione.

Risposta

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del AWS Glue lavoro.

Errori

- `AccessDeniedException`
- `AlreadyExistsException`
- `InvalidInputException`
- `ValidationException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Esecuzioni di processi

L'API Jobs Runs descrive i tipi di dati e l'API relativi all'avvio, all'arresto o alla visualizzazione delle esecuzioni di job e alla reimpostazione dei segnalibri dei processi, in. AWS Glue

Tipi di dati

- JobRun struttura

- Struttura Predecessor
- JobBookmarkEntry struttura
- BatchStopJobRunSuccessfulSubmission struttura
- BatchStopJobRunError struttura

JobRun struttura

Contiene informazioni su una esecuzione di processo.

Campi

- `Id`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di questa esecuzione di processo.
- `Attempt`: numero (intero).

  Il numero di tentativi di esecuzione di questo processo.
- `PreviousRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID dell'esecuzione precedente di questo processo. Ad esempio, il `JobRunId` specificato nell'operazione `StartJobRun`.
- `TriggerName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger che ha avviato questa esecuzione progetto.
- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione di processo in uso in questa esecuzione.
- `StartedOn`: timestamp.

  La data e ora in cui questa esecuzione di processo è stata avviata.
- `LastModifiedOn`: timestamp.

  L'ultima volta in cui questa esecuzione di processo è stata modificata.

- `CompletedOn`: timestamp.

  La data e ora in cui questa elaborazione di processo è stata completata.

- `JobRunState`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING`).

  Lo stato attuale del processo eseguito. Per ulteriori informazioni sugli stati dei processi terminati in modo anomalo, consulta AWS Glue Stati di esecuzione dei processi di .

- `Arguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Gli argomenti del processo associati a questa esecuzione. Per questa esecuzione di processo, sostituiscono gli argomenti predefiniti impostati nella definizione del processo stessa.

  Qui è possibile specificare gli argomenti utilizzati dal proprio script di esecuzione del lavoro, nonché gli argomenti utilizzati a sua volta. AWS Glue

  Gli argomenti del processo potrebbero essere registrati. Non passare segreti in testo chiaro come argomenti. Recupera i segreti da una AWS Glue connessione AWS Secrets Manager o da un altro meccanismo di gestione dei segreti se intendi mantenerli all'interno del Job.

  Per informazioni su come specificare e utilizzare gli argomenti del proprio processo, fai riferimento a Chiamare le API AWS Glue in Python nella guida per gli sviluppatori.

  Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Spark, consulta la pagina Special Parameters Used by AWS Glue nella Guida per gli sviluppatori.

  Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Ray, consulta la pagina Using job parameters in Ray jobs nella Guida per gli sviluppatori.

- `ErrorMessage`: stringa UTF-8.

  Un messaggio di errore associato a questa esecuzione di processo.

- `PredecessorRuns`: una matrice di oggetti Predecessor.

  Un elenco di predecessori di questa esecuzione di processo.

- `AllocatedCapacity`: numero (intero).

  in quanto obsoleto. Usare invece `MaxCapacity`.

  Il numero di unità di elaborazione AWS Glue dati (DPU) assegnate a questo scopo. JobRun Si possono allocare da 2 a 100 DPU; il valore di default è 10. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

- `ExecutionTime`: numero (intero).

  Quantità di tempo (in secondi) durante cui l'esecuzione del processo ha utilizzato le risorse.

- `Timeout`: numero (intero), almeno 1.

  Timeout di `JobRun` (in minuti). Indica il tempo massimo durante cui l'esecuzione di un processo può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Questo valore sostituisce il valore di timeout impostato nel processo padre.

  I processi di streaming non hanno un timeout. Il valore predefinito per i processi non in streaming è 2.880 minuti (48 ore).

- `MaxCapacity`: numero (doppio).

  Per i lavori di Glue versione 1.0 o precedente, utilizzando il tipo di worker standard, il numero di unità di elaborazione AWS Glue dati (DPU) che possono essere allocate durante l'esecuzione di questo lavoro. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  Per i processi Glue versione 2.0 e successive, non è possibile specificare il valore `Maximum capacity`. Si deve invece specificare un `Worker type` e un `Number of workers`.

  Non impostare `MaxCapacity` se usi `WorkerType` e `NumberOfWorkers`.

  Il valore che è possibile allocare per `MaxCapacity` varia a seconda che si esegua un processo shell di Python, un processo ETL di Apache Spark o un processo ETL di streaming di Apache Spark:

  - Quando si specifica un processo shell di Python (`JobCommand.Name`="pythonshell"), è possibile allocare 0,0625 o 1 DPU. Il valore di default è 0,0625 DPU.

- Quando si specifica un processo ETL Apache Spark (`JobCommand.Name`="glueetl") o un processo ETL di streaming Apache Spark (`JobCommand.Name`="gluestreaming"), è possibile allocare da 2 a 100 DPU. Il valore di default è 10 DPU. Questo tipo di processo non può avere un'allocazione DPU frazionata.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard`="" | `G.1X`="" | `G.2X`="" | `G.025X`="" | `G.4X`="" | `G.8X`="" | `Z.2X`="").

  Il tipo di worker predefinito allocato quando viene eseguito un processo. Accetta un valore di G.1X, G.2X, G.4X, G.8X o G.025X per i processi Spark. Accetta il valore Z.2X per i processi Ray.

  - Per il tipo di worker `G.1X`, ciascun worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.2X`, ciascun worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria) con disco da 128 GB (circa 77 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.4X`, ciascun worker esegue la mappatura su 4 DPU (16 vCPU, 64 GB di memoria) con disco da 256 GB (circa 235 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di lavoratore è disponibile solo per i lavori Spark ETL AWS Glue versione 3.0 o successiva AWS nelle seguenti regioni: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda) ed Europa (Stoccolma).
  - Per il tipo di worker `G.8X`, ciascun worker esegue la mappatura su 8 DPU (32 vCPU, 128 GB di memoria) con disco da 512 GB (circa 487 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i job Spark ETL AWS Glue versione 3.0 o successiva, nelle stesse AWS regioni supportate per il tipo di `G.4X` lavoratore.
  - Per il tipo di worker `G.025X`, ciascun worker esegue la mappatura su 0,25 DPU (2 vCPU, 4 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Consigliamo questo tipo di worker per i processi di streaming a basso volume. Questo tipo di worker è disponibile solo per i lavori di streaming della AWS Glue versione 3.0.

  - Per il tipo di worker `Z.2X`, ciascun worker esegue la mappatura su 2 M-DPU (8 vCPU, 64 GB di memoria) con disco da 128 GB (circa 120 GB liberi) e fornisce un massimo di 8 worker Ray in base all'autoscaler.
- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` specifico allocati quando viene eseguito un processo.
- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da usare con questa esecuzione del processo.
- `LogGroupName`: stringa UTF-8.

  Il nome del gruppo di log per la registrazione sicura che può essere crittografato lato server in Amazon utilizzando. CloudWatch AWS KMS Questo nome può essere `/aws-glue/jobs/` e in questo caso la crittografia di default è `NONE`. Se si aggiunge un nome di ruolo e il nome `SecurityConfiguration` (in altre parole, `/aws-glue/jobs-yourRoleName-yourSecurityConfigurationName/`), la configurazione di sicurezza viene utilizzata per crittografare il gruppo di log.
- `NotificationProperty`: un oggetto NotificationProperty.

  Specifica le proprietà di configurazione di una notifica di esecuzione di un processo.
- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Nei job Spark, `GlueVersion` determina le versioni di Apache Spark e Python disponibili in un job. AWS Glue La versione Python indica la versione supportata per i processi di tipo Spark.

  I processi Ray devono impostare il valore di `GlueVersion` su `4.0` o superiore. Tuttavia, le versioni di Ray, Python e le librerie aggiuntive disponibili nel processo Ray sono determinate dal parametro `Runtime` del comando del processo.

  Per ulteriori informazioni sulle AWS Glue versioni disponibili e sulle versioni corrispondenti di Spark e Python, consulta la versione Glue nella guida per sviluppatori.

  Processi creati senza specificare una versione Glue utilizzano Glue 0.9 per impostazione predefinita.
- `DPUSeconds`: numero (doppio).

Questo campo viene compilato solo quando si esegue un processo Auto Scaling e rappresenta il tempo totale di esecuzione di ciascun esecutore durante il ciclo di vita di un processo in secondi, moltiplicato per un fattore DPU (1 per `G.1X`, 2 per `G.2X` o 0,25 per `G.025X` worker). Questo valore potrebbe essere diverso da quello `executionEngineRuntime` * `MaxCapacity` come nel caso dei processi di Auto Scaling, poiché il numero di esecutori in esecuzione in un determinato momento potrebbe essere inferiore a `MaxCapacity`. Pertanto, è possibile che il valore di `DPUSeconds` sia minore di `executionEngineRuntime` * `MaxCapacity`.

- `ExecutionClass`: una stringa UTF-8, non superiore a 16 byte di lunghezza (valori validi: `FLEX=""` | `STANDARD=""`).

  Indica se il processo viene eseguito con una classe di esecuzione standard o flessibile. La classe di esecuzione standard è ideale per carichi di lavoro sensibili al tempo che richiedono un avvio rapido dei processi e risorse dedicate.

  La classe di esecuzione flessibile è appropriata per i processi non sensibili al tempo i cui tempi di inizio e completamento possono variare.

  Potranno essere `ExecutionClass` impostati solo i lavori con la AWS Glue versione 3.0 e successive e `glueetl` il tipo di comando. `FLEX` La classe di esecuzione flessibile è disponibile per i processi Spark.
- `ProfileName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di un profilo di AWS Glue utilizzo associato all'esecuzione del processo.

Struttura Predecessor

Un'esecuzione di processo che è stata usata nel predicato di un trigger condizionale che ha attivato l'esecuzione di processo corrente.

Campi

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione di processo usata dall'esecuzione del processo predecessore.
- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID dell'esecuzione di processo dell'esecuzione processo predecessore.

JobBookmarkEntry struttura

Definisce un punto nel quale un processo può riprendere l'elaborazione.

Campi

- `JobName`: stringa UTF-8.

  Il nome del processo in questione.
- `Version`: numero (intero).

  Versione del processo.
- `Run`: numero (intero).

  Il numero di ID dell'esecuzione.
- `Attempt`: numero (intero).

  Il numero di ID del tentativo.
- `PreviousRunId`: stringa UTF-8.

  Identificatore di esecuzione univoco associato all'esecuzione del processo precedente.
- `RunId`: stringa UTF-8.

  Il numero di ID dell'esecuzione.
- `JobBookmark`: stringa UTF-8.

  Il segnalibro stesso.

BatchStopJobRunSuccessfulSubmission struttura

Registra una richiesta di arresto riuscita per un `JobRun` specificato.

Campi

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Nome della definizione di processo usata nell'esecuzione del processo che è stata arrestata.

- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Oggetto `JobRunId` dell'esecuzione del processo arrestata.

BatchStopJobRunError struttura

Registra un errore che si è verificato durante il tentativo di arrestare un'esecuzione di un processo specifica.

Campi

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione di processo usata nell'esecuzione del processo in questione.
- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  `JobRunId` dell'esecuzione del processo in questione.
- `ErrorDetail`: un oggetto ErrorDetail.

  Specifica dettagli relativi all'errore che si è verificato.

Operazioni

- StartJobRun azione (Python: start_job_run)
- BatchStopJobRun azione (Python: batch_stop_job_run)
- GetJobRun azione (Python: get_job_run)
- GetJobRuns azione (Python: get_job_runs)
- GetJobBookmark azione (Python: get_job_bookmark)
- GetJobBookmarks azione (Python: get_job_bookmarks)
- ResetJobBookmark azione (Python: reset_job_bookmark)

StartJobRun azione (Python: start_job_run)

Avvia un'esecuzione di un processo usando una definizione di processo.

Richiesta

- `JobName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione di processo da usare.
- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di un precedente `JobRun` da ripetere.
- `Arguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Gli argomenti del processo associati a questa esecuzione. Per questa esecuzione di processo, sostituiscono gli argomenti predefiniti impostati nella definizione del processo stessa.

  Qui è possibile specificare gli argomenti utilizzati dal proprio script di esecuzione del lavoro, nonché gli argomenti utilizzati dal proprio script di esecuzione del lavoro. AWS Glue

  Gli argomenti del processo potrebbero essere registrati. Non passare segreti in testo chiaro come argomenti. Recupera i segreti da una AWS Glue connessione AWS Secrets Manager o da un altro meccanismo di gestione dei segreti se intendi mantenerli all'interno del Job.

  Per informazioni su come specificare e utilizzare gli argomenti del proprio processo, fai riferimento a Chiamare le API AWS Glue in Python nella guida per gli sviluppatori.

  Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Spark, consulta la pagina Special Parameters Used by AWS Glue nella Guida per gli sviluppatori.

  Per informazioni sugli argomenti che puoi fornire a questo campo durante la configurazione dei processi Ray, consulta la pagina Using job parameters in Ray jobs nella Guida per gli sviluppatori.
- `AllocatedCapacity`: numero (intero).

in quanto obsoleto. Usare invece `MaxCapacity`.

Il numero di unità di elaborazione AWS Glue dati (DPU) da assegnare a questo. JobRun È possibile allocare un minimo di 2 DPU; l'impostazione di default è 10. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

- `Timeout`: numero (intero), almeno 1.

  Timeout di `JobRun` (in minuti). Indica il tempo massimo durante cui l'esecuzione di un processo può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Questo valore sostituisce il valore di timeout impostato nel processo padre.

  I processi di streaming non hanno un timeout. Il valore predefinito per i processi non in streaming è 2.880 minuti (48 ore).

- `MaxCapacity`: numero (doppio).

  Per i lavori di Glue versione 1.0 o precedente, utilizzando il tipo di worker standard, il numero di unità di elaborazione AWS Glue dati (DPU) che possono essere allocate durante l'esecuzione di questo lavoro. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  Per i processi Glue versione 2.0 e successive, non è possibile specificare il valore `Maximum capacity`. Si deve invece specificare un `Worker type` e un `Number of workers`.

  Non impostare `MaxCapacity` se usi `WorkerType` e `NumberOfWorkers`.

  Il valore che è possibile allocare per `MaxCapacity` varia a seconda che si esegua un processo shell di Python, un processo ETL di Apache Spark o un processo ETL di streaming di Apache Spark:

  - Quando si specifica un processo shell di Python (`JobCommand.Name`="pythonshell"), è possibile allocare 0,0625 o 1 DPU. Il valore di default è 0,0625 DPU.
  - Quando si specifica un processo ETL Apache Spark (`JobCommand.Name`="glueetl") o un processo ETL di streaming Apache Spark (`JobCommand.Name`="gluestreaming"), è possibile allocare da 2 a 100 DPU. Il valore di default è 10 DPU. Questo tipo di processo non può avere un'allocazione DPU frazionata.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da usare con questa esecuzione del processo.

- `NotificationProperty`: un oggetto NotificationProperty.

  Specifica le proprietà di configurazione di una notifica di esecuzione di un processo.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato quando viene eseguito un processo. Accetta un valore di G.1X, G.2X, G.4X, G.8X o G.025X per i processi Spark. Accetta il valore Z.2X per i processi Ray.

  - Per il tipo di worker `G.1X`, ciascun worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.2X`, ciascun worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria) con disco da 128 GB (circa 77 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.4X`, ciascun worker esegue la mappatura su 4 DPU (16 vCPU, 64 GB di memoria) con disco da 256 GB (circa 235 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di lavoratore è disponibile solo per i lavori Spark ETL AWS Glue versione 3.0 o successiva AWS nelle seguenti regioni: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda) ed Europa (Stoccolma).
  - Per il tipo di worker `G.8X`, ciascun worker esegue la mappatura su 8 DPU (32 vCPU, 128 GB di memoria) con disco da 512 GB (circa 487 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i job Spark ETL AWS Glue versione 3.0 o successiva, nelle stesse AWS regioni supportate per il tipo di `G.4X` lavoratore.
  - Per il tipo di worker `G.025X`, ciascun worker esegue la mappatura su 0,25 DPU (2 vCPU, 4 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Consigliamo questo

tipo di worker per i processi di streaming a basso volume. Questo tipo di worker è disponibile solo per i lavori di streaming della AWS Glue versione 3.0.

- Per il tipo di worker `Z.2X`, ciascun worker esegue la mappatura su 2 M-DPU (8 vCPU, 64 GB di memoria) con disco da 128 GB (circa 120 GB liberi) e fornisce un massimo di 8 worker Ray in base all'autoscaler.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` specifico allocati quando viene eseguito un processo.

- `ExecutionClass`: una stringa UTF-8, non superiore a 16 byte di lunghezza (valori validi: `FLEX=""` | `STANDARD=""`).

  Indica se il processo viene eseguito con una classe di esecuzione standard o flessibile. La classe di esecuzione standard è ideale per carichi di lavoro sensibili al tempo che richiedono un avvio rapido dei processi e risorse dedicate.

  La classe di esecuzione flessibile è appropriata per i processi non sensibili al tempo i cui tempi di inizio e completamento possono variare.

  Solo i lavori con AWS Glue versione 3.0 e successive e il tipo di comando `glueetl` potranno essere `ExecutionClass` impostati su`FLEX`. La classe di esecuzione flessibile è disponibile per i processi Spark.

- `ProfileName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di un profilo di AWS Glue utilizzo associato all'esecuzione del processo.

Risposta

- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID assegnato a questa esecuzione processo.

Errori

- `InvalidInputException`
- `EntityNotFoundException`

- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

BatchStopJobRun azione (Python: batch_stop_job_run)

Arresta una o più esecuzioni del processo per una definizione di processo specificata.

Richiesta

- `JobName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione di processo per cui arrestare le esecuzioni del processo.
- `JobRunIds` obbligatorio: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 25 stringhe.

  Elenco degli oggetti `JobRunIds` che dovrebbero essere arrestati per la definizione di processo.

Risposta

- `SuccessfulSubmissions`: una matrice di oggetti BatchStopJobRunSuccessfulSubmission.

  Un elenco di quelli che sono stati inviati correttamente per l'interruzione. JobRuns
- `Errors`: una matrice di oggetti BatchStopJobRunError.

  Un elenco degli errori rilevati nel tentativo di arrestare `JobRuns`, incluso il `JobRunId` per il quale si è verificato ciascun errore e i dettagli sull'errore stesso.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobRun azione (Python: get_job_run)

Recupera i metadati per una determinata esecuzione di processo.

Richiesta

- `JobName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della definizione di processo in esecuzione.
- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID dell'esecuzione processo.
- `PredecessorsIncluded`: booleano.

  True se un elenco delle esecuzioni predecessore deve essere restituito.

Risposta

- `JobRun`: un oggetto JobRun.

  I metadati di esecuzione del processo richiesti.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobRuns azione (Python: get_job_runs)

Recupera i metadati per tutte le esecuzioni di una definizione di processo specifica.

Richiesta

- `JobName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Nome della definizione di processo per cui recuperare tutte le esecuzioni del processo.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

- `MaxResults`— Numero (intero), non inferiore a 1 o superiore a 200.

  La dimensione massima della risposta.

Risposta

- `JobRuns`: una matrice di oggetti JobRun.

  Un elenco di oggetti metadati esecuzione processo.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non tutte le esecuzioni di processo richieste sono state restituite.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobBookmark azione (Python: get_job_bookmark)

Restituisce informazioni su una voce del segnalibro di processo.

Per ulteriori informazioni sull'abilitazione e l'utilizzo dei segnalibri di processo, consulta:

- Monitoraggio dei dati elaborati mediante segnalibri di processo
- Parametri Job utilizzati da AWS Glue
- Struttura del processo

Richiesta

- `JobName`. Obbligatorio: stringa UTF-8.

  Il nome del processo in questione.

- `Version`: numero (intero).

  Versione del processo.

- `RunId`: stringa UTF-8.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di processo.

Risposta

- `JobBookmarkEntry`: un oggetto JobBookmarkEntry.

  Struttura che definisce un punto in cui un processo può riprendere l'elaborazione.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ValidationException`

GetJobBookmarks azione (Python: get_job_bookmarks)

Restituisce informazioni sulle voci del segnalibro di processo. L'elenco è ordinato sui numeri di versione decrescenti.

Per ulteriori informazioni sull'abilitazione e l'utilizzo dei segnalibri di processo, consulta:

- Monitoraggio dei dati elaborati mediante segnalibri di processo
- Parametri Job utilizzati da AWS Glue
- Struttura del processo

Richiesta

- `JobName`. Obbligatorio: stringa UTF-8.

  Il nome del processo in questione.
- `MaxResults`: numero (intero).

  La dimensione massima della risposta.
- `NextToken`: numero (intero).

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `JobBookmarkEntries`: una matrice di oggetti JobBookmarkEntry.

  Elenco di voci del segnalibro di processo che definisce un punto in cui un processo può riprendere l'elaborazione.
- `NextToken`: numero (intero).

  Un token di continuazione, che ha un valore pari a 1 se vengono restituite tutte le voci, oppure > 1 se non vengono restituite tutte le esecuzioni di processo richieste.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

ResetJobBookmark azione (Python: reset_job_bookmark)

Ripristina una voce segnalibro.

Per ulteriori informazioni sull'abilitazione e l'utilizzo dei segnalibri di processo, consulta:

- Monitoraggio dei dati elaborati mediante segnalibri di processo
- Parametri Job utilizzati da AWS Glue

- Struttura del processo

Richiesta

- `JobName`. Obbligatorio: stringa UTF-8.

  Il nome del processo in questione.
- `RunId`: stringa UTF-8.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di processo.

Risposta

- `JobBookmarkEntry`: un oggetto JobBookmarkEntry.

  La voce di ripristino del segnalibro.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Trigger

L'API Triggers descrive i tipi di dati e l'API relativa alla creazione, all'aggiornamento o all'eliminazione e l'avvio e l'arresto di attivatori di processi in AWS Glue.

Tipi di dati

- Struttura trigger
- TriggerUpdate struttura
- Struttura predicato
- Struttura condizione
- Struttura operazione

- EventBatchingCondition struttura

Struttura trigger

Informazioni su un trigger specifico.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger.

- `WorkflowName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del flusso di lavoro associato al trigger.

- `Id`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Riservato per uso futuro.

- `Type`: stringa UTF-8 (valori validi: `SCHEDULED` | `CONDITIONAL` | `ON_DEMAND` | `EVENT`).

  Il tipo di trigger.

- `State`: stringa UTF-8 (valori validi: `CREATING` | `CREATED` | `ACTIVATING` | `ACTIVATED` | `DEACTIVATING` | `DEACTIVATED` | `DELETING` | `UPDATING`).

  Lo stato corrente del trigger.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione di questo trigger.

- `Schedule`: stringa UTF-8.

  Espressione `cron` usata per specificare la pianificazione (consulta Pianificazioni basate sul tempo per processi e crawler. Ad esempio, per eseguire un processo ogni giorno alle 12:15 UTC, devi specificare: `cron(15 12 * * ? *)`.

- `Actions`: una matrice di oggetti Azione.

Le operazioni avviate da questo trigger.

- `Predicate`: un oggetto Predicate.

  Il predicato di questo trigger, che definisce quando verrà attivato.

- `EventBatchingCondition`: un oggetto EventBatchingCondition.

  Condizione del batch che deve essere soddisfatta (numero specificato di eventi ricevuti o periodo di tempo del batch scaduto) prima che venga attivato EventBridge l'attivazione dell'evento.

TriggerUpdate struttura

Una struttura utilizzata per fornire informazioni per l'aggiornamento di un trigger. Questo oggetto aggiorna la definizione trigger precedente sovrascrivendola completamente.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Riservato per uso futuro.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione di questo trigger.

- `Schedule`: stringa UTF-8.

  Espressione `cron` usata per specificare la pianificazione (consulta Pianificazioni basate sul tempo per processi e crawler. Ad esempio, per eseguire un processo ogni giorno alle 12:15 UTC, devi specificare: `cron(15 12 * * ? *)`.

- `Actions`: una matrice di oggetti Azione.

  Le operazioni avviate da questo trigger.

- `Predicate`: un oggetto Predicate.

  Il predicato di questo trigger, che definisce quando verrà attivato.

- `EventBatchingCondition`: un oggetto EventBatchingCondition.

Condizione del batch che deve essere soddisfatta (numero specificato di eventi ricevuti o periodo di tempo del batch scaduto) prima che venga attivato EventBridge l'attivazione dell'evento.

Struttura predicato

Definisce il predicato del trigger, che determina il momento in cui viene attivato.

Campi

- `Logical`: stringa UTF-8 (valori validi: `AND` | `ANY`).

  Campo opzionale se è elencata una sola condizione. Se sono elencate più condizioni, questo campo è obbligatorio.
- `Conditions`: una matrice di oggetti Condition.

  Un elenco delle condizioni che determinano il momento in cui il trigger verrà attivato.

Struttura condizione

Definisce una condizione nella quale un trigger si attiva.

Campi

- `LogicalOperator`: stringa UTF-8 (valori validi: `EQUALS`).

  Un operatore logico.
- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del processo ali cui `JobRuns` si applica questa condizione e su cui attende questo trigger.
- `State`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING`).

  Lo stato della condizione. Attualmente, gli unici processi che stabiliscono che un trigger può essere ascoltato sono `SUCCEEDED`, `STOPPED`, `FAILED` e `TIMEOUT`. Gli unici crawler che stabiliscono che un trigger può ascoltare sono `SUCCEEDED`, `FAILED` e `CANCELLED`.
- `CrawlerName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome del crawler a cui si applica questa condizione.

- `CrawlState`: stringa UTF-8 (valori validi: `RUNNING` | `CANCELLING` | `CANCELLED` | `SUCCEEDED` | `FAILED` | `ERROR`).

  Lo stato del crawler a cui si applica questa condizione.

Struttura operazione

Definisce un'operazione che deve essere avviata da un trigger.

Campi

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del processo che viene eseguito.
- `Arguments`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Gli argomenti del processo utilizzati quando viene attivato il trigger. Per questa esecuzione di processo, sostituiscono gli argomenti predefiniti impostati nella definizione del processo stessa.

  Qui puoi specificare gli argomenti utilizzati dal tuo script di esecuzione processo, nonché gli argomenti utilizzati da AWS Glue stesso.

  Per informazioni su come specificare e utilizzare gli argomenti del proprio processo, fai riferimento a Chiamare le API AWS Glue in Python nella guida per gli sviluppatori.

  Per informazioni sulle coppie chiave-valore che AWS Glue utilizza per configurare il processo, fai riferimento a Parametri speciali utilizzati da AWS Glue contenuto nella guida per gli sviluppatori.
- `Timeout`: numero (intero), almeno 1.

  Timeout di `JobRun` (in minuti). Indica il tempo massimo durante cui l'esecuzione di un processo può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore). Questo valore sostituisce il valore di timeout impostato nel processo padre.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da usare con questa operazione.
- `NotificationProperty`: un oggetto NotificationProperty.

  Specifica le proprietà di configurazione di una notifica di esecuzione di un processo.
- `CrawlerName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del crawler da usare con questa operazione.

EventBatchingCondition struttura

Condizione del batch che deve essere soddisfatta (numero specificato di eventi ricevuti o periodo di tempo del batch scaduto) prima che venga attivato EventBridge l'attivazione dell'evento.

Campi

- `BatchSize` obbligatorio: numero (intero), non inferiore a 1 o superiore a 100.

  Numero di eventi che devono essere ricevuti da Amazon EventBridge prima che si EventBridge verifichi l'evento.
- `BatchWindow`: numero (intero), non inferiore a 1 o superiore a 900.

  Intervallo di tempo in secondi dopo il quale si attiva EventBridge l'attivazione dell'evento. La finestra inizia quando viene ricevuto il primo evento.

Operazioni

- CreateTrigger azione (Python: create_trigger)
- StartTrigger azione (Python: start_trigger)
- GetTrigger azione (Python: get_trigger)
- GetTriggers azione (Python: get_triggers)
- UpdateTrigger azione (Python: update_trigger)
- StopTrigger azione (Python: stop_trigger)
- DeleteTrigger azione (Python: delete_trigger)

- ListTriggers azione (Python: list_triggers)
- BatchGetTriggers azione (Python: batch_get_triggers)

CreateTrigger azione (Python: create_trigger)

Crea un nuovo trigger.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger.
- `WorkflowName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del flusso di lavoro associato al trigger.
- `Type` – Obbligatorio: stringa UTF-8 (valori validi: `SCHEDULED` | `CONDITIONAL` | `ON_DEMAND` | `EVENT`).

  Il tipo del nuovo trigger.
- `Schedule`: stringa UTF-8.

  Espressione `cron` usata per specificare la pianificazione (consulta Pianificazioni basate sul tempo per processi e crawler. Ad esempio, per eseguire un processo ogni giorno alle 12:15 UTC, devi specificare: `cron(15 12 * * ? *)`.

  Questo campo è obbligatorio quando il tipo di trigger è SCHEDULED (PIANIFICATO).
- `Predicate`: un oggetto Predicate.

  Un predicato per specificare quando occorre attivare il nuovo trigger.

  Questo campo è obbligatorio quando il tipo di trigger è `CONDITIONAL`.
- `Actions`: obbligatorio: una matrice di oggetti Azione.

  Le operazioni avviate da questo trigger al momento dell'attivazione.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

Una descrizione del nuovo trigger.

- `StartOnCreation`: booleano.

  Imposta su `true` per avviare i trigger `SCHEDULED` e `CONDITIONAL` al momento della creazione. True non è supportato per i trigger `ON_DEMAND`.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag da usare con questo trigger. Puoi usare i tag per limitare l'accesso al trigger. Per ulteriori informazioni sui tag in AWS Glue, consulta Tag AWS in AWS Glue nella guida per gli sviluppatori.

- `EventBatchingCondition`: un oggetto EventBatchingCondition.

  Condizione del batch che deve essere soddisfatta (numero specificato di eventi ricevuti o periodo di tempo del batch scaduto) prima che venga attivato EventBridge l'attivazione dell'evento.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger.

Errori

- `AlreadyExistsException`
- `EntityNotFoundException`
- `InvalidInputException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

StartTrigger azione (Python: start_trigger)

Avvia un trigger esistente. Consulta la sezione Avvio dei processi per informazioni sull'avvio dei diversi tipi di trigger.

Richiesta

- Name: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger da avviare.

Risposta

- Name: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger avviato.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

GetTrigger azione (Python: get_trigger)

Recupera la definizione di un trigger.

Richiesta

- Name: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del trigger da recuperare.

Risposta

- `Trigger`: un oggetto Trigger.

  La definizione del trigger richiesta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetTriggers azione (Python: get_triggers)

Ottiene tutti i trigger associati a un processo.

Richiesta

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

- `DependentJobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del processo per cui recuperare i trigger. Il trigger che può avviare questo processo viene restituito e, se non esiste nessun trigger di questo tipo, vengono restituiti tutti i trigger.

- `MaxResults`— Numero (intero), non inferiore a 1 o superiore a 200.

  La dimensione massima della risposta.

Risposta

- `Triggers`: una matrice di oggetti Trigger.

  Un elenco di trigger per il processo specificato.

- `NextToken`: stringa UTF-8.

Un token di continuazione, se non sono ancora stati restituiti tutti i trigger richiesti.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

UpdateTrigger azione (Python: update_trigger)

Aggiorna una definizione del trigger.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger da aggiornare.
- `TriggerUpdate`: obbligatorio: un oggetto TriggerUpdate.

  I nuovi valori con cui aggiornare il trigger.

Risposta

- `Trigger`: un oggetto Trigger.

  La definizione del trigger risultante.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

StopTrigger azione (Python: stop_trigger)

Arresta un trigger specificato.

Richiesta

- Name: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger da arrestare.

Risposta

- Name: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger che è stato arrestato.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

DeleteTrigger azione (Python: delete_trigger)

Elimina un trigger specificato. Se il trigger non viene trovato, non viene generata alcuna eccezione.

Richiesta

- Name: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger da eliminare.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del trigger che è stato eliminato.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

ListTriggers azione (Python: list_triggers)

Recupera i nomi di tutte le risorse trigger in questo account AWS oppure le risorse con il tag specificato. Questa operazione consente di vedere quali risorse sono disponibili nel proprio account e i relativi nomi.

L'operazione accetta il campo facoltativo `Tags` che si può utilizzare come filtro per la risposta in modo che le risorse con tag possano essere recuperate come gruppo. Se si sceglie di utilizzare il filtro dei tag, potranno essere recuperate solo le risorse con tag.

Richiesta

- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.
- `DependentJobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del processo per cui recuperare i trigger. Viene restituito il trigger che può avviare questo processo. Se non esiste un trigger di questo tipo, vengono restituiti tutti i trigger.
- `MaxResults`— Numero (intero), non inferiore a 1 o superiore a 200.

  La dimensione massima di un elenco da restituire.
- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

Specifica che vengono restituite solo le risorse con tag.

Risposta

- `TriggerNames`: una matrice di stringhe UTF-8.

  I nomi di tutti i trigger nell'account oppure i trigger con i tag specificati.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se l'elenco restituito non contiene l'ultimo parametro disponibile.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchGetTriggers azione (Python: batch_get_triggers)

Restituisce un elenco di metadati di risorse per un determinato elenco di nomi di trigger. Dopo aver chiamato l'operazione `ListTriggers`, puoi chiamare questa operazione per accedere ai dati a cui sono state concesse le autorizzazioni. Questa operazione supporta tutte le autorizzazioni IAM, tra cui le condizioni di autorizzazione che utilizzano i tag.

Richiesta

- `TriggerNames`. Obbligatorio: matrice di stringhe UTF-8.

  L'elenco dei nomi di trigger che potrebbero essere i nomi restituiti dall'operazione `ListTriggers`.

Risposta

- `Triggers`: una matrice di oggetti Trigger.

  Un elenco di definizioni di trigger.
- `TriggersNotFound`: una matrice di stringhe UTF-8.

  Un elenco di nomi di trigger non trovati.

Errori

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

API Sessioni interattive

L'API delle sessioni interattive descrive l' AWS Glue API relativa all'utilizzo di sessioni AWS Glue interattive per creare e testare script di estrazione, trasformazione e caricamento (ETL) per l'integrazione dei dati.

Tipi di dati

- Struttura sessione
- SessionCommand struttura
- Struttura istruzione
- StatementOutput struttura
- StatementOutputData struttura

Struttura sessione

Il periodo in cui è in esecuzione un ambiente di runtime Spark remoto.

Campi

- `Id`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID della sessione.

- `CreatedOn`: timestamp.

  La data e l'ora di creazione della sessione.

- `Status`: stringa UTF-8 (valori validi: `PROVISIONING` | `READY` | `FAILED` | `TIMEOUT` | `STOPPING` | `STOPPED`).

  Lo stato della sessione.

- `ErrorMessage`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Il messaggio di errore visualizzato durante la sessione.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La descrizione della sessione.

- `Role`: stringa UTF-8, non inferiore a 20 o superiore a 2048 byte di lunghezza, corrispondente a Custom string pattern #21.

  Il nome o l'Amazon Resource Name (ARN) del ruolo IAM associato alla sessione.

- `Command`: un oggetto SessionCommand.

  Il comando Object.see. SessionCommand

- `DefaultArguments`: una matrice di mappe con coppie chiave-valore, non superiore alle 75 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Ogni valore è una stringa UTF-8, non superiore a 4096 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una matrice della mappa di coppie chiave-valore. Il massimo è 75 coppie.

- `Connections`: un oggetto ConnectionsList.

  Il numero di connessioni utilizzate per la sessione.

- `Progress`: numero (doppio).

L'avanzamento dell'esecuzione del codice della sessione.

- `MaxCapacity`: numero (doppio).

  Il numero di unità di elaborazione AWS Glue dati (DPU) che possono essere allocate durante l'esecuzione del processo. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della SecurityConfiguration struttura da utilizzare con la sessione.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  La AWS Glue versione determina le versioni di Apache Spark e Python supportate. AWS Glue GlueVersion Deve essere maggiore di 2.0.

- `DataAccessId`: stringa UTF-8, non inferiore a 1 o superiore a 36 byte di lunghezza.

  L'ID di accesso ai dati della sessione.

- `PartitionId`: stringa UTF-8, non inferiore a 1 o superiore a 36 byte di lunghezza.

  L'ID di partizione della sessione.

- `NumberOfWorkers`: numero (intero).

  Il numero di dipendenti di uno specifico `WorkerType` da utilizzare per la sessione.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato quando viene eseguita una sessione. Accetta un valore di `G.1X`, `G.2X`, `G.4X` o `G.8X` per le sessioni Spark. Accetta il valore `Z.2X` per le sessioni Ray.

- `CompletedOn`: timestamp.

  La data e ora in cui questa sessione è stata completata.

- `ExecutionTime`: numero (doppio).

  Il tempo totale di esecuzione della sessione.

- `DPUSeconds`: numero (doppio).

Le DPU utilizzate dalla sessione (formula: ExecutionTime * MaxCapacity).

- `IdleTimeout`: numero (intero).

  Il numero di minuti di inattività prima del timeout della sessione.

- `ProfileName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di un profilo di AWS Glue utilizzo associato alla sessione.

SessionCommand struttura

Il `SessionCommand` che esegue questo lavoro.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Speciifica il nome di. SessionCommand Può essere 'glueetl' o 'gluestreaming'.

- `PythonVersion`: stringa UTF-8, corrispondente a Custom string pattern #16.

  Specifica la versione di Python. La versione Python indica la versione supportata per i processi di tipo Spark.

Struttura istruzione

La dichiarazione o la richiesta di un'operazione particolare in una sessione.

Campi

- `Id`: numero (intero).

  L'ID della dichiarazione.

- `Code`: stringa UTF-8.

  Il codice di esecuzione della dichiarazione.

- `State`: stringa UTF-8 (valori validi: `WAITING` | `RUNNING` | `AVAILABLE` | `CANCELLING` | `CANCELLED` | `ERROR`).

Lo stato mentre viene eseguita la richiesta.

- `Output`: un oggetto StatementOutput.

  L'output in JSON.

- `Progress`: numero (doppio).

  L'avanzamento dell'esecuzione del codice.

- `StartedOn`: numero (lungo).

  L'ora e la data unix in cui è stata avviata la definizione del processo.

- `CompletedOn`: numero (lungo).

  L'ora e la data unix in cui è stata completata la definizione del processo.

StatementOutput struttura

Output dell'esecuzione del codice in formato JSON.

Campi

- `Data`: un oggetto StatementOutputData.

  L'output dell'esecuzione del codice.

- `ExecutionCount`: numero (intero).

  Il numero di esecuzioni dell'output.

- `Status`: stringa UTF-8 (valori validi: `WAITING` | `RUNNING` | `AVAILABLE` | `CANCELLING` | `CANCELLED` | `ERROR`).

  Lo stato dell’output di esecuzione del codice.

- `ErrorName`: stringa UTF-8.

  Il nome dell'errore nell'output.

- `ErrorValue`: stringa UTF-8.

  Il valore dell’errore dell'output.

- `Traceback`: una matrice di stringhe UTF-8.

L’analisi dell'output.

StatementOutputData struttura

Output dell'esecuzione del codice in formato JSON.

Campi

- `TextPlain`: stringa UTF-8.

  L'output dell'esecuzione del codice in formato testo.

Operazioni

- CreateSession azione (Python: create_session)
- StopSession azione (Python: stop_session)
- DeleteSession azione (Python: delete_session)
- GetSession azione (Python: get_session)
- ListSessions azione (Python: list_sessions)
- RunStatement azione (Python: run_statement)
- CancelStatement azione (Python: cancel_statement)
- GetStatement azione (Python: get_statement)
- ListStatements azione (Python: list_statements)

CreateSession azione (Python: create_session)

Crea una nuova sessione.

Richiesta

Richiedi la creazione di una nuova sessione.

- `Id`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID della richiesta della sessione.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La descrizione della sessione.

- `Role`: obbligatorio: stringa UTF-8, non inferiore a 20 o superiore a 2048 byte di lunghezza, corrispondente a Custom string pattern #21.

  L'ARN del ruolo IAM

- `Command`: obbligatorio: un oggetto SessionCommand.

  Il `SessionCommand` che esegue questo lavoro.

- `Timeout`: numero (intero), almeno 1.

  Il numero di minuti prima che la sessione scada. L'impostazione predefinita per i processi ETL di Spark è 48 ore (2.880 minuti), la durata massima della sessione per questo tipo di processo. Consulta la documentazione per altri tipi di processo.

- `IdleTimeout`: numero (intero), almeno 1.

  Il numero di minuti di inattività prima del timeout della sessione. L'impostazione predefinita per i processi ETL di Spark è il valore di timeout. Consulta la documentazione per altri tipi di processo.

- `DefaultArguments`: una matrice di mappe con coppie chiave-valore, non superiore alle 75 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Ogni valore è una stringa UTF-8, non superiore a 4096 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una matrice della mappa di coppie chiave-valore. Il massimo è 75 coppie.

- `Connections`: un oggetto ConnectionsList.

  Il numero di connessioni da utilizzare per la sessione.

- `MaxCapacity`: numero (doppio).

  Il numero di unità di elaborazione AWS Glue dati (DPU) che possono essere allocate durante l'esecuzione del processo. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria.

- `NumberOfWorkers`: numero (intero).

  Il numero di dipendenti di uno specifico `WorkerType` da utilizzare per la sessione.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato quando viene eseguito un processo. Accetta un valore di G.1X, G.2X, G.4X o G.8X per i processi Spark. Accetta il valore Z.2X per i notebook Ray.

  - Per il tipo di worker `G.1X`, ciascun worker esegue la mappatura su 1 DPU (4 vCPU, 16 GB di memoria) con disco da 84 GB (circa 34 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.2X`, ciascun worker esegue la mappatura su 2 DPU (8 vCPU, 32 GB di memoria) con disco da 128 GB (circa 77 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per carichi di lavoro come trasformazioni di dati, join e query, in quanto offrono un modo scalabile ed economico per eseguire la maggior parte dei processi.
  - Per il tipo di worker `G.4X`, ciascun worker esegue la mappatura su 4 DPU (16 vCPU, 64 GB di memoria) con disco da 256 GB (circa 235 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di lavoratore è disponibile solo per i lavori Spark ETL AWS Glue versione 3.0 o successiva AWS nelle seguenti regioni: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda) ed Europa (Stoccolma).
  - Per il tipo di worker `G.8X`, ciascun worker esegue la mappatura su 8 DPU (32 vCPU, 128 GB di memoria) con disco da 512 GB (circa 487 GB liberi) e fornisce 1 esecutore. Questi tipi di worker sono raccomandati per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i requisiti più elevati. Questo tipo di worker è disponibile solo per i job Spark ETL AWS Glue versione 3.0 o successiva, nelle stesse AWS regioni supportate per il tipo di `G.4X` lavoratore.
  - Per il tipo di worker `Z.2X`, ciascun worker esegue la mappatura su 2 M-DPU (8 vCPU, 64 GB di memoria) con disco da 128 GB (circa 120 GB liberi) e fornisce un massimo di 8 worker Ray in base all'autoscaler.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome della SecurityConfiguration struttura da utilizzare con la sessione

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  La AWS Glue versione determina le versioni di Apache Spark e Python supportate. AWS Glue GlueVersion Deve essere maggiore di 2.0.

- `DataAccessId`: stringa UTF-8, non inferiore a 1 o superiore a 36 byte di lunghezza.

  L'ID di accesso ai dati della sessione.

- `PartitionId`: stringa UTF-8, non inferiore a 1 o superiore a 36 byte di lunghezza.

  L'ID di partizione della sessione.

- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  La mappa delle coppie di valori chiave (tag) appartenenti alla sessione.

- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  L’origine della richiesta.

- `ProfileName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di un profilo di AWS Glue utilizzo associato alla sessione.

Risposta

- `Session`: un oggetto Sessione.

  Restituisce l'oggetto di sessione nella risposta.

Errori

- `AccessDeniedException`

- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`

StopSession azione (Python: stop_session)

Interrompe la sessione.

Richiesta

- `Id`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID della sessione da interrompere.
- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  L'origine della richiesta.

Risposta

- `Id`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Restituisce l'ID della sessione interrotta.

Errori

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

- `IllegalSessionStateException`
- `ConcurrentModificationException`

DeleteSession azione (Python: delete_session)

Elimina la sessione.

Richiesta

- `Id`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID della sessione da eliminare.
- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Il nome dell'origine della richiesta di eliminazione della sessione.

Risposta

- `Id`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Restituisce l'ID della sessione eliminata.

Errori

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`
- `ConcurrentModificationException`

GetSession azione (Python: get_session)

Recupera la sessione.

Richiesta

- `Id`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID della sessione.
- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  L'origine della richiesta.

Risposta

- `Session`: un oggetto Sessione.

  Salva l'oggetto di sessione restituito nella risposta.

Errori

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

ListSessions azione (Python: list_sessions)

Recupera un elenco di sessioni.

Richiesta

- `NextToken`: stringa UTF-8, non superiore a 400000 byte di lunghezza.

  Il token per il successivo set di risultati oppure null se non ci sono altri risultati.

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Il numero massimo di risultati.

- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Tag appartenenti alla sessione.

- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  L'origine della richiesta.

Risposta

- `Ids`: una matrice di stringhe UTF-8.

  Restituisce l'ID della sessione.

- `Sessions`: una matrice di oggetti Sessione.

  Restituisce l'oggetto di sessione.

- `NextToken`: stringa UTF-8, non superiore a 400000 byte di lunghezza.

  Il token per il successivo set di risultati oppure null se non ci sono altri risultati.

Errori

- `AccessDeniedException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

RunStatement azione (Python: run_statement)

Esegue l'struzione.

Richiesta

- `SessionId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di sessione dell'istruzione da eseguire.

- `Code`: obbligatorio: stringa UTF-8, non superiore a 68000 byte di lunghezza.

  Il codice dell'istruzione da eseguire.

- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  L'origine della richiesta.

Risposta

- `Id`: numero (intero).

  Restituisce l'ID dell'istruzione che è stata eseguita.

Errori

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `ResourceNumberLimitExceededException`
- `IllegalSessionStateException`

CancelStatement azione (Python: cancel_statement)

Annulla l'istruzione.

Richiesta

- `SessionId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di sessione dell'istruzione da annullare.
- `Id`: obbligatorio: numero (intero).

  L'ID dell'istruzione da annullare.
- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  L’origine della richiesta di annullare l'istruzione.

Risposta

- Nessun parametro di risposta.

Errori

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`

GetStatement azione (Python: get_statement)

Recupera l'istruzione.

Richiesta

- `SessionId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID sessione dell'istruzione.

- `Id`: obbligatorio: numero (intero).

  L'ID dell'istruzione.
- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  L'origine della richiesta.

Risposta

- `Statement`: un oggetto Dichiarazione.

  Restituisce l'istruzione.

Errori

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`

ListStatements azione (Python: list_statements)

Elenca le istruzioni per la sessione.

Richiesta

- `SessionId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID sessione delle istruzioni.
- `RequestOrigin`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  L'origine della richiesta di elencare le istruzioni.

- `NextToken`: stringa UTF-8, non superiore a 400000 byte di lunghezza.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `Statements`: una matrice di oggetti Dichiarazione.

  Restituisce l'elenco delle istruzioni.
- `NextToken`: stringa UTF-8, non superiore a 400000 byte di lunghezza.

  Un token di continuazione, se non sono ancora stati restituiti tutte le istruzioni.

Errori

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`

API endpoint di sviluppo

L'API di endpoint di sviluppo descrive l'API AWS Glue relativa al testing utilizzando un DevEndpoint personalizzato.

Tipi di dati

- Struttura DevEndpoint
- Struttura DevEndpointCustomLibraries

Struttura DevEndpoint

Un endpoint di sviluppo in cui uno sviluppatore può eseguire in remoto il debug, la trasformazione e il caricamento degli script ETL.

Campi

- `EndpointName`: stringa UTF-8.

  Nome della `DevEndpoint`.

- `RoleArn`: stringa UTF-8, corrispondente a AWS IAM ARN string pattern.

  Amazon Resource Name (ARN) del ruolo IAM utilizzato in questo `DevEndpoint`.

- `SecurityGroupIds`: una matrice di stringhe UTF-8.

  Un elenco degli identificatori dei gruppi di sicurezza utilizzati in questo `DevEndpoint`.

- `SubnetId`: stringa UTF-8.

  La sottorete ID per questo `DevEndpoint`.

- `YarnEndpointAddress`: stringa UTF-8.

  L'indirizzo dell'endpoint YARN utilizzato da questo `DevEndpoint`.

- `PrivateAddress`: stringa UTF-8.

  Un indirizzo IP privato per accedere `DevEndpoint` all'interno di un VPC se in uno di essi viene creato `DevEndpoint` Il campo `PrivateAddress` è presente solo quando viene creato `DevEndpoint` all'interno del VPC.

- `ZeppelinRemoteSparkInterpreterPort`: numero (intero).

  La porta Apache Zeppelin per l'interprete Apache Spark remoto.

- `PublicAddress`: stringa UTF-8.

  L'indirizzo IP pubblico utilizzato da questo `DevEndpoint`. Il campo `PublicAddress` è presente solo quando si crea un `DevEndpoint` non VPC.

- `Status`: stringa UTF-8.

  Lo stato corrente di questo `DevEndpoint`.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato all'endpoint di sviluppo. Accetta un valore Standard, G.1X o G.2X.

  - Per il tipo di worker `Standard`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 50 GB e 2 esecutori.
  - Per il tipo di worker `G.1X`, ciascun worker si mappa a 1 DPU (4 vCPU, 16 GB di memoria, disco da 64 GB) e fornisce 1 esecutore. Consigliamo questo tipo di worker per i processi ad alto consumo di memoria.
  - Per il tipo di worker `G.2X`, ciascun worker si mappa a 2 DPU (8 vCPU, 32 GB di memoria, disco da 128 GB) e fornisce 1 esecutore. Consigliamo questo tipo di worker per i processi ad alto consumo di memoria.

  Problema noto: quando viene creato un endpoint di sviluppo con la configurazione `G.2X` `WorkerType`, i driver Spark per l'endpoint di sviluppo verranno eseguiti su 4 vCPU, 16 GB di memoria e un disco da 64 GB.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  La versione Glue determina le versioni di Apache Spark e Python supportate da AWS Glue. La versione Python indica la versione supportata per l'esecuzione degli script ETL sugli endpoint di sviluppo.

  Per ulteriori informazioni sulle versioni di AWS Glue disponibili e sulle versioni di Spark e Python corrispondenti, consulta la sezione relativa alla versione Glue nella guida per gli sviluppatori.

  Endpoint di sviluppo creati senza specificare una versione Glue impostata in modo predefinito su Glue 0.9.

  Puoi specificare una versione del supporto Python per gli endpoint di sviluppo utilizzando il parametro `Arguments` nelle API `UpdateDevEndpoint` o `CreateDevEndpoint`. Se non vengono forniti argomenti, per impostazione predefinita la versione è Python 2.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` definito allocati all'endpoint di sviluppo.

  Il numero massimo di worker che è possibile definire è 299 `G.1X` e 149 per `G.2X`.

- `NumberOfNodes`: numero (intero).

  Il numero di unità di elaborazione dati (DPU) di AWS Glue allocato per questo `DevEndpoint`.

- `AvailabilityZone`: stringa UTF-8.

La zona di disponibilità AWS in cui si trova `DevEndpoint`.

- `VpcId`: stringa UTF-8.

  L'ID del Virtual Private Cloud (VPC) utilizzato da questo `DevEndpoint`.

- `ExtraPythonLibsS3Path`: stringa UTF-8.

  Percorsi a una o più librerie Python in un bucket Amazon S3 che devono essere caricati nel `DevEndpoint`. I valori multipli devono essere percorsi completi separati da virgola.

  > **Note**
  >
  > Con un `DevEndpoint` è possibile utilizzare solo librerie Python pure. Le librerie che si basano sulle estensioni C, come la libreria di analisi dati Python pandas, non sono ancora supportate.

- `ExtraJarsS3Path`: stringa UTF-8.

  Percorsi a uno o più file `.jar` Java in un bucket S3 che devono essere caricati nel `DevEndpoint`.

  > **Note**
  >
  > Con un `DevEndpoint` è possibile utilizzare solo librerie Java/Scala pure.

- `FailureReason`: stringa UTF-8.

  Il motivo di un errore corrente in questo `DevEndpoint`.

- `LastUpdateStatus`: stringa UTF-8.

  Lo stato dell'ultimo aggiornamento.

- `CreatedTimestamp`: timestamp.

  Il momento in cui è stato creato il DevEndpoint.

- `LastModifiedTimestamp`: timestamp.

  Il momento dell'ultima modifica di questo `DevEndpoint`.

- `PublicKey`: stringa UTF-8.

La chiave pubblica che deve essere utilizzata da questo `DevEndpoint` per l'autenticazione. Questo attributo viene fornito per la compatibilità con le versioni precedenti, in quanto l'attributo consigliato da usare è quello delle chiavi pubbliche.

- `PublicKeys`: una matrice di stringhe UTF-8, non più di 5 stringhe.

  Elenco di chiavi pubbliche che devono essere utilizzate da `DevEndpoints` per l'autenticazione. L'uso di questo attributo è preferibile rispetto a una singola chiave pubblica, perché le chiavi pubbliche permettono di avere una chiave privata diversa per ogni client.

  > **Note**
  >
  > Se è già stato creato un endpoint con una chiave pubblica, è necessario rimuovere tale chiave per poter impostare un elenco di chiavi pubbliche. Chiama l'operazione API `UpdateDevEndpoint` con il contenuto della chiave pubblica nell'attributo `deletePublicKeys` e l'elenco delle nuove chiavi nell'attributo `addPublicKeys`.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da utilizzare con questo `DevEndpoint`.

- `Arguments` – Una matrice di mappe con coppie chiave-valore, non superiore alle 100 coppie.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Mappa di argomenti usati per configurare `DevEndpoint`.

  Gli argomenti validi sono:

  - `"--enable-glue-datacatalog": ""`

  Puoi specificare una versione del supporto Python per gli endpoint di sviluppo utilizzando il parametro `Arguments` nelle API `UpdateDevEndpoint` o `CreateDevEndpoint`. Se non vengono forniti argomenti, per impostazione predefinita la versione è Python 2.

Struttura DevEndpointCustomLibraries

Librerie personalizzate da caricare in un endpoint di sviluppo.

Campi

- `ExtraPythonLibsS3Path`: stringa UTF-8.

  I percorsi a una o più librerie Python in un bucket Amazon Simple Storage Service (Amazon S3) che devono essere caricati nel `DevEndpoint`. I valori multipli devono essere percorsi completi separati da virgola.

  > **Note**
  >
  > Con un `DevEndpoint` è possibile utilizzare solo librerie Python pure. Le librerie che si basano sulle estensioni C, come la libreria di analisi dati Python pandas, non sono ancora supportate.

- `ExtraJarsS3Path`: stringa UTF-8.

  Percorsi a uno o più file `.jar` Java in un bucket S3 che devono essere caricati nel `DevEndpoint`.

  > **Note**
  >
  > Con un `DevEndpoint` è possibile utilizzare solo librerie Java/Scala pure.

Operazioni

- Operazione CreateDevEndpoint (Python: create_dev_endpoint)
- Operazione UpdateDevEndpoint (Python: update_dev_endpoint)
- Operazione DeleteDevEndpoint (Python: delete_dev_endpoint)
- Operazione GetDevEndpoint (Python: get_dev_endpoint)
- Operazione GetDevEndpoints (Python: get_dev_endpoints)
- Operazione BatchGetDevEndpoints (Python: batch_get_dev_endpoints)
- Operazione ListDevEndpoints (Python: list_dev_endpoints)

Operazione CreateDevEndpoint (Python: create_dev_endpoint)

Crea un nuovo endpoint di sviluppo.

Richiesta

- `EndpointName`. Obbligatorio: stringa UTF-8.

  Il nome da assegnare al nuovo `DevEndpoint`.

- `RoleArn`: obbligatorio: stringa UTF-8, corrispondente a AWS IAM ARN string pattern.

  Il ruolo IAM per il `DevEndpoint`.

- `SecurityGroupIds`: una matrice di stringhe UTF-8.

  Gli ID del gruppo di sicurezza per i gruppi di sicurezza che devono essere utilizzati dal nuovo `DevEndpoint`.

- `SubnetId`: stringa UTF-8.

  La sottorete ID per il nuovo `DevEndpoint` da utilizzare.

- `PublicKey`: stringa UTF-8.

  La chiave pubblica che deve essere utilizzata da questo `DevEndpoint` per l'autenticazione. Questo attributo viene fornito per la compatibilità con le versioni precedenti, in quanto l'attributo consigliato da usare è quello delle chiavi pubbliche.

- `PublicKeys`: una matrice di stringhe UTF-8, non più di 5 stringhe.

  Elenco di chiavi pubbliche che devono essere usate dagli endpoint di sviluppo per l'autenticazione. L'uso di questo attributo è preferibile rispetto a una singola chiave pubblica, perché le chiavi pubbliche permettono di avere una chiave privata diversa per ogni client.

  > **Note**
  >
  > Se è già stato creato un endpoint con una chiave pubblica, è necessario rimuovere tale chiave per poter impostare un elenco di chiavi pubbliche. Chiama l'API `UpdateDevEndpoint` con il contenuto della chiave pubblica nell'attributo `deletePublicKeys` e l'elenco delle nuove chiavi nell'attributo `addPublicKeys`.

- `NumberOfNodes`: numero (intero).

  Il numero di unità di elaborazione dati (DPU) di AWS Glue da allocare per questo `DevEndpoint`.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

Il tipo di worker predefinito allocato all'endpoint di sviluppo. Accetta un valore Standard, G.1X o G.2X.

- Per il tipo di worker `Standard`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 50 GB e 2 esecutori.
- Per il tipo di worker `G.1X`, ciascun worker si mappa a 1 DPU (4 vCPU, 16 GB di memoria, disco da 64 GB) e fornisce 1 esecutore. Consigliamo questo tipo di worker per i processi ad alto consumo di memoria.
- Per il tipo di worker `G.2X`, ciascun worker si mappa a 2 DPU (8 vCPU, 32 GB di memoria, disco da 128 GB) e fornisce 1 esecutore. Consigliamo questo tipo di worker per i processi ad alto consumo di memoria.

Problema noto: quando viene creato un endpoint di sviluppo con la configurazione `G.2X` `WorkerType`, i driver Spark per l'endpoint di sviluppo verranno eseguiti su 4 vCPU, 16 GB di memoria e un disco da 64 GB.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  La versione Glue determina le versioni di Apache Spark e Python supportate da AWS Glue. La versione Python indica la versione supportata per l'esecuzione degli script ETL sugli endpoint di sviluppo.

  Per ulteriori informazioni sulle versioni di AWS Glue disponibili e sulle versioni di Spark e Python corrispondenti, consulta la sezione relativa alla versione Glue nella guida per gli sviluppatori.

  Endpoint di sviluppo creati senza specificare una versione Glue impostata in modo predefinito su Glue 0.9.

  Puoi specificare una versione del supporto Python per gli endpoint di sviluppo utilizzando il parametro `Arguments` nelle API `UpdateDevEndpoint` o `CreateDevEndpoint`. Se non vengono forniti argomenti, per impostazione predefinita la versione è Python 2.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` definito allocati all'endpoint di sviluppo.

  Il numero massimo di worker che è possibile definire è 299 `G.1X` e 149 per `G.2X`.

- `ExtraPythonLibsS3Path`: stringa UTF-8.

Percorsi a una o più librerie Python in un bucket Amazon S3 che devono essere caricati nel `DevEndpoint`. I valori multipli devono essere percorsi completi separati da virgola.

> **Note**
>
> Con un `DevEndpoint` è possibile utilizzare solo librerie Python pure. Le librerie che si basano sulle estensioni C, come la libreria di analisi dati Python pandas, non sono ancora supportate.

- `ExtraJarsS3Path`: stringa UTF-8.

  Percorsi a uno o più file `.jar` Java in un bucket S3 che devono essere caricati nel `DevEndpoint`.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da utilizzare con questo `DevEndpoint`.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag da usare con questo DevEndpoint. Puoi usare i tag per limitare l'accesso al DevEndpoint. Per ulteriori informazioni sui tag in AWS Glue, consulta Tag AWS in AWS Glue nella guida per gli sviluppatori.

- `Arguments` – Una matrice di mappe con coppie chiave-valore, non superiore alle 100 coppie.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Mappa di argomenti usati per configurare `DevEndpoint`.

Risposta

- `EndpointName`: stringa UTF-8.

  Il nome assegnato al nuovo `DevEndpoint`.

- `Status`: stringa UTF-8.

  Lo stato corrente del nuovo `DevEndpoint`.

- `SecurityGroupIds`: una matrice di stringhe UTF-8.

  I gruppi di sicurezza assegnati al nuovo `DevEndpoint`.

- `SubnetId`: stringa UTF-8.

  L'ID di sottorete assegnato al nuovo `DevEndpoint`.

- `RoleArn`: stringa UTF-8, corrispondente a AWS IAM ARN string pattern.

  L'Amazon Resource Name (ARN) del ruolo assegnato al nuovo `DevEndpoint`.

- `YarnEndpointAddress`: stringa UTF-8.

  L'indirizzo dell'endpoint YARN utilizzato da questo `DevEndpoint`.

- `ZeppelinRemoteSparkInterpreterPort`: numero (intero).

  La porta Apache Zeppelin per l'interprete Apache Spark remoto.

- `NumberOfNodes`: numero (intero).

  Il numero di unità di elaborazione dati (DPU) di AWS Glue allocato per questo DevEndpoint.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato all'endpoint di sviluppo. Può essere un valore Standard, G.1X o G.2X.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  La versione Glue determina le versioni di Apache Spark e Python supportate da AWS Glue. La versione Python indica la versione supportata per l'esecuzione degli script ETL sugli endpoint di sviluppo.

  Per ulteriori informazioni sulle versioni di AWS Glue disponibili e sulle versioni di Spark e Python corrispondenti, consulta la sezione relativa alla versione Glue nella guida per gli sviluppatori.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` definito allocati all'endpoint di sviluppo.

- `AvailabilityZone`: stringa UTF-8.

  La zona di disponibilità AWS in cui si trova `DevEndpoint`.

- `VpcId`: stringa UTF-8.

  L'ID del Virtual Private Cloud (VPC) utilizzato da questo `DevEndpoint`.

- `ExtraPythonLibsS3Path`: stringa UTF-8.

  Percorsi a una o più librerie Python in un bucket S3 che verranno caricati nel `DevEndpoint`.

- `ExtraJarsS3Path`: stringa UTF-8.

  Percorsi a uno o più file `.jar` Java in un bucket S3 che devono essere caricati nel `DevEndpoint`.

- `FailureReason`: stringa UTF-8.

  Il motivo di un errore corrente in questo `DevEndpoint`.

- `SecurityConfiguration`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della struttura `SecurityConfiguration` da utilizzare con questo `DevEndpoint`.

- `CreatedTimestamp`: timestamp.

  Il momento in cui questo `DevEndpoint` è stato creato.

- `Arguments` – Una matrice di mappe con coppie chiave-valore, non superiore alle 100 coppie.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  Mappa di argomenti usati per configurare questo `DevEndpoint`.

  Gli argomenti validi sono:

  - `"--enable-glue-datacatalog": ""`

  Puoi specificare una versione del supporto Python per gli endpoint di sviluppo utilizzando il parametro `Arguments` nelle API `UpdateDevEndpoint` o `CreateDevEndpoint`. Se non vengono forniti argomenti, per impostazione predefinita la versione è Python 2.

Errori

- `AccessDeniedException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `ResourceNumberLimitExceededException`

Operazione UpdateDevEndpoint (Python: update_dev_endpoint)

Aggiorna un endpoint di sviluppo specificato.

Richiesta

- `EndpointName`. Obbligatorio: stringa UTF-8.

  Nome del `DevEndpoint` da aggiornare.
- `PublicKey`: stringa UTF-8.

  La chiave pubblica che deve essere utilizzata da `DevEndpoint`.
- `AddPublicKeys`: una matrice di stringhe UTF-8, non più di 5 stringhe.

  L'elenco delle chiavi pubbliche che devono essere utilizzate da `DevEndpoint`.
- `DeletePublicKeys`: una matrice di stringhe UTF-8, non più di 5 stringhe.

  Elenco di chiavi pubbliche da eliminare da `DevEndpoint`.
- `CustomLibraries`: un oggetto DevEndpointCustomLibraries.

  Librerie Python o Java personalizzate da caricare nel `DevEndpoint`.
- `UpdateEtlLibraries`: booleano.

  `True` se l'elenco di librerie personalizzate da caricare nell'endpoint di sviluppo deve essere aggiornato, in caso contrario `False`.
- `DeleteArguments`: una matrice di stringhe UTF-8.

L'elenco delle chiavi di argomento da eliminare dalla mappa di argomenti utilizzati per configurare il `DevEndpoint`.

- `AddArguments` – Una matrice di mappe con coppie chiave-valore, non superiore alle 100 coppie.

  Ogni chiave è una stringa UTF-8.

  Ogni valore è una stringa UTF-8.

  La mappa di argomenti da aggiungere alla mappa di argomenti utilizzati per configurare il `DevEndpoint`.

  Gli argomenti validi sono:

  - `"--enable-glue-datacatalog": ""`

  Puoi specificare una versione del supporto Python per gli endpoint di sviluppo utilizzando il parametro `Arguments` nelle API `UpdateDevEndpoint` o `CreateDevEndpoint`. Se non vengono forniti argomenti, per impostazione predefinita la versione è Python 2.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`

Operazione DeleteDevEndpoint (Python: delete_dev_endpoint)

Elimina un endpoint di sviluppo specificato.

Richiesta

- `EndpointName`. Obbligatorio: stringa UTF-8.

Nome della `DevEndpoint`.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Operazione GetDevEndpoint (Python: get_dev_endpoint)

Recupera informazioni su un endpoint di sviluppo specificato.

> **Note**
>
> Quando viene creato un endpoint di sviluppo in un virtual private cloud (VPC), AWS Glue restituisce solo un indirizzo IP privato e il campo dell'indirizzo IP pubblico non è popolato. Quando crei un endpoint non di sviluppo non VPC, AWS Glue restituisce solo un indirizzo IP pubblico.

Richiesta

- `EndpointName`. Obbligatorio: stringa UTF-8.

  Nome del `DevEndpoint` per cui recuperare le informazioni.

Risposta

- `DevEndpoint`: un oggetto DevEndpoint.

  Una definizione del `DevEndpoint`.

Errori

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Operazione GetDevEndpoints (Python: get_dev_endpoints)

Recupera tutti gli endpoint di sviluppo in questo account AWS.

> **Note**
>
> Quando viene creato un endpoint di sviluppo in un virtual private cloud (VPC), AWS Glue restituisce solo un indirizzo IP pubblico e il campo dell'indirizzo IP pubblico non è popolato. Quando crei un endpoint non di sviluppo non VPC, AWS Glue restituisce solo un indirizzo IP pubblico.

Richiesta

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima di informazioni da restituire.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `DevEndpoints`: una matrice di oggetti DevEndpoint.

  Un elenco di definizioni di `DevEndpoint`.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non sono ancora state restituite tutte le definizioni di `DevEndpoint`.

Errori

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Operazione BatchGetDevEndpoints (Python: batch_get_dev_endpoints)

Restituisce un elenco di metadati di risorse per un elenco di nomi di endpoint di sviluppo. Dopo aver chiamato l'operazione `ListDevEndpoints`, puoi chiamare questa operazione per accedere ai dati a cui sono state concesse le autorizzazioni. Questa operazione supporta tutte le autorizzazioni IAM, tra cui le condizioni di autorizzazione che utilizzano i tag.

Richiesta

- `customerAccountId`: stringa UTF-8.

  ID dell'account AWS.
- `DevEndpointNames`. Obbligatorio: una serie di stringhe UTF-8, non inferiore a 1 o superiore a 25 stringhe.

  L'elenco dei nomi di `DevEndpoint` che potrebbero essere i nomi restituiti dall'operazione `ListDevEndpoint`.

Risposta

- `DevEndpoints`: una matrice di oggetti DevEndpoint.

  Un elenco di definizioni di `DevEndpoint`.
- `DevEndpointsNotFound` – Una serie di stringhe UTF-8, non inferiore a 1 o superiore a 25 stringhe.

  Un elenco di `DevEndpoints` non trovati.

Errori

- `AccessDeniedException`

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Operazione ListDevEndpoints (Python: list_dev_endpoints)

Recupera i nomi di tutte le risorse `DevEndpoint` in questo account AWS oppure le risorse con il tag specificato. Questa operazione consente di vedere quali risorse sono disponibili nel proprio account e i relativi nomi.

L'operazione accetta il campo facoltativo `Tags` che si può utilizzare come filtro per la risposta in modo che le risorse con tag possano essere recuperate come gruppo. Se si sceglie di utilizzare il filtro dei tag, potranno essere recuperate solo le risorse con tag.

Richiesta

- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima di un elenco da restituire.
- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Specifica che vengono restituite solo le risorse con tag.

Risposta

- `DevEndpointNames`: una matrice di stringhe UTF-8.

  I nomi di tutti i `DevEndpoint` nell'account oppure i `DevEndpoint` con i tag specificati.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se l'elenco restituito non contiene l'ultimo parametro disponibile.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Registro degli schemi

L'API del registro Schema descrive i tipi di dati e le API relativi all'utilizzo degli schemi in AWS Glue.

Tipi di dati

- RegistryId struttura
- RegistryListItem struttura
- MetadataInfo struttura
- OtherMetadataValueListItem struttura
- SchemaListItem struttura
- SchemaVersionListItem struttura
- MetadataKeyValuePair struttura
- SchemaVersionErrorItem struttura
- ErrorDetails struttura
- SchemaVersionNumber struttura
- SchemaId struttura

RegistryId struttura

Una struttura wrapper che può contenere il nome del registro e l'Amazon Resource Name (ARN).

Campi

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

Nome del registro. Utilizzato solo per la ricerca. Deve essere fornito `RegistryArn` o `RegistryName`.

- `RegistryArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  Arn del registro da aggiornare. Deve essere fornito `RegistryArn` o `RegistryName`.

RegistryListItem struttura

Una struttura contenente i dettagli di un registro.

Campi

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro.

- `RegistryArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) del registro.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del registro.

- `Status`: stringa UTF-8 (valori validi: `AVAILABLE` | `DELETING`).

  Lo stato del registro.

- `CreatedTime`: stringa UTF-8.

  La data in cui è stato creato il registro.

- `UpdatedTime`: stringa UTF-8.

  La data in cui è stato aggiornato il registro.

MetadataInfo struttura

Una struttura contenente informazioni sui metadati per una versione dello schema.

Campi

- `MetadataValue`: stringa UTF-8, non inferiore a 1 o superiore a 256 byte di lunghezza, corrispondente a Custom string pattern #27.

  Il valore corrispondente alla chiave di metadati.
- `CreatedTime`: stringa UTF-8.

  L'ora in cui è stata creata la voce.
- `OtherMetadataValueList`: una matrice di oggetti OtherMetadataValueListItem.

  Altri metadati appartenenti alla stessa chiave di metadati.

OtherMetadataValueListItem struttura

Struttura contenente altri metadati per una versione dello schema appartenente alla stessa chiave di metadati.

Campi

- `MetadataValue` – stringa UTF-8, non inferiore a 1 o superiore a 256 byte di lunghezza, corrispondente a Custom string pattern #27.

  Il valore corrispondente della chiave di metadati per gli altri metadati appartenenti alla stessa chiave dei metadati.
- `CreatedTime`: stringa UTF-8.

  L'ora in cui è stata creata la voce.

SchemaListItem struttura

Un oggetto che contiene dettagli minimi per uno schema.

Campi

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro in cui si trova lo schema.

- `SchemaName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome dello schema.

- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) per lo schema.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione per lo schema.

- `SchemaStatus`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `DELETING`).

  Lo stato dello schema.

- `CreatedTime`: stringa UTF-8.

  La data e l'ora di creazione dello schema.

- `UpdatedTime`: stringa UTF-8.

  La data e l'ora di aggiornamento dello schema.

SchemaVersionListItem struttura

Oggetto contenente i dettagli relativi a una versione dello schema.

Campi

- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) dello schema.

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'identificatore univoco della versione dello schema.
- `VersionNumber` – Numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione dello schema.
- `Status`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  Lo stato della versione dello schema.
- `CreatedTime`: stringa UTF-8.

  La data e l'ora in cui è stata creata la versione dello schema.

MetadataKeyValuePair struttura

La struttura contenente una coppia chiave-valore per i metadati.

Campi

- `MetadataKey`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #27.

  Una chiave di metadati.
- `MetadataValue` – stringa UTF-8, non inferiore a 1 o superiore a 256 byte di lunghezza, corrispondente a Custom string pattern #27.

  Il valore corrispondente a una chiave di metadati.

SchemaVersionErrorItem struttura

Oggetto che contiene i dettagli di errore per un'operazione su una versione dello schema.

Campi

- `VersionNumber`: numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione dello schema.
- `ErrorDetails`: un oggetto ErrorDetails.

I dettagli dell'errore per la versione dello schema.

ErrorDetails struttura

Un oggetto contenente dettagli di errore.

Campi

- `ErrorCode`: stringa UTF-8.

  Il codice di errore per un errore.
- `ErrorMessage`: stringa UTF-8.

  Il messaggio di errore relativo a un errore.

SchemaVersionNumber struttura

Una struttura contenente le informazioni sulla versione dello schema.

Campi

- `LatestVersion`: booleano.

  La versione più recente disponibile per lo schema.
- `VersionNumber` – Numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione dello schema.

SchemaId struttura

L'ID univoco dello schema nel registro degli AWS Glue schemi.

Campi

- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) dello schema. Deve essere fornito `SchemaArn` o `SchemaName`.

- `SchemaName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome dello schema. Deve essere fornito `SchemaArn` o `SchemaName`.

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro degli schemi che contiene lo schema.

Operazioni

- CreateRegistry azione (Python: create_registry)
- CreateSchema azione (Python: create_schema)
- GetSchema azione (Python: get_schema)
- ListSchemaVersions azione (Python: list_schema_versions)
- GetSchemaVersion azione (Python: get_schema_version)
- GetSchemaVersionsDiff azione (Python: get_schema_versions_diff)
- ListRegistries azione (Python: list_registries)
- ListSchemas azione (Python: list_schemas)
- RegisterSchemaVersion azione (Python: register_schema_version)
- UpdateSchema azione (Python: update_schema)
- CheckSchemaVersionValidity azione (Python: check_schema_version_idity)
- UpdateRegistry azione (Python: update_registry)
- GetSchemaByDefinition azione (Python: get_schema_by_definition)
- GetRegistry azione (Python: get_registry)
- PutSchemaVersionMetadata azione (Python: put_schema_version_metadata)
- QuerySchemaVersionMetadata azione (Python: query_schema_version_metadata)
- RemoveSchemaVersionMetadata azione (Python: remove_schema_version_metadata)
- DeleteRegistry azione (Python: delete_registry)
- DeleteSchema azione (Python: delete_schema)
- DeleteSchemaVersions azione (Python: delete_schema_versions)

CreateRegistry azione (Python: create_registry)

Crea un nuovo registro che può essere utilizzato per contenere una raccolta di schemi.

Richiesta

- `RegistryName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro da creare, che può avere una lunghezza massima di 255 caratteri e può contenere solo lettere, numeri, trattino, trattino basso, simbolo del dollaro o cancelletto. Non sono ammessi spazi.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del registro. Se la descrizione non viene fornita, non ci sarà alcun valore di default.
- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  AWS tag che contengono una coppia chiave-valore e possono essere ricercati tramite console, riga di comando o API.

Risposta

- `RegistryArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'Amazon Resource Name (ARN) del registro appena creato.
- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del registro.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag per il registro.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

CreateSchema azione (Python: create_schema)

Crea un nuovo set di schemi e registra la definizione dello schema. Restituisce un errore se il set di schemi esiste già senza registrare effettivamente la versione.

Quando viene creato il set di schemi, un checkpoint della versione verrà impostato sulla prima versione. La modalità di compatibilità "DISABLED" limita l'aggiunta di eventuali versioni aggiuntive dello schema dopo la prima versione. Per tutte le altre modalità di compatibilità, la convalida delle impostazioni di compatibilità verrà applicata solo a partire dalla seconda versione quando viene utilizzata l'API `RegisterSchemaVersion`.

Quando questa API viene chiamata senza un `RegistryId`, verrà creata una voce per un "registro predefinito" nelle tabelle del database del registro, se non è già presente.

Richiesta

- `RegistryId`: un oggetto RegistryId.

  Si tratta di una forma wrapper che contiene i campi di identità del registro. Se non viene fornito, verrà utilizzato il registro predefinito. Il formato ARN per lo stesso sarà: `arn:aws:glue:us-east-2:<customer id>:registry/default-registry:random-5-letter-id`.

- `SchemaName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome dello schema da creare, che può avere una lunghezza massima di 255 caratteri e può contenere solo lettere, numeri, trattini, carattere di sottolineatura, simbolo del dollaro o segno di hash. Non sono ammessi spazi.

- `DataFormat`: obbligatorio: stringa UTF-8 (valori validi: `AVRO` | `JSON` | `PROTOBUF`).

  Il formato dei dati della definizione dello schema. Al momento sono supportati `AVRO`, `JSON` e `PROTOBUF`.

- `Compatibility`: stringa UTF-8 (valori validi: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  La modalità di compatibilità dello schema. I valori possibili sono:

  - NONE: non si applica alcuna modalità di compatibilità. È possibile utilizzare questa opzione negli scenari di sviluppo o se non si conosce la modalità di compatibilità da applicare agli schemi. Qualsiasi nuova versione aggiunta sarà accettata senza essere sottoposta a un controllo di compatibilità.
  - DISABLED: questa scelta di compatibilità impedisce il controllo delle versioni per uno schema specifico. È possibile utilizzare questa opzione per impedire il controllo delle versioni future di uno schema.
  - BACKWARD: questa scelta di compatibilità è consigliata in quanto consente ai ricevitori di dati di leggere sia la versione attuale che quella precedente dello schema. Ciò significa che, ad esempio, una nuova versione dello schema non può eliminare i campi dati o modificarne il tipo, in modo che non possano essere letti dai lettori che utilizzano la versione precedente.
  - BACKWARD_ALL:: questa scelta di compatibilità consente ai ricevitori di dati di leggere sia la versione attuale che tutte quelle precedenti dello schema. È possibile utilizzare questa opzione quando è necessario eliminare campi o aggiungere campi facoltativi e verificare la compatibilità con tutte le versioni precedenti dello schema.
  - FORWARD: questa scelta di compatibilità consente ai ricevitori di dati di leggere sia la versione attuale che una versione successiva dello schema, ma non necessariamente le versioni più recenti. È possibile utilizzare questa opzione quando è necessario aggiungere campi o eliminare campi facoltativi, ma solo verificare la compatibilità con l'ultima versione dello schema.
  - FORWARD_ALL: questa scelta di compatibilità consente ai ricevitori di dati di leggere scritti dai produttori di qualsiasi nuovo schema registrato. È possibile utilizzare questa opzione quando è

necessario aggiungere campi o eliminare campi facoltativi e verificare la compatibilità con tutte le versioni precedenti dello schema.

- FULL: questa scelta di compatibilità consente ai ricevitori di dati di leggere i dati scritti dai produttori utilizzando la versione precedente o successiva dello schema, ma non necessariamente versioni precedenti o successive. È possibile utilizzare questa opzione quando è necessario aggiungere o eliminare campi facoltativi, ma solo verificare la compatibilità con l'ultima versione dello schema.
- FULL_ALL: questa scelta di compatibilità consente ai ricevitori di dati di leggere i dati scritti dai produttori utilizzando tutte le versioni precedenti dello schema. È possibile utilizzare questa opzione quando è necessario aggiungere o eliminare campi facoltativi e verificare la compatibilità con tutte le versioni precedenti dello schema.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione facoltativa dello schema. Se la descrizione non viene fornita, non ci sarà alcun valore predefinito automatico.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  AWS tag che contengono una coppia chiave-valore e possono essere ricercati tramite console, riga di comando o API. Se specificato, segue lo AWS tags-on-create schema.

- `SchemaDefinition`: stringa UTF-8, non inferiore a 1 o superiore a 170000 byte di lunghezza, corrispondente a Custom string pattern #26.

  La definizione dello schema che utilizza l'impostazione `DataFormat` per `SchemaName`.

Risposta

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro.

- `RegistryArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

L'ARN (Amazon Resource Name) del registro.

- `SchemaName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome dello schema.

- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) dello schema.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione dello schema, se specificata al momento della creazione.

- `DataFormat`: stringa UTF-8 (valori validi: `AVRO` | `JSON` | `PROTOBUF`).

  Il formato dei dati della definizione dello schema. Al momento sono supportati `AVRO`, `JSON` e `PROTOBUF`.

- `Compatibility`: stringa UTF-8 (valori validi: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  La modalità di compatibilità dello schema.

- `SchemaCheckpoint` – Numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione del checkpoint (l'ultima volta che la modalità di compatibilità è stata modificata).

- `LatestSchemaVersion` – Numero (intero), non inferiore a 1 o superiore a 100000.

  La versione più recente dello schema associata alla definizione dello schema restituita.

- `NextSchemaVersion` – Numero (intero), non inferiore a 1 o superiore a 100000.

  La versione successiva dello schema associata alla definizione dello schema restituita.

- `SchemaStatus`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `DELETING`).

  Lo stato dello schema.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

I tag per lo schema.

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'identificatore univoco della prima versione dello schema.

- `SchemaVersionStatus`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  Lo stato della prima versione dello schema creata.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

GetSchema azione (Python: get_schema)

Descrive lo schema specificato nel dettaglio.

Richiesta

- `SchemaId`: obbligatorio: un oggetto SchemaId.

  Si tratta di una struttura wrapper che contiene i campi di identità dello schema. La struttura include:

  - SchemaId$SchemaArn: L'Amazon Resource Name (ARN) dello schema. Deve essere fornito `SchemaArn` oppure `SchemaName` e `RegistryName`.
  - SchemaId$SchemaName: il nome dello schema. Deve essere fornito `SchemaArn` oppure `SchemaName` e `RegistryName`.

Risposta

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro.
- `RegistryArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) del registro.
- `SchemaName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome dello schema.
- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) dello schema.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione dello schema, se specificata al momento della creazione
- `DataFormat`: stringa UTF-8 (valori validi: `AVRO` | `JSON` | `PROTOBUF`).

  Il formato dei dati della definizione dello schema. Al momento sono supportati `AVRO`, `JSON` e `PROTOBUF`.
- `Compatibility`: stringa UTF-8 (valori validi: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  La modalità di compatibilità dello schema.
- `SchemaCheckpoint` – Numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione del checkpoint (l'ultima volta che la modalità di compatibilità è stata modificata).
- `LatestSchemaVersion` – Numero (intero), non inferiore a 1 o superiore a 100000.

  La versione più recente dello schema associata alla definizione dello schema restituita.
- `NextSchemaVersion` – Numero (intero), non inferiore a 1 o superiore a 100000.

La versione successiva dello schema associata alla definizione dello schema restituita.

- `SchemaStatus`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `DELETING`).

  Lo stato dello schema.

- `CreatedTime`: stringa UTF-8.

  La data e l'ora di creazione dello schema.

- `UpdatedTime`: stringa UTF-8.

  La data e l'ora di aggiornamento dello schema.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

ListSchemaVersions azione (Python: list_schema_versions)

Restituisce un elenco delle versioni dello schema create dall'utente, con informazioni minime. Le versioni dello schema nello stato Deleted non verranno incluse nei risultati. Se non sono disponibili versioni dello schema, verranno restituiti risultati vuoti.

Richiesta

- `SchemaId`: obbligatorio: un oggetto SchemaId.

  Si tratta di una struttura wrapper che contiene i campi di identità dello schema. La struttura include:

  - SchemaId$SchemaArn: L'Amazon Resource Name (ARN) dello schema. Deve essere fornito `SchemaArn` oppure `SchemaName` e `RegistryName`.
  - SchemaId$SchemaName: il nome dello schema. Deve essere fornito `SchemaArn` oppure `SchemaName` e `RegistryName`.

- `MaxResults` – Numero (intero), non inferiore a 1 o superiore a 100.

Numero massimo di risultati richiesti per pagina. Se il valore non viene fornito, sarà impostato in modo predefinito su 25 per pagina.

- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `Schemas`: una matrice di oggetti SchemaVersionListItem.

  Una matrice di oggetti `SchemaVersionList` contenenti i dettagli di ogni versione dello schema.
- `NextToken`: stringa UTF-8.

  Un token di continuazione per impaginare l'elenco restituito di token, restituiti se il segmento corrente dell'elenco non è l'ultimo.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetSchemaVersion azione (Python: get_schema_version)

Ottiene lo schema specificato in base al relativo ID univoco assegnato alla creazione o alla registrazione di una versione dello schema. Le versioni dello schema nello stato Deleted non verranno incluse nei risultati.

Richiesta

- `SchemaId`: un oggetto SchemaId.

  Si tratta di una struttura wrapper che contiene i campi di identità dello schema. La struttura include:

  - SchemaId$SchemaArn: L'Amazon Resource Name (ARN) dello schema. Deve essere fornito `SchemaArn` oppure `SchemaName` e `RegistryName`.

  - SchemaId$SchemaName: il nome dello schema. Deve essere fornito `SchemaArn` oppure `SchemaName` e `RegistryName`.
- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  La versione `SchemaVersionId` dello schema. Questo campo è obbligatorio per il recupero in base all'ID dello schema. Deve essere fornito questo o il wrapper `SchemaId`.
- `SchemaVersionNumber`: un oggetto SchemaVersionNumber.

  Il numero di versione dello schema.

Risposta

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  La versione `SchemaVersionId` dello schema.
- `SchemaDefinition` – stringa UTF-8, non inferiore a 1 o superiore a 170000 byte di lunghezza, corrispondente a Custom string pattern #26.

  La definizione dello schema per l'ID dello schema.
- `DataFormat`: stringa UTF-8 (valori validi: `AVRO` | `JSON` | `PROTOBUF`).

  Il formato dei dati della definizione dello schema. Al momento sono supportati `AVRO`, `JSON` e `PROTOBUF`.
- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) dello schema.
- `VersionNumber` – Numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione dello schema.
- `Status`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  Lo stato della versione dello schema.
- `CreatedTime`: stringa UTF-8.

  La data e l'ora in cui è stata creata la versione dello schema.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetSchemaVersionsDiff azione (Python: get_schema_versions_diff)

Recupera la differenza di versione dello schema nel tipo di differenza specificato tra due versioni dello schema archiviate nel registro degli schemi.

Questa API consente di confrontare due versioni dello schema tra due definizioni dello schema nello stesso schema.

Richiesta

- `SchemaId`: obbligatorio: un oggetto SchemaId.

  Si tratta di una struttura wrapper che contiene i campi di identità dello schema. La struttura include:

  - SchemaId$SchemaArn: L'Amazon Resource Name (ARN) dello schema. Deve essere fornito `SchemaArn` o `SchemaName`.
  - SchemaId$SchemaName: il nome dello schema. Deve essere fornito `SchemaArn` o `SchemaName`.
- `FirstSchemaVersionNumber`: obbligatorio: un oggetto SchemaVersionNumber.

  La prima delle due versioni dello schema da confrontare.
- `SecondSchemaVersionNumber`: obbligatorio: un oggetto SchemaVersionNumber.

  La seconda delle due versioni dello schema da confrontare.
- `SchemaDiffType`. Obbligatorio: stringa UTF-8 (valori validi: `SYNTAX_DIFF`).

  Si riferisce a `SYNTAX_DIFF`, che è il tipo di diff attualmente supportato.

Risposta

- `Diff`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 340000 byte di lunghezza, corrispondente a Custom string pattern #26.

  La differenza tra gli schemi come stringa in JsonPatch formato.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`

ListRegistries azione (Python: list_registries)

Restituisce un elenco di registri dall'utente, con informazioni minime. I registri nello stato `Deleting` non verranno inclusi nei risultati. Se non sono disponibili registri, verranno restituiti risultati vuoti.

Richiesta

- `MaxResults` – Numero (intero), non inferiore a 1 o superiore a 100.

  Numero massimo di risultati richiesti per pagina. Se il valore non viene fornito, sarà impostato in modo predefinito su 25 per pagina.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `Registries`: una matrice di oggetti RegistryListItem.

  Una matrice di oggetti `RegistryDetailedListItem` contenenti dettagli minimi di ogni registro.
- `NextToken`: stringa UTF-8.

  Un token di continuazione per impaginare l'elenco restituito di token, restituiti se il segmento corrente dell'elenco non è l'ultimo.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

ListSchemas azione (Python: list_schemas)

Restituisce un elenco di schemi con dettagli minimi. Gli schemi nello stato Deleting non verranno inclusi nei risultati. Se non sono disponibili schemi, verranno restituiti risultati vuoti.

Quando il `RegistryId` non viene fornito, tutti gli schemi nei registri faranno parte della risposta dell'API.

Richiesta

- `RegistryId`: un oggetto RegistryId.

  Una struttura wrapper che può contenere il nome del registro e l'Amazon Resource Name (ARN).
- `MaxResults` – Numero (intero), non inferiore a 1 o superiore a 100.

  Numero massimo di risultati richiesti per pagina. Se il valore non viene fornito, sarà impostato in modo predefinito su 25 per pagina.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `Schemas`: una matrice di oggetti SchemaListItem.

  Una matrice di oggetti `SchemaListItem` contenenti i dettagli di ogni schema.
- `NextToken`: stringa UTF-8.

  Un token di continuazione per impaginare l'elenco restituito di token, restituiti se il segmento corrente dell'elenco non è l'ultimo.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

RegisterSchemaVersion azione (Python: register_schema_version)

Aggiunge una nuova versione allo schema esistente. Restituisce un errore se la nuova versione dello schema non soddisfa i requisiti di compatibilità del set di schemi. Questa API non creerà un nuovo set di schemi e restituirà un errore 404 se il set di schemi non è già presente nel registro degli schemi.

Se si tratta della prima definizione dello schema da registrare nel registro degli schemi, questa API archivierà la versione dello schema e restituirà immediatamente. In caso contrario, l'esecuzione di questa chiamata potrebbe durare più a lungo rispetto ad altre operazioni a causa delle modalità di compatibilità. È possibile chiamare l'API `GetSchemaVersion` con `SchemaVersionId` per controllare le modalità di compatibilità.

Se la stessa definizione di schema è già archiviata nel registro degli schemi come versione, viene restituito al chiamante l'ID dello schema esistente.

Richiesta

- `SchemaId`: obbligatorio: un oggetto SchemaId.

  Si tratta di una struttura wrapper che contiene i campi di identità dello schema. La struttura include:

  - SchemaId$SchemaArn: L'Amazon Resource Name (ARN) dello schema. Deve essere fornito `SchemaArn` oppure `SchemaName` e `RegistryName`.
  - SchemaId$SchemaName: il nome dello schema. Deve essere fornito `SchemaArn` oppure `SchemaName` e `RegistryName`.
- `SchemaDefinition`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 170000 byte di lunghezza, corrispondente a Custom string pattern #26.

  La definizione dello schema che utilizza l'impostazione `DataFormat` per `SchemaName`.

Risposta

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID univoco che rappresenta la versione di questo schema.
- `VersionNumber` – Numero (intero), non inferiore a 1 o superiore a 100000.

  La versione di questo schema (solo per flusso di sincronizzazione, se si tratta della prima versione).
- `Status`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  Lo stato della versione dello schema.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

UpdateSchema azione (Python: update_schema)

Aggiorna la descrizione, l'impostazione di compatibilità o il checkpoint della versione per un set di schemi.

Per aggiornare l'impostazione di compatibilità, la chiamata non convaliderà la compatibilità per l'intero set di versioni dello schema con la nuova impostazione di compatibilità. Se il valore per `Compatibility` viene fornito, è necessario fornire anche `VersionNumber` (un checkpoint). L'API convaliderà il numero di versione del checkpoint per garantire coerenza.

Se il valore per `VersionNumber` (checkpoint) è fornito, `Compatibility` è facoltativo e può essere usato per impostare/reimpostare un checkpoint per lo schema.

Questo aggiornamento verrà eseguito solo se lo schema si trova nello stato AVAILABLE.

Richiesta

- `SchemaId`: obbligatorio: un oggetto SchemaId.

  Si tratta di una struttura wrapper che contiene i campi di identità dello schema. La struttura include:

  - SchemaId$SchemaArn: L'Amazon Resource Name (ARN) dello schema. Deve essere fornito `SchemaArn` o `SchemaName`.
  - SchemaId$SchemaName: il nome dello schema. Deve essere fornito `SchemaArn` o `SchemaName`.
- `SchemaVersionNumber`: un oggetto SchemaVersionNumber.

  Numero di versione richiesto per il checkpoint. Deve essere fornito `VersionNumber` o `Compatibility`.
- `Compatibility`: stringa UTF-8 (valori validi: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  La nuova impostazione di compatibilità per lo schema.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La nuova descrizione per lo schema.

Risposta

- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) dello schema.
- `SchemaName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome dello schema.
- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro che contiene lo schema.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InternalServiceException`

CheckSchemaVersionValidity azione (Python: check_schema_version_idity)

Convalida lo schema fornito. Questa chiamata non ha effetti collaterali, convalida semplicemente usando lo schema fornito, utilizzando `DataFormat` come formato. Poiché non prevede un nome di set di schemi, non vengono eseguiti controlli di compatibilità.

Richiesta

- `DataFormat`: obbligatorio: stringa UTF-8 (valori validi: `AVRO` | `JSON` | `PROTOBUF`).

  Il formato dei dati della definizione dello schema. Al momento sono supportati `AVRO`, `JSON` e `PROTOBUF`.
- `SchemaDefinition` – Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 170000 byte di lunghezza, corrispondente a Custom string pattern #26.

  La definizione dello schema che deve essere convalidato.

Risposta

- `Valid`: booleano.

  Se lo schema è valido restituisce true, in caso contrario false.
- `Error`: stringa UTF-8, non inferiore a 1 o superiore a 5000 byte di lunghezza.

  Un messaggio di errore di convalida.

Errori

- `InvalidInputException`

- `AccessDeniedException`
- `InternalServiceException`

UpdateRegistry azione (Python: update_registry)

Aggiorna un registro esistente che viene utilizzato per contenere una raccolta di schemi. Le proprietà aggiornate si riferiscono al registro e non modificano gli schemi all'interno del registro.

Richiesta

- `RegistryId`: obbligatorio: un oggetto RegistryId.

  Si tratta di una struttura wrapper che può contenere il nome del registro e l'Amazon Resource Name (ARN).
- `Description`: obbligatorio: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del registro. Se la descrizione non viene fornita, questo campo non verrà aggiornato.

Risposta

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro aggiornato.
- `RegistryArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'Amazon Resource name (ARN) del registro aggiornato.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ConcurrentModificationException`

- `InternalServiceException`

GetSchemaByDefinition azione (Python: get_schema_by_definition)

Recupera uno schema mediante la `SchemaDefinition`. La definizione dello schema viene inviata al registro degli schemi, canonicalizzata e sottoposta ad hashing. Se l'hash è abbinato nell'ambito del `SchemaName` o ARN (o il registro di default, se non viene fornito), vengono restituiti i metadati dello schema. In caso contrario, viene restituito un errore 404 o. NotFound Le versioni dello schema nello stato `Deleted` non verranno incluse nei risultati.

Richiesta

- `SchemaId`: obbligatorio: un oggetto SchemaId.

  Si tratta di una struttura wrapper che contiene i campi di identità dello schema. La struttura include:

  - SchemaId$SchemaArn: L'Amazon Resource Name (ARN) dello schema. Deve essere fornito `SchemaArn` o `SchemaName`.
  - SchemaId$SchemaName: il nome dello schema. Deve essere fornito `SchemaArn` o `SchemaName`.
- `SchemaDefinition`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 170000 byte di lunghezza, corrispondente a Custom string pattern #26.

  La definizione dello schema per il quale sono necessari i dettagli dello schema.

Risposta

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID dello schema della versione dello schema.
- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) dello schema.
- `DataFormat`: stringa UTF-8 (valori validi: `AVRO` | `JSON` | `PROTOBUF`).

  Il formato dei dati della definizione dello schema. Al momento sono supportati `AVRO`, `JSON` e `PROTOBUF`.

- `Status`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  Lo stato della versione dello schema.

- `CreatedTime`: stringa UTF-8.

  La data e l'ora di creazione dello schema.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetRegistry azione (Python: get_registry)

Descrive il registro specificato nel dettaglio.

Richiesta

- `RegistryId`: obbligatorio: un oggetto RegistryId.

  Si tratta di una struttura wrapper che può contenere il nome del registro e l'Amazon Resource Name (ARN).

Risposta

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro.

- `RegistryArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) del registro.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

Una descrizione del registro.

- `Status`: stringa UTF-8 (valori validi: `AVAILABLE` | `DELETING`).

  Lo stato del registro.

- `CreatedTime`: stringa UTF-8.

  La data e l'ora di creazione del registro.

- `UpdatedTime`: stringa UTF-8.

  La data e l'ora di aggiornamento del registro.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

PutSchemaVersionMetadata azione (Python: put_schema_version_metadata)

Inserisce la coppia chiave-valore dei metadati per un ID versione dello schema specificato. Sarà consentito un massimo di 10 coppie chiave-valore per ciascuna versione dello schema. Possono essere aggiunte su una o più chiamate.

Richiesta

- `SchemaId`: un oggetto SchemaId.

  L'ID univoco dello schema.

- `SchemaVersionNumber`: un oggetto SchemaVersionNumber.

  Il numero di versione dello schema.

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

L'ID versione univoco della versione dello schema.

- `MetadataKeyValue`: obbligatorio: un oggetto MetadataKeyValuePair.

  Il valore corrispondente a una chiave di metadati.

Risposta

- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) per lo schema.

- `SchemaName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome per lo schema.

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome per il registro.

- `LatestVersion`: booleano.

  La versione più recente dello schema.

- `VersionNumber` – Numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione dello schema.

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID versione univoco della versione dello schema.

- `MetadataKey` – stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #27.

  La chiave di metadati.

- `MetadataValue` – stringa UTF-8, non inferiore a 1 o superiore a 256 byte di lunghezza, corrispondente a Custom string pattern #27.

  Il valore della chiave di metadati.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`

QuerySchemaVersionMetadata azione (Python: query_schema_version_metadata)

Query per le informazioni sui metadati della versione dello schema.

Richiesta

- `SchemaId`: un oggetto SchemaId.

  Una struttura wrapper che può contenere il nome dello schema e l'Amazon Resource Name (ARN).
- `SchemaVersionNumber`: un oggetto SchemaVersionNumber.

  Il numero di versione dello schema.
- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID versione univoco della versione dello schema.
- `MetadataList`: una matrice di oggetti MetadataKeyValuePair.

  Cerca coppie chiave-valore per i metadati, se non vengono forniti verranno recuperate tutte le informazioni sui metadati.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 50.

  Numero massimo di risultati richiesti per pagina. Se il valore non viene fornito, sarà impostato in modo predefinito su 25 per pagina.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se si tratta di una chiamata di continuazione.

Risposta

- `MetadataInfoMap`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #27.

  Ogni valore è un oggetto MetadataInfo.

  Una mappa di una chiave di metadati e dei valori associati.

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID versione univoco della versione dello schema.

- `NextToken`: stringa UTF-8.

  Un token di continuazione per impaginare l'elenco restituito di token, restituiti se il segmento corrente dell'elenco non è l'ultimo.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`

RemoveSchemaVersionMetadata azione (Python: remove_schema_version_metadata)

Rimuove una coppia chiave-valore dai metadati della versione dello schema per l'ID versione dello schema specificato.

Richiesta

- `SchemaId`: un oggetto SchemaId.

  Una struttura wrapper che può contenere il nome dello schema e l'Amazon Resource Name (ARN).

- `SchemaVersionNumber`: un oggetto SchemaVersionNumber.

Il numero di versione dello schema.

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID versione univoco della versione dello schema.

- `MetadataKeyValue`: obbligatorio: un oggetto MetadataKeyValuePair.

  Il valore della chiave di metadati.

Risposta

- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN (Amazon Resource Name) dello schema.

- `SchemaName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome dello schema.

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro.

- `LatestVersion`: booleano.

  La versione più recente dello schema.

- `VersionNumber` – Numero (intero), non inferiore a 1 o superiore a 100000.

  Il numero di versione dello schema.

- `SchemaVersionId` – stringa UTF-8, non inferiore a 36 o superiore a 36 byte di lunghezza, corrispondente a Custom string pattern #12.

  L'ID versione per la versione dello schema.

- `MetadataKey` – stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #27.

  La chiave di metadati.

- `MetadataValue` – stringa UTF-8, non inferiore a 1 o superiore a 256 byte di lunghezza, corrispondente a Custom string pattern #27.

  Il valore della chiave di metadati.

Errori

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`

DeleteRegistry azione (Python: delete_registry)

Elimina l'intero registro, inclusi gli schemi e tutte le relative versioni. Per ottenere lo stato dell'operazione di eliminazione, è possibile chiamare l'API `GetRegistry` dopo la chiamata asincrona. L'eliminazione di un registro comporta la disattivazione di tutte le operazioni online per il registro, ad esempio le API `UpdateRegistry`, `CreateSchema`, `UpdateSchema` e `RegisterSchemaVersion`.

Richiesta

- `RegistryId`: obbligatorio: un oggetto RegistryId.

  Si tratta di una struttura wrapper che può contenere il nome del registro e l'Amazon Resource Name (ARN).

Risposta

- `RegistryName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome del registro in fase di eliminazione.
- `RegistryArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'Amazon Resource Name (ARN) del registro in fase di eliminazione.
- `Status`: stringa UTF-8 (valori validi: `AVAILABLE` | `DELETING`).

Lo stato del registro. Un'operazione riuscita restituirà lo stato `Deleting`.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

DeleteSchema azione (Python: delete_schema)

Elimina l'intero set di schemi, inclusi il set di schemi e tutte le relative versioni. Per ottenere lo stato dell'operazione di eliminazione, è possibile chiamare l'API `GetSchema` dopo la chiamata asincrona. L'eliminazione di un registro comporta la disattivazione di tutte le operazioni online per lo schema, ad esempio le API `GetSchemaByDefinition` e `RegisterSchemaVersion`.

Richiesta

- `SchemaId`: obbligatorio: un oggetto SchemaId.

  Si tratta di una struttura wrapper che può contenere il nome dello schema e l'Amazon Resource Name (ARN).

Risposta

- `SchemaArn` – stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'Amazon Resource Name (ARN) dello schema in fase di eliminazione.
- `SchemaName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #13.

  Il nome dello schema in fase di eliminazione.
- `Status`: stringa UTF-8 (valori validi: `AVAILABLE` | `PENDING` | `DELETING`).

  Lo stato dello schema.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

DeleteSchemaVersions azione (Python: delete_schema_versions)

Rimuove le versioni dallo schema specificato. Può essere fornito un numero di versione o un intervallo. Se la modalità di compatibilità impedisce l'eliminazione di una versione necessaria, ad esempio BACKWARDS_FULL, viene restituito un errore. Chiamare l'API `GetSchemaVersions` dopo questa chiamata elencherà lo stato delle versioni eliminate.

Quando l'intervallo di numeri di versione contiene la versione di checkpoint, l'API restituirà un conflitto 409 e non procederà con l'eliminazione. Prima di utilizzare questa API è necessario rimuovere il checkpoint usando l'API `DeleteSchemaCheckpoint`

Non è possibile utilizzare l'API `DeleteSchemaVersions` per eliminare la prima versione dello schema nel set di schemi. La prima versione dello schema può essere eliminata solo dall'API `DeleteSchema`. Questa operazione eliminerà anche i `SchemaVersionMetadata` collegati nelle versioni dello schema. Le eliminazioni definitive verranno applicate al database.

Se la modalità di compatibilità impedisce l'eliminazione di una versione necessaria, ad esempio BACKWARDS_FULL, viene restituito un errore.

Richiesta

- `SchemaId`: obbligatorio: un oggetto SchemaId.

  Si tratta di una struttura wrapper che può contenere il nome dello schema e l'Amazon Resource Name (ARN).
- `Versions`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 100000 byte di lunghezza, corrispondente a Custom string pattern #28.

  Può essere fornita un intervallo di versioni nel formato:
  - un unico numero di versione, 5
  - un intervallo, 5-8: elimina le versioni 5, 6, 7, 8

Risposta

- `SchemaVersionErrors`: una matrice di oggetti SchemaVersionErrorItem.

  Un elenco di oggetti `SchemaVersionErrorItem`, ciascuno contenente un errore e una versione dello schema.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

Flussi di lavoro

L'API Workflows descrive i tipi di dati e l'API relativi alla creazione, all'aggiornamento o alla visualizzazione dei flussi di lavoro in. AWS Glue

Tipi di dati

- JobNodeDetails struttura
- CrawlerNodeDetails struttura
- TriggerNodeDetails struttura
- Struttura crawl
- Struttura nodo
- Struttura edge
- Struttura flusso di lavoro
- WorkflowGraph struttura
- WorkflowRun struttura
- WorkflowRunStatistics struttura
- StartingEventBatchCondition struttura
- Struttura schema
- BlueprintDetails struttura

- LastActiveDefinition struttura
- BlueprintRun struttura

JobNodeDetails struttura

I dettagli di un nodo processo presenti nel flusso di lavoro.

Campi

- `JobRuns`: una matrice di oggetti JobRun.

  Le informazioni sulle esecuzioni del processo rappresentate dal nodo processo.

CrawlerNodeDetails struttura

I dettagli di un nodo crawler presenti nel flusso di lavoro.

Campi

- `Crawls`: una matrice di oggetti Crawl.

  Un elenco di esecuzioni del crawler rappresentato dal nodo crawler.

TriggerNodeDetails struttura

I dettagli di un nodo Trigger presenti nel flusso di lavoro.

Campi

- `Trigger`: un oggetto Trigger.

  Le informazioni del trigger rappresentate dal nodo trigger.

Struttura crawl

I dettagli di una esecuzione del crawler nel flusso di lavoro.

Campi

- `State`: stringa UTF-8 (valori validi: `RUNNING` | `CANCELLING` | `CANCELLED` | `SUCCEEDED` | `FAILED` | `ERROR`).

  Lo stato del crawler.
- `StartedOn`: timestamp.

  La data e l'ora in cui è stata avviata l'esecuzione del crawler.
- `CompletedOn`: timestamp.

  La data e l'ora in cui si è conclusa l'esecuzione del crawler.
- `ErrorMessage`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Il messaggio di errore associato al crawler.
- `LogGroup`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Log group string pattern.

  Il gruppo di log associato al crawler.
- `LogStream`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza, corrispondente a Log-stream string pattern.

  Il flusso di log associato all'esecuzione del crawler.

Struttura nodo

Un nodo rappresenta un AWS Glue componente (trigger, crawler o job) su un grafico del flusso di lavoro.

Campi

- `Type`: stringa UTF-8 (valori validi: `CRAWLER` | `JOB` | `TRIGGER`).

  Il tipo di AWS Glue componente rappresentato dal nodo.
- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del AWS Glue componente rappresentato dal nodo.

- `UniqueId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID univoco assegnato al nodo all'interno del flusso di lavoro.

- `TriggerDetails`: un oggetto TriggerNodeDetails.

  Le informazioni sul trigger quando il nodo rappresenta un trigger.

- `JobDetails`: un oggetto JobNodeDetails.

  Le informazioni sul processo quando il nodo rappresenta un processo.

- `CrawlerDetails`: un oggetto CrawlerNodeDetails.

  Dettagli del crawler quando il nodo rappresenta un crawler.

Struttura edge

Un bordo rappresenta una connessione diretta tra due AWS Glue componenti che fanno parte del flusso di lavoro a cui appartiene il bordo.

Campi

- `SourceId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'id univoco del nodo all'interno del flusso di lavoro in cui ha origine l'edge.

- `DestinationId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'id univoco del nodo all'interno del flusso di lavoro in cui termina l'edge.

Struttura flusso di lavoro

Un workflow è una raccolta di più AWS Glue job e crawler dipendenti che vengono eseguiti per completare un'attività ETL complessa. Ogni flusso di lavoro gestisce l'esecuzione e il monitoraggio di tutti i suoi processi e crawler.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro.
- `Description`: stringa UTF-8.

  Una descrizione del flusso di lavoro.
- `DefaultRunProperties`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8.

  Un insieme di proprietà da utilizzare come parte di ogni esecuzione del flusso di lavoro. Le proprietà di esecuzione vengono rese disponibili per ogni processo nel flusso di lavoro. Un processo può modificare le proprietà dei processi successivi nel flusso.
- `CreatedOn`: timestamp.

  La data e l'ora in cui il flusso di lavoro è stato creato.
- `LastModifiedOn`: timestamp.

  La data e l'ora più recenti in cui il flusso di lavoro è stato modificato.
- `LastRun`: un oggetto WorkflowRun.

  Le informazioni relative all'ultima esecuzione del flusso di lavoro.
- `Graph`: un oggetto WorkflowGraph.

  Il grafico che rappresenta tutti i AWS Glue componenti che appartengono al flusso di lavoro come nodi e le connessioni dirette tra di essi come bordi.
- `CreationStatus`: stringa UTF-8 (valori validi: `CREATING` | `CREATED` | `CREATION_FAILED`).

  Lo stato della creazione del flusso di lavoro.
- `MaxConcurrentRuns`: numero (intero).

  È possibile utilizzare questo parametro per impedire aggiornamenti multipli indesiderati dei dati, per controllare i costi o, in alcuni casi, per evitare il superamento del numero massimo di esecuzioni

simultanee di uno qualsiasi dei processi componenti. Se si lascia questo parametro vuoto, non è previsto alcun limite al numero di esecuzioni simultanee del flusso di lavoro.

- `BlueprintDetails`: un oggetto BlueprintDetails.

  Questa struttura indica i dettagli del piano da cui viene creato questo particolare flusso di lavoro.

WorkflowGraph struttura

Un diagramma del flusso di lavoro rappresenta il flusso di lavoro completo che contiene tutti i componenti di AWS Glue presenti nel flusso di lavoro e tutte le connessioni orientate esistenti tra essi.

Campi

- `Nodes`: una matrice di oggetti Nodo.

  Un elenco dei AWS Glue componenti appartengono al flusso di lavoro rappresentato come nodi.
- `Edges`: una matrice di oggetti Edge.

  Un elenco di tutte le connessioni orientate tra i nodi appartenenti al flusso di lavoro.

WorkflowRun struttura

Un'esecuzione di un flusso di lavoro è costituita da tutte le informazioni di runtime sull'esecuzione del flusso stesso.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro che è stato eseguito.
- `WorkflowRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di questa esecuzione del flusso di lavoro.
- `PreviousRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID dell'esecuzione del flusso di lavoro precedente.

- `WorkflowRunProperties`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8.

  Le proprietà di esecuzione del flusso di lavoro impostate durante l'esecuzione.

- `StartedOn`: timestamp.

  La data e l'ora in cui è stata avviata l'esecuzione del flusso di lavoro.

- `CompletedOn`: timestamp.

  La data e l'ora in cui si è conclusa l'esecuzione del flusso di lavoro.

- `Status`: stringa UTF-8 (valori validi: `RUNNING` | `COMPLETED` | `STOPPING` | `STOPPED` | `ERROR`).

  Lo stato dell'esecuzione del flusso di lavoro.

- `ErrorMessage`: stringa UTF-8.

  Questo messaggio di errore descrive qualsiasi errore che potrebbe essersi verificato durante l'avvio dell'esecuzione del flusso di lavoro. Attualmente l'unico messaggio di errore è "Esecuzioni simultanee superate per il flusso di lavoro: `foo`".

- `Statistics`: un oggetto WorkflowRunStatistics.

  Le statistiche dell'esecuzione.

- `Graph`: un oggetto WorkflowGraph.

  Il grafico che rappresenta tutti i AWS Glue componenti che appartengono al flusso di lavoro come nodi e le connessioni dirette tra di essi come bordi.

- `StartingEventBatchCondition`: un oggetto StartingEventBatchCondition.

  La condizione batch che ha avviato l'esecuzione del flusso di lavoro.

WorkflowRunStatistics struttura

Le statistiche di esecuzione del flusso di lavoro forniscono le statistiche sull'esecuzione del flusso di lavoro.

Campi

- `TotalActions`: numero (intero).

  Numero totale di operazioni nell'esecuzione del flusso di lavoro.

- `TimeoutActions`: numero (intero).

  Numero totale di operazioni andate in timeout.

- `FailedActions`: numero (intero).

  Numero totale di operazioni che non si sono concluse correttamente.

- `StoppedActions`: numero (intero).

  Numero totale di operazioni che sono state interrotte.

- `SucceededActions`: numero (intero).

  Numero totale di operazioni che si sono concluse correttamente.

- `RunningActions`: numero (intero).

  Numero totale di operazioni in stato di esecuzione.

- `ErroredActions`: numero (intero).

  Indica il numero di esecuzioni del processo nello stato ERROR (ERRORE) nell'esecuzione del flusso di lavoro.

- `WaitingActions`: numero (intero).

  Indica il numero di esecuzioni del processo nello stato WAITING (IN ATTESA) nell'esecuzione del flusso di lavoro.

StartingEventBatchCondition struttura

La condizione batch che ha avviato l'esecuzione del flusso di lavoro. È arrivato il numero di eventi nella dimensione del batch, nel qual caso il BatchSize membro è diverso da zero, oppure la finestra del batch è scaduta, nel qual caso il BatchWindow membro è diverso da zero.

Campi

- `BatchSize`: numero (intero).

  Numero di eventi nel batch.
- `BatchWindow`: numero (intero).

  Durata del periodo di batch in secondi.

Struttura schema

I dettagli di un piano.

Campi

- `Name` – stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Nome del piano.
- `Description`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  La descrizione del piano.
- `CreatedOn`: timestamp.

  La data e l'ora di registrazione del piano.
- `LastModifiedOn`: timestamp.

  La data e l'ora dell'ultima modifica apportata al piano.
- `ParameterSpec`: stringa UTF-8, non inferiore a 1 o superiore a 131072 byte di lunghezza.

  Una stringa JSON che indica l'elenco delle specifiche dei parametri per il piano.
- `BlueprintLocation`: stringa UTF-8.

Specifica il percorso in Amazon S3 in cui è pubblicato il piano.

- `BlueprintServiceLocation`: stringa UTF-8.

  Specifica un percorso in Amazon S3 in cui il piano viene copiato quando si chiama `CreateBlueprint/UpdateBlueprint` per registrare il piano in AWS Glue.

- `Status`: stringa UTF-8 (valori validi: `CREATING` | `ACTIVE` | `UPDATING` | `FAILED`).

  Stato della registrazione del piano.

  - Creating (Creazione): la registrazione del piano è in corso.
  - Active (Attivo): il piano è stato registrato correttamente.
  - Updating (Aggiornamento): è in corso un aggiornamento della registrazione del piano.
  - Failed (Non riuscito): registrazione del piano non riuscita.

- `ErrorMessage`: stringa UTF-8.

  Un messaggio di errore.

- `LastActiveDefinition`: un oggetto LastActiveDefinition.

  Quando sono presenti più versioni di un piano e la versione più recente presenta alcuni errori, questo attributo indica l'ultima definizione del piano riuscita disponibile con il servizio.

BlueprintDetails struttura

I dettagli di un piano.

Campi

- `BlueprintName` – stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Nome del piano.

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  ID esecuzione per questo piano.

LastActiveDefinition struttura

Quando sono presenti più versioni di un piano e la versione più recente presenta alcuni errori, questo attributo indica l'ultima definizione del piano riuscita disponibile con il servizio.

Campi

- `Description`: stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  La descrizione del piano.
- `LastModifiedOn`: timestamp.

  La data e l'ora dell'ultima modifica apportata al piano.
- `ParameterSpec` – stringa UTF-8, non inferiore a 1 o superiore a 131072 byte di lunghezza.

  Una stringa JSON che specifica i parametri per il piano.
- `BlueprintLocation`: stringa UTF-8.

  Speciifica un percorso in Amazon S3 in cui il blueprint viene pubblicato dallo sviluppatore. AWS Glue
- `BlueprintServiceLocation`: stringa UTF-8.

  Specifica un percorso in Amazon S3 in cui viene copiato il progetto quando crei o aggiorni il progetto.

BlueprintRun struttura

I dettagli dell'esecuzione di un piano.

Campi

- `BlueprintName` – stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Nome del piano.
- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID esecuzione per l'esecuzione del piano.

- `WorkflowName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome di un flusso di lavoro creato a seguito di un'esecuzione del piano riuscita. Se l'esecuzione di un piano presenta un errore, non verrà creato alcun flusso di lavoro.

- `State`: stringa UTF-8 (valori validi: `RUNNING` | `SUCCEEDED` | `FAILED` | `ROLLING_BACK`).

  Stato dell'esecuzione del piano. I valori possibili sono:

  - Running (In esecuzione): l'esecuzione del piano è in corso.
  - Succeeded (Riuscito): l'esecuzione del piano è stata completata correttamente.
  - Failed (Non riuscito): l'esecuzione del piano non è riuscita e il ripristino dello stato precedente è completato.
  - Rolling Back (Ripristino dello stato precedente): l'esecuzione del piano non è riuscita ed è in corso il ripristino dello stato precedente.

- `StartedOn`: timestamp.

  La data e l'ora in cui è stata avviata l'esecuzione del piano.

- `CompletedOn`: timestamp.

  La data e l'ora in cui è stata completata l'esecuzione del piano.

- `ErrorMessage`: stringa UTF-8.

  Indica eventuali errori rilevati durante l'esecuzione del piano.

- `RollbackErrorMessage`: stringa UTF-8.

  Se ci sono errori durante la creazione delle entità di un flusso di lavoro, si tenta di ripristinare le entità create fino a quel punto ed eliminarle. Questo attributo indica gli errori rilevati durante il tentativo di eliminare le entità create.

- `Parameters` – stringa UTF-8, non inferiore a 1 o superiore a 131072 byte di lunghezza.

  I parametri del piano come stringa. Dovrai fornire un valore per ogni chiave richiesta dalla specifica del parametro, definita nella `Blueprint$ParameterSpec`.

- `RoleArn`: stringa UTF-8, non inferiore a 1 o superiore a 1024 byte di lunghezza, corrispondente a Custom string pattern #21.

  ARN del ruolo. Questo ruolo verrà assunto dal AWS Glue servizio e verrà utilizzato per creare il flusso di lavoro e altre entità di un flusso di lavoro.

Operazioni

- CreateWorkflow azione (Python: create_workflow)
- UpdateWorkflow azione (Python: update_workflow)
- DeleteWorkflow azione (Python: delete_workflow)
- GetWorkflow azione (Python: get_workflow)
- ListWorkflows azione (Python: list_workflows)
- BatchGetWorkflows azione (Python: batch_get_workflows)
- GetWorkflowRun azione (Python: get_workflow_run)
- GetWorkflowRuns azione (Python: get_workflow_runs)
- GetWorkflowRunProperties azione (Python: get_workflow_run_properties)
- PutWorkflowRunProperties azione (Python: put_workflow_run_properties)
- CreateBlueprint azione (Python: create_blueprint)
- UpdateBlueprint azione (Python: update_blueprint)
- DeleteBlueprint azione (Python: delete_blueprint)
- ListBlueprints azione (Python: list_blueprints)
- BatchGetBlueprints azione (Python: batch_get_blueprints)
- StartBlueprintRun azione (Python: start_blueprint_run)
- GetBlueprintRun azione (Python: get_blueprint_run)
- GetBlueprintRuns azione (Python: get_blueprint_runs)
- StartWorkflowRun azione (Python: start_workflow_run)
- StopWorkflowRun azione (Python: stop_workflow_run)
- ResumeWorkflowRun azione (Python: resume_workflow_run)

CreateWorkflow azione (Python: create_workflow)

Crea un nuovo flusso di lavoro.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome da assegnare al nuovo flusso di lavoro. Deve essere univoco all'interno dell'account.

- `Description`: stringa UTF-8.

  Una descrizione del flusso di lavoro.

- `DefaultRunProperties`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8.

  Un insieme di proprietà da utilizzare come parte di ogni esecuzione del flusso di lavoro.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag da utilizzare con questo flusso di lavoro.

- `MaxConcurrentRuns`: numero (intero).

  È possibile utilizzare questo parametro per impedire aggiornamenti multipli indesiderati dei dati, per controllare i costi o, in alcuni casi, per evitare il superamento del numero massimo di esecuzioni simultanee di uno qualsiasi dei processi componenti. Se si lascia questo parametro vuoto, non è previsto alcun limite al numero di esecuzioni simultanee del flusso di lavoro.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro fornito come parte della richiesta.

Errori

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`

- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

UpdateWorkflow azione (Python: update_workflow)

Aggiorna un flusso di lavoro esistente.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro da aggiornare.
- `Description`: stringa UTF-8.

  La descrizione del flusso di lavoro.
- `DefaultRunProperties`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8.

  Un insieme di proprietà da utilizzare come parte di ogni esecuzione del flusso di lavoro.
- `MaxConcurrentRuns`: numero (intero).

  È possibile utilizzare questo parametro per impedire aggiornamenti multipli indesiderati dei dati, per controllare i costi o, in alcuni casi, per evitare il superamento del numero massimo di esecuzioni simultanee di uno qualsiasi dei processi componenti. Se si lascia questo parametro vuoto, non è previsto alcun limite al numero di esecuzioni simultanee del flusso di lavoro.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro fornito nella richiesta.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

DeleteWorkflow azione (Python: delete_workflow)

Elimina un flusso di lavoro.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro da eliminare.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro fornito nella richiesta.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

GetWorkflow azione (Python: get_workflow)

Recupera i metadati delle risorse per un flusso di lavoro.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro da recuperare.
- `IncludeGraph`: booleano.

  Specifica se includere un diagramma al momento della restituzione dei metadati delle risorse del flusso di lavoro.

Risposta

- `Workflow`: un oggetto Flusso di lavoro.

  I metadati delle risorse per il flusso di lavoro.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

ListWorkflows azione (Python: list_workflows)

Elenca i nomi dei flussi di lavoro creati nell'account.

Richiesta

- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.
- `MaxResults`— Numero (intero), non inferiore a 1 o superiore a 25.

  La dimensione massima di un elenco da restituire.

Risposta

- `Workflows`: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 25 stringhe.

  Elenco dei nomi dei flussi di lavoro nell'account.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non tutti i nomi di flussi di lavoro sono stati restituiti.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchGetWorkflows azione (Python: batch_get_workflows)

Restituisce un elenco di metadati di risorse per un elenco di nomi di flussi di lavoro. Dopo aver chiamato l'operazione `ListWorkflows`, puoi chiamare questa operazione per accedere ai dati a cui sono state concesse le autorizzazioni. Questa operazione supporta tutte le autorizzazioni IAM, tra cui le condizioni di autorizzazione che utilizzano i tag.

Richiesta

- `Names`. Obbligatorio: una serie di stringhe UTF-8, non inferiore a 1 o superiore a 25 stringhe.

  L'elenco dei nomi di flussi di lavoro, che potrebbero essere i nomi restituiti dall'operazione `ListWorkflows`.
- `IncludeGraph`: booleano.

  Specifica se includere un diagramma al momento della restituzione dei metadati delle risorse del flusso di lavoro.

Risposta

- `Workflows`: una matrice di oggetti Flusso di lavoro, non inferiore a 1 o superiore a 25 strutture.

  Un elenco di metadati delle risorse del flusso di lavoro.

- `MissingWorkflows` – una serie di stringhe UTF-8, non inferiore a 1 o superiore a 25 stringhe.

  Un elenco di nomi di flussi di lavoro non trovati.

Errori

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

GetWorkflowRun azione (Python: get_workflow_run)

Consente di recuperare i metadati di una specifica esecuzione di un flusso di lavoro.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro in esecuzione.
- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID dell'esecuzione del flusso di lavoro.
- `IncludeGraph`: booleano.

  Specifica se includere o meno il diagramma del flusso di lavoro nella risposta.

Risposta

- `Run`: un oggetto WorkflowRun.

  I metadati dell'esecuzione del flusso di lavoro richiesti.

Errori

- `InvalidInputException`
- `EntityNotFoundException`

- `InternalServiceException`
- `OperationTimeoutException`

GetWorkflowRuns azione (Python: get_workflow_runs)

Recupera i metadati di tutte le esecuzioni di un dato flusso di lavoro.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro i cui metadati di esecuzione devono essere restituiti.
- `IncludeGraph`: booleano.

  Specifica se includere o meno il diagramma del flusso di lavoro nella risposta.
- `NextToken`: stringa UTF-8.

  La dimensione massima della risposta.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Il numero massimo di esecuzioni del flusso di lavoro da includere nella risposta.

Risposta

- `Runs`: una matrice di oggetti WorkflowRun, non inferiore a 1 o superiore a 1.000 strutture.

  Un elenco oggetti che rappresentano i metadati dell'esecuzione di un flusso di lavoro.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non tutte le esecuzioni del flusso di lavoro richieste sono state restituite.

Errori

- `InvalidInputException`
- `EntityNotFoundException`

- `InternalServiceException`
- `OperationTimeoutException`

GetWorkflowRunProperties azione (Python: get_workflow_run_properties)

Recupera le proprietà dell'esecuzione del flusso di lavoro che sono state impostate durante l'esecuzione.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro che è stato eseguito.
- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del flusso di lavoro le cui proprietà dell'esecuzione devono essere restituite.

Risposta

- `RunProperties`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8.

  Le proprietà dell'esecuzione del flusso di lavoro che sono state impostate durante l'esecuzione specificata.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

PutWorkflowRunProperties azione (Python: put_workflow_run_properties)

Imposta le proprietà dell'esecuzione del flusso di lavoro specificate per la specifica esecuzione del flusso di lavoro. Se una proprietà esiste già per l'esecuzione specificata, il vecchio valore viene sovrascritto, altrimenti aggiunge la proprietà alle proprietà esistenti.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro che è stato eseguito.
- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID dell'esecuzione del flusso di lavoro per il quale è necessario aggiornare le proprietà dell'esecuzione.
- `RunProperties`: obbligatorio: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8.

  Le proprietà da impostare per l'esecuzione specificata.

Risposta

- Nessun parametro di risposta.

Errori

- `AlreadyExistsException`
- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

- `ConcurrentModificationException`

CreateBlueprint azione (Python: create_blueprint)

Registra un blueprint con. AWS Glue

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Nome del piano.
- `Description` – stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  Una descrizione del piano.
- `BlueprintLocation`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 8192 byte di lunghezza, corrispondente a Custom string pattern #23.

  Specifica il percorso in Amazon S3 in cui è pubblicato il piano.
- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Tag da applicare a questo piano.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Restituisce il nome del piano registrato.

Errori

- `AlreadyExistsException`
- `InvalidInputException`

- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

UpdateBlueprint azione (Python: update_blueprint)

Aggiorna un piano registrato.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Nome del piano.
- `Description` – stringa UTF-8, non inferiore a 1 o superiore a 512 byte di lunghezza.

  Una descrizione del piano.
- `BlueprintLocation`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 8192 byte di lunghezza, corrispondente a Custom string pattern #23.

  Specifica il percorso in Amazon S3 in cui è pubblicato il piano.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Restituisce il nome del piano aggiornato.

Errori

- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

- `IllegalBlueprintStateException`

DeleteBlueprint azione (Python: delete_blueprint)

Elimina un piano esistente.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del piano da eliminare.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Restituisce il nome del piano eliminato.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListBlueprints azione (Python: list_blueprints)

Elenca tutti i nomi dei piani in un account.

Richiesta

- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.
- `MaxResults`— Numero (intero), non inferiore a 1 o superiore a 25.

  La dimensione massima di un elenco da restituire.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Filtra l'elenco in base a un tag di AWS risorsa.

Risposta

- `Blueprints`: una matrice di stringhe UTF-8.

  Elenco dei nomi dei piani nell'account.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non sono stati restituiti tutti i nomi di piani.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchGetBlueprints azione (Python: batch_get_blueprints)

Recupera le informazioni su un elenco di piani.

Richiesta

- `Names`. Obbligatorio: una serie di stringhe UTF-8, non inferiore a 1 o superiore a 25 stringhe.

  Un elenco di nomi di piani.
- `IncludeBlueprint`: booleano.

  Specifica se includere o meno il piano nella risposta.
- `IncludeParameterSpec`: booleano.

  Specifica se includere o meno i parametri, come stringa JSON, per il piano nella risposta.

Risposta

- `Blueprints`: una matrice di oggetti Piano.

  Restituisce un elenco di piani come oggetto `Blueprints`.
- `MissingBlueprints`: una matrice di stringhe UTF-8.

  Restituisce un elenco di `BlueprintNames` che non sono stati trovati.

Errori

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

StartBlueprintRun azione (Python: start_blueprint_run)

Avvia una nuova esecuzione del piano specificato.

Richiesta

- `BlueprintName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Nome del piano.
- `Parameters` – stringa UTF-8, non inferiore a 1 o superiore a 131072 byte di lunghezza.

  Specifica i parametri come oggetto `BlueprintParameters`.
- `RoleArn`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 1024 byte di lunghezza, corrispondente a Custom string pattern #21.

  Specifica il ruolo IAM utilizzato per creare il flusso di lavoro.

Risposta

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID esecuzione per l'esecuzione del piano.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`
- `EntityNotFoundException`
- `IllegalBlueprintStateException`

GetBlueprintRun azione (Python: get_blueprint_run)

Recupera i dettagli dell'esecuzione di un piano.

Richiesta

- `BlueprintName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza, corrispondente a Custom string pattern #22.

  Nome del piano.
- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID esecuzione per l'esecuzione del piano che si intende recuperare.

Risposta

- `BlueprintRun`: un oggetto BlueprintRun.

  Restituisce un oggetto `BlueprintRun`.

Errori

- `EntityNotFoundException`
- `InternalServiceException`

- `OperationTimeoutException`

GetBlueprintRuns azione (Python: get_blueprint_runs)

Recupera i dettagli delle esecuzioni del piano per un piano specificato.

Richiesta

- `BlueprintName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del piano.
- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima di un elenco da restituire.

Risposta

- `BlueprintRuns`: una matrice di oggetti BlueprintRun.

  Restituisce un elenco di oggetti `BlueprintRun`.
- `NextToken`: stringa UTF-8.

  Un token di continuazione, se non sono state restituite tutte le esecuzioni del piano.

Errori

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

StartWorkflowRun azione (Python: start_workflow_run)

Avvia una nuova esecuzione del flusso di lavoro specificato.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro da avviare.
- `RunProperties`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa UTF-8.

  Le proprietà dell'esecuzione del flusso di lavoro per la nuova esecuzione del flusso di lavoro.

Risposta

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un ID per la nuova esecuzione.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

StopWorkflowRun azione (Python: stop_workflow_run)

Interrompe l'esecuzione del flusso di lavoro specificato.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro da arrestare.
- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID dell'esecuzione del flusso di lavoro da arrestare.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `IllegalWorkflowStateException`

ResumeWorkflowRun azione (Python: resume_workflow_run)

Riavvia i nodi selezionati di una precedente esecuzione del flusso di lavoro parzialmente completata e riprende l'esecuzione del flusso di lavoro. Vengono eseguiti i nodi selezionati e tutti i nodi che sono a valle dei nodi selezionati.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome del flusso di lavoro da recuperare.
- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID dell'esecuzione del flusso di lavoro da arrestare.

- `NodeIds`. Obbligatorio: una matrice di stringhe UTF-8.

  Un elenco degli ID dei nodi per i nodi da riavviare. I nodi che devono essere riavviati devono avere un tentativo di esecuzione nell'esecuzione originale.

Risposta

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nuovo ID assegnato all'esecuzione del flusso di lavoro ripresa. Ogni ripresa dell'esecuzione del flusso di lavoro avrà un nuovo ID esecuzione.
- `NodeIds`: una matrice di stringhe UTF-8.

  Un elenco degli ID dei nodi che sono stati effettivamente riavviati.

Errori

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentRunsExceededException`
- `IllegalWorkflowStateException`

API machine learning

L'API Machine learning descrive i tipi di dati relativi al machine learning e include le API per la creazione, l'eliminazione o l'aggiornamento di una trasformazione, oppure l'avvio dell'esecuzione di un'attività di machine learning.

Tipi di dati

- Struttura TransformParameters

- Struttura EvaluationMetrics
- Struttura MLTransform
- Struttura FindMatchesParameters
- Struttura FindMatchesMetrics
- Struttura ConfusionMatrix
- Struttura GlueTable
- Struttura TaskRun
- Struttura TransformFilterCriteria
- Struttura TransformSortCriteria
- Struttura TaskRunFilterCriteria
- Struttura TaskRunSortCriteria
- Struttura TaskRunProperties
- Struttura FindMatchesTaskRunProperties
- Struttura ImportLabelsTaskRunProperties
- Struttura ExportLabelsTaskRunProperties
- Struttura LabelingSetGenerationTaskRunProperties
- Struttura SchemaColumn
- Struttura TransformEncryption
- Struttura MLUserDataEncryption
- Struttura ColumnImportance

Struttura TransformParameters

I parametri specifici dell'algoritmo che sono associati alla trasformazione basata su machine learning.

Campi

- `TransformType`. Obbligatorio: stringa UTF-8 (valori validi: `FIND_MATCHES`).

  Il tipo di trasformazione basata su machine learning.

  Per ulteriori informazioni sui tipi di trasformazioni basate su machine learning, consultare Creazione di trasformazioni basate su machine learning.
- `FindMatchesParameters`: un oggetto FindMatchesParameters.

I parametri dell'algoritmo di rilevamento delle corrispondenze.

Struttura EvaluationMetrics

I parametri di valutazione forniscono una stima della qualità della trasformazione basata su machine learning.

Campi

- `TransformType`. Obbligatorio: stringa UTF-8 (valori validi: `FIND_MATCHES`).

  Il tipo di trasformazione basata su machine learning.
- `FindMatchesMetrics`: un oggetto FindMatchesMetrics.

  I parametri di valutazione per l'algoritmo di rilevamento delle corrispondenze.

Struttura MLTransform

Una struttura per la trasformazione basata su machine learning.

Campi

- `TransformId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID univoco della trasformazione generato per la trasformazione basata su machine learning. L'ID è garantito univoco e non si modifica nel tempo.
- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome definito dall'utente per la trasformazione basata su machine learning. I nomi non sono garantite come univoci e possono essere modificati in qualsiasi momento.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una testo descrittivo esteso definito dall'utente per la trasformazione basata su machine learning. Le descrizioni non sono garantite come univoche e possono essere modificate in qualsiasi momento.

- `Status`: stringa UTF-8 (valori validi: `NOT_READY` | `READY` | `DELETING`).

  Lo stato corrente della trasformazione basata su machine learning.

- `CreatedOn`: timestamp.

  Un Timestamp. La data e l'ora di creazione di questa trasformazione basata su machine learning.

- `LastModifiedOn`: timestamp.

  Un Timestamp. L'ultimo istante temporale in cui questa trasformazione basata su machine learning è stata modificata.

- `InputRecordTables`: una matrice di oggetti GlueTable, non superiore a 10 strutture.

  Un elenco di definizioni di tabella di AWS Glue utilizzate dalla trasformazione.

- `Parameters`: un oggetto TransformParameters.

  Oggetto `TransformParameters`. È possibile utilizzare i parametri per ottimizzare (personalizzare) il comportamento della trasformazione basata su machine learning specificando i dati da utilizzare per l'addestramento e le preferenze sui vari compromessi (ad esempio precisione vs. recupero o accuratezza vs. costo).

- `EvaluationMetrics`: un oggetto EvaluationMetrics.

  Oggetto `EvaluationMetrics`. I parametri di valutazione forniscono una stima della qualità della trasformazione basata su machine learning.

- `LabelCount`: numero (intero).

  Un identificatore numerico per il conteggio dei file di etichettatura generati da AWS Glue per questa trasformazione. Man mano che si crea una trasformazione migliore, è possibile scaricare, etichettare e caricare il file di etichettatura in modo iterativo.

- `Schema`: una matrice di oggetti SchemaColumn, non superiore a 100 strutture.

  Una mappa di coppie chiave-valore che rappresenta le colonne e i tipi di dati sui quali può essere eseguita questa trasformazione. È imposto un limite massimo di 100 colonne.

- `Role`: stringa UTF-8.

  Il nome o il nome della risorsa Amazon (ARN) del ruolo IAM con le autorizzazioni richieste. Le autorizzazioni necessarie includono le autorizzazioni del ruolo di servizio AWS Glue per le risorse AWS Glue e le autorizzazioni Amazon S3 richieste dalla trasformazione.

  - Questo ruolo richiede autorizzazioni del ruolo di servizio AWS Glue per consentire l'accesso alle risorse in AWS Glue. Consulta Collegamento di una policy agli utenti IAM che accedono a AWS Glue.
  - Questo ruolo ha bisogno dell'autorizzazione per accedere a origini, destinazioni, cartella temporanea, script e librerie di Amazon Simple Storage Service (Amazon S3) utilizzate dall'esecuzione di questa attività di trasformazione.
- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Questo valore determina la versione di AWS Glue con cui è compatibile questa trasformazione di machine learning. Glue 1.0 è consigliata per la maggior parte dei clienti. Se il valore non è impostato, la compatibilità di Glue è impostata per default su Glue 0.9. Per ulteriori informazioni, consulta Versioni di AWS Glue nella guida per gli sviluppatori.
- `MaxCapacity`: numero (doppio).

  Il numero di unità di elaborazione dati (Data Processing Units, DPU) di AWS Glue allocate per l'esecuzione di questo processo di trasformazione. È possibile allocare da 2 a 100 DPUs; l'impostazione di default è 10. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  `MaxCapacity` è un'opzione mutuamente esclusiva con `NumberOfWorkers` e `WorkerType`.
  - Se `NumberOfWorkers` o `WorkerType` è impostata, `MaxCapacity` può essere impostata.
  - Se `MaxCapacity` è impostata, né `NumberOfWorkers` né `WorkerType` possono essere impostate.
  - Se `WorkerType` è impostata, `NumberOfWorkers` è obbligatoria (e viceversa).
  - `MaxCapacity` e `NumberOfWorkers` devono essere entrambe almeno 1.

  Quando il campo `WorkerType` è impostato su un valore diverso da `Standard`, il campo `MaxCapacity` è impostato automaticamente e diventa di sola lettura.
- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato al momento dell'esecuzione di un'attività di questa trasformazione. Accetta un valore Standard, G.1X o G.2X.

  - Per il tipo di worker `Standard`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 50 GB e 2 esecutori.
  - Per il tipo di worker `G.1X`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 64 GB e 1 esecutore.
  - Per il tipo di worker `G.2X`, ciascun worker fornisce 8 vCPU, 32 GB di memoria, disco da 128 GB e 1 esecutore.

  `MaxCapacity` è un'opzione mutuamente esclusiva con `NumberOfWorkers` e `WorkerType`.

  - Se `NumberOfWorkers` o `WorkerType` è impostata, `MaxCapacity` può essere impostata.
  - Se `MaxCapacity` è impostata, né `NumberOfWorkers` né `WorkerType` possono essere impostate.
  - Se `WorkerType` è impostata, `NumberOfWorkers` è obbligatoria (e viceversa).
  - `MaxCapacity` e `NumberOfWorkers` devono essere entrambe almeno 1.
- `NumberOfWorkers`: numero (intero).

  Il numero di worker di uno specifico `workerType` allocati al momento dell'esecuzione di un'attività della trasformazione.

  Se `WorkerType` è impostata, `NumberOfWorkers` è obbligatoria (e viceversa).
- `Timeout`: numero (intero), almeno 1.

  Il timeout in minuti della trasformazione basata su machine learning.
- `MaxRetries`: numero (intero).

  Il numero massimo di tentativi dopo la conclusione non corretta di un `MLTaskRun` della trasformazione basata su machine learning.
- `TransformEncryption`: un oggetto TransformEncryption.

  Le impostazioni di crittografia dei dati a riposo della trasformazione che si applicano all'accesso ai dati utente. Le trasformazioni di machine learning possono accedere ai dati utente crittografati in Amazon S3 utilizzando il servizio di gestione delle chiavi.

Struttura FindMatchesParameters

I parametri per configurare la trasformazione di rilevamento delle corrispondenze.

Campi

- `PrimaryKeyColumnName`: stringa UTF-8, non inferiore a 1 o superiore a 1024 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di una colonna che identifica in modo univoco le righe nella tabella di origine. Utilizzata per identificare i registri corrispondenti.

- `PrecisionRecallTradeoff`: numero (doppio), non superiore a 1,0.

  Il valore selezionato durante l'ottimizzazione della trasformazione per indicare la distribuzione tra precisione e recupero. Il valore 0,5 significa nessuna preferenza; il valore 1 significa una tendenza esclusiva verso la precisione e un valore di 0 significa una tendenza al recupero. Poiché si tratta di un compromesso, la scelta di valori prossimi a 1 significa recupero molto basso mentre la scelta di valori prossimi a 0 comporta una precisione molto bassa.

  Il parametro precisione indica la frequenza con cui il modello risulta corretto quando prevede una corrispondenza.

  Il parametro recupero indica quanto spesso il modello riesce a prevedere la corrispondenza quando questa è in effetti presente.

- `AccuracyCostTradeoff` – Numero (doppio), non superiore a 1,0.

  Il valore selezionato durante l'ottimizzazione della trasformazione per indicare la distribuzione tra accuratezza e costo. Il valore 0,5 significa che il sistema bilancia accuratezza e costi. Il valore 1 significa una tendenza esclusiva verso l'accuratezza, che spesso implica un costo superiore, talvolta notevolmente superiore. Il valore 0 significa una tendenza esclusiva verso il costo, che può portare a una trasformazione `FindMatches` meno accurata, talvolta con un livello di accuratezza inaccettabilmente basso.

  L'accuratezza misura la capacità della trasformazione di individuare veri positivi e veri negativi. L'incremento dell'accuratezza implica maggiori risorse di elaborazione e costi superiori. Tuttavia permette di raggiungere anche un livello maggiore di recupero.

  Il costo misura la quantità di risorse di elaborazione, e quindi di denaro, che viene utilizzata per eseguire la trasformazione.

- `EnforceProvidedLabels`: booleano.

  Il valore che server per attivare o disattivare la forzatura dell'output affinché corrisponda alle etichette fornite dagli utenti. Se il valore è `True`, la trasformazione `find matches` forza

l'output affinché corrisponda alle etichette fornite. I risultati sostituiscono i normali risultati della combinazione. Se il valore è `False`, la trasformazione `find matches` non garantisce che tutte le etichette fornite siano rispettate e i risultati si basano sul modello addestrato.

Si noti che l'impostazione di questo valore su true può incrementare il tempo di esecuzione della combinazione.

Struttura FindMatchesMetrics

I parametri di valutazione per l'algoritmo di rilevamento delle corrispondenze. La qualità della trasformazione basata su machine learning è misurato chiedendo alla trasformazione di prevedere alcune corrispondenze e confrontando i risultati con alcune corrispondenze note dello stesso set di dati. I parametri di qualità sono basati su un sottoinsieme dei dati, perciò non sono assolutamente precisi.

Campi

- `AreaUnderPRCurve` – Numero (doppio), non superiore a 1,0.

  L'area sotto la curva di precisione/recupero (AUPRC) è un singolo numero che misura la qualità complessiva della trasformazione, indipendente dalla scelta effettuata tra precisione e recupero. Valori più elevati indicano che si dispone di un compromesso tra precisione e recupero più interessante.

  Per ulteriori informazioni, consulta la voce Precisione e recupero su Wikipedia.
- `Precision` – Numero (doppio), non superiore a 1,0.

  Il parametro precisione indica la frequenza con cui la trasformazione risulta corretta quando prevede una corrispondenza. Nello specifico, misura la capacità della trasformazione di individuare i veri positivi rispetto al totale dei veri positivi possibili.

  Per ulteriori informazioni, consulta la voce Precisione e recupero su Wikipedia.
- `Recall` – Numero (doppio), non superiore a 1,0.

  Il parametro recupero indica quanto spesso la trasformazione riesce a prevedere la corrispondenza quando questa è in effetti presente. Nello specifico, misura la capacità della trasformazione di individuare i veri positivi rispetto al totale dei registri che compongono i dati di origine.

  Per ulteriori informazioni, consulta la voce Precisione e recupero su Wikipedia.

- `F1` – Numero (doppio), non superiore a 1,0.

  Il parametro F1 massimo indica l'accuratezza della trasformazione con un valore tra 0 e 1, dove 1 è la migliore precisione.

  Per ulteriori informazioni, consulta la voce F1 score su Wikipedia.
- `ConfusionMatrix`: un oggetto ConfusionMatrix.

  La matrice di confusione mostra gli elementi che la trasformazione sta predicendo in modo accurato e quali tipi di errori sta commettendo.

  Per ulteriori informazioni, consulta la voce MAtrice di confusione su Wikipedia.
- `ColumnImportances` – Una serie di oggetti ColumnImportance, non superiore a 100 strutture.

  Un elenco di strutture `ColumnImportance` contenenti parametri sull'importanza delle colonne, ordinate in ordine di importanza decrescente.

Struttura ConfusionMatrix

La matrice di confusione mostra gli elementi che la trasformazione sta predicendo in modo accurato e quali tipi di errori sta commettendo.

Per ulteriori informazioni, consulta la voce MAtrice di confusione su Wikipedia.

Campi

- `NumTruePositives`: numero (lungo).

  Il numero di corrispondenze nei dati correttamente rilevate dalla trasformazione, nella matrice di confusione della trasformazione.
- `NumFalsePositives`: numero (lungo).

  Il numero di mancate corrispondenze nei dati che la trasformazione ha erroneamente classificato come corrispondenza, nella matrice di confusione della trasformazione.
- `NumTrueNegatives`: numero (lungo).

  Il numero di mancate corrispondenze nei dati che la trasformazione ha correttamente rifiutato, nella matrice di confusione della trasformazione.
- `NumFalseNegatives`: numero (lungo).

Il numero di corrispondenze nei dati che la trasformazione non ha rilevato, nella matrice di confusione della trasformazione.

Struttura GlueTable

Il database e la tabella in AWS Glue Data Catalog usati per i dati in ingresso o in uscita.

Campi

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome del database in AWS Glue Data Catalog.
- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome della tabella in AWS Glue Data Catalog.
- `CatalogId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un identificatore univoco per AWS Glue Data Catalog.
- `ConnectionName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della connessione a AWS Glue Data Catalog.
- `AdditionalOptions`: una matrice di mappe di coppie chiave-valore, non meno di 1 o più di 10 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è una stringa Description, non superiore a 2.048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Opzioni aggiuntive per la tabella. Al momento sono supportate due chiavi:

  - `pushDownPredicate`: filtra le partizioni senza dover elencare e leggere tutti i file nel set di dati.

- `catalogPartitionPredicate`: per utilizzare l'eliminazione delle partizioni lato server utilizzando gli indici delle partizioni in AWS Glue Data Catalog.

Struttura TaskRun

I parametri di campionamento associati alla trasformazione basata su machine learning.

Campi

- `TransformId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione.
- `TaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco di questa esecuzione dell'attività.
- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  Lo stato corrente dell'esecuzione dell'attività invocata.
- `LogGroupName`: stringa UTF-8.

  I nomi dei gruppi di log per la conservazione sicura dei log, associati a questa esecuzione dell'attività.
- `Properties`: un oggetto TaskRunProperties.

  Specifica le proprietà di configurazione associate a questa esecuzione dell'attività.
- `ErrorString`: stringa UTF-8.

  L'elenco delle stringhe di errore associate a questa esecuzione dell'attività.
- `StartedOn`: timestamp.

  La data e l'ora in cui è stata avviata questa esecuzione dell'attività.
- `LastModifiedOn`: timestamp.

  L'ultimo istante temporale in cui è stata modificata l'esecuzione dell'attività invocata.
- `CompletedOn`: timestamp.

L'ultimo istante temporale in cui è stata conclusa l'esecuzione dell'attività invocata.

- `ExecutionTime`: numero (intero).

  Quantità di tempo (in secondi) durante la quale l'esecuzione dell'attività ha utilizzato le risorse.

Struttura TransformFilterCriteria

I criteri utilizzati per filtrare la trasformazione basata su machine learning.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome di trasformazione univoco utilizzato per filtrare la trasformazione basata su machine learning.

- `TransformType`: stringa UTF-8 (valori validi: `FIND_MATCHES`).

  Il tipo di trasformazione basata su machine learning utilizzata per filtrare le trasformazioni basate su machine learning.

- `Status`: stringa UTF-8 (valori validi: `NOT_READY` | `READY` | `DELETING`).

  Filtra l'elenco delle trasformazioni basate su machine learning in base all'ultimo stato della trasformazione (per valutare se una trasformazione può essere utilizzata o meno). Uno dei valori "NOT_READY", "READY" o "DELETING".

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Questo valore determina la versione di AWS Glue con cui è compatibile questa trasformazione di machine learning. Glue 1.0 è consigliata per la maggior parte dei clienti. Se il valore non è impostato, la compatibilità di Glue è impostata per default su Glue 0.9. Per ulteriori informazioni, consulta Versioni di AWS Glue nella guida per gli sviluppatori.

- `CreatedBefore`: timestamp.

  La data e l'ora prima della quale le trasformazioni sono state create.

- `CreatedAfter`: timestamp.

La data e l'ora dopo la quale le trasformazioni sono state create.

- `LastModifiedBefore`: timestamp.

  Filtra le trasformazioni la cui ultima modifica è avvenuta prima di questa data.

- `LastModifiedAfter`: timestamp.

  Filtra le trasformazioni la cui ultima modifica è avvenuta dopo questa data.

- `Schema` – Una serie di oggetti SchemaColumn, non superiore a 100 strutture.

  Filtra i set di dati con uno specifico schema. L'oggetto `Map<Column, Type>` è una matrice di coppie chiave-valore che rappresenta lo schema accettato da questa trasformazione, dove `Column` è il nome di una colonna e `Type` è il tipo di dati, ad esempio un intero o una stringa. È imposto un limite massimo di 100 colonne.

Struttura TransformSortCriteria

I criteri di ordinamento associati alla trasformazione basata su machine learning.

Campi

- `Column`. Obbligatorio: stringa UTF-8 (valori validi: `NAME` | `TRANSFORM_TYPE` | `STATUS` | `CREATED` | `LAST_MODIFIED`).

  La colonna da utilizzare nel criterio di ordinamento associato alla trasformazione basata su machine learning.

- `SortDirection`: obbligatorio: stringa UTF-8 (valori validi: `DESCENDING` | `ASCENDING`).

  Il tipo di ordinamento da utilizzare nel criterio di ordinamento associato alla trasformazione basata su machine learning.

Struttura TaskRunFilterCriteria

I criteri che vengono utilizzati per filtrare le esecuzioni di attività della trasformazione basata su machine learning.

Campi

- `TaskRunType`: stringa UTF-8 (valori validi: `EVALUATION` | `LABELING_SET_GENERATION` | `IMPORT_LABELS` | `EXPORT_LABELS` | `FIND_MATCHES`).

  Il tipo di esecuzione dell'attività.

- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  Lo stato attuale dell'esecuzione dell'attività.

- `StartedBefore`: timestamp.

  Filtra le esecuzioni delle attività avviate prima di questa data.

- `StartedAfter`: timestamp.

  Filtra le esecuzioni delle attività avviate dopo questa data.

Struttura TaskRunSortCriteria

I criteri di ordinamento utilizzati per ordinare l'elenco delle esecuzioni delle attività della trasformazione basata su machine learning.

Campi

- `Column`: obbligatorio: stringa UTF-8 (valori validi: `TASK_RUN_TYPE` | `STATUS` | `STARTED`).

  La colonna da usare per ordinare l'elenco delle esecuzioni delle attività della trasformazione basata su machine learning.

- `SortDirection`: obbligatorio: stringa UTF-8 (valori validi: `DESCENDING` | `ASCENDING`).

  Il tipo di ordinamento da usare per ordinare l'elenco delle esecuzioni delle attività della trasformazione basata su machine learning.

Struttura TaskRunProperties

Le proprietà di configurazione dell'esecuzione dell'attività.

Campi

- `TaskType`: stringa UTF-8 (valori validi: `EVALUATION` | `LABELING_SET_GENERATION` | `IMPORT_LABELS` | `EXPORT_LABELS` | `FIND_MATCHES`).

  Il tipo di esecuzione dell'attività.

- `ImportLabelsTaskRunProperties`: un oggetto ImportLabelsTaskRunProperties.

  Le proprietà di configurazione per l'esecuzione di un'attività di importazione di etichette.

- `ExportLabelsTaskRunProperties`: un oggetto ExportLabelsTaskRunProperties.

  Le proprietà di configurazione per l'esecuzione di un'attività di esportazione di etichette.

- `LabelingSetGenerationTaskRunProperties`: un oggetto LabelingSetGenerationTaskRunProperties.

  Le proprietà di configurazione per l'esecuzione di un'attività di generazione di un set di etichettatura.

- `FindMatchesTaskRunProperties`: un oggetto FindMatchesTaskRunProperties.

  Le proprietà di configurazione per l'esecuzione di un'attività di rilevamento delle corrispondenze.

Struttura FindMatchesTaskRunProperties

Specifica le proprietà di configurazione per l'esecuzione di un'attività di rilevamento delle corrispondenze.

Campi

- `JobId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del processo dell'esecuzione di un'attività di rilevamento delle corrispondenze.

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome assegnato al processo dell'esecuzione di un'attività di rilevamento delle corrispondenze.

- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID di esecuzione del processo dell'esecuzione di un'attività di rilevamento delle corrispondenze.

Struttura ImportLabelsTaskRunProperties

Specifica le proprietà di configurazione per l'esecuzione di un'attività di importazione delle etichette.

Campi

- `InputS3Path`: stringa UTF-8.

  Il percorso Amazon Simple Storage Service (Amazon S3) da dove saranno importate le etichette.
- `Replace`: booleano.

  Indica se sovrascrivere le etichette esistenti.

Struttura ExportLabelsTaskRunProperties

Specifica le proprietà di configurazione per l'esecuzione di un'attività di esportazione delle etichette.

Campi

- `OutputS3Path`: stringa UTF-8.

  Il percorso Amazon Simple Storage Service (Amazon S3) dove saranno esportate le etichette.

Struttura LabelingSetGenerationTaskRunProperties

Specifica le proprietà di configurazione per l'esecuzione di un'attività di generazione di un set di etichettatura.

Campi

- `OutputS3Path`: stringa UTF-8.

  Il percorso Amazon Simple Storage Service (Amazon S3) dove sarà generato il set di etichettatura.

Struttura SchemaColumn

Una coppia chiave-valore che rappresenta una colonna e un tipo di dati sui quali può essere eseguita questa trasformazione. Il parametro `Schema` di `MLTransform` può contenere fino a 100 di queste strutture.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 1024 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della colonna.

- `DataType`: stringa UTF-8, non superiore a 131072 byte di lunghezza, corrispondente a Single-line string pattern.

  Il tipo di dati della colonna.

Struttura TransformEncryption

Le impostazioni di crittografia dei dati a riposo della trasformazione che si applicano all'accesso ai dati utente. Le trasformazioni di machine learning possono accedere ai dati utente crittografati in Amazon S3 utilizzando il servizio di gestione delle chiavi.

Inoltre, le etichette importate e le trasformazioni addestrate possono ora essere crittografate utilizzando una chiave del servizio di gestione delle chiavi fornita dal cliente.

Campi

- `MlUserDataEncryption`: un oggetto MLUserDataEncryption.

  Un oggetto `MLUserDataEncryption` contenente la modalità di crittografia e l'ID chiave KMS fornito dal cliente.

- `TaskRunSecurityConfigurationName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della configurazione di sicurezza.

Struttura MLUserDataEncryption

Le impostazioni di crittografia dei dati a riposo della trasformazione che si applicano all'accesso ai dati utente.

Campi

- `MlUserDataEncryptionMode`: obbligatorio: stringa UTF-8 (valori validi: `DISABLED` | `SSE-KMS="SSEKMS"`).

  La modalità di crittografia applicata ai dati utente. I valori validi sono:

  - DISABLED: la crittografia è disattivata
  - SSEKMS: utilizzo della crittografia lato server con AWS Key Management Service (SSE-KMS) per i dati utente memorizzati in Amazon S3.
- `KmsKeyId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID della chiave KMS fornita dal cliente.

Struttura ColumnImportance

Una struttura contenente il nome della colonna e il punteggio di importanza della colonna per una colonna.

L'importanza delle colonne consente di comprendere il modo in cui queste contribuiscono al modello, identificando quali colonne nei registri sono più importanti di altre.

Campi

- `ColumnName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome di una colonna.
- `Importance` – Numero (doppio), non superiore a 1,0.

  Punteggio di importanza per la colonna, come numero decimale.

Operazioni

Operazione CreateMLTransform (Python: create_ml_transform)

Crea una trasformazione basata su machine learning di AWS Glue. Questa operazione crea la trasformazione e tutti i parametri necessari per l'addestramento.

Richiamare questa operazione come primo passo del processo di utilizzo di una trasformazione basata su machine learning (come ad esempio la trasformazione `FindMatches`) per la deduplicazione dei dati. È possibile fornire una `Description` facoltativa, nonché i parametri che si desiderano utilizzare per l'algoritmo.

Inoltre, è necessario specificare determinati parametri per le attività che AWS Glue esegue per conto dell'utente come parte del processo di addestramento a partire dai dati e di creazione di una trasformazione basata su machine learning di alta qualità. Questi parametri includono `Role` e, facoltativamente, `AllocatedCapacity`, `Timeout` e `MaxRetries`. Per ulteriori informazioni, consulta la pagina sui processi.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome univoco assegnato alla trasformazione al momento della creazione.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione della trasformazione basata su machine learning che viene definita. L'impostazione predefinita è una stringa vuota.

- `InputRecordTables`: obbligatorio: una matrice di oggetti GlueTable, non superiore a 10 strutture.

  Un elenco di definizioni di tabella di AWS Glue utilizzate dalla trasformazione.

- `Parameters`: obbligatorio: un oggetto TransformParameters.

  I parametri algoritmici specifici per il tipo di trasformazione usata. Condizionalmente dipendenti dal tipo di trasformazione.

- `Role`. Obbligatorio: stringa UTF-8.

  Il nome o il nome della risorsa Amazon (ARN) del ruolo IAM con le autorizzazioni richieste. Le autorizzazioni necessarie includono le autorizzazioni del ruolo di servizio AWS Glue per le risorse AWS Glue e le autorizzazioni Amazon S3 richieste dalla trasformazione.

  - Questo ruolo richiede autorizzazioni del ruolo di servizio AWS Glue per consentire l'accesso alle risorse in AWS Glue. Consulta Collegamento di una policy agli utenti IAM che accedono a AWS Glue.
  - Questo ruolo ha bisogno dell'autorizzazione per accedere a origini, destinazioni, cartella temporanea, script e librerie di Amazon Simple Storage Service (Amazon S3) utilizzate dall'esecuzione di questa attività di trasformazione.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Questo valore determina la versione di AWS Glue con cui è compatibile questa trasformazione di machine learning. Glue 1.0 è consigliata per la maggior parte dei clienti. Se il valore non è impostato, la compatibilità di Glue è impostata per default su Glue 0.9. Per ulteriori informazioni, consulta Versioni di AWS Glue nella guida per gli sviluppatori.

- `MaxCapacity`: numero (doppio).

  Il numero di unità di elaborazione dati (Data Processing Units, DPU) di AWS Glue allocate per l'esecuzione di questo processo di trasformazione. È possibile allocare da 2 a 100 DPUs; l'impostazione di default è 10. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  `MaxCapacity` è un'opzione mutuamente esclusiva con `NumberOfWorkers` e `WorkerType`.

  - Se `NumberOfWorkers` o `WorkerType` è impostata, `MaxCapacity` può essere impostata.
  - Se `MaxCapacity` è impostata, né `NumberOfWorkers` né `WorkerType` possono essere impostate.
  - Se `WorkerType` è impostata, `NumberOfWorkers` è obbligatoria (e viceversa).
  - `MaxCapacity` e `NumberOfWorkers` devono essere entrambe almeno 1.

  Quando il campo `WorkerType` è impostato su un valore diverso da `Standard`, il campo `MaxCapacity` è impostato automaticamente e diventa di sola lettura.

  Quando il campo `WorkerType` è impostato su un valore diverso da `Standard`, il campo `MaxCapacity` è impostato automaticamente e diventa di sola lettura.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato quando viene eseguita questa attività. Accetta un valore Standard, G.1X o G.2X.

  - Per il tipo di worker `Standard`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 50 GB e 2 esecutori.
  - Per il tipo di worker `G.1X`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 64 GB e 1 esecutore.
  - Per il tipo di worker `G.2X`, ciascun worker fornisce 8 vCPU, 32 GB di memoria, disco da 128 GB e 1 esecutore.

  `MaxCapacity` è un'opzione mutuamente esclusiva con `NumberOfWorkers` e `WorkerType`.

  - Se `NumberOfWorkers` o `WorkerType` è impostata, `MaxCapacity` può essere impostata.
  - Se `MaxCapacity` è impostata, né `NumberOfWorkers` né `WorkerType` possono essere impostate.

  - Se `WorkerType` è impostata, `NumberOfWorkers` è obbligatoria (e viceversa).
  - `MaxCapacity` e `NumberOfWorkers` devono essere entrambe almeno 1.
- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` specifico allocati quando viene eseguita questa attività.

  Se `WorkerType` è impostata, `NumberOfWorkers` è obbligatoria (e viceversa).
- `Timeout`: numero (intero), almeno 1.

  Il timeout dell'esecuzione dell'attività per questa trasformazione in minuti. Questo è il periodo di tempo massimo durante il quale un'attività in esecuzione per questa trasformazione può consumare risorse prima di essere terminata e impostata allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).
- `MaxRetries`: numero (intero).

  Il numero massimo di tentativi di un'attività della trasformazione dopo un'esecuzione conclusa con esito negativo.
- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag da utilizzare con questa trasformazione basata su machine learning. È possibile utilizzare tag per limitare l'accesso alla trasformazione basata su machine learning. Per ulteriori informazioni sui tag in AWS Glue, consulta Tag AWS in AWS Glue nella guida per gli sviluppatori.
- `TransformEncryption`: un oggetto TransformEncryption.

  Le impostazioni di crittografia dei dati a riposo della trasformazione che si applicano all'accesso ai dati utente. Le trasformazioni di machine learning possono accedere ai dati utente crittografati in Amazon S3 utilizzando il servizio di gestione delle chiavi.

Risposta

- `TransformId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un identificatore univoco generato per la trasformazione.

Errori

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `AccessDeniedException`
- `ResourceNumberLimitExceededException`
- `IdempotentParameterMismatchException`

Operazione UpdateMLTransform (Python: update_ml_transform)

Aggiorna una trasformazione basata su machine learning esistente. Richiamare questa operazione per ottimizzare i parametri dell'algoritmo al fine di ottenere risultati migliori.

Dopo aver invocato questa operazione, è possibile richiamare l'operazione `StartMLEvaluationTaskRun` per valutare in che modo i nuovi parametri hanno raggiunto gli obiettivi (ad esempio migliorare la qualità della trasformazione basata su machine learning o renderla più conveniente).

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un identificatore univoco generato al momento della creazione della trasformazione.
- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome univoco assegnato alla trasformazione al momento della creazione.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione della trasformazione. L'impostazione predefinita è una stringa vuota.
- `Parameters`: un oggetto TransformParameters.

I parametri di configurazione specifici per il tipo di trasformazione (algoritmo) utilizzato. Condizionalmente dipendenti dal tipo di trasformazione.

- `Role`: stringa UTF-8.

  Il nome o il nome della risorsa Amazon (ARN) del ruolo IAM con le autorizzazioni richieste.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Questo valore determina la versione di AWS Glue con cui è compatibile questa trasformazione di machine learning. Glue 1.0 è consigliata per la maggior parte dei clienti. Se il valore non è impostato, la compatibilità di Glue è impostata per default su Glue 0.9. Per ulteriori informazioni, consulta Versioni di AWS Glue nella guida per gli sviluppatori.

- `MaxCapacity`: numero (doppio).

  Il numero di unità di elaborazione dati (Data Processing Units, DPU) di AWS Glue allocate per l'esecuzione di questo processo di trasformazione. È possibile allocare da 2 a 100 DPUs; l'impostazione di default è 10. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  Quando il campo `WorkerType` è impostato su un valore diverso da `Standard`, il campo `MaxCapacity` è impostato automaticamente e diventa di sola lettura.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato quando viene eseguita questa attività. Accetta un valore Standard, G.1X o G.2X.

  - Per il tipo di worker `Standard`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 50 GB e 2 esecutori.
  - Per il tipo di worker `G.1X`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 64 GB e 1 esecutore.
  - Per il tipo di worker `G.2X`, ciascun worker fornisce 8 vCPU, 32 GB di memoria, disco da 128 GB e 1 esecutore.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` specifico allocati quando viene eseguita questa attività.

- `Timeout`: numero (intero), almeno 1.

  Il timeout dell'esecuzione dell'attività per questa trasformazione in minuti. Questo è il periodo di tempo massimo durante il quale un'attività in esecuzione per questa trasformazione può consumare risorse prima di essere terminata e impostata allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).

- `MaxRetries`: numero (intero).

  Il numero massimo di tentativi di un'attività della trasformazione dopo un'esecuzione conclusa con esito negativo.

Risposta

- `TransformId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Identificatore univoco della trasformazione che è stata aggiornata.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `AccessDeniedException`

Operazione DeleteMLTransform (Python: delete_ml_transform)

Elimina una trasformazione basata su machine learning di AWS Glue. Le trasformazioni basate su machine learning sono un tipo speciale di trasformazione che utilizza il machine learning per interpretare i dettagli della trasformazione da eseguire imparando da esempi forniti da operatori umani. Queste trasformazioni sono poi salvate da AWS Glue. Se una trasformazione non è più necessaria, è possibile eliminarla invocando `DeleteMLTransforms`. In questo caso i processi di AWS Glue che ancora fanno riferimento alla trasformazione eliminata non potranno più essere eseguiti correttamente.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione da eliminare.

Risposta

- `TransformId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione che è stata eliminata.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Operazione GetMLTransform (Python: get_ml_transform)

Restituisce un artefatto relativo a una trasformazione basata su machine learning di AWS Glue e a tutti i metadati corrispondenti. Le trasformazioni basate su machine learning sono un tipo speciale di trasformazione che utilizza il machine learning per interpretare i dettagli della trasformazione da eseguire imparando da esempi forniti da operatori umani. Queste trasformazioni sono poi salvate da AWS Glue. È possibile recuperare i metadati invocando `GetMLTransform`.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione, generato al momento della creazione della trasformazione.

Risposta

- `TransformId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione, generato al momento della creazione della trasformazione.
- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome univoco assegnato alla trasformazione al momento della creazione.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione della trasformazione.
- `Status`: stringa UTF-8 (valori validi: `NOT_READY` | `READY` | `DELETING`).

  L'ultima stato noto della trasformazione (per indicare se può essere utilizzata o meno). Uno dei valori "NOT_READY", "READY" o "DELETING".
- `CreatedOn`: timestamp.

  La data e l'ora di creazione della trasformazione.
- `LastModifiedOn`: timestamp.

  La data e l'ora in cui la trasformazione è stata modificata l'ultima volta.
- `InputRecordTables` – Una serie di oggetti GlueTable, non superiore a 10 strutture.

  Un elenco di definizioni di tabella di AWS Glue utilizzate dalla trasformazione.
- `Parameters`: un oggetto TransformParameters.

  I parametri di configurazione specifici per l'algoritmo utilizzato.
- `EvaluationMetrics`: un oggetto EvaluationMetrics.

  I parametri di valutazione più recenti.
- `LabelCount`: numero (intero).

  Il numero di etichette disponibili per questa trasformazione.
- `Schema` – Una serie di oggetti SchemaColumn, non superiore a 100 strutture.

L'oggetto `Map<Column, Type>` che rappresenta lo schema accettato da questa trasformazione. È imposto un limite massimo di 100 colonne.

- `Role`: stringa UTF-8.

  Il nome o il nome della risorsa Amazon (ARN) del ruolo IAM con le autorizzazioni richieste.

- `GlueVersion`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Custom string pattern #15.

  Questo valore determina la versione di AWS Glue con cui è compatibile questa trasformazione di machine learning. Glue 1.0 è consigliata per la maggior parte dei clienti. Se il valore non è impostato, la compatibilità di Glue è impostata per default su Glue 0.9. Per ulteriori informazioni, consulta Versioni di AWS Glue nella guida per gli sviluppatori.

- `MaxCapacity`: numero (doppio).

  Il numero di unità di elaborazione dati (Data Processing Units, DPU) di AWS Glue allocate per l'esecuzione di questo processo di trasformazione. È possibile allocare da 2 a 100 DPUs; l'impostazione di default è 10. Una DPU è una misura relativa della potenza di elaborazione ed è costituita da 4 vCPU di capacità di elaborazione e 16 GB di memoria. Per ulteriori informazioni, consulta la pagina dei prezzi di AWS Glue.

  Quando il campo `WorkerType` è impostato su un valore diverso da `Standard`, il campo `MaxCapacity` è impostato automaticamente e diventa di sola lettura.

- `WorkerType`: stringa UTF-8 (valori validi: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  Il tipo di worker predefinito allocato quando viene eseguita questa attività. Accetta un valore Standard, G.1X o G.2X.

  - Per il tipo di worker `Standard`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 50 GB e 2 esecutori.
  - Per il tipo di worker `G.1X`, ciascun worker fornisce 4 vCPU, 16 GB di memoria, disco da 64 GB e 1 esecutore.
  - Per il tipo di worker `G.2X`, ciascun worker fornisce 8 vCPU, 32 GB di memoria, disco da 128 GB e 1 esecutore.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker di un `workerType` specifico allocati quando viene eseguita questa attività.

- `Timeout`: numero (intero), almeno 1.

  Il timeout dell'esecuzione dell'attività per questa trasformazione in minuti. Questo è il periodo di tempo massimo durante il quale un'attività in esecuzione per questa trasformazione può consumare risorse prima di essere terminata e impostata allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).
- `MaxRetries`: numero (intero).

  Il numero massimo di tentativi di un'attività della trasformazione dopo un'esecuzione conclusa con esito negativo.
- `TransformEncryption`: un oggetto TransformEncryption.

  Le impostazioni di crittografia dei dati a riposo della trasformazione che si applicano all'accesso ai dati utente. Le trasformazioni di machine learning possono accedere ai dati utente crittografati in Amazon S3 utilizzando il servizio di gestione delle chiavi.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Operazione GetMLTransforms (Python: get_ml_transforms)

Consente di ottenere un elenco filtrabile e ordinabile delle trasformazioni basate su machine learning di AWS Glue esistenti. Le trasformazioni basate su machine learning sono un tipo speciale di trasformazione che utilizza il machine learning per interpretare i dettagli della trasformazione da eseguire imparando da esempi forniti da operatori umani. Queste trasformazioni sono poi salvate da AWS Glue ed è possibile recuperare i metadati chiamando `GetMLTransforms`.

Richiesta

- `NextToken`: stringa UTF-8.

  Un token di paginazione per partizionare i risultati.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

Numero massimo di risultati da restituire.

- `Filter`: un oggetto TransformFilterCriteria.

  Il criterio di filtraggio della trasformazione.

- `Sort`: un oggetto TransformSortCriteria.

  Il criterio di ordinamento.

Risposta

- `Transforms`: obbligatorio: una matrice di oggetti MLTransform.

  Un elenco di trasformazioni basate su machine learning.

- `NextToken`: stringa UTF-8.

  Un token di impaginazione, se sono disponibili altri risultati.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Operazione ListMLTransforms (Python: list_ml_transforms)

Recupera un elenco ordinabile e filtrabile delle trasformazioni di machine learning AWS Glue esistenti in questo account AWS o le risorse con il tag specificato. L'operazione accetta il campo facoltativo `Tags` che si può utilizzare come filtro per la risposta in modo che le risorse con tag possano essere recuperate come gruppo. Se si sceglie di utilizzare il filtro dei tag, potranno essere recuperate solo le risorse con tag.

Richiesta

- `NextToken`: stringa UTF-8.

  Token di continuazione, se si tratta di una richiesta di continuazione.

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  La dimensione massima di un elenco da restituire.

- `Filter`: un oggetto TransformFilterCriteria.

  Un elemento `TransformFilterCriteria` utilizzato per filtrare la trasformazione basata su machine learning.

- `Sort`: un oggetto TransformSortCriteria.

  Un elemento `TransformSortCriteria` usato per ordinare le trasformazioni basate su machine learning.

- `Tags` – Una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Specifica che vengono restituite solo le risorse con tag.

Risposta

- `TransformIds`. Obbligatorio: una matrice di stringhe UTF-8.

  Gli identificatori di tutte le trasformazioni basate su machine learning nell'account o le trasformazioni basate su machine learning con i tag specificati.

- `NextToken`: stringa UTF-8.

  Token di continuazione, se l'elenco restituito non contiene l'ultimo parametro disponibile.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Operazione StartMLEvaluationTaskRun (Python: start_ml_evaluation_task_run)

Avvia un'attività per stimare la qualità della trasformazione.

Quando si forniscono set di etichette come esempi di verità, il machine learning di AWS Glue utilizza alcuni di tali esempi come fonte di apprendimento. Le altre etichette sono impiegate come test per stimare la qualità.

Restituisce un identificatore univoco dell'esecuzione. È possibile invocare `GetMLTaskRun` per ottenere ulteriori informazioni sulle statistiche di `EvaluationTaskRun`.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione basata su machine learning.

Risposta

- `TaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco associato a questa esecuzione.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ConcurrentRunsExceededException`
- `MLTransformNotReadyException`

Operazione StartMLLabelingSetGenerationTaskRun (Python: start_ml_labeling_set_generation_task_run)

Avvia il flusso di apprendimento attivo della trasformazione basata su machine learning per migliorare la qualità della trasformazione generando set di etichette e aggiungendo etichette.

Al termine di `StartMLLabelingSetGenerationTaskRun`, AWS Glue avrà generato un "set di etichettatura" o un set di domande a cui l'operatore umano è chiamato a rispondere.

Nel caso della trasformazione `FindMatches`, queste domande seguono la seguente struttura: "Qual è il modo corretto per raggruppare queste righe in gruppi costituiti interamente di registri corrispondenti?"

Dopo il completamento del processo di etichettatura, è possibile caricare le etichette con una chiamata a `StartImportLabelsTaskRun`. Al termine di `StartImportLabelsTaskRun`, tutte le esecuzioni successive della trasformazione basata su machine learning utilizzeranno le etichette nuove e migliorate ed eseguiranno una trasformazione di maggiore qualità.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione basata su machine learning.
- `OutputS3Path`. Obbligatorio: stringa UTF-8.

  Il percorso Amazon Simple Storage Service (Amazon S3) dove si genera il set di etichettatura.

Risposta

- `TaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

Errori

- `EntityNotFoundException`
- `InvalidInputException`

- `OperationTimeoutException`
- `InternalServiceException`
- `ConcurrentRunsExceededException`

Operazione GetMLTaskRun (Python: get_ml_task_run)

Recupera i dettagli di una specifica esecuzione di attività su una trasformazione basata su machine learning. Le esecuzioni di attività di machine learning sono operazioni asincrone che AWS Glue esegue per conto del cliente come parte di molteplici flussi di lavoro basati su machine learning. È possibile verificare le statistiche di ogni esecuzione di attività invocando `GetMLTaskRun` con il `TaskRunID` e il `TransformID` della sua trasformazione padre.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione basata su machine learning.
- `TaskRunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione dell'attività.

Risposta

- `TransformId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione dell'attività.
- `TaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.
- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  Lo stato di questa esecuzione dell'attività.

- `LogGroupName`: stringa UTF-8.

  I nomi dei gruppi di log associati all'esecuzione dell'attività.
- `Properties`: un oggetto TaskRunProperties.

  L'elenco delle proprietà associate all'esecuzione dell'attività.
- `ErrorString`: stringa UTF-8.

  Le stringhe di errore associate all'esecuzione dell'attività.
- `StartedOn`: timestamp.

  La data e l'ora in cui è stata avviata questa esecuzione dell'attività.
- `LastModifiedOn`: timestamp.

  La data e l'ora in questa esecuzione dell'attività è stata modificata l'ultima volta.
- `CompletedOn`: timestamp.

  La data e l'ora in cui eseguire questa esecuzione dell'attività è stata completata.
- `ExecutionTime`: numero (intero).

  Quantità di tempo (in secondi) durante la quale l'esecuzione dell'attività ha utilizzato le risorse.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Operazione GetMLTaskRuns (Python: get_ml_task_runs)

Restituisce un elenco di esecuzioni di una trasformazione basata su machine learning. Le esecuzioni di attività di machine learning sono operazioni asincrone che AWS Glue esegue per conto del cliente come parte di molteplici flussi di lavoro basati su machine learning. È possibile ottenere un elenco filtrabile e ordinabile delle esecuzioni delle attività di machine learning invocando `GetMLTaskRuns` con il `TransformID` della trasformazione padre e altri parametri facoltativi come documentato in questa sezione.

Questa operazione restituisce un elenco di storico di esecuzioni e deve essere paginato.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione basata su machine learning.
- `NextToken`: stringa UTF-8.

  Un token per l'impaginazione dei risultati. L'impostazione predefinita è vuota.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Numero massimo di risultati da restituire.
- `Filter`: un oggetto TaskRunFilterCriteria.

  I criteri di filtro, nella struttura `TaskRunFilterCriteria`, per l'esecuzione dell'attività.
- `Sort`: un oggetto TaskRunSortCriteria.

  I criteri di ordinamento, nella struttura `TaskRunSortCriteria`, per l'esecuzione dell'attività.

Risposta

- `TaskRuns`: una matrice di oggetti TaskRun.

  Un elenco delle esecuzioni di attività associate alla trasformazione.
- `NextToken`: stringa UTF-8.

  Un token di impaginazione, se sono disponibili altri risultati.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Operazione CancelMLTaskRun (Python: cancel_ml_task_run)

Annulla (interrompe) un'esecuzione dell'attività. Le esecuzioni di attività di machine learning sono operazioni asincrone che AWS Glue esegue per conto del cliente come parte di molteplici flussi di lavoro basati su machine learning. È possibile annullare un'attività di machine learning in qualsiasi momento invocando `CancelMLTaskRun` con l'`TransformID` della trasformazione padre dell'esecuzione dell'attività e il `TaskRunId` dell'esecuzione dell'attività.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione basata su machine learning.
- `TaskRunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un identificatore univoco dell'esecuzione dell'attività.

Risposta

- `TransformId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione basata su machine learning.
- `TaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione dell'attività.
- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  Lo stato di questa esecuzione.

Errori

- `EntityNotFoundException`
- `InvalidInputException`

- `OperationTimeoutException`
- `InternalServiceException`

Operazione StartExportLabelsTaskRun (Python: start_export_labels_task_run)

Avvia un'operazione asincrona di esportazione di tutti i dati etichettati per una determinata trasformazione. Questa attività è l'unica chiamata API relativa alle etichette che non fa parte del tipico flusso di lavoro di addestramento attivo. Generalmente si utilizza `StartExportLabelsTaskRun` quando si desidera operare contemporaneamente su tutte le etichette, ad esempio quando si desidera rimuovere o modificare delle etichette che sono state indicate in precedenza come verità. Questa operazione API accetta il `TransformId` le cui etichette si desiderano esportare e un percorso su Amazon Simple Storage Service (Amazon S3) su cui esportare le etichette. L'operazione restituisce un `TaskRunId`. È possibile controllare lo stato dell'esecuzione dell'attività invocando l'API `GetMLTaskRun`.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco della trasformazione basata su machine learning.
- `OutputS3Path`. Obbligatorio: stringa UTF-8.

  Il percorso di Amazon S3 su cui esportare le etichette.

Risposta

- `TaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione dell'attività.

Errori

- `EntityNotFoundException`
- `InvalidInputException`

- `OperationTimeoutException`
- `InternalServiceException`

Operazione StartImportLabelsTaskRun (Python: start_import_labels_task_run)

Consente di fornire ulteriori etichette (esempi di verità) da utilizzare per addestrare la trasformazione basata su machine learning e migliorarne la qualità. Questa operazione API viene in genere utilizzata come parte del flusso di lavoro di addestramento attivo che inizia con l'invocazione di `StartMLLabelingSetGenerationTaskRun` e che ha come risultato finale il miglioramento della qualità della trasformazione basata su machine learning.

Al completamento di `StartMLLabelingSetGenerationTaskRun`, il machine learning di AWS Glue avrà generato una serie di domande a cui l'operatore umano è chiamato a rispondere. (L'attività di risposta a tali domande è spesso denominata "etichettatura" all'interno dei flussi di lavoro di machine learning). Nel caso della trasformazione `FindMatches`, queste domande seguono la seguente struttura: "Qual è il modo corretto per raggruppare queste righe in gruppi costituiti interamente di registri corrispondenti?" Dopo il completamento del processo di etichettatura, gli utenti caricano le proprie risposte/etichette con una chiamata a `StartImportLabelsTaskRun`. Al termine di `StartImportLabelsTaskRun`, tutte le esecuzioni successive della trasformazione basata su machine learning utilizzeranno le etichette nuove e migliorate ed eseguiranno una trasformazione di maggiore qualità.

Per impostazione predefinita, `StartMLLabelingSetGenerationTaskRun` apprende continuamente dalle etichette caricate e le combina a meno che il parametro `Replace` non sia impostato su "true". Se `Replace` è impostato su "true", `StartImportLabelsTaskRun` elimina e dimentica tutte le etichette caricate in precedenza e apprende solo dal set esatto appena caricato. La sostituzione delle etichette può essere utile se ci si rende conto di aver precedentemente caricato delle etichette errate e si ritiene che ciò possa avere ripercussioni negative sulla qualità della trasformazione.

È possibile controllare lo stato dell'esecuzione dell'attività invocando l'operazione `GetMLTaskRun`.

Richiesta

- `TransformId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'identificatore univoco della trasformazione basata su machine learning.

- `InputS3Path`. Obbligatorio: stringa UTF-8.

  Il percorso Amazon Simple Storage Service (Amazon S3) da cui si importano le etichette.

- `ReplaceAllLabels`: booleano.

  Indica se sovrascrivere le etichette esistenti.

Risposta

- `TaskRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione dell'attività.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`

API di qualità dei dati

L'API di qualità dei dati descrive i tipi di dati relativi alla qualità dei dati e include l'API per la creazione, l'eliminazione o l'aggiornamento dei set di regole, le esecuzioni e le valutazioni della qualità.

Tipi di dati

- DataSource struttura
- DataQualityRulesetListDetails struttura
- DataQualityTargetTable struttura
- DataQualityRulesetEvaluationRunDescription struttura

- DataQualityRulesetEvaluationRunFilter struttura
- DataQualityEvaluationRunAdditionalRunOptions struttura
- DataQualityRuleRecommendationRunDescription struttura
- DataQualityRuleRecommendationRunFilter struttura
- DataQualityResult struttura
- DataQualityAnalyzerResult struttura
- DataQualityObservation struttura
- MetricBasedObservation struttura
- DataQualityMetricValues struttura
- DataQualityRuleResult struttura
- DataQualityResultDescription struttura
- DataQualityResultFilterCriteria struttura
- DataQualityRulesetFilterCriteria struttura

DataSource struttura

Una fonte di dati (una AWS Glue tabella) per la quale desideri ottenere risultati sulla qualità dei dati.

Campi

- `GlueTable`: obbligatorio: un oggetto GlueTable.

  Una AWS Glue tabella.

DataQualityRulesetListDetails struttura

Descrive un set di regole di qualità dei dati restituito da `GetDataQualityRuleset`.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del set di regole di qualità dei dati.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del set di regole di qualità dei dati.

- `CreatedOn`: timestamp.

  La data e l'ora di creazione del set di regole della qualità dei dati.

- `LastModifiedOn`: timestamp.

  La data e l'ora di modifica del set di regole della qualità dei dati.

- `TargetTable`: un oggetto DataQualityTargetTable.

  Un oggetto che rappresenta una AWS Glue tabella.

- `RecommendationRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Quando un set di regole è stato creato da un'esecuzione di raccomandazione, questo ID di esecuzione viene generato per collegare i due.

- `RuleCount`: numero (intero).

  Il numero di regole nel set di regole.

DataQualityTargetTable struttura

Un oggetto che rappresenta una AWS Glue tabella.

Campi

- `TableName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della AWS Glue tabella.

- `DatabaseName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del database in cui esiste la AWS Glue tabella.

- `CatalogId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'ID del catalogo dove esiste la AWS Glue tabella.

DataQualityRulesetEvaluationRunDescription struttura

Descrive il risultato di un'esecuzione di valutazione del set di regole della qualità dei dati.

Campi

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.
- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  Lo stato di questa esecuzione.
- `StartedOn`: timestamp.

  La data e l'ora di inizio dell'esecuzione.
- `DataSource`: un oggetto DataSource.

  L'origine dati (una AWS Glue tabella) associata all'esecuzione.

DataQualityRulesetEvaluationRunFilter struttura

I criteri di filtro.

Campi

- `DataSource`: obbligatorio: un oggetto DataSource.

  Filtro basato su una fonte di dati (una AWS Glue tabella) associata all'esecuzione.
- `StartedBefore`: timestamp.

  Filtra i risultati in base alle esecuzioni iniziate prima di questo momento.
- `StartedAfter`: timestamp.

  Filtra i risultati in base alle esecuzioni iniziate dopo questo momento.

DataQualityEvaluationRunAdditionalRunOptions struttura

Opzioni di esecuzione aggiuntive che è possibile specificare per l'esecuzione di una valutazione.

Campi

- `CloudWatchMetricsEnabled`: booleano.

  Se abilitare o meno le CloudWatch metriche.

- `ResultsS3Prefix`: stringa UTF-8.

  Prefisso per Amazon S3 per archiviare i risultati.

DataQualityRuleRecommendationRunDescription struttura

Descrive il risultato dell'esecuzione di una raccomandazione per una regola di qualità dei dati.

Campi

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  Lo stato di questa esecuzione.

- `StartedOn`: timestamp.

  La data e l'ora in cui è stata avviata questa esecuzione.

- `DataSource`: un oggetto DataSource.

  L'origine dati (AWS Glue tabella) associata all'esecuzione della raccomandazione.

DataQualityRuleRecommendationRunFilter struttura

Un filtro per elencare le esecuzioni delle raccomandazioni per la qualità dei dati.

Campi

- `DataSource`: obbligatorio: un oggetto DataSource.

  Filtro basato su una fonte di dati specificata (AWS Glue tabella).

- `StartedBefore`: timestamp.

  Filtra in base all'ora per i risultati avviati prima dell'ora indicata.

- `StartedAfter`: timestamp.

  Filtra in base all'ora per i risultati avviati dopo l'ora indicata.

DataQualityResult struttura

Descrive un risultato di qualità dei dati.

Campi

- `ResultId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un ID di risultato univoco per il risultato della qualità dei dati.

- `Score`: numero (doppio), non superiore a 1,0.

  Un punteggio aggregato della qualità dei dati. Rappresenta il rapporto tra le regole inviate e il numero totale di regole.

- `DataSource`: un oggetto DataSource.

  La tabella associata al risultato della qualità dei dati, se presente.

- `RulesetName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del set di regole associato al risultato della qualità dei dati.

- `EvaluationContext`: stringa UTF-8.

  Nel contesto di un lavoro in AWS Glue Studio, a ogni nodo del canvas viene in genere assegnato un nome e i nodi di qualità dei dati avranno dei nomi. Nel caso di più nodi, `evaluationContext` può differenziare i nodi.

- `StartedOn`: timestamp.

  La data e ora di inizio di questa esecuzione della qualità dei dati.

- `CompletedOn`: timestamp.

  La data e ora di completamento dell'esecuzione della qualità dei dati.

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del processo associato al risultato della qualità dei dati, se presente.

- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di esecuzione del processo associato al risultato della qualità dei dati, se presente.

- `RulesetEvaluationRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di esecuzione univoco per la valutazione del set di regole per questo risultato di qualità dei dati.

- `RuleResults`: una matrice di oggetti DataQualityRuleResult, non superiore a 2000 strutture.

  Un elenco di oggetti `DataQualityRuleResult` che rappresentano i risultati per ogni regola.

- `AnalyzerResults`: una matrice di oggetti DataQualityAnalyzerResult, non superiore a 2000 strutture.

  Un elenco di oggetti `DataQualityAnalyzerResult` che rappresentano i risultati per ogni analizzatore.

- `Observations`: una matrice di oggetti DataQualityObservation, non superiore a 50 strutture.

  Un elenco di oggetti `DataQualityObservation` che rappresentano le osservazioni generate dopo la valutazione di regole e analizzatori.

DataQualityAnalyzerResult struttura

Descrive il risultato della valutazione di un analizzatore della qualità dei dati.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome dell'analizzatore della qualità dei dati.

- `Description`: stringa UTF-8, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione dell'analizzatore della qualità dei dati.

- `EvaluationMessage`: stringa UTF-8, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Un messaggio di valutazione.

- `EvaluatedMetrics`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è un numero (doppio).

  Una mappa di metriche associate alla valutazione dell'analizzatore.

DataQualityObservation struttura

Descrive l'osservazione generata dopo la valutazione delle regole e degli analizzatori.

Campi

- `Description`: stringa UTF-8, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione dell'osservazione sulla qualità dei dati.

- `MetricBasedObservation`: un oggetto MetricBasedObservation.

  Un oggetto di tipo `MetricBasedObservation` che rappresenta l'osservazione basata su metriche di qualità dei dati valutate.

MetricBasedObservation struttura

Descrive l'osservazione basata su metriche generata sulla base di metriche valutate sulla qualità dei dati.

Campi

- `MetricName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della metrica di qualità dei dati utilizzata per generare l'osservazione.
- `MetricValues`: un oggetto DataQualityMetricValues.

  Un oggetto di tipo `DataQualityMetricValues` che rappresenta l'analisi del valore della metrica di qualità dei dati.
- `NewRules`: una matrice di stringhe UTF-8.

  Un elenco di nuove regole sulla qualità dei dati generate come parte dell'osservazione basata sul valore della metrica di qualità dei dati.

DataQualityMetricValues struttura

Descrive il valore della metrica di qualità dei dati in base all'analisi dei dati storici.

Campi

- `ActualValue`: numero (doppio).

  Il valore effettivo della metrica di qualità dei dati.
- `ExpectedValue`: numero (doppio).

  Il valore atteso della metrica di qualità dei dati in base all'analisi dei dati storici.
- `LowerLimit`: numero (doppio).

  Il limite inferiore del valore della metrica di qualità dei dati in base all'analisi dei dati storici.
- `UpperLimit`: numero (doppio).

  Il limite superiore del valore della metrica di qualità dei dati in base all'analisi dei dati storici.

DataQualityRuleResult struttura

Descrive il risultato della valutazione del set di regole della qualità dei dati.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della regola di qualità dei dati.
- `Description`: stringa UTF-8, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione della regola di qualità dei dati.
- `EvaluationMessage`: stringa UTF-8, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Un messaggio di valutazione.
- `Result`: stringa UTF-8 (valori validi: `PASS` | `FAIL` | `ERROR`).

  Lo stato positivo o negativo per la regola.
- `EvaluatedMetrics`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è un numero (doppio).

  Una mappa dei parametri associati alla valutazione della regola.

DataQualityResultDescription struttura

Descrive un risultato di qualità dei dati.

Campi

- `ResultId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID del risultato univoco per questo risultato della qualità dei dati.

- `DataSource`: un oggetto DataSource.

  Il nome della tabella associata al risultato della qualità dei dati.
- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del processo associato al risultato della qualità dei dati.
- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di esecuzione del processo associato al risultato della qualità dei dati.
- `StartedOn`: timestamp.

  L'ora di inizio dell'esecuzione per questo risultato di qualità dei dati.

DataQualityResultFilterCriteria struttura

Criteri utilizzati per restituire i risultati della qualità dei dati.

Campi

- `DataSource`: un oggetto DataSource.

  Filtra i risultati in base all'origine dati specificata. Ad esempio, recuperare tutti i risultati per una AWS Glue tabella.
- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Filtra i risultati in base al nome del processo specificato.
- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Filtra i risultati in base all'ID di esecuzione del processo specificato.
- `StartedAfter`: timestamp.

  Filtra i risultati in base alle esecuzioni iniziate dopo questo momento.
- `StartedBefore`: timestamp.

  Filtra i risultati in base alle esecuzioni iniziate prima di questo momento.

DataQualityRulesetFilterCriteria struttura

I criteri utilizzati per filtrare i set di regole della qualità dei dati.

Campi

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del criterio di filtro del set di regole.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  La descrizione dei criteri di filtro del set di regole.
- `CreatedBefore`: timestamp.

  Filtra i set di regole creati prima di questa data.
- `CreatedAfter`: timestamp.

  Filtra i set di regole creati dopo questa data.
- `LastModifiedBefore`: timestamp.

  Filtra i set di regole modificati per l'ultima volta prima di questa data.
- `LastModifiedAfter`: timestamp.

  Filtra i set di regole modificati per l'ultima volta dopo questa data.
- `TargetTable`: un oggetto DataQualityTargetTable.

  Il nome e il nome del database della tabella di destinazione.

Operazioni

- StartDataQualityRulesetEvaluationRun azione (Python: start_data_quality_ruleset_evaluation_run)
- CancelDataQualityRulesetEvaluationRun azione (Python: cancel_data_quality_ruleset_evaluation_run)
- GetDataQualityRulesetEvaluationRun azione (Python: get_data_quality_ruleset_evaluation_run)
- ListDataQualityRulesetEvaluationRuns azione (Python: list_data_quality_ruleset_evaluation_runs)

- StartDataQualityRuleRecommendationRun azione (Python: start_data_quality_rule_recommendation_run)
- CancelDataQualityRuleRecommendationRun azione (Python: cancel_data_quality_rule_recommendation_run)
- GetDataQualityRuleRecommendationRun azione (Python: get_data_quality_rule_recommendation_run)
- ListDataQualityRuleRecommendationRuns azione (Python: list_data_quality_rule_recommendation_runs)
- GetDataQualityResult azione (Python: get_data_quality_result)
- BatchGetDataQualityResult azione (Python: batch_get_data_quality_result)
- ListDataQualityResults azione (Python: list_data_quality_results)
- CreateDataQualityRuleset azione (Python: create_data_quality_ruleset)
- DeleteDataQualityRuleset azione (Python: delete_data_quality_ruleset)
- GetDataQualityRuleset azione (Python: get_data_quality_ruleset)
- ListDataQualityRulesets azione (Python: list_data_quality_rulesets)
- UpdateDataQualityRuleset azione (Python: update_data_quality_ruleset)

StartDataQualityRulesetEvaluationRun azione (Python: start_data_quality_ruleset_evaluation_run)

Una volta ottenuta una definizione del set di regole (consigliata o personalizzata), si chiama questa operazione per valutare il set di regole rispetto a una fonte di dati (tabella). AWS Glue La valutazione calcola i risultati che è possibile recuperare con l'API `GetDataQualityResult`.

Richiesta

- `DataSource`: obbligatorio: un oggetto DataSource.

  L'origine dati (AWS Glue tabella) associata a questa esecuzione.
- `Role`: obbligatorio: stringa UTF-8.

  Un IAM ruolo fornito per crittografare i risultati dell'esecuzione.
- `NumberOfWorkers`: numero (intero).

  Il numero di worker `G.1X` da utilizzare nell'esecuzione. Il predefinito è 5.

- `Timeout`: numero (intero), almeno 1.

  Il timeout per una esecuzione (in minuti). Questo è il tempo massimo durante il quale un'esecuzione può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).

- `ClientToken`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Utilizzato per l'idempotenza e consigliato per l'impostazione su un ID casuale (come un UUID) per evitare di creare o avviare più istanze della stessa risorsa.

- `AdditionalRunOptions`: un oggetto DataQualityEvaluationRunAdditionalRunOptions.

  Opzioni di esecuzione aggiuntive che è possibile specificare per l'esecuzione di una valutazione.

- `RulesetNames` obbligatorio: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 10 stringhe.

  Un elenco di nomi di set di regole.

- `AdditionalDataSources`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è un oggetto DataSource.

  Una mappa di stringhe di riferimento a origini dati aggiuntive che è possibile specificare per l'esecuzione di una valutazione.

Risposta

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

Errori

- `InvalidInputException`
- `EntityNotFoundException`

- `OperationTimeoutException`
- `InternalServiceException`
- `ConflictException`

CancelDataQualityRulesetEvaluationRun azione (Python: cancel_data_quality_ruleset_evaluation_run)

Annulla un'esecuzione in cui un set di regole viene valutato rispetto a un'origine dati.

Richiesta

- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRulesetEvaluationRun azione (Python: get_data_quality_ruleset_evaluation_run)

Richiama un'esecuzione in cui un set di regole viene valutato rispetto a un'origine dati.

Richiesta

- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

Risposta

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

- `DataSource`: un oggetto DataSource.

  L'origine dati (una tabella) associata a questa esecuzione di valutazione. AWS Glue

- `Role`: stringa UTF-8.

  Un IAM ruolo fornito per crittografare i risultati dell'esecuzione.

- `NumberOfWorkers`: numero (intero).

  Il numero di worker `G.1X` da utilizzare nell'esecuzione. Il predefinito è 5.

- `Timeout`: numero (intero), almeno 1.

  Il timeout per una esecuzione (in minuti). Questo è il tempo massimo durante il quale un'esecuzione può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).

- `AdditionalRunOptions`: un oggetto DataQualityEvaluationRunAdditionalRunOptions.

  Opzioni di esecuzione aggiuntive che è possibile specificare per l'esecuzione di una valutazione.

- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  Lo stato di questa esecuzione.

- `ErrorString`: stringa UTF-8.

  Le stringhe di errore associate all'esecuzione.

- `StartedOn`: timestamp.

  La data e l'ora in cui è stata avviata questa esecuzione.

- `LastModifiedOn`: timestamp.

Un Timestamp. L'ultimo momento in cui questa raccomandazione della regola di qualità dei dati è stata modificata.

- `CompletedOn`: timestamp.

  La data e l'ora in cui è stata completata questa esecuzione.

- `ExecutionTime`: numero (intero).

  La quantità di tempo (in secondi) durante la quale l'esecuzione ha utilizzato le risorse.

- `RulesetNames`: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 10 stringhe.

  Un elenco di nomi dei set di regole per l'esecuzione.

- `ResultIds`: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 10 stringhe.

  Un elenco di ID dei risultati per i risultati della qualità dei dati per l'esecuzione.

- `AdditionalDataSources`: una matrice della mappa di coppie chiave-valore.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Ogni valore è un oggetto DataSource.

  Una mappa di stringhe di riferimento a origini dati aggiuntive che è possibile specificare per l'esecuzione di una valutazione.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRulesetEvaluationRuns azione (Python: list_data_quality_ruleset_evaluation_runs)

Elenca tutte le esecuzioni che soddisfano i criteri di filtro, in cui un set di regole viene valutato rispetto a un'origine dati.

Richiesta

- `Filter`: un oggetto DataQualityRulesetEvaluationRunFilter.

  I criteri di filtro.
- `NextToken`: stringa UTF-8.

  Un token di paginazione per partizionare i risultati.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Numero massimo di risultati da restituire.

Risposta

- `Runs`: una matrice di oggetti DataQualityRulesetEvaluationRunDescription.

  Un elenco di oggetti `DataQualityRulesetEvaluationRunDescription` che rappresentano le esecuzioni del set di regole della qualità dei dati.
- `NextToken`: stringa UTF-8.

  Un token di impaginazione, se sono disponibili altri risultati.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

StartDataQualityRuleRecommendationRun azione (Python: start_data_quality_rule_recommendation_run)

Avvia un'esecuzione di raccomandazioni che viene utilizzata per generare regole quando non sai quali regole scrivere. AWS Glue Data Quality analizza i dati e fornisce consigli per un potenziale set di regole. Puoi quindi classificare il set di regole e modificare il set di regole generato a tuo piacimento.

Le esecuzioni di consigli vengono eliminate automaticamente dopo 90 giorni.

Richiesta

- `DataSource`: obbligatorio: un oggetto DataSource.

  L'origine dati (AWS Glue tabella) associata a questa esecuzione.
- `Role`: obbligatorio: stringa UTF-8.

  Un IAM ruolo fornito per crittografare i risultati dell'esecuzione.
- `NumberOfWorkers`: numero (intero).

  Il numero di worker `G.1X` da utilizzare nell'esecuzione. Il predefinito è 5.
- `Timeout`: numero (intero), almeno 1.

  Il timeout per una esecuzione (in minuti). Questo è il tempo massimo durante il quale un'esecuzione può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).
- `CreatedRulesetName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome per il set di regole.
- `ClientToken`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Utilizzato per l'idempotenza e consigliato per l'impostazione su un ID casuale (come un UUID) per evitare di creare o avviare più istanze della stessa risorsa.

Risposta

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

- `ConflictException`

CancelDataQualityRuleRecommendationRun azione (Python: cancel_data_quality_rule_recommendation_run)

Annulla l'esecuzione della raccomandazione specificata utilizzata per generare le regole.

Richiesta

- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRuleRecommendationRun azione (Python: get_data_quality_rule_recommendation_run)

Ottiene l'esecuzione della raccomandazione specificata utilizzata per generare le regole.

Richiesta

- `RunId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.

Risposta

- `RunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'identificatore univoco dell'esecuzione associato a questa esecuzione di attività.
- `DataSource`: un oggetto DataSource.

  L'origine dati (una tabella) associata a questa esecuzione. AWS Glue
- `Role`: stringa UTF-8.

  Un IAM ruolo fornito per crittografare i risultati dell'esecuzione.
- `NumberOfWorkers`: numero (intero).

  Il numero di worker `G.1X` da utilizzare nell'esecuzione. Il predefinito è 5.
- `Timeout`: numero (intero), almeno 1.

  Il timeout per una esecuzione (in minuti). Questo è il tempo massimo durante il quale un'esecuzione può utilizzare le risorse prima di essere terminata e passare allo stato `TIMEOUT`. Il valore di default è 2.880 minuti (48 ore).
- `Status`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  Lo stato di questa esecuzione.
- `ErrorString`: stringa UTF-8.

  Le stringhe di errore associate all'esecuzione.
- `StartedOn`: timestamp.

  La data e l'ora in cui è stata avviata questa esecuzione.
- `LastModifiedOn`: timestamp.

  Un Timestamp. L'ultimo momento in cui questa raccomandazione della regola di qualità dei dati è stata modificata.
- `CompletedOn`: timestamp.

  La data e l'ora in cui è stata completata questa esecuzione.
- `ExecutionTime`: numero (intero).

La quantità di tempo (in secondi) durante la quale l'esecuzione ha utilizzato le risorse.

- `RecommendedRuleset`: stringa UTF-8, non inferiore a 1 o superiore a 65.536 byte di lunghezza.

  Una volta completata l'esecuzione di una raccomandazione della regola di avvio, viene creato un set di regole consigliato (una serie di regole). Questo membro ha queste regole nel formato DQDL (Data Quality Definition Language).

- `CreatedRulesetName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del set di regole che è stato creato dall'esecuzione.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRuleRecommendationRuns azione (Python: list_data_quality_rule_recommendation_runs)

Elenca le esecuzioni delle raccomandazioni che soddisfano i criteri di filtro.

Richiesta

- `Filter`: un oggetto DataQualityRuleRecommendationRunFilter.

  I criteri di filtro.

- `NextToken`: stringa UTF-8.

  Un token di paginazione per partizionare i risultati.

- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Numero massimo di risultati da restituire.

Risposta

- `Runs`: una matrice di oggetti DataQualityRuleRecommendationRunDescription.

  Elenco di oggetti `DataQualityRuleRecommendationRunDescription`.

- `NextToken`: stringa UTF-8.

  Un token di impaginazione, se sono disponibili altri risultati.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityResult azione (Python: get_data_quality_result)

Recupera il risultato di una valutazione della regola della qualità dei dati.

Richiesta

- `ResultId`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un ID di risultato univoco per il risultato della qualità dei dati.

Risposta

- `ResultId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un ID di risultato univoco per il risultato della qualità dei dati.

- `Score`: numero (doppio), non superiore a 1,0.

  Un punteggio aggregato della qualità dei dati. Rappresenta il rapporto tra le regole inviate e il numero totale di regole.

- `DataSource`: un oggetto DataSource.

La tabella associata al risultato della qualità dei dati, se presente.

- `RulesetName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del set di regole associato al risultato della qualità dei dati.

- `EvaluationContext`: stringa UTF-8.

  Nel contesto di un lavoro in AWS Glue Studio, a ogni nodo del canvas viene in genere assegnato un nome e i nodi di qualità dei dati avranno dei nomi. Nel caso di più nodi, `evaluationContext` può differenziare i nodi.

- `StartedOn`: timestamp.

  La data e ora di inizio dell'esecuzione di questo risultato della qualità dei dati.

- `CompletedOn`: timestamp.

  La data e ora di completamento dell'esecuzione di questo risultato della qualità dei dati.

- `JobName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del processo associato al risultato della qualità dei dati, se presente.

- `JobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di esecuzione del processo associato al risultato della qualità dei dati, se presente.

- `RulesetEvaluationRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID di esecuzione univoco associato alla valutazione del set di regole.

- `RuleResults`: una matrice di oggetti DataQualityRuleResult, non superiore a 2000 strutture.

  Un elenco di oggetti `DataQualityRuleResult` che rappresentano i risultati per ogni regola.

- `AnalyzerResults`: una matrice di oggetti DataQualityAnalyzerResult, non superiore a 2000 strutture.

  Un elenco di oggetti `DataQualityAnalyzerResult` che rappresentano i risultati per ogni analizzatore.

- `Observations`: una matrice di oggetti DataQualityObservation, non superiore a 50 strutture.

Un elenco di oggetti `DataQualityObservation` che rappresentano le osservazioni generate dopo la valutazione di regole e analizzatori.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `EntityNotFoundException`

BatchGetDataQualityResult azione (Python: batch_get_data_quality_result)

Recupera un elenco di risultati della qualità dei dati per gli ID dei risultati specificati.

Richiesta

- `ResultIds` obbligatorio: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 100 stringhe.

  Un elenco di ID dei risultati univoci per i risultati della qualità dei dati.

Risposta

- `Results`: obbligatorio: una matrice di oggetti DataQualityResult.

  Un elenco di oggetti `DataQualityResult` che rappresentano i risultati della qualità dei dati.
- `ResultsNotFound`: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 100 stringhe.

  Un elenco di ID dei risultati per i quali non sono stati trovati risultati.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityResults azione (Python: list_data_quality_results)

Restituisce tutti i risultati di esecuzione della qualità dei dati per il tuo account.

Richiesta

- `Filter`: un oggetto DataQualityResultFilterCriteria.

  I criteri di filtro.
- `NextToken`: stringa UTF-8.

  Un token di paginazione per partizionare i risultati.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Numero massimo di risultati da restituire.

Risposta

- `Results`: obbligatorio: una matrice di oggetti DataQualityResultDescription.

  Elenco di oggetti `DataQualityResultDescription`.
- `NextToken`: stringa UTF-8.

  Un token di impaginazione, se sono disponibili altri risultati.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

CreateDataQualityRuleset azione (Python: create_data_quality_ruleset)

Crea un set di regole di qualità dei dati con regole DQDL applicate a una tabella specificata. AWS Glue

Il set di regole viene creato utilizzando il Data Quality Definition Language (DQDL). Per ulteriori informazioni, consulta la guida per gli sviluppatori. AWS Glue

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome univoco per il set di regole di qualità dei dati.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del set di regole di qualità dei dati.

- `Ruleset`: obbligatorio: stringa UTF-8, lunghezza non inferiore a 1 o non superiore a 65.536 byte.

  Un set di regole Data Quality Definition Language (DQDL). Per ulteriori informazioni, consulta la guida per AWS Glue gli sviluppatori.

- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Un elenco di tag applicati al set di regole di qualità dei dati.

- `TargetTable`: un oggetto DataQualityTargetTable.

  Una tabella di destinazione associata al set di regole di qualità dei dati.

- `RecommendationRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un ID di esecuzione univoco per l'esecuzione della raccomandazione.

- `ClientToken`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Utilizzato per l'idempotenza e consigliato per l'impostazione su un ID casuale (come un UUID) per evitare di creare o avviare più istanze della stessa risorsa.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Un nome univoco per il set di regole di qualità dei dati.

Errori

- `InvalidInputException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

DeleteDataQualityRuleset azione (Python: delete_data_quality_ruleset)

Elimina un set di regole di qualità dei dati.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome per il set di regole di qualità dei dati.

Risposta

- Nessun parametro di risposta.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRuleset azione (Python: get_data_quality_ruleset)

Restituisce un set di regole esistente per identificatore o nome.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del set di regole.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del set di regole.

- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del set di regole.

- `Ruleset`: stringa UTF-8, non inferiore a 1 o superiore a 65.536 byte di lunghezza.

  Un set di regole Data Quality Definition Language (DQDL). Per ulteriori informazioni, consulta la guida per gli sviluppatori. AWS Glue

- `TargetTable`: un oggetto DataQualityTargetTable.

  Il nome e il nome del database della tabella di destinazione.

- `CreatedOn`: timestamp.

  Un Timestamp. La data e l'ora di creazione del set di regole di qualità dei dati.

- `LastModifiedOn`: timestamp.

  Un Timestamp. L'ultimo momento in cui questo set di regole di qualità dei dati è stato modificato.

- `RecommendationRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Quando un set di regole è stato creato da un'esecuzione di raccomandazione, questo ID di esecuzione viene generato per collegare i due.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRulesets azione (Python: list_data_quality_rulesets)

Restituisce un elenco impaginato di set di regole per l'elenco di tabelle specificato. AWS Glue

Richiesta

- `NextToken`: stringa UTF-8.

  Un token di paginazione per partizionare i risultati.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Numero massimo di risultati da restituire.
- `Filter`: un oggetto DataQualityRulesetFilterCriteria.

  I criteri di filtro.
- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Un elenco di tag di coppie chiave-valore.

Risposta

- `Rulesets`: una matrice di oggetti DataQualityRulesetListDetails.

  Un elenco impaginato di set di regole per l'elenco di tabelle specificato. AWS Glue
- `NextToken`: stringa UTF-8.

  Un token di impaginazione, se sono disponibili altri risultati.

Errori

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

UpdateDataQualityRuleset azione (Python: update_data_quality_ruleset)

Aggiorna il set di regole di qualità dei dati specificato.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del set di regole di qualità dei dati.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del set di regole.
- `Ruleset`: stringa UTF-8, non inferiore a 1 o superiore a 65.536 byte di lunghezza.

  Un set di regole Data Quality Definition Language (DQDL). Per ulteriori informazioni, consulta la guida per gli sviluppatori. AWS Glue

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del set di regole di qualità dei dati.
- `Description`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Una descrizione del set di regole.
- `Ruleset`: stringa UTF-8, non inferiore a 1 o superiore a 65.536 byte di lunghezza.

Un set di regole Data Quality Definition Language (DQDL). Per ulteriori informazioni, consulta la guida per AWS Glue gli sviluppatori.

Errori

- `EntityNotFoundException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

API di rilevamento dati sensibili

L'API di rilevamento dati sensibili descrive le API utilizzate per rilevare i dati sensibili nelle colonne e nelle righe dei dati strutturati.

Tipi di dati

- Struttura CustomEntityType

Struttura CustomEntityType

Un oggetto che rappresenta un modello personalizzato per il rilevamento di dati sensibili tra colonne e righe dei dati strutturati.

Campi

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome per il pattern personalizzato che consente di recuperarlo o cancellarlo in un secondo momento. Il nome deve essere univoco per account AWS.

- `RegexString`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Stringa di espressione regolare utilizzata per rilevare dati sensibili in un modello personalizzato.
- `ContextWords`: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 20 stringhe.

  Un elenco di parole contestuali. Se nessuna di queste parole contestuali viene trovata nelle vicinanze dell'espressione regolare, i dati non verranno rilevati come dati sensibili.

  Se non vengono passate parole contestuali, viene controllata solo un'espressione regolare.

Operazioni

- Operazione CreateCustomEntityType (Python: create_custom_entity_type)
- Operazione DeleteCustomEntityType (Python: delete_custom_entity_type)
- Operazione GetCustomEntityType (Python: get_custom_entity_type)
- Operazione BatchGetCustomEntityTypes (Python: batch_get_custom_entity_types)
- Operazione ListCustomEntityTypes Action (Python: list_custom_entity_types)

Operazione CreateCustomEntityType (Python: create_custom_entity_type)

Crea un modello personalizzato utilizzato per rilevare dati sensibili tra le colonne e le righe dei dati strutturati.

Ogni modello personalizzato creato specifica un'espressione regolare e un elenco facoltativo di parole contestuali. Se non vengono passate parole contestuali, viene controllata solo un'espressione regolare.

Richiesta

- `Name`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Un nome per il pattern personalizzato che consente di recuperarlo o cancellarlo in un secondo momento. Il nome deve essere univoco per account AWS.
- `RegexString`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Stringa di espressione regolare utilizzata per rilevare dati sensibili in un modello personalizzato.

- `ContextWords`: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 20 stringhe.

  Un elenco di parole contestuali. Se nessuna di queste parole contestuali viene trovata nelle vicinanze dell'espressione regolare, i dati non verranno rilevati come dati sensibili.

  Se non vengono passate parole contestuali, viene controllata solo un'espressione regolare.

- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Un elenco di tag applicati al tipo di entità personalizzato.

Risposta

- `Name`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del modello personalizzato che hai creato.

Errori

- `AccessDeniedException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

Operazione DeleteCustomEntityType (Python: delete_custom_entity_type)

Elimina un modello personalizzato specificandone il nome.

Richiesta

- Name: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del modello personalizzato da eliminare.

Risposta

- Name: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del modello personalizzato che hai eliminato.

Errori

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Operazione GetCustomEntityType (Python: get_custom_entity_type)

Recupera i dettagli di un modello personalizzato specificandone il nome.

Richiesta

- Name: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome del modello personalizzato da recuperare.

Risposta

- Name: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

Il nome del modello personalizzato recuperato.

- `RegexString`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Stringa di espressione regolare utilizzata per rilevare dati sensibili in un modello personalizzato.

- `ContextWords`: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 20 stringhe.

  Un elenco di parole contestuali, se specificato quando è stato creato il modello personalizzato. Se nessuna di queste parole contestuali viene trovata nelle vicinanze dell'espressione regolare, i dati non verranno rilevati come dati sensibili.

Errori

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Operazione BatchGetCustomEntityTypes (Python: batch_get_custom_entity_types)

Recupera i dettagli per i modelli personalizzati specificati da un elenco di nomi.

Richiesta

- `Names`: obbligatorio: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 50 stringhe.

  Un elenco di nomi dei modelli personalizzati da recuperare.

Risposta

- `CustomEntityTypes`: una matrice di oggetti CustomEntityType.

  Un elenco di oggetti `CustomEntityType` che rappresentano i modelli personalizzati creati.

- `CustomEntityTypesNotFound`: una matrice di stringhe UTF-8, non inferiore a 1 o superiore a 50 stringhe.

  Un elenco dei nomi dei modelli personalizzati che non sono stati trovati.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Operazione ListCustomEntityTypes Action (Python: list_custom_entity_types)

Elenca tutti i modelli personalizzati che sono stati creati.

Richiesta

- `NextToken`: stringa UTF-8.

  Un token di paginazione per partizionare i risultati.
- `MaxResults`: numero (intero), non inferiore a 1 o superiore a 1000.

  Numero massimo di risultati da restituire.
- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Un elenco di tag di coppie chiave-valore.

Risposta

- `CustomEntityTypes`: una matrice di oggetti CustomEntityType.

  Un elenco di oggetti `CustomEntityType` che rappresentano modelli personalizzati.
- `NextToken`: stringa UTF-8.

Un token di impaginazione, se sono disponibili altri risultati.

Errori

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

API per l'assegnazione di tag in AWS Glue

Tipi di dati

- Struttura tag

Struttura tag

L'oggetto `Tag` rappresenta un'etichetta che puoi assegnare a una risorsa AWS. Ogni tag è composto da una chiave e da un valore opzionale, entrambi personalizzabili.

Per ulteriori informazioni sui tag e sul controllo dell'accesso alle risorse in AWS Glue, consulta Tag AWS in AWS Glue e Come specificare gli ARN delle risorse AWS Glue nella guida per gli sviluppatori.

Campi

- `key`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  La chiave di tag: La chiave è obbligatoria quando si crea un tag per un oggetto. La chiave rispetta la distinzione tra maiuscole e minuscole e non deve contenere il prefisso aws.
- `value`: stringa UTF-8, non superiore a 256 byte di lunghezza.

  Il valore del tag. Il valore è facoltativo quando si crea un tag per un oggetto. Il valore rispetta la distinzione tra maiuscole e minuscole e non deve contenere il prefisso aws.

Operazioni

Operazione TagResource (Python: tag_resource)

Aggiunge tag a una risorsa. Un tag è un'etichetta che è possibile assegnare a una risorsa AWS. In AWS Glue, è possibile applicare i tag solo ad alcune risorse. Per informazioni sulle risorse cui è possibile applicare un tag, consulta Tag AWS in AWS Glue.

Oltre alle autorizzazioni di applicazione di tag per chiamare le API contrassegnate con tali tag, sono necessarie anche l'autorizzazione `glue:GetConnection`, per chiamare le API di applicazione di tag sulle connessioni, e l'autorizzazione `glue:GetDatabase`, per chiamare le API di applicazione di tag sui database.

Richiesta

- `ResourceArn`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'ARN della risorsa di AWS Glue a cui aggiungere i tag. Per ulteriori informazioni sugli ARN delle risorse di AWS Glue, consulta Modello di stringa ARN AWS Glue.
- `TagsToAdd`: obbligatorio: una matrice di mappe di coppie chiave-valore, non superiore a 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  Tag da aggiungere a questa risorsa.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Operazione UntagResource (Python: untag_resource)

Rimuove i tag da una risorsa.

Richiesta

- `ResourceArn`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'Amazon Resource Name (ARN) della risorsa da cui rimuovere i tag.
- `TagsToRemove`: obbligatorio: una matrice di stringhe UTF-8, non superiore a 50 stringhe.

  Tag da rimuovere da questa risorsa.

Risposta

- Nessun parametro di risposta.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Operazione GetTags (Python: get_tags)

Recupera un elenco di tag associati a una risorsa.

Richiesta

- `ResourceArn`. Obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 10240 byte di lunghezza, corrispondente a Custom string pattern #17.

  L'Amazon Resource Name (ARN) della risorsa per cui recuperare i tag.

Risposta

- `Tags`: una matrice di mappe con coppie chiave-valore, non superiore alle 50 coppie.

  Ogni chiave è una stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

  Ogni valore è una stringa UTF-8, lunga non più di 256 byte.

  I tag richiesti.

Errori

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Tipi di dati comuni

I tipi di dati comuni descrivono i vari tipi di dati comuni in AWS Glue.

Struttura tag

L'`Tag`oggetto rappresenta un'etichetta che è possibile assegnare a una AWS risorsa. Ogni tag è composto da una chiave e da un valore opzionale, entrambi personalizzabili.

Per ulteriori informazioni sui tag e sul controllo dell'accesso alle risorse in AWS Glue, consulta AWS Tags in AWS Glue e Specificing AWS Glue Resource ARNs nella guida per sviluppatori.

Campi

- `key`: stringa UTF-8, non inferiore a 1 o superiore a 128 byte di lunghezza.

La chiave di tag: La chiave è obbligatoria quando si crea un tag per un oggetto. La chiave rispetta la distinzione tra maiuscole e minuscole e non deve contenere il prefisso aws.

- `value`: stringa UTF-8, non superiore a 256 byte di lunghezza.

  Il valore del tag. Il valore è facoltativo quando si crea un tag per un oggetto. Il valore rispetta la distinzione tra maiuscole e minuscole e non deve contenere il prefisso aws.

DecimalNumber struttura

Contiene un valore numerico nel formato decimale.

Campi

- `UnscaledValue`: obbligatorio: blob.

  Il valore numerico non scalato.
- `Scale`: obbligatorio: numero (intero).

  La scala che determina la posizione del punto decimale nel valore non scalato.

ErrorDetail struttura

Contiene dettagli su un errore.

Campi

- `ErrorCode`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il codice associato a questo errore.
- `ErrorMessage`: stringa di descrizione, non superiore a 2048 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  Messaggio che descrive l'errore.

PropertyPredicate struttura

Definisce il predicato di una proprietà.

Campi

- `Key` – Stringa Value, non superiore a 1024 byte di lunghezza.

  La chiave della proprietà.
- `Value` – Stringa Value, non superiore a 1024 byte di lunghezza.

  Valore della proprietà.
- `Comparator`: stringa UTF-8 (valori validi: `EQUALS` | `GREATER_THAN` | `LESS_THAN` | `GREATER_THAN_EQUALS` | `LESS_THAN_EQUALS`).

  Il comparatore utilizzato per confrontare questa proprietà con altre.

ResourceUri struttura

Gli URI delle risorse di funzione.

Campi

- `ResourceType`: stringa UTF-8 (valori validi: `JAR` | `FILE` | `ARCHIVE`).

  Il tipo di risorsa.
- `Uri`: uniform resource identifier (uri), non inferiore a 1 e non superiore a 1024 byte di lunghezza, corrispondente a URI address multi-line string pattern.

  L'URI per l'accesso alla risorsa.

ColumnStatistics struttura

Rappresenta le statistiche a livello di colonna generate per una tabella o una partizione.

Campi

- `ColumnName`: obbligatorio: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Nome della colonna a cui appartengono le statistiche.
- `ColumnType`: obbligatorio: il nome del tipo, non superiore a 20000 byte di lunghezza, corrispondente a Single-line string pattern.

Il tipo di dati della colonna.

- `AnalyzedTime`: obbligatorio: timestamp.

  Il timestamp dell'ora di generazione delle statistiche di colonna.

- `StatisticsData`: obbligatorio: un oggetto ColumnStatisticsData.

  Un oggetto `ColumnStatisticData` che contiene i valori dei dati delle statistiche.

ColumnStatisticsError struttura

Incapsula un oggetto `ColumnStatistics` non riuscito e il motivo dell'errore.

Campi

- `ColumnStatistics`: un oggetto ColumnStatistics.

  `ColumnStatistics` della colonna.

- `Error`: un oggetto ErrorDetail.

  Un messaggio di errore con il motivo dell'errore di un'operazione.

ColumnError struttura

Incapsula il nome di una colonna non riuscita e il motivo dell'errore.

Campi

- `ColumnName`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  Il nome della colonna non riuscita.

- `Error`: un oggetto ErrorDetail.

  Un messaggio di errore con il motivo dell'errore di un'operazione.

ColumnStatisticsData struttura

Contiene i singoli tipi di dati delle statistiche delle colonne. Solo un oggetto dati deve essere impostato e indicato dall'attributo `Type`.

Campi

- `Type`. Obbligatorio: stringa UTF-8 (valori validi: `BOOLEAN` | `DATE` | `DECIMAL` | `DOUBLE` | `LONG` | `STRING` | `BINARY`).

  Il tipo di dati delle statistiche delle colonne.
- `BooleanColumnStatisticsData`: un oggetto BooleanColumnStatisticsData.

  Dati statistici di colonna booleani.
- `DateColumnStatisticsData`: un oggetto DateColumnStatisticsData.

  Dati statistici di colonna date.
- `DecimalColumnStatisticsData`: un oggetto DecimalColumnStatisticsData.

  Dati statistici delle colonne decimali. UnscaledValues all'interno ci sono oggetti binari codificati in Base64 che memorizzano rappresentazioni big-endian, due come complemento, del valore non scalato del decimale.
- `DoubleColumnStatisticsData`: un oggetto DoubleColumnStatisticsData.

  Dati statistici di colonna doppi.
- `LongColumnStatisticsData`: un oggetto LongColumnStatisticsData.

  Dati statistici di colonna long.
- `StringColumnStatisticsData`: un oggetto StringColumnStatisticsData.

  Dati statistici di colonna stringa.
- `BinaryColumnStatisticsData`: un oggetto BinaryColumnStatisticsData.

  Dati statistici di colonna binari.

BooleanColumnStatisticsData struttura

Definisce le statistiche di colonna supportate per le colonne di dati booleani.

Campi

- `NumberOfTrues`: obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori true nella colonna.
- `NumberOfFalses` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori false nella colonna.
- `NumberOfNulls` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori null nella colonna.

DateColumnStatisticsData struttura

Definisce le statistiche di colonna supportate per le colonne di dati timestamp.

Campi

- `MinimumValue`: timestamp.

  Il valore più basso nella colonna.
- `MaximumValue`: timestamp.

  Il valore più alto nella colonna.
- `NumberOfNulls` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori null nella colonna.
- `NumberOfDistinctValues` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori distinti in una colonna.

DecimalColumnStatisticsData struttura

Definisce le statistiche di colonna supportate per le colonne di dati con numeri a virgola fissa.

Campi

- `MinimumValue`: un oggetto DecimalNumber.

  Il valore più basso nella colonna.

- `MaximumValue`: un oggetto DecimalNumber.

  Il valore più alto nella colonna.
- `NumberOfNulls` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori null nella colonna.
- `NumberOfDistinctValues` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori distinti in una colonna.

DoubleColumnStatisticsData struttura

Definisce le statistiche di colonna supportate per le colonne di dati con numeri a virgola mobile.

Campi

- `MinimumValue`: numero (doppio).

  Il valore più basso nella colonna.
- `MaximumValue`: numero (doppio).

  Il valore più alto nella colonna.
- `NumberOfNulls` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori null nella colonna.
- `NumberOfDistinctValues` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori distinti in una colonna.

LongColumnStatisticsData struttura

Definisce le statistiche di colonna supportate per le colonne di dati interi.

Campi

- `MinimumValue`: numero (lungo).

  Il valore più basso nella colonna.
- `MaximumValue`: numero (lungo).

Il valore più alto nella colonna.

- `NumberOfNulls` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori null nella colonna.

- `NumberOfDistinctValues` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori distinti in una colonna.

StringColumnStatisticsData struttura

Definisce le statistiche di colonna supportate per i valori dei dati di sequenza.

Campi

- `MaximumLength` – Obbligatorio: numero (long), non superiore a Nessuno.

  La dimensione della stringa più lunga nella colonna.

- `AverageLength`: obbligatorio: numero (long), non superiore a Nessuno.

  La lunghezza media della stringa nella colonna.

- `NumberOfNulls` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori null nella colonna.

- `NumberOfDistinctValues` – Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori distinti in una colonna.

BinaryColumnStatisticsData struttura

Definisce le statistiche di colonna supportate per i valori dei dati di sequenza di bit.

Campi

- `MaximumLength` – Obbligatorio: numero (long), non superiore a Nessuno.

  La dimensione della sequenza di bit più lunga nella colonna.

- `AverageLength`. Obbligatorio: numero (long), non superiore a Nessuno.

La lunghezza media della sequenza di bit nella colonna.

- `NumberOfNulls`. Obbligatorio: numero (long), non superiore a Nessuno.

  Il numero di valori null nella colonna.

Modelli di stringa

L'API usa le seguenti espressioni regolari per definire i contenuti validi per vari membri e parametri di stringa:

- Modello di stringa a una riga: "`[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF \uDFFF\t]*`"
- IndirizzoModello di stringa a più righe per indirizzo URI: "`[\u0020-\uD7FF\uE000-\uFFFD \uD800\uDC00-\uDBFF\uDFFF\r\n\t]*`"
- Modello di stringa Logstash Grok: "`[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF \uDFFF\r\t]*`"
- Modello di stringa identificatore: "`[A-Za-z_][A-Za-z0-9_]*`"
- Modello di stringa ARN AWS IAM: "`arn:aws:iam::\d{12}:role/.*`"
- Modello di stringa di versione: "`^[a-zA-Z0-9-_]+$`"
- Modello di stringa gruppo di log: "`[\.\-_/#A-Za-z0-9]+`"
- Modello di stringa flusso di log: "`[^:*]*`"
- Pattern di stringa personalizzato n. 10: "`[^\r\n]`"
- Pattern di stringa personalizzato n. 11: "`[\p{L}\p{N}\p{P}]*`"
- Pattern di stringa personalizzato n. 12: "`[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}`"
- Pattern di stringa personalizzato n. 13: "`[a-zA-Z0-9-_$#.]+`"
- Pattern di stringa personalizzato n. 14: "`^\w+\.\w+\.\w+$`"
- Pattern di stringa personalizzato n. 15: "`^\w+\.\w+$`"
- Pattern di stringa personalizzato n. 16: "`^([2-3]|3[.]9)$`"
- Pattern di stringa personalizzato n. 17: "`arn:(aws|aws-us-gov|aws-cn):glue:.*`"
- Pattern di stringa personalizzato n. 18: "`(^arn:aws:iam::\w{12}:root)`"

- Pattern di stringa personalizzato n. 19: `"^arn:aws(-(cn|us-gov|iso(-[bef])?))?:iam::[0-9]{12}:role/.+"`
- Pattern di stringa personalizzato n. 20: `"arn:aws:kms:.*"`
- Pattern di stringa personalizzato n. 21: `"arn:aws[^:]*:iam::[0-9]*:role/.+"`
- Pattern di stringa personalizzato n. 22: `"[\.\-_A-Za-z0-9]+"`
- Pattern di stringa personalizzato n. 23: `"^s3://([^/]+)/([^/]+/)*([^/]+)$"`
- Pattern di stringa personalizzato n. 24: `".*"`
- Pattern di stringa personalizzato n. 25: `"[a-zA-Z0-9_.-]+"`
- Pattern di stringa personalizzato n. 26: `".*\S.*"`
- Pattern di stringa personalizzato n. 27: `"[a-zA-Z0-9-=._/@]+"`
- Pattern di stringa personalizzato n. 28: `"[1-9][0-9]*|[1-9][0-9]*-[1-9][0-9]*"`
- Pattern di stringa personalizzato n. 29: `"[\s\S]*"`
- Pattern di stringa personalizzato n. 30: `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n"'= ;])*"`
- Pattern di stringa personalizzato n. 31: `"[*A-Za-z0-9_-]*"`
- Pattern di stringa personalizzato n. 32: `"([\u0020-\u007E\r\s\n])*"`
- Pattern di stringa personalizzato n. 33: `"[A-Za-z0-9_-]*"`
- Pattern di stringa personalizzato n 34: `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n"'])*"`
- Modello di stringa personalizzato n. 35: `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n])*"`
- Schema di stringhe personalizzato #36 — `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\s])*»`
- Schema di stringhe personalizzato #37 — `"([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\r\n])*»`

Eccezioni

Questa sezione descrive eccezioni AWS Glue che è possibile utilizzare per individuare l'origine dei problemi e correggerli. Per ulteriori informazioni sui codici di errore HTTP e sulle stringhe per le eccezioni correlate al machine learning, vedere the section called “Eccezioni di machine learning AWS Glue”.

Struttura AccessDeniedException

L'accesso a una risorsa è stato rifiutato.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura AlreadyExistsException

Una risorsa da creare o aggiungere esiste già.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura ConcurrentModificationException

Due processi stanno tentando di modificare una risorsa contemporaneamente.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura ConcurrentRunsExceededException

Troppi processi vengono eseguiti simultaneamente.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura CrawlerNotRunningException

Il crawler specificato non è in esecuzione.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura CrawlerRunningException

L'operazione non è stata eseguita perché il crawler è già in esecuzione.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura CrawlerStoppingException

Il crawler specificato è in fase di arresto.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura EntityNotFoundException

Un'entità specificata non esiste.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

- `FromFederationSource`: booleano.

  Indica se l'eccezione si riferisce o meno a un'origine federata.

Struttura FederationSourceException

Un'origine di federazione ha riscontrato un errore.

Campi

- `FederationSourceErrorCode`: stringa UTF-8 (valori validi: `InvalidResponseException` | `OperationTimeoutException` | `OperationNotSupportedException` | `InternalServiceException` | `ThrottlingException`).

  Il codice di errore del problema.
- `Message`: stringa UTF-8.

  Il messaggio che descrive il problema.

Struttura FederationSourceRetryableException

Un'origine di federazione ha riscontrato un errore, ma l'operazione può essere ritentata.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura GlueEncryptionException

Un'operazione di crittografia non è riuscita.

Campi

- `Message`: stringa UTF-8.

  Il messaggio che descrive il problema.

Struttura IdempotentParameterMismatchException

Lo stesso identificatore univoco è stato associato a due record diversi.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura IllegalWorkflowStateException

Il flusso di lavoro non è valido per eseguire un'operazione richiesta.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura InternalServiceException

Si è verificato un errore di servizio interno.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura InvalidExecutionEngineException

È stato specificato un motore di esecuzione sconosciuto o non valido.

Campi

- `message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura InvalidInputException

L'input fornito non è valido.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.
- `FromFederationSource`: booleano.

  Indica se l'eccezione si riferisce o meno a un'origine federata.

Struttura InvalidStateException

Un errore che indica che i dati sono in uno stato non valido.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura InvalidTaskStatusTransitionException

La corretta transizione da un'attività a quella successiva non è riuscita.

Campi

- `message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura JobDefinitionErrorException

Una definizione di processo non è valida.

Campi

- `message`: stringa UTF-8.

Messaggio che descrive il problema.

Struttura JobRunInTerminalStateException

Lo stato terminale dell'esecuzione di un processo segnala un errore.

Campi

- `message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura JobRunInvalidStateTransitionException

L'esecuzione di un processo ha riscontrato una transizione non valida dallo stato di origine allo stato di destinazione.

Campi

- `jobRunId`: stringa UTF-8, non inferiore a 1 o superiore a 255 byte di lunghezza, corrispondente a Single-line string pattern.

  L'ID dell'esecuzione del processo in questione.
- `message`: stringa UTF-8.

  Messaggio che descrive il problema.
- `sourceState`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING`).

  Lo stato di origine.
- `targetState`: stringa UTF-8 (valori validi: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING`).

  Lo stato di destinazione.

Struttura JobRunNotInTerminalStateException

Un'esecuzione di processo non è in uno stato terminale.

Campi

- `message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura LateRunnerException

L'esecuzione di un processo è in ritardo.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura NoScheduleException

Non è presente una pianificazione applicabile.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura OperationTimeoutException

Timeout dell'operazione.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura ResourceNotReadyException

Una risorsa non era pronta per una transazione.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura ResourceNumberLimitExceededException

Il limite numerico di una risorsa è stato superato.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura SchedulerNotRunningException

Il pianificatore specificato non è in esecuzione.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura SchedulerRunningException

Il pianificatore specificato è già in esecuzione.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura SchedulerTransitioningException

Il pianificatore specificato è in transizione.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura UnrecognizedRunnerException

L'esecuzione del processo non è stata riconosciuta.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura ValidationException

Un valore non può essere convalidato.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

Struttura VersionMismatchException

Si è verificato un conflitto di versione.

Campi

- `Message`: stringa UTF-8.

  Messaggio che descrive il problema.

AWS Glue Esempi di codice API con AWS SDK

I seguenti esempi di codice mostrano come utilizzarlo AWS Glue con un kit di sviluppo AWS software (SDK).

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Nozioni di base

Salve AWS Glue

L'esempio di codice seguente mostra come iniziare a utilizzare AWS Glue.

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
namespace GlueActions;

public class HelloGlue
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonGlue>()
                .AddTransient<GlueWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloGlue>();
        var glueClient = host.Services.GetRequiredService<IAmazonGlue>();

        var request = new ListJobsRequest();

        var jobNames = new List<string>();

        do
        {
            var response = await glueClient.ListJobsAsync(request);
            jobNames.AddRange(response.JobNames);
            request.NextToken = response.NextToken;
        }
        while (request.NextToken is not null);

        Console.Clear();
        Console.WriteLine("Hello, Glue. Let's list your existing Glue Jobs:");
        if (jobNames.Count == 0)
        {
            Console.WriteLine("You don't have any AWS Glue jobs.");
        }
        else
        {
            jobNames.ForEach(Console.WriteLine);
        }
    }
}
```

- Per i dettagli sull'API, consulta la ListJobssezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Codice per il file CMake C MakeLists .txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_glue.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Codice per il file di origine hello_glue.cpp.

```
#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 *  A "Hello Glue" starter application which initializes an AWS Glue client and
 lists the
 *  AWS Glue job definitions.
 *
 *  main function
 *
 *  Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
```

```
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken;  // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }

            Aws::Glue::Model::ListJobsOutcome listRunsOutcome =
 glueClient.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());

                nextToken = listRunsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Your account has " << jobs.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < jobs.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << jobs[i] << std::endl;
        }
    }
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Per i dettagli sull'API, consulta la sezione API ListJobsReference AWS SDK for C++ .

Java

SDK per Java 2.x

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- Per i dettagli sull'API, consulta la ListJobssezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import { ListJobsCommand, GlueClient } from "@aws-sdk/client-glue";

const client = new GlueClient({});

export const main = async () => {
  const command = new ListJobsCommand({});

  const { JobNames } = await client.send(command);
  const formattedJobNames = JobNames.join("\n");
  console.log("Job names: ");
  console.log(formattedJobNames);
  return JobNames;
};
```

- Per i dettagli sull'API, consulta la ListJobssezione AWS SDK for JavaScript API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- Per i dettagli sulle API, consulta la ListJobsguida di riferimento all'API AWS SDK for Rust.

Esempi di codice

- Azioni per l' AWS Glue utilizzo degli AWS SDK
  - Crea un AWS Glue crawler utilizzando un SDK AWS
  - Creare una definizione AWS Glue di lavoro utilizzando un AWS SDK
  - Eliminare un AWS Glue crawler utilizzando un SDK AWS
  - Eliminare un database dall' AWS Glue Data Catalog utilizzo di un AWS SDK
  - Eliminare una definizione di AWS Glue processo utilizzando un AWS SDK
  - Eliminare una tabella da un AWS Glue Data Catalog database utilizzando un AWS SDK
  - Ottieni un AWS Glue crawler utilizzando un SDK AWS
  - Ottieni un database AWS Glue Data Catalog utilizzando un AWS SDK
  - Esegui un AWS Glue lavoro utilizzando un AWS SDK
  - Ottieni l'elenco dei database AWS Glue Data Catalog utilizzando un AWS SDK
  - Ottieni un lavoro AWS Glue Data Catalog utilizzando un AWS SDK
  - Esegui le esecuzioni di un AWS Glue lavoro utilizzando un AWS SDK
  - Ottieni tabelle da un database AWS Glue Data Catalog utilizzando un AWS SDK
  - Elenca le definizioni dei AWS Glue lavori utilizzando un AWS SDK
  - Avvia un AWS Glue crawler utilizzando un SDK AWS
  - Avvio di un AWS Glue processo utilizzando un AWS SDK
- Scenari per l' AWS Glue utilizzo degli AWS SDK
  - Inizia a eseguire AWS Glue crawler e job utilizzando un SDK AWS

Azioni per l' AWS Glue utilizzo degli AWS SDK

I seguenti esempi di codice mostrano come eseguire AWS Glue azioni individuali con gli AWS SDK. Questi estratti richiamano l' AWS Glue API e sono estratti di codice di programmi più grandi che devono essere eseguiti nel contesto. Ogni esempio include un collegamento a GitHub, dove è possibile trovare le istruzioni per la configurazione e l'esecuzione del codice.

Gli esempi seguenti includono solo le operazioni più comunemente utilizzate. Per un elenco completo, consulta la Documentazione di riferimento delle API AWS Glue.

Esempi

- Crea un AWS Glue crawler utilizzando un SDK AWS
- Creare una definizione AWS Glue di lavoro utilizzando un AWS SDK
- Eliminare un AWS Glue crawler utilizzando un SDK AWS
- Eliminare un database dall' AWS Glue Data Catalog utilizzo di un AWS SDK
- Eliminare una definizione di AWS Glue processo utilizzando un AWS SDK
- Eliminare una tabella da un AWS Glue Data Catalog database utilizzando un AWS SDK
- Ottieni un AWS Glue crawler utilizzando un SDK AWS
- Ottieni un database AWS Glue Data Catalog utilizzando un AWS SDK
- Esegui un AWS Glue lavoro utilizzando un AWS SDK
- Ottieni l'elenco dei database AWS Glue Data Catalog utilizzando un AWS SDK
- Ottieni un lavoro AWS Glue Data Catalog utilizzando un AWS SDK
- Esegui le esecuzioni di un AWS Glue lavoro utilizzando un AWS SDK
- Ottieni tabelle da un database AWS Glue Data Catalog utilizzando un AWS SDK
- Elenca le definizioni dei AWS Glue lavori utilizzando un AWS SDK
- Avvia un AWS Glue crawler utilizzando un SDK AWS
- Avvio di un AWS Glue processo utilizzando un AWS SDK

Crea un AWS Glue crawler utilizzando un SDK AWS

I seguenti esempi di codice mostrano come creare un AWS Glue crawler.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Create an AWS Glue crawler.
   /// </summary>
   /// <param name="crawlerName">The name for the crawler.</param>
   /// <param name="crawlerDescription">A description of the crawler.</param>
   /// <param name="role">The AWS Identity and Access Management (IAM) role to
   /// be assumed by the crawler.</param>
   /// <param name="schedule">The schedule on which the crawler will be
executed.</param>
   /// <param name="s3Path">The path to the Amazon Simple Storage Service
(Amazon S3)
   /// bucket where the Python script has been stored.</param>
   /// <param name="dbName">The name to use for the database that will be
   /// created by the crawler.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> CreateCrawlerAsync(
       string crawlerName,
       string crawlerDescription,
       string role,
       string schedule,
       string s3Path,
       string dbName)
   {
       var s3Target = new S3Target
       {
           Path = s3Path,
       };

       var targetList = new List<S3Target>
       {
```

```
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Per i dettagli sull'API, consulta la CreateCrawlersezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);
```

```
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
            return false;
        }
```

- Per i dettagli sull'API, consulta la CreateCrawlersezione AWS SDK for C++ API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;
import software.amazon.awssdk.services.glue.model.CrawlerTargets;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.S3Target;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <IAM> <s3Path> <cron> <dbName> <crawlerName>

                Where:
                    IAM - The ARN of the IAM role that has AWS Glue and S3
 permissions.\s
                    s3Path - The Amazon Simple Storage Service (Amazon S3) target
 that contains data (for example, CSV data).
                    cron - A cron expression used to specify the schedule  (i.e.,
 cron(15 12 * * ? *).
                    dbName - The database name.\s
                    crawlerName - The name of the crawler.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
```

```
        String crawlerName = args[4];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        glueClient.close();
    }

    public static void createGlueCrawler(GlueClient glueClient,
            String iam,
            String s3Path,
            String cron,
            String dbName,
            String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                    .path(s3Path)
                    .build();

            // Add the S3Target to a list.
            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);

            CrawlerTargets targets = CrawlerTargets.builder()
                    .s3Targets(targetList)
                    .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                    .databaseName(dbName)
                    .name(crawlerName)
                    .description("Created by the AWS Glue Java API")
                    .targets(targets)
                    .role(iam)
                    .schedule(cron)
                    .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la CreateCrawlersezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, consulta la CreateCrawlersezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String
) {
    val s3Target = S3Target {
        path = s3Path
    }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb = CrawlerTargets {
        s3Targets = targetList
    }

    val request = CreateCrawlerRequest {
        databaseName = dbName
        name = crawlerName
        description = "Created by the AWS Glue Kotlin API"
        targets = targetOb
        role = iam
        schedule = cron
    }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
```

```
}
```

- Per i dettagli sull'API, CreateCrawlerconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path):
Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- Per i dettagli sull'API, consulta la CreateCrawlersezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
 data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
 Access
                         Management (IAM) role that grants permission to let AWS
 Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
 crawler.
        :param db_prefix: The prefix to give any database tables that are created
 by
                          the crawler.
```

```
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta CreateCrawler AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
```

```
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end
```

- Per i dettagli sull'API, consulta la CreateCrawlersezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;
```

- Per i dettagli sulle API, consulta la CreateCrawlerguida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creare una definizione AWS Glue di lavoro utilizzando un AWS SDK

I seguenti esempi di codice mostrano come creare una definizione di AWS Glue processo.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
```

```
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Per i dettagli sull'API, consulta la CreateJobsezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Per i dettagli sull'API, consulta la CreateJobsezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Per creare un processo di trasformazione dei dati

L'esempio `create-job` seguente crea un processo di streaming che esegue uno script archiviato in S3.

```
aws glue create-job \
```

```
    --name my-testing-job \
    --role AWSGlueServiceRoleDefault \
    --command '{ \
        "Name": "gluestreaming", \
        "ScriptLocation": "s3://DOC-EXAMPLE-BUCKET/folder/" \
    }' \
    --region us-east-1 \
    --output json \
    --default-arguments '{ \
        "--job-language":"scala", \
        "--class":"GlueApp" \
    }' \
    --profile my-profile \
    --endpoint https://glue.us-east-1.amazonaws.com
```

Contenuto di `test_script.scala`.

```
import com.amazonaws.services.glue.ChoiceOption
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.ResolveSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
    def main(sysArgs: Array[String]) {
        val spark: SparkContext = new SparkContext()
        val glueContext: GlueContext = new GlueContext(spark)
        // @params: [JOB_NAME]
        val args = GlueArgParser.getResolvedOptions(sysArgs,
 Seq("JOB_NAME").toArray)
        Job.init(args("JOB_NAME"), glueContext, args.asJava)
        // @type: DataSource
        // @args: [database = "tempdb", table_name = "s3-source",
 transformation_ctx = "datasource0"]
        // @return: datasource0
        // @inputs: []
```

```
        val datasource0 = glueContext.getCatalogSource(database = "tempdb",
 tableName = "s3-source", redshiftTmpDir = "", transformationContext =
 "datasource0").getDynamicFrame()
        // @type: ApplyMapping
        // @args: [mapping = [("sensorid", "int", "sensorid", "int"),
 ("currenttemperature", "int", "currenttemperature", "int"), ("status", "string",
 "status", "string")], transformation_ctx = "applymapping1"]
        // @return: applymapping1
        // @inputs: [frame = datasource0]
        val applymapping1 = datasource0.applyMapping(mappings = Seq(("sensorid",
 "int", "sensorid", "int"), ("currenttemperature", "int", "currenttemperature",
 "int"), ("status", "string", "status", "string")), caseSensitive = false,
 transformationContext = "applymapping1")
        // @type: SelectFields
        // @args: [paths = ["sensorid", "currenttemperature", "status"],
 transformation_ctx = "selectfields2"]
        // @return: selectfields2
        // @inputs: [frame = applymapping1]
        val selectfields2 = applymapping1.selectFields(paths = Seq("sensorid",
 "currenttemperature", "status"), transformationContext = "selectfields2")
        // @type: ResolveChoice
        // @args: [choice = "MATCH_CATALOG", database = "tempdb", table_name =
 "my-s3-sink", transformation_ctx = "resolvechoice3"]
        // @return: resolvechoice3
        // @inputs: [frame = selectfields2]
        val resolvechoice3 = selectfields2.resolveChoice(choiceOption =
 Some(ChoiceOption("MATCH_CATALOG")), database = Some("tempdb"), tableName =
 Some("my-s3-sink"), transformationContext = "resolvechoice3")
        // @type: DataSink
        // @args: [database = "tempdb", table_name = "my-s3-sink",
 transformation_ctx = "datasink4"]
        // @return: datasink4
        // @inputs: [frame = resolvechoice3]
        val datasink4 = glueContext.getCatalogSink(database = "tempdb",
 tableName = "my-s3-sink", redshiftTmpDir = "", transformationContext =
 "datasink4").writeDynamicFrame(resolvechoice3)
        Job.commit()
    }
}
```

Output:

```
{
```

```
    "Name": "my-testing-job"
}
```

Per ulteriori informazioni, consulta Authoring Jobs in AWS Glue nella Glue Developer Guide.AWS

- Per i dettagli sull'API, consulta CreateJob AWS CLICommand Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, consulta la CreateJobsezione AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- Per i dettagli sull'API, consulta la CreateJobsezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
 that can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
 permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
 run as
                                part of the job. The script defines how the data
 is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
```

```
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta CreateJob AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end
```

```
# Creates a new job with the specified configuration.
#
# @param name [String] The name of the job.
# @param description [String] The description of the job.
# @param role_arn [String] The ARN of the IAM role to be used by the job.
# @param script_location [String] The location of the ETL script for the job.
# @return [void]
def create_job(name, description, role_arn, script_location)
  @glue_client.create_job(
    name: name,
    description: description,
    role: role_arn,
    command: {
      name: "glueetl",
      script_location: script_location,
      python_version: "3"
    },
    glue_version: "3.0"
  )
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not create job #{name}: \n#{e.message}")
  raise
end
```

- Per i dettagli sull'API, consulta la CreateJobsezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
```

```
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;
```

- Per i dettagli sulle API, consulta la CreateJobguida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Eliminare un AWS Glue crawler utilizzando un SDK AWS

I seguenti esempi di codice mostrano come eliminare un AWS Glue crawler.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Per i dettagli sull'API, DeleteCrawlerconsulta AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
 client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
```

- Per i dettagli sull'API, DeleteCrawlerconsulta AWS SDK for C++ API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, DeleteCrawlerconsulta AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Per i dettagli sull'API, DeleteCrawlerconsulta AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DeleteCrawler AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end
```

- Per i dettagli sull'API, DeleteCrawlerconsulta AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Per i dettagli sulle API, consulta il riferimento DeleteCrawlerall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Eliminare un database dall' AWS Glue Data Catalog utilizzo di un AWS SDK

Gli esempi di codice seguenti mostrano come eliminare un database da AWS Glue Data Catalog.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Per i dettagli sull'API, DeleteDatabaseconsulta AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Per i dettagli sull'API, DeleteDatabaseconsulta AWS SDK for C++ API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, DeleteDatabaseconsulta AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
    {
```

```
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Per i dettagli sull'API, DeleteDatabaseconsulta AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- Per i dettagli sull'API, consulta DeleteDatabase AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- Per i dettagli sull'API, DeleteDatabaseconsulta AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Per i dettagli sulle API, consulta il riferimento DeleteDatabaseall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Eliminare una definizione di AWS Glue processo utilizzando un AWS SDK

I seguenti esempi di codice mostrano come eliminare una definizione di AWS Glue processo e tutte le esecuzioni associate.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Per i dettagli sull'API, consulta la DeleteJobsezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Per i dettagli sull'API, consulta la DeleteJobsezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Per eliminare un processo

L'esempio `delete-job` seguente elimina un processo non più necessario.

```
aws glue delete-job \
    --job-name my-testing-job
```

Output:

```
{
    "JobName": "my-testing-job"
}
```

Per ulteriori informazioni, consulta Working with Jobs on the AWS Glue Console nella AWS Glue Developer Guide.

- Per i dettagli sull'API, consulta DeleteJob AWS CLICommand Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, consulta la DeleteJobsezione AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
```

```
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Per i dettagli sull'API, consulta la DeleteJobsezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DeleteJob AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
```

```
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Per i dettagli sull'API, consulta la DeleteJobsezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Per i dettagli sulle API, consulta la DeleteJobguida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Eliminare una tabella da un AWS Glue Data Catalog database utilizzando un AWS SDK

I seguenti esempi di codice mostrano come eliminare una tabella da un AWS Glue Data Catalog database.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Per i dettagli sull'API, consulta la DeleteTablesezione AWS SDK for .NET API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
```

```
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, consulta la DeleteTablesezione AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- Per i dettagli sull'API, consulta la DeleteTablesezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta DeleteTable AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Per i dettagli sull'API, consulta la DeleteTablesezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }
```

- Per i dettagli sulle API, consulta la DeleteTableguida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Ottieni un AWS Glue crawler utilizzando un SDK AWS

I seguenti esempi di codice mostrano come creare un AWS Glue crawler.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

       if (outcome.IsSuccess()) {
           Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
           std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                           crawlerState)
                   << "." << std::endl;
       }
       else {
           std::cerr << "Error retrieving a crawler.  "
                     << outcome.GetError().GetMessage() << std::endl;
           deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
           return false;
       }
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK for C++ API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetCrawler {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <crawlerName>

                Where:
                    crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        getSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }

    public static void getSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            Instant createDate = response.crawler().creationTime();

            // Convert the Instant to readable date
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the Crawler is " +
 createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun getSpecificCrawler(crawlerName: String?) {

    val request = GetCrawlerRequest {
        name = crawlerName
    }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
```

```
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

```
        return crawler
```

- Per i dettagli sull'API, consulta GetCrawler AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
 if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
```

```
end
```

- Per i dettagli sull'API, GetCrawlerconsulta AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
            let tmp_crawler = glue
                .get_crawler()
                .name(self.crawler())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

- Per i dettagli sulle API, consulta il riferimento GetCrawlerall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Ottieni un database AWS Glue Data Catalog utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come ottenere un database da AWS Glue Data Catalog.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetDatabaseRequest request;
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for C++ API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetDatabase {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <databaseName>

                Where:
                    databaseName - The name of the database.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String databaseName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        getSpecificDatabase(glueClient, databaseName);
```

```
        glueClient.close();
    }

    public static void getSpecificDatabase(GlueClient glueClient, String
 databaseName) {
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            GetDatabaseResponse response =
 glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " +
 createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun getSpecificDatabase(databaseName: String?) {

    val request = GetDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $databaseName = "doc-example-database-$uniqid";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

- Per i dettagli sull'API, consulta GetDatabase AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get database #{name}: \n#{e.message}")
  raise
  end
```

- Per i dettagli sull'API, GetDatabaseconsulta AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
```

```
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;
```

- Per i dettagli sulle API, consulta il riferimento GetDatabaseall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esegui un AWS Glue lavoro utilizzando un AWS SDK

I seguenti esempi di codice mostrano come eseguire un AWS Glue job.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
```

```
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }
```

- Per i dettagli sull'API, GetJobRunconsulta AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
```

```
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Per i dettagli sull'API, GetJobRunconsulta AWS SDK for C++ API Reference.

CLI

AWS CLI

Per ottenere informazioni relative all'esecuzione di un processo

L'esempio `get-job-run` seguente recupera le informazioni relative all'esecuzione di un processo.

```
aws glue get-job-run \
    --job-name "Combine legistators data" \
    --run-id jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e
```

Output:

```
{
    "JobRun": {
        "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
        "Attempt": 0,
        "JobName": "Combine legistators data",
        "StartedOn": 1602873931.255,
        "LastModifiedOn": 1602874075.985,
        "CompletedOn": 1602874075.985,
        "JobRunState": "SUCCEEDED",
        "Arguments": {
            "--enable-continuous-cloudwatch-log": "true",
            "--enable-metrics": "",
            "--enable-spark-ui": "true",
            "--job-bookmark-option": "job-bookmark-enable",
            "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
```

```
        },
        "PredecessorRuns": [],
        "AllocatedCapacity": 10,
        "ExecutionTime": 117,
        "Timeout": 2880,
        "MaxCapacity": 10.0,
        "WorkerType": "G.1X",
        "NumberOfWorkers": 10,
        "LogGroupName": "/aws-glue/jobs",
        "GlueVersion": "2.0"
    }
}
```

Per ulteriori informazioni, consulta Esecuzioni di processi nella Guida per gli sviluppatori di AWS Glue.

- Per i dettagli sull'API, consulta GetJobRun AWS CLICommand Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, GetJobRunconsulta AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- Per i dettagli sull'API, GetJobRunconsulta AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]
```

- Per i dettagli sull'API, consulta GetJobRun AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Per i dettagli sull'API, GetJobRunconsulta AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let get_job_run = || async {
            Ok::<JobRun, GlueMvpError>(
                glue.get_job_run()
                    .job_name(self.job())
                    .run_id(job_run_id.to_string())
                    .send()
                    .await
                    .map_err(GlueMvpError::from_glue_sdk)?
                    .job_run()
                    .ok_or_else(|| GlueMvpError::Unknown("Failed to get
job_run".into()))?
                    .to_owned(),
            )
        };

        let mut job_run = get_job_run().await?;
        let mut state =
job_run.job_run_state().unwrap_or(&unknown_state).to_owned();

        while matches!(
            state,
            JobRunState::Starting | JobRunState::Stopping | JobRunState::Running
        ) {
            info!(?state, "Waiting for job to finish");
            tokio::time::sleep(self.wait_delay).await;

            job_run = get_job_run().await?;
            state = job_run.job_run_state().unwrap_or(&unknown_state).to_owned();
        }
```

- Per i dettagli sulle API, consulta il riferimento GetJobRunall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Ottieni l'elenco dei database AWS Glue Data Catalog utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come ottenere un elenco di database da AWS Glue Data Catalog.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

CLI

AWS CLI

Per elencare le definizioni di alcuni o tutti i database del AWS Glue Data Catalog

L'esempio `get-databases` seguente restituisce informazioni sui database del Catalogo dati.

```
aws glue get-databases
```

Output:

```
{
    "DatabaseList": [
        {
            "Name": "default",
            "Description": "Default Hive database",
            "LocationUri": "file:/spark-warehouse",
            "CreateTime": 1602084052.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
```

```
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "flights-db",
            "CreateTime": 1587072847.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "legislators",
            "CreateTime": 1601415625.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "tempdb",
            "CreateTime": 1601498566.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
```

```
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        }
    ]
}
```

Per ulteriori informazioni, consulta Definizione di un database nel catalogo dati nella Guida per gli sviluppatori di AWS Glue.

- Per i dettagli sulle API, consultate GetDatabases AWS CLICommand Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const getDatabases = () => {
  const client = new GlueClient({});

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- Per i dettagli sull'API, consulta la GetDatabasessezione AWS SDK for JavaScript API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Ottieni un lavoro AWS Glue Data Catalog utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come ottenere un processo da AWS Glue Data Catalog.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

CLI

AWS CLI

Per recuperare le informazioni relative a un processo

L'esempio `get-job` seguente recupera le informazioni relative a un processo.

```
aws glue get-job \
    --job-name my-testing-job
```

Output:

```
{
    "Job": {
        "Name": "my-testing-job",
        "Role": "Glue_DefaultRole",
        "CreatedOn": 1602805698.167,
        "LastModifiedOn": 1602805698.167,
        "ExecutionProperty": {
            "MaxConcurrentRuns": 1
        },
        "Command": {
            "Name": "gluestreaming",
            "ScriptLocation": "s3://janetst-bucket-01/Scripts/test_script.scala",
            "PythonVersion": "2"
        },
        "DefaultArguments": {
            "--class": "GlueApp",
            "--job-language": "scala"
        },
        "MaxRetries": 0,
```

```
        "AllocatedCapacity": 10,
        "MaxCapacity": 10.0,
        "GlueVersion": "1.0"
    }
}
```

Per ulteriori informazioni, consulta Processi nella Guida per gli sviluppatori di AWS Glue.

- Per i dettagli sull'API, consulta GetJob AWS CLICommand Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const getJob = (jobName) => {
  const client = new GlueClient({});

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, GetJobconsulta AWS SDK for JavaScript API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esegui le esecuzioni di un AWS Glue lavoro utilizzando un AWS SDK

I seguenti esempi di codice mostrano come eseguire un AWS Glue processo.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }
```

```
        return jobRuns;
    }
```

- Per i dettagli sull'API, GetJobRunsconsulta AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
       getJobRunsRequest.SetJobName(jobName);

       Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
               getJobRunsRequest);

       if (jobRunsOutcome.IsSuccess()) {
           std::vector<Aws::Glue::Model::JobRun> jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
           std::cout << "There are " << jobRuns.size() << " runs in the job '"
                     <<
                     jobName << "'." << std::endl;

           for (size_t i = 0; i < jobRuns.size(); ++i) {
               std::cout << "   " << i + 1 << ". " << jobRuns[i].GetJobName()
                         << std::endl;
           }
```

```
            int runIndex = askQuestionForIntRange(
                    Aws::String("Enter a number between 1 and ") +
                    std::to_string(jobRuns.size()) +
                    " to see details for a run: ",
                    1, static_cast<int>(jobRuns.size()));
            jobRunID = jobRuns[runIndex - 1].GetId();
        }
        else {
            std::cerr << "Error getting job runs. "
                      << jobRunsOutcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Per i dettagli sull'API, GetJobRunsconsulta AWS SDK for C++ API Reference.

CLI

AWS CLI

Per ottenere informazioni su tutte le esecuzioni di processo per un determinato processo

L'esempio `get-job-runs` seguente recupera informazioni sulle esecuzioni di processo per un determinato processo.

```
aws glue get-job-runs \
    --job-name "my-testing-job"
```

Output:

```
{
    "JobRuns": [
        {
            "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
            "Attempt": 0,
            "JobName": "my-testing-job",
            "StartedOn": 1602873931.255,
            "LastModifiedOn": 1602874075.985,
            "CompletedOn": 1602874075.985,
            "JobRunState": "SUCCEEDED",
```

```
            "Arguments": {
                "--enable-continuous-cloudwatch-log": "true",
                "--enable-metrics": "",
                "--enable-spark-ui": "true",
                "--job-bookmark-option": "job-bookmark-enable",
                "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
            },
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 117,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_2",
            "Attempt": 2,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "JobName": "my-testing-job",
            "StartedOn": 1602811168.496,
            "LastModifiedOn": 1602811282.39,
            "CompletedOn": 1602811282.39,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
                Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                Request ID: 021AAB703DB20A2D;
                S3 Extended Request ID: teZk24Y09TkXzBvMPG502L5VJBhe9DJuWA9/
TXtuGOqfByajkfL/Tlqt5JBGdEGpigAqzdMDM/U=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 110,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
```

```
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "Attempt": 1,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f",
            "JobName": "my-testing-job",
            "StartedOn": 1602811020.518,
            "LastModifiedOn": 1602811138.364,
            "CompletedOn": 1602811138.364,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
                 Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                 Request ID: 2671D37856AE7ABB;
                 S3 Extended Request ID: RLJCJw20brV
+PpC6GpORahyF2fp9flB5SSb2bTGPnUSPVizLXRl1PN3QZldb+v1o9qRVktNYbW8=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 113,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        }
    ]
}
```

Per ulteriori informazioni, consulta Esecuzioni di processi nella Guida per gli sviluppatori di AWS Glue.

- Per i dettagli sull'API, consulta GetJobRuns AWS CLICommand Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, GetJobRunsconsulta AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
 Result
    {
        $arguments = ['JobName' => $jobName];
```

```
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- Per i dettagli sull'API, GetJobRunsconsulta AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]
```

- Per i dettagli sull'API, consulta GetJobRuns AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
```

```
def get_job_runs(job_name)
  response = @glue_client.get_job_runs(job_name: job_name)
  response.job_runs
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get job runs: \n#{e.message}")
end
```

- Per i dettagli sull'API, GetJobRunsconsulta AWS SDK for Ruby API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Ottieni tabelle da un database AWS Glue Data Catalog utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come ottenere tabelle da un database in AWS Glue Data Catalog.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Get a list of tables for an AWS Glue database.
/// </summary>
/// <param name="dbName">The name of the database.</param>
```

```
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }
```

- Per i dettagli sull'API, consulta la GetTablessezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);
```

```
       if (outcome.IsSuccess()) {
           const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
           std::cout << "The database contains " << tables.size()
                     << (tables.size() == 1 ?
                         " table." : "tables.") << std::endl;
           std::cout << "Here is a list of the tables in the database.";
           for (size_t index = 0; index < tables.size(); ++index) {
               std::cout << "    " << index + 1 << ":  " <<
tables[index].GetName()
                         << std::endl;
           }

           if (!tables.empty()) {
               int tableIndex = askQuestionForIntRange(
                       "Enter an index to display the database detail ",
                       1, static_cast<int>(tables.size()));
               std::cout << tables[tableIndex -
1].Jsonize().View().WriteReadable()
                         << std::endl;
           }
       }
       else {
           std::cerr << "Error getting the tables. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
           return false;
       }
```

- Per i dettagli sull'API, consulta la GetTablessezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Per elencare le definizioni di alcune o tutte le tabelle del database specificato

L'esempio `get-tables` seguente restituisce le informazioni relative alle tabelle del database specificato.

```
aws glue get-tables --database-name 'tempdb'
```

Output:

```
{
    "TableList": [
        {
            "Name": "my-s3-sink",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730539.0,
            "UpdateTime": 1602730539.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/test-s3-output/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
```

```
        {
            "Name": "s3-source",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730658.0,
            "UpdateTime": 1602730658.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
        {
            "Name": "test-kinesis-input",
            "DatabaseName": "tempdb",
            "CreateTime": 1601507001.0,
            "UpdateTime": 1601507001.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
```

```
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "my-testing-stream",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "Parameters": {
                    "kinesisUrl": "https://kinesis.us-east-1.amazonaws.com",
                    "streamName": "my-testing-stream",
                    "typeOfData": "kinesis"
                },
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        }
    ]
}
```

Per ulteriori informazioni, consulta Definizione delle tabelle nel AWS Glue Data Catalog nella AWS Glue Developer Guide.

- Per i dettagli sulle API, consulta GetTables AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbName> <tableName>

                Where:
                    dbName - The database name.\s
                    tableName - The name of the table.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbName = args[0];
        String tableName = args[1];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        getGlueTable(glueClient, dbName, tableName);
        glueClient.close();
    }

    public static void getGlueTable(GlueClient glueClient, String dbName, String
 tableName) {
        try {
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .databaseName(dbName)
                    .name(tableName)
                    .build();

            GetTableResponse tableResponse = glueClient.getTable(tableRequest);
            Instant createDate = tableResponse.table().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the table is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la GetTablessezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, consulta la GetTablessezione AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);
```

```
    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Per i dettagli sull'API, consulta la GetTablessezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]
```

- Per i dettagli sull'API, consulta GetTables AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
```

```
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end
```

- Per i dettagli sull'API, consulta la GetTablessezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

- Per i dettagli sulle API, consulta la GetTablesguida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Elenca le definizioni dei AWS Glue lavori utilizzando un AWS SDK

I seguenti esempi di codice mostrano come elencare le definizioni dei AWS Glue processi.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }
```

- Per i dettagli sull'API, consulta la ListJobssezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::ListJobsRequest listJobsRequest;
       Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
               listJobsRequest);

       if (listRunsOutcome.IsSuccess()) {
           const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
           std::cout << "Your account has " << jobNames.size() << " jobs."
                     << std::endl;
           for (size_t i = 0; i < jobNames.size(); ++i) {
               std::cout << "   " << i + 1 << ". " << jobNames[i] << std::endl;
           }
           int jobIndex = askQuestionForIntRange(
                   Aws::String("Enter a number between 1 and ") +
                   std::to_string(jobNames.size()) +
                   " to see the list of runs for a job: ",
                   1, static_cast<int>(jobNames.size()));

           jobName = jobNames[jobIndex - 1];
       }
       else {
           std::cerr << "Error listing jobs. "
                     << listRunsOutcome.GetError().GetMessage()
                     << std::endl;
       }
```

- Per i dettagli sull'API, consulta la ListJobssezione AWS SDK for C++ API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const listJobs = () => {
  const client = new GlueClient({});

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- Per i dettagli sull'API, consulta la ListJobssezione AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
```

```
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
 Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- Per i dettagli sull'API, consulta la ListJobssezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]
```

- Per i dettagli sull'API, consulta ListJobs AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
```

```
    @logger = logger
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end
```

- Per i dettagli sull'API, consulta la ListJobssezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- Per i dettagli sulle API, consulta la ListJobsguida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Avvia un AWS Glue crawler utilizzando un SDK AWS

I seguenti esempi di codice mostrano come avviare un AWS Glue crawler.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Start an AWS Glue crawler.
/// </summary>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StartCrawlerAsync(string crawlerName)
{
    var crawlerRequest = new StartCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Per i dettagli sull'API, consulta la StartCrawlersezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::StartCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


       if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                   outcome.GetError().GetErrorType())) {
           if (!outcome.IsSuccess()) {
               std::cout << "Crawler was already started." << std::endl;
           }
           else {
               std::cout << "Successfully started crawler." << std::endl;
           }

           std::cout << "This may take a while to run." << std::endl;

           Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
           int iterations = 0;
           while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
               std::this_thread::sleep_for(std::chrono::seconds(1));
```

```
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                    crawlerState)
                              << ". After " << iterations
                              << " seconds elapsed."
                              << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                              << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

```
```

- Per i dettagli sull'API, consulta la StartCrawlersezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Per avviare un crawler

L'esempio `start-crawler` seguente avvia un crawler.

```
aws glue start-crawler --name my-crawler
```

Output:

```
None
```

Per ulteriori informazioni, consulta Definizione di crawler nella Guida per gli sviluppatori di AWS Glue.

- Per i dettagli sull'API, consulta StartCrawler AWS CLICommand Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <crawlerName>

                Where:
                    crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        startSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.startCrawler(crawlerRequest);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la StartCrawlersezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, consulta la StartCrawlersezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun startSpecificCrawler(crawlerName: String?) {

    val request = StartCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- Per i dettagli sull'API, StartCrawlerconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);
```

```
    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Per i dettagli sull'API, consulta la StartCrawlersezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Per i dettagli sull'API, consulta StartCrawler AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
```

```
    raise
  end
```

- Per i dettagli sull'API, consulta la StartCrawlersezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let start_crawler =
glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

- Per i dettagli sulle API, consulta la StartCrawlerguida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Avvio di un AWS Glue processo utilizzando un AWS SDK

I seguenti esempi di codice mostrano come avviare l'esecuzione di un AWS Glue job.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Nozioni di base su crawler e processi

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Start an AWS Glue job run.
/// </summary>
/// <param name="jobName">The name of the job.</param>
/// <returns>A string representing the job run Id.</returns>
public async Task<string> StartJobRunAsync(
    string jobName,
    string inputDatabase,
    string inputTable,
    string bucketName)
{
    var request = new StartJobRunRequest
    {
        JobName = jobName,
        Arguments = new Dictionary<string, string>
        {
            {"--input_database", inputDatabase},
            {"--input_table", inputTable},
            {"--output_bucket_url", $"s3://{bucketName}/"}
        }
    };
```

```
        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }
```

- Per i dettagli sull'API, StartJobRunconsulta AWS SDK for .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::StartJobRunRequest request;
       request.SetJobName(JOB_NAME);

       Aws::Map<Aws::String, Aws::String> arguments;
       arguments["--input_database"] = CRAWLER_DATABASE_NAME;
       arguments["--input_table"] = tableName;
       arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
       request.SetArguments(arguments);

       Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully started the job." << std::endl;

           Aws::String jobRunId = outcome.GetResult().GetJobRunId();
```

```
            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<
```

```
Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                        jobRunState) <<
                                ". " << iterator <<
                                " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                         clientConfig);
            return false;
        }
```

- Per i dettagli sull'API, StartJobRunconsulta AWS SDK for C++ API Reference.

CLI

AWS CLI

Per avviare l'esecuzione di un processo

L'esempio `start-job-run` seguente avvia un processo.

```
aws glue start-job-run \
    --job-name my-job
```

Output:

```
{
    "JobRunId":
 "jr_22208b1f44eb5376a60569d4b21dd20fcb8621e1a366b4e7b2494af764b82ded"
}
```

Per ulteriori informazioni, consulta Creazione di processi nella Guida per gli sviluppatori di AWS Glue.

- Per i dettagli sull'API, consulta StartJobRun AWS CLICommand Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};
```

- Per i dettagli sull'API, StartJobRunconsulta AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl): Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- Per i dettagli sull'API, StartJobRunconsulta AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_job_run(self, name, input_database, input_table,
 output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
 tables
                               that describe the source data. This is typically
 created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
 the
            # Python ETL script to determine the location of input and output
 data.
```

```
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]
```

- Per i dettagli sull'API, consulta StartJobRun AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
```

```
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end
```

- Per i dettagli sull'API, StartJobRunconsulta AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        let job_run_output = glue
            .start_job_run()
```

```
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .get(0)
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

- Per i dettagli sulle API, consulta il riferimento StartJobRunall'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scenari per l' AWS Glue utilizzo degli AWS SDK

I seguenti esempi di codice mostrano come implementare scenari comuni AWS Glue con gli AWS SDK. Questi scenari mostrano come eseguire attività specifiche richiamando più funzioni all'interno. AWS Glue Ogni scenario include un collegamento a GitHub, dove è possibile trovare istruzioni su come configurare ed eseguire il codice.

Esempi

- Inizia a eseguire AWS Glue crawler e job utilizzando un SDK AWS

Inizia a eseguire AWS Glue crawler e job utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come:

- Crea un crawler che esegue la scansione di un bucket Amazon S3 pubblico e genera un database di metadati in formato CSV.
- Elenca le informazioni su database e tabelle nel tuo AWS Glue Data Catalog.
- Crea un processo per estrarre i dati CSV dal bucket S3, trasformare i dati e caricare l'output in formato JSON in un altro bucket S3.
- Elenca le informazioni sulle esecuzioni dei processi, visualizza i dati trasformati e pulisci le risorse.

Per ulteriori informazioni, consulta Tutorial: Guida introduttiva a AWS Glue Studio.

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea una classe che racchiuda le AWS Glue funzioni utilizzate nello scenario.

```
using System.Net;

namespace GlueActions;

public class GlueWrapper
{
    private readonly IAmazonGlue _amazonGlue;

    /// <summary>
    /// Constructor for the AWS Glue actions wrapper.
    /// </summary>
    /// <param name="amazonGlue"></param>
    public GlueWrapper(IAmazonGlue amazonGlue)
    {
```

```
        _amazonGlue = amazonGlue;
    }

    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
(Amazon S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
```

```
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
```

```
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
```

```
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }
```

```
    /// <summary>
    /// Get information about the state of an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A value describing the state of the crawler.</returns>
    public async Task<CrawlerState> GetCrawlerStateAsync(string crawlerName)
    {
        var response = await _amazonGlue.GetCrawlerAsync(
            new GetCrawlerRequest { Name = crawlerName });
        return response.Crawler.State;
    }


    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }


    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
 { JobName = jobName, RunId = jobRunId });
        return response.JobRun;
```

```
    }


    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }


    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();
```

```
        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }


    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }


    /// <summary>
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }

}
```

Creazione di una classe che esegue lo scenario.

```
global using Amazon.Glue;
global using GlueActions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
```

```
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.Glue.Model;
using Amazon.S3;
using Amazon.S3.Model;

namespace GlueBasics;

public class GlueBasics
{
    private static ILogger logger = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonGlue>()
            .AddTransient<GlueWrapper>()
            .AddTransient<UiWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
        .CreateLogger<GlueBasics>();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // These values are stored in settings.json
```

```
        // Once you have run the CDK script to deploy the resources,
        // edit the file to set "BucketName", "RoleName", and "ScriptURL"
        // to the appropriate values. Also set "CrawlerName" to the name
        // you want to give the crawler when it is created.
        string bucketName = _configuration["BucketName"]!;
        string bucketUrl = _configuration["BucketUrl"]!;
        string crawlerName = _configuration["CrawlerName"]!;
        string roleName = _configuration["RoleName"]!;
        string sourceData = _configuration["SourceData"]!;
        string dbName = _configuration["DbName"]!;
        string cron = _configuration["Cron"]!;
        string scriptUrl = _configuration["ScriptURL"]!;
        string jobName = _configuration["JobName"]!;

        var wrapper = host.Services.GetRequiredService<GlueWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UiWrapper>();

        uiWrapper.DisplayOverview();
        uiWrapper.PressEnter();

        // Create the crawler and wait for it to be ready.
        uiWrapper.DisplayTitle("Create AWS Glue crawler");
        Console.WriteLine("Let's begin by creating the AWS Glue crawler.");

        var crawlerDescription = "Crawler created for the AWS Glue Basics
scenario.";
        var crawlerCreated = await wrapper.CreateCrawlerAsync(crawlerName,
crawlerDescription, roleName, cron, sourceData, dbName);
        if (crawlerCreated)
        {
            Console.WriteLine($"The crawler: {crawlerName} has been created. Now
let's wait until it's ready.");
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't create crawler {crawlerName}.");
```

```
            return; // Exit the application.
        }

        uiWrapper.DisplayTitle("Start AWS Glue crawler");
        Console.WriteLine("Now let's wait until the crawler has successfully
started.");
        var crawlerStarted = await wrapper.StartCrawlerAsync(crawlerName);
        if (crawlerStarted)
        {
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't start the crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.PressEnter();

        Console.WriteLine($"\nLet's take a look at the database: {dbName}");
        var database = await wrapper.GetDatabaseAsync(dbName);

        if (database != null)
        {
            uiWrapper.DisplayTitle($"{database.Name} Details");
            Console.WriteLine($"{database.Name} created on
{database.CreateTime}");
            Console.WriteLine(database.Description);
        }

        uiWrapper.PressEnter();

        var tables = await wrapper.GetTablesAsync(dbName);
        if (tables.Count > 0)
        {
            tables.ForEach(table =>
            {
```

```
                Console.WriteLine($"{table.Name}\tCreated:
{table.CreateTime}\tUpdated: {table.UpdateTime}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create AWS Glue job");
        Console.WriteLine("Creating a new AWS Glue job.");
        var description = "An AWS Glue job created using the AWS SDK for .NET";
        await wrapper.CreateJobAsync(dbName, tables[0].Name, bucketUrl, jobName,
roleName, description, scriptUrl);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Starting AWS Glue job");
        Console.WriteLine("Starting the new AWS Glue job...");
        var jobRunId = await wrapper.StartJobRunAsync(jobName, dbName,
tables[0].Name, bucketName);
        var jobRunComplete = false;
        var jobRun = new JobRun();
        do
        {
            jobRun = await wrapper.GetJobRunAsync(jobName, jobRunId);
            if (jobRun.JobRunState == "SUCCEEDED" || jobRun.JobRunState ==
"STOPPED" ||
                jobRun.JobRunState == "FAILED" || jobRun.JobRunState ==
"TIMEOUT")
            {
                jobRunComplete = true;
            }
        } while (!jobRunComplete);

        uiWrapper.DisplayTitle($"Data in {bucketName}");

        // Get the list of data stored in the S3 bucket.
        var s3Client = new AmazonS3Client();

        var response = await s3Client.ListObjectsAsync(new ListObjectsRequest
{ BucketName = bucketName });
        response.S3Objects.ForEach(s3Object =>
        {
            Console.WriteLine(s3Object.Key);
        });
```

```
        uiWrapper.DisplayTitle("AWS Glue jobs");
        var jobNames = await wrapper.ListJobsAsync();
        jobNames.ForEach(jobName =>
        {
            Console.WriteLine(jobName);
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Get AWS Glue job run information");
        Console.WriteLine("Getting information about the AWS Glue job.");
        var jobRuns = await wrapper.GetJobRunsAsync(jobName);

        jobRuns.ForEach(jobRun =>
        {

 Console.WriteLine($"{jobRun.JobName}\t{jobRun.JobRunState}\t{jobRun.CompletedOn}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Deleting resources");
        Console.WriteLine("Deleting the AWS Glue job used by the example.");
        await wrapper.DeleteJobAsync(jobName);

        Console.WriteLine("Deleting the tables from the database.");
        tables.ForEach(async table =>
        {
            await wrapper.DeleteTableAsync(dbName, table.Name);
        });

        Console.WriteLine("Deleting the database.");
        await wrapper.DeleteDatabaseAsync(dbName);

        Console.WriteLine("Deleting the AWS Glue crawler.");
        await wrapper.DeleteCrawlerAsync(crawlerName);

        Console.WriteLine("The AWS Glue scenario has completed.");
        uiWrapper.PressEnter();
    }
}
```

```
namespace GlueBasics;

public class UiWrapper
{
    public readonly string SepBar = new string('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Amazon Glue: get started with crawlers and jobs");

        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t 1. Create a crawler, pass it the IAM role and the
 URL to the public S3 bucket that contains the source data");
        Console.WriteLine("\t 2. Start the crawler.");
        Console.WriteLine("\t 3. Get the database created by the crawler and the
 tables in the database.");
        Console.WriteLine("\t 4. Create a job.");
        Console.WriteLine("\t 5. Start a job run.");
        Console.WriteLine("\t 6. Wait for the job run to complete.");
        Console.WriteLine("\t 7. Show the data stored in the bucket.");
        Console.WriteLine("\t 8. List jobs for the account.");
        Console.WriteLine("\t 9. Get job run details for the job that was run.");
        Console.WriteLine("\t10. Delete the demo job.");
        Console.WriteLine("\t11. Delete the database and tables created for the
 demo.");
        Console.WriteLine("\t12. Delete the crawler.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPlease press <Enter> to continue. ");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
```

```
    /// <param name="strToCenter">The string to center on the screen.</param>
    /// <returns>The string padded to make it center on the screen.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for .NET .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns

- GetTables
- ListJobs
- StartCrawler
- StartJobRun

C++

SDK per C++

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Scenario which demonstrates using AWS Glue to add a crawler and run a job.
/*!
 \\sa runGettingStartedWithGlueScenario()
 \param bucketName: An S3 bucket created in the setup.
 \param roleName: An AWS Identity and Access Management (IAM) role created in the
 setup.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String
 &bucketName,
                                                     const Aws::String &roleName,
                                                     const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::Glue::GlueClient client(clientConfig);

    Aws::String roleArn;
    if (!getRoleArn(roleName, roleArn, clientConfig)) {
        std::cerr << "Error getting role ARN for role." << std::endl;
        return false;
    }

    // 1. Upload the job script to the S3 bucket.
    {
```

```
        std::cout << "Uploading the job script '"
                  << AwsDoc::Glue::PYTHON_SCRIPT
                  << "'." << std::endl;

        if (!AwsDoc::Glue::uploadFile(bucketName,
                                      AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                      AwsDoc::Glue::PYTHON_SCRIPT,
                                      clientConfig)) {
            std::cerr << "Error uploading the job file." << std::endl;
            return false;
        }
    }

    // 2. Create a crawler.
    {
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
            return false;
        }
    }

    // 3. Get a crawler.
```

```
    {
        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 4. Start a crawler.
    {
        Aws::Glue::Model::StartCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


        if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                    outcome.GetError().GetErrorType())) {
            if (!outcome.IsSuccess()) {
                std::cout << "Crawler was already started." << std::endl;
            }
            else {
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;
```

```
            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                      crawlerState)
                              << ". After " << iterations
                              << " seconds elapsed."
                              << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                              << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
            return false;
        }
    }

    // 5. Get a database.
    {
        Aws::Glue::Model::GetDatabaseRequest request;
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
            return false;
        }
    }

    // 6. Get tables.
    Aws::String tableName;
    {
        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);

        if (outcome.IsSuccess()) {
            const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
            std::cout << "The database contains " << tables.size()
```

```
                    << (tables.size() == 1 ?
                        " table." : "tables.") << std::endl;
            std::cout << "Here is a list of the tables in the database.";
            for (size_t index = 0; index < tables.size(); ++index) {
                std::cout << "    " << index + 1 << ":  " <<
tables[index].GetName()
                          << std::endl;
            }

            if (!tables.empty()) {
                int tableIndex = askQuestionForIntRange(
                        "Enter an index to display the database detail ",
                        1, static_cast<int>(tables.size()));
                std::cout << tables[tableIndex -
1].Jsonize().View().WriteReadable()
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error getting the tables. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 7. Create a job.
    {
        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 8. Start a job run.
    {
        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
```

```
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                         jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
```

```
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                         clientConfig);
            return false;
        }
    }

    // 9. List the output data stored in the S3 bucket.
    {
        Aws::S3::S3Client s3Client;
        Aws::S3::Model::ListObjectsRequest request;
        request.SetBucket(bucketName);
        request.SetPrefix(OUTPUT_FILE_PREFIX);

        Aws::S3::Model::ListObjectsOutcome outcome =
s3Client.ListObjects(request);

        if (outcome.IsSuccess()) {
            const std::vector<Aws::S3::Model::Object> &objects =
outcome.GetResult().GetContents();
            std::cout << "Data from your job is in " << objects.size() <<
                      " files in the S3 bucket, " << bucketName << "." <<
std::endl;

            for (size_t i = 0; i < objects.size(); ++i) {
                std::cout << "    " << i + 1 << ". " << objects[i].GetKey()
                          << std::endl;
            }

            int objectIndex = askQuestionForIntRange(
                    std::string(
                            "Enter the number of a block to download it and see
the first ") +
                    std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
                    " lines of JSON output in the block: ", 1,
                    static_cast<int>(objects.size()));
```

```
            Aws::String objectKey = objects[objectIndex - 1].GetKey();

            std::stringstream stringStream;
            if (getObjectFromBucket(bucketName, objectKey, stringStream,
                                    clientConfig)) {
                for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream;
++i) {
                    std::string line;
                    std::getline(stringStream, line);
                    std::cout << "    " << line << std::endl;
                }
            }
            else {
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                             clientConfig);
                return false;
            }
        }
        else {
            std::cerr << "Error listing objects. " <<
outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    // 10. List all the jobs.
    Aws::String jobName;
    {
        Aws::Glue::Model::ListJobsRequest listJobsRequest;
        Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                listJobsRequest);

        if (listRunsOutcome.IsSuccess()) {
            const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
            std::cout << "Your account has " << jobNames.size() << " jobs."
                      << std::endl;
            for (size_t i = 0; i < jobNames.size(); ++i) {
                std::cout << "   " << i + 1 << ". " << jobNames[i] << std::endl;
            }
            int jobIndex = askQuestionForIntRange(
                    Aws::String("Enter a number between 1 and ") +
```

```
                    std::to_string(jobNames.size()) +
                    " to see the list of runs for a job: ",
                    1, static_cast<int>(jobNames.size()));

            jobName = jobNames[jobIndex - 1];
        }
        else {
            std::cerr << "Error listing jobs. "
                      << listRunsOutcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    // 11. Get the job runs for a job.
    Aws::String jobRunID;
    if (!jobName.empty()) {
        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
                getJobRunsRequest);

        if (jobRunsOutcome.IsSuccess()) {
            std::vector<Aws::Glue::Model::JobRun> jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
            std::cout << "There are " << jobRuns.size() << " runs in the job '"
                      <<
                      jobName << "'." << std::endl;

            for (size_t i = 0; i < jobRuns.size(); ++i) {
                std::cout << "   " << i + 1 << ". " << jobRuns[i].GetJobName()
                          << std::endl;
            }

            int runIndex = askQuestionForIntRange(
                    Aws::String("Enter a number between 1 and ") +
                    std::to_string(jobRuns.size()) +
                    " to see details for a run: ",
                    1, static_cast<int>(jobRuns.size()));
            jobRunID = jobRuns[runIndex - 1].GetId();
        }
        else {
            std::cerr << "Error getting job runs. "
                      << jobRunsOutcome.GetError().GetMessage()
```

```
                      << std::endl;
        }
    }

    // 12. Get a single job run.
    if (!jobRunID.empty()) {
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
 jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                        clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
 \\sa deleteAssets()
 \param crawler: Name of an AWS Glue crawler.
 \param database: The name of an AWS Glue database.
 \param job: The name of an AWS Glue job.
 \param bucketName: The name of an S3 bucket.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
 &database,
```

```
                                const Aws::String &job, const Aws::String
&bucketName,
                                const Aws::Client::ClientConfiguration
&clientConfig) {
   const Aws::Glue::GlueClient client(clientConfig);
   bool result = true;

   // 13. Delete a job.
   if (!job.empty()) {
       Aws::Glue::Model::DeleteJobRequest request;
       request.SetJobName(job);

       Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


       if (outcome.IsSuccess()) {
           std::cout << "Successfully deleted the job." << std::endl;
       }
       else {
           std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           result = false;
       }
   }

   // 14. Delete a database.
   if (!database.empty()) {
       Aws::Glue::Model::DeleteDatabaseRequest request;
       request.SetName(database);

       Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
               request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully deleted the database." << std::endl;
       }
       else {
           std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           result = false;
       }
   }
```

```
    // 15. Delete a crawler.
    if (!crawler.empty()) {
        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
 client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    // 16. Delete the job script and run data from the S3 bucket.
    result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
                                                       clientConfig);
    return result;
}

//! Routine which uploads a file to an S3 bucket.
/*!
 \\sa uploadFile()
 \param bucketName: An S3 bucket created in the setup.
 \param filePath: The path of the file to upload.
 \param fileName The name for the uploaded file.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                         const Aws::String &filePath,
                         const Aws::String &fileName,
                         const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);
```

```
    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          filePath.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << filePath << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Added object '" << filePath << "' to bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
 \\sa deleteAllObjectsInS3Bucket()
 \param bucketName: The S3 bucket name.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                              const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsRequest listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);
```

```
   Aws::S3::Model::ListObjectsOutcome listObjectsOutcome = client.ListObjects(
           listObjectsRequest);

   bool result = false;
   if (listObjectsOutcome.IsSuccess()) {
       const std::vector<Aws::S3::Model::Object> &objects =
listObjectsOutcome.GetResult().GetContents();
       if (!objects.empty()) {
           Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
           deleteObjectsRequest.SetBucket(bucketName);

           std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
           for (const Aws::S3::Model::Object &object: objects) {
               objectIdentifiers.push_back(

Aws::S3::Model::ObjectIdentifier().WithKey(object.GetKey()));
           }
           Aws::S3::Model::Delete objectsDelete;
           objectsDelete.SetObjects(objectIdentifiers);
           objectsDelete.SetQuiet(true);
           deleteObjectsRequest.SetDelete(objectsDelete);

           Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                   client.DeleteObjects(deleteObjectsRequest);

           if (!deleteObjectsOutcome.IsSuccess()) {
               std::cerr << "Error deleting objects. " <<
                         deleteObjectsOutcome.GetError().GetMessage() <<
std::endl;
           }
           else {
               std::cout << "Successfully deleted the objects." << std::endl;
               result = true;
           }
       }
       else {
           std::cout << "No objects to delete in '" << bucketName << "'." <<
std::endl;
       }
   }
   else {
       std::cerr << "Error listing objects. "
                 << listObjectsOutcome.GetError().GetMessage() << std::endl;
   }
```

```
    return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
 \\sa getObjectFromBucket()
 \param bucketName: The S3 bucket name.
 \param objectKey: The object's name.
 \param objectStream: A stream to receive the retrieved data.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);


    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." <<
 std::endl;
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
    }
    else {
        std::cerr << "Error retrieving object. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for C++ .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

                Where:
                    iam - The ARN of the IAM role that has AWS Glue and S3
 permissions.\s
                    s3Path - The Amazon Simple Storage Service (Amazon S3) target
 that contains data (for example, CSV data).
                    cron - A cron expression used to specify the schedule  (i.e.,
 cron(15 12 * * ? *).
                    dbName - The database name.\s
```

```
                    crawlerName - The name of the crawler.\s
                    jobName - The name you assign to this job definition.
                    scriptLocation - The Amazon S3 path to a script that runs a
job.
                    locationUri - The location of the database
                    bucketNameSc - The Amazon S3 bucket name used when creating a
job
                    """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        createDatabase(glueClient, dbName, locationUri);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
```

```
        getSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        startSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        getSpecificDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName = getGlueTables(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        createJob(glueClient, jobName, iam, scriptLocation);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        getAllJobs(glueClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        getJobRuns(glueClient, jobName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        deleteJob(glueClient, jobName);
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
```

```
        TimeUnit.MINUTES.sleep(5);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        deleteDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName,
String locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                    .description("Built with the AWS SDK for Java V2")
                    .name(dbName)
                    .locationUri(locationUri)
                    .build();

            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                    .databaseInput(input)
                    .build();

            glueClient.createDatabase(request);
            System.out.println(dbName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createGlueCrawler(GlueClient glueClient,
            String iam,
            String s3Path,
            String cron,
```

```
            String dbName,
            String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                    .path(s3Path)
                    .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                    .s3Targets(targetList)
                    .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                    .databaseName(dbName)
                    .name(crawlerName)
                    .description("Created by the AWS Glue Java API")
                    .targets(targets)
                    .role(iam)
                    .schedule(cron)
                    .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response =
glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
```

```
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
```

```
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " +
createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getGlueTables(GlueClient glueClient, String dbName) {
        String myTableName = "";
        try {
            GetTablesRequest tableRequest = GetTablesRequest.builder()
                    .databaseName(dbName)
                    .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return myTableName;
    }

    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
            String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
```

```
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                    .workerType(WorkerType.G_1_X)
                    .numberOfWorkers(10)
                    .arguments(myMap)
                    .jobName(jobName)
                    .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createJob(GlueClient glueClient, String jobName, String
iam, String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                    .pythonVersion("3")
                    .name("glueetl")
                    .scriptLocation(scriptLocation)
                    .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .description("A Job created by using the AWS SDK for Java
V2")
                    .glueVersion("2.0")
                    .workerType(WorkerType.G_1_X)
                    .numberOfWorkers(10)
                    .name(jobName)
                    .role(iam)
                    .command(command)
                    .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                    .maxResults(10)
                    .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                    .jobName(jobName)
                    .maxResults(20)
                    .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
```

```
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                    .jobName(jobName)
                    .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabase(GlueClient glueClient, String databaseName)
{
```

```
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
 DeleteCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable

- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Creare e avviare un crawler in grado di eseguire il crawling di un bucket pubblico di Amazon Simple Storage Service (Amazon S3) generando un database di metadati che descrive i dati rilevati in formato CSV.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });
```

```
  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const crawlerExists = async ({ getCrawler }, crawlerName) => {
  try {
    await getCrawler(crawlerName);
    return true;
  } catch {
    return false;
  }
};

const makeCreateCrawlerStep = (actions) => async (context) => {
  if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
    log("Crawler already exists. Skipping creation.");
  } else {
    await actions.createCrawler(
      process.env.CRAWLER_NAME,
      process.env.ROLE_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_PREFIX,
      process.env.S3_TARGET_PATH
    );
```

```
    log("Crawler created successfully.", { type: "success" });
  }

  return { ...context };
};

/**
 * @param {(name: string) => Promise<import('@aws-sdk/client-
glue').GetCrawlerCommandOutput>} getCrawler
 * @param {string} crawlerName
 */
const waitForCrawler = async (getCrawler, crawlerName) => {
  const waitTimeInSeconds = 30;
  const { Crawler } = await getCrawler(crawlerName);

  if (!Crawler) {
    throw new Error(`Crawler with name ${crawlerName} not found.`);
  }

  if (Crawler.State === "READY") {
    return;
  }

  log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
  await wait(waitTimeInSeconds);
  return waitForCrawler(getCrawler, crawlerName);
};

const makeStartCrawlerStep =
  ({ startCrawler, getCrawler }) =>
  async (context) => {
    log("Starting crawler.");
    await startCrawler(process.env.CRAWLER_NAME);
    log("Crawler started.", { type: "success" });

    log("Waiting for crawler to finish running. This can take a while.");
    await waitForCrawler(getCrawler, process.env.CRAWLER_NAME);
    log("Crawler ready.", { type: "success" });

    return { ...context };
  };
```

Elenca le informazioni su database e tabelle nel tuo AWS Glue Data Catalog.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};

const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
  };

const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
  };
```

Creare e avviare un processo che estrae i dati CSV dal bucket Amazon S3 di origine, li trasforma rimuovendo e rinominando i campi e carica l'output in formato JSON in un altro bucket Amazon S3.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
```

```
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY,
    );
    log("Job created.", { type: "success" });

    return { ...context };
  };

/**
 * @param {(name: string, runId: string) => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput> }  getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "TIMEOUT":
    case "STOPPED":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`,
      );
    case "RUNNING":
      break;
    case "SUCCEEDED":
      return;
    default:
      throw new Error(`Unknown job run state: ${JobRun.JobRunState}`);
  }

  log(
    `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...`,
  );
  await wait(waitTimeInSeconds);
  return waitForJobRun(getJobRun, jobName, jobRunId);
};
```

```
/**
 * @param {{ prompter: { prompt: () => Promise<{ shouldOpen: boolean }>} }}
 context
 */
const promptToOpen = async (context) => {
  const { shouldOpen } = await context.prompter.prompt({
    name: "shouldOpen",
    type: "confirm",
    message: "Open the output bucket in your browser?",
  });

  if (shouldOpen) {
    return open(
      `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME} to
 view the output.`,
    );
  }
};

const makeStartJobRunStep =
  ({ startJobRun, getJobRun }) =>
  async (context) => {
    log("Starting job.");
    const { JobRunId } = await startJobRun(
      process.env.JOB_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_NAME,
      process.env.BUCKET_NAME,
    );
    log("Job started.", { type: "success" });

    log("Waiting for job to finish running. This can take a while.");
    await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
    log("Job run succeeded.", { type: "success" });

    await promptToOpen(context);

    return { ...context };
  };
```

Elencare le informazioni sulle esecuzioni dei processi e visualizzare alcuni dei dati trasformati.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};

const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
  const { JobRun } = await getJobRun(jobName, jobRunId);
  log(JobRun, { type: "object" });
};

const makePickJobRunStep =
  ({ getJobRuns, getJobRun }) =>
  async (context) => {
    if (context.selectedJobName) {
      const { JobRuns } = await getJobRuns(context.selectedJobName);

      const { jobRunId } = await context.prompter.prompt({
        name: "jobRunId",
        type: "list",
        message: "Select a job run to see details.",
        choices: JobRuns.map((run) => run.Id),
      });

      logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
    }

    return { ...context };
  };
```

Eliminare tutte le risorse create dalla demo.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};

const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};

const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};

const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
```

```
  const { selectedJobNames } = await context.prompter.prompt({
    name: "selectedJobNames",
    type: "checkbox",
    message: "Let's clean up jobs. Select jobs to delete.",
    choices: jobNames,
  });

  if (selectedJobNames.length === 0) {
    log("No jobs selected.");
  } else {
    log("Deleting jobs.");
    await Promise.all(
      selectedJobNames.map((n) => deleteJobFn(n).catch(console.error))
    );
    log("Jobs deleted.", { type: "success" });
  }
};

const makeCleanUpJobsStep =
  ({ listJobs, deleteJob }) =>
  async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
      await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
  };

const deleteTables = (deleteTable, databaseName, tableNames) =>
  Promise.all(
    tableNames.map((tableName) =>
      deleteTable(databaseName, tableName).catch(console.error)
    )
  );

const makeCleanUpTablesStep =
  ({ getTables, deleteTable }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
      () => ({ TableList: null })
    );

    if (TableList && TableList.length > 0) {
```

```
      const { tableNames } = await context.prompter.prompt({
        name: "tableNames",
        type: "checkbox",
        message: "Let's clean up tables. Select tables to delete.",
        choices: TableList.map((t) => t.Name),
      });

      if (tableNames.length === 0) {
        log("No tables selected.");
      } else {
        log("Deleting tables.");
        await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
        log("Tables deleted.", { type: "success" });
      }
    }

    return { ...context };
  };

const deleteDatabases = (deleteDatabase, databaseNames) =>
  Promise.all(
    databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error))
  );

const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
        message: "Let's clean up databases. Select databases to delete.",
        choices: DatabaseList.map((db) => db.Name),
      });

      if (dbNames.length === 0) {
        log("No databases selected.");
      } else {
        log("Deleting databases.");
        await deleteDatabases(deleteDatabase, dbNames);
        log("Databases deleted.", { type: "success" });
      }
```

```
    }

    return { ...context };
  };

const cleanUpCrawlerStep = async (context) => {
  log(`Deleting crawler.`);

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log(`Crawler is already deleted.`);
    } else {
      throw err;
    }
  }

  return { ...context };
};
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for JavaScript .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables

- ListJobs
- StartCrawler
- StartJobRun

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service
 (Amazon S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }
```

```
    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(dbName: String?, locationUriVal: String?) {

    val input = DatabaseInput {
        description = "Built with the AWS SDK for Kotlin"
        name = dbName
        locationUri = locationUriVal
    }

    val request = CreateDatabaseRequest {
        databaseInput = input
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
```

```
}

suspend fun createCrawler(iam: String?, s3Path: String?, cron: String?, dbName:
 String?, crawlerName: String) {

    val s3Target = S3Target {
        path = s3Path
    }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb = CrawlerTargets {
        s3Targets = targetList
    }

    val crawlerRequest = CreateCrawlerRequest {
        databaseName = dbName
        name = crawlerName
        description = "Created by the AWS Glue Java API"
        targets = targetOb
        role = iam
        schedule = cron
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {

    val request = GetCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

```
suspend fun startCrawler(crawlerName: String) {

    val crawlerRequest = StartCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {

    val request = GetDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {

    val tableRequest = GetTablesRequest {
        databaseName = dbName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {

    val runRequest = StartJobRunRequest {
        workerType = WorkerType.G1X
        numberOfWorkers = 10
```

```
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(jobName: String, iam: String?, scriptLocationVal: String?)
 {

    val commandOb = JobCommand {
        pythonVersion = "3"
        name = "MyJob1"
        scriptLocation = scriptLocationVal
    }

    val jobRequest = CreateJobRequest {
        description = "A Job created by using the AWS SDK for Java V2"
        glueVersion = "2.0"
        workerType = WorkerType.G1X
        numberOfWorkers = 10
        name = jobName
        role = iam
        command = commandOb
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {

    val request = GetJobsRequest {
        maxResults = 10
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
```

```
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {

    val request = GetJobRunsRequest {
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {

    val jobRequest = DeleteJobRequest {
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {

    val request = DeleteDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
```

```
    val request = DeleteCrawlerRequest {
        name = crawlerName
    }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Kotlin.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

PHP

SDK per PHP

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;
```

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);
```

```
        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);
```

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
```

```
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path):
Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
```

```
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl): Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
```

```
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

```
    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for PHP .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns

- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Python

SDK per Python (Boto3)

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea una classe che racchiuda le AWS Glue funzioni utilizzate nello scenario.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
```

```
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created
by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

```
    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
that can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
```

```
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_job_run(self, name, input_database, input_table,
output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
the
            # Python ETL script to determine the location of input and output
data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
```

```
                "--input_table": input_table,
                "--output_bucket_url": f"s3://{output_bucket_name}/",
            },
        )
    except ClientError as err:
        logger.error(
            "Couldn't start job run %s. Here's why: %s: %s",
            name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return response["JobRunId"]


def list_jobs(self):
    """
    Lists the names of job definitions in your account.

    :return: The list of job definition names.
    """
    try:
        response = self.glue_client.list_jobs()
    except ClientError as err:
        logger.error(
            "Couldn't list jobs. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return response["JobNames"]


def get_job_runs(self, job_name):
    """
    Gets information about runs that have been performed for a specific job
    definition.

    :param job_name: The name of the job definition to look up.
    :return: The list of job runs.
    """
    try:
```

```
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
```

```
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Creazione di una classe che esegue lo scenario.

```
class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job
 and use
    them to transform data from CSV to JSON format.
    """

    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
        :param glue_service_role: An AWS Identity and Access Management (IAM)
 role
                                  that AWS Glue can assume to gain access to the
                                  resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output
 data
```

```
                            from AWS Glue job runs.
        """
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
    def wait(seconds, tick=12):
        """
        Waits for a specified number of seconds, while also displaying an
animated
        spinner.

        :param seconds: The number of seconds to wait.
        :param tick: The number of frames per second used to animate the spinner.
        """
        progress = "|/-\\"
        waited = 0
        while waited < seconds:
            for frame in range(tick):
                sys.stdout.write(f"\r{progress[frame % len(progress)]}")
                sys.stdout.flush()
                time.sleep(1 / tick)
            waited += 1

    def upload_job_script(self, job_script):
        """
        Uploads a Python ETL script to an S3 bucket. The script is used by the
AWS Glue
        job to transform data.

        :param job_script: The relative path to the job script.
        """
        try:
            self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
            print(f"Uploaded job script '{job_script}' to the example bucket.")
        except S3UploadFailedError as err:
            logger.error("Couldn't upload job script. Here's why: %s", err)
            raise

    def run(self, crawler_name, db_name, db_prefix, data_source, job_script,
job_name):
        """
```

```
        Runs the scenario. This is an interactive experience that runs at a
command
        prompt and asks you for input throughout.

        :param crawler_name: The name of the crawler used in the scenario. If the
                             crawler does not exist, it is created.
        :param db_name: The name to give the metadata database created by the
crawler.
        :param db_prefix: The prefix to give tables added to the database by the
                          crawler.
        :param data_source: The location of the data source that is targeted by
the
                            crawler and extracted during job runs.
        :param job_script: The job script that is used to transform data during
job
                           runs.
        :param job_name: The name to give the job definition that is created
during the
                         scenario.
        """
        wrapper = GlueWrapper(self.glue_client)
        print(f"Checking for crawler {crawler_name}.")
        crawler = wrapper.get_crawler(crawler_name)
        if crawler is None:
            print(f"Creating crawler {crawler_name}.")
            wrapper.create_crawler(
                crawler_name,
                self.glue_service_role.arn,
                db_name,
                db_prefix,
                data_source,
            )
            print(f"Created crawler {crawler_name}.")
            crawler = wrapper.get_crawler(crawler_name)
        pprint(crawler)
        print("-" * 88)

        print(
            f"When you run the crawler, it crawls data stored in {data_source}
and "
            f"creates a metadata database in the AWS Glue Data Catalog that
describes "
            f"the data in the data source."
        )
```

```
        print("In this example, the source data is in CSV format.")
        ready = False
        while not ready:
            ready = Question.ask_question(
                "Ready to start the crawler? (y/n) ", Question.is_yesno
            )
        wrapper.start_crawler(crawler_name)
        print("Let's wait for the crawler to run. This typically takes a few
minutes.")
        crawler_state = None
        while crawler_state != "READY":
            self.wait(10)
            crawler = wrapper.get_crawler(crawler_name)
            crawler_state = crawler["State"]
            print(f"Crawler is {crawler['State']}.")
        print("-" * 88)

        database = wrapper.get_database(db_name)
        print(f"The crawler created database {db_name}:")
        pprint(database)
        print(f"The database contains these tables:")
        tables = wrapper.get_tables(db_name)
        for index, table in enumerate(tables):
            print(f"\t{index + 1}. {table['Name']}")
        table_index = Question.ask_question(
            f"Enter the number of a table to see more detail: ",
            Question.is_int,
            Question.in_range(1, len(tables)),
        )
        pprint(tables[table_index - 1])
        print("-" * 88)

        print(f"Creating job definition {job_name}.")
        wrapper.create_job(
            job_name,
            "Getting started example job.",
            self.glue_service_role.arn,
            f"s3://{self.glue_bucket.name}/{job_script}",
        )
        print("Created job definition.")
        print(
            f"When you run the job, it extracts data from {data_source},
transforms it "
            f"by using the {job_script} script, and loads the output into "
```

```
            f"S3 bucket {self.glue_bucket.name}."
        )
        print(
            "In this example, the data is transformed from CSV to JSON, and only
a few "
            "fields are included in the output."
        )
        job_run_status = None
        if Question.ask_question(f"Ready to run? (y/n) ", Question.is_yesno):
            job_run_id = wrapper.start_job_run(
                job_name, db_name, tables[0]["Name"], self.glue_bucket.name
            )
            print(f"Job {job_name} started. Let's wait for it to run.")
            while job_run_status not in ["SUCCEEDED", "STOPPED", "FAILED",
"TIMEOUT"]:
                self.wait(10)
                job_run = wrapper.get_job_run(job_name, job_run_id)
                job_run_status = job_run["JobRunState"]
                print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
        print("-" * 88)

        if job_run_status == "SUCCEEDED":
            print(
                f"Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':"
            )
            try:
                keys = [
                    obj.key for obj in
self.glue_bucket.objects.filter(Prefix="run-")
                ]
                for index, key in enumerate(keys):
                    print(f"\t{index + 1}: {key}")
                lines = 4
                key_index = Question.ask_question(
                    f"Enter the number of a block to download it and see the
first {lines} "
                    f"lines of JSON output in the block: ",
                    Question.is_int,
                    Question.in_range(1, len(keys)),
                )
                job_data = io.BytesIO()
                self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)
                job_data.seek(0)
```

```
                for _ in range(lines):
                    print(job_data.readline().decode("utf-8"))
            except ClientError as err:
                logger.error(
                    "Couldn't get job run data. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
            print("-" * 88)

        job_names = wrapper.list_jobs()
        if job_names:
            print(f"Your account has {len(job_names)} jobs defined:")
            for index, job_name in enumerate(job_names):
                print(f"\t{index + 1}. {job_name}")
            job_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_names)} to see the list
of runs for "
                f"a job: ",
                Question.is_int,
                Question.in_range(1, len(job_names)),
            )
            job_runs = wrapper.get_job_runs(job_names[job_index - 1])
            if job_runs:
                print(f"Found {len(job_runs)} runs for job {job_names[job_index -
1]}:")
                for index, job_run in enumerate(job_runs):
                    print(
                        f"\t{index + 1}. {job_run['JobRunState']} on "
                        f"{job_run['CompletedOn']:%Y-%m-%d %H:%M:%S}"
                    )
                run_index = Question.ask_question(
                    f"Enter a number between 1 and {len(job_runs)} to see details
for a run: ",
                    Question.is_int,
                    Question.in_range(1, len(job_runs)),
                )
                pprint(job_runs[run_index - 1])
            else:
                print(f"No runs found for job {job_names[job_index - 1]}")
        else:
            print("Your account doesn't have any jobs defined.")
        print("-" * 88)
```

```
        print(
            f"Let's clean up. During this example we created job definition
 '{job_name}'."
        )
        if Question.ask_question(
            "Do you want to delete the definition and all runs? (y/n) ",
            Question.is_yesno,
        ):
            wrapper.delete_job(job_name)
            print(f"Job definition '{job_name}' deleted.")
        tables = wrapper.get_tables(db_name)
        print(f"We also created database '{db_name}' that contains these
 tables:")
        for table in tables:
            print(f"\t{table['Name']}")
        if Question.ask_question(
            "Do you want to delete the tables and the database? (y/n) ",
            Question.is_yesno,
        ):
            for table in tables:
                wrapper.delete_table(db_name, table["Name"])
                print(f"Deleted table {table['Name']}.")
            wrapper.delete_database(db_name)
            print(f"Deleted database {db_name}.")
        print(f"We also created crawler '{crawler_name}'.")
        if Question.ask_question(
            "Do you want to delete the crawler? (y/n) ", Question.is_yesno
        ):
            wrapper.delete_crawler(crawler_name)
            print(f"Deleted crawler {crawler_name}.")
        print("-" * 88)


def parse_args(args):
    """
    Parse command line arguments.

    :param args: The command line arguments.
    :return: The parsed arguments.
    """
    parser = argparse.ArgumentParser(
        description="Runs the AWS Glue getting started with crawlers and jobs
 scenario. "
```

```
        "Before you run this scenario, set up scaffold resources by running "
        "'python scaffold.py deploy'."
    )
    parser.add_argument(
        "role_name",
        help="The name of an IAM role that AWS Glue can assume. This role must
 grant access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole "
        "managed policy.",
    )
    parser.add_argument(
        "bucket_name",
        help="The name of an S3 bucket that AWS Glue can access to get the job
 script and "
        "put job results.",
    )
    parser.add_argument(
        "--job_script",
        default="flight_etl_job_script.py",
        help="The name of the job script file that is used in the scenario.",
    )
    return parser.parse_args(args)


def main():
    args = parse_args(sys.argv[1:])
    try:
        print("-" * 88)
        print(
            "Welcome to the AWS Glue getting started with crawlers and jobs
 scenario."
        )
        print("-" * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client("glue"),
            boto3.resource("iam").Role(args.role_name),
            boto3.resource("s3").Bucket(args.bucket_name),
        )
        scenario.upload_job_script(args.job_script)
        scenario.run(
            "doc-example-crawler",
            "doc-example-database",
            "doc-example-",
            "s3://crawler-public-us-east-1/flight/2016/csv",
```

```
            args.job_script,
            "doc-example-job",
        )
        print("-" * 88)
        print(
            "To destroy scaffold resources, including the IAM role and S3 bucket
 "
            "used in this scenario, run 'python scaffold.py destroy'."
        )
        print("\nThanks for watching!")
        print("-" * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")
```

Crea uno script ETL utilizzato da AWS Glue per estrarre, trasformare e caricare i dati durante l'esecuzione dei processi.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
```

```
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
```

```
)

job.commit()
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Python (Boto3).
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea una classe che racchiuda le AWS Glue funzioni utilizzate nello scenario.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
 if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
 the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
```

```
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
rescue Aws::Glue::Errors::GlueException => e
```

```
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
    end

    # Retrieves a list of tables in the specified database.
    #
    # @param db_name [String] The name of the database to retrieve tables from.
    # @return [Array<Aws::Glue::Types::Table>]
    def get_tables(db_name)
      response = @glue_client.get_tables(database_name: db_name)
      response.table_list
    rescue Aws::Glue::Errors::GlueException => e
      @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
      raise
    end

    # Creates a new job with the specified configuration.
    #
    # @param name [String] The name of the job.
    # @param description [String] The description of the job.
    # @param role_arn [String] The ARN of the IAM role to be used by the job.
    # @param script_location [String] The location of the ETL script for the job.
    # @return [void]
    def create_job(name, description, role_arn, script_location)
      @glue_client.create_job(
        name: name,
        description: description,
        role: role_arn,
        command: {
          name: "glueetl",
          script_location: script_location,
          python_version: "3"
        },
        glue_version: "3.0"
      )
    rescue Aws::Glue::Errors::GlueException => e
      @logger.error("Glue could not create job #{name}: \n#{e.message}")
      raise
    end

    # Starts a job run for the specified job.
    #
    # @param name [String] The name of the job to start the run for.
    # @param input_database [String] The name of the input database for the job.
```

```
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
```

```
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
file to.
  # @return [void]
```

```
  def upload_job_script(file_path, bucket_resource)
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
       body: file,
       bucket: bucket_resource.name,
       key: file_path
     })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end

end
```

Creazione di una classe che esegue lo scenario.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
  end

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)

    new_step(1, "Create a crawler")
    puts "Checking for crawler #{crawler_name}."
    crawler = wrapper.get_crawler(crawler_name)
    if crawler == false
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
 db_prefix, data_source)
      puts "Successfully created #{crawler_name}:"
      crawler = wrapper.get_crawler(crawler_name)
      puts JSON.pretty_generate(crawler).yellow
    end
    print "\nDone!\n".green

    new_step(2, "Run a crawler to output a database.")
```

```
    puts "Location of input data analyzed by crawler: #{data_source}"
    puts "Outputs: a Data Catalog database in CSV format containing metadata on
input."
    wrapper.start_crawler(crawler_name)
    puts "Starting crawler... (this typically takes a few minutes)"
    crawler_state = nil
    while crawler_state != "READY"
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]["state"]
      print "Status check: #{crawler_state}.".yellow
    end
    print "\nDone!\n".green

    new_step(3, "Query the database.")
    database = wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    print "#{database}".yellow
    puts "\nThe database contains these tables:"
    tables = wrapper.get_tables(db_name)
    tables.each_with_index do |table, index|
      print "\t#{index + 1}. #{table['name']}".yellow
    end
    print "\nDone!\n".green

    new_step(4, "Create a job definition that runs an ETL script.")
    puts "Uploading Python ETL script to S3..."
    wrapper.upload_job_script(job_script, @glue_bucket)
    puts "Creating job definition #{job_name}:\n"
    response = wrapper.create_job(job_name, "Getting started example job.",
@glue_service_role.arn, "s3://#{@glue_bucket.name}/#{job_script}")
    puts JSON.pretty_generate(response).yellow
    print "\nDone!\n".green

    new_step(5, "Start a new job")
    job_run_status = nil
    job_run_id = wrapper.start_job_run(
      job_name,
      db_name,
      tables[0]["name"],
      @glue_bucket.name
    )
    puts "Job #{job_name} started. Let's wait for it to run."
    until ["SUCCEEDED", "STOPPED", "FAILED", "TIMEOUT"].include?(job_run_status)
```

```
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]["job_run_state"]
      print "Status check: #{job_name}/#{job_run_id} - #{job_run_status}.".yellow
    end
    print "\nDone!\n".green

    new_step(6, "View results from a successful job run.")
    if job_run_status == "SUCCEEDED"
      puts "Data from your job run is stored in your S3 bucket
'#{@glue_bucket.name}'. Files include:"
      begin

        # Print the key name of each object in the bucket.
        @glue_bucket.objects.each do |object_summary|
          if object_summary.key.include?("run-")
            print "#{object_summary.key}".yellow
          end
        end

        # Print the first 256 bytes of a run file
        desired_sample_objects = 1
        @glue_bucket.objects.each do |object_summary|
          if object_summary.key.include?("run-")
            if desired_sample_objects > 0
              sample_object = @glue_bucket.object(object_summary.key)
              sample = sample_object.get(range: "bytes=0-255").body.read
              puts "\nSample run file contents:"
              print "#{sample}".yellow
              desired_sample_objects -= 1
            end
          end
        end
      rescue Aws::S3::Errors::ServiceError => e
        logger.error(
          "Couldn't get job run data. Here's why: %s: %s",
          e.response.error.code, e.response.error.message
        )
        raise
      end
    end
    print "\nDone!\n".green

    new_step(7, "Delete job definition and crawler.")
```

```
    wrapper.delete_job(job_name)
    puts "Job deleted: #{job_name}."
    wrapper.delete_crawler(crawler_name)
    puts "Crawler deleted: #{crawler_name}."
    wrapper.delete_table(db_name, tables[0]["name"])
    puts "Table deleted: #{tables[0]["name"]} in #{db_name}."
    wrapper.delete_database(db_name)
    puts "Database deleted: #{db_name}."
    print "\nDone!\n".green
  end
end

def main

  banner("../../helpers/banner.txt")
  puts
 "#########################################################################################
  puts "#
                            #".yellow
  puts "#                                          EXAMPLE CODE DEMO:
                            #".yellow
  puts "#                                              AWS Glue
                            #".yellow
  puts "#
                            #".yellow
  puts
 "#########################################################################################
  puts ""
  puts "You have launched a demo of AWS Glue using the AWS for Ruby v3 SDK. Over
 the next 60 seconds, it will"
  puts "do the following:"
  puts "    1. Create a crawler."
  puts "    2. Run a crawler to output a database."
  puts "    3. Query the database."
  puts "    4. Create a job definition that runs an ETL script."
  puts "    5. Start a new job."
  puts "    6. View results from a successful job run."
  puts "    7. Delete job definition and crawler."
  puts ""

  confirm_begin
  billing
  security
  puts "\e[H\e[2J"
```

```
  # Set input file names
  job_script_filepath = "job_script.py"
  resource_names = YAML.load_file("resource_names.yaml")

  # Instantiate existing IAM role.
  iam = Aws::IAM::Resource.new(region: "us-east-1")
  iam_role_name = resource_names["glue_service_role"]
  iam_role = iam.role(iam_role_name)

  # Instantiate existing S3 bucket.
  s3 = Aws::S3::Resource.new(region: "us-east-1")
  s3_bucket_name = resource_names["glue_bucket"]
  s3_bucket = s3.bucket(s3_bucket_name)

  scenario = GlueCrawlerJobScenario.new(
    Aws::Glue::Client.new(region: "us-east-1"),
    iam_role,
    s3_bucket,
    @logger
  )

  random_int = rand(10 ** 4)
  scenario.run(
    "doc-example-crawler-#{random_int}",
    "doc-example-database-#{random_int}",
    "doc-example-#{random_int}-",
    "s3://crawler-public-us-east-1/flight/2016/csv",
    job_script_filepath,
    "doc-example-job-#{random_int}"
  )

  puts "-" * 88
  puts "You have reached the end of this tour of AWS Glue."
  puts "To destroy CDK-created resources, run:\n       cdk destroy"
  puts "-" * 88

end
```

Crea uno script ETL utilizzato da AWS Glue per estrarre, trasformare e caricare i dati durante l'esecuzione dei processi.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
```

```
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Ruby .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase

- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Creare e avviare un crawler in grado di eseguire il crawling di un bucket pubblico di Amazon Simple Storage Service (Amazon S3) generando un database di metadati che descrive i dati rilevati in formato CSV.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
```

```
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;

        let start_crawler =
glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

Elenca le informazioni su database e tabelle nel tuo AWS Glue Data Catalog.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;

        let tables = glue
            .get_tables()
```

```
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

Creare e avviare un processo che estrae i dati CSV dal bucket Amazon S3 di origine, li trasforma rimuovendo e rinominando i campi e carica l'output in formato JSON in un altro bucket Amazon S3.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;

        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .get(0)
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
```

```
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

Eliminare tutte le risorse create dalla demo.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }

        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Per informazioni dettagliate sulle API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Rust.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Sicurezza in AWS Glue

Per AWS, la sicurezza del cloud ha la massima priorità. In quanto cliente AWS, è possibile trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle organizzazioni più esigenti a livello di sicurezza.

La sicurezza è una responsabilità condivisa tra te e AWS. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- La sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che esegue AWS i servizi nel cloud AWS. AWS fornisce, inoltre, servizi utilizzabili in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformità AWS. Per ulteriori informazioni sui programmi di conformità che si applicano a AWS Glue, consulta Servizi coperti dal programma di conformitàAWS.
- Sicurezza nel cloud: la tua responsabilità è determinata dal servizio AWS che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda, le leggi e le normative applicabili.

Questa documentazione facilitaconsenteladicomprensionecomprenderedell'applicazionecomedelapplicare il modello di responsabilità condivisa quando utilizzisi usaAWS Glue. I seguenti argomenti illustrano come configurare AWS Glue per soddisfare gli obiettivi di sicurezza e conformità. Scoprirai anche come utilizzare altri servizi di AWS per monitorare e proteggere le risorse AWS Glue.

Argomenti

- Protezione dei dati in AWS Glue
- Gestione delle identità e degli accessi per AWS Glue
- Registrazione e monitoraggio in AWS Glue
- Convalida della conformità per AWS Glue
- Resilienza in AWS Glue
- Sicurezza dell'infrastruttura in AWS Glue

Protezione dei dati in AWS Glue

AWS Glue offre diverse caratteristiche che permettono di proteggere i dati.

Argomenti

Crittografia dei dati inattivi

AWS Glue supporta la crittografia dei dati inattivi per Creazione di processi ETL visivi con AWS Glue Studio e Utilizzo di endpoint per lo sviluppo di script. È possibile configurare i processi di estrazione, trasformazione e caricamento (ETL) e gli endpoint di sviluppo per usare AWS Key Management Service (AWS KMS) durante la scrittura dei dati inattivi criptati. È inoltre possibile crittografare i metadati memorizzati in AWS Glue Data Catalog con chiavi gestite con AWS KMS. Infine, è possibile utilizzare le chiavi di AWS KMS per crittografare i riferimenti ai processi e i log generati da crawler e processi ETL.

Oltre ai dati scritti su Amazon Simple Storage Service (Amazon S3) e Amazon Logs, puoi crittografare gli oggetti di metadati AWS Glue Data Catalog presenti nel tuo computer oltre ai dati scritti in Amazon Simple Storage Service (Amazon S3) e CloudWatch Amazon Logs per job, crawler ed endpoint di sviluppo. Quando si creano processi, crawler ed endpoint di sviluppo in AWS Glue, è possibile fornire impostazioni di crittografia collegando una configurazione di protezione. Le configurazioni di sicurezza contengono chiavi di crittografia lato server gestite da Amazon S3 (SSE-S3) o chiavi master cliente (CMK) memorizzate in AWS KMS (SSE-KMS). È possibile creare configurazioni di sicurezza utilizzando la console AWS Glue.

Puoi attivare la crittografia dell'intero catalogo dati nel tuo account. Puoi farlo specificando CMK memorizzati in AWS KMS.

> **Important**
>
> AWS Gluesupporta solo chiavi simmetriche gestite dal cliente. Per ulteriori informazioni, consulta Customer Managed Keys (CMK) nella Guida per gli AWS Key Management Service sviluppatori.

Con la crittografia attivata, quando aggiungi oggetti del catalogo dati, esegui crawler o processi o avvii endpoint di sviluppo, vengono usate chiavi SSE-S3 o SSE-KMS per scrivere i dati a riposo. Puoi anche configurare AWS Glue per accedere ai datastore Java Database Connectivity (JDBC) solo attraverso un protocollo Transport Layer Security (TLS) attendibile.

In AWS Glue è possibile controllare le impostazioni di crittografia nelle posizioni seguenti:

- Le impostazioni del catalogo dati.
- Configurazioni della sicurezza create.
- Impostazione di crittografia lato server (SSE-S3 o SSE-KMS) passata come parametro al processo ETL (estrazione, trasformazione e caricamento) di AWS Glue.

Per ulteriori informazioni su come configurare le crittografia, consulta Configurazione della crittografia in AWS Glue.

Argomenti

- Crittografia del catalogo dati
- Crittografia delle password di connessione
- Crittografia dei dati scritti da AWS Glue

Crittografia del catalogo dati

AWS Glue Data Catalogla crittografia offre una maggiore sicurezza per i dati sensibili. AWS Gluesi integra con AWS Key Management Service (AWS KMS) per crittografare i metadati archiviati nel Data Catalog. È possibile abilitare o disabilitare le impostazioni di crittografia per le risorse nel Data Catalog utilizzando la AWS Glue console o il. AWS CLI

Quando abiliti la crittografia per il tuo Data Catalog, tutti i nuovi oggetti che crei verranno crittografati. Quando disabiliti la crittografia, i nuovi oggetti che crei non verranno crittografati, ma gli oggetti crittografati esistenti rimarranno crittografati.

È possibile crittografare l'intero catalogo dati utilizzando chiavi di crittografia AWS gestite o chiavi di crittografia gestite dal cliente. Per ulteriori informazioni sui tipi e sugli stati delle chiavi, consulta AWS Key Management Servicei concetti nella Guida per gli AWS Key Management Service sviluppatori.

Chiavi gestite da AWS

AWSIe chiavi gestite sono chiavi KMS presenti nel tuo account che vengono create, gestite e utilizzate per tuo conto da un AWS servizio integrato con. AWS KMS Puoi visualizzare le chiavi AWS gestite nel tuo account, visualizzare le relative politiche chiave e verificarne l'utilizzo nei AWS CloudTrail log. Tuttavia, non puoi gestire queste chiavi o modificarne le autorizzazioni.

Encryption at rest si integra automaticamente con AWS KMS la gestione delle chiavi AWS gestite AWS Glue utilizzate per crittografare i metadati. Se non esiste una chiave AWS gestita quando abiliti la crittografia dei metadati, crea AWS KMS automaticamente una nuova chiave per te.

Per ulteriori informazioni, consulta chiavi gestite da AWS .

Chiavi gestite dal cliente

Le chiavi gestite dal cliente sono chiavi KMS nell'Account AWS create, possedute e gestite dall'utente. Hai il pieno controllo su queste chiavi KMS. È possibile:

- Stabilisci e mantieni le loro politiche chiave, le politiche IAM e le sovvenzioni
- Abilitali e disabilitali
- Ruota il loro materiale crittografico
- Aggiunta di tag
- Crea alias che si riferiscono ad essi
- Pianificali per l'eliminazione

Per ulteriori informazioni sulla gestione delle autorizzazioni di una chiave gestita dal cliente, consulta Chiavi gestite dal cliente.

> **Important**
>
> AWS Gluesupporta solo chiavi simmetriche gestite dal cliente. L'elenco delle chiavi KMS mostra solo chiavi simmetriche. Tuttavia, se si seleziona Scegli un ARN per una chiave KMS, la console consente di inserire un ARN per qualsiasi tipo di chiave. Inserisci solo ARN per le chiavi simmetriche.
> Per creare una chiave simmetrica gestita dal cliente, segui i passaggi per la creazione di chiavi simmetriche gestite dal cliente nella Guida per gli sviluppatori. AWS Key Management Service

Quando abiliti la crittografia Data Catalog at Rest, i seguenti tipi di risorse vengono crittografati utilizzando chiavi KMS:

- Database
- Tabelle
- Partizioni
- Versioni di tabella
- Statistiche delle colonne
- Funzioni definite dall'utente
- Visualizzazioni del catalogo dati

Contesto di crittografia AWS Glue

Un contesto di crittografia è un set opzionale di coppie chiave-valore che contiene ulteriori informazioni contestuali sui dati. AWS KMS utilizza il contesto di crittografia come dati autenticati aggiuntivi per supportare crittografia autenticata. Quando includi un contesto di crittografia in una richiesta di crittografia dei dati, AWS KMS lega il contesto di crittografia ai dati crittografati. Per decrittografare i dati, includi lo stesso contesto di crittografia nella richiesta. AWS Glueutilizza lo stesso contesto di crittografia in tutte le operazioni AWS KMS crittografiche, in cui la chiave è `glue_catalog_id` e il valore è il. `catalogId`

```
"encryptionContext": {
    "glue_catalog_id": "111122223333"
}
```

Quando si utilizza una chiave AWS gestita o una chiave simmetrica gestita dal cliente per crittografare il catalogo dati, è possibile utilizzare il contesto di crittografia anche nei record e nei registri di controllo per identificare come viene utilizzata la chiave. Il contesto di crittografia viene visualizzato anche nei log generati da or logs. AWS CloudTrail Amazon CloudWatch

Attivazione della crittografia

È possibile abilitare la crittografia per AWS Glue Data Catalog gli oggetti nelle impostazioni del Data Catalog nella AWS Glue console o utilizzando ilAWS CLI.

Console

Per abilitare la crittografia usando la console

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Scegli Data Catalog nel pannello di navigazione.
3. Nella pagina delle impostazioni di Data Catalog, seleziona la casella di controllo Crittografia dei metadati e scegli una AWS KMS chiave.

   Quando abiliti la crittografia, se non specifichi una chiave gestita dal cliente, le impostazioni di crittografia utilizzano una chiave KMS AWS gestita.
4. (Facoltativo) Quando utilizzi una chiave gestita dal cliente per crittografare il tuo Data Catalog, il Data Catalog offre la possibilità di registrare un ruolo IAM per crittografare e decrittografare le risorse. Devi concedere al ruolo IAM le autorizzazioni che AWS Glue puoi assumere per tuo conto. Ciò include AWS KMS le autorizzazioni per crittografare e decrittografare i dati.

   Quando crei una nuova risorsa nel Data Catalog, AWS Glue assume il ruolo IAM fornito per crittografare i dati. Allo stesso modo, quando un consumatore accede alla risorsa, AWS Glue assume il ruolo IAM di decrittografare i dati. Se registri un ruolo IAM con le autorizzazioni richieste, il principale chiamante non necessita più delle autorizzazioni per accedere alla chiave e decrittografare i dati.

   > **Important**
   >
   > Puoi delegare le operazioni KMS a un ruolo IAM solo quando utilizzi una chiave gestita dal cliente per crittografare le risorse del Data Catalog. Al momento, la funzionalità di delega dei ruoli KMS non supporta l'utilizzo di chiavi AWS gestite per la crittografia delle risorse del Data Catalog.

   > **Warning**
   >
   > Quando abiliti un ruolo IAM per delegare le operazioni KMS, non puoi più accedere alle risorse del Data Catalog che in precedenza erano crittografate con una chiave gestita. AWS

a. Per abilitare un ruolo IAM che AWS Glue può pretendere di crittografare e decrittografare i dati per tuo conto, seleziona l'opzione Delega le operazioni KMS a un ruolo IAM.

b. Quindi, scegli un ruolo IAM.

   Per creare un ruolo IAM, consulta l'argomento relativo alla creazione di ruoli IAM per AWS Glue.

   Il ruolo IAM che AWS Glue presuppone l'accesso al Data Catalog deve disporre delle autorizzazioni per crittografare e decrittografare i metadati nel Data Catalog. Puoi creare un ruolo IAM e allegare le seguenti politiche in linea:

   - Aggiungi la seguente policy per includere le AWS KMS autorizzazioni per crittografare e decrittografare il Data Catalog.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:<region>:<account-id>:key/<key-id>"
        }
    ]
}
```

   - Quindi, aggiungi la seguente policy di fiducia al ruolo affinché il AWS Glue servizio assuma il ruolo IAM.

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

```
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- Quindi, aggiungi l'`iam:PassRole`autorizzazione al ruolo IAM.

```
                    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<account-id>:role/<encryption-role-name>"
            ]
        }
    ]
}
```

Quando abiliti la crittografia, se non hai specificato un ruolo IAM AWS Glue da assumere, il principale che accede al Data Catalog deve disporre delle autorizzazioni per eseguire le seguenti operazioni API:

- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKey`

AWS CLI

Per abilitare la crittografia usando l'SDK o AWS CLI

- Usa l'operazione API `PutDataCatalogEncryptionSettings`. Se non viene specificata alcuna chiave, AWS Glue utilizza la chiave di crittografia AWS gestita per l'account del cliente per crittografare il Data Catalog.

```
aws glue put-data-catalog-encryption-settings \
  --data-catalog-encryption-settings '{
      "EncryptionAtRest": {
      "CatalogEncryptionMode": "SSE-KMS-WITH-SERVICE-ROLE",
      "SseAwsKmsKeyId": "arn:aws:kms:<region>:<account-id>:key/<key-id>",
      "CatalogEncryptionServiceRole":"arn:aws:iam::<account-
id>:role/<encryption-role-name>"
    }

  }'
```

Quando abiliti la crittografia, tutti gli oggetti che crei negli oggetti del Data Catalog vengono crittografati. Se si deseleziona questa impostazione, gli oggetti creati nel Data Catalog non vengono più crittografati. Puoi continuare ad accedere agli oggetti crittografati esistenti nel Data Catalog con le autorizzazioni KMS richieste.

> **Important**
>
> La chiave AWS KMS deve rimanere disponibile nell'archivio chiavi AWS KMS per tutti gli oggetti crittografati con tale chiave nel catalogo dati. Se rimuovi la chiave, non sarà più possibile decrittografare gli oggetti. In alcuni scenari ciò potrebbe essere necessario per impedire l'accesso ai metadati del catalogo dati.

Monitoraggio delle chiavi KMS per AWS Glue

Quando utilizzi le chiavi KMS con le risorse del tuo Data Catalog, puoi utilizzare AWS CloudTrail or Amazon CloudWatch Logs per tenere traccia delle richieste inviate AWS Glue a. AWS KMS AWS

CloudTrailmonitora e registra le operazioni KMS relative alle AWS Glue chiamate di accesso ai dati crittografati dalle tue chiavi KMS.

Gli esempi seguenti sono AWS CloudTrail gli eventi relativi alle operazioni and. `Decrypt` `GenerateDataKey`

Decrypt


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAXPHTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAXPHTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-01-10T14:33:56Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-01-10T15:18:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "eu-west-2",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "glue_catalog_id": "111122223333"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
```

```
    },
    "responseElements": null,
    "requestID": "43b019aa-34b8-4798-9b98-ee968b2d63df",
    "eventID": "d7614763-d3fe-4f84-a1e1-3ca4d2a5bbd5",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:<region>:111122223333:key/<key-id>"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
}
```

GenerateDataKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId":
 "AROAXPHTESTANDEXAMPLE:V_00_GLUE_KMS_GENERATE_DATA_KEY_111122223333",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/
V_00_GLUE_KMS_GENERATE_DATA_KEY_111122223333",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAXPHTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-01-05T21:15:47Z",
```

```
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-01-05T21:15:47Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "eu-west-2",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:eu-west-2:AKIAIOSFODNN7EXAMPLE:key/
AKIAIOSFODNN7EXAMPLE",
        "encryptionContext": {
            "glue_catalog_id": "111122223333"
        },
        "keySpec": "AES_256"
    },
    "responseElements": null,
    "requestID": "64d1783a-4b62-44ba-b0ab-388b50188070",
    "eventID": "1c73689b-2ef2-443b-aed7-8c126585ca5e",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-west-2:111122223333:key/AKIAIOSFODNN7EXAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Crittografia delle password di connessione

È possibile recuperare le password di connessione in AWS Glue Data Catalog utilizzando le operazioni API `GetConnections` e `GetConnection`. Queste password sono memorizzate nella connessione del catalogo dati e vengono utilizzate quando AWS Glue si connette a un datastore JDBC (Java Database Connectivity). Quando la connessione è stata creata o aggiornata, un'opzione nelle impostazioni del catalogo dati ha determinato se la password era crittografata e, nel caso, quale chiave AWS Key Management Service (AWS KMS) è stata specificata.

Sulla console AWS Glue, puoi attivare questa opzione nella pagina Data catalog settings (Impostazioni catalogo dati).

Crittografare le password di connessione

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Scegliere Settings (Impostazioni) nel riquadro di navigazione.
3. Nella pagina Data catalog settings (Impostazione catalogo dati), seleziona la casella di controllo Encrypt connection passwords (Crittografa password di connessione) e scegli una chiave AWS KMS.

   > **Important**
   >
   > AWS Glue supporta solo chiavi master del cliente simmetriche (CMK). L'elenco di chiavi AWS KMS mostra solo le chiavi simmetriche. Tuttavia, selezionando Choose a AWS KMS key ARN (Scegli un ARN chiave), la console consente di inserire un ARN per qualsiasi tipo di chiave. Inserisci solo ARN per le chiavi simmetriche.

   Per ulteriori informazioni, consulta Uso delle impostazioni dei cataloghi dati nella console AWS Glue.

Crittografia dei dati scritti da AWS Glue

Una configurazione della sicurezza è un set di proprietà di sicurezza che AWS Glue può usare. Puoi usare una configurazione della sicurezza per crittografare i dati inattivi. Gli scenari seguenti mostrano alcuni modi in cui è possibile usare una configurazione della sicurezza.

- Collega una configurazione di sicurezza a un AWS Glue crawler per scrivere Amazon CloudWatch Logs crittografati. Per ulteriori informazioni su come allegare configurazioni di sicurezza ai crawler, consulta. the section called “Fase 3: configurare le impostazioni di sicurezza”
- Collega una configurazione di sicurezza a un processo di estrazione, trasformazione e caricamento (ETL) per scrivere obiettivi Amazon Simple Storage Service (Amazon S3) crittografati e log crittografati. CloudWatch
- Collegare una configurazione della sicurezza a un processo ETL per scrivere i segnalibri dei processi come dati Amazon S3 crittografati.
- Collegare una configurazione della sicurezza a un endpoint di sviluppo per scrivere destinazioni Amazon S3 crittografate.

> **Important**
>
> Attualmente, una configurazione della sicurezza sostituisce qualsiasi impostazione di crittografia lato server (SSE-S3) passata come parametro di un processo ETL. Pertanto, se a un processo sono associati sia una configurazione della sicurezza che un parametro SSE-S3, il parametro SSE-S3 viene ignorato.

Per ulteriori informazioni sulle configurazioni della sicurezza, consulta Uso di configurazioni di sicurezza nella console AWS Glue.

Argomenti

- Impostazione di AWS Glue per l'uso di configurazioni della sicurezza
- Creazione di una route a AWS KMS per i crawler e i processi VPC
- Uso di configurazioni di sicurezza nella console AWS Glue

Impostazione di AWS Glue per l'uso di configurazioni della sicurezza

Segui queste fasi per impostare l'ambiente AWS Glue per l'uso di configurazioni della sicurezza.

1. Crea o aggiorna le tue chiavi AWS Key Management Service (AWS KMS) per concedere le AWS KMS autorizzazioni ai ruoli IAM che vengono passate ai AWS Glue crawler e ai job per crittografare i log. CloudWatch Per ulteriori informazioni, consulta Encrypt Log Data in CloudWatch Logs Using AWS KMS nella Amazon CloudWatch Logs User Guide.

Nell'esempio seguente *"role1"*, *"role2"* e *"role3"* sono ruoli IAM passati a crawler e processi.

```
{
       "Effect": "Allow",
       "Principal": { "Service": "logs.region.amazonaws.com",
       "AWS": [
                "role1",
                "role2",
                "role3"
             ] },
                    "Action": [
                           "kms:Encrypt*",
                           "kms:Decrypt*",
                           "kms:ReEncrypt*",
                           "kms:GenerateDataKey*",
                           "kms:Describe*"
                    ],
                    "Resource": "*"
}
```

L'`Service`istruzione, mostrata come`"Service": "logs.region.amazonaws.com"`, è obbligatoria se utilizzi la chiave per crittografare i log. CloudWatch

2. Verificare che la chiave AWS KMS sia `ENABLED` affinché sia possibile usarla.

> **Note**
>
> Se utilizzi Iceberg come framework di data lake, le tabelle Iceberg dispongono di meccanismi propri per abilitare la crittografia lato server. È necessario abilitare questa configurazione oltre alla configurazione di sicurezza di AWS Glue. Per abilitare la crittografia lato server sulle tabelle Iceberg, consulta le indicazioni contenute nella documentazione di Iceberg.

Creazione di una route a AWS KMS per i crawler e i processi VPC

Puoi connetterti direttamente a AWS KMS attraverso un endpoint privato nel cloud privato virtuale (VPC, Virtual Private Cloud) invece che tramite Internet. Quando usi un endpoint VPC, la comunicazione tra il VPC e AWS KMS avviene interamente all'interno della rete AWS.

Puoi creare un endpoint VPC AWS KMS all'interno di un VPC. Senza questa fase, i processi o i crawler potrebbero avere esito negativo, con un errore `kms timeout` nei processi o un'eccezione `internal service exception` nei crawler. Per istruzioni dettagliate, consulta Connessione a AWS KMS mediante un endpoint VPC nella Guida per gli sviluppatori di AWS Key Management Service.

Mentre segui le istruzioni, nella console VPC esegui queste operazioni:

- Selezionare Abilita nome DNS privato.
- Scegli il gruppo di sicurezza (con regola autoreferenziale) da usare per il processo o il crawler che accede a JDBC (Java Database Connectivity). Per ulteriori informazioni sulle connessioni AWS Glue, consulta Connessione ai dati.

Quando aggiungi una configurazione della sicurezza a un crawler o a un processo che accede ai datastore JDBC, è necessario che AWS Glue disponga di una route all'endpoint AWS KMS. Puoi fornire la route con un gateway Network Address Translation (NAT) o con un endpoint VPC AWS KMS. Per creare un gateway NAT, consulta Gateway NAT nella Guida per l'utente di Amazon VPC.

Uso di configurazioni di sicurezza nella console AWS Glue

> **Warning**
>
> Le configurazioni di sicurezza di AWS Glue non sono attualmente supportate nei processi Ray.

Una configurazione di sicurezza in AWS Glue contiene le proprietà necessarie per la scrittura di dati crittografati. Puoi creare configurazioni di sicurezza nella console AWS Glue per specificare le proprietà di crittografia usate da crawler, processi ed endpoint di sviluppo.

Per visualizzare un elenco delle configurazioni di sicurezza che hai creato, apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/ e scegli Security configurations (Configurazioni di sicurezza) nel pannello di navigazione.

L'elenco Security configurations (Configurazioni di sicurezza) visualizza le proprietà seguenti su ogni configurazione:

Nome

Nome univoco che hai assegnato quando hai creato la configurazione. Il nome può contenere un massimo di 255 caratteri, tra cui lettere (A-Z), numeri (0-9), trattini (-) o caratteri di sottolineatura (_).

Attivazione della crittografia in Amazon S3

Se attivata, la modalità di crittografia Amazon Simple Storage Service (Amazon S3), ad esempio `SSE-KMS` o `SSE-S3`, è abilitata per l'archiviazione di metadati nel catalogo dati.

Abilitazione della crittografia dei file di log Amazon CloudWatch

Se attivata, la modalità di crittografia di Amazon S3, ad esempio `SSE-KMS`, è abilitata durante la scrittura dei log su Amazon CloudWatch.

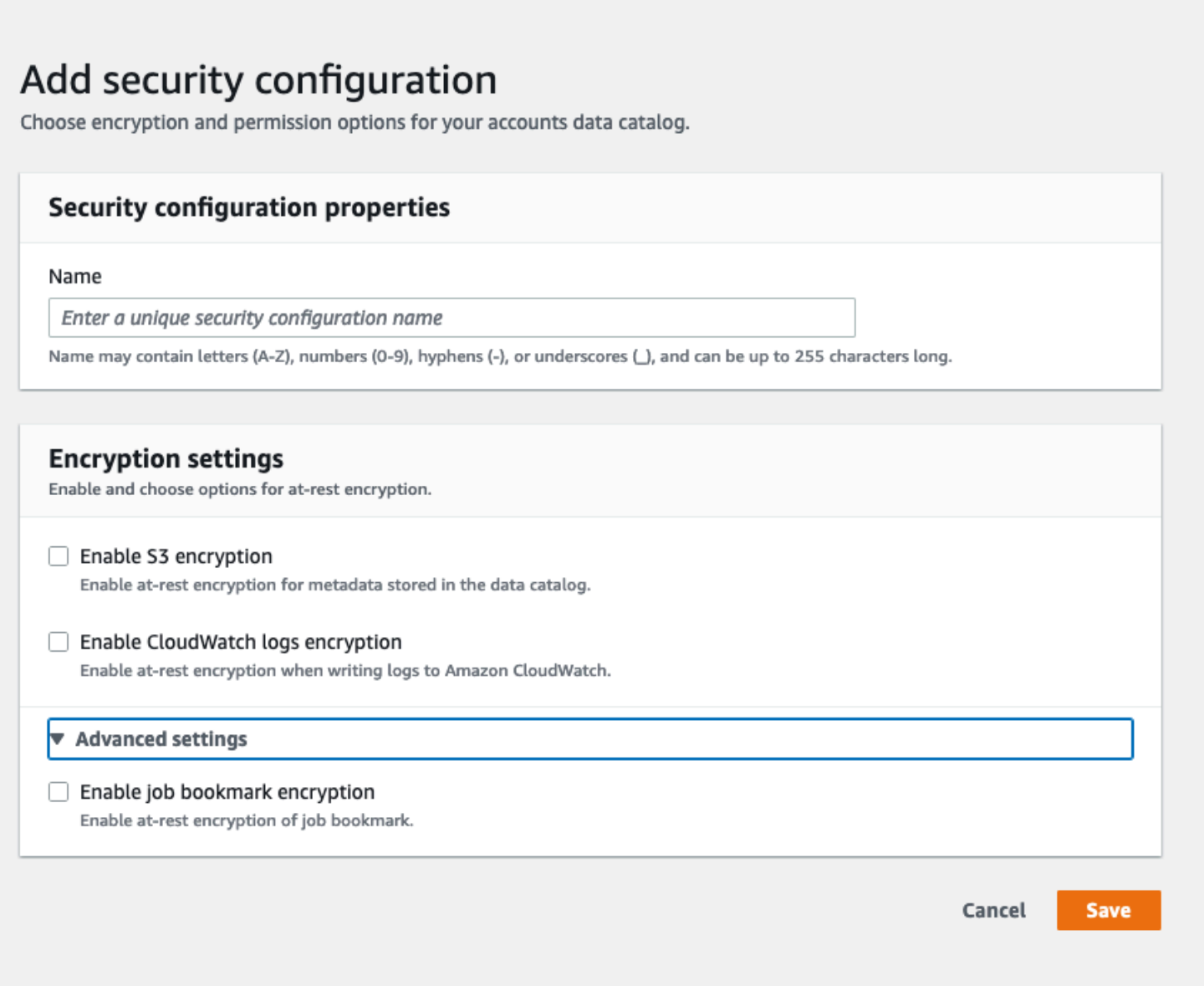
Impostazioni avanzate: abilitazione della crittografia dei segnalibri del processo

Se attivata, la modalità di crittografia di Amazon S3, ad esempio `CSE-KMS`, è abilitata quando i processi vengono aggiunti ai segnalibri.

Puoi aggiungere o eliminare configurazioni nella sezione Security configurations (Configurazioni di sicurezza) nella console. Per visualizzare altri dettagli relativi a una configurazione, scegli il nome della configurazione nell'elenco. I dettagli includono le informazioni che hai definito quando hai creato la configurazione.

Aggiunta di una configurazione di sicurezza

Per aggiungere una configurazione di sicurezza usando la console AWS Glue, nella pagina Security configurations (Configurazioni di sicurezza) scegli Add security configuration (Aggiungi configurazione di sicurezza.



Proprietà di configurazione di sicurezza

Inserisci un nome univoco per la configurazione di sicurezza. Il nome può contenere un massimo di 255 caratteri, tra cui lettere (A-Z), numeri (0-9), trattini (-) o caratteri di sottolineatura (_).

Impostazioni di crittografia

È possibile abilitare la crittografia dei dati inattivi per i metadati archiviati in Catalogo dati di Amazon S3 e i log in Amazon CloudWatch. Per configurare la crittografia dei dati e dei metadati con chiavi AWS Key Management Service (AWS KMS) nella console AWS Glue, aggiungi una policy all'utente della console. Questa policy deve specificare le risorse consentite come ARN di chiavi usati per crittografare datastore Amazon S3, come nell'esempio seguente.

```
{
"Version": "2012-10-17",
  "Statement": {
  "Effect": "Allow",
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt",
    "kms:Encrypt"],
  "Resource": "arn:aws:kms:region:account-id:key/key-id"}
}
```

> **Important**
>
> Quando una configurazione di sicurezza viene collegata a un crawler o a un processo, il ruolo IAM passato deve avere autorizzazioni AWS KMS. Per ulteriori informazioni, consulta Crittografia dei dati scritti da AWS Glue.

Quando definisci una configurazione, puoi specificare i valori per le proprietà seguenti:

Abilitazione della crittografia in S3

Quando scrivi dati Amazon S3, puoi usare la crittografia lato server con chiavi gestite da Amazon S3 (SSE-S3) o la crittografia lato server con chiavi gestite da AWS KMS (SSE-KMS). Questo campo è facoltativo. Per permettere l'accesso ad Amazon S3, scegli una chiave AWS KMS oppure seleziona Enter a key ARN (Inserisci un ARN chiave) e specifica l'ARN per la chiave. Immetti l'ARN usando questo formato: `arn:aws:kms:region:account-id:key/key-id`. Puoi specificare l'ARN anche sotto forma di alias di chiavi, ad esempio `arn:aws:kms:region:account-id:alias/alias-name`.

Se abiliti l'interfaccia utente Spark per il processo, il file di log dell'interfaccia utente di Spark caricato su Amazon S3 verrà applicato con la stessa crittografia.

> **Important**
>
> AWS Glue supporta solo chiavi master del cliente simmetriche (CMK). L'elenco di chiavi AWS KMS mostra solo le chiavi simmetriche. Tuttavia, selezionando Choose a AWS KMS key ARN (Scegli un ARN chiave), la console consente di inserire un ARN per qualsiasi tipo di chiave. Inserisci solo ARN per le chiavi simmetriche.

Abilitazione della crittografia CloudWatch Logs

Crittografia lato server (SSE-KMS) usata per crittografare CloudWatch Logs. Questo campo è facoltativo. Per attivare questa crittografia, scegli una chiave AWS KMS oppure seleziona Enter a key ARN (Inserisci ARN chiave) e specifica l'ARN per la chiave. Immetti l'ARN usando questo formato: `arn:aws:kms:`*`region`*`:`*`account-id`*`:key/`*`key-id`*. Puoi specificare l'ARN anche sotto forma di alias di chiavi, ad esempio `arn:aws:kms:`*`region`*`:`*`account-id`*`:alias/`*`alias-name`*.

Impostazioni avanzate: crittografia dei segnalibri del processo

Crittografia lato client (CSE-KMS) usata per crittografare segnalibri dei processi. Questo campo è facoltativo. I dati dei segnalibri vengono crittografati prima di essere inviati ad Amazon S3 per lo storage. Per attivare questa crittografia, scegli una chiave AWS KMS oppure seleziona Enter a key ARN (Inserisci ARN chiave) e specifica l'ARN per la chiave. Immetti l'ARN usando questo formato: `arn:aws:kms:`*`region`*`:`*`account-id`*`:key/`*`key-id`*. Puoi specificare l'ARN anche sotto forma di alias di chiavi, ad esempio `arn:aws:kms:`*`region`*`:`*`account-id`*`:alias/`*`alias-name`*.

Per ulteriori informazioni, consulta i seguenti argomenti nella Guida per l'utente di Amazon Simple Storage Service:

- Per informazioni su `SSE-S3`, consulta Protezione dei dati mediante la crittografia lato server con chiavi crittografia gestite da Amazon S3 (SSE-S3).
- Per ulteriori informazioni su `SSE-KMS`, consulta la pagina Protecting Data Using Server-Side Encryption with AWS KMS keys.
- Per ulteriori informazioni su `CSE-KMS`, consulta la pagina Using a KMS key stored in AWS KMS.

Crittografia dei dati in transito

AWS fornisce la crittografia Transport Layer Security (TLS) per i dati in movimento. È possibile configurare le impostazioni di crittografia per crawler, processi ETL ed endpoint di sviluppo utilizzando le configurazioni di sicurezza in AWS Glue. È possibile attivare la crittografia AWS Glue Data Catalog tramite le impostazioni per il catalogo dati.

A partire dal 4 settembre 2018 è supportata la AWS KMS (chiave personalizzata e crittografia lato server) per ETL AWS Glue e AWS Glue Data Catalog.

Conformità a FIPS

Se si richiedono moduli crittografici convalidati FIPS 140-2 quando si accede ad AWS tramite una CLI o un'API, utilizzare un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Gestione delle chiavi

È possibile usare AWS Identity and Access Management (IAM) con AWS Glue per definire utenti, risorse AWS, gruppi, ruoli e policy ad alta granularità riguardanti accesso, interdizione e molto altro.

È possibile definire l'accesso ai metadati utilizzando policy basate sulle risorse e basate sull'identità, a seconda delle esigenze dell'organizzazione. Le policy basate sulle risorse elencano le entità alle quali viene concesso o negato l'accesso alle risorse, permettendo di configurare policy come l'accesso multi-account. Le policy basate sull'identità sono specificamente collegate a utenti, gruppi e ruoli all'interno di IAM.

Per un esempio dettagliato, consulta Restrict access to your AWS Glue Data Catalog with resource-level IAM permissions and resource-based policies nel blog dei Big Data di AWS.

La parte dedicata all'accesso granulare della policy è definita all'interno della clausola `Resource`. Questa porzione definisce sia l'oggetto AWS Glue Data Catalog sul quale può essere invocata l'operazione, sia quali oggetti risultati vengono restituiti da tale operazione.

Un endpoint di sviluppo è un ambiente che puoi usare per sviluppare e testare gli script AWS Glue. Puoi aggiungere, eliminare o ruotare la chiave SSH di un endpoint di sviluppo.

A partire dal 4 settembre 2018 è supportata la AWS KMS (chiave personalizzata e crittografia lato server) per ETL AWS Glue e AWS Glue Data Catalog.

Dipendenza di AWS Glue da altri servizi AWS

Per poter usare la console di AWS Glue, un utente deve disporre di un set minimo di autorizzazioni che gli permetta di usare le risorse di AWS Glue per l'account AWS. Oltre a queste autorizzazioni AWS Glue, la console richiede le autorizzazioni dei servizi seguenti:

- Autorizzazioni Amazon CloudWatch Logs per visualizzare i log.
- Autorizzazioni AWS Identity and Access Management (IAM) per elencare e passare i ruoli.

- Autorizzazioni AWS CloudFormation per usare gli stack.
- Autorizzazioni Amazon Elastic Compute Cloud (Amazon EC2) per elencare cloud privati virtuali (VPC), sottoreti, gruppi di sicurezza, istanze e altri oggetti (per impostare elementi Amazon EC2 quali, ad esempio, i VPC durante l'esecuzione di processi, crawler e creazione di endpoint di sviluppo).
- Autorizzazioni Amazon Simple Storage Service (Amazon S3) per elencare bucket e oggetti e per recuperare e salvare script.
- Autorizzazioni Amazon Redshift necessarie per l'utilizzo dei cluster.
- Autorizzazioni Amazon Relational Database Service (Amazon RDS) per elencare le istanze.

Endpoint di sviluppo

Un endpoint di sviluppo è un ambiente che puoi usare per sviluppare e testare gli script AWS Glue. È possibile utilizzare AWS Glue per creare, modificare ed eliminare gli endpoint di sviluppo. È possibile elencare tutti gli endpoint di sviluppo che vengono creati. Puoi aggiungere, eliminare o ruotare la chiave SSH di un endpoint di sviluppo. Puoi anche creare notebook che utilizzano l'endpoint di sviluppo.

Puoi fornire i valori di configurazione per effettuare il provisioning degli ambienti di sviluppo. Questi valori indicano a AWS Glue come configurare la rete in modo da poter accedere all'endpoint di sviluppo in modo sicuro e da poter accedere ai datastore. Quindi, è possibile creare un notebook che si colleghi all'endpoint di sviluppo. È possibile utilizzare il notebook per creare e testare lo script ETL.

Scegliere un ruolo AWS Identity and Access Management (IAM) con autorizzazioni simili a quelle del ruolo IAM usato per eseguire i processi ETL di AWS Glue. Utilizzare un cloud privato virtuale (VPC), una sottorete e un gruppo di sicurezza per creare un endpoint di sviluppo che sia in grado di connettersi alle risorse dati in modo sicuro. È possibile generare una coppia di chiavi SSH per connettersi all'ambiente di sviluppo tramite SSH.

È possibile creare endpoint di sviluppo per i dati Amazon S3 e all'interno di un VPC che è possibile utilizzare per accedere ai set di dati utilizzando JDBC.

È possibile installare un notebook Jupyter sul computer locale e utilizzarlo per eseguire il debug e testare gli script ETL in un endpoint di sviluppo. In alternativa, è possibile utilizzare un notebook Sagemaker per creare script ETL in JupyterLab su AWS. Consulta Utilizzo di un notebook SageMaker con l'endpoint di sviluppo .

AWS Glue assegna dei tag alle istanze Amazon EC2 con un nome che include il prefisso `aws-glue-dev-endpoint`.

È possibile configurare un server notebook in un endpoint di sviluppo per eseguire PySpark con estensioni AWS Glue.

Gestione delle identità e degli accessi per AWS Glue

AWS Identity and Access Management (IAM) è un Servizio AWS che consente agli amministratori di controllare in modo sicuro l'accesso alle risorse AWS. Gli amministratori IAM controllano chi è autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare risorse AWS Glue. IAM è un Servizio AWS il cui uso non comporta costi aggiuntivi.

> **Note**
>
> È possibile concedere l'accesso ai dati nel catalogo dati di AWS Glue utilizzando metodi AWS Glue o concessioni AWS Lake Formation. Puoi utilizzare policy (IAM) AWS Identity and Access Management per impostare un controllo granulare degli accessi con metodi AWS Glue. Lake Formation utilizza un modello di autorizzazioni `GRANT/REVOKE` più semplice, che è simile ai comandi `GRANT/REVOKE` in un sistema di database relazionale.
> Questa sezione include informazioni su come utilizzare i metodi AWS Glue. Per ulteriori informazioni sull'utilizzo delle autorizzazioni Lake Formation, consulta Concedere autorizzazioni Lake Formation nella Guida per gli sviluppatori di AWS Lake Formation.

Argomenti

- Risoluzione dei problemi di identità e accesso in AWS Glue

Destinatari

Le modalità di utilizzo di AWS Identity and Access Management (IAM) cambiano in base alle operazioni eseguite in AWS Glue.

Utente del servizio: se utilizzi il servizio AWS Glue per eseguire il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di funzionalità AWS Glue utilizzate per il lavoro, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di AWS Glue, consulta Risoluzione dei problemi di identità e accesso in AWS Glue.

Amministratore del servizio: se sei il responsabile delle risorse AWS Glue presso la tua azienda, probabilmente disponi dell'accesso completo a AWS Glue. Il tuo compito è determinare le caratteristiche e le risorse AWS Glue a cui gli utenti del servizio devono accedere. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con AWS Glue, consulta Funzionamento di AWS Glue con IAM.

Amministratore IAM: un amministratore IAM potrebbe essere interessato a ottenere dei dettagli su come scrivere policy per gestire l'accesso ad AWS Glue. Per visualizzare policy basate su identità di AWS Glue di esempio che puoi utilizzare in IAM, consulta Esempi di policy basate su identità per AWS Glue.

Autenticazione con identità

L'autenticazione è la procedura di accesso ad AWS con le credenziali di identità. Devi essere autenticato (connesso a AWS) come utente root Utente root dell'account AWS, come utente IAM o assumere un ruolo IAM.

Puoi accedere ad AWS come identità federata utilizzando le credenziali fornite attraverso un'origine di identità. AWS IAM Identity Center Gli esempi di identità federate comprendono gli utenti del centro identità IAM, l'autenticazione Single Sign-On (SSO) dell'azienda e le credenziali di Google o Facebook. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Se accedi ad AWS tramite la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere alla AWS Management Console o al portale di accesso AWS. Per ulteriori informazioni sull'accesso ad AWS, consulta la sezione Come accedere al tuo Account AWS nella Guida per l'utente di Accedi ad AWS.

Se accedi ad AWS in modo programmatico, AWS fornisce un Software Development Kit (SDK) e un'interfaccia della linea di comando (CLI) per firmare crittograficamente le richieste utilizzando le tue credenziali. Se non utilizzi gli strumenti AWS, devi firmare le richieste personalmente. Per ulteriori informazioni sulla firma delle richieste, consultare Firma delle richieste AWS nella Guida per l'utente di IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. AWS consiglia ad esempio di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza dell'account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente di IAM.

Utente root di un Account AWS

Quando crei un Account AWS, inizi con una singola identità di accesso che ha accesso completo a tutti i Servizi AWS e le risorse nell'account. Tale identità è detta utente root Account AWS ed è possibile accedervi con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Tasks that require root user credentials (Attività che richiedono le credenziali dell'utente root) nella Guida per l'utente di IAM.

Identità federata

Come best practice, richiedere agli utenti umani, compresi quelli che richiedono l'accesso di amministratore, di utilizzare la federazione con un provider di identità per accedere a Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory degli utenti aziendali, un provider di identità Web, AWS Directory Service, la directory Identity Center o qualsiasi utente che accede ad Servizi AWS utilizzando le credenziali fornite tramite un'origine di identità. Quando le identità federate accedono ad Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. È possibile creare utenti e gruppi in IAM Identity Center oppure connettersi e sincronizzarsi con un

gruppo di utenti e gruppi nell'origine di identità per utilizzarli in tutte le applicazioni e gli Account AWS. Per ulteriori informazioni su IAM Identity Center, consulta Cos'è IAM Identity Center? nella Guida per l'utente di AWS IAM Identity Center.

Utenti e gruppi IAM

Un utente IAM è una identità all'interno del tuo Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, se si hanno casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente di IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato Amministratori IAM e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità all'interno di Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. È possibile assumere temporaneamente un ruolo IAM nella AWS Management Console mediante lo scambio di ruoli. È possibile assumere un ruolo chiamando un'operazione AWS CLI o API AWS oppure utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene

autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center.

- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, per alcuni dei Servizi AWS, è possibile collegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.
- Accesso multi-servizio: alcuni Servizi AWS utilizzano funzionalità che in altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Inoltro delle sessioni di accesso (FAS): quando utilizzi un ruolo o un utente IAM per eseguire azioni in AWS, sei considerato un principale. Quando utilizzi alcuni servizi, puoi eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altre risorse o Servizi AWS per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Inoltro sessioni di accesso.
  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli

collegati ai servizi sono visualizzati nell'account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: è possibile utilizzare un ruolo IAM per gestire credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 che eseguono richieste di AWS CLI o dell'API AWS. Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un ruolo AWS a un'istanza EC2, affinché sia disponibile per tutte le relative applicazioni, puoi creare un profilo dell'istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Per controllare l'accesso a AWS è possibile creare policy e collegarle a identità o risorse AWS. Una policy è un oggetto in AWS che, quando associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste policy quando un principale IAM (utente, utente root o sessione ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle policy viene archiviata in AWS sotto forma di documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare l'accesso ai diversi elementi. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla AWS Management Console, la AWS CLI o l'API AWS.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo utente, gruppo o ruolo. Le policy gestite sono policy autonome che possono essere collegate a più utenti, gruppi e ruoli in Account AWS. Le policy gestite includono le policy gestite da AWS e le policy gestite dal cliente. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente di IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di trust dei ruoli IAM e le policy dei 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 allegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy gestite da AWS da IAM in una policy basata su risorse.

Liste di controllo accessi (ACL)

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 sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3, AWS WAF e Amazon VPC sono esempi di servizi che supportano le ACL. Per maggiori informazioni sulle ACL, consulta Panoramica delle liste di controllo degli accessi (ACL) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta altri tipi di policy meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Policy di controllo dei servizi (SCP): le SCP sono policy JSON che specificano il numero massimo di autorizzazioni per un'organizzazione o unità organizzativa (OU) in AWS Organizations. AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata degli Account AWS multipli di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. La SCP limita le autorizzazioni per le entità negli account membri, compreso ogni Utente root dell'account AWS. Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella Guida per l'utente di IAM.

Funzionamento di AWS Glue con IAM

Prima di utilizzare IAM per gestire l'accesso ad AWS Glue, scopri quali funzionalità IAM possono essere utilizzate con AWS Glue.

Funzionalità IAM che è possibile utilizzare con AWS Glue

Funzionalità IAM	Supporto per AWS Glue
Policy basate su identità	Sì
Policy basate su risorse	Parziale
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì
Liste di controllo degli accessi	No
ABAC (tag nelle policy)	Parziale
Credenziali temporanee	Sì
Autorizzazioni del principale	No
Ruoli di servizio	Sì
Ruoli collegati al servizio	No

Per ottenere un quadro generale del funzionamento di AWS Glue e di altri servizi AWS con la maggior parte delle funzionalità IAM, consulta Servizi AWS supportati da IAM nella Guida per l'utente IAM.

Policy basate su identità per AWS Glue

Supporta le policy basate su identità	Sì

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Con le policy basate su identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o rifiutate, nonché le condizioni in base alle quali le operazioni sono consentite o rifiutate. Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per 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 di IAM.

AWS Glue supporta le policy basate su identità (policy IAM) per tutte le operazioni AWS Glue. Collegando una policy, puoi concedere le autorizzazioni per creare, accedere o modificare una risorsa AWS Glue, ad esempio una tabella nel AWS Glue Data Catalog.

Esempi di policy basate su identità per AWS Glue

Per visualizzare esempi di policy basate su identità di AWS Glue, consulta Esempi di policy basate su identità per AWS Glue.

Policy basate su risorse all'interno di AWS Glue

Supporta le policy basate su risorse	Parziale

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di trust dei ruoli IAM e le policy dei 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 allegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Per consentire l'accesso multi-account, puoi specificare un intero account o entità IAM in un altro account come principale in una policy basata sulle risorse. L'aggiunta di un principale multi-account a una policy basata su risorse rappresenta solo una parte della relazione di attendibilità. Quando l'entità principale e la risorsa si trovano in diversi Account AWS, un amministratore IAM nell'account

attendibile deve concedere all'entità principale (utente o ruolo) anche l'autorizzazione per accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una policy basata su risorse concede l'accesso a un principale nello stesso account, non sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

> **Note**
>
> Una policy delle risorse AWS Glue può essere utilizzata solo per gestire le autorizzazioni per le risorse di catalogo dati. Non è possibile collegarla ad altre risorse AWS Glue, ad esempio processi, trigger, endpoint di sviluppo, crawler o classificatori.
> È ammessa solo una policy sulle risorse per catalogo e la sua dimensione è limitata a 10 KB.

In AWS Glue, una policy delle risorse è collegata a un catalogo che è un container virtuale per tutti i tipi di risorse del catalogo dati menzionati in precedenza. Ogni account AWS dispone di un singolo catalogo in una regione AWS, il cui ID catalogo è lo stesso ID account AWS. Non è possibile eliminare o modificare un catalogo.

Una policy della risorsa viene valutata per tutte le chiamate al catalogo effettuate dall'API. In questo caso, il principale del chiamante è incluso nel blocco `"Principal"` del documento delle policy.

Per visualizzare esempi di policy basate su risorse AWS Glue, consulta Esempi di policy basate su risorse per AWS Glue.

Operazioni di policy per AWS Glue

Supporta le azioni di policy	Sì

Gli amministratori possono utilizzare le policy JSON AWS per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Le azioni di policy hanno spesso lo stesso nome dell'operazione API AWS. Ci sono alcune eccezioni, ad esempio le azioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più azioni in una policy. Queste azioni aggiuntive sono denominate azioni dipendenti.

Includi le azioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di operazioni AWS Glue, consulta Operazioni definite da AWS Glue nella documentazione di riferimento per l'autorizzazione del servizio.

Le operazioni di policy in AWS Glue utilizzano il seguente prefisso prima dell'operazione:

```
glue
```

Per specificare più azioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "glue:action1",
      "glue:action2"
         ]
```

È possibile specificare più operazioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le operazioni che iniziano con la parola `Get`, includi la seguente operazione:

```
"Action": "glue:Get*"
```

Per visualizzare le policy di esempio, consulta Esempi di policy di controllo degli accessi di AWS Glue.

Risorse di policy per AWS Glue

Supporta le risorse di policy	Sì

Gli amministratori possono utilizzare le policy JSON AWS per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni 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. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Puoi eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per ulteriori informazioni su come controllare l'accesso alle risorse AWS Glue utilizzando gli ARN, consulta Come specificare gli ARN delle risorse AWS Glue.

Per visualizzare un elenco di tipi di risorse AWS Glue e dei relativi ARN, consulta le Risorse definite da AWS Glue nella documentazione di riferimento per l'autorizzazione del servizio. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consulta le Operazioni definite da AWS Glue.

Chiavi di condizione delle policy per AWS Glue

Supporta le chiavi di condizione delle policy specifiche del servizio	Sì

Gli amministratori possono utilizzare le policy JSON AWS per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition` (o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition` è facoltativo. Puoi compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND` logica. Se specifichi più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione `OR` logica. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche per il servizio. Per visualizzare tutte le chiavi di condizione globali di AWS, consulta Chiavi di contesto delle condizioni globali di AWS nella Guida per l'utente di IAM.

Per visualizzare un elenco di chiavi di condizione di AWS Glue, consulta le Chiavi di condizione per AWS Glue nella documentazione di riferimento per l'autorizzazione del servizio. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consulta la sezione Operazioni definite da AWS Glue.

Per visualizzare le policy di esempio, consulta Controllo delle impostazioni utilizzando le chiavi di condizione o di contesto.

ACL in AWS Glue

Supporta le ACL	No

Le policy di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni ad accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con AWS Glue

Supporta ABAC (tag nelle policy)	Parziale

Il controllo degli accessi basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, tali attributi sono denominati tag. È possibile collegare dei tag alle entità IAM (utenti o ruoli) e a numerose risorse AWS. L'assegnazione di tag alle entità e alle risorse è il primo passaggio di ABAC. In seguito, vengono progettate policy ABAC per consentire operazioni quando il tag dell'entità principale corrisponde al tag sulla risorsa a cui si sta provando ad accedere.

La strategia ABAC è utile in ambienti soggetti a una rapida crescita e aiuta in situazioni in cui la gestione delle policy diventa impegnativa.

Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Partial (Parziale).

Per ulteriori informazioni su ABAC, consulta Che cos'è ABAC? nella Guida per l'utente di IAM. Per visualizzare un tutorial con le fasi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

> **Important**
>
> Le chiavi di contesto della condizione si applicano solo alle operazioni API AWS Glue su crawler, processi, trigger ed endpoint di sviluppo. Per ulteriori informazioni sulle operazioni API interessate, consulta Chiavi di condizione per AWS Glue.
> Le operazioni dell'API AWS Glue catalogo dati attualmente non supportano le chiavi di contesto delle condizioni globali `aws:referer` e `aws:UserAgent`.

Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Autorizzazione dell'accesso utilizzando tag.

Utilizzo di credenziali temporanee con AWS Glue

Supporta le credenziali temporanee	Sì

Alcuni Servizi AWS non funzionano quando si accede utilizzando credenziali temporanee. Per ulteriori informazioni, inclusi i Servizi AWS che funzionano con le credenziali temporanee, consulta Servizi AWS supportati da IAM nella Guida per l'utente di IAM.

Le credenziali temporanee sono utilizzate se si accede alla AWS Management Console utilizzando qualsiasi metodo che non sia la combinazione di nome utente e password. Ad esempio, quando accedi alla AWS utilizzando il collegamento Single Sign-On (SSO) della tua azienda, tale processo crea in automatico credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sullo scambio dei ruoli, consulta Cambio di un ruolo (console) nella Guida per l'utente di IAM.

È possibile creare manualmente credenziali temporanee utilizzando la AWS CLI o l'API AWS. È quindi possibile utilizzare tali credenziali temporanee per accedere ad AWS. AWS consiglia di

generare le credenziali temporanee dinamicamente anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza provvisorie in IAM.

Autorizzazioni del principale tra servizi per AWS Glue

Supporta l'inoltro delle sessioni di accesso (FAS)	No

Quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, si viene considerati un principale. Quando utilizzi alcuni servizi, puoi eseguire un'operazione che attiva un'altra operazione in un servizio differente. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altre risorse o Servizi AWS per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Inoltro sessioni di accesso.

Ruoli di servizio per AWS Glue

Supporta i ruoli di servizio	Sì

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe compromettere la funzionalità di AWS Glue. Modifica i ruoli di servizio solo quando AWS Glue fornisce le indicazioni per farlo.

Per istruzioni dettagliate sulla creazione di un ruolo di servizio per AWS Glue, consulta Fase 1: creare una policy IAM per il servizio AWS Glue e Fase 2: creare un ruolo IAM per AWS Glue.

Ruoli collegati ai servizi per AWS Glue

Supporta i ruoli collegati ai servizi	No

Un ruolo collegato ai servizi è un tipo di ruolo di servizio che è collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Configurazione delle autorizzazioni IAM per AWS Glue

È possibile utilizzare AWS Identity and Access Management (IAM) per definire le policy e i ruoli che AWS Glue utilizza per accedere alle risorse. I passaggi seguenti illustrano varie opzioni per la configurazione delle autorizzazioni per AWS Glue. A seconda delle necessità aziendali, potrebbe essere necessario aggiungere o ridurre l'accesso alle risorse.

> **Note**
>
> Se invece desideri iniziare a utilizzare le autorizzazioni IAM di base per AWS Glue, consulta la pagina Impostazione delle autorizzazioni IAM per AWS Glue.

1. Crea una policy IAM per il servizio AWS Glue: crea una policy del servizio che permette l'accesso alle risorse AWS Glue.
2. Crea un ruolo IAM per AWS Glue: crea un ruolo IAM e collega la policy di servizio AWS Glue e una policy per le risorse Amazon Simple Storage Service (Amazon S3) utilizzate da AWS Glue.
3. Collega una policy agli utenti o ai gruppi che accedono a AWS Glue: collega le policy a qualsiasi utente o gruppo che accede alla console AWS Glue.
4. Crea una policy IAM per i notebook: crea una policy di server notebook da usare per la creazione di server notebook negli endpoint di sviluppo.

5. Crea un ruolo IAM per i notebook: crea un ruolo IAM e collega la policy del server notebook.
6. Crea una policy IAM per i notebook Amazon SageMaker: crea una policy IAM da utilizzare durante la creazione di notebook Amazon SageMaker sugli endpoint di sviluppo.
7. Crea un ruolo IAM per i notebook Amazon SageMaker: crea un ruolo IAM e collega la policy per concedere le autorizzazioni durante la creazione di notebook Amazon SageMaker sugli endpoint di sviluppo.

Fase 1: creare una policy IAM per il servizio AWS Glue

Per qualsiasi operazione che accede ai dati di un'altra risorsa AWS, ad esempio l'accesso agli oggetti in Amazon S3, AWS Glue richiede l'autorizzazione di accesso alla risorsa per tuo conto. È possibile fornire queste autorizzazioni usando AWS Identity and Access Management (IAM).

> **Note**
>
> Puoi ignorare questa fase se usi la policy gestita AWS **`AWSGlueServiceRole`**.

In questa fase, crei una policy simile a `AWSGlueServiceRole`. Puoi trovare la versione più recente di `AWSGlueServiceRole` nella console IAM.

Per creare una policy IAM per AWS Glue

Questa policy concede l'autorizzazione per alcune operazioni Amazon S3 per gestire le risorse nell'account richieste da AWS Glue quando assume il ruolo usando la policy. Alcune delle risorse specificate nella policy fanno riferimento a nomi predefiniti usati da AWS Glue per i bucket Amazon S3, gli script ETL Amazon S3, le voci di CloudWatch Logs e le risorse Amazon EC2. Per semplicità, AWS Glue scrive alcuni oggetti Amazon S3 nei bucket nell'account con il prefisso `aws-glue-*` per impostazione predefinita.

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra seleziona Policies (Policy).
3. Scegliere Create Policy (Crea policy).
4. Nella schermata Create Policy (Crea policy), passa alla scheda per modificare JSON. Crea un documento di policy con le seguenti istruzioni JSON, quindi scegli Review policy (Verifica policy).

> **Note**
>
> Aggiungi le autorizzazioni necessarie per le risorse Amazon S3. Potresti voler esplorare la sezione delle risorse della policy di accesso solo con le risorse necessarie.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetBucketAcl",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeRouteTables",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*/*",
                "arn:aws:s3:::*/*aws-glue-*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::crawler-public*",
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:AssociateKmsKey"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws-glue/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Condition": {
```

```
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "aws-glue-service-resource"
                    ]
                }
            },
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:instance/*"
            ]
        }
    ]
}
```

La tabella seguente descrive le autorizzazioni concesse dalla policy.

Action	Resource (Risorsa)	Descrizione
`"glue:*"`	`"*"`	Concede l'autorizzazione per eseguire tutte le operazioni API AWS Glue.
`"s3:GetBucketLocation", "s3:ListBucket", "s3:ListAllMyBuckets", "s3:GetBucketAcl",`	`"*"`	Permette di elencare i bucket Amazon S3 da crawler, processi, endpoint di sviluppo e server notebook.

Action	Resource (Risorsa)	Descrizione
`"ec2:DescribeVpcEndpoints", "ec2:DescribeRouteTables", "ec2:CreateNetworkInterface", "ec2:DeleteNetworkInterface", "ec2:DescribeNetworkInterfaces", "ec2:DescribeSecurityGroups", "ec2:DescribeSubnets", "ec2:DescribeVpcAttribute",`	`"*"`	Consente l'installazione di elementi di rete Amazon EC2, ad esempio cloud privati virtuali (VPC) durante l'esecuzione di processi, crawler ed endpoint di sviluppo.
`"iam:ListRolePolicies", "iam:GetRole", "iam:GetRolePolicy"`	`"*"`	Permette di elencare i ruoli IAM da crawler, processi, endpoint di sviluppo e server notebook.
`"cloudwatch:PutMetricData"`	`"*"`	Permette di scrivere i parametri CloudWatch per i processi.
`"s3:CreateBucket", "s3:PutBucketPublicAccessBlock"`	`"arn:aws:s3:::aws-glue-*"`	Consente la creazione di bucket Amazon S3 nel tuo account da processi e server notebook. Convenzione per la denominazione: utilizza cartelle Amazon S3 denominate aws-glue-. Consente ad AWS Glue di creare i bucket che bloccano l'accesso pubblico.

Action	Resource (Risorsa)	Descrizione
`"s3:GetObject", "s3:PutObject", "s3:DeleteObject"`	`"arn:aws:s3:::aws-glue-*/*", "arn:aws:s3:::*/*aws-glue-*/*"`	Permette di ottenere, inserire ed eliminare oggetti Amazon S3 nell'account quando vengono archiviati i oggetti come script ETL e posizioni dei server notebook. Convenzione per la denominazione: concede l'autorizzazione alle cartelle o ai bucket Amazon S3 i cui nomi hanno il prefisso aws-glue-.
`"s3:GetObject"`	`"arn:aws:s3:::crawler-public*", "arn:aws:s3:::aws-glue-*"`	Permette di ottenere gli oggetti Amazon S3 usati da esempi e tutorial da crawler e processi. Convenzione per la denominazione: i nomi di bucket Amazon S3 iniziano con crawler-public e aws-glue-.
`"logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents"`	`"arn:aws:logs:*:*:log-group:/aws-glue/*"`	Consente di scrivere i log in CloudWatch Logs. Convenzione di denominazione: AWS Glue scrive i log in gruppi di log i cui nomi iniziano con aws-glue.

Action	Resource (Risorsa)	Descrizione
`"ec2:CreateTags", "ec2:DeleteTags"`	`"arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:instance/*"`	Consente il tagging delle risorse Amazon EC2 create per gli endpoint di sviluppo. Convenzione di denominazione: AWS Glue applica i tag a interfacce di rete Amazon EC2, gruppi di sicurezza e istanze con aws-glue-service-resource.

5. Nella schermata Review Policy (Verifica policy), inserisci il Policy Name (Nome policy), ad esempio GlueServiceRolePolicy. Digita una descrizione facoltativa e, al termine, seleziona Create policy (Crea policy).

Fase 2: creare un ruolo IAM per AWS Glue

A questo punto devi concedere le autorizzazioni del ruolo IAM che AWS Glue può assumere quando chiama altri servizi per tuo conto. Ciò include l'accesso ad Amazon S3 per le origini, le destinazioni, gli script e le directory temporanee che usi con AWS Glue. Le autorizzazioni sono necessarie per crawler, processi ed endpoint di sviluppo.

È possibile fornire queste autorizzazioni usando AWS Identity and Access Management (IAM). Aggiungi una policy al ruolo IAM passato a AWS Glue.

Per creare un ruolo IAM per AWS Glue

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra seleziona Ruoli.
3. Scegliere Crea ruolo.
4. Per il tipo di ruolo, scegli AWS Service (Servizio AWS), trova e seleziona Glue e quindi Next: Permissions (Successivo: autorizzazioni).
5. Nella pagina Attach permissions policy (Allega policy di autorizzazioni), scegli le policy che contengono le autorizzazioni necessarie, ad esempio la policy `AWSGlueServiceRole` gestita da AWS per le autorizzazioni AWS Glue generali e la policy gestita da AWS

AmazonS3FullAccess per l'accesso alle risorse Amazon S3. Scegli quindi Next: Review (Avanti: Verifica).

> **Note**
>
> Verifica che una delle policy in questo ruolo conceda le autorizzazioni per le origini e le destinazioni Amazon S3. Puoi fornire una policy personalizzata per l'accesso a risorse Amazon S3 specifiche. Le origini dati richiedono le autorizzazioni `s3:ListBucket` e `s3:GetObject`. Le destinazioni dati richiedono le autorizzazioni `s3:ListBucket`, `s3:PutObject` e `s3:DeleteObject`. Per ulteriori informazioni sulla creazione di una policy Amazon S3 per le risorse, vedi Specificare le risorse in una policy. Per un esempio di policy Amazon S3, consulta la pagina relativa alla scrittura di policy IAM per concedere l'accesso a un bucket Amazon S3.
> Se prevedi di accedere a origini e destinazioni Amazon S3 crittografate con SSE-KMS, collega una policy che permetta a crawler, processi ed endpoint di sviluppo AWS Glue di decrittografare i dati. Per ulteriori informazioni, vedi Protezione dei dati mediante la crittografia lato server con chiavi gestite da AWS KMS (SSE-KMS).
> Di seguito è riportato un esempio.
>
> ```
> {
>    "Version":"2012-10-17",
>    "Statement":[
>       {
>          "Effect":"Allow",
>          "Action":[
>             "kms:Decrypt"
>          ],
>          "Resource":[
>             "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
>          ]
>       }
>    ]
> }
> ```

6. Per Role name (Nome ruolo), digita un nome per il ruolo, ad esempio `AWSGlueServiceRoleDefault`. Crea il ruolo usando come prefisso del nome la stringa `AWSGlueServiceRole` per permettere il passaggio del ruolo dagli utenti della console al servizio. Le policy fornite da AWS Glue prevedono ruoli di servizio IAM che iniziano con `AWSGlueServiceRole`. In caso contrario, è necessario aggiungere una policy per concedere

agli utenti l'autorizzazione `iam:PassRole` per i ruoli IAM in modo da soddisfare la convenzione per la denominazione. Selezionare Create Role (Crea ruolo).

> **Note**
>
> Quando crei un notebook con un ruolo, tale ruolo viene passato alle sessioni interattive in modo che lo stesso ruolo possa essere utilizzato in entrambe le posizioni. Come tale, il permesso `iam:PassRole` deve essere parte della policy del ruolo.
> Crea una nuova policy per il tuo ruolo utilizzando l'esempio seguente. Sostituisci il numero di account con il tuo e il nome del ruolo.
>
> ```
> {
>     "Version": "2012-10-17",
>     "Statement": [
>         {
>             "Effect": "Allow",
>             "Action": "iam:PassRole",
>             "Resource": "arn:aws:iam::090000000210:role/<role_name>"
>         }
>     ]
> }
> ```

Fase 3: Collegamento di una policy agli utenti o ai gruppi che accedono a AWS Glue

L'amministratore deve assegnare le autorizzazioni a qualsiasi utente, gruppo o ruolo utilizzando la console AWS Glue o la AWS Command Line Interface (AWS CLI). Puoi fornire queste autorizzazioni usando AWS Identity and Access Management (IAM), tramite le policy. Questa fase descrive l'assegnazione di autorizzazioni a utenti o gruppi.

Una volta completata questa fase, all'utente o al gruppo sono collegate le policy seguenti:

- La policy gestita da AWS `AWSGlueConsoleFullAccess` o la policy personalizzata GlueConsoleAccessPolicy
- **`AWSGlueConsoleSageMakerNotebookFullAccess`**
- **`CloudWatchLogsReadOnlyAccess`**
- **`AWSCloudFormationReadOnlyAccess`**

- **`AmazonAthenaFullAccess`**

Per collegare una policy inline e incorporarla in un utente o in un gruppo

Puoi collegare una policy gestita da AWS o una policy inline a un utente o a un gruppo per accedere alla console AWS Glue. Alcune delle risorse specificate nella policy fanno riferimento a nomi predefiniti usati da AWS Glue per i bucket Amazon S3, gli script ETL Amazon S3, le voci di CloudWatch Logs, AWS CloudFormation e le risorse Amazon EC2. Per semplicità, AWS Glue scrive alcuni oggetti Amazon S3 nei bucket nell'account con il prefisso `aws-glue-*` per impostazione predefinita.

> **Note**
>
> Puoi ignorare questa fase se usi la policy gestita da AWS **`AWSGlueConsoleFullAccess`**.

> **Important**
>
> AWS Glue richiede l'autorizzazione per assumere un ruolo usato per eseguire il lavoro per tuo conto. A tale scopo, aggiungi le autorizzazioni **`iam:PassRole`** agli utenti o ai gruppi AWS Glue. Questa policy concede l'autorizzazione ai ruoli che iniziano con `AWSGlueServiceRole` per i ruoli di servizio AWS Glue e `AWSGlueServiceNotebookRole` per i ruoli necessari al momento della creazione di un server notebook. Puoi anche creare una policy per le autorizzazioni `iam:PassRole` che segue la convenzione per la denominazione.
> In base alle best practice di sicurezza, si consiglia di limitare l'accesso rafforzando le policy per limitare ulteriormente l'accesso al bucket Amazon S3 e ai gruppi di log Amazon CloudWatch. Per un esempio di policy Amazon S3, consulta la pagina relativa alla scrittura di policy IAM per concedere l'accesso a un bucket Amazon S3.

In questa fase, crei una policy simile a `AWSGlueConsoleFullAccess`. Puoi trovare la versione più recente di `AWSGlueConsoleFullAccess` nella console IAM.

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, scegli Utenti o Gruppi di utenti.

3. Nell'elenco, scegli il nome dell'utente o del gruppo in cui integrare una policy.
4. Scegliere la scheda Permissions (Autorizzazioni) e, se necessario, espandere la sezione Permissions policies (Policy autorizzazioni).
5. Scegli il collegamento Add Inline policy (Aggiungi policy inline).
6. Nella schermata Create Policy (Crea policy), passa alla scheda per modificare JSON. Crea un documento di policy con le seguenti istruzioni JSON, quindi scegli Review policy (Verifica policy).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "redshift:DescribeClusters",
                "redshift:DescribeClusterSubnetGroups",
                "iam:ListRoles",
                "iam:ListUsers",
                "iam:ListGroups",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListAttachedRolePolicies",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeRouteTables",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeInstances",
                "rds:DescribeDBInstances",
                "rds:DescribeDBClusters",
                "rds:DescribeDBSubnetGroups",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "cloudformation:DescribeStacks",
                "cloudformation:GetTemplateSummary",
                "dynamodb:ListTables",
                "kms:ListAliases",
```

```
                "kms:DescribeKey",
                "cloudwatch:GetMetricData",
                "cloudwatch:ListDashboards"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*/*aws-glue-*/*",
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPublicAccessBlock"           ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:/aws-glue/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack"
            ],
            "Resource": "arn:aws:cloudformation:*:*:stack/aws-glue*/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:image/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:volume/*"
            ]
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
```

```
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceNotebookRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/service-role/AWSGlueServiceRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

La tabella seguente descrive le autorizzazioni concesse dalla policy.

Action	Resource (Risorsa)	Descrizione
`"glue:*"`	`"*"`	Concede l'autorizzazione per eseguire tutte le operazioni API AWS Glue. Se in precedenza la policy è stata creata senza l'operazione `"glue:*"`, è necessario aggiungere le seguenti autorizzazioni individuali alla policy: • "glue:ListCrawlers" • "glue:BatchGetCrawlers" • "glue:ListTriggers" • "glue:BatchGetTriggers" • "glue:ListDevEndpoints" • "glue:BatchGetDevEndpoints" • "glue:ListJobs" • "glue:BatchGetJobs"
`"redshift:DescribeClusters", "redshift:DescribeClusterSubnetGroups"`	`"*"`	Permette la creazione di connessioni ad Amazon Redshift.

Action	Resource (Risorsa)	Descrizione
`"iam:ListRoles", "iam:ListRolePolicies", "iam:GetRole", "iam:GetRolePolicy", "iam:ListAttachedRolePolicies"`	`"*"`	Consente l'elenco dei ruoli IAM quando si utilizzano crawler, processi, endpoint di sviluppo e server notebook.
`"ec2:DescribeSecurityGroups", "ec2:DescribeSubnets", "ec2:DescribeVpcs", "ec2:DescribeVpcEndpoints", "ec2:DescribeRouteTables", "ec2:DescribeVpcAttribute", "ec2:DescribeKeyPairs", "ec2:DescribeInstances"`	`"*"`	Permette la configurazione degli elementi di rete Amazon EC2, ad esempio i VPC, quando vengono eseguiti processi, crawler ed endpoint di sviluppo.
`"rds:DescribeDBInstances"`	`"*"`	Consente la creazione di connessioni ad Amazon RDS.
`"s3:ListAllMyBuckets", "s3:ListBucket", "s3:GetBucketAcl", "s3:GetBucketLocation"`	`"*"`	Permette di elencare i bucket Amazon S3 quando vengono usati crawler, processi, endpoint di sviluppo e server notebook.
`"dynamodb:ListTables"`	`"*"`	Permette di elencare tabelle DynamoDB.
`"kms:ListAliases", "kms:DescribeKey"`	`"*"`	Permette di usare le chiavi KMS.

Action	Resource (Risorsa)	Descrizione
`"cloudwatch:GetMetricData", "cloudwatch:ListDashboards"`	`"*"`	Permette di usare i parametri CloudWatch.
`"s3:GetObject", "s3:PutObject"`	`"arn:aws:s3::: aws-glue-*/*", "arn:aws:s3::: */*aws-glue-*/*", "arn:aws:s3::: aws-glue-*"`	Permette di ottenere e inserire oggetti Amazon S3 nell'account quando vengono archiviati oggetti come script ETL e posizioni dei server notebook. Convenzione per la denominazione: concede l'autorizzazione alle cartelle o ai bucket Amazon S3 i cui nomi hanno il prefisso aws-glue-.
`"tag:GetResources"`	`"*"`	Consente il recupero di tag AWS.

Action	Resource (Risorsa)	Descrizione
`"s3:CreateBucket", "s3:PutBucketPublicAccessBlock"`	`"arn:aws:s3::: aws-glue-*"`	Permette di creare un bucket Amazon S3 nell'account quando vengono archiviati oggetti come script ETL e posizioni dei server notebook. Convenzione per la denominazione: concede l'autorizzazione alle cartelle o ai bucket Amazon S3 i cui nomi hanno il prefisso aws-glue-. Consente ad AWS Glue di creare i bucket che bloccano l'accesso pubblico.
`"logs:GetLogEvents"`	`"arn:aws:logs:*:*: /aws-glue/*"`	Consente il recupero di CloudWatch Logs. Convenzione di denominazione: AWS Glue scrive i log in gruppi di log i cui nomi iniziano con aws-glue-.

Action	Resource (Risorsa)	Descrizione
`"cloudformation:CreateStack", "cloudformation:DeleteStack"`	`"arn:aws:cloudformation:*:*:stack/aws-glue*/*"`	Permette la gestione di stack AWS CloudFormation quando vengono usati server notebook. Convenzione di denominazione: AWS Glue crea stack i cui nomi iniziano con aws-glue.
`"ec2:RunInstances"`	`"arn:aws:ec2:*:*:instance/*", "arn:aws:ec2:*:*:key-pair/*", "arn:aws:ec2:*:*:image/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:subnet/*", "arn:aws:ec2:*:*:volume/*"`	Consente l'esecuzione di endpoint di sviluppo e server notebook.
`"iam:PassRole"`	`"arn:aws:iam::*:role/ AWSGlueServiceRole*"`	Permette a AWS Glue di usare l'autorizzazione `PassRole` per i ruoli che iniziano con `AWSGlueServiceRole` .

Action	Resource (Risorsa)	Descrizione
`"iam:PassRole"`	`"arn:aws:iam::*:role/ AWSGlueServiceNotebookRole*"`	Permette ad Amazon EC2 di usare l'autorizzazione `PassRole` per i ruoli che iniziano con `AWSGlueServiceNotebookRole` .
`"iam:PassRole"`	`"arn:aws:iam::*:role/service-role/AWSGlueServiceRole*"`	Permette a AWS Glue di usare l'autorizzazione `PassRole` per i ruoli che iniziano con `service-role/AWSGlueServiceRole` .

7. Nella schermata Review policy (Esamina policy), immettere il nome della policy, ad esempio GlueConsoleAccessPolicy. Al termine, scegliere Create policy (Crea policy). Assicurati che non siano presenti errori in una casella rossa nella parte superiore dello schermo. Correggi gli eventuali errori segnalati.

> **Note**
>
> Se Use autoformatting (Usa formattazione automatica) è selezionato, la policy viene riformattata ogni volta che pari una policy oppure scegli Validate Policy (Convalida policy).

Per collegare la policy gestita AWSGlueConsoleFullAccess

Puoi collegare la policy `AWSGlueConsoleFullAccess` per fornire le autorizzazioni necessarie all'utente della console AWS Glue.

> **Note**
>
> Puoi ignorare questa fase se hai creato una policy personalizzata per l'accesso alla console AWS Glue.

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Policies (Policy).
3. Nell'elenco delle policy, seleziona la casella di controllo accanto a AWSGlueConsoleFullAccess. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco di policy.
4. Scegli Policy actions (Operazioni di policy), quindi Attach (Collega).
5. Sceglie l'utente a cui collegare la policy. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco delle entità principali. Dopo aver scelto l'utente a cui collegare la policy, scegli Attach policy (Collega policy).

Per collegare la policy gestita **`AWSGlueConsoleSageMakerNotebookFullAccess`**.

Puoi collegare la policy `AWSGlueConsoleSageMakerNotebookFullAccess` a un utente per gestire i notebook SageMaker creati nella console AWS Glue. Oltre alle altre richieste di autorizzazioni della console AWS Glue, questa policy concede l'accesso alle risorse necessarie per gestire i notebook SageMaker .

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Policies (Policy).
3. Nell'elenco delle policy, seleziona la casella di controllo accanto ad AWSGlueConsoleSageMakerNotebookFullAccess. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco di policy.
4. Scegli Policy actions (Operazioni di policy), quindi Attach (Collega).
5. Sceglie l'utente a cui collegare la policy. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco delle entità principali. Dopo aver scelto l'utente a cui collegare la policy, scegli Attach policy (Collega policy).

Per collegare la policy gestita CloudWatchLogsReadOnlyAccess

Puoi collegare la policy CloudWatchLogsReadOnlyAccess a un utente per visualizzare i log creati da AWS Glue nella console CloudWatch Logs.

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Policies (Policy).
3. Nell'elenco delle policy, seleziona la casella di controllo accanto a CloudWatchLogsReadOnlyAccess. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco di policy.
4. Scegli Policy actions (Operazioni di policy), quindi Attach (Collega).
5. Sceglie l'utente a cui collegare la policy. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco delle entità principali. Dopo aver scelto l'utente a cui collegare la policy, scegli Attach policy (Collega policy).

Per collegare la policy gestita AWSCloudFormationReadOnlyAccess

Puoi collegare la policy AWSCloudFormationReadOnlyAccess a un utente per visualizzare gli stack AWS CloudFormation usati da AWS Glue nella console AWS CloudFormation.

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Policies (Policy).
3. Nell'elenco delle policy, seleziona la casella di controllo accanto a AWSCloudFormationReadOnlyAccess. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco di policy.
4. Scegli Policy actions (Operazioni di policy), quindi Attach (Collega).
5. Sceglie l'utente a cui collegare la policy. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco delle entità principali. Dopo aver scelto l'utente a cui collegare la policy, scegli Attach policy (Collega policy).

Per collegare la policy gestita AmazonAthenaFullAccess

Puoi collegare la policy AmazonAthenaFullAccess a un utente per visualizzare i dati Amazon S3 nella console Athena.

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Policies (Policy).
3. Nell'elenco delle policy, seleziona la casella di controllo accanto a AmazonAthenaFullAccess. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco di policy.
4. Scegli Policy actions (Operazioni di policy), quindi Attach (Collega).
5. Sceglie l'utente a cui collegare la policy. Puoi usare il menu Filter (Filtro) e la casella di ricerca per filtrare l'elenco delle entità principali. Dopo aver scelto l'utente a cui collegare la policy, scegli Attach policy (Collega policy).

Fase 4: creare una policy IAM per i server notebook

Se prevedi di utilizzare i notebook con gli endpoint di sviluppo, devi specificare le autorizzazioni quando crei il server notebook. È possibile fornire queste autorizzazioni usando AWS Identity and Access Management (IAM).

Questa policy concede l'autorizzazione per alcune operazioni Amazon S3 per gestire le risorse nell'account richieste da AWS Glue quando assume il ruolo usando la policy. Alcune delle risorse specificate nella policy fanno riferimento a nomi predefiniti usati da AWS Glue per i bucket Amazon S3, gli script ETL Amazon S3 e le risorse Amazon EC2. Per semplicità, AWS Glue scrive alcuni oggetti Amazon S3 nei bucket nell'account con il prefisso `aws-glue-*` per impostazione predefinita.

> **Note**
>
> Puoi ignorare questa fase se usi la policy gestita da AWS **`AWSGlueServiceNotebookRole`**.

In questa fase, crei una policy simile a `AWSGlueServiceNotebookRole`. Puoi trovare la versione più recente di `AWSGlueServiceNotebookRole` nella console IAM.

Per creare una policy IAM per i notebook

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra seleziona Policies (Policy).
3. Scegliere Create Policy (Crea policy).

4. Nella schermata Create Policy (Crea policy), passa alla scheda per modificare JSON. Crea un documento di policy con le seguenti istruzioni JSON, quindi scegli Review policy (Verifica policy).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "glue:CreateDatabase",
            "glue:CreatePartition",
            "glue:CreateTable",
            "glue:DeleteDatabase",
            "glue:DeletePartition",
            "glue:DeleteTable",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:GetTable",
            "glue:GetTableVersions",
            "glue:GetTables",
            "glue:UpdateDatabase",
            "glue:UpdatePartition",
            "glue:UpdateTable",
            "glue:GetJobBookmark",
            "glue:ResetJobBookmark",
            "glue:CreateConnection",
            "glue:CreateJob",
            "glue:DeleteConnection",
            "glue:DeleteJob",
            "glue:GetConnection",
            "glue:GetConnections",
            "glue:GetDevEndpoint",
            "glue:GetDevEndpoints",
            "glue:GetJob",
            "glue:GetJobs",
            "glue:UpdateJob",
            "glue:BatchDeleteConnection",
            "glue:UpdateConnection",
            "glue:GetUserDefinedFunction",
            "glue:UpdateUserDefinedFunction",
            "glue:GetUserDefinedFunctions",
```

```
            "glue:DeleteUserDefinedFunction",
            "glue:CreateUserDefinedFunction",
            "glue:BatchGetPartition",
            "glue:BatchDeletePartition",
            "glue:BatchCreatePartition",
            "glue:BatchDeleteTable",
            "glue:UpdateDevEndpoint",
            "s3:GetBucketLocation",
            "s3:ListBucket",
            "s3:ListAllMyBuckets",
            "s3:GetBucketAcl"
         ],
         "Resource":[
            "*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::crawler-public*",
            "arn:aws:s3:::aws-glue*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:DeleteObject"
         ],
         "Resource":[
            "arn:aws:s3:::aws-glue*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:CreateTags",
            "ec2:DeleteTags"
         ],
         "Condition":{
            "ForAllValues:StringEquals":{
```

```
                "aws:TagKeys":[
                   "aws-glue-service-resource"
                ]
             }
          },
          "Resource":[
             "arn:aws:ec2:*:*:network-interface/*",
             "arn:aws:ec2:*:*:security-group/*",
             "arn:aws:ec2:*:*:instance/*"
          ]
       }
    ]
}
```

La tabella seguente descrive le autorizzazioni concesse dalla policy.

Action	Resource (Risorsa)	Descrizione
`"glue:*"`	`"*"`	Concede l'autorizzazione per eseguire tutte le operazioni API AWS Glue.
`"s3:GetBucketLocation", "s3:ListBucket", "s3:ListAllMyBuckets", "s3:GetBucketAcl"`	`"*"`	Permette di elencare i bucket Amazon S3 dai server notebook.
`"s3:GetObject"`	`"arn:aws:s3:::crawler-public*", "arn:aws:s3:::aws-glue-*"`	Permette di ottenere gli oggetti Amazon S3 usati da esempi e tutorial dai notebook. Convenzione per la denominazione: i nomi di bucket Amazon S3 iniziano con crawler-public e aws-glue-.

Action	Resource (Risorsa)	Descrizione
`"s3:PutObject", "s3:DeleteObject"`	`"arn:aws:s3:::aws-glue*"`	Permette di inserire ed eliminare oggetti Amazon S3 nell'account dai notebook. Convenzione per la denominazione: utilizza cartelle Amazon S3 denominate aws-glue.
`"ec2:CreateTags", "ec2:DeleteTags"`	`"arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:instance/*"`	Consente il tagging delle risorse Amazon EC2 create per i server notebook. Convenzione di denominazione: AWS Glue applica i tag alle istanze Amazon EC2 con aws-glue-service-resource.

5. Nella schermata Review Policy (Verifica policy), immettere il Policy Name (Nome policy), ad esempio GlueServiceNotebookPolicyDefault. Digita una descrizione facoltativa e, al termine, seleziona Create policy (Crea policy).

Fase 5: creare un ruolo IAM per i server notebook

Se prevedi di utilizzare i notebook con gli endpoint di sviluppo, devi concedere le autorizzazioni per il ruolo IAM. Puoi fornire queste autorizzazioni utilizzando AWS Identity and Access Management IAM, tramite un ruolo IAM.

> **Note**
>
> Quando crei un ruolo IAM utilizzando la console IAM, la console crea automaticamente un profilo dell'istanza e le assegna lo stesso nome del ruolo a cui corrisponde.

Per creare un ruolo IAM per i notebook

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra seleziona Ruoli.
3. Scegliere Crea ruolo.
4. Per tipo di ruolo, scegli AWS Service (Servizio AWS), trova e seleziona EC2, il caso d'uso EC2 e quindi Next: Permissions (Successivo: autorizzazioni).
5. Nella pagina Attach permissions policy (Allega policy di autorizzazioni) scegli le policy che contengono le autorizzazioni necessarie, ad esempio AWSGlueServiceNotebookRole per le autorizzazioni AWS Glue generali e la policy AmazonS3FullAccess gestita da AWS per l'accesso alle risorse Amazon S3. Scegli quindi Next: Review (Avanti: Verifica).

> **Note**
>
> Verifica che una delle policy in questo ruolo conceda le autorizzazioni per le origini e le destinazioni Amazon S3. Conferma che le tue policy concedano l'accesso completo al percorso in cui archivi i notebook al momento della creazione di un server notebook. Puoi fornire una policy personalizzata per l'accesso a risorse Amazon S3 specifiche. Per ulteriori informazioni sulla creazione di una policy Amazon S3 per le risorse, vedi Specificare le risorse in una policy.
> Se prevedi di accedere a origini e destinazioni Amazon S3 crittografate con SSE-KMS, collega una policy che permetta ai notebook di decrittografare i dati. Per ulteriori informazioni, vedi Protezione dei dati mediante la crittografia lato server con chiavi gestite da AWS KMS (SSE-KMS).
> Di seguito è riportato un esempio.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
         ]
```

```
        }
    ]
}
```

6. In Role name (Nome ruolo), digita un nome per il ruolo. Crea il ruolo usando come prefisso del nome la stringa `AWSGlueServiceNotebookRole` per permettere il passaggio del ruolo dagli utenti della console al server notebook. Le policy fornite da AWS Glue prevedono ruoli di servizio IAM che iniziano con `AWSGlueServiceNotebookRole`. In caso contrario, è necessario aggiungere una policy per concedere agli utenti l'autorizzazione `iam:PassRole` per i ruoli IAM in modo da soddisfare la convenzione per la denominazione. Ad esempio, specifica `AWSGlueServiceNotebookRoleDefault`. Quindi seleziona Create role (Crea ruolo).

Fase 6: creare una policy IAM per i notebook SageMaker

Se prevedi di utilizzare i notebook SageMaker con gli endpoint di sviluppo, devi specificare le autorizzazioni quando crei il server notebook. È possibile fornire queste autorizzazioni usando AWS Identity and Access Management (IAM).

Per creare una policy IAM per i notebook SageMaker

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra seleziona Policies (Policy).
3. Scegliere Create Policy (Crea policy).
4. Nella pagina Create Policy (Crea policy), passa alla scheda per modificare JSON. Crea il tuo documento di policy con le istruzioni JSON seguenti. Modificare il *`nome bucket`*, il *`codice regionale`* e l'*`ID account`* per l'ambiente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucket-name"
```

```
            ]
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucket-name*"
            ]
        },
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:logs:region-code:account-id:log-group:/aws/sagemaker/*",
                "arn:aws:logs:region-code:account-id:log-group:/aws/sagemaker/
*:log-stream:aws-glue-*"
            ]
        },
        {
            "Action": [
                "glue:UpdateDevEndpoint",
                "glue:GetDevEndpoint",
                "glue:GetDevEndpoints"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:glue:region-code:account-id:devEndpoint/*"
            ]
        },
        {
            "Action": [
                "sagemaker:ListTags"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:sagemaker:region-code:account-id:notebook-instance/*"
            ]
```

```
        }
    ]
}
```

Selezionare Review policy (Esamina policy).

La tabella seguente descrive le autorizzazioni concesse dalla policy.

Action	Resource (Risorsa)	Descrizione
`"s3:ListBucket*"`	`"arn:aws:s3::: bucket-name "`	Concede l'autorizzazione per elencare i bucket Amazon S3.
`"s3:GetObject"`	`"arn:aws:s3::: bucket-name *"`	Concede l'autorizzazione per ottenere gli oggetti Amazon S3 utilizzati dai notebook SageMaker.
`"logs:CreateLogStream", "logs:DescribeLogStreams", "logs:PutLogEvents", "logs:CreateLogGroup"`	`"arn:aws:logs: region-code :account-id :log-group:/aws/sagemaker/*", "arn:aws:logs: region-code :account-id :log-group:/aws/sagemaker/*:log-stream:aws-glue-*"`	Concede l'autorizzazione per la scrittura di log su Amazon CloudWatch Logs dai notebook. Convenzione per la denominazione: scrive per registrare i gruppi i cui nomi iniziano con aws-glue.
`"glue:UpdateDevEndpoint", "glue:GetDevEndpoint", "glue:GetDevEndpoints"`	`"arn:aws:glue: region-code :account-id :devEndpoint/*"`	Autorizza l'utente a utilizzare un endpoint di sviluppo dai notebook SageMaker.

Action	Resource (Risorsa)	Descrizione
`"sagemaker:ListTags"`	`"arn:aws:sagemaker:region-code:account-id:notebook-instance/*"`	Concede l'autorizzazione per restituire i tag per una risorsa SageMaker. Il tag `aws-glue-dev-endpoint` è necessario sul notebook SageMaker per connettere il notebook a un endpoint di sviluppo.

5. Nella schermata Review policy (Verifica policy), inserisci il Policy Name (Nome policy), ad esempio `AWSGlueSageMakerNotebook`. Digita una descrizione facoltativa e, al termine, seleziona Create policy (Crea policy).

Fase 7: creare un ruolo IAM per inotebook SageMaker

Se prevedi di utilizzare i notebook SageMaker con gli endpoint di sviluppo, devi concedere le autorizzazioni per il ruolo IAM. Puoi fornire queste autorizzazioni utilizzando AWS Identity and Access Management (IAM), tramite un ruolo IAM.

Per creare un ruolo IAM per i notebook SageMaker

1. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra seleziona Ruoli.
3. Scegliere Crea ruolo.
4. Per tipo di ruolo, scegli AWS Service (Servizio AWS), trova e seleziona SageMaker, quindi il caso d'uso SageMaker - Execution (SageMaker - Esecuzione). Quindi scegliere Next: Permissions (Successivo: Autorizzazioni).
5. Nella pagina Attach permissions policy (Collega policy di autorizzazioni), scegli le policy che contengono le autorizzazioni necessarie, ad esempio AmazonSageMakerFullAccess. Seleziona Next: Review (Successivo: Rivedi).

   Se prevedi di accedere a origini e destinazioni Amazon S3 crittografate con SSE-KMS, collega una policy che permetta ai notebook di decrittografare i dati, come mostrato nell'esempio

seguente. Per ulteriori informazioni, vedi Protezione dei dati mediante la crittografia lato server con chiavi gestite da AWS KMS (SSE-KMS).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
         ]
      }
   ]
}
```

6. In Role name (Nome ruolo), digita un nome per il ruolo. Per permettere il passaggio del ruolo dagli utenti della console a SageMaker, utilizza un nome che abbia come prefisso la stringa `AWSGlueServiceSageMakerNotebookRole`. Le policy fornite da AWS Glue prevedono ruoli IAM che iniziano con `AWSGlueServiceSageMakerNotebookRole`. In caso contrario, è necessario aggiungere una policy per concedere agli utenti l'autorizzazione `iam:PassRole` per i ruoli IAM in modo da soddisfare la convenzione per la denominazione.

   Per esempio, inserisci `AWSGlueServiceSageMakerNotebookRole-Default` e quindi seleziona Create role (Crea ruolo).

7. Dopo aver creato il ruolo, collega una policy che abiliti autorizzazioni aggiuntive necessarie per creare notebook SageMaker da AWS Glue.

   Apto il ruolo appena creato, `AWSGlueServiceSageMakerNotebookRole-Default` e scegli Attach policies (Collega policy). Collega la policy creata denominata `AWSGlueSageMakerNotebook` al ruolo.

Esempi di policy di controllo degli accessi di AWS Glue

Questa sezione contiene esempi di policy di controllo degli accessi basate sull'identità (IAM) e sulle risorse di AWS Glue.

Indice

Esempi di policy basate su identità per AWS Glue

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse AWS Glue. Inoltre, non sono in grado di eseguire attività utilizzando la AWS Management Console, l'AWS Command Line Interface (AWS CLI) o l'API AWS. Per concedere agli utenti l'autorizzazione per eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Per informazioni dettagliate sulle operazioni e sui tipi di risorse definiti da AWS Glue, incluso il formato degli ARN per ogni tipo di risorsa, consulta Operazioni, risorse e chiavi di condizione per AWS Glue nella documentazione di riferimento per l'autorizzazione del servizio.

> **Note**
>
> Gli esempi forniti in questa sezione utilizzano tutti la Regione `us-west-2`. Puoi sostituirla con la Regione AWS che desideri utilizzare.

Argomenti

- Best practice per le policy
- Le autorizzazioni a livello di risorsa si applicano solo agli specifici oggetti di AWS Glue
- Utilizzo della console di AWS Glue
- Consentire agli utenti di visualizzare le loro autorizzazioni
- Concessione di autorizzazioni di sola lettura a una tabella
- Filtraggio delle tabelle mediante l'autorizzazione GetTables
- Concessione di accesso completo a una tabella e a tutte le partizioni
- Controllo degli accessi per nome prefisso e diniego esplicito
- Autorizzazione dell'accesso utilizzando tag
- Negazione dell'accesso utilizzando tag
- Utilizzo di tag con operazioni API in elenco e in batch
- Controllo delle impostazioni utilizzando le chiavi di condizione o di contesto
- Negare a un'identità la possibilità di creare sessioni di anteprima dei dati

Best practice per le policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse AWS Glue nel tuo account. Queste operazioni possono comportare costi aggiuntivi per l'Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Nozioni di base sulle policy gestite da AWSe passaggio alle autorizzazioni con privilegio minimo: per le informazioni di base su come concedere autorizzazioni a utenti e carichi di lavoro, utilizza le policy gestite da AWSche concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili

nel tuo Account AWS. Ti consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente di AWSspecifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente IAM.

- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Puoi farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo 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 a operazioni e risorse puoi aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi inoltre utilizzare le condizioni per concedere l'accesso alle operazioni di servizio, ma solo se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente di IAM.
- Richiesta dell'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o utenti root nel tuo Account AWS, attiva MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori informazioni, consulta Configurazione dell'accesso alle API protetto 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.

Le autorizzazioni a livello di risorsa si applicano solo agli specifici oggetti di AWS Glue

È possibile definire il controllo granulare solo per specifici oggetti di AWS Glue. Pertanto è necessario scrivere la policy IAM del client in modo che le operazioni delle API che consentono l'utilizzo dei nomi della risorsa Amazon (ARN) per l'istruzione `Resource` non si mescolino con operazioni delle API che non consentono l'uso degli ARN.

Ad esempio, la seguente policy di IAM consente operazioni delle API per `GetClassifier` e `GetJobRun`. Definisce il valore `Resource` come * perché AWS Glue non consente l'utilizzo degli ARN per le esecuzioni di classificatori e processi. Poiché gli ARN sono abilitati per operazioni API specifiche, ad esempio `GetDatabase` e `GetTable`, gli ARN possono essere specificati nella seconda metà della policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetClassifier*",
                "glue:GetJobRun*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:Get*"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:123456789012:catalog",
                "arn:aws:glue:us-east-1:123456789012:database/default",
                "arn:aws:glue:us-east-1:123456789012:table/default/e*1*",
                "arn:aws:glue:us-east-1:123456789012:connection/connection2"
            ]
        }
    ]
}
```

Per un elenco di oggetti di AWS Glue che consentono l'utilizzo degli ARN, consultare ARN delle risorse

Utilizzo della console di AWS Glue

Per accedere alla console di AWS Glue è necessario disporre di un insieme di autorizzazioni minimo. Queste autorizzazioni devono consentire di elencare e visualizzare i dettagli relativi alle risorse AWS Glue nel tuo Account AWS. Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario concedere le autorizzazioni minime della console agli utenti che effettuano chiamate solo alla AWS CLI o all'API AWS. Al contrario, concedi l'accesso solo alle operazioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che gli utenti e i ruoli possano continuare a utilizzare la console di AWS Glue, collega alle entità anche la policy gestita AWS *ConsoleAccess* o *ReadOnly* di AWS Glue. Per ulteriori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente IAM.

Per poter usare la console di AWS Glue, un utente deve disporre di un set minimo di autorizzazioni che gli permetta di usare le risorse di AWS Glue per l'account AWS. Oltre a queste autorizzazioni AWS Glue, la console richiede le autorizzazioni dei servizi seguenti:

- Autorizzazioni Amazon CloudWatch Logs per visualizzare i log.
- Autorizzazioni AWS Identity and Access Management (IAM) per elencare e passare i ruoli.
- Autorizzazioni AWS CloudFormation per usare le pile.
- Autorizzazioni Amazon Elastic Compute Cloud (Amazon EC2) per elencare VPC, sottoreti, gruppi di sicurezza, istanze e altri oggetti.
- Autorizzazioni Amazon Simple Storage Service (Amazon S3) per elencare bucket e oggetti e per recuperare e salvare script.
- Autorizzazioni Amazon Redshift necessarie per l'utilizzo dei cluster.
- Autorizzazioni Amazon Relational Database Service (Amazon RDS) per elencare le istanze.

Per ulteriori informazioni sulle autorizzazioni necessarie agli utenti per visualizzare e usare la console di AWS Glue, consultare Fase 3: Collegamento di una policy agli utenti o ai gruppi che accedono a AWS Glue.

Se decidi di creare una policy IAM più restrittiva delle autorizzazioni minime richieste, la console non funzionerà come previsto per gli utenti con tale policy IAM. Per garantire che gli utenti possano continuare a usare la console AWS Glue, collega anche la policy gestita `AWSGlueConsoleFullAccess`, come descritto in Policy gestite da AWS (predefinite) per AWS Glue.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono allegate alla relativa identità utente. La policy include le

autorizzazioni per completare questa azione sulla console o a livello di programmazione utilizzando la AWS CLIo l'API AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Concessione di autorizzazioni di sola lettura a una tabella

La policy seguente consente di concedere autorizzazioni di sola lettura per una tabella `books` nel database `db1`. Per ulteriori informazioni sull'utilizzo degli Amazon Resource Names (ARN), consultare ARN del catalogo dati

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetTablesActionOnBooks",
      "Effect": "Allow",
      "Action": [
        "glue:GetTables",
        "glue:GetTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:123456789012:catalog",
        "arn:aws:glue:us-west-2:123456789012:database/db1",
        "arn:aws:glue:us-west-2:123456789012:table/db1/books"
      ]
    }
  ]
}
```

Questa policy consente di concedere autorizzazioni di sola lettura per una tabella `books` nel database denominato `db1`. Per concedere l'autorizzazione `Get` a una tabella è richiesta anche l'autorizzazione alle risorse del database e del catalogo.

La policy seguente concede il livello minimo di autorizzazioni necessarie per creare una tabella `tbl1` nel database `db1`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:123456789012:table/db1/tbl1",
        "arn:aws:glue:us-west-2:123456789012:database/db1",
        "arn:aws:glue:us-west-2:123456789012:catalog"
      ]
    }
  ]
}
```

Filtraggio delle tabelle mediante l'autorizzazione GetTables

Supponiamo che ci siano tre tabelle (`customers`, `stores` e `store_sales`) nel database `db1`. La policy seguente concede l'autorizzazione `GetTables` a `stores` e `store_sales`, ma non a `customers`. Quando chiami `GetTables` con questa policy, il risultato contiene solo le due tabelle autorizzate (la tabella `customers` non viene restituita).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesExample",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/store_sales",
                "arn:aws:glue:us-west-2:123456789012:table/db1/stores"
            ]
        }
    ]
}
```

È possibile semplificare la policy precedente utilizzando `store*` per includere qualsiasi nome di tabella che inizi con `store`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesExample2",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/store*"
```

```
            ]
        }
    ]
}
```

Analogamente, utilizzando `/db1/*` per includere tutte le tabelle incluse nella cartella `db1`, la policy seguente concede l'autorizzazione `GetTables` a tutte le tabelle presenti in `db1`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesReturnAll",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/*"
            ]
        }
    ]
}
```

Se non viene fornito nessun ARN di tabella, una chiamata a `GetTables` si conclude correttamente ma restituisce un elenco vuoto:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesEmptyResults",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1"
            ]
```

```
        }
    ]
}
```

Se la policy non contiene l'ARN del database, una chiamata a `GetTables` ha esito negativo e restituisce `AccessDeniedException`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesAccessDeny",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:table/db1/*"
            ]
        }
    ]
}
```

Concessione di accesso completo a una tabella e a tutte le partizioni

La policy seguente concede tutte le autorizzazioni su una tabella denominata `books` nel database db1. Questo include le autorizzazioni di lettura e scrittura sulla tabella stessa, sulle versioni archiviate e su tutte le partizioni.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessOnTable",
            "Effect": "Allow",
            "Action": [
                "glue:CreateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:UpdateTable",
                "glue:DeleteTable",
```

```
                "glue:BatchDeleteTable",
                "glue:GetTableVersion",
                "glue:GetTableVersions",
                "glue:DeleteTableVersion",
                "glue:BatchDeleteTableVersion",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:UpdatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                        "arn:aws:glue:us-west-2:123456789012:catalog",
                        "arn:aws:glue:us-west-2:123456789012:database/db1",
                        "arn:aws:glue:us-west-2:123456789012:table/db1/books"
                        ]
        }
    ]
}
```

Nella pratica, la policy precedente può essere semplificata:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessOnTable",
            "Effect": "Allow",
            "Action": [
                "glue:*Table*",
                "glue:*Partition*"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/books"
            ]
        }
    ]
}
```

tieni presente che il livello minimo di granularità del controllo degli accessi è a livello di tabella. Questo significa che non è possibile concedere a un utente l'accesso ad alcune partizioni in una tabella, ma non ad altre, o ad alcune colonne ma non ad altre. Un utente ha accesso a tutte le parti di una tabella o a nessuna.

Controllo degli accessi per nome prefisso e diniego esplicito

In questo esempio, supponiamo che i database e le tabelle in AWS Glue Data Catalog siano organizzati utilizzando un prefisso del nome. I database nella fase di sviluppo hanno il nome prefisso `dev-` e quelli in produzione hanno il nome prefisso `prod-`. Puoi utilizzare le seguenti policy per concedere agli sviluppatori l'accesso completo a tutti i database, tabelle, UDF e così via che hanno il prefisso `dev-`. Tuttavia, puoi anche concedere l'accesso in sola lettura a tutti gli elementi con il prefisso `prod-`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DevAndProdFullAccess",
            "Effect": "Allow",
            "Action": [
                "glue:*Database*",
                "glue:*Table*",
                "glue:*Partition*",
                "glue:*UserDefinedFunction*",
                "glue:*Connection*"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/dev-*",
                "arn:aws:glue:us-west-2:123456789012:database/prod-*",
                "arn:aws:glue:us-west-2:123456789012:table/dev-*/*",
                "arn:aws:glue:us-west-2:123456789012:table/*/dev-*",
                "arn:aws:glue:us-west-2:123456789012:table/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:table/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/dev-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/dev-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:connection/dev-*",
                "arn:aws:glue:us-west-2:123456789012:connection/prod-*"
            ]
```

```
        },
        {
            "Sid": "ProdWriteDeny",
            "Effect": "Deny",
            "Action": [
                "glue:*Create*",
                "glue:*Update*",
                "glue:*Delete*"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:database/prod-*",
                "arn:aws:glue:us-west-2:123456789012:table/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:table/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:connection/prod-*"
            ]
        }
    ]
}
```

La seconda istruzione nella policy precedente utilizza il codice esplicito `deny`. Puoi utilizzare il codice esplicito `deny` per sovrascrivere qualsiasi autorizzazione `allow` concessa al principale. Questo consente di bloccare l'accesso a risorse critiche e a impedire a un'altra policy di concedere accidentalmente l'accesso a esse.

Nell'esempio precedente, anche se la prima istruzione concede l'accesso completo alle risorse `prod-`, la seconda istruzione revoca esplicitamente l'accesso in scrittura, mantenendo solo l'accesso in lettura alle risorse `prod-`.

Autorizzazione dell'accesso utilizzando tag

Supporre ad esempio che si voglia limitare l'accesso al trigger `t2` a un utente specifico denominato `Tom` nel proprio account. Tutti gli altri utenti, tra cui `Sam`, hanno accesso al trigger `t1`. I trigger `t1` e `t2` hanno le seguenti proprietà.

```
aws glue get-triggers
{
    "Triggers": [
        {
            "State": "CREATED",
            "Type": "SCHEDULED",
```

```
            "Name": "t1",
            "Actions": [
                {
                    "JobName": "j1"
                }
            ],
            "Schedule": "cron(0 0/1 * * ? *)"
        },
        {
            "State": "CREATED",
            "Type": "SCHEDULED",
            "Name": "t2",
            "Actions": [
                {
                    "JobName": "j1"
                }
            ],
            "Schedule": "cron(0 0/1 * * ? *)"
        }
    ]
}
```

L'amministratore AWS Glue ha associato il valore di tag `Tom` (`aws:ResourceTag/Name": "Tom"`) al trigger `t2`. L'amministratore AWS Glue ha inoltre fornito a Tom una policy IAM con un'istruzione di condizione basata sul tag. Di conseguenza, Tom può utilizzare solo un'operazione AWS Glue che agisce sulle risorse con il valore di tag `Tom`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
         "Effect": "Allow",
         "Action": "glue:*",
         "Resource": "*",
         "Condition": {
             "StringEquals": {
                "aws:ResourceTag/Name": "Tom"
              }
          }
         }
    ]
}
```

Quando Tom cerca di accedere al trigger `t1` riceve un messaggio di accesso rifiutato. Allo stesso tempo, può recuperare regolarmente il trigger `t2`.

```
aws glue get-trigger --name t1

An error occurred (AccessDeniedException) when calling the GetTrigger operation:
 User: Tom is not authorized to perform: glue:GetTrigger on resource: arn:aws:glue:us-
east-1:123456789012:trigger/t1

aws glue get-trigger --name t2
{
    "Trigger": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j1"
            }
        ],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}
```

Tom non può usare l'operazione dell'API `GetTriggers` plurale per elencare i trigger in quanto questa operazione non supporta il filtro sui tag.

Per concedere a Tom l'accesso a `GetTriggers`, l'amministratore di AWS Glue crea una policy che divide le autorizzazioni in due sezioni. Una sezione consente a Tom di accedere a tutti i trigger con l'operazione API `GetTriggers`. La seconda sezione consente a Tom di accedere alle operazioni API che sono contrassegnate con il valore `Tom`. Con questa policy, a Tom è consentito l'accesso `GetTriggers` e `GetTrigger` al trigger `t2`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "glue:GetTriggers",
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "glue:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Name": "Tom"
                }
            }
        }
    ]
}
```

Negazione dell'accesso utilizzando tag

Un altro approccio per una policy delle risorse consiste nel negare in modo esplicito l'accesso alle risorse.

> **Important**
>
> Una policy di negazione esplicita non funziona per le operazioni dell'API plurali, come `GetTriggers`.

Nella seguente policy di esempio, sono consentite tutte le operazioni di processo AWS Glue Tuttavia, la seconda dichiarazione `Effect` nega esplicitamente l'accesso ai processi contrassegnati con la chiave `Team` e il valore `Special`.

Quando un amministratore collega le seguenti policy a un'identità, questa può accedere a tutti i processi tranne quelli contrassegnati con la chiave `Team` e il valore `Special`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "glue:*",
      "Resource": "arn:aws:glue:us-east-1:123456789012:job/*"
    },
    {
      "Effect": "Deny",
      "Action": "glue:*",
```

```
      "Resource": "arn:aws:glue:us-east-1:123456789012:job/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Team": "Special"
        }
      }
    }
  ]
}
```

Utilizzo di tag con operazioni API in elenco e in batch

Un terzo approccio per scrivere una policy basata sulle risorse consiste nel consentire l'accesso alle risorse utilizzando un'operazione API `List` per elencare le risorse corrispondenti a un dato valore di tag. Quindi, si utilizza l'operazione API `Batch` corrispondente per consentire l'accesso ai dettagli delle risorse specifiche. Con questo approccio, l'amministratore non ha bisogno di consentire l'accesso alle operazioni API `GetCrawlers`, `GetDevEndpoints`, `GetJobs` o `GetTriggers` plurali. Puoi invece consentire la possibilità di elencare le risorse con le seguenti operazioni API:

- `ListCrawlers`
- `ListDevEndpoints`
- `ListJobs`
- `ListTriggers`

Puoi inoltre consentire la possibilità di ottenere i dettagli delle singole risorse con le seguenti operazioni API:

- `BatchGetCrawlers`
- `BatchGetDevEndpoints`
- `BatchGetJobs`
- `BatchGetTriggers`

In qualità di amministratore, per l'utilizzo di questo approccio, è possibile:

1. Aggiungere i tag a crawler, endpoint di sviluppo, processi e trigger.
2. Rifiutare l'accesso degli utenti alle operazioni API `Get`, ad esempio `GetCrawlers`, `GetDevEndponts`, `GetJobs` e `GetTriggers`.

3. Per permettere agli utenti di determinare a quali risorse con tag hanno accesso, consentire l'accesso degli utenti alle operazioni API `List`, ad esempio `ListCrawlers`, `ListDevEndponts`, `ListJobs` e `ListTriggers`.
4. Rifiutare l'accesso degli utenti alle API con tag AWS Glue, ad esempio `TagResource` e `UntagResource`.
5. Consentire l'accesso degli utenti ai dettagli delle risorse con le operazioni API `BatchGet`, ad esempio `BatchGetCrawlers`, `BatchGetDevEndponts`, `BatchGetJobs` e `BatchGetTriggers`.

Ad esempio, quando si richiama l'operazione `ListCrawlers`, fornire un valore di tag che corrisponda al nome dell'utente. Quindi il risultato è un elenco di crawler corrispondenti ai valori di tag forniti. Fornisci l'elenco dei nomi a `BatchGetCrawlers` per ottenere informazioni dettagliate su ogni crawler con il tag specificato.

Ad esempio, se Tom deve essere in grado di recuperare solo i dettagli dei trigger con il tag `Tom`, l'amministratore può aggiungere i tag ai trigger per `Tom`, rifiutare l'accesso all'operazione API `GetTriggers` a tutti gli utenti e consentire l'accesso di tutti gli utenti a `ListTriggers` e `BatchGetTriggers`.

Ecco la policy basata sulle risorse che l'amministratore di AWS Glue concede a Tom. Nella prima sezione della policy, le operazioni API AWS Glue sono rifiutate per `GetTriggers`. Nella seconda sezione della policy, `ListTriggers` è consentito per tutte le risorse. Tuttavia, nella terza sezione, tali risorse con il tag `Tom` possono eseguire l'accesso `BatchGetTriggers`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "glue:GetTriggers",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListTriggers"
            ],
            "Resource": [
                "*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:BatchGetTriggers"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Name": "Tom"
                }
            }
        }
    ]
}
```

Usando gli stessi trigger dell'esempio precedente, Tom può accedere al trigger `t2`, ma non al trigger `t1`. L'esempio seguente mostra i risultati quando Tom cerca di accedere a `t1` e `t2` con `BatchGetTriggers`.

```
aws glue batch-get-triggers --trigger-names t2
{
    "Triggers": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j2"
            }
        ],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}

aws glue batch-get-triggers  --trigger-names t1

An error occurred (AccessDeniedException) when calling the BatchGetTriggers operation:
 No access to any requested resource.
```

L'esempio seguente mostra i risultati quando Tom cerca di accedere al trigger `t2` e `t3` (che non esiste) nella stessa chiamata `BatchGetTriggers`. Nota che poiché Tom ha accesso al trigger `t2` esistente, viene restituito solo `t2`. Sebbene a Tom sia consentito accedere al trigger `t3`, il trigger `t3` non esiste e pertanto `t3` viene restituito nella risposta in un elenco di `"TriggersNotFound": []`.

```
aws glue batch-get-triggers --trigger-names t2 t3
{
    "Triggers": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j2"
            }
        ],
        "TriggersNotFound": ["t3"],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}
```

Controllo delle impostazioni utilizzando le chiavi di condizione o di contesto

Quando si concedono le autorizzazioni per creare e aggiornare i processi, è possibile utilizzare le chiavi di condizione o le chiavi di contesto. Queste sezioni trattano le chiavi:

- Policy di controllo che controllano le impostazioni utilizzando le chiavi di condizione
- Policy di controllo che controllano le impostazioni utilizzando le chiavi di contesto

Policy di controllo che controllano le impostazioni utilizzando le chiavi di condizione

AWS Glue offre tre tasti di condizione IAM: `glue:VpcIds`, `glue:SubnetIds` e `glue:SecurityGroupIds`. Quando si concedono le autorizzazioni per creare e aggiornare i processi, è possibile utilizzare le chiavi di condizione nelle policy IAM. È possibile utilizzare questa impostazione per garantire che i processi o le sessioni non vengano creati (o aggiornati) per essere eseguiti al di fuori dell'ambiente VPC desiderato. Le informazioni sull'impostazione del VPC non sono un input diretto dalla richiesta `CreateJob`, ma vengono inferite dal campo "connessioni" del processo che punta a una connessione AWS Glue.

Esempio di utilizzo

Creazione di una connessione di tipo di rete AWS Glue denominata "connessione monitorata dal traffico" con il VpcId desiderato "vpc-id1234", SubnetIds e SecurityGroupIds.

Specifica la condizione delle chiavi di condizione per le operazioni `CreateJob` e `UpdateJob`nella policy IAM.

```
{
  "Effect": "Allow",
  "Action": [
    "glue:CreateJob",
    "glue:UpdateJob"
  ],
  "Resource": [
    "*"
  ],
  "Condition": {
    "ForAnyValue:StringLike": {
      "glue:VpcIds": [
        "vpc-id1234"
      ]
    }
  }
}
```

È possibile creare una policy IAM simile per vietare la creazione di un processo AWS Glue senza specificare le informazioni di connessione.

Limitazione delle sessioni sui VPC

Per imporre l'esecuzione delle sessioni create all'interno di un VPC specificato, puoi limitare l'autorizzazione del ruolo aggiungendo un effetto `Deny` all'operazione `glue:CreateSession`, a condizione che glue:vpc-id non sia uguale a vpc-<123>. Per esempio:

```
"Effect": "Deny",
"Action": [
    "glue:CreateSession"
 ],
"Condition": {
    "StringNotEquals" : {"glue:VpcIds" : ["vpc-123"]}
}
```

È inoltre possibile imporre l'esecuzione delle sessioni create all'interno di un VPC aggiungendo un effetto `Deny` all'operazione `glue:CreateSession` a condizione che `glue:vpc-id` sia nullo. Per esempio:

```
{
    "Effect": "Deny",
    "Action": [
        "glue:CreateSession"
    ],
      "Condition": {
        "Null": {"glue:VpcIds": true}
    }
},
{
    "Effect": "Allow",
    "Action": [
        "glue:CreateSession"
    ],
    "Resource": ["*"]
}
```

Policy di controllo che controllano le impostazioni utilizzando le chiavi di contesto

AWS Glue fornisce una chiave di contesto (`glue:CredentialIssuingService=glue.amazonaws.com`) a ogni sessione di ruolo che AWS Gluemette a disposizione dell'endpoint di processo e sviluppatore. Ciò consente di implementare i controlli di sicurezza per le operazioni effettuate da script AWS Glue. AWS Glue fornisce un'altra chiave di contesto (`glue:RoleAssumedBy=glue.amazonaws.com`) a ciascuna sessione di ruolo dove AWS Glue fa una chiamata a un altro servizio AWS per conto del cliente (non da un endpoint processo/sviluppatore, ma direttamente dal servizio AWS Glue).

Esempio di utilizzo

Specifica l'autorizzazione condizionale nella policy IAM e allegala al ruolo che deve essere utilizzato da un processo AWS Glue. Ciò garantisce che determinate operazioni siano permesse/negate in base al fatto che la sessione di ruolo sia utilizzata per un ambiente di runtime di un processo AWS Glue.

```
{
    "Effect": "Allow",
```

```
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::confidential-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:CredentialIssuingService": "glue.amazonaws.com"
        }
    }
}
```

Negare a un'identità la possibilità di creare sessioni di anteprima dei dati

Questa sezione contiene un esempio di policy IAM utilizzato per negare a un'identità la possibilità di creare sessioni di anteprima dei dati. Collega questa policy all'identità, che è distinta dal ruolo utilizzato dalla sessione di anteprima dei dati durante la sua esecuzione.

```
{
    "Sid": "DatapreviewDeny",
    "Effect": "Deny",
     "Action": [
           "glue:CreateSession"
     ],
     "Resource": [
          "arn:aws:glue:*:*:session/glue-studio-datapreview*"
      ]
 }
```

Esempi di policy basate su risorse per AWS Glue

Questa sezione contiene le policy di esempio basate su risorse, tra cui le policy che concedono l'accesso multi-account.

Gli esempi utilizzano AWS Command Line Interface (AWS CLI) per l'interazione con le operazioni delle API del servizio AWS Glue. Puoi eseguire le stesse operazioni sulla console AWS Glue o utilizzare un SDK AWS.

> **Important**
>
> Modificando una policy della risorsa AWS Glue, potresti accidentalmente revocare le autorizzazioni per gli utenti AWS Glue esistenti nel tuo account e provocare interruzioni impreviste. Prova questi esempi solo con gli account di sviluppo o di test e verifica che non interrompano nessun flusso di lavoro esistente prima di apportare le modifiche.

Argomenti

Considerazioni sull'utilizzo di policy basate su risorse con AWS Glue

> **Note**
>
> Sia le policy IAM che una policy della risorsa AWS Glue richiedono pochi secondi, per la propagazione. Dopo aver collegato una nuova policy, potresti anche notare che la policy precedente è ancora in vigore finché la nuova policy non viene propagata attraverso il sistema.

È possibile utilizzare un documento di policy scritte in formato JSON per creare o modificare una policy della risorsa. La sintassi della policy è la stessa di una policy IAM basata sulle identità (consulta la documentazione di riferimento sulle policy JSON IAM), con le seguenti eccezioni:

- Un blocco `"Principal"` o `"NotPrincipal"` è obbligatorio per ogni istruzione di policy.
- Il `"Principal"` o il `"NotPrincipal"` deve identificare principali esistenti validi. I modelli dei caratteri jolly (ad esempio `arn:aws:iam::`*`account-id`*`:user/*`) non sono consentiti.
- Il blocco `"Resource"` nella policy richiede che tutte le risorse ARN corrispondano alla seguente sintassi di espressione regolare (in cui il primo `%s` corrisponde alla *`regione`* e il secondo `%s` corrisponde all'*`id-account`*):

  ```
  *arn:aws:glue:%s:%s:(\*|[a-zA-Z\*]+\/?.*)
  ```

  Ad esempio, sia `arn:aws:glue:us-west-2:`*`account-id`*`:*` che `arn:aws:glue:us-west-2:`*`account-id`*`:database/default` sono consentiti, ma non è consentito `*`.
- A differenza delle policy basate sulle identità, una policy della risorsa AWS Glue deve contenere solo Amazon Resource Names (ARN) di risorse appartenenti al catalogo a cui la policy è collegata. Tali ARN iniziano sempre con `arn:aws:glue:`.
- Una policy non può impedire l'ulteriore creazione o modifica dell'identità che la crea.
- La dimensione di un documento JSON di policy della risorsa non può superare 10 KB.

Utilizza una policy della risorsa per controllare gli accessi nello stesso account

In questo esempio, un utente admin nell'account A crea una policy della risorsa che concede all'utente IAM `Alice` dell'account A l'accesso completo al catalogo. Alice non ha alcuna policy IAM collegata.

Per eseguire questa operazione, l'utente amministratore esegue il comando seguente nella AWS CLI.

```
# Run as admin of Account A
$ aws glue put-resource-policy --profile administrator-name --region us-west-2 --
policy-in-json '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": [
          "arn:aws:iam::account-A-id:user/Alice"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:*"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:account-A-id:*"
      ]
    }
  ]
}'
```

Invece di inserire il documento di policy JSON come parte del tuo comando AWS CLI, è possibile risparmiare un documento di policy in un file e fare riferimento al percorso di file nel comando AWS CLI che ha il prefisso `file://`. Di seguito è riportato un esempio di come svolgere questa operazione.

```
$ echo '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": [
          "arn:aws:iam::account-A-id:user/Alice"
```

```
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:*"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:account-A-id:*"
      ]
    }
  ]
}' > /temp/policy.json

$ aws glue put-resource-policy --profile admin1 \
    --region us-west-2 --policy-in-json file:///temp/policy.json
```

Dopo la propagazione di questa policy basata sulle risorse, Alice può accedere a tutte le risorse di AWS Glue nell'account A, come segue.

```
# Run as user Alice
$ aws glue create-database --profile alice --region us-west-2 --database-input '{
    "Name": "new_database",
    "Description": "A new database created by Alice",
    "LocationUri": "s3://my-bucket"
}'

$ aws glue get-table --profile alice --region us-west-2 --database-name "default" --
table-name "tbl1"}
```

in risposta alla chiamata `get-table` di Alice, il servizio AWS Glue restituisce quanto segue.

```
{
  "Table": {
    "Name": "tbl1",
    "PartitionKeys": [],
    "StorageDescriptor": {
        ......
    },
    ......
  }
}
```

Policy gestite da AWS per AWS Glue

Una policy gestita da AWSè una policy autonoma creata e amministrata da AWS. Le policy gestite da AWSsono progettate per fornire autorizzazioni per molti casi d'uso comuni in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Ricorda che le policy gestite da AWSpotrebbero non concedere autorizzazioni con privilegi minimi per i tuoi casi d'uso specifici perché possono essere utilizzate da tutti i clienti AWS. Consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i tuoi casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle policy gestite da AWS. Se AWSaggiorna le autorizzazioni definite in una policy gestita da AWS, l'aggiornamento riguarda tutte le identità principali (utenti, gruppi e ruoli) a cui è collegata la policy. È molto probabile che AWSaggiorni una policy gestita da AWSquando viene lanciato un nuovo Servizio AWSo nuove operazioni API diventano disponibili per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWSnella Guida per l'utente di IAM.

Policy gestite da AWS (predefinite) per AWS Glue

AWS gestisce molti casi di utilizzo comune con policy IAM autonome create e amministrate da AWS. Queste policy gestite da AWS concedono le autorizzazioni necessarie per i casi di utilizzo comune in modo da non dover cercare quali sono le autorizzazioni richieste. Per ulteriori informazioni, consulta AWSpolicy gestite nella Guida per l'utente di IAM.

Le seguenti policy gestite da AWS, che puoi collegare alle identità nel tuo account, sono specifiche di AWS Glue e sono raggruppate per scenario di caso d'uso:

- AWSGlueConsoleFullAccess— Garantisce l'accesso completo alle AWS Glue risorse quando un'identità a cui è associata la politica utilizza ilAWS Management Console. Se segui la convenzione per la denominazione per le risorse specificate nella policy, gli utenti hanno la piena funzionalità della console. Questa policy è in genere collegata agli utenti della console AWS Glue.
- AWSGlueServiceRole— Garantisce l'accesso alle risorse che AWS Glue i vari processi richiedono per l'esecuzione per conto dell'utente. Queste risorse includono Amazon S3AWS Glue, IAM, CloudWatch Logs e Amazon EC2. Se segui la convenzione di denominazione per le risorse specificata in questa policy, i processi AWS Glue dispongono delle autorizzazioni richieste. Questa policy è in genere collegata ai ruoli specificati quando si definiscono crawler, processi ed endpoint di sviluppo.

- AwsGlueSessionUserRestrictedServiceRole— Fornisce l'accesso completo a tutte le AWS Glue risorse ad eccezione delle sessioni. Permette agli utenti di creare e utilizzare solo le sessioni interattive associate all'utente. Questa policy include altre autorizzazioni necessarie ad AWS Glue per gestire le risorse AWS Glue in altri servizi AWS. La policy permette anche l'aggiunta di tag a risorse AWS Glue in altri servizi AWS.

  > **Note**
  >
  > Per ottenere tutti i vantaggi della sicurezza, non concedere questa policy a un utente a cui sia stata assegnata la policy `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` o `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedPolicy— Fornisce l'accesso per creare sessioni AWS Glue interattive utilizzando l'operazione `CreateSession` API solo se vengono forniti una chiave di tag «proprietario» e un valore che corrispondono all'ID AWS utente dell'assegnatario. Questa policy di identità è collegata all'utente IAM che richiama l'operazione dell'API `CreateSession`. Questa policy permette inoltre all'assegnatario di interagire con le risorse di sessione interattive AWS Glue che sono state create con un tag "proprietario" e un valore corrispondente alla sua ID utente AWS. Questa policy nega l'autorizzazione di modificare o rimuovere i tag "proprietario" da una risorsa di sessione AWS Glue dopo la creazione della sessione.

  > **Note**
  >
  > Per ottenere tutti i vantaggi della sicurezza, non concedere questa policy a un utente a cui sia stata assegnata la policy `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` o `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedNotebookServiceRuolo: fornisce un accesso sufficiente alla sessione del AWS Glue Studio notebook per interagire con risorse di sessione AWS Glue interattive specifiche. Si tratta di risorse create con il valore del tag "proprietario" corrispondente all'ID utente AWS del principale (utente IAM o ruolo) che crea il notebook. Per ulteriori informazioni su questi tag, consulta il grafico Valori della chiave dell'entità principale nella Guida per l'utente IAM.

  Questa policy del ruolo di servizio viene collegata al ruolo specificato con un comando magic all'interno del notebook o viene passata come ruolo all'operazione `CreateSession` dell'API. Questa policy permette inoltre al principale di creare una sessione interattiva AWS Glue dall'interfaccia AWS Glue Studio Notebook solo se una chiave di tag "proprietario" e il valore

corrispondono all'ID utente AWS del principale. Questa policy nega l'autorizzazione di modificare o rimuovere i tag "proprietario" da una risorsa di sessione AWS Glue dopo la creazione della sessione. Questa policy include anche le autorizzazioni per la scrittura e la lettura da bucket Amazon S3, la CloudWatch scrittura di log e la creazione ed eliminazione di tag per le risorse Amazon EC2 utilizzate da. AWS Glue

> **Note**
>
> Per ottenere tutti i vantaggi della sicurezza, non concedere questa policy a un ruolo a cui sia stata assegnata la policy `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` o `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedNotebookPolicy— Fornisce l'accesso per creare una sessione AWS Glue interattiva dall'interfaccia del AWS Glue Studio notebook solo se sono presenti una chiave di tag «proprietario» e un valore che corrispondono AWS all'ID utente del principale (utente o ruolo IAM) che crea il notebook. Per ulteriori informazioni su questi tag, consulta il grafico Valori della chiave dell'entità principale nella Guida per l'utente IAM.

  Questa policy è associata al principale (utente o ruolo IAM) che crea sessioni dall'interfaccia AWS Glue Studio notebook. Inoltre, questa policy offre un accesso adeguato al notebook AWS Glue Studio per interagire con specifiche risorse della sessione interattiva AWS Glue. Si tratta di risorse create con il valore del tag "proprietario" che corrisponde all'ID utente AWS del principale. Questa policy nega l'autorizzazione di modificare o rimuovere i tag "proprietario" da una risorsa di sessione AWS Glue dopo la creazione della sessione.
- AWSGlueServiceNotebookRole— Concede l'accesso alle AWS Glue sessioni avviate su un AWS Glue Studio notebook. Questa policy permette di elencare e ottenere informazioni sulla sessione per tutte le sessioni, ma permette solo agli utenti di creare e utilizzare le sessioni contrassegnate con il loro ID utente AWS. Questa policy nega l'autorizzazione di modificare o rimuovere i tag "proprietario" da risorse di sessione AWS Glue taggate con il loro ID AWS.

  Assegna questa policy all'utente AWS che crea lavori utilizzando l'interfaccia notebook in AWS Glue Studio.
- AWSGlueConsoleSageMakerNotebookFullAccess— Garantisce l'accesso completo a AWS Glue e alle SageMaker risorse quando l'identità a cui è associata la politica utilizza ilAWS Management Console. Se segui la convenzione per la denominazione per le risorse specificate nella policy, gli utenti hanno la piena funzionalità della console. Questa policy viene in genere associata agli utenti della AWS Glue console che gestiscono i SageMaker notebook.

- AWSGlueSchemaRegistryFullAccess— Concede l'accesso completo alle risorse del registro AWS Glue dello schema quando l'identità a cui è associata la politica utilizza o. AWS Management Console AWS CLI Se segui la convenzione per la denominazione per le risorse specificate nella policy, gli utenti hanno la piena funzionalità della console. Questa policy è in genere allegata agli utenti della console AWS Glue o della AWS CLI che gestiscono il registro dello schema di AWS Glue.
- AWSGlueSchemaRegistryReadonlyAccess— Concede l'accesso in sola lettura alle risorse del registro AWS Glue dello schema quando un'identità a cui è associata la politica utilizza o. AWS Management Console AWS CLI Se segui la convenzione per la denominazione per le risorse specificate nella policy, gli utenti hanno la piena funzionalità della console. Questa policy è in genere collegata agli utenti della console AWS Glue o della AWS CLI che utilizzano il registro dello schema di AWS Glue.

> **Note**
>
> Per esaminare queste policy di autorizzazione, accedi alla console IAM ed esegui la ricerca delle policy specifiche.

Puoi anche creare policy IAM personalizzate per concedere le autorizzazioni per operazioni e risorse AWS Glue. Puoi associare queste policy personalizzate agli utenti o ai gruppi IAM che richiedono tali autorizzazioni.

Aggiornamenti AWS Glue e alle policy gestite da AWS

Visualizza i dettagli sugli aggiornamenti alle policy gestite da AWS per AWS Glue da quando questo servizio ha iniziato a tenere traccia delle modifiche. Per gli avvisi automatici sulle modifiche apportate a questa pagina, sottoscrivi il feed RSS nella pagina della cronologia dei documenti di AWS Glue.

Modifica	Descrizione	Data
AWSGlueServiceNoteboo kRole — Aggiornamento minore di una politica esistente.	Aggiungi `glue:StartCompletion` e `glue:GetCompletion` alla policy. Necessario	TBD

Modifica	Descrizione	Data
	per l'integrazione dei dati di Amazon Q in AWS Glue.	
AwsGlueSessionUser RestrictedNotebookPolicy — Aggiornamento minore di una politica esistente.	Aggiungi `glue:StartCompletion` e `glue:GetCompletion` alla policy. Necessario per l'integrazione dei dati di Amazon Q in AWS Glue.	29 novembre 2023
AWSGlueServiceNotebookRole — Aggiornamento minore di una politica esistente.	Aggiungi `codewhisperer:GenerateRecommendations` alla policy. Necessario per una nuova funzionalità in cui AWS Glue genera CodeWhisperer consigli.	9 ottobre 2023
AWSGlueServiceRole — Aggiornamento minore di una politica esistente.	Restringi l'ambito delle CloudWatch autorizzazioni per riflettere meglio la registrazione di AWS Glue.	4 agosto 2023
AWSGlueConsoleFullAccess — Aggiornamento minore di una politica esistente.	Aggiungi le autorizzazioni Elenca e Descrivi per le ricette `databrew` alla policy. Necessario per fornire l'accesso amministrativo completo alle nuove funzionalità in cui AWS Glue può accedere alle ricette.	9 maggio 2023

Modifica	Descrizione	Data
AWSGlueConsoleFullAccess — Aggiornamento minore di una politica esistente.	Aggiungi `cloudformation:ListStacks` alla policy. Conserva le funzionalità esistenti dopo le modifiche ai requisiti di autorizzazione di AWS CloudFormation.	28 marzo 2023
Aggiunte nuove policy gestite per la funzionalità sessioni interattive: • AwsGlueSessionUserRestrictedServiceRole • AwsGlueSessionUserRestrictedPolicy • AwsGlueSessionUserRestrictedNotebookServiceRuolo • AwsGlueSessionUserRestrictedNotebookPolicy	Queste policy sono state progettate per fornire ulteriore sicurezza per le sessioni interattive e i notebook in AWS Glue Studio. Le policy limitano l'accesso all'operazione dell'API `CreateSession`, in modo che solo il proprietario abbia accesso.	30 novembre 2021

Modifica	Descrizione	Data
AWSGlueConsoleSageMakerNotebookFullAccess — Aggiornamento a una politica esistente.	Rimosso una risorsa ARN ridondante (`arn:aws:s3:::aws-glue-*/*`) per l'operazione che concede le autorizzazioni di lettura/scrittura sui bucket Amazon S3 che AWS Glue utilizza per archiviare script e file temporanei. Risolto un problema di sintassi modificando `"StringEquals"` in `"ForAnyValue:StringLike"` e spostate le righe `"Effect": "Allow"` per precedere la riga `"Action":` in ogni luogo in cui erano fuori uso.	15 luglio 2021
AWSGlueConsoleFullAccess — Aggiornamento a una politica esistente.	Rimosso una risorsa ARN ridondante (`arn:aws:s3:::aws-glue-*/*`) per l'operazione che concede le autorizzazioni di lettura/scrittura sui bucket Amazon S3 che AWS Glue utilizza per archiviare script e file temporanei.	15 luglio 2021
AWS Glue ha iniziato il rilevamento delle modifiche.	AWS Glue ha iniziato il rilevamento delle modifiche per le relative policy gestite da AWS.	10 giugno 2021

Come specificare gli ARN delle risorse AWS Glue

In AWS Glue, è possibile controllare l'accesso alle risorse utilizzando una policy AWS Identity and Access Management (IAM). In una policy, devi utilizzare un Amazon Resource Name (ARN) per identificare la risorsa a cui si applica la policy stessa. Non tutte le risorse in AWS Glue supportano gli ARN.

Argomenti

ARN del catalogo dati

Le risorse del catalogo dati sono organizzate secondo una struttura gerarchica nella quale `catalog` funge da root.

```
arn:aws:glue:region:account-id:catalog
```

Ogni account AWS ha un singolo catalogo dati in una regione AWS con ID account da 12 cifre come ID catalogo. Alle risorse sono associati ARN univoci come descritto nella seguente tabella.

Tipo di risorsa	Formato ARN
Catalogo	`arn:aws:glue: region:account-id :catalog` Ad esempio: `arn:aws:glue:us-east-1:123456789012: catalog`
Database	`arn:aws:glue: region:account-id :database/ database name` Ad esempio: `arn:aws:glue:us-east-1:123456789012: database/db1`

Tipo di risorsa	Formato ARN
Tabella	`arn:aws:glue:` *`region`*`:`*`account-id`* `:table/`*`database name`*`/`*`table name`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:table/db1/tbl1`
Funzione definita dall'utente	`arn:aws:glue:` *`region`*`:`*`account-id`* `:userDefinedFunction/` *`database name`*`/`*`user-defined function name`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:userDefinedFunction/db1/func1`
Connessione	`arn:aws:glue:` *`region`*`:`*`account-id`* `:connection/` *`connection name`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:connection/connection1`
Sessioni interattive	`arn:aws:glue:` *`region`*`:`*`account-id`* `:session/` *`interactive session id`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:session/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`

Per abilitare il controllo capillare degli accessi, puoi utilizzare questi ARN nelle policy e nelle policy delle risorse IAM per concedere o negare l'accesso a risorse specifiche. Nelle policy sono ammessi i caratteri jolly. Ad esempio, il seguente ARN corrisponde a tutte le tabelle nel database `default`.

```
arn:aws:glue:us-east-1:123456789012:table/default/*
```

> **Important**
>
> Tutte le operazioni eseguite su una risorsa del catalogo dati richiedono l'autorizzazione alla risorsa e tutti i predecessori di tale risorsa. Ad esempio, la creazione di una partizione per una tabella necessita dell'autorizzazione su tabella, database e catalogo in cui si trova la tabella

stessa. L'esempio seguente mostra le autorizzazioni necessarie per creare le partizioni sulla tabella `PrivateTable` nel database `PrivateDatabase` nel catalogo dati.

```
{
   "Sid": "GrantCreatePartitions",
   "Effect": "Allow",
   "Action": [
       "glue:BatchCreatePartitions"
   ],
   "Resource": [
       "arn:aws:glue:us-east-1:123456789012:table/PrivateDatabase/PrivateTable",
       "arn:aws:glue:us-east-1:123456789012:database/PrivateDatabase",
       "arn:aws:glue:us-east-1:123456789012:catalog"
   ]
}
```

Oltre alle autorizzazioni per la risorsa e tutti i suoi predecessori, tutte le operazioni di eliminazione richiedono l'autorizzazione su tutti i figli di tale risorsa. Ad esempio, per eliminare un database è necessario disporre di autorizzazioni per tutte le tabelle e per le funzioni definite dall'utente del database, così come per il database e il catalogo in cui si trova il database. L'esempio seguente mostra le autorizzazioni necessarie per eliminare il database `PrivateDatabase` nel catalogo dati.

```
{
   "Sid": "GrantDeleteDatabase",
   "Effect": "Allow",
   "Action": [
       "glue:DeleteDatabase"
   ],
   "Resource": [
       "arn:aws:glue:us-east-1:123456789012:table/PrivateDatabase/*",
       "arn:aws:glue:us-east-1:123456789012:userDefinedFunction/PrivateDatabase/
*",
       "arn:aws:glue:us-east-1:123456789012:database/PrivateDatabase",
       "arn:aws:glue:us-east-1:123456789012:catalog"
   ]
}
```

Riepilogando, le operazioni sulle risorse del catalogo dati seguono queste regole di autorizzazione:

- Le operazioni sul catalogo richiedono solo l'autorizzazione per il catalogo.
- Le operazioni su un database richiedono l'autorizzazione su database e catalogo.
- Le operazioni di eliminazione su un database richiedono l'autorizzazione su database e catalogo, oltre che su tutte le tabelle e funzioni definite dall'utente del database.
- Le operazioni su una tabella, partizione o versione di una tabella richiedono l'autorizzazione su tabella, database e catalogo.
- Le operazioni su una funzione definita dall'utente richiedono l'autorizzazione su funzione definita dall'utente, catalogo e database.
- Le operazioni su una connessione richiedono l'autorizzazione su connessione e catalogo.

ARN per oggetti non inclusi in AWS Glue

Alcune risorse AWS Glue consentono autorizzazioni a livello di risorsa per controllare l'accesso utilizzando un ARN. Puoi utilizzare questi ARN nelle tue policy IAM per consentire un controllo capillare degli accessi. La tabella seguente elenca le risorse che possono contenere gli ARN della risorsa.

Tipo di risorsa	Formato ARN
Crawler	`arn:aws:glue:` *`region`*`:`*`account-id`* `:crawler/` *`crawler-name`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:crawler/mycrawler`
Processo	`arn:aws:glue:` *`region`*`:`*`account-id`* `:job/`*`job-name`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:job/testjob`
Trigger	`arn:aws:glue:` *`region`*`:`*`account-id`* `:trigger/` *`trigger-name`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:trigger/sampletrigger`

Tipo di risorsa	Formato ARN
Endpoint di sviluppo	`arn:aws:glue:` *`region`*`:`*`account-id`* `:devEndpoint/` *`development-endpoint-name`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:devEndpoint/temporarydevendpoint`
Trasformazione basata su machine learning	`arn:aws:glue:` *`region`*`:`*`account-id`* `:mlTransform/` *`transform-id`* Ad esempio: `arn:aws:glue:us-east-1:123456789012:mlTransform/tfm-1234567890`

Controllo accessi per operazioni API singole non nel catalogo AWS Glue

Le operazioni delle API singole non nel catalogo AWS Glue, agiscono su un singolo elemento (endpoint di sviluppo). Alcuni esempi sono `GetDevEndpoint`, `CreateUpdateDevEndpoint` e `UpdateDevEndpoint`. Per queste operazioni, una policy deve inserire il nome dell'API nel blocco `"action"` e la risorsa ARN nel blocco `"resource"`.

Supponiamo che si desideri consentire a un utente di chiamare l'operazione `GetDevEndpoint`. La policy seguente concede il livello minimo di autorizzazioni necessarie per un endpoint denominato `myDevEndpoint-1`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "MinimumPermissions",
            "Effect": "Allow",
            "Action": "glue:GetDevEndpoint",
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/
myDevEndpoint-1"
        }
    ]
}
```

La policy seguente consente l'accesso di `UpdateDevEndpoint` alle risorse che corrispondono a `myDevEndpoint-` con un carattere jolly (*):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionWithWildcard",
            "Effect": "Allow",
            "Action": "glue:UpdateDevEndpoint",
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/myDevEndpoint-
*"
        }
    ]
}
```

Puoi combinare le due policy come nell'esempio seguente. È possibile vedere `EntityNotFoundException` per ogni endpoint di sviluppo il cui nome inizia con `A`. Tuttavia, quando si cerca di accedere ad altri endpoint di sviluppo viene restituito un errore di accesso negato.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CombinedPermissions",
            "Effect": "Allow",
            "Action": [
                "glue:UpdateDevEndpoint",
                "glue:GetDevEndpoint"
            ],
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/A*"
        }
    ]
}
```

Controllo degli accessi per le operazioni API di AWS Glue per oggetti non inclusi nel catalogo che richiamano più elementi

Alcune operazioni API AWS Glue richiamano più elementi (più endpoint di sviluppo); ad esempio, `GetDevEndpoints`. Per questa operazione, è possibile specificare solo risorsa tramite carattere jolly (*), non ARN specifici.

Ad esempio, per includere `GetDevEndpoints` nella policy, la risorsa deve rientrare tra quelle identificate dal carattere jolly (*). L'ambito delle operazioni singole (`GetDevEndpoint`, `CreateDevEndpoint` e `DeleteDevendpoint`) viene definito anche per tutte le risorse (*) nell'esempio.

```
{
            "Sid": "PluralAPIIncluded",
            "Effect": "Allow",
            "Action": [
                "glue:GetDevEndpoints",
                "glue:GetDevEndpoint",
                "glue:CreateDevEndpoint",
                "glue:UpdateDevEndpoint"
            ],
            "Resource": [
                "*"
            ]
}
```

Controllo degli accessi per le operazioni API BatchGet di AWS Glue per oggetti non in catalogo

Alcune operazioni API AWS Glue richiamano più elementi (più endpoint di sviluppo); ad esempio, `BatchGetDevEndpoints`. Per questa operazione, è possibile specificare un ARN per limitare l'ambito delle risorse accessibili.

Ad esempio, per consentire l'accesso a un determinato endpoint di sviluppo, includere `BatchGetDevEndpoints` nella policy con il relativo ARN di risorsa.

```
{
            "Sid": "BatchGetAPIIncluded",
            "Effect": "Allow",
            "Action": [
                "glue:BatchGetDevEndpoints"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:123456789012:devEndpoint/de1"
            ]
}
```

Con questa policy puoi accedere all'endpoint di sviluppo denominato `de1`. Tuttavia, se cerchi di accedere all'endpoint di sviluppo `de2`, viene restituito un errore.

```
An error occurred (AccessDeniedException) when calling the BatchGetDevEndpoints
 operation: No access to any requested resource.
```

> **Important**
>
> Per approcci alternativi alla configurazione delle policy IAM, ad esempio usando le operazioni API `List` e `BatchGet`, consultare Esempi di policy basate su identità per AWS Glue.

Come concedere l'accesso multi-account

Autorizzazione dell'accesso alle risorse del catalogo dati tra account consente ai processi di estrazione, trasformazione e caricamento (ETL) di eseguire query e join di dati da account diversi.

Argomenti

- Metodi per concedere l'accesso multi-account in AWS Glue
- Aggiunta o aggiornamento della policy della risorsa di catalogo dati
- Come effettuare una chiamata API multi-account
- Come effettuare una chiamata ETL multi-account
- Registrazione tra più account CloudTrail
- Proprietà e fatturazione delle risorse multi-account
- Restrizioni di accesso multi-account

Metodi per concedere l'accesso multi-account in AWS Glue

Puoi concedere l'accesso ai tuoi dati ad AWS account esterni utilizzando AWS Glue metodi o utilizzando sovvenzioni AWS Lake Formation tra account. I AWS Glue metodi utilizzano policy AWS Identity and Access Management (IAM) per ottenere un controllo granulare degli accessi. Lake Formation utilizza un modello di autorizzazioni `GRANT/REVOKE` più semplice, simile ai comandi `GRANT/REVOKE` in un sistema di database relazionale.

In questa sezione viene descritto l'utilizzo dei metodi AWS Glue. Per ulteriori informazioni sull'utilizzo delle autorizzazioni multi-account Lake Formation, consulta Concedere autorizzazioni Lake Formation nella Guida per gli sviluppatori di AWS Lake Formation .

Sono disponibili due metodi AWS Glue, per concedere l'accesso multi-account a una risorsa:

- Utilizzare una policy della risorsa del catalogo dati
- Utilizzare un ruolo IAM

Concedere l'accesso multi-account utilizzando una policy della risorsa

Di seguito sono riportati i passaggi generali per concedere l'accesso multi-account utilizzando una policy della risorsa del catalogo dati:

1. Un amministratore (o un altro tipo di identità autorizzato) nell'Account A collega una policy della risorsa del catalogo dati nell'account A. Questa policy concede all'account B autorizzazioni multi-account specifiche per eseguire operazioni su una risorsa in un catalogo dell'account A.
2. Un amministratore nell'account B collega una policy IAM a un'identità IAM nell'account B che delega le autorizzazioni ricevute dall'Account A.

   L'identità nell'account B ora ha accesso alla risorsa specificata nell'account A.

   Per poter accedere alla risorsa, l'identità necessita dell'autorizzazione sia da parte del proprietario della risorsa (Account A), sia del relativo account padre (Account B).

Concedere l'accesso multi-account utilizzando un ruolo IAM

Di seguito sono riportati i passaggi generali per concedere l'accesso multi-account utilizzando un ruolo IAM:

1. Un amministratore (o un altro tipo di identità autorizzato) nell'account proprietario della risorsa (Account A) crea un ruolo IAM.
2. L'amministratore nell'Account A collega una policy al ruolo che concede autorizzazioni multi-account per accedere alla risorsa in questione.
3. L'amministratore dell'account A collega una policy di attendibilità al ruolo che identifica un'identità IAM in un account differente (Account B) come il principale che può assumere il ruolo.

Il responsabile della politica di fiducia può anche essere un responsabile del AWS servizio se si desidera concedere a un AWS servizio l'autorizzazione ad assumere il ruolo.

4. Un amministratore nell'account B ora delega le autorizzazioni a una o più identità IAM nell'account B in modo che possano assumere quel ruolo. In questo modo, consente alle identità nell'account B di accedere alla risorsa nell'account A.

Per ulteriori informazioni sull'uso di IAM per delegare le autorizzazioni, consultare Gestione degli accessi nella Guida per l'utente di IAM. Per ulteriori informazioni su utenti, gruppi, ruoli e autorizzazioni, consultare Identità (utenti, gruppi e ruoli) nella Guida per l'utente di IAM.

Per confrontare questi due approcci, consulta In che modo i ruoli IAM differiscono dalle policy basate sulla risorsa nella Guida per l'utente IAM. AWS Glue supporta entrambe le opzioni, con la limitazione che una policy della risorsa può concedere l'accesso solo alle risorse di catalogo dati.

Ad esempio, per concedere al ruolo `Dev` nell'account B l'accesso al database `db1` nell'account A, collegare al catalogo nell'account A le seguenti policy sulle risorse.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase"
      ],
      "Principal": {"AWS": [
        "arn:aws:iam::account-B-id:role/Dev"
      ]},
      "Resource": [
        "arn:aws:glue:us-east-1:account-A-id:catalog",
        "arn:aws:glue:us-east-1:account-A-id:database/db1"
      ]
    }
  ]
}
```

Inoltre, prima che B possa effettivamente accedere a `db1` nell'account A, l'account B dovrebbe collegare al ruolo `Dev` la seguente policy IAM.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:account-A-id:catalog",
        "arn:aws:glue:us-east-1:account-A-id:database/db1"
      ]
    }
  ]
}
```

Aggiunta o aggiornamento della policy della risorsa di catalogo dati

Puoi aggiungere o aggiornare la politica delle risorse di AWS Glue Data Catalog utilizzando la console, l'API o AWS Command Line Interface (AWS CLI).

> **Important**
>
> Se hai già concesso autorizzazioni multi-account dal tuo account con AWS Lake Formation, l'aggiunta o l'aggiornamento della policy della risorsa del catalogo dati richiede un passaggio aggiuntivo. Per ulteriori informazioni, consulta Gestire autorizzazioni multi-account tramite AWS Glue e Lake Formation nella Guida per gli sviluppatori di AWS Lake Formation . Per determinare se esistono concessioni multi-account di Lake Formation, utilizza l'operazione dell'API `glue:GetResourcePolicies` o la AWS CLI. Se `glue:GetResourcePolicies` restituisce policy diverse da una policy catalogo dati già esistente, allora esistono concessioni Lake Formation. Per ulteriori informazioni, consulta Visualizzazione di tutte le sovvenzioni tra account utilizzando il funzionamento dell' GetResourcePolicies API nella Guida per gli AWS Lake Formation sviluppatori.

Aggiunta o aggiornamento della policy della risorsa del catalogo dati (console)

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.

   Accedi come utente amministrativo AWS Identity and Access Management (IAM) con l'`glue:PutResourcePolicy`autorizzazione.

2. Nel pannello di navigazione scegli Impostazioni.
3. Nella pagina Data catalog settings (Impostazioni del catalogo dati), sottoPermissions (Autorizzazioni) incolla una policy della risorsa nell'area di testo. Quindi scegli Save (Salva).

   Se nella console viene visualizzato un avviso che indica che le autorizzazioni della policy saranno in aggiunta alle autorizzazioni concesse tramite Lake Formation, scegli Proceed (Continua).

Per aggiungere o aggiornare la policy della risorsa del catalogo dati (AWS CLI)

- Invia un comando `aws glue put-resource-policy`. Se esistono già concessioni per la Lake Formation, assicurati di includere l'opzione `--enable-hybrid` con il valore `'TRUE'`.

  Per esempi di utilizzo di questo comando, consulta Esempi di policy basate su risorse per AWS Glue.

Come effettuare una chiamata API multi-account

Tutte le operazioni AWS Glue Data Catalog dispongono di un campo `CatalogId`. Se sono state concesse le autorizzazioni necessarie per l'accesso a più account, un chiamante è in grado di effettuare chiamate API del catalogo dati su tutti gli account. Il chiamante passa l'ID dell'account AWS di destinazione nel `CatalogId` in modo da accedere alla risorsa in tale account di destinazione.

Se non viene specificato alcun valore `CatalogId`, AWS Glue utilizza il proprio ID account del chiamante per impostazione predefinita e la chiamata non è multi-account.

Come effettuare una chiamata ETL multi-account

Alcune API AWS Glue PySpark e Scala dispongono di un campo ID del catalogo. Se sono state PySpark concesse tutte le autorizzazioni necessarie per consentire l'accesso tra account diversi, un job ETL può effettuare chiamate Scala alle operazioni API tra più account inserendo l'ID dell'account di destinazione nel campo ID del catalogo per accedere alle risorse del Data Catalog in un AWS account di destinazione.

Se non viene specificato alcun valore ID catalogo, AWS Glue utilizza il proprio ID account del chiamante per impostazione predefinita e la chiamata non è multi-account.

Per le PySpark API che supportano, consulta. `catalog_id` Classe GlueContext Per le API Scala che supportano `catalogId`, consultare API di AWS Glue Scala GlueContext.

L'esempio seguente mostra le autorizzazioni richieste dall'assegnatario per l'esecuzione di un processo ETL. In questo esempio, *grantee-account-id*è il `catalog-id` client che esegue il job ed *grantor-account-id*è il proprietario della risorsa. Questo esempio illustra come concedere le autorizzazioni per tutte le risorse del catalogo nell'account del concedente. Per limitare l'ambito delle risorse autorizzate, è possibile fornire ARN specifici per catalogo, database, tabella e connessione.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetConnection",
                "glue:GetDatabase",
                "glue:GetTable",
                "glue:GetPartition"
            ],
            "Principal": {"AWS": ["arn:aws:iam::grantee-account-id:root"]},
            "Resource": [
                "arn:aws:glue:us-east-1:grantor-account-id:*"
            ]
        }
    ]
}
```

> **Note**
>
> Se una tabella nell'account del concedente punta a una posizione Amazon S3 che si trova anche nell'account del concedente, il ruolo IAM utilizzato per eseguire un processo ETL negli account dell'assegnatario deve avere le autorizzazioni per elencare e ottenere gli oggetti dall'account del concedente.

Poiché il client nell'Account A dispone già dell'autorizzazione per creare ed eseguire processi ETL, i passaggi di base per configurare un processo ETL per l'accesso multi-account sono i seguenti:

1. Consentire l'accesso ai dati multi-account (salta questo passaggio se l'accesso multi-account Amazon S3 è già configurato).

a. Aggiornare la policy nel bucket Amazon S3 nell'account B per consentire l'accesso multi-account dall'Account A.
b. Aggiornare la policy IAM nel bucket A per concedere l'accesso al bucket nell'account B.

2. Consentire l'accesso al catalogo dati multi-account

a. Creare o aggiornare la policy della risorsa associata al catalogo dati nell'account B per consentire l'accesso da parte di un account A.
b. Aggiornare la policy IAM nell'account A per consentire l'accesso al catalogo dati nell'account B.

Registrazione tra più account CloudTrail

Quando un job AWS Glue di estrazione, trasformazione e caricamento (ETL) accede ai dati sottostanti di una tabella Data Catalog condivisa tramite concessioni tra AWS Lake Formation account, si verifica un comportamento di registrazione aggiuntivo. AWS CloudTrail

Ai fini di questa discussione, l' AWS account che ha condiviso la tabella è l'account del proprietario e l'account con cui è stata condivisa la tabella è l'account del destinatario. Quando un processo ETL nell'account del destinatario accede ai dati nella tabella dell'account del proprietario, l' CloudTrail evento di accesso ai dati che viene aggiunto ai registri dell'account del destinatario viene copiato nei registri dell'account del proprietario. CloudTrail In questo modo gli account proprietari possono tenere traccia degli accessi ai dati da parte dei vari account destinatari. Per impostazione predefinita, gli CloudTrail eventi non includono un identificatore principale leggibile dall'uomo (ARN principale). Un amministratore nell'account destinatario può scegliere di includere l'ARN principale nei log.

Per ulteriori informazioni, consulta Registrazione tra account CloudTrail nella Guida per gli sviluppatori.AWS Lake Formation

Vedi anche

- the section called “Registrazione e monitoraggio”

Proprietà e fatturazione delle risorse multi-account

Quando un utente di un AWS account (Account A) crea una nuova risorsa, ad esempio un database in un altro account (Account B), quella risorsa è quindi di proprietà dell'Account B, l'account in cui è

stata creata. Un amministratore nell'account B ottiene automaticamente tutte le autorizzazioni per accedere alla nuova risorsa, tra cui la lettura, la scrittura e la concessione di autorizzazioni di accesso a un terzo account. L'utente nell'account A può accedere alla risorsa che ha creato solo se dispone delle autorizzazioni appropriate concesse dall'account B.

I costi di storage e gli altri costi direttamente associati alla nuova risorsa vengono fatturati all'account B, il proprietario della risorsa. Il costo delle richieste dall'utente che ha creato la risorsa viene fatturato all'account del richiedente, l'account A.

Per ulteriori informazioni sulla AWS Glue fatturazione e sui prezzi, consulta Come funzionano AWS i prezzi.

Restrizioni di accesso multi-account

L'accesso multi-account AWS Glue presenta le restrizioni seguenti:

- L'accesso tra account AWS Glue non è permesso se hai creato database e tabelle utilizzando Amazon Athena o Amazon Redshift Spectrum prima del supporto di una Regione per AWS Glue e se l'account del proprietario della risorsa non ha migrato il catalogo dati di Amazon Athena verso AWS Glue. Puoi trovare l'attuale stato di migrazione utilizzando GetCatalogImportStatus (get_catalog_import_status). Per ulteriori dettagli su come migrare un catalogo Athena verso, consulta l'aggiornamento AWS Glue alla versione contenuta AWS Glue Data Catalog step-by-step nella Amazon Athena User Guide.
- L'accesso multi-account è supportato solo per le risorse del catalogo dati, tra cui database, tabelle, funzioni definite dall'utente e connessioni.
- L'accesso multi-account al catalogo dati da Athena richiede di registrare il catalogo come risorsa Athena `DataCatalog`. Per istruzioni, consulta Registrazione di un AWS Glue Data Catalog da un altro account nella Guida per l'utente di Amazon Athena.

Risoluzione dei problemi di identità e accesso in AWS Glue

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di AWS Glue e di IAM.

Argomenti

- Non sono autorizzato a eseguire un'operazione in AWS Glue
- Non sono autorizzato a eseguire iam:PassRole

- Voglio consentire alle persone esterne al mio account Account AWS di accedere alle mie risorse AWS Glue

Non sono autorizzato a eseguire un'operazione in AWS Glue

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `glue:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 glue:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `glue:`*`GetWidget`*.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è colui che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam:PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a AWS Glue.

Alcuni Servizi AWS consentono di passare un ruolo esistente a tale servizio, invece di creare un nuovo ruolo di servizio o un ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per passare il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in AWS Glue. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è colui che ti ha fornito le credenziali di accesso.

Voglio consentire alle persone esterne al mio account Account AWS di accedere alle mie risorse AWS Glue

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 accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per capire se AWS Glue supporta queste funzionalità, consulta Funzionamento di AWS Glue con IAM.
- Per informazioni su come garantire l'accesso alle risorse negli Account AWS che possiedi, consulta Fornire l'accesso a un utente IAM in un altro Account AWS in tuo possesso nella Guida per l'utente di IAM.
- Per informazioni su come fornire l'accesso alle risorse ad Account AWS di terze parti, consulta Fornire l'accesso agli Account AWS di proprietà di terze parti nella Guida per l'utente di 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 di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consultare Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

Registrazione e monitoraggio in AWS Glue

È possibile automatizzare l'esecuzione dei processi ETL (Extract, Transform and Load, estrazione, trasformazione e caricamento). AWS Glue fornisce alcuni parametri di crawler e processi che è possibile monitorare. Dopo aver configurato il AWS Glue Data Catalog con i metadati necessari, AWS Glue fornisce le statistiche sull'integrità dell'ambiente. Puoi automatizzare la chiamata di crawler e processi con una pianificazione basata sul tempo tramite un cron. Puoi anche attivare i processi quando si attiva un trigger basato su evento.

AWS Glue è integrato con AWS CloudTrail, un servizio che offre un record delle operazioni eseguite da un utente, un ruolo o un servizio AWS in AWS Glue. Se crei un percorso, puoi abilitare la distribuzione continua di eventi CloudTrail in un bucket Amazon Simple Storage Service (Amazon S3), Amazon CloudWatch Logs e Amazon CloudWatch Events. Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta.

Amazon CloudWatch Events ti consente di automatizzare i servizi AWS e rispondere automaticamente a eventi di sistema, come i problemi relativi alla disponibilità delle applicazioni o le modifiche delle risorse. Gli eventi dei servizi AWS vengono recapitati a CloudWatch Events quasi in tempo reale. È possibile scrivere semplici regole che indichino quali eventi sono da considerare di interesse e quali operazioni automatizzate intraprendere quando si verifica un evento previsto da una regola.

Consulta anche

- Automazione di AWS Glue con CloudWatch Events
- Registrazione tra più account CloudTrail

Un aspetto importante della sicurezza nel cloud è la registrazione. È necessario configurare la registrazione in modo da non acquisire segreti e materiale riservato mentre si acquisiscono le informazioni necessarie per eseguire il debug e proteggere l'infrastruttura cloud. Assicurati di sapere con ciò che è stato registrato.

Convalida della conformità per AWS Glue

Per sapere se il Servizio AWS è coperto da programmi di conformità specifici, consulta i Servizi AWS coperti dal programma di conformità e scegli il programma di conformità desiderato. Per informazioni generali, consulta Programmi per la conformità di AWS.

È possibile scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Download di report in AWS Artifact.

La responsabilità di conformità durante l'utilizzo dei Servizi AWS è determinata dalla riservatezza dei dati, dagli obiettivi di conformità dell'azienda e dalle normative vigenti. Per semplificare il rispetto della conformità, AWS mette a disposizione le seguenti risorse:

- Guide Quick Start per la sicurezza e conformità: queste guide all'implementazione illustrano considerazioni relative all'architettura e forniscono la procedura per l'implementazione di ambienti di base su AWS incentrati sulla sicurezza e sulla conformità.
- Architetture per la sicurezza e la conformità HIPAA su Amazon Web Services: questo whitepaper descrive come le aziende possono utilizzare AWS per creare applicazioni conformi alla normativa HIPAA.

> **Note**
>
> Non tutti i Servizi AWS sono conformi ai requisiti HIPAA. Per ulteriori informazioni, consulta la sezione Riferimenti sui servizi conformi ai requisiti HIPAA.

- Risorse per la conformità AWS: una raccolta di cartelle di lavoro e guide suddivise per settore e area geografica.
- Guide alla conformità per i clienti AWS: acquisisci nozioni sul modello di responsabilità condivisa attraverso la lente di conformità. Le guide riassumono le best practice per la protezione dei Servizi AWS e tracciano 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).
- Valutazione delle risorse con le regole nella Guida per gli sviluppatori di AWS Config: il servizio AWS Config valuta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti.
- AWS Security Hub: questo Servizio AWS fornisce una visione completa dello stato di sicurezza all'interno di AWS. La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.
- AWS Audit Manager: questo Servizio AWS aiuta a effettuare un audit costante dell'utilizzo di AWS per semplificare la gestione dei rischi e della conformità alle normative e agli standard di settore.

Resilienza in AWS Glue

L'infrastruttura globale di AWS è basata su Regioni e zone di disponibilità AWS. AWS Le Regioni forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e velocità effettiva elevata. Con le zone di disponibilità, è possibile progettare e gestire le applicazioni e database che eseguono il failover automatico tra zone

di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo.

Per ulteriori informazioni sulle Regioni AWS e sulle zone di disponibilit, consulta Infrastruttura globale di AWS.

Sicurezza dell'infrastruttura in AWS Glue

In qualità di servizio gestito, AWS Glue è protetto dalle procedure di sicurezza di rete globali di AWS descritte nel whitepaper Amazon Web Services: Panoramica dei processi di sicurezza.

Utilizza le chiamate API pubblicate di AWS per accedere a AWS Glue tramite la rete. I client devono supportare Transport Layer Security (TLS) 1.0 o versioni successive. È consigliabile TLS 1.2 o versioni successive. I client devono, inoltre, supportare le suite di cifratura con PFS (Perfect Forward Secrecy), 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.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, è possibile utilizzare AWS Security Token Service (AWS STS) per generare le credenziali di sicurezza temporanee per sottoscrivere le richieste.

Argomenti

- AWS Glue ed endpoint VPC dell'interfaccia (AWS PrivateLink)
- VPC Amazon condivisi

AWS Glue ed endpoint VPC dell'interfaccia (AWS PrivateLink)

È possibile stabilire una connessione privata tra il VPC e AWS Glue creando un endpoint VPC di interfaccia. Gli endpoint dell'interfaccia sono basati su AWS PrivateLink, una tecnologia che consente di accedere privatamente alle API AWS Glue senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione AWS Direct Connect. Le istanze presenti nel VPC non richiedono indirizzi IP pubblici per comunicare con le API AWS Glue. Il traffico tra il tuo VPC e AWS Glue non esce dalla rete Amazon.

Ogni endpoint dell'interfaccia è rappresentato da una o più interfacce di rete elastiche nelle sottoreti.

Per ulteriori informazioni, consulta Endpoint VPC di interfaccia (AWS PrivateLink) nella Guida per l'utente di Amazon VPC.

Considerazioni sugli endpoint VPC di AWS Glue

Prima di impostare un endpoint VPC dell'interfaccia per AWS Glue, esamina le Proprietà e le limitazioni degli endpoint di interfaccia nella Guida per l'utente di Amazon VPC.

AWS Glue supporta l'esecuzione di chiamate a tutte le sue operazioni API all'interno del VPC.

Creazione di un endpoint di interfaccia VPC per AWS Glue

È possibile creare un endpoint VPC per il servizio AWS Glue utilizzando la console Amazon VPC o la AWS Command Line Interface (AWS CLI). Per ulteriori informazioni, consulta Creazione di un endpoint di interfaccia nella Guida per l'utente di Amazon VPC.

Crea un endpoint VPC per AWS Glue, utilizzando il seguente nome di servizio:

- com.amazonaws.*region*.glue

Se si abilita il DNS privato per l'endpoint, è possibile effettuare richieste API verso AWS Glue utilizzando il nome DNS predefinito per la regione, ad esempio `glue.us-east-1.amazonaws.com`.

Per ulteriori informazioni, consulta Accesso a un servizio tramite un endpoint di interfaccia in Guida per l'utente di Amazon VPC.

Creazione di una policy di endpoint VPC per AWS Glue.

È possibile allegare un criterio endpoint all'endpoint VPC che controlla l'accesso a AWS Glue. La policy specifica le informazioni riportate di seguito:

- Il principale che può eseguire operazioni.
- Le operazioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire operazioni.

Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC in Guida per l'utente di Amazon VPC.

Ad esempio, policy di endpoint VPC affinché AWS Glue permetta la creazione e l'aggiornamento del processo

Di seguito è riportato un esempio di una policy endpoint per AWS Glue. Se collegato a un endpoint, questo criterio concede l'accesso alle operazioni elencate AWS Glue per tutti i principali su tutte le risorse.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
             "glue:CreateJob",
             "glue:UpdateJob",
             "iam:PassRole"
         ],
         "Resource":"*"
      }
   ]
}
```

Esempio: policy di endpoint VPC per permettere l'accesso in sola lettura al catalogo dati

Di seguito è riportato un esempio di una policy endpoint per AWS Glue. Se collegato a un endpoint, questo criterio concede l'accesso alle operazioni elencate AWS Glue per tutti i principali su tutte le risorse.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetTableVersion",
                "glue:GetTableVersions",
                "glue:GetPartition",
                "glue:GetPartitions",
```

```
                "glue:BatchGetPartition",
                "glue:SearchTables"
            ],
            "Resource": "*"
        }
    ]
}
```

VPC Amazon condivisi

AWS Glue supporta i cloud privati virtuali condivisi (VPC) in Amazon Virtual Private Cloud. La condivisione Amazon VPC consente a più account AWS di creare le loro risorse dell'applicazione, ad esempio istanze Amazon EC2 e database Amazon Relational Database Service (Amazon RDS), in VPC Amazon condivisi e gestiti centralmente. In questo modello l'account che possiede il VPC (proprietario) condivide una o più sottoreti con altri account (partecipanti) che appartengono alla stessa organizzazione di AWS Organizations. Una volta condivisa una sottorete, i partecipanti possono visualizzare, creare, modificare ed eliminare le proprie risorse delle applicazioni nelle sottoreti che sono condivise con loro stessi.

In AWS Glue, per creare una connessione con una sottorete condivisa, è necessario creare un gruppo di sicurezza all'interno dell'account e collegare il gruppo di sicurezza alla sottorete condivisa.

Per ulteriori informazioni, consulta i seguenti argomenti:

- Per ulteriori informazioni, consulta Utilizzo dei VPC condivisi nella Guida per l'utente di Amazon VPC
- Che cos'è AWS Organizations? nella Guida per l'utente di AWS Organizations

Risoluzione dei problemi relativi a AWS Glue

Argomenti

Raccolta di informazioni per la risoluzione dei problemi di AWS Glue

Se si verificano errori o un comportamento imprevisto in AWS Glue e devi contattare AWS Support, devi innanzitutto raccogliere informazioni su nomi, ID e log associati all'operazione non riuscita. Con queste informazioni a disposizione, per AWS Support sarà più facile aiutarti a risolvere i problemi riscontrati.

Oltre all'ID account, raccogli le seguenti informazioni per ognuno di questi tipi di errori:

Quando un crawler ha esito negativo, raccogli le informazioni riportate di seguito.

- Il nome del crawler

  I log delle esecuzioni dei crawler si trovano in CloudWatch Logs in `/aws-glue/crawlers`.

Quando una connessione di prova ha esito negativo, raccogli le informazioni riportate di seguito.

- Nome della connessione
- ID connessione
- Stringa di connessione JDBC nel modulo `jdbc:protocol://host:port/database-name`.

  I log delle connessioni di test si trovano in CloudWatch Logs in `/aws-glue/testconnection`.

Quando un processo ha esito negativo, raccogli le informazioni riportate di seguito.

- Nome processo
- ID di esecuzione del processo nel formato `jr_xxxxx`.

I log delle esecuzioni dei processi si trovano in CloudWatch Logs in `/aws-glue/jobs`.

Risoluzione degli errori in AWS Glue per Spark

Se riscontri errori in AWS Glue, usa le soluzioni seguenti per trovare l'origine del problema e correggerlo.

> **Note**
>
> Il AWS Glue GitHub repository contiene ulteriori istruzioni per la risoluzione dei problemi nelle AWS GlueDomande frequenti.

Argomenti

Errore: risorsa non disponibile

Se AWS Glue restituisce un messaggio di risorsa non disponibile, puoi visualizzare i log o i messaggi di errore per ulteriori informazioni sul problema. Le attività seguenti descrivono i metodi generali per la risoluzione dei problemi.

- Per le connessioni e gli endpoint di sviluppo in uso, controlla che il cluster non abbia esaurito le interfacce di rete elastiche.

Errore: impossibile trovare l'endpoint S3 o il gateway NAT per il subnetId in VPC

Controlla l'ID sottorete e l'ID VPC nel messaggio per diagnosticare il problema.

- Verifica che sia configurato un endpoint VPC Amazon S3, che è necessario con AWS Glue. Quindi, controlla se il gateway NAT fa parte della configurazione. Per ulteriori informazioni, consulta Endpoint Amazon VPC per Amazon S3.

Errore: regola in entrata obbligatoria nel gruppo di sicurezza

Almeno un gruppo di sicurezza deve aprire tutte le porte di ingresso. Per limitare il traffico, è possibile limitare il gruppo di sicurezza di origine in una regola in entrata allo stesso gruppo di sicurezza.

- Per le connessioni in uso, verifica nel tuo gruppo di sicurezza la presenza di una regola in entrata autoreferenziale. Per ulteriori informazioni, consulta Impostazione dell'accesso di rete agli archivi di dati.
- Quando usi un endpoint di sviluppo, verifica nel gruppo di sicurezza la presenza di una regola in entrata autoreferenziale. Per ulteriori informazioni, consulta Impostazione dell'accesso di rete agli archivi di dati.

Errore: regola in uscita obbligatoria nel gruppo di sicurezza

Almeno un gruppo di sicurezza deve aprire tutte le porte di uscita. Per limitare il traffico, è possibile limitare il gruppo di sicurezza di origine in una regola in uscita allo stesso gruppo di sicurezza.

- Per le connessioni in uso, verifica nel tuo gruppo di sicurezza la presenza di una regola in uscita autoreferenziale. Per ulteriori informazioni, consulta Impostazione dell'accesso di rete agli archivi di dati.
- Quando usi un endpoint di sviluppo, verifica nel gruppo di sicurezza la presenza di una regola in uscita autoreferenziale. Per ulteriori informazioni, consulta Impostazione dell'accesso di rete agli archivi di dati.

Errore: l'esecuzione del processo non è riuscita perché al ruolo passato devono essere assegnate le autorizzazioni per usare il ruolo per il servizio AWS Glue

L'utente che definisce un processo deve avere l'autorizzazione `iam:PassRole` per AWS Glue.

- Quando un utente crea un processo AWS Glue, verifica che il ruolo dell'utente contenga una policy che include `iam:PassRole` AWS Glue. Per ulteriori informazioni, consulta Fase 3: Collegamento di una policy agli utenti o ai gruppi che accedono a AWS Glue.

Errore: DescribeVpcEndpoints azione non autorizzata. impossibile convalidare l'ID VPC vpc-id

- Controlla che nella policy passata a AWS Glue sia presente l'autorizzazione `ec2:DescribeVpcEndpoints`.

Errore: DescribeRouteTables azione non autorizzata. impossibile convalidare l'id di sottorete: Subnet-ID in VPC id: vpc-id

- Controlla che nella policy passata a AWS Glue sia presente l'autorizzazione `ec2:DescribeRouteTables`.

Errore: chiamata a ec2 non riuscita: DescribeSubnets

- Controlla che nella policy passata a AWS Glue sia presente l'autorizzazione `ec2:DescribeSubnets`.

Errore: chiamata a ec2 non riuscita: DescribeSecurityGroups

- Controlla che nella policy passata a AWS Glue sia presente l'autorizzazione `ec2:DescribeSecurityGroups`.

Errore: impossibile trovare la sottorete per la zona di disponibilità

- La zona di disponibilità potrebbe non essere disponibile per AWS Glue. Crea e utilizza una nuova sottorete in una zona di disponibilità diversa da quella specificata nel messaggio.

Errore: eccezione dell'esecuzione del processo durante la scrittura in una destinazione JDBC

Quando esegui un processo che scrive in una destinazione JDBC, il processo potrebbe riscontrare errori nei seguenti scenari:

- Se il processo scrive in una tabella di Microsoft SQL Server e la tabella ha colonne definite di tipo `Boolean`, la tabella deve essere predefinita nel database SQL Server. Quando definisci il processo nella console AWS Glue usando una destinazione SQL Server con l'opzione Create tables in your data target (Crea tabelle nella destinazione dati), non mappare eventuali colonne di origine a una colonna di destinazione con il tipo di dati `Boolean`. Potresti riscontrare un errore durante l'esecuzione del processo.

  Puoi evitare gli errori seguendo questa procedura:

- Scegli una tabella esistente con la colonna Boolean (Booleano).
- Modifica la trasformazione `ApplyMapping` e mappa la colonna Boolean (Booleano) nell'origine a un numero o una stringa della destinazione.
- Modifica la trasformazione `ApplyMapping` per rimuovere la colonna Boolean (Booleano) dall'origine.

- Se il processo scrive in una tabella Oracle, potrebbe essere necessario regolare la lunghezza dei nomi degli oggetti Oracle. In alcune versioni di Oracle, la lunghezza massima degli identificatori è limitata a 30 byte o 128 byte. Questo limite riguarda i nomi di tabella e di colonna dei datastore di destinazione di Oracle.

  Puoi evitare gli errori seguendo questa procedura:

  - Denomina le tabelle di destinazione di Oracle entro il limite della tua versione.
  - I nomi di colonna predefiniti sono generati dai nomi dei campi nei dati. Per gestire il caso in cui i nomi di colonna sono più lunghi del limite, utilizza le trasformazioni `ApplyMapping` o `RenameField` per modificare il nome della colonna in modo che sia entro il limite.

Errore: timeout di Amazon S3

Se AWS Glue restituisce un errore di timeout della connessione, è possibile che stia tentando di accedere a un bucket Amazon S3 in un'altra regione AWS.

- Un endpoint VPC Amazon S3 può instradare il traffico solo verso i bucket all'interno di una regione AWS. Se hai bisogno di connetterti ai bucket di altre regioni, una possibile soluzione alternativa è quella di utilizzare un gateway NAT. Per ulteriori informazioni, consulta Gateway NAT.

Errore: accesso ad Amazon S3 negato

Se AWS Glue restituisce un errore di accesso negato a un bucket o a un oggetto Amazon S3, è possibile che il ruolo IAM fornito non disponga di una policy con l'autorizzazione per il datastore.

- Un processo ETL deve avere accesso a un datastore Amazon S3 usato come origine o destinazione. Un crawler deve avere accesso a un datastore Amazon S3 in cui viene eseguito il crawling. Per ulteriori informazioni, consulta Fase 2: creare un ruolo IAM per AWS Glue.

Errore: l'ID chiave di accesso Amazon S3 non esiste

Se AWS Glue restituisce un errore di ID chiave di accesso inesistente durante l'esecuzione di un processo, il motivo può essere uno dei seguenti:

- Un processo ETL usa un ruolo IAM per accedere ai datastore. Verifica che il ruolo IAM per il processo non sia stato eliminato prima dell'inizio del processo.
- Un ruolo IAM contiene le autorizzazioni per accedere ai tuoi datastore. Verifica che qualsiasi policy Amazon S3 collegata contenente `s3:ListBucket` sia corretta.

Errore: l'esecuzione del processo restituisce un errore durante l'accesso ad Amazon S3 con un URI **`s3a://`**

Se un processo restituisce un errore, ad esempio Failed to parse XML document with handler class (Impossibile analizzare il documento XML con classe del gestore), potrebbe essere a causa di un errore nel tentativo di elencare centinaia di file utilizzando un URI `s3a://`. Accedi al datastore utilizzando un URI `s3://`. La traccia di eccezione seguente evidenzia gli errori da cercare:

```
1. com.amazonaws.SdkClientException: Failed to parse XML document with handler class
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser$ListBucketHandler
2. at
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser.parseXmlInputStream(XmlRespons
3. at
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser.parseListBucketObjectsResponse
4. at com.amazonaws.services.s3.model.transform.Unmarshallers
$ListObjectsUnmarshaller.unmarshall(Unmarshallers.java:70)
5. at com.amazonaws.services.s3.model.transform.Unmarshallers
$ListObjectsUnmarshaller.unmarshall(Unmarshallers.java:59)
6. at
 com.amazonaws.services.s3.internal.S3XmlResponseHandler.handle(S3XmlResponseHandler.java:62)
7. at
 com.amazonaws.services.s3.internal.S3XmlResponseHandler.handle(S3XmlResponseHandler.java:31)
8. at
 com.amazonaws.http.response.AwsResponseHandlerAdapter.handle(AwsResponseHandlerAdapter.java:70
9. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.handleResponse(AmazonHttpClient.java:1554)
10. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeOneRequest(AmazonHttpClient.java:1272)
```

```
11. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeHelper(AmazonHttpClient.java:1056)
12. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.doExecute(AmazonHttpClient.java:743)
13. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeWithTimer(AmazonHttpClient.java:717)
14. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.execute(AmazonHttpClient.java:699)
15. at com.amazonaws.http.AmazonHttpClient$RequestExecutor.access
$500(AmazonHttpClient.java:667)
16. at com.amazonaws.http.AmazonHttpClient
$RequestExecutionBuilderImpl.execute(AmazonHttpClient.java:649)
17. at com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:513)
18. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4325)
19. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4272)
20. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4266)
21. at com.amazonaws.services.s3.AmazonS3Client.listObjects(AmazonS3Client.java:834)
22. at org.apache.hadoop.fs.s3a.S3AFileSystem.getFileStatus(S3AFileSystem.java:971)
23. at
 org.apache.hadoop.fs.s3a.S3AFileSystem.deleteUnnecessaryFakeDirectories(S3AFileSystem.java:115
24. at org.apache.hadoop.fs.s3a.S3AFileSystem.finishedWrite(S3AFileSystem.java:1144)
25. at org.apache.hadoop.fs.s3a.S3AOutputStream.close(S3AOutputStream.java:142)
26. at org.apache.hadoop.fs.FSDataOutputStream
$PositionCache.close(FSDataOutputStream.java:74)
27. at org.apache.hadoop.fs.FSDataOutputStream.close(FSDataOutputStream.java:108)
28. at org.apache.parquet.hadoop.ParquetFileWriter.end(ParquetFileWriter.java:467)
29. at
 org.apache.parquet.hadoop.InternalParquetRecordWriter.close(InternalParquetRecordWriter.java:1
30. at
 org.apache.parquet.hadoop.ParquetRecordWriter.close(ParquetRecordWriter.java:112)
31. at
 org.apache.spark.sql.execution.datasources.parquet.ParquetOutputWriter.close(ParquetOutputWrit
32. at org.apache.spark.sql.execution.datasources.FileFormatWriter
$SingleDirectoryWriteTask.releaseResources(FileFormatWriter.scala:252)
33. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun
$org$apache$spark$sql$execution$datasources$FileFormatWriter$$executeTask
$3.apply(FileFormatWriter.scala:191)
34. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun
$org$apache$spark$sql$execution$datasources$FileFormatWriter$$executeTask
$3.apply(FileFormatWriter.scala:188)
35. at org.apache.spark.util.Utils
$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1341)
36. at org.apache.spark.sql.execution.datasources.FileFormatWriter$.org$apache$spark
$sql$execution$datasources$FileFormatWriter$$executeTask(FileFormatWriter.scala:193)
```

```
37. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun$write$1$
$anonfun$3.apply(FileFormatWriter.scala:129)
38. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun$write$1$
$anonfun$3.apply(FileFormatWriter.scala:128)
39. at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:87)
40. at org.apache.spark.scheduler.Task.run(Task.scala:99)
41. at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:282)
42. at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
43. at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
44. at java.lang.Thread.run(Thread.java:748)
```

Errore: token di servizio Amazon S3 scaduto

Durante il trasferimento di dati da e verso Amazon Redshift, vengono utilizzate le credenziali Amazon S3 temporanee che scadono dopo un'ora. Se stai eseguendo un processo lungo, potrebbe non riuscire. Per ulteriori informazioni su come configurare i processi lunghi per trasferire i dati da e verso Amazon Redshift, consulta aws-glue-programming-etl-connect-redshift-home.

Errore: non è stato trovato alcun DNS privato per l'interfaccia di rete

Se un processo ha esito negativo o un endpoint di sviluppo non è in grado di effettuare il provisioning, potrebbe essere a causa di un problema della configurazione della rete.

- Se utilizzi il DNS fornito da Amazon, il valore di `enableDnsHostnames` deve essere impostato su "true". Per ulteriori informazioni, consulta DNS.

Errore: provisioning dell'endpoint di sviluppo non riuscito

Se AWS Glue non è in grado di effettuare il provisioning di un endpoint di sviluppo, la causa potrebbe essere un problema nella configurazione di rete.

- Quando definisci un endpoint di sviluppo, il VPC, la sottorete e i gruppi di sicurezza vengono convalidati per confermare che soddisfano determinati requisiti.
- Se hai fornito la chiave pubblica SSH opzionale, controlla che si tratti di una chiave pubblica SSH valida.
- Controlla nella console VPC che il tuo VPC utilizzi un set di opzioni DHCP valido. Per maggiori informazioni, consulta Set di opzioni DHCP.
- Se il cluster rimane nello stato PROVISIONING, contatta AWS Support.

Errore: server notebook CREATE_FAILED

Se AWS Glue non è in grado di creare il server notebook per un endpoint di sviluppo, la causa potrebbe essere uno dei problemi seguenti:

- AWS Glue passa un ruolo IAM ad Amazon EC2 quando configura il server notebook. Il ruolo IAM deve avere una relazione di trust per Amazon EC2.
- Il ruolo IAM deve disporre di un profilo dell'istanza con lo stesso nome. Quando crei il ruolo per Amazon EC2 con la console IAM, il profilo dell'istanza con lo stesso nome viene creato automaticamente. Verifica nel log la presenza di un errore relativo al nome del profilo dell'istanza `iamInstanceProfile.name` che non è valido. Per ulteriori informazioni, consulta Utilizzo dei profili di istanza.
- Verifica che il tuo ruolo sia autorizzato ad accedere ai bucket `aws-glue*` nella policy che passi per creare il server notebook.

Errore: impossibile avviare il notebook locale

Se il notebook locale non si avvia e segnala errori indicanti che una directory o una cartella non viene trovata, potrebbe essere a causa di uno dei seguenti problemi:

- Se utilizzi Microsoft Windows, accertati che la variabile di ambiente `JAVA_HOME` punti alla directory di Java corretta. È possibile aggiornare Java senza aggiornare questa variabile; se fanno riferimento a una cartella che non esiste più, i notebook Jupyter non riescono ad avviarsi.

Errore: esecuzione del crawler non riuscita

Se AWS Glue non è in grado di eseguire un crawler per catalogare i dati, la causa potrebbe essere uno dei motivi seguenti. Per prima cosa controlla se è presente un errore nell'elenco dei crawler della console AWS Glue. Controlla se è presente un'icona con il punto esclamativo accanto al nome del crawler e passa il puntatore sopra l'icona per visualizzare i messaggi associati.

- Controlla i log del crawler eseguito nella sezione Logs in CloudWatch . `/aws-glue/crawlers`

Errore: le partizioni non sono state aggiornate

Nel caso in cui le partizioni non siano state aggiornate nel Data Catalog durante l'esecuzione di un job ETL, queste istruzioni di registro della `DataSink` classe nei log possono essere utili: CloudWatch

- "`Attempting to fast-forward updates to the Catalog - nameSpace:`" — Mostra quale database, tabella e catalogId il processo sta tentando di modificare. Se questa istruzione non è presente qui, verifica se `enableUpdateCatalog` è impostato su true e se è stato correttamente passato come parametro `getSink()` o in `additional_options`.
- "`Schema change policy behavior:`" — Mostra il valore `updateBehavior` dello schema passato.
- "`Schemas qualify (schema compare):`" — Sarà vero o falso.
- "`Schemas qualify (case-insensitive compare):`" — Sarà vero o falso.
- Se entrambi sono falsi e il tuo non `updateBehavior` è impostato su `UPDATE_IN_DATABASE`, lo DynamicFrame schema deve essere identico o contenere un sottoinsieme delle colonne visualizzate nello schema della tabella Data Catalog.

Per ulteriori informazioni sull'aggiornamento delle partizioni, consulta la pagina Creazione di tabelle, aggiornamento dello schema e aggiunta di nuove partizioni nel catalogo dati da processi ETL AWS Glue.

Errore: aggiornamento del segnalibro del processo non riuscito a causa della mancata corrispondenza delle versioni

Prova a parametrizzare i processi AWS Glue per applicare la stessa trasformazione/logica su set di dati diversi in Amazon S3. Desideri tenere traccia dei file elaborati nelle posizioni fornite. Quando esegui lo stesso processo sullo stesso bucket di origine e scrivi contemporaneamente nella stessa destinazione o in una destinazione diversa (simultaneità >1), il processo ha esito negativo con il seguente errore:

```
py4j.protocol.Py4JJavaError: An error occurred while
 callingz:com.amazonaws.services.glue.util.Job.commit.:com.amazonaws.services.gluejobexecutor.m
 Continuation update failed due to version mismatch. Expected version 2 but found
 version 3
```

Soluzione: imposta la simultaneità su 1 o non eseguire il processo contemporaneamente.

Al momento i segnalibri di AWS Glue non supportano le esecuzioni simultanee e pertanto i commit non riusciranno.

Errore: un processo sta rielaborando i dati mentre i segnalibri del processo sono abilitati

Potrebbero esserci dei casi in cui vengono abilitati i segnalibri del processo AWS Glue, ma il processo ETL rielabora i dati già elaborati in un'esecuzione precedente. Controlla queste cause comuni di questo errore:

Max concorrenza

Assicurati che il numero massimo di esecuzioni simultanee consentite per il processo sia 1. Per ulteriori informazioni, consulta la discussione sulla concorrenza massima in Aggiunta di processi in AWS Glue. Quando si dispone di più processi simultanei e la concorrenza massima è impostata su 1, il segnalibro del processo non funziona correttamente.

Oggetto del processo mancante

Assicurati che lo script di esecuzione del lavoro termini con il seguente commit:

```
job.commit()
```

Quando si include questo oggetto, AWS Glue registra il timestamp e il percorso dell'esecuzione del processo. Se si esegue nuovamente il processo con lo stesso percorso, AWS Glue elabora solo i nuovi file. Se non includi questo oggetto e i segnalibri del processo sono abilitati, il processo rielabora i file già elaborati con i nuovi file e crea la ridondanza nel datastore di destinazione del processo.

Parametro del contesto di trasformazione mancante

Il contesto di trasformazione è un parametro facoltativo nella classe `GlueContext`, ma i segnalibri non funzionano se non lo includi. Per risolvere questo errore, aggiungi il parametro del contesto di trasformazione quando crei il DynamicFrame, come illustrato di seguito:

```
sample_dynF=create_dynamic_frame_from_catalog(database,
 table_name,transformation_ctx="sample_dynF")
```

Origine input

Se si sta usando un database relazionale (una connessione JDBC) per l'origine input, i segnalibri del processo funzionano solo se le chiavi primarie della tabella sono in ordine sequenziale. I segnalibri del processo funzionano per le nuove righe, ma non per le righe aggiornate. Questo perché i

segnalibri processo cercano le chiavi primarie, già esistenti. Questo non si applica se l'origine di input è Amazon Simple Storage Service (Amazon S3).

Ora ultima modifica

Per origini di input Amazon S3, i segnalibri del processo controllano l'ora dell'ultima modifica piuttosto che i nomi dei file, per verificare gli oggetti da rielaborare. Se i dati dell'origine di input sono stati modificati dall'ultima esecuzione del processo, i file vengono rielaborati alla nuova esecuzione del processo.

Errore: comportamento di failover tra i VPC in AWS Glue

Il seguente processo viene utilizzato per il failover dei processi in AWS Glue 4.0 e nelle versioni precedenti.

Riepilogo: viene selezionata una connessione AWS Glue al momento dell'invio di un'esecuzione del processo. Se l'esecuzione del processo riscontra dei problemi (mancanza di indirizzi IP, connettività all'origine, problema di instradamento), essa avrà esito negativo. Se sono configurati nuovi tentativi, AWS Glue ritenterà con la stessa connessione.

1. Al momento dell'invio dell'esecuzione del processo, AWS Glue seleziona la connessione in base all'ordine in cui è elencata nella configurazione del processo. Esegue una convalida e, in caso di esito positivo, AWS Glue utilizzerà quella connessione. Se una convalida ha esito negativo, AWS Glue prova la connessione successiva.
2. AWS Glue convalida la connessione come segue:
   - Verifica la presenza di ID e sottorete Amazon VPC validi.
   - Verifica l'esistenza di un gateway NAT o di un endpoint Amazon VPC.
   - Verifica che la sottorete abbia più di 0 indirizzi IP allocati.
   - Verifica che l'AZ sia integra.

   AWS Glue non è in grado di verificare la connettività al momento dell'invio dell'esecuzione del processo.
3. Per i processi che utilizzano Amazon VPC, tutti i driver e gli esecutori verranno creati nella stessa AZ con la connessione selezionata al momento dell'invio dell'esecuzione del processo.
4. Se sono configurati nuovi tentativi, AWS Glue ritenterà con la stessa connessione. Questo perché non è detto che i problemi con questa connessione durino a lungo. Se un'AZ riscontra un errore, le esecuzioni di processo esistenti (a seconda della fase in cui si trovano) in tale AZ possono

riscontrare a loro volta un errore. Un nuovo tentativo dovrebbe rilevare un errore di AZ e scegliere un'altra AZ per la nuova esecuzione.

Risolvi gli errori del crawler quando il crawler utilizza le credenziali di Lake Formation

Utilizza le informazioni seguenti per diagnosticare e risolvere vari problemi durante la configurazione del crawler che utilizza le credenziali di Lake Formation.

Errore: la posizione S3: s3://examplepath non è registrata

Affinché un crawler possa funzionare utilizzando le credenziali di Lake Formation, devi prima configurare le autorizzazioni di Lake Formation. Per risolvere questo errore, registra la posizione Amazon S3 di destinazione con Lake Formation. Per ulteriori informazioni, consulta la pagina Registrazione di una posizione Amazon S3.

Errore: l'utente/ruolo non è autorizzato ad eseguire: lakeformation:GetDataAccess sulla risorsa

Aggiungi l'autorizzazione `lakeformation:GetDataAccess` al ruolo del crawler utilizzando la console IAM o AWS CLI. Con questa autorizzazione, Lake Formation concede la richiesta di credenziali temporanee per accedere ai dati. Vedi la politicy di seguito:

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

Errore: autorizzazioni Lake Formation insufficienti su (nome del database: exampleDatabase, nome tabella: exampleTable)

Nella console di Lake Formation (https://console.aws.amazon.com/lakeformation/), concedi al ruolo del crawler le autorizzazioni di accesso (`Create`,`Describe`,`Alter`) sul database, che è

specificato come database di output. Puoi concedere le autorizzazioni anche sulla tabella. Per ulteriori informazioni, consulta Concessione delle autorizzazioni al database tramite il metodo delle risorse denominate.

Errore: autorizzazioni di Lake Formation insufficienti su s3://examplepath

1. Crawling tra più account
   a. Accedi alla console di Lake Formation (https://console.aws.amazon.com/lakeformation/) utilizzando l'account in cui è registrato il bucket Amazon S3 (account B). Concedi le autorizzazioni per la posizione dei dati all'account in cui verrà eseguito il crawler. Ciò consentirà al crawler di leggere i dati dalla posizione Amazon S3 di destinazione.
   b. Nell'account in cui viene creato il crawler (account A), concedi le autorizzazioni relative alla posizione dei dati nella posizione Amazon S3 di destinazione al ruolo IAM utilizzato per l'esecuzione del crawler, in modo che quest'ultimo possa leggere i dati dalla destinazione in Lake Formation. Per ulteriori informazioni, consulta Concessione delle autorizzazioni per la posizione dei dati (account esterno).
2. Nel crawling dell'account (il crawler e la posizione Amazon S3 sono nello stesso account): Concedi le autorizzazioni relative alla posizione dei dati al ruolo IAM utilizzato per l'esecuzione del crawler sulla posizione Amazon S3, in modo che il crawler possa leggere i dati dalla destinazione in Lake Formation. Per ulteriori informazioni, consulta la pagina Concessione delle autorizzazioni per la posizione dei dati (stesso account).

Domande frequenti sulla configurazione del crawler utilizzando le credenziali di Lake Formation

1. Come posso configurare un crawler per l'esecuzione utilizzando le credenziali di Lake Formation tramite la console AWS?

   Nella console AWS Glue (https://console.aws.amazon.com/glue/), durante la configurazione del crawler, seleziona l'opzione Use Lake Formation credentials for crawling Amazon S3 data source (Usa le credenziali di Lake Formation per eseguire il crawling dell'origine dati Amazon S3). Per il crawling tra più account, specifica l'ID Account AWS nel quale la posizione Amazon S3 di destinazione è registrata con Lake Formation. Per effettuare il crawling all'interno dell'account, il campo accountId è facoltativo.
2. Come posso configurare un crawler per l'esecuzione utilizzando le credenziali di Lake Formation tramite AWS CLI?

Durante la chiamata API `CreateCrawler`, aggiungi `LakeFormationConfiguration`:

```
"LakeFormationConfiguration": {
    "UseLakeFormationCredentials": true,
    "AccountId": "111111111111" (AWS account ID where the target Amazon S3 location
 is registered with Lake Formation)
  }
```

3. Quali sono le destinazioni supportate per un crawler che utilizza le credenziali di Lake Formation?

   Un crawler che utilizza le credenziali Lake Formation è supportato solo per Amazon S3 (crawling in un account e tra più account), per le destinazioni Catalogo dati in un account (dove la posizione sottostante è Amazon S3) e per le destinazioni Apache Iceberg.

4. Posso eseguire il crawling di più bucket Amazon S3 come parte di un singolo crawler utilizzando le credenziali di Lake Formation?

   No, per le destinazioni del crawling che utilizzano la distribuzione delle credenziali Lake Formation, le posizioni Amazon S3 sottostanti devono appartenere allo stesso bucket. Ad esempio, i clienti possono utilizzare più posizioni di destinazione (`s3://bucket1/folder1, s3://bucket1/folder2`) se sono sotto lo stesso bucket (bucket1). Specificare bucket diversi (s3://bucket1/folder1, s3://bucket2/folder2) non è supportato.

Risoluzione dei problemi relativi agli errori di AWS Glue per Ray nei log

AWS Glue fornisce l'accesso ai log che vengono emessi dai processi Ray durante l'esecuzione del processo. Se si riscontrano errori o comportamenti imprevisti nei processi Ray, raccogliere innanzitutto le informazioni dai log per determinare la causa dell'errore. Forniamo log simili anche per le sessioni interattive. I log delle sessioni sono forniti con il prefisso `/aws-glue/ray/sessions`.

Le righe di log vengono inviate a CloudWatch in tempo reale man mano che il processo viene eseguito. Le istruzioni di stampa vengono aggiunte ai log di CloudWatch al termine dell'esecuzione. I log vengono conservati per due settimane dopo l'esecuzione di un processo.

Ispezione dei log dei processi Ray

Quando un processo non riesce, raccogli il nome e l'ID di esecuzione del processo. Sono disponibili nella console AWS Glue. Accedi alla pagina del processo, quindi passa alla scheda Runs (Esecuzioni). I log dei processi Ray sono archiviati nei seguenti gruppi di log CloudWatch dedicati.

- `/aws-glue/ray/jobs/script-log/`: archivia i log emessi dallo script principale di Ray.
- `/aws-glue/ray/jobs/ray-monitor-log/`: archivia i log emessi dal processo autoscaler di Ray. Questi log vengono generati per il nodo principale e non per altri nodi worker.
- `/aws-glue/ray/jobs/ray-gcs-logs/`: archivia i log emessi dal processo GCS (global control store). Questi log vengono generati per il nodo principale e non per altri nodi worker.
- `/aws-glue/ray/jobs/ray-process-logs/`: archivia i log emessi da altri processi Ray (principalmente l'agente del pannello di controllo) in esecuzione sul nodo principale. Questi log vengono generati per il nodo principale e non per altri nodi worker.
- `/aws-glue/ray/jobs/ray-raylet-logs/`: archivia i log emessi da ciascun processo raylet. Questi log vengono raccolti in un unico flusso per ogni nodo worker, incluso il nodo principale.
- `/aws-glue/ray/jobs/ray-worker-out-logs/`: archivia i log `stdout` per ogni worker nel cluster. Questi log vengono generati per ogni nodo worker, incluso il nodo principale.
- `/aws-glue/ray/jobs/ray-worker-err-logs/`: archivia i log `stderr` per ogni worker nel cluster. Questi log vengono generati per ogni nodo worker, incluso il nodo principale.
- `/aws-glue/ray/jobs/ray-runtime-env-log/`: archivia i log relativi al processo di configurazione di Ray. Questi log vengono generati per ogni nodo worker, incluso il nodo principale.

Risoluzione degli errori relativi ai processi Ray

Per comprendere l'organizzazione dei gruppi di log di Ray e per individuare i gruppi di log che consentiranno di risolvere gli errori, è utile disporre di informazioni di base sull'architettura Ray.

In AWS Glue ETL, un worker corrisponde a un'istanza. Quando si configurano i worker per un processo AWS Glue, si impostano il tipo e la quantità di istanze dedicate al processo. Ray usa il termine worker in diversi modi.

Ray utilizza i termini nodo principale e nodo worker per distinguere le responsabilità di un'istanza all'interno di un cluster Ray. Un nodo worker Ray può ospitare processi con più attori che eseguono calcoli per ottenere il risultato del calcolo distribuito. Gli attori che eseguono una replica di una

funzione sono chiamati repliche. Gli attori di replica possono anche essere chiamati processi worker. Le repliche possono essere eseguite anche sul nodo principale, così chiamato perché esegue processi aggiuntivi per coordinare il cluster.

Ogni attore che contribuisce al calcolo genera il proprio flusso di log. Questo ci fornisce alcune informazioni dettagliate:

- Il numero minimo di processi che emettono i log potrebbe essere maggiore del numero di worker assegnati al processo. Spesso, ogni core di ogni istanza ha un attore.
- I nodi Ray principali emettono log di avvio e di gestione dei cluster. Al contrario, i nodi worker Ray emettono solo i log relativi al lavoro svolto su di essi.

Per ulteriori informazioni sull'architettura di Ray, consulta Whitepaper sull'architettura nella documentazione di Ray.

Area problematica: accesso ad Amazon S3

Controlla il messaggio di errore dell'esecuzione del processo. Se ciò non fornisce informazioni sufficienti, controlla `/aws-glue/ray/jobs/script-log/`.

Area problematica: gestione delle dipendenze PIP

Controlla `/aws-glue/ray/jobs/ray-runtime-env-log/`.

Area problematica: ispezione dei valori intermedi nel processo principale

Scrivi su `stderr` o `stdout` dal tuo script principale e recupera i log da `/aws-glue/ray/jobs/script-log/`.

Area problematica: ispezione dei valori intermedi in un processo secondario

Scrivi su `stderr` o `stdout` dalla funzione `remote`. Quindi, recupera i log da `/aws-glue/ray/jobs/ray-worker-out-logs/` o `/aws-glue/ray/jobs/ray-worker-err-logs/`. La funzione potrebbe essere stata eseguita su qualsiasi replica, quindi potrebbe essere necessario esaminare più log per trovare l'output previsto.

Area problematica: interpretazione degli indirizzi IP nei messaggi di errore

In alcune situazioni di errore, il processo potrebbe emettere un messaggio di errore contenente un indirizzo IP. Questi indirizzi IP sono temporanei e vengono utilizzati dal cluster per identificare i nodi

e per la comunicazione tra essi. I log di un nodo vengono pubblicati in un flusso di log con un suffisso univoco basato sull'indirizzo IP.

In CloudWatch, è possibile filtrare i log per controllare quelli specifici di questo indirizzo IP identificando tale suffisso. Ad esempio, dati *FAILED_IP* e *JOB_RUN_ID*, è possibile identificare il suffisso con:

```
filter @logStream like /JOB_RUN_ID/
| filter @message like /IP-/
| parse @message "IP-[*]" as ip
| filter ip like /FAILED_IP/
| fields replace(ip, ":", "_") as uIP
| stats count_distinct by uIP as logStreamSuffix
| display logStreamSuffix
```

Eccezioni di machine learning AWS Glue

In questo argomento vengono descritti i codici di errore HTTP e le stringhe per le eccezioni AWS Glue correlate al machine learning. I codici di errore e le stringhe di errore vengono forniti per ogni attività di machine learning che può verificarsi quando si esegue un'operazione. Inoltre, è possibile verificare se è possibile riprovare l'operazione che ha provocato l'errore.

CancelMLTaskRunActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”
  - “Nessuna esecuzione di attività ML trovata per [taskRunId]: nell'account [accountId] per la trasformazione [transformName].”

  OK per riprovare: No.

CreateMLTaskRunActivity

Questa attività presenta le seguenti eccezioni:

- InvalidInputException (400)

- “Errore del servizio interno a causa di un input imprevisto.”
- “Una sorgente di input della tabella AWS Glue deve essere specificata nella trasformazione.”
- “La colonna di origine di input [columnName] ha un tipo di dati non valido definito nel catalogo.”
- “Deve essere fornita esattamente una tabella di record di input.”
- “Dovrebbe specificare il nome del database.”
- “Dovrebbe specificare il nome della tabella.”
- “Lo schema non è definito nella trasformazione.”
- “Lo schema dovrebbe contenere una chiave primaria data: [primaryKey].”
- “Problema durante il recupero dello schema del catalogo dati: [message].”
- “Impossibile impostare la capacità massima e il num/tipo lavoratore contemporaneamente.”
- “Sia WorkerType che NumberOfWorkers dovrebbero essere impostati.”
- “MaxCapacity dovrebbe essere >= [maxCapacity].”
- “NumberOfWorkers dovrebbe essere >= [maxCapacity].”
- “I tentativi massimi dovrebbero essere non negativi.”
- “I parametri della ricerca di corrispondenze non sono stati impostati.”
- “È necessario specificare una chiave primaria nei parametri della ricerca di corrispondenze.”

OK per riprovare: No.

- AlreadyExistsException (400)
  - “La trasformazione con nome [transformName] esiste già.”

  OK per riprovare: No.
- IdempotentParameterMismatchException (400)
  - “La richiesta di creazione idempotent per la trasformazione [transformName] aveva parametri non corrispondenti.”

  OK per riprovare: No.
- InternalServiceException (500)
  - “Fallimento delle dipendenze.”

  OK per riprovare: Sì.
- ResourceNumberLimitExceededException (400)
  - “Il numero di trasformazioni ML ([count]) ha superato il limite di [limit] trasformazioni.”

OK per riprovare: Sì, una volta eliminata una trasformazione per fare spazio a questa nuova.

DeleteMLTransformActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [AccountID] con handle [TransformName]”

  OK per riprovare: No.

GetMLTaskRunActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”
  - “Nessuna esecuzione di attività ML trovata per [taskRunId]: nell'account [accountId] per la trasformazione [transformName].”

  OK per riprovare: No.

GetMLTaskRunsActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”
  - “Nessuna esecuzione di attività ML trovata per [taskRunId]: nell'account [accountId] per la trasformazione [transformName].”

  OK per riprovare: No.

GetMLTransformActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”

  OK per riprovare: No.

GetMLTransformsActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “L'ID dell'account non può essere vuoto.”
  - “Ordinamento non supportato per la colonna [column].”
  - “[column] non può essere vuota.”
  - “Errore del servizio interno a causa di un input imprevisto.”

  OK per riprovare: No.

GetSaveLocationForTransformArtifactActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “Tipo di artefatto non supportato [ArtifactType].”
  - “Errore del servizio interno a causa di un input imprevisto.”

  OK per riprovare: No.

getTaskRunArtifactActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”
  - “Nessuna esecuzione di attività ML trovata per [taskRunId]: nell'account [accountId] per la trasformazione [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “Nome file '[fileName]' non valido per la pubblicazione.”
  - “Impossibile recuperare artefatto per il tipo di attività [taskType].”
  - “Impossibile recuperare artefatto per [artifactType].”
  - “Errore del servizio interno a causa di un input imprevisto.”

  OK per riprovare: No.

PublishMLTransformModelActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”
  - “Un modello esistente con versione - [version] non può essere trovato per ID account - [accountId] - e id di trasformazione - [transformId].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “Nome file '[fileName]' non valido per la pubblicazione.”
  - “Segno meno iniziale non valido sulla stringa non firmata [string].”
  - “Cifra non valida alla fine di [string].”
  - “Il valore della stringa [string] supera l'intervallo di unsigned long.”
  - “Errore del servizio interno a causa di un input imprevisto.”

OK per riprovare: No.

PullLatestMLTransformModelActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “Errore del servizio interno a causa di un input imprevisto.”

  OK per riprovare: No.
- ConcurrentModificationException (400)
  - “Impossibile creare la versione del modello da addestrare a causa di inserti racing con parametri non corrispondenti.”
  - “Il modello di trasformazione ML per id di trasformazione [transformId] è obsoleto o aggiornato da un altro processo; Riprovare.”

  OK per riprovare: Sì.

PutJobMetadataForMLTransformActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”
  - “Nessuna esecuzione di attività ML trovata per [taskRunId]: nell'account [accountId] per la trasformazione [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “Errore del servizio interno a causa di un input imprevisto.”
  - “Tipo di metadati processo sconosciuto [jobType].”

- “È necessario fornire un ID di esecuzione attività per aggiornare.”

OK per riprovare: No.

StartExportLabelsTaskRunActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”
  - “Nessun set di etichette esiste per transformId [transformId] nell'ID account [accountId].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “[message].”
  - “Il percorso S3 fornito non si trova nella stessa regione della trasformazione. Regione prevista - [region], ma ottenuta - [region].”

  OK per riprovare: No.

StartImportLabelsTaskRunActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “[message].”
  - “Percorso del file di etichetta non valido.”
  - “Impossibile accedere al file di etichetta in [labelPath]. [message].”
  - “Impossibile utilizzare il ruolo IAM fornito nella trasformazione. Ruolo: [role].”
  - “File etichetta di dimensione 0 non valido.”

- “Il percorso S3 fornito non si trova nella stessa regione della trasformazione. Regione prevista - [region], ma ottenuta - [region].”

  OK per riprovare: No.
- ResourceNumberLimitExceededException (400)
  - “Il file di etichetta ha superato il limite di [limit] MB.”

  OK per riprovare: No. Considerare la possibilità di suddividere il file di etichetta in diversi file più piccoli.

StartMLEvaluationTaskRunActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “Deve essere fornita esattamente una tabella di record di input.”
  - “Dovrebbe specificare il nome del database.”
  - “Dovrebbe specificare il nome della tabella.”
  - “I parametri della ricerca di corrispondenze non sono stati impostati.”
  - “È necessario specificare una chiave primaria nei parametri della ricerca di corrispondenze.”

  OK per riprovare: No.
- MLTransformNotReadyException (400)
  - “Questa operazione può essere applicata solo a una trasformazione che si trova in uno stato READY.”

  OK per riprovare: No.
- InternalServiceException (500)
  - “Fallimento delle dipendenze.”

  OK per riprovare: Sì.
- ConcurrentRunsExceededException (400)

- “Il numero di esecuzioni attività ML [count] ha superato il limite di trasformazione delle esecuzioni di attività [limit].”
- “Il numero di esecuzioni attività ML [count] ha superato il limite di [limit] esecuzioni di attività.”

OK per riprovare: Sì, dopo aver atteso il completamento dell'esecuzione dell'attività.

StartMLLabelingSetGenerationTaskRunActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - “Deve essere fornita esattamente una tabella di record di input.”
  - “Dovrebbe specificare il nome del database.”
  - “Dovrebbe specificare il nome della tabella.”
  - “I parametri della ricerca di corrispondenze non sono stati impostati.”
  - “È necessario specificare una chiave primaria nei parametri della ricerca di corrispondenze.”

  OK per riprovare: No.
- InternalServiceException (500)
  - “Fallimento delle dipendenze.”

  OK per riprovare: Sì.
- ConcurrentRunsExceededException (400)
  - “Il numero di esecuzioni attività ML [count] ha superato il limite di trasformazione delle esecuzioni di attività [limit].”

  OK per riprovare: Sì, una volta completata l'esecuzione dell'attività.

UpdateMLTransformActivity

Questa attività presenta le seguenti eccezioni:

- EntityNotFoundException (400)
  - “Impossibile trovare MLTransform nell'account [accountId] con handle [transformName].”

  OK per riprovare: No.
- InvalidInputException (400)
  - «Esiste già un'altra trasformazione con nome [TransformName].”
  - “[message].”
  - “Il nome della trasformazione non può essere vuoto.”
  - “Impossibile impostare la capacità massima e il num/tipo lavoratore contemporaneamente.”
  - “Sia WorkerType che NumberOfWorkers dovrebbero essere impostati.”
  - “MaxCapacity dovrebbe essere >= [minMaxCapacity].»
  - “NumberOfWorkers dovrebbe essere >= [minNumWorkers].”
  - “I tentativi massimi dovrebbero essere non negativi.”
  - “Errore del servizio interno a causa di un input imprevisto.”
  - “I parametri della ricerca di corrispondenze non sono stati impostati.”
  - “È necessario specificare una chiave primaria nei parametri della ricerca di corrispondenze.”

  OK per riprovare: No.
- AlreadyExistsException (400)
  - “La trasformazione con nome [transformName] esiste già.”

  OK per riprovare: No.
- IdempotentParameterMismatchException (400)
  - “La richiesta di creazione idempotent per la trasformazione [transformName] aveva parametri non corrispondenti.”

  OK per riprovare: No.

Quote AWS Glue

È possibile contattare il AWS Support per richiedere un aumento delle quote per le Service Quotas elencate in Riferimenti generali di AWS. Salvo dove diversamente specificato, ogni quota si applica a una regione specifica. Per ulteriori informazioni, consulta Endpoint e quote per AWS Glue.

Miglioramento AWS Glue delle prestazioni

Strategia di base per l'ottimizzazione delle prestazioni

Per migliorare AWS Glue le prestazioni, potresti prendere in considerazione l'aggiornamento di alcuni parametri relativi alle AWS Glue prestazioni. Quando ci si prepara all'ottimizzazione dei parametri, utilizza la seguenti best practice:

- Determinare gli obiettivi di prestazioni prima di iniziare a identificare i problemi.
- Prima di provare di modificare i parametri di ottimizzazione, utilizza i parametri per identificare i problemi.

Per ottenere risultati più coerenti durante l'ottimizzazione di un processo, sviluppa una strategia di base per il lavoro di ottimizzazione.

In genere, l'ottimizzazione delle prestazioni viene eseguita nel seguente flusso di lavoro:

1. Determinazione degli obiettivi delle prestazioni.
2. Misurazione dei parametri.
3. Identificazione dei colli di bottiglia.
4. Riduzione dell'impatto dei colli di bottiglia.
5. Ripeti i passaggi da 2 a 4 fino a raggiungere l'obiettivo prefissato.

Strategie di ottimizzazione per il tipo di lavoro

Lavori Spark: segui le indicazioni contenute nelle migliori pratiche per l'ottimizzazione delle prestazioni per i lavori di Apache Spark su AWS Glue Prescriptive Guidance. AWS

Altri lavori: è possibile eseguire l'ottimizzazione AWS Glue per i job della shell Ray e AWS Glue Python adattando le strategie disponibili in altri ambienti di runtime.

Miglioramento delle prestazioni per i processi AWS Glue per Apache Spark

Per migliorare le prestazioni di AWS Glue per Spark, potresti prendere in considerazione l'aggiornamento di alcuni parametri relativi alle prestazioni di AWS Glue e di Spark.

Per ulteriori informazioni sulle strategie specifiche per identificare i colli di bottiglia attraverso i parametri e ridurne l'impatto, consulta Best practice per l'ottimizzazione delle prestazioni per i processi di AWS Glue per Apache nel Prontuario AWS. Questa guida introduce gli argomenti chiave applicabili ad Apache Spark in tutti gli ambienti di runtime, come l'architettura Spark e i set di dati distribuiti resilienti. Utilizzando questi argomenti, la guida guida all'implementazione di strategie specifiche di ottimizzazione delle prestazioni, come l'ottimizzazione degli shuffle e la parallelizzazione delle attività.

Puoi identificare i punti deboli configurando la visualizzazione dell'interfaccia utente di Spark. AWS Glue Per ulteriori informazioni, consulta the section called “Monitoraggio con l'interfaccia utente di Spark”.

Inoltre, AWS Glue offre funzionalità prestazionali che possono essere applicabili al tipo specifico di archivio dati a cui si connette il job. Le informazioni di riferimento sui parametri prestazionali per gli archivi dati sono disponibili inthe section called “Parametri di connessione”.

Ottimizzazione delle letture con pushdown in AWS Glue ETL

Il pushdown è una tecnica di ottimizzazione che avvicina la logica di recupero dei dati all'origine dati. L'origine potrebbe essere un database o un file system come Amazon S3. Quando si eseguono determinate operazioni direttamente sull'origine, è possibile risparmiare tempo e potenza di elaborazione evitando di trasferire tutti i dati della rete al motore Spark gestito da AWS Glue.

Un altro modo per dire ciò è che il pushdown riduce la scansione dei dati. Per ulteriori informazioni sul processo di identificazione del momento in cui questa tecnica è appropriata, consulta Ridurre la quantità di scansione dei dati nella guida Best practice per l'ottimizzazione delle prestazioni dei processi AWS per Apache Spark nel Prontuario AWS.

Il predicato pushdown sui file archiviati su Amazon S3

Quando lavori su Amazon S3 con file organizzati per prefisso, puoi filtrare i percorsi Amazon S3 di destinazione definendo un predicato pushdown. Invece di leggere il set di dati completo e applicare filtri all'interno di un `DynamicFrame`, è possibile applicare il filtro direttamente ai metadati della partizione archiviati in Catalogo dati AWS Glue. Questo approccio consente di elencare e leggere in modo selettivo solo i dati necessari. Per ulteriori informazioni su questo processo, inclusa la scrittura per partizioni in un bucket, consulta la pagina the section called “Gestione delle partizioni”.

È possibile ottenere il pushdown dei predicati in Amazon S3 utilizzando il parametro `push_down_predicate`. Prendi in considerazione un bucket in Amazon S3 partizionato per anno,

mese e giorno. Se desideri recuperare i dati dei clienti relativi al mese di giugno 2022, puoi indicare ad AWS Glue di leggere solo i percorsi Amazon S3 pertinenti. Il `push_down_predicate` in questo caso è `year='2022' and month='06'`. Mettendo tutto insieme, l'operazione di lettura può essere eseguita come segue:

Python

```
customer_records = glueContext.create_dynamic_frame.from_catalog(
    database = "customer_db",
    table_name = "customer_tbl",
    push_down_predicate = "year='2022' and month='06'"
)
```

Scala

```
val customer_records = glueContext.getCatalogSource(
database="customer_db",
tableName="customer_tbl",
pushDownPredicate="year='2022' and month='06'"
).getDynamicFrame()
```

Nello scenario precedente, `push_down_predicate` recupera un elenco di tutte le partizioni da Catalogo dati AWS Glue e le filtra prima di leggere i file Amazon S3 sottostanti. Mentre elencare le partizioni è utile nella maggior parte dei casi, questo processo può richiedere molto tempo quando si lavora con set di dati che contengono milioni di partizioni. Per risolvere questo problema e migliorare le prestazioni è possibile utilizzare l'eliminazione delle partizioni lato server. Questo viene fatto creando un indice di partizione per i dati in Catalogo dati AWS Glue. Per ulteriori informazioni sugli indici di partizione, consulta la pagina the section called “Utilizzo degli indici delle partizioni”. È quindi possibile utilizzare l'opzione `catalogPartitionPredicate` per fare riferimento all'indice. Per un esempio di recupero di partizioni con `catalogPartitionPredicate`, consulta la pagina the section called “Predicati di partizione di catalogo”.

Esecuzione del pushdown quando si utilizzano origini JDBC

Il lettore JDBC di AWS Glue utilizzato nel `GlueContext` supporta il pushdown sui database supportati fornendo query SQL personalizzate che possono essere eseguite direttamente sull'origine. Ciò può essere ottenuto impostando il parametro `sampleQuery`. La tua query di esempio può specificare quali colonne selezionare e fornire un predicato di pushdown per limitare i dati trasferiti al motore Spark.

Per impostazione predefinita, le query di esempio funzionano su un singolo nodo, il che può causare errori di processo quando si gestiscono grandi volumi di dati. Per utilizzare questa funzionalità per eseguire query sui dati su larga scala, è necessario configurare il partizionamento delle query impostando `enablePartitioningForSampleQuery` su true, che distribuirà la query su più nodi attraverso una chiave a scelta. Il partizionamento delle query richiede anche alcuni altri parametri di configurazione necessari. Per ulteriori informazioni sul partizionamento delle query, consulta la pagina the section called “Lettura in parallelo da JDBC”.

Durante l'impostazione di `enablePartitioningForSampleQuery`, AWS Glue combina il predicato di pushdown con un predicato di partizionamento durante l'esecuzione della query sul database. La `sampleQuery` deve terminare con un `AND` affinché AWS Glue aggiunga le condizioni di partizionamento. (Se non fornisci un predicato pushdown, `sampleQuery` deve terminare con un `WHERE`). Di seguito puoi trovare un esempio in cui effettuiamo il pushdown di un predicato per recuperare solo le righe il cui `id` è maggiore di 1.000. Questa `sampleQuery` restituirà solo le colonne del nome e della posizione per le righe il cui `id` è maggiore del valore specificato:

Python

```
sample_query = "select name, location from customer_tbl WHERE id>=1000 AND"
customer_records = glueContext.create_dynamic_frame.from_catalog(
    database="customer_db",
    table_name="customer_tbl",
    sample_query = "select name, location from customer_tbl WHERE id>=1000 AND",

    additional_options = {
                           "hashpartitions": 36 ,
                           "hashfield":"id",
                           "enablePartitioningForSampleQuery":True,
                           "sampleQuery":sample_query
                          }
)
```

Scala

```
val additionalOptions = Map(
        "hashpartitions" -> "36",
        "hashfield" -> "id",
        "enablePartitioningForSampleQuery" -> "true",
        "sampleQuery" -> "select name, location from customer_tbl WHERE id >= 1000
 AND"
        )
```

```
    val customer_records = glueContext.getCatalogSource(
        database="customer_db",
        tableName="customer_tbl").getDynamicFrame()
```

> **Note**
>
> Se `customer_tbl` ha un nome diverso nel tuo Data Catalog e nel datastore sottostante, devi fornire il nome della tabella sottostante in sample_query, poiché la query viene passata al datastore sottostante.

È possibile eseguire query sulle tabelle JDBC anche senza l'integrazione con Catalogo dati AWS Glue. Invece di fornire nome utente e password come parametri del metodo, è possibile riutilizzare le credenziali di una connessione preesistente fornendo `useConnectionProperties` e `connectionName`. In questo esempio, recuperiamo le credenziali da una connessione chiamata `my_postgre_connection`.

Python

```
connection_options_dict = {
    "useConnectionProperties": True,
    "connectionName": "my_postgre_connection",
    "dbtable":"customer_tbl",
    "sampleQuery":"select name, location from customer_tbl WHERE id>=1000 AND",
    "enablePartitioningForSampleQuery":True,
    "hashfield":"id",
    "hashpartitions":36
    }

customer_records = glueContext.create_dynamic_frame.from_options(
    connection_type="postgresql",
    connection_options=connection_options_dict
    )
```

Scala

```
val connectionOptionsJson = """
      {
```

```
        "useConnectionProperties": true,
        "connectionName": "my_postgre_connection",
        "dbtable": "customer_tbl",
        "sampleQuery": "select name, location from customer_tbl WHERE id>=1000 AND",
        "enablePartitioningForSampleQuery" : true,
        "hashfield" : "id",
        "hashpartitions" : 36
      }
    """

    val connectionOptions = new JsonOptions(connectionOptionsJson)

    val dyf = glueContext.getSource("postgresql",
connectionOptions).getDynamicFrame()
```

Note e limitazioni sul pushdown in AWS Glue

Il pushdown, come concetto, è applicabile alla lettura da origini non in streaming. AWS Glue supporta una varietà di origini: la capacità di eseguire il pushdown dipende dall'origine e dal connettore.

- Quando ti connetti a Snowflake, puoi utilizzare l'opzione `query`. Funzionalità simili esistono nel connettore Redshift in AWS Glue 4.0 e versioni successive. Per ulteriori informazioni sulla lettura da Snowflake con `query`, consulta la pagina the section called “Lettura da Snowflake”.
- Il lettore DynamoDB ETL non supporta filtri o predicati pushdown. Inoltre, MongoDB e DocumentDB non supportano questo tipo di funzionalità.
- Quando si leggono dati in formati di tabelle aperte archiviati in Amazon S3, il metodo di partizionamento dei file in Amazon S3 da solo non è più sufficiente. Per leggere e scrivere da partizioni utilizzando formati di tabelle aperte, consulta la documentazione relativa al formato.
- DynamicFrame i metodi non eseguono il pushdown di proiezione di Amazon S3. Tutte le colonne verranno lette dai file che soddisfanno il filtro dei predicati.
- Quando si utilizzano i connettori `custom.jdbc` in AWS Glue, la capacità di eseguire il pushdown dipende dall'origine e dal connettore. Consulta la documentazione del connettore appropriata per verificare se e come supporta il pushdown in AWS Glue.

Utilizzo di Auto Scaling per AWS Glue

Auto Scaling è disponibile per il tuo ETL AWS Glue e processi di streaming con AWS Glue versione 3.0 o successive.

Con Auto Scaling abilitato, otterrai i seguenti vantaggi:

- AWS Glue aggiunge automaticamente e rimuove i lavoratori del cluster a seconda del parallelismo in ogni fase o microbatch del processo in esecuzione.
- Elimina la necessità di sperimentare e decidere il numero di dipendenti da assegnare per i tuoi processi ETL AWS Glue.
- Se si sceglie il numero massimo di worker, AWS Glue sceglierà le risorse della dimensione giusta per il carico di lavoro.
- È possibile vedere come cambia la dimensione del cluster durante l'esecuzione del processo esaminando le CloudWatch metriche nella pagina dei dettagli dell'esecuzione del lavoro in AWS Glue Studio.

Auto Scaling per i processi ETL AWS Glue e i processi di streaming consente il dimensionamento verso l'alto e la riduzione verticale on demand delle risorse di calcolo dei tuoi processi AWS Glue. Il dimensionamento verticale on demand consente di allocare solo le risorse di calcolo richieste inizialmente all'avvio dell'esecuzione del processo e anche di effettuare il provisioning delle risorse richieste in base alla domanda durante il processo.

Auto Scaling supporta anche la riduzione dinamica verticale delle risorse dei processi AWS Glue nel corso di un processo. Durante l'esecuzione di un processo, quando vengono richiesti più esecutori dall'applicazione Spark, verrà aggiunto al cluster un numero maggiore di dipendenti. Quando l'esecutore sarà inattivo e non avrà attività di calcolo in corso, l'esecutore e il dipendente corrispondente verranno rimossi.

Scenari comuni in cui Auto Scaling aiuta a ridurre i costi e l'utilizzo delle applicazioni Spark includono un driver Spark che elenca un numero elevato di file in Amazon S3 o che esegue un carico mentre gli esecutori sono inattivi, le fasi di Spark sono in esecuzione con pochi esecutori a causa dell'overprovisioning e delle distorsioni dei dati o della domanda di calcolo irregolare nelle fasi di Spark.

Requisiti

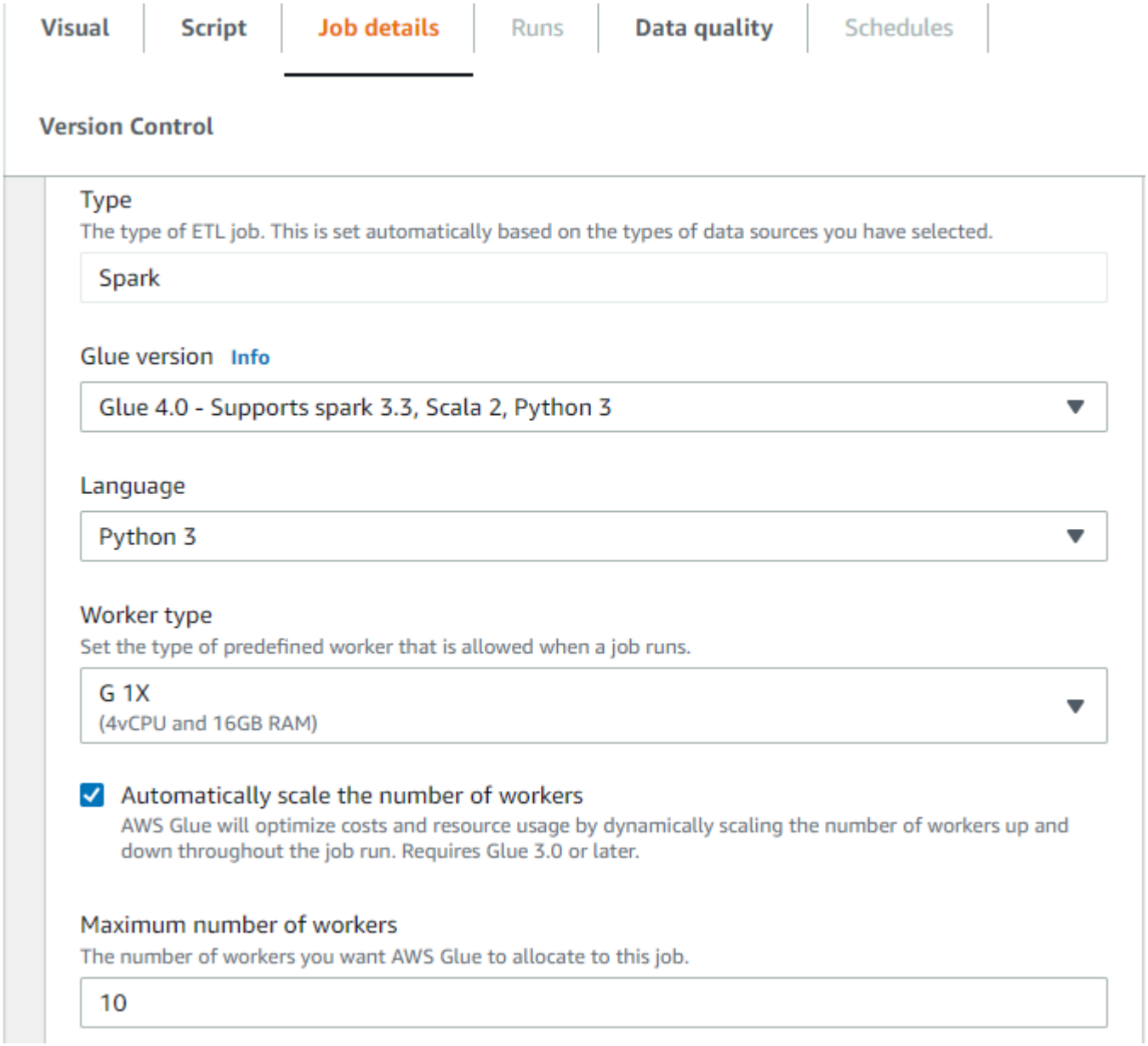
Dimensionamento automatico è disponibile solo per AWS Glue versione 3.0 o successiva. Per utilizzare Dimensionamento automatico, consulta la Guida alla migrazione per migrare i processi esistenti a AWS Glue 3.0 o versioni successive oppure creare nuovi processi con AWS Glue 3.0 o versioni successive.

Il dimensionamento automatico è disponibile per i processi AWS Glue con i tipi di worker `G.1X`, `G.2X`, `G.4X`, `G.8X` o `G.025X` (solo per i processi di streaming). Le DPU standard non sono supportate.

Abilitazione dell'Auto Scaling in AWS Glue Studio

Sulla scheda Job details (Dettagli processo) in AWS Glue Studio, scegli il tipo Spark o Spark Streaming e Glue version (Versione di Glue) come **`Glue 3.0`** o **`Glue 4.0`**. In seguito, una casella di controllo verrà visualizzata sotto Worker type (tipo di worker).

- Seleziona l'opzione Dimensiona automaticamente il numero di worker.
- Imposta la proprietà Numero massimo di dipendenti per definire il numero massimo di dipendenti che possono essere ceduti all'esecuzione del processo.



Abilitazione di Auto Scaling con il CLI o SDK di AWS

Per abilitare Auto Scaling dal CLI di AWS per eseguire il processo, eseguire`start-job-run` con la configurazione seguente:

```
{
    "JobName": "<your job name>",
    "Arguments": {
        "--enable-auto-scaling": "true"
    },
    "WorkerType": "G.2X", // G.1X and G.2X are allowed for Auto Scaling Jobs
```

```
    "NumberOfWorkers": 20, // represents Maximum number of workers
    ...other job run configurations...
}
```

Una volta terminata l'esecuzione del processo ETL, puoi anche chiamare`get-job-run` per verificare l'effettivo utilizzo delle risorse del processo eseguito in secondi DPU. Nota: il nuovo campo DPUSeconds verrà visualizzato solo per i processi batch su AWS Glue 3.0 o versioni successive abilitati con Dimensionamento automatico. Questo campo non è supportato per i processi di streaming.

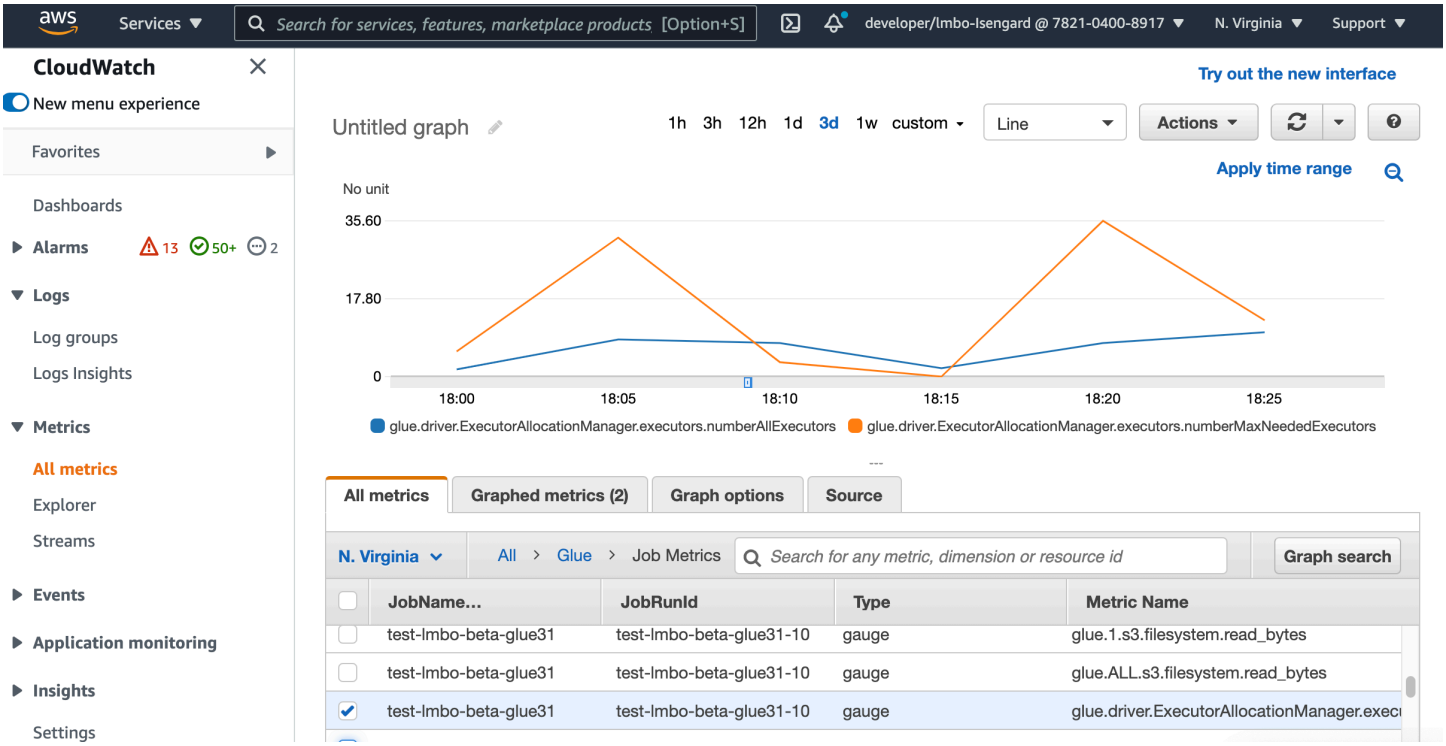
```
$ aws glue get-job-run --job-name your-job-name --run-id jr_xx --endpoint https://
glue.us-east-1.amazonaws.com --region us-east-1
{
    "JobRun": {
        ...
        "GlueVersion": "3.0",
        "DPUSeconds": 386.0
    }
}
```

È inoltre possibile configurare le esecuzioni dei processi con Auto Scaling utilizzando l'SDKAWS Glue con la stessa configurazione.

Monitoraggio dell'Auto Scaling con i parametri di Amazon CloudWatch

Le metriche dell' CloudWatch executor sono disponibili per i job AWS Glue 3.0 o versioni successive se abiliti Auto Scaling. I parametri possono essere impiegati per monitorare la domanda e l'utilizzo ottimizzato degli esecutori nelle applicazioni Spark abilitate con Auto Scaling. Per ulteriori informazioni, consulta Monitoraggio di AWS Glue con i parametri di Amazon CloudWatch.

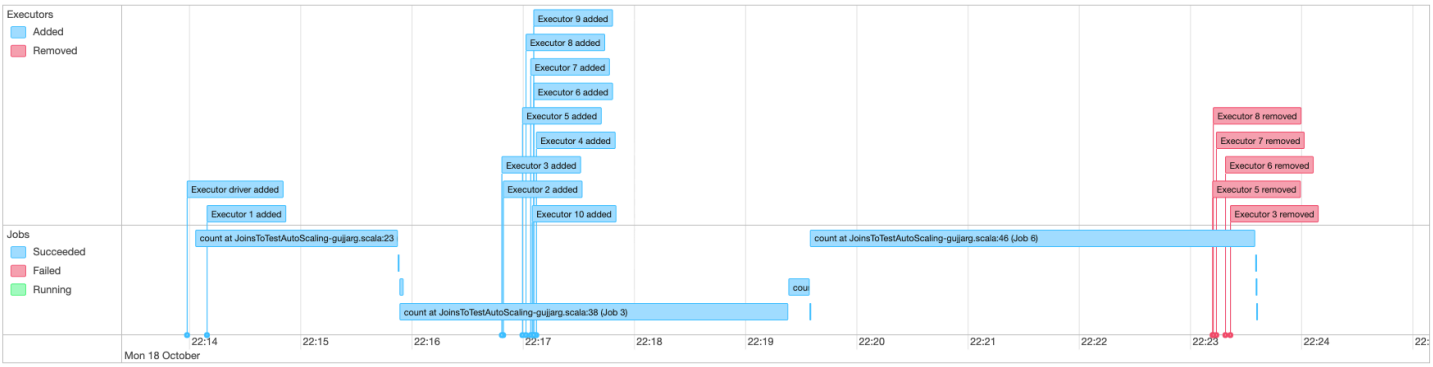
- colla.driver. ExecutorAllocationManager.esecutori. numberAllExecutors
- colla.driver. ExecutorAllocationManager.esecutori. numberMaxNeededEsecutori



Per ulteriori dettagli su questi parametri, consulta Monitoraggio per la pianificazione della capacità DPU.

Monitoraggio di Auto Scaling con interfaccia utente di Spark

Con Auto Scaling abilitato, puoi anche monitorare gli esecutori aggiunti e rimossi con il dimensionamento automatico verso l'alto e la riduzione verticale in base alla domanda nei tuoi processi AWS Glue l'interfaccia utente di Spark Glue. Per ulteriori informazioni, consulta Abilitazione dell'interfaccia utente Web di Apache Spark per processi AWS Glue.



Monitoraggio dell'utilizzo della DPU di esecuzione del processo Auto Scaling

È possibile utilizzare la Vista esecuzione dei processi AWS Glue Studio per controllare l'utilizzo della DPU da parte dei processi di dimensionamento automatico.

1. Scegli Monitoraggio dal riquadro di navigazione di AWS Glue Studio. Viene visualizzata la pagina Monitoraggio.
2. Scorri in basso fino all'elenco Job runs (Esecuzioni processo).
3. Passa all'esecuzione del processo che ti interessa e scorri fino alla colonna ore DPU per verificare l'utilizzo per l'esecuzione specifica del processo.

Limitazioni

AWS GlueLo streaming Auto Scaling attualmente non supporta un DataFrame join in streaming con un elemento statico DataFrame creato all'esterno di. `ForEachBatch` Una statica DataFrame creata all'interno di `ForEachBatch` funzionerà come previsto.

Partizionamento del carico di lavoro con esecuzione delimitata

Gli errori nelle applicazioni Spark derivano in genere da script Spark inefficienti, esecuzione in memoria di trasformazioni su larga scala e anomalie del set di dati. Esistono molte ragioni che possono causare problemi di memoria del driver o dell'executor, ad esempio una differenza di dati, un elenco di troppi oggetti o una riproduzione casuale di dati di grandi dimensioni. Questi spesso si verificano nell'elaborazione di enormi quantità di dati backlog con Spark.

AWS Glue consente di risolvere i problemi di eccezione di memoria e semplificare l'elaborazione ETL con il partizionamento del carico di lavoro. Quando il partizionamento del carico di lavoro è abilitato, ogni esecuzione del processo ETL seleziona solo i dati non elaborati, con il limite superiore impostato sulla dimensione del set di dati o sul numero di file da elaborare nell'esecuzione del processo. Le future esecuzioni dei processi elaboreranno i dati rimanenti. Ad esempio, se sono presenti 1000 file da elaborare, è possibile impostare il numero di file su 500 e dividerli in due esecuzioni del processo.

Il partizionamento del carico di lavoro è supportato solo per le origini dati di Amazon S3.

Abilitazione del partizionamento del carico di lavoro

È possibile abilitare l'esecuzione delimitata impostando manualmente le opzioni nello script o aggiungendo le proprietà della tabella catalogo.

Per abilitare il partizionamento del carico di lavoro con esecuzione delimitata nello script:

1. Per evitare di rielaborare i dati, abilitare i segnalibri di processo nel nuovo processo o nel processo esistente. Per ulteriori informazioni, consulta Monitoraggio dei dati elaborati mediante segnalibri di processo.
2. Modifica lo script e imposta il limite nelle opzioni aggiuntive nella finestra API `getSource` AWS Glue. Devi inoltre impostare il contesto di trasformazione per il segnalibro di processo affinché memorizzi l'elemento `state`. Ad esempio:

   Python

   ```
   glueContext.create_dynamic_frame.from_catalog(
       database = "database",
       table_name = "table_name",
       redshift_tmp_dir = "",
       transformation_ctx = "datasource0",
       additional_options = {
           "boundedFiles" : "500", # need to be string
         # "boundedSize" : "1000000000" unit is byte
       }
   )
   ```

   Scala

   ```
   val datasource0 = glueContext.getCatalogSource(
       database = "database", tableName = "table_name", redshiftTmpDir = "",
       transformationContext = "datasource0",
       additionalOptions = JsonOptions(
           Map("boundedFiles" -> "500") // need to be string
         //"boundedSize" -> "1000000000" unit is byte
       )
   ).getDynamicFrame()
   ```

   ```
   val connectionOptions = JsonOptions(
       Map("paths" -> List(baseLocation), "boundedFiles" -> "30")
   ```

```
)
val source = glueContext.getSource("s3", connectionOptions, "datasource0", "")
```

Per abilitare il partizionamento del carico di lavoro con esecuzione delimitata nella tabella del catalogo dati:

1. Imposta le coppie di chiave-valore nel campo `parameters` della struttura della tabella nel catalogo dati. Per ulteriori informazioni, consulta Visualizzazione e modifica dei dettagli tabella.
2. Imposta il limite superiore per la dimensione del set di dati o il numero di file elaborati:
   - Impostaboundedsize alla dimensione target del set di dati in byte. L'esecuzione del processo si interromperà dopo aver raggiunto la dimensione target dalla tabella.
   - Imposta `boundedFiles` al numero target dei file. L'esecuzione del processo si interromperà dopo aver elaborato il numero target dei file.

> **Note**
>
> Devi impostare solo uno tra `boundedSize` e `boundedFiles`, in quanto è supportato un solo limite.

Configurazione di un trigger AWS Glue per l'esecuzione automatica del processo

Una volta abilitata l'esecuzione delimitata, è possibile impostare un trigger AWS Glue per eseguire automaticamente il processo e caricare in modo incrementale i dati in esecuzioni sequenziali. Accedi alla console AWS Glue e crea un trigger, imposta l'orario di pianificazione e allegalo al tuo lavoro. Attiverà automaticamente la successiva esecuzione del processo ed elaborerà il nuovo batch di dati.

È possibile utilizzare i flussi di lavoro AWS Glue anche per orchestrare più processi allo scopo di elaborare dati da partizioni diverse in parallelo. Per ulteriori informazioni, consulta Trigger AWS Glue e Flussi di lavoro AWS Glue.

Per ulteriori informazioni sui casi d'uso e sulle opzioni, consulta il blog Optimizing Spark applications with workload partitioning in AWS Glue.

Problemi noti per AWS Glue

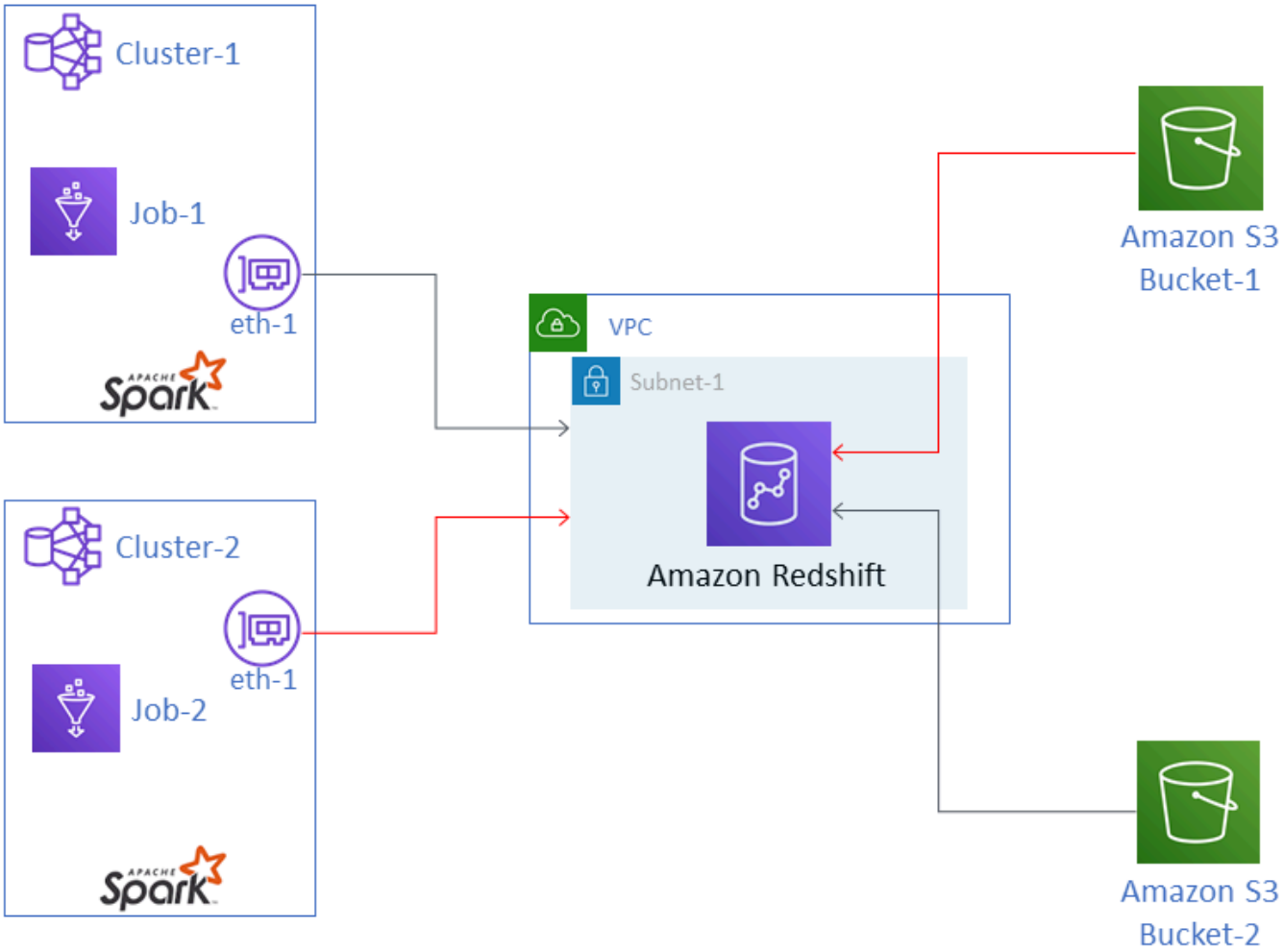
Prendi nota dei seguenti problemi noti per AWS Glue.

Argomenti

Prevenzione dell'accesso ai dati tra processi

Considera la situazione in cui disponi di due processi Spark AWS Glue in un singolo account AWS, ciascuno eseguito in un cluster Spark AWS Glue separato. I processi utilizzano le connessioni AWS Glue per accedere alle risorse nello stesso cloud privato virtuale (VPC, Virtual Private Cloud). In questo caso, un processo in esecuzione in un cluster potrebbe essere in grado di accedere ai dati dal processo in esecuzione nell'altro cluster.

Il seguente diagramma illustra un esempio di questa situazione.



Nel diagramma, AWS Glue `Job-1` è in esecuzione in `Cluster-1` e Job-2 è in esecuzione in `Cluster-2`. Entrambi i processi funzionano con la stessa istanza di Amazon Redshift, che si trova in `Subnet-1` di un VPC. `Subnet-1` potrebbe essere una sottorete pubblica o privata.

`Job-1` sta trasformando i dati da Amazon Simple Storage Service (Amazon S3) `Bucket-1` e scrivendo i dati in Amazon Redshift. `Job-2` sta facendo lo stesso con i dati in `Bucket-2`. `Job-1` utilizza il ruolo AWS Identity and Access Management (IAM) `Role-1` (non mostrato), che fornisce l'accesso a `Bucket-1`. `Job-2` utilizza `Role-2` (non mostrato), che fornisce l'accesso a `Bucket-2`.

Questi processi dispongono di percorsi di rete che consentono la comunicazione con i cluster reciproci e quindi di accedere ai dati reciproci. Ad esempio, `Job-2` può accedere ai dati in `Bucket-1`. Nel diagramma, questo viene mostrato come il percorso in rosso.

Per evitare questa situazione, ti consigliamo di collegare diverse configurazioni di sicurezza a `Job-1` e `Job-2`. Collegando le configurazioni di sicurezza, l'accesso ai dati tra processi viene bloccato in virtù dei certificati creati da AWS Glue. Le configurazioni di sicurezza possono essere configurazioni fittizie . Ciò significa che puoi creare le configurazioni di sicurezza senza abilitare la crittografia dei dati Amazon S3, dei dati Amazon CloudWatch o dei segnalibri dei processi. Tutte e tre le opzioni di crittografia possono essere disabilitate.

Per ulteriori informazioni sulle configurazioni di sicurezza, consulta the section called “Crittografia dei dati scritti da AWS Glue”.

Per collegare una configurazione di sicurezza a un processo

1. Apri la console AWS Glue all'indirizzo https://console.aws.amazon.com/glue/.
2. Nella pagina Configure the job properties (Configura le proprietà del processo) per il processo, espandere la sezione Security configuration, script libraries, and job parameters (Configurazione di sicurezza, librerie di script e parametri di processi).
3. Selezionare una configurazione di sicurezza nell'elenco.

Cronologia della documentazione per AWS Glue

Modifica	Descrizione	Data
Integrazione dei dati di Amazon Q in AWS Glue (anteprima)	L'integrazione dei dati di Amazon Q in AWS Glue è una nuova funzionalità di IA generativa di AWS Glue che consente ai data engineer e agli sviluppatori ETL di creare processi di integrazione dei dati utilizzando il linguaggio naturale. Gli ingegneri e gli sviluppatori possono chiedere a Q di creare processi, risolvere problemi e rispondere a domande su AWS Glue e sull'integrazione dei dati. Per ulteriori informazioni, consulta Integrazione dei dati di Amazon Q in AWS Glue. Questa funzionalità include un aggiornamento della policy `AwsGlueSessionUserRestrictedNotebookPolicy` gestita da AWS. Per ulteriori informazioni, consulta l'argomento relativo agli Aggiornamenti di AWS Glue sulle policy gestite da AWS.	30 gennaio 2024
Aggiornamento della documentazione in relazione ad AWS Glue Streaming	Aggiunto un nuovo capitolo con contenuti nuovi e riorganizzati per AWS Glue lo streaming. Questi	27 dicembre 2023

	contenuti descrivono il funzionamento di AWS Glue Streaming, le caratteristiche dell'elaborazione dei dati in tempo reale e le modalità di monitoraggio dei processi di streaming. Per ulteriori informazioni, consulta la pagina AWS Glue Streaming.	
Supporto per l'utilizzo del rilevamento dei dati sensibili granulari	La trasformazione relativa al rilevamento dei dati sensibili fornisce la possibili tà di rilevare, mascherare o rimuovere le entità che hai definito o che sono predefinite da AWS Glue. Le azioni granulari consentono inoltre di applicare un'azione specifica per entità. Per ulteriori informazioni, consulta Utilizzo del rilevamento dei dati sensibili granulari.	26 novembre 2023
Supporto per il monitoraggio dei processi con parametri AWS Glue di osservabilità	Utilizza i parametri AWS Glue di osservabilità per generare approfondimenti su ciò che accade all'interno di AWS Glue per i processi di Apache Spark e migliorare la classificazione e l'analisi dei problemi. Per ulteriori informazioni, consulta Monitoraggio con parametri AWS Glue di osservabilità.	26 novembre 2023

Supporto per il rilevamento delle anomalie in Qualità dei dati di AWS Glue	Il rilevamento delle anomalie relative a Qualità dei dati di AWS Glue applica nel tempo algoritmi di machine learning (ML) alle statistiche sui dati per rilevare modelli anomali e problemi nascosti di qualità dei dati che sono difficili da individuare attraverso le regole. Per ulteriori informazioni, consulta Rilevamento delle anomalie in Qualità dei dati di AWS Glue.	26 novembre 2023
Aggiornamento al comportamento di registrazione predefinito dell'interfaccia utente di Spark	I processi Spark che generano i log dell'interfaccia utente di Spark ora verranno scritti con un modello di nome file diverso per supportare l'interfaccia utente di Spark nella console AWS Glue. Ciò non modifica il comportamento del CloudWatch registro. È possibile ripristinare il comportamento legacy aggiornando la configurazione del processo. Per ulteriori informazioni, consulta Monitoraggio dei processi tramite l'interfaccia utente Web di Apache Spark.	17 novembre 2023

Supporto per nuove origini dati in AWS Glue per Spark	Le connessioni ad Amazon OpenSearch Service, Azure SQL, Azure Cosmos for NoSQL, SAP HANA Teradata Vantage e Vertica sono ora supportate nativamente all'interno. AWS Glue Inoltre, le connessioni a queste origini dati, oltre a MongoDB, sono ora disponibili per l'uso nell'editor visivo AWS Glue Studio. Per ulteriori informazioni, consulta Tipi e opzioni di connessione per ETL in AWS Glue per Spark per informazioni su AWS Glue per il supporto di Spark e Aggiunta di una connessione AWS Glue per informazioni sull'uso nell'editor visivo AWS Glue Studio.	17 novembre 2023
Supporto per generare le statistiche delle colonne	È possibile calcolare statistiche a livello di colonna per tabelle AWS Glue Data Catalog in formati di dati come Parquet, ORC, JSON, ION, CSV e XML senza configurare pipeline di dati aggiuntive. Per ulteriori informazioni, consulta Utilizzo delle statistiche delle colonne.	16 novembre 2023

Supporto per la compattazione dei dati per le tabelle Iceberg	Per migliorare le prestazioni di lettura tramite servizi di analisi AWS come Amazon Athena e Amazon EMR e i processi ETL AWS Glue, il Catalogo dati offre la compattazione gestita (un processo che compatta piccoli oggetti Amazon S3 in oggetti più grandi) per le tabelle Iceberg. Per ulteriori informazioni, consulta Ottimizzazione delle tabelle Iceberg.	13 novembre 2023
Aggiornamento al comportamento di attesa dell'esecuzione del processo	Le esecuzioni del processo standard di shell (interprete di comandi) Spark e Python ora passeranno a `WAITING` in determinate situazioni, anziché passare immediatamente a `FAILED`. Per ulteriori informazioni, consulta Stati di esecuzione dei processi AWS Glue.	8 novembre 2023
Guida per l'utente AWS Glue Studio consolidata nella guida per sviluppatori AWS Glue	La guida per l'utente AWS Glue Studio è stata spostata nella guida per sviluppatori per creare un'unica guida utente completa per AWS Glue Studio, la console AWS Glue e l'accesso a AWS Glue Studio a livello di programmazione.	25 ottobre 2023

Aggiornamento alla policy gestita AWSGlueServiceNotebookRole AWS	Sono state aggiunte informazioni su un aggiornamento minore alla politica AWSGlueServiceNotebookRole AWS gestita. Per ulteriori informazioni, consulta l'argomento relativo agli aggiornamenti AWS Glue sulle policy gestite AWS.	9 ottobre 2023
AWS Glue Studio supporta cinque nuove trasformazioni integrate	AWS Glue Studio supporta le seguenti cinque nuove trasformazioni integrate: Corrispondenza dei record, Rimuovi righe nulle, Analizza colonna JSON, Estrai percorso JSON ed Estrattore Regex. Per ulteriori informazioni, consulta la pagina Editing AWS Glue managed data transform nodes.	11 agosto 2023
Aggiornamento della politica AWSGlueServiceRole AWS gestita	Sono state aggiunte informazioni su un aggiornamento minore alla politica AWSGlueServiceRole AWS gestita. Per ulteriori informazioni, consulta l'argomento relativo agli aggiornamenti AWS Glue sulle policy gestite AWS.	4 agosto 2023

Supporto per il crawling delle tabelle Apache Hudi	Sono state aggiunte informazioni sull'utilizzo di AWS Glue per eseguire il crawling delle tabelle Hudi nei bucket Amazon S3 e sulla registrazione delle tabelle Hudi in AWS Glue Data Catalog. Per ulteriori informazioni, consulta le pagine Which data stores can I crawl? e Crawler properties.	21 luglio 2023
Aggiornamento della politica AWSGlueConsoleFullAccess AWS gestita	Sono state aggiunte informazioni su un aggiornamento minore alla politica AWSGlueConsoleFullAccess AWS gestita. Per ulteriori informazioni, consulta l'argomento relativo agli aggiornamenti AWS Glue sulle policy gestite AWS.	14 luglio 2023
Supporto per il crawling delle tabelle Apache Iceberg	Sono state aggiunte informazioni sull'utilizzo di AWS Glue per eseguire il crawling delle tabelle Iceberg nei bucket Amazon S3 e sulla registrazione delle tabelle Iceberg in AWS Glue Data Catalog. Per ulteriori informazioni, consulta le pagine Which data stores can I crawl? e Crawler properties.	7 luglio 2023

Supporto per AWS Glue con Ray	Sono state aggiunte informazioni su AWS Glue Ray, un nuovo motore in grado di supportare i processi AWS Glue. È stato riorganizzato il contenuto esistente relativo a AWS Glue con Spark per chiarire le ambiguità.	30 maggio 2023
Supporto per Qualità dei dati di AWS Glue (GA)	Qualità dei dati di AWS Glue è ora disponibile a livello generale. AWS Glue Qualità dei dati consente di valutare e monitorare la qualità dei dati. Per informazioni su come utilizzare Qualità dei dati di AWS Glue con Catalogo dati, consulta la pagina Qualità dei dati di AWS Glue. Per ulteriori informazioni su Qualità dei dati di AWS Glue per AWS Glue Studio, consulta la pagina Evaluating data quality with AWS Glue Studio.	24 maggio 2023

Supporto per tipi di worker di grandi dimensioni per i processi Apache Spark	È ora disponibile il supporto per l'uso dei tipi di worker `G.4X` e `G.8X` per i processi Apache Spark. Questi tipi di worker sono adatti per i processi i cui carichi di lavoro contengono trasformazioni, aggregazioni, join e query con i maggiori requisiti. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.	8 maggio 2023
Supporto per la creazione di indici di partizione durante il crawling delle tabelle	Sono state aggiunte informazioni sul modo in cui i crawler supportano la creazione di indici di partizione per le tabelle rilevate dal crawler. Per ulteriori informazioni, consulta la pagina Setting the partition index crawler configuration option.	24 aprile 2023
Supporto per i parametri di utilizzo delle risorse	Sono state aggiunte informazioni sulla visualizzazione dell'utilizzo delle risorse del servizio e sulla configurazione degli allarmi in Amazon. CloudWatch Per ulteriori informazioni, consulta la pagina AWS Glue resource monitoring.	7 aprile 2023

Aggiornamento della politica gestita AWSGlueConsoleFullAccess AWS	Sono state aggiunte informazioni su un aggiornamento minore alla politica AWSGlueConsoleFullAccess AWS gestita. Per ulteriori informazioni, consulta l'argomento relativo agli aggiornamenti AWS Glue sulle policy gestite AWS.	28 marzo 2023
È stata aggiunta una guida corredata da esempi per l'utilizzo di AWS Glue con un SDK AWS	La Guida per gli sviluppatori AWS Glue contiene due nuove sezioni che forniscono informazioni utili per l'utilizzo di AWS Glue con un SDK AWS. Per ulteriori informazioni, consulta Utilizzo di AWS Glue con un SDK AWS e Esempi di codice per AWS Glue utilizzando SDK AWS.	23 febbraio 2023
Aggiornamento della documentazione per IAM con AWS Glue	Riorganizzazione e aggiunta di informazioni sull'uso di IAM con AWS Glue. Per ulteriori informazioni, consulta Identity and Access Management per AWS Glue.	15 febbraio 2023

Supporto per l'esecuzione di processi ETL di streaming in AWS Glue versione 4.0	Sono state aggiunte informazioni sul supporto per l'esecuzione di processi ETL di streaming in Glue versione 4.0 e nuove opzioni per la connessione a un cluster Kafka o a un cluster Amazon Managed Streaming per Apache Kafka e flussi di dati Amazon Kinesis. Per ulteriori informazioni, consulta Aggiunta di processi ETL di streaming in AWS Glue e Tipi di connessione e opzioni per ETL in AWS Glue.	8 febbraio 2023
Supporto per il crawling delle origini dati MongoDB Atlas	Sono state aggiunte informazioni sull'utilizzo di AWS Glue per il crawling delle origini dati MongoDB Atlas. Per ulteriori informazioni, consulta Quali archivi di dati posso sottoporre a scansione? , proprietà di connessione MongoDB e MongoDB Atlas e Utilizzo di una connessione MongoDB o MongoDB Atlas.	6 febbraio 2023

Supporto per il crawling delle tabelle Delta Lake con un connettore Delta Lake nativo	Sono state aggiunte informazioni sull'utilizzo di AWS Glue per il crawling delle tabelle Delta Lake tramite un connettore Delta Lake nativo. Questa funzionalità ti consente di utilizzare i motori di query AWS per interrogare direttamente il log delle transazioni Delta e utilizzare funzionalità come i viaggi nel tempo e le garanzie ACID e per sincronizzare i metadati di Delta Lake dai file delle transazioni di Amazon S3 nel catalogo di dati per abilitare le autorizzazioni delle colonne sulle query in Lake Formation. Per ulteriori informazioni, consulta Come specificare le opzioni di configurazione per un archivio di dati Delta Lake e Interrogazione delle tabelle Delta Lake.	15 dicembre 2022

Supporto per AWS Glue Data Quality (anteprima)	Il supporto è ora disponibile per AWS Glue Data Quality (anteprima). AWS Glue Data Quality consente di valutare e monitorare la qualità dei dati quando si usa AWS Glue 3.0. Per informazioni su come utilizzare AWS Glue Data Quality con il catalogo di dati, consulta AWS Glue Data Quality (anteprima). Per ulteriori informazioni su Qualità dei dati di AWS Glue per AWS Glue Studio, consulta la pagina Evaluating data quality with AWS Glue Studio.	30 novembre 2022
Supporto per un nuovo connettore Amazon Redshift Spark con nuove funzionalità e miglioramenti delle prestazioni	È ora disponibile il supporto per un nuovo connettore Amazon Redshift Spark con un nuovo driver JDBC da utilizzare con i processi ETL di AWS Glue per creare applicazioni Apache Spark che leggono e scrivono dati in Amazon Redshift come parte delle pipeline di importazione dei dati e trasformazione dei dati. Per ulteriori informazioni, consulta Spostamento di dati da e verso Amazon Redshift.	29 novembre 2022

Supporto per AWS Glue versione 4.0.	Aggiunte informazioni sul supporto per AWS Glue 4.0. Le funzionalità includono il supporto nativo per i framework data lake aperti con Apache Hudi, Delta Lake e Apache Iceberg e il supporto nativo per il plug-in di archiviazione cloud shuffle basato su Amazon S3 (un plug-in Apache Spark) per utilizzare Amazon S3 per la capacità di archiviazione shuffle ed elastica. Per ulteriori informazioni, consulta Note di rilascio di AWS Glue e Migrazione dei processi AWS Glue a AWS Glue versione 4.0.	28 novembre 2022
AWS Glue Studio ora offre trasformazioni visive personalizzate	Le trasformazioni visive personalizzate consentono ai clienti di definire, riutilizzare e condividere la logica ETL specifica dell'azienda tra i propri team. Per ulteriori informazioni, consulta Trasformazioni visive personalizzate.	28 novembre 2022

Supporto per l'utilizzo del crawler AWS Glue per pubblicare i metadati per i datastore JDBC	È ora disponibile il supporto per l'utilizzo del crawler AWS Glue per pubblicare metadati come commenti e tipi non elaborati nel catalogo di dati per i datastore JDBC. Per ulteriori informazioni, consulta Parametri impostati nelle tabelle del catalogo dati per crawler, proprietà del crawler e struttura. JdbcTarget	18 novembre 2022
Supporto per il crawling di datastore Snowflake	È ora disponibile il supporto per l'utilizzo di AWS Glue per eseguire il crawling delle tabelle e delle viste Snowflake e per pubblicare i metadati nel catalogo di dati come voce della tabella. Per le tabelle esterne Snowflake in Amazon S3, il crawler esegue il crawling anche della posizione Amazon S3 e del tipo di formato di file della tabella esterna e lo compila come parametri della tabella. Per ulteriori informazioni, consulta Quali datastore posso sottoporre a crawling? , Proprietà della connessione a AWS Glue e Parametri impostati nelle tabelle del catalogo di dati dal crawler.	18 novembre 2022

Supporto per una migliore gestione dello shuffle delle applicazioni Spark	È ora disponibile il supporto per un nuovo plug-in di archiviazione cloud shuffle per Apache Spark. Per ulteriori informazioni, consulta Plug-in shuffle di AWS Glue Spark con Amazon S3 e Plug-in di archiviazione cloud shuffle per Apache Spark.	15 novembre 2022
È stato aggiunto il supporto per le destinazioni catalogo dati quando si accelerano le notifiche di evento Amazon S3 del crawling	Oltre al supporto esistente per le destinazioni Amazon S3, ora è disponibile il supporto per accelerare il crawling per le destinazioni catalogo dati tramite notifiche di eventi Amazon S3. Per ulteriori informazioni, consulta Accelerazione del crawling usando le notifiche eventi di Amazon S3.	13 ottobre 2022
Supporto per specificare il numero massimo di tabelle che un crawler può creare	È ora disponibile il supporto per specificare il numero massimo di tabelle che il crawler può creare. Per ulteriori informazioni, consulta la pagina Come specificare il numero massimo di tabelle che il crawler può creare.	6 settembre 2022

Supporto per Python 3.9 in processi della shell Python in AWS Glue	È ora disponibile il supporto per l'esecuzione di script compatibili con Python 3.9 nei processi della shell (interpre te di comandi) Python in AWS Glue e per la scelta dell'uso di set di librerie preconfez ionati. Per ulteriori informazi oni, consulta Processi della shell Python in AWS Glue.	11 agosto 2022
Supporto per l'esecuzione di processi AWS Glue non urgenti o non sensibili al tempo sulla capacità inutilizz ata	È ora disponibile il supporto per la configurazione di esecuzioni flessibili per processi non urgenti come processi di pre-produzione, test e caricamenti di dati una tantum. Per ulteriori informazi oni, consulta Aggiunta di processi in AWS Glue.	9 agosto 2022
Il supporto per un nuovo tipo di worker per i processi di streaming	Il supporto di questo servizio di Support per l'uso del tipo di worker `G.025X` per processi di streaming a basso volume. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.	14 luglio 2022
Il supporto per l'utilizzo di Kafka SASL in AWS Glue connessioni	Il supporto di Kafka SASL è ora disponibile in AWS Glue connessioni. Per ulteriori informazioni, consulta AWS Glue Proprietà di connessio ne Kafka per l'autenticazione client.	5 luglio 2022

Supporto per il connettore Apache Kafka per gli schemi protobuf	Il supporto di Apache Kafka Connector è attualmente disponibile per gli schemi Protobuf. Per ulteriori informazioni, consulta Registro degli schemi di AWS Glue.	9 giugno 2022
Supporto per Auto Scaling per processi AWS Glue (GA)	Informazioni aggiuntive sull'utilizzo di Auto Scaling per i processi in AWS Glue versione 3.0 per scalare dinamicamente le risorse di calcolo. Per ulteriori informazioni, consulta Utilizzo di Auto Scaling per AWS Glue.	14 aprile 2022
Aggiornamento della documentazione per lo sviluppo di AWS Glue e i test di script di processo di AWS Glue	Informazioni riorganizzate e aggiunte sui metodi di sviluppo e test disponibili per AWS Glue, incluse le istruzioni per lo sviluppo con Docker. Per ulteriori informazioni, consulta Sviluppo e test di script di processo di AWS Glue.	14 marzo 2022
Aggiunta di buffer del protocollo (protobuf) come formato di dati supportato per il registro degli schemi di AWS Glue	Aggiunte informazioni su Protobuf come formato dati supportato (oltre ad AVRO e JSON). Per ulteriori informazioni, consulta Registro degli schemi di AWS Glue.	25 febbraio 2022

Supporto per il crawling delle tabelle Delta Lake	Sono state aggiunte informazioni sull'utilizzo di AWS Glue per sottoporre a crawling le tabelle Delta Lake. Per ulteriori informazioni, consulta How to specify configuration options for a Delta Lake data store. (Come specificare le opzioni di configurazione per un archivio dati Delta Lake).	24 febbraio 2022
Supporto per informazioni sul processo di AWS Glue	Aggiunte informazioni sull'utilizzo delle informazioni sul processo di AWS Glue per semplificare il debug e l'ottimizzazione dei lavori per i tuoi processi AWS Glue. Per ulteriori informazioni, consulta Monitoraggio tramite le informazioni sui processi di AWS Glue.	8 febbraio 2022

Supporto per il crawling di tabelle Catalogo dati supportate da Amazon S3 utilizzando un endpoint VPC	Oltre all'archivio dati di Amazon S3, si possono configurare le tabelle Catalogo dati supportate da Amazon S3 per consentire l'accesso solo a un ambiente Amazon Virtual Private Cloud (Amazon VPC) per motivi di sicurezza , audit o controllo. Per ulteriori informazioni, consulta Crawling di un datastore Amazon S3 o di tabelle Catalogo dati supportate da Amazon S3 utilizzando un endpoint VPC.	3 febbraio 2022
Supporto per le tavole governate dalla Lake Formation	Aggiunte informazioni sul supporto AWS Glue per tabelle governate da Lake Formation, che supportano le transazioni ACID, la compattazione automatica dei dati e le query di viaggio nel tempo. Per ulteriori informazioni, consulta API AWS Glue, e Guida per gli sviluppatori di AWS Lake Formation.	30 novembre 2021
Nuove policy AWS gestite aggiunte per sessioni interattive e notebook	Le nuove policy gestite per IAM forniscono una maggiore sicurezza per l'utilizzo di AWS Glue con sessioni interattive e notebook. Per ulteriori informazioni, consulta la sezione Policy gestite da AWS per AWS Glue.	30 novembre 2021

Il registro dello schema Glue ora supportato con i processi di streaming	È possibile creare processi di streaming che accedono alle tabelle che fanno parte di Glue Schema Registry. Per ulteriori informazioni, consulta AWS Glue Schema Registry e Aggiunta di processi di streaming ETL in AWS Glue.	15 novembre 2021
Supporto per nuove caratteristiche di machine learning	Aggiunte informazioni sulle nuove funzionalità per la trasformazione di machine learning Ricerca corrispondenze, tra cui la corrispondenza incrementale e il punteggio di corrispondenza. Per ulteriori informazioni, consulta Ricerca di corrispondenze incrementali e Stima della qualità delle corrispondenze utilizzando i punteggi di confidenza delle corrispondenze.	31 ottobre 2021
(Anteprima privata) Supporto per processi flessibili di AWS Glue	Aggiunte informazioni sulla configurazione dei processi AWS Glue Spark con una classe di esecuzione flessibile, appropriata per i processi non sensibili al tempo i cui tempi di inizio e completamento possono variare. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.	29 ottobre 2021

Supporto per accelerare il crawling usando le notifiche eventi di Amazon S3	Sono state aggiunte informazioni sull'accelerazione del crawling usando le notifiche eventi di Amazon S3. Per ulteriori informazioni, consulta Accelerazione del crawling usando le notifiche eventi di Amazon S3.	15 ottobre 2021
Ulteriori opzioni di configurazione della sicurezza relative al controllo degli accessi e ai VPC	Sono state aggiunte informazioni su come configurare nuove autorizzazioni per il controllo degli accessi su AWS Glue e le configurazione dei VPC. Per ulteriori informazioni, consulta Tag AWS in AWS Glue, Policy basate su identità (policy IAM) che controllano le impostazioni utilizzando chiavi di condizione o chiavi di contesto, e Configurazione di tutte le chiamate AWS affinché passino attraverso il VPC.	13 ottobre 2021
Supporto per le policy di endpoint VPC	Aggiunte informazioni sul supporto per policy endpoint Virtual Private Cloud (VPC) in AWS Glue. Per ulteriori informazioni consulta AWS Glue ed endpoint VPC di interfaccia (AWS PrivateLink).	11 ottobre 2021
Glue Studio è ora disponibile in Cina	AWS Glue Studio è ora disponibile nelle Regioni cinesi di Pechino e Ningxia.	11 ottobre 2021

AWS Glue Studio offre la creazione di notebook per la modifica interattiva dei processi	I notebook consentono di scrivere ed eseguire codice, visualizzare i risultati e condividere informazioni. In genere, i data scientist utilizzano i notebook per esperimenti e attività di esplorazione dei dati. Per ulteriori informazioni, consulta Utilizzo di notebook.	1° ottobre 2021
L'accesso diretto alle fonti di streaming ora disponibile	Quando si aggiungono origini dati al processo ETL nell'editor visivo, è possibile fornire informazioni per accedere al flusso di dati, anziché utilizzare un database e una tabella di Data Catalog.	30 settembre 2021
Documentata la policy di supporto versione AWS Glue	Aggiunte informazioni sulla policy di supporto versione AWS Glue e sulle fasi di fine vita per alcune versioni AWS Glue. Per ulteriori informazioni, consulta Policy di supporto versione AWS Glue.	24 settembre 2021
I connettori personalizzati possono ora essere utilizzati con le anteprime dei dati	Quando modifichi il nodo dell'origine dati utilizzando un connettore personalizzato, puoi visualizzare in anteprima il set di dati scegliendo la scheda Anteprima dati. Per ulteriori informazioni, consulta Connettori personalizzati.	24 settembre 2021

Supporto per sessioni interattive AWS Glue (anteprima privata)	(Anteprima privata) Aggiunte informazioni sull'uso delle sessioni interattive AWS Glue per eseguire carichi di lavoro Spark nel cloud da qualsiasi Jupyter Notebook. Le sessioni interattive sono il metodo preferito per sviluppare il tuo AWS Glue di estrazione, trasformazione e caricamento del codice (ETL), quando utilizzi AWS Glue 2.0 o versioni successive. Per ulteriori informazioni, consulta Configurazione e funzionamento delle sessioni interattive AWS Glue per notebook Jupyter.	24 agosto 2021
Supporto per la creazione di flussi di lavoro dai progetti (GA)	(Anteprima pubblica) Sono state aggiunte informazioni sulla codifica dei casi d'uso comuni di estrazione, trasformazione e caricamento (ETL) nei piani e sulla creazione di flussi di lavoro dai piani. Consente agli analisti di dati di creare ed eseguire facilmente processi ETL complessi. Per ulteriori informazioni, consulta Esecuzione di attività ETL complesse utilizzando gli schemi e i flussi di lavoro in AWS Glue.	23 agosto 2021

Supporto per AWS Glue versione 3.0.	Aggiunte informazioni sul supporto per AWS Glue versione 3.0 che supporta l'aggiornamento del motore Apache Spark 3.0 per l'esecuzione dei processi ETL di Apache Spark e altre ottimizzazioni e aggiornamenti. Per ulteriori informazioni, consulta Note di rilascio di AWS Glue e Migrating AWS Glue jobs to AWS Glue version 3.0. Altre caratteristiche di questa versione includono lo shuffle manager AWS Glue, un lettore CSV vettorizzato SIMD e predicati delle partizioni del catalogo. Per ulteriori informazioni, consulta AWS Glue Spark shuffle manager with Amazon S3, Opzioni di formato per input e output ETL in AWS Glue, e Server-side filtering using catalog partition predicates.	18 agosto 2021
AWS GovCloud (US) Region	AWS Glue Studio ora è disponibile in AWS GovCloud (US) Region	18 agosto 2021

Creazione di shell Python disponibile in AWS Glue Studio	Quando si crea un nuovo processo, è ora possibile scegliere di creare un processo di shell Python. Per ulteriori informazioni, consulta Avvio della creazione del processo e Modifica di processi shell Python in AWS Glue Studio.	13 agosto 2021
Support per l'avvio di un flusso di lavoro con un EventBridge evento Amazon	Aggiunte informazioni su come AWS Glue può essere un consumatore di eventi in un'architettura basata su eventi. Per ulteriori informazioni, consulta Avvio di un AWS Glue flusso di lavoro con un EventBridge evento Amazon e Visualizzazione degli EventBridge eventi che hanno avviato un flusso di lavoro.	14 luglio 2021
Aggiunta di JSON come formato di dati supportato per il registro degli schemi di AWS Glue	Aggiunte informazioni su JSON come formato dati supportato (oltre ad AVRO). Per ulteriori informazioni, consulta Registro degli schemi di AWS Glue.	30 giugno 2021

Creare processi di streaming AWS Glue senza una tabella del catalogo dati	La funzione Python `create_data_frame_from_options` o `getSource` per gli script Scala supportano la creazione di processi ETL di streaming che fanno riferimento direttamente ai flussi di dati anziché richiedere una tabella del catalogo dati.	15 giugno 2021
Le trasformazioni di machine learning in AWS Glue ora supportano le chiavi AWS Key Management Service	È possibile specificare una configurazione di protezione o chiave AWS KMS durante la configurazione delle trasformazioni di Machine Learning AWS Glue con la console, la CLI o le API AWS Glue. Per ulteriori informazioni, consulta Utilizzo della crittografia dati con le trasformazioni basate su machine learning e API di Machine Learning AWS Glue.	15 giugno 2021
Aggiornamento della politica AWSGlueConsoleFullAccess AWS gestita	Sono state aggiunte informazioni su un aggiornamento minore alla politica AWSGlueConsoleFullAccess AWS gestita. Per ulteriori informazioni, consulta l'argomento relativo agli aggiornamenti AWS Glue sulle policy gestite AWS.	10 giugno 2021

Visualizzare il set di dati del processo durante la creazione e la modifica dei processi	È possibile utilizzare la nuova scheda di anteprima dati per un nodo nel diagramma del processo per visualizzare un esempio dei dati elaborati da tale nodo. Per ulteriori informazioni, consulta Utilizzo delle anteprime dei dati nell'editor visivo dei processi.	7 giugno 2021
Supporto per specificare un valore che indica la posizione della tabella per l'output del crawler.	Sono state aggiunte informazioni su come specificare un valore che indica la posizione della tabella durante la configurazione dell'output del crawler. Per ulteriori informazioni, consulta Come specificare la posizione della tabella.	4 giugno 2021
Supporto per il crawling di un campione di file in un set di dati durante il crawling di un archivio dati Amazon S3	Sono state aggiunte informazioni su come eseguire il crawling di un campione di file durante il crawling di Amazon S3. Per ulteriori informazioni, consulta Proprietà del crawler.	10 maggio 2021

Supporto per il AWS Glue writer parquet ottimizzato	Sono state aggiunte informazioni sull'utilizzo del parquet writer AWS Glue ottimizzato DynamicFrames per creare o aggiornare tabelle con la `parquet` classificazione. Per ulteriori informazioni, consulta Creazione di tabelle, aggiornamento dello schema e aggiunta di nuove partizioni nel catalogo dati da processi ETL AWS Glue e Opzioni di formato per input e output ETL in AWS Glue.	4 maggio 2021
Supporto per le password di autenticazione client Kafka	Sono state aggiunte informazioni su come lo streaming di processi ETL in AWS Glue supporta l'autenticazione del certificato client SSL con i produttori del flusso Apache Kafka. È ora possibile fornire un certificato personalizzato durante la definizione di una connessione AWS Glue a un cluster Apache Kafka, che AWS Glue utilizzerà durante l'autenticazione con esso. Per ulteriori informazioni, consulta Proprietà della connessione AWS Glue e API di connessione.	28 Aprile 2021

Supporto per l'utilizzo di dati da Amazon Kinesis Data Streams in un altro account nei processi ETL di streaming	Sono state aggiunte informazioni su come creare un processo ETL di streaming per utilizzare i dati da Amazon Kinesis Data Streams in un altro account. Per ulteriori informazioni, consulta Aggiunta di processi di streaming ETL in AWS Glue.	30 marzo 2021
Trasformazione SQL disponibile	Puoi utilizzare un nodo di trasformazione SQL per scrivere la tua trasformazione sotto forma di query SQL. Per ulteriori informazioni, consulta Utilizzo di una query SQL per trasformare i dati.	23 marzo 2021
Supporto per la creazione di flussi di lavoro dagli schemi (anteprima pubblica)	(Anteprima pubblica) Sono state aggiunte informazioni sulla codifica dei casi d'uso comuni di estrazione, trasformazione e caricamento (ETL) nei piani e sulla creazione di flussi di lavoro dai piani. Consente agli analisti di dati di creare ed eseguire facilmente processi ETL complessi. Per ulteriori informazioni, consulta Esecuzione di attività ETL complesse utilizzando gli schemi e i flussi di lavoro in AWS Glue.	22 marzo 2021

I connettori possono essere utilizzati per le destinazioni dati	Adesso è supportato l'utilizzo di un connettore personalizzato o Marketplace AWS per la destinazione dati. Per ulteriori informazioni, consulta Creazione di processi con connettori personalizzati.	15 marzo 2021
Supporto per i parametri sull'importanza delle colonne per le trasformazioni di machine learning AWS Glue	Aggiunte informazioni sulla visualizzazione delle metriche relative all'importanza delle colonne quando si lavora con le trasformazioni basate su machine learning AWS Glue. Per ulteriori informazioni, consulta Operare con le trasformazioni basate su machine learning nella console di AWS Glue	5 febbraio 2021
La pianificazione dei processi è ora disponibile in AWS Glue Studio	È possibile definire una pianificazione basata sul tempo per le esecuzioni del processo in AWS Glue Studio. È possibile utilizzare la console per creare una pianificazione di base o definire una pianificazione più complessa utilizzando la sintassi cron di tipo Unix. Per ulteriori informazioni, consulta Pianificazione delle esecuzioni.	21 dicembre 2020

Rilascio di AWS Glue Custom Connectors	AWS Glue Custom Connectors ti consente di scoprire e sottoscrivere i connettori in Marketplace AWS. Abbiamo anche rilasciato le interfacce del runtime di Spark AWS Glue per collegare connettori creati per origine dati Apache Spark, query federata Athena e API JDBC. Per ulteriori informazioni, consulta Utilizzo di connettori e connessioni con AWS Glue Studio.	21 dicembre 2020
Supporto per l'esecuzione di processi ETL di streaming in AWS Glue versione 2.0	Aggiunte informazioni sull'esecuzione di processi ETL di streaming in Glue versione 2.0. Per ulteriori informazioni, consulta Aggiunta di processi di streaming ETL in AWS Glue.	18 dicembre 2020
Supporto per il partizionamento del carico di lavoro con esecuzione limitata	Aggiunte informazioni sull'abilitazione del partizionamento del carico di lavoro per configurare i limiti superiori della dimensione del set di dati o il numero di file elaborati nelle esecuzioni dei processi ETL. Per ulteriori informazioni, consulta Partizionamento del carico di lavoro con esecuzione limitata.	23 novembre 2020

Supporto per una gestione avanzata delle partizioni	Sono state aggiunte informazioni su come utilizzare le nuove API per aggiungere o eliminare un indice di partizione a/da una tabella esistente. Per ulteriori informazioni, consulta Utilizzo degli indici delle partizioni.	23 novembre 2020
Supporto per il registro degli schemi di AWS Glue	Sono state aggiunte informazioni sull'utilizzo del registro degli schemi di AWS Glue per l'individuazione, il controllo e l'evoluzione in modo centralizzato degli schemi. Per ulteriori informazioni, consulta Registro degli schemi di AWS Glue.	19 novembre 2020
Supporto per il formato di input Grok nei processi ETL di streaming	Aggiunte informazioni sull'applicazione dei pattern Grok alle origini di streaming, ad esempio i file di log. Per ulteriori informazioni, consulta Applicazione di pattern Grok alle sorgenti di streaming.	17 novembre 2020
Supporto per l'aggiunta di tag ai flussi di lavoro nella console AWS Glue	Sono state aggiunte informazioni sull'aggiunta di tag durante la creazione di un flusso di lavoro utilizzando la console AWS Glue. Per ulteriori informazioni, consulta Creazione e generazione di un flusso di lavoro nella console AWS Glue.	27 ottobre 2020

Supporto per le esecuzione incrementale del crawler	Aggiunte informazioni sul supporto per le esecuzioni di crawler incrementali, che eseguono il crawling solo delle cartelle Amazon S3 aggiunte dall'ultima esecuzione. Per ulteriori informazioni, consulta Crawling incrementale.	21 ottobre 2020
Supporto per il rilevamento dello schema per le origini dati ETL di streaming. supporto per le origini dei dati ETL di streaming Avro e Kafka autogestito	I processi di estrazione, trasformazione e caricamento (ETL) di streaming in AWS Glue ora possono rilevare automaticamente lo schema dei registri in entrata e gestire le modifiche dello schema per ogni registro. Sono ora supportate le origini di dati Kafka autogestite. I processi ETL di streaming ora supportano il formato Avro nelle origini dati. Per ulteriori informazioni, consulta Streaming ETL inAWS Glue, Definizione delle proprietà dell'operazione per un'operazione ETL di streaming eNote e restrizioni per le origini di streaming Avro.	7 ottobre 2020

Supporto per il crawling delle origini dei dati MongoDB e DocumentDB	Aggiunte informazioni sul supporto per il crawling delle origini dati MongoDB e Amazon DocumentDB (con compatibilità MongoDB). Per ulteriori informazioni, consulta Definizione di crawler.	5 ottobre 2020
Supporto per la conformità a FIPS	Aggiunte informazioni sugli endpoint FIPS per i clienti che necessitano di moduli crittografici convalidati FIPS 140-2 quando accedono ai dati con AWS Glue. Per ulteriori informazioni, consulta la pagina Conformità FIPS.	23 settembre 2020
AWS Glue Studio fornisce un'interfaccia visiva facile da usare per la creazione e il monitoraggio dei processi	Ora è possibile utilizzare una semplice interfaccia grafica per comporre lavori che spostano e trasformano i dati ed eseguirli su AWS Glue. È quindi possibile utilizzar e il pannello di controllo di esecuzione dei processi in AWS Glue Studio per monitorare l'esecuzione di ETL e garantire che i processi funzionino come previsto. Per ulteriori informazioni, consulta la Guida per l'utente di AWS Glue Studio.	23 settembre 2020

Supporto per la creazione di indici di tabella per migliorare le prestazioni delle query	Aggiunte informazioni sulla creazione di indici di tabella per consentire il recupero di un sottoinsieme di partizioni da una tabella. Per ulteriori informazioni, consulta Utilizzo degli indici delle partizioni.	9 settembre 2020
Supporto per tempi di startup ridotti durante l'esecuzione di processi ETL di Apache Spark in AWS Glue versione 2.0.	Aggiunte informazioni sul supporto per AWS Glue versione 2.0, che fornisce un'infrastruttura aggiornata per l'esecuzione di processi ETL di Apache Spark con tempi di startup ridotti, modifiche nella registrazione e supporto per la specifica di moduli Python aggiuntivi a livello di processo. Per ulteriori informazioni, consulta Note di rilascio di AWS Glue ed Esecuzione di processi ETL Spark con tempi di avvio ridotti.	10 agosto 2020
Supporto per limitare il numero di esecuzioni simultanee del flusso di lavoro.	Aggiunte informazioni su come limitare il numero di esecuzioni simultanee per un determinato flusso di lavoro. Per ulteriori informazioni, consulta Creazione e generazione di un flusso di lavoro utilizzando la console AWS Glue.	10 agosto 2020

Supporto per il crawling di un datastore Amazon S3 utilizzando un endpoint VPC	Aggiunte informazioni sulla configurazione dell'archivio dati Amazon S3 per consentire l'accesso solo a un ambiente Amazon Virtual Private Cloud (Amazon VPC) per motivi di sicurezza, audit o controllo. Per ulteriori informazioni, consulta Crawling di un datastore Amazon S3 utilizzando un endpoint VPC.	7 agosto 2020
Supporto per la ripresa delle esecuzioni del flusso di lavoro	Aggiunte informazioni su come riprendere le esecuzioni del flusso di lavoro completate solo parzialmente perché uno o più nodi (processi o crawler) non sono stati completati correttamente. Per ulteriori informazioni, consulta Ripresa e ripristino dell'esecuzione di un flusso di lavoro.	27 luglio 2020
Supporto per l'abilitazione di certificati emessi da CA privati nelle connessioni Kafka in AWS Glue.	Aggiunte informazioni sulle nuove opzioni di connessione che supportano l'abilitazione dei certificati emessi da una CA privati per le connessioni Kafka in AWS Glue. Per ulteriori informazioni, consulta Tipi di connessione e opzioni per ETL in AWS Glue e Parametri speciali utilizzati daAWS Glue.	20 luglio 2020

Supporto per la lettura dei dati DynamoDB in un altro account	Aggiunte informazioni sul supporto AWS Glue per la lettura di dati da una tabella DynamoDB di un altro account AWS. Per ulteriori informazioni, consulta Lettura dai dati DynamoDB in un altro account.	17 luglio 2020
Supporto per una connessione al writer DynamoDB nella versione AWS Glue 1.0 o successive	Aggiunte informazioni sul supporto per il writer DynamoDB e opzioni di connessione nuove o aggiornate per la lettura o la scrittura di DynamoDB. Per ulteriori informazioni, consulta Tipi di connessione e opzioni per ETL in AWS Glue.	17 luglio 2020
Supporto per i collegamenti alle risorse e per il controllo degli accessi tra account utilizzando sia AWS Glue che Lake Formation	Aggiunti contenuti relativi a nuovi oggetti del catalogo dati denominati link alle risorse e a come gestire la condivisione delle risorse del catalogo dati tra gli account sia con AWS Glue che con AWS Lake Formation. Per ulteriori informazioni, consulta Concedere l'accesso multi-account e Link alle risorse della tabella.	7 luglio 2020

Supporto per il campionamento dei registri durante il crawling dei datastore DynamoDB	Sono state aggiunte informazioni sulle nuove proprietà che puoi configurare durante il crawling di un datastore DynamoDB. Per ulteriori informazioni, consulta Proprietà del crawler.	12 giugno 2020
Supporto per l'arresto di un'esecuzione del flusso di lavoro.	Sono state aggiunte informazioni su come interrompere l'esecuzione di un flusso di lavoro per un determinato flusso di lavoro. Per ulteriori informazioni, vedere Arresto di un'esecuzione del flusso di lavoro.	14 maggio 2020
Supporto per i processi ETL di streaming Spark	Sono state aggiunte informazioni sulla creazione di processi ETL (Extract, Transform and Load) con origini dati in streaming. Per ulteriori informazioni, consulta Aggiunta di processi di streaming ETL in AWS Glue.	27 aprile 2020

Supporto per la creazione di tabelle, l'aggiornamento dello schema e l'aggiunta di nuove partizioni nel catalogo dati dopo l'esecuzione di un processo ETL	Sono state aggiunte informazioni su come abilitare la creazione di tabelle, l'aggiornamento dello schema e l'aggiunta di nuove partizioni per visualizzare i risultati del processo ETL nel catalogo dati. Per ulteriori informazioni, consulta Creazione di tabelle, aggiornamento dello schema e aggiunta di nuove partizioni nel catalogo dati da processi ETL AWS Glue.	2 aprile 2020
Supporto per specificare una versione per il formato dati Apache Avro come input e output ETL in AWS Glue	Aggiunte informazioni su come specificare una versione per il formato dati Apache Avro come input e output ETL in AWS Glue. La versione predefinita 1.7. Puoi utilizzare l'opzione del formato `version` per specificare Avro versione 1.8 per abilitare la lettura/scrittura logica. Per ulteriori informazioni, consulta Opzioni di formato per gli input e output ETL in AWS Glue.	31 marzo 2020

Supporto per il committer ottimizzato EMRFS S3 per la scrittura di dati Parquet in Amazon S3	Sono state aggiunte informazioni su come impostare un nuovo flag per abilitare il committer ottimizzato EMRFR S3 per la scrittura dei dati Parquet in Amazon S3 durante la creazione o l'aggiornamento di un processo AWS Glue. Per ulteriori informazioni, consulta Parametri speciali usati da AWS Glue.	30 marzo 2020
Supporto per le trasformazioni di machine learning come risorsa gestita dai tag delle risorse AWS	Sono state aggiunte informazioni sull'utilizzo dei tag delle risorse AWS per gestire e controllare l'accesso alle trasformazioni di machine learning in AWS Glue. È possibile assegnare i tag delle risorse AWS a processi, trigger, endpoint, crawler e trasformazioni di machine learning in AWS Glue. Per ulteriori informazioni sui tag, consultare Tag AWS in AWS Glue.	2 marzo 2020
Supporto per argomenti di lavoro non sovrascrivibili	Aggiunte informazioni sul supporto per parametri di lavoro speciali che non possono essere sovrascritti nei trigger o quando si esegue il processo. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.	12 febbraio 2020

Supporto per nuove trasformazioni per l'utilizzo con set di dati in Amazon S3	Sono state aggiunte informazioni sulle nuove trasformazioni (Merge, Purge e Transition) ed esclusioni delle classi di storage Amazon S3 per applicazioni Apache Spark per l'utilizzo con set di dati in Amazon S3. Per ulteriori informazioni sul supporto per queste trasformazioni per Python, mergeDynamicFrameconsulta Working with Datasets in Amazon S3. Per Scala, vedi mergeDynamicFramese le API di Scala. AWS Glue GlueContext	16 gennaio 2020
Supporto per l'aggiornamento del Catalogo Dati con nuove informazioni di partizione da un processo ETL	Sono state aggiunte informazioni su come codificare uno script di estrazione, trasformazione e caricamento (ETL) per aggiornare AWS Glue Data Catalog con le nuove informazioni sulla partizione. Con questa caratteristica, non è più necessario eseguire nuovamente il crawler al termine del processo per visualizzare le nuove partizioni. Per ulteriori informazioni, consulta Aggiornamento del catalogo dati con nuove partizioni.	15 gennaio 2020

Nuovo tutorial: usare un notebook SageMaker	È stato aggiunto un tutorial che dimostra come usare un SageMaker notebook Amazon per aiutarti a sviluppare i tuoi script ETL e di machine learning. Vedi il tutorial: Usa un Amazon SageMaker Notebook con il tuo endpoint di sviluppo.	3 gennaio 2020
Supporto per la lettura da MongoDB e Amazon DocumentDB (compatibile con MongoDB)	Aggiunte informazioni sui nuovi tipi di connessione e opzioni di connessio ne per leggere e scrivere su MongoDB e Amazon DocumentDB (con compatibi lità MongoDB). Per ulteriori informazioni, consulta Tipi di connessione e opzioni per ETL in AWS Glue.	17 dicembre 2019
Varie correzioni e chiarimenti	Sono state aggiunte diverse correzioni e chiarimenti. Sono state rimosse delle voci dal capitolo Problemi noti. Sono stati aggiunti avvisi indicanti che AWS Glue supporta solo le chiavi master del cliente (CMK) simmetriche quando si specificano le impostazi oni di crittografia di catalogo dati e si creano configurazioni di sicurezza. Aggiunta una nota indicante che AWS Glue non supporta la scrittura in Amazon DynamoDB.	9 dicembre 2019

Supporto per driver JDBC personalizzati	Aggiunte informazioni sulla connessione a origini dati e destinazioni con driver JDBC non supportate da AWS Glue in modo nativo, ad esempio MySQL versione 8 e Oracle Database versione 18. Per ulteriori informazioni, vedere Valori JDBC ConnectionType.	25 novembre 2019
Support per il collegamento di SageMaker notebook a diversi endpoint di sviluppo	Sono state aggiunte informazioni su come collegare un SageMaker notebook a diversi endpoint di sviluppo. Aggiornamenti per descrivere la nuova azione della console per il passaggio a un nuovo endpoint di sviluppo e la nuova SageMaker policy IAM. Per ulteriori informazioni, consulta Working with Notebooks on the AWS Glue Console e Creazione di una policy IAM per Amazon Notebooks. SageMaker	21 novembre 2019

Supporto per la versione AWS Glue nelle trasformazioni di machine learning	Sono state aggiunte informazioni sulla definizione della versione AWS Glue in una trasformazione di machine learning per indicare la versione di AWS Glue con cui è compatibile una trasformazione di machine learning. Per ulteriori informazioni, consulta Operare con le trasformazioni basate su machine learning nella console di AWS Glue.	21 novembre 2019
Supporto per il riavvolgimento dei segnalibri di processo	Sono state aggiunte informazioni sul riavvolgimento dei segnalibri di processo per qualsiasi esecuzione precedente, con conseguente rielaborazione dei dati dell'esecuzione del processo successivo solo dall'esecuzione del processo con il segnalibro. Sono descritte due nuove opzioni secondarie per l'opzione `job-bookmark-pause` che consentono di eseguire un processo tra due segnalibri. Per ulteriori informazioni, consulta Monitoraggio dei dati elaborati mediante segnalibri di processo e Parametri speciali usati da AWS Glue.	22 ottobre 2019

Supporto per certificati JDBC personalizzati per la connessione a un archivio dati	Sono state aggiunte informazioni sul supporto AWS Glue di certificati JDBC personalizzati per connessioni SSL a origini dati o destinazioni AWS Glue. Per ulteriori informazioni, consulta Uso di connessioni nella console AWS Glue.	10 ottobre 2019
Supporto per Python wheel	Sono state aggiunte informazioni sul supporto AWS Glue di file wheel (insieme ai file egg) come dipendenze per processi shell di Python. Per ulteriori informazioni, consulta Fornire la propria libreria Python.	26 settembre 2019
Supporto per il controllo delle versioni di endpoint di sviluppo in AWS Glue	Sono state aggiunte informazioni sulla definizione di `Glue version` negli endpoint di sviluppo. `Glue version` determina le versioni di Apache Spark e Python supportate da AWS Glue. Per ulteriori informazioni, consulta Aggiunta di un endpoint di sviluppo.	19 settembre 2019

Supporto per il monitoraggio di AWS Glue tramite l'interfaccia utente di Spark	Sono state aggiunte informazioni sull'utilizzo dell'interfaccia utente di Apache Spark per monitorare ed eseguire il debug di processi ETL AWS Glue in esecuzione sul sistema di processi AWS Glue e applicazioni Spark negli su endpoint di sviluppo AWS Glue. Per ulteriori informazioni, consulta Monitoraggio di AWS Glue mediante l'interfaccia utente di Spark.	19 settembre 2019
Miglioramento del supporto per lo sviluppo di script ETL locali tramite la libreria ETL AWS Glue pubblica	È stato aggiornato il contenuto della libreria ETL AWS Glue per riflettere che AWS Glue versione 1.0 è ora supportata. Per ulteriori informazioni, consulta Sviluppo e test di script ETL in locale tramite la libreria ETL di AWS Glue.	18 settembre 2019
Supporto per l'esclusione delle classi di archiviazione Amazon S3 durante l'esecuzione di processi	Aggiunte informazioni sull'esclusione delle classi di storage Amazon S3 durante l'esecuzione di processi ETL AWS Glue che leggono file o partizioni da Amazon S3. Per ulteriori informazioni, consulta Esclusione delle classi di storage Amazon S3.	29 agosto 2019

Supporto per lo sviluppo di script ETL locali tramite la libreria ETL AWS Glue pubblica	Aggiunte informazioni su come sviluppare e testare script ETL Python e Scala in locale senza la necessità di una connessione di rete. Per ulteriori informazioni, consulta Sviluppo e test di script ETL in locale tramite la libreria ETL di AWS Glue.	28 agosto 2019
Problemi noti	Sono state aggiunte informazioni sui problemi noti in AWS Glue. Per ulteriori informazioni, consulta Problemi noti per AWS Glue.	28 agosto 2019
Supporto per le trasformazioni di machine learning in AWS Glue	Sono state aggiunte informazioni sulle caratteristiche di machine learning fornite da AWS Glue per creare trasformazioni personalizzate. È possibile creare queste trasformazioni al momento della creazione di un processo. Per ulteriori informazioni, consulta Trasformazioni basate su machine learning in AWS Glue.	8 agosto 2019
Supporto per Amazon Virtual Private Cloud condiviso	Aggiunte informazioni sul supporto AWS Glue per Amazon Virtual Private Cloud condiviso. Per ulteriori informazioni, consulta VPC Amazon condivisi.	6 agosto 2019

Supporto per il controllo delle versioni in AWS Glue	Sono state aggiunte informazioni sulla definizione di `Glue version` nelle proprietà del processo. AWS Glue determina le versioni di Apache Spark e Python supportate da AWS Glue. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.	24 luglio 2019
Supporto per le opzioni di configurazione aggiuntive per gli endpoint di sviluppo	Sono state aggiunte informazioni sulle opzioni di configurazione per gli endpoint di sviluppo con carichi di lavoro intensi in termini di memoria. È possibile scegliere tra due nuove configurazioni che offrono maggiore quantità di memoria per esecutore. Per ulteriori informazioni, consulta Uso di endpoint di sviluppo nella console AWS Glue.	24 luglio 2019

Supporto per l'esecuzione di attività di estrazione, trasformazione e caricamento (ETL) utilizzando i flussi di lavoro	Aggiunte informazioni sull'utilizzo di un nuovo costrutto denominato flusso di lavoro per progettare un'attività complessa multiprocesso di estrazione, trasformazione e caricamento (ETL) che AWS Glue è in grado di eseguire e monitorare come una singola entità. Per ulteriori informazioni, consulta Esecuzione di attività ETL complesse utilizzando i flussi di lavoro in AWS Glue.	20 giugno 2019
Supporto per Python 3.6 in processi shell di Python	Sono state aggiunte informazioni sul supporto di Python 3.6 in processi shell di Python. Puoi specificare Python 2.7 o Python 3.6 come proprietà di un processo. Per ulteriori informazioni, consulta Aggiunta di processi shell di Python in AWS Glue.	5 giugno 2019

Supporto di endpoint di cloud privato virtuale (VPC, Virtual Private Cloud)	Sono state aggiunte informazioni sulla connessione diretta ad AWS Glue attraverso un endpoint di interfaccia nel VPC. Quando si utilizza un endpoint VPC di interfaccia, la comunicazione tra il VPC e AWS Glue avviene in modo completo e sicuro all'interno della rete AWS. Per ulteriori informazioni, consulta la pagina relativa all'utilizzo di AWS Glue con endpoint VPC.	4 giugno 2019
Supporto per la registrazione continua in tempo reale per processi AWS Glue.	Sono state aggiunte informazioni sull'attivazione e la visualizzazione dei log dei job di Apache Spark in tempo reale, CloudWatch tra cui i log dei driver, i log di ogni executor e una barra di avanzamento dei job Spark. Per ulteriori informazioni, consulta l'articolo relativo alla registrazione continua dei processi AWS Glue.	28 maggio 2019

Supporto per le tabelle del Catalogo Dati esistenti come origini crawler	Sono state aggiunte informazioni su come specificare un elenco di tabelle del catalogo dati esistenti come origini crawler. I crawler possono quindi rilevare le modifiche agli schemi di tabella, aggiornare le definizioni di tabella e registrare nuove partizioni quando i nuovi dati diventano disponibili. Per ulteriori informazioni, consulta Proprietà dei crawler.	10 maggio 2019
Supporto per le opzioni di configurazione aggiuntive per i processi con elevati requisiti di memoria	Sono state aggiunte informazioni sulle opzioni di configurazione per i processi Apache Spark con carichi di lavoro con elevati requisiti di memoria. È possibile scegliere tra due nuove configurazioni che offrono maggiore quantità di memoria per esecutore. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.	5 aprile 2019
Supporto per classificatori CSV personalizzati	Sono state aggiunte informazioni sull'utilizzo di un classificatore CSV personalizzato per dedurre lo schema di vari tipi di dati CSV. Per ulteriori informazioni, consulta Scrittura di classificatori personalizzati.	26 marzo 2019

Supporto per i tag delle risorse AWS	Sono state aggiunte informazioni sull'utilizzo dei tag delle risorse AWS per gestire e controllare l'accesso alle risorse AWS Glue. È possibile assegnare i tag delle risorse AWS a processi, trigger, endpoint e crawler in AWS Glue. Per ulteriori informazioni sui tag, consultare Tag AWS in AWS Glue.	20 marzo 2019
Supporto del Catalogo Dati per i processi Spark SQL	Sono state aggiunte informazioni sulla configurazione dei processi e degli endpoint di sviluppo di AWS Glue per usare AWS Glue Data Catalog come metastore Apache Hive esterno. In questo modo i processi e gli endpoint di sviluppo eseguono le query Apache Spark SQL direttamente sulle tabelle archiviate in AWS Glue Data Catalog. Per ulteriori informazioni, consulta l'argomento relativo al AWS Glue Data Catalogsupporto di per i processi Spark SQL.	14 marzo 2019

Supporto per processi shell di Python	Aggiunte informazioni sui processi shell di Python e il nuovo campo Maximum capacity (Capacità massima). Per ulteriori informazioni, consulta l'argomento relativo all'aggiunta di processi shell di Python in AWS Glue.	18 gennaio 2019
Supporto per le notifiche quando sono presenti modifiche di database e di tabelle	Aggiunte informazioni sugli eventi generati a causa di modifiche al database, alla tabella e alle chiamate all'API della partizione. È possibile configurare le azioni in Eventi per rispondere a questi eventi CloudWatch . Per ulteriori informazioni, consulta Automazione AWS Glue con CloudWatch eventi.	16 gennaio 2019
Supporto per la crittografia delle password di connessione	Aggiunte informazioni sulla crittografia di password utilizzate in oggetti di connessione. Per ulteriori informazioni, consulta Crittografia delle password di connessione.	11 dicembre 2018
Supporto per le autorizzazioni a livello della risorsa e per le policy basate sulla risorsa	Aggiunte informazioni sull'util izzo di autorizzazioni a livello della risorsa e delle policy basate sulla risorsa con AWS Glue. Per ulteriori informazioni, consulta gli argomenti indicati in Sicurezza in AWS Glue.	15 ottobre 2018

Support per SageMaker notebook	Sono state aggiunte informazioni sull'utilizzo di SageMaker notebook con endpoint di sviluppo. AWS Glue Per ulteriori informazioni, consulta Gestione di notebook.	5 ottobre 2018
Supporto per la crittografia	Aggiunta di informazioni sull'uso della crittografia con AWS Glue. Per ulteriori informazioni, consulta Crittografia dei dati inattivi, Crittografia dei dati in transito e Configurazione della crittografia in AWS Glue.	24 agosto 2018
Supporto per i parametri di processo Apache Spark	Aggiunta di informazioni sull'uso dei parametri Apache Spark per migliorare il debug e la profilatura dei processi ETL. È possibile tenere facilmente traccia dei parametri di runtime come i byte letti e scritti, l'uso della memoria e il carico della CPU del driver e degli executor, nonché la distribuzione dei dati tra executor dalla console AWS Glue. Per ulteriori informazioni, consulta Monitoring AWS Glue Using CloudWatch Metrics, Job Monitoring and Debugging e Working with Jobs on the Console. AWS Glue	13 luglio 2018

Supporto di DynamoDB come origine dati	Aggiunta di informazioni sul crawling di DynamoDB e su come usarlo come origine dati dei processi ETL. Per ulteriori informazioni, consulta Catalogazione di tabelle con un crawler e Parametri di connessione.	10 luglio 2018
Aggiornamenti alla procedura di creazione di un server notebook	Aggiornamento delle informazioni su come creare un server notebook in un'istanza Amazon EC2 associata a un endpoint di sviluppo. Per ulteriori informazioni, consulta Creazione di un server notebook associato a un endpoint di sviluppo.	9 luglio 2018
Aggiornamenti ora disponibili tramite RSS	È ora possibile abbonarsi a un feed RSS per ricevere notifiche sugli aggiornamenti alla Guida per gli sviluppatori di AWS Glue.	25 giugno 2018
Supporto delle notifiche di ritardo per i processi	Aggiunte informazioni sulla configurazione di una soglia di ritardo durante l'esecuzione di un processo. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.	25 maggio 2018

Configurazione di un crawler per aggiungere nuove colonne	Sono state aggiunte informazioni sulla nuova opzione di configurazione per i crawler,. MergeNewColumns Per maggiori informazioni, consulta Configurazione di un crawler.	7 maggio 2018
Supporto del timeout dei processi	Aggiunte informazioni sull'impostazione di una soglia di timeout durante l'esecuzione di un processo. Per ulteriori informazioni, consulta Aggiunta di processi in AWS Glue.	10 aprile 2018
Supporto script Scala ETL e processi trigger basati su stati di esecuzione aggiuntivi	Informazioni aggiunte sull'utilizzo di Scala come linguaggio di programmazione ETL. Ora l'API trigger supporta anche l'attivazione se viene soddisfatta una qualsiasi delle condizioni (in aggiunta a tutte le condizioni). Inoltre, i processi possono essere attivati sulla base di un'esecuzione processo "non riuscita" o "arrestata" (in aggiunta a un'esecuzione processo "riuscita").	12 gennaio 2018

Aggiornamenti precedenti

La tabella seguente descrive le modifiche importanti apportate in ogni versione della Guida per sviluppatori AWS Glue prima di gennaio 2018.

Modifica	Descrizione	Data
Supporto origini dati XML e nuova opzione di configurazione crawler	Informazioni aggiunte sulla classificazione di origini dati XML e nuova opzione crawler per modifiche della partizione.	16 novembre 2017
Nuove trasforma zioni, supporto per motori di database Amazon RDS aggiuntivi e miglioramenti degli endpoint di sviluppo	Informazioni aggiunte sulle trasformazioni di filtraggio e mappatura, supporto per Amazon RDS Microsoft SQL Server e Amazon RDS Oracle e nuove caratteri stiche per gli endpoint di sviluppo.	29 settembre 2017
Versione iniziale di AWS Glue	Questa è la versione iniziale della Guida per gli sviluppatori di AWS Glue.	14 agosto 2017

AWS Glossario

Per la AWS terminologia più recente, consultate il AWS glossario nella sezione Reference.Glossario AWS

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.