Orchestrare pipeline ETL su AWS con AWS Glue, AWS StepFunctions e AWS Cloudformation

1 Aprile 2021 - 7 min. read

Amazon S3 *AWS CloudFormation* *AWS Glue* *AWS Step Functions* *ETL*

L'analisi dei Big Data sta diventando sempre più importante per delineare le principali scelte di business in aziende di tutte le dimensioni. Tuttavia, raccogliere, aggregare, unire e analizzare (validare) enormi quantità di dati archiviati in diversi datastore con una struttura eterogenea (ad esempio database, CRM, testo non strutturato, ecc.) è spesso un compito arduo e richiede molto tempo.

Il cloud computing viene spesso in soccorso fornendo soluzioni di storage, computing e data lake economici e scalabili e, in particolare, AWS si pone come leader di settore grazie al servizio Glue / S3, molto versatile e che consente agli utenti di importare trasformazioni e normalizzare set di dati di tutte le dimensioni. Inoltre, Glue Catalog e Athena consentono agli utenti di eseguire facilmente query SQL basate su Presto su dati normalizzati presenti nei data lake S3, i cui risultati possono essere facilmente archiviati e analizzati mediante strumenti di business intelligence come QuickSight.

Nonostante i grandi vantaggi offerti da Glue e S3, la creazione e il mantenimento di complessi flussi ETL multistadio di Glue è spesso un'attività che richiede molto tempo: i job di Glue sono, per loro natura, disaccoppiati e il loro codice è memorizzato su S3. Ciò rende molto difficile integrare diversi lavori e svilupparli in un progetto software ben strutturato e coeso.

Un piccolo aiuto ci viene fornito dai Glue workflows: utilizzando queste pipeline integrate di Glue, è possibile eseguire automaticamente diversi workflows e / o crawler in un determinato ordine. Tuttavia, a questo strumento, seppur molto utile, mancano diverse funzionalità molto comuni a molti strumenti di controllo del flusso, come diramazioni condizionali (if-else), loop, mappe dinamiche e step personalizzati.

Un'alternativa migliore è fornita da AWS StepFunctions. StepFunctions è uno strumento di orchestrazione AWS molto potente e versatile in grado di gestire la maggior parte dei servizi AWS, direttamente o tramite integrazioni con funzioni lambda.

Nelle sezioni seguenti spiegheremo come funzionano le StepFunctions e come integrare e sviluppare sia l'infrastruttura che il codice per Glue Jobs.

Perchè dovremmo aver bisogno di StepFunctions?

Proviamo ad elaborare un job ETL molto semplice, ma allo stesso tempo realistico, per l'importazione e la trasformazione di dati, in modo da spiegare perché un servizio di orchestrazione in generale e, in particolare su AWS, StepFunctions, rappresenta una componente essenziale nella toolbox di un data engineer. Ecco i componenti logici per il nostro flusso di lavoro ETL di esempio:

1. I dati devono essere acquisiti da un database relazionale. Schemi e tabelle multipli.
2. I dati acquisiti devono essere caricati su S3 e sottoposti a scansione per estrarre un Glue DataCatalog per le query AWS Athena.
3. È necessario unire diverse tabelle del catalogo dati, utilizzando regole non banali per creare un set di dati su S3 da utilizzare in un processo di Machine Learning per la segmentazione dei clienti.
4. L'output del lavoro di segmentazione dei dati deve essere archiviato sia nel data lake di S3, sia essere copiato, aggiornato, nel database relazionale per l'accesso da parte di altri strumenti aziendali.

Questi quattro passaggi descrivono un caso d'uso relativamente semplice ma molto comune. Ora proviamo a redigere un elenco di passaggi che dobbiamo eseguire in AWS Glue per completare il flusso di lavoro descritto finora:

1. Scansionare il database originale tramite una connessione JDBC.

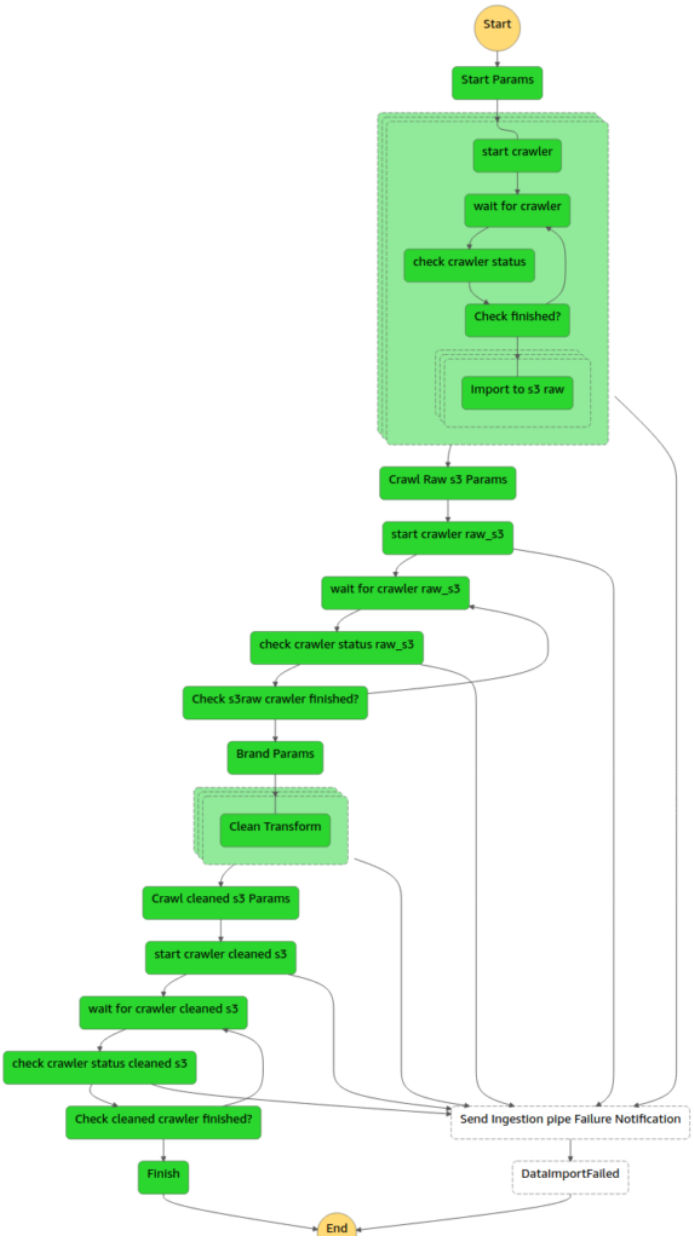
2. Utilizzare un Glue Job per spostare i dati dal database a S3. Alcune tabelle possono utilizzare i segnalibri ma altre no.
3. Scansionare il bucket S3 di destinazione.
4. Eseguire un job di Glue Spark dedicato per operare una join sul data lake di S3. Scrivere i risultati su un'altra partizione o bucket S3.
5. Eseguire la scansione della partizione di destinazione per rendere facilmente interrogabili i risultati della join mediante AWS Athena.
6. Lanciare il job di ML (SageMaker o workflows di Glue ML).
7. Scansionare il set di dati risultante.
8. Eseguire un processo ETL finale di Glue per caricare il nuovo set di dati nel database originale.

Tutti questi passaggi devono essere eseguiti nell'ordine indicato e, in caso di problemi, sarebbe bello essere avvisati e avere un modo semplice per capire cos'è andato storto.

Senza utilizzare AWS StepFunctions, la gestione manuale di questi passaggi sarebbe estremamente difficoltosa e probabilmente avremmo bisogno di uno strumento di orchestrazione esterno o di creare uno script di orchestrazione personalizzato da eseguire su un'istanza EC2 o su un container Fargate.

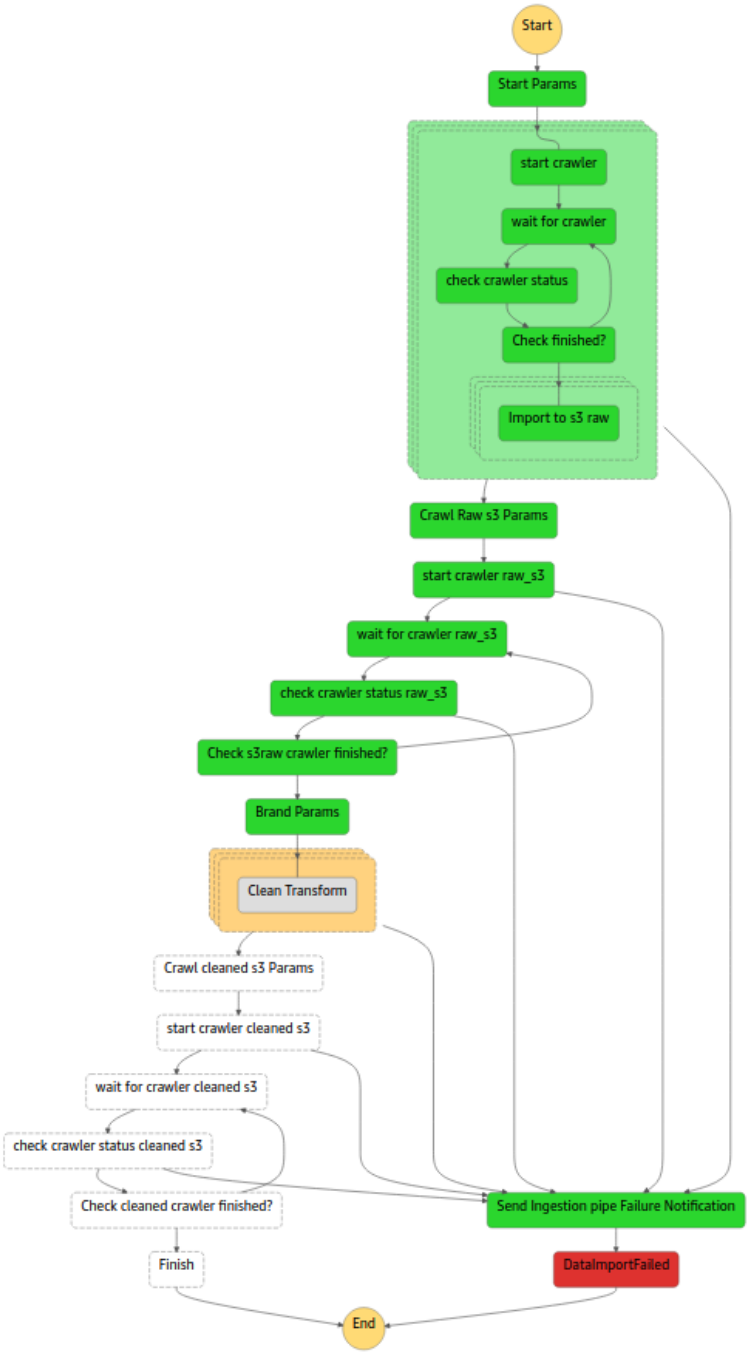
Ma perché preoccuparsi? AWS StepFunctions fa tutto questo per noi, ed essendo in grado di interagire direttamente con molti servizi AWS, molte integrazioni sono un gioco da ragazzi: ad esempio, con poche righe di linguaggio Stepfunctions, possiamo catturare tutti gli errori in una pipe e inoltrarli a un topic SNS per ricevere un'e-mail in caso di errore (o una notifica slack, SMS o qualsiasi altra alternativa si preferisca)

La gestione di flussi complessi diventa così sicura e relativamente facile. Ecco un esempio:



StepFunctions flow

Se uno di questi passaggi dovesse fallire, riceveremo una notifica tramite posta elettronica dal topic SNS, avremmo quindi, un feedback visivo del passaggio non riuscito e anche i log corrispondenti.



Step di errore e relativi log

StepFunctions sembra quindi essere un jolly perfetto, con molte buone caratteristiche e nessun inconveniente significativo, tuttavia, come tutti sappiamo, questo non è quasi mai vero nel mondo IT, quindi qual è il trucco?

Il vero problema è la gestione del codice: il linguaggio di StepFunctions è basato su un modello JSON dichiarativo, risultando quindi non banale da scrivere e mantenere, anche utilizzando strumenti dedicati come plug-in specifici per Visual Studio.

Inoltre, sarebbe molto utile poter mantenere sia il codice StepFunctions che i Glue Jobs e l'eventuale codice Lambda in un unico progetto integrato.

Cloudformation con Troposphere o AWS CDK

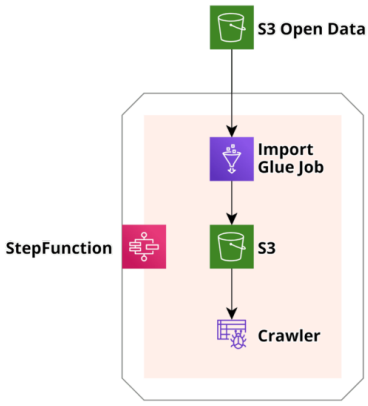
Lo strumento più ovvio che possiamo utilizzare per mantenere le StepFunctions, i Glue Jobs e il resto della nostra infrastruttura ETL, in modo coeso, è Cloudformation, da integrare come strumento di distribuzione per tutto il progetto. Tuttavia, il codice Cloudformation è un linguaggio YML / JSON dichiarativo non troppo diverso dal codice delli funzioni di StepFunctions, e includere tale codice in questi template, di solito è piuttosto doloroso poiché implica l'inclusione di stringhe JSON complesse nel nostro file YML di Cloud Formation.

Una soluzione molto più efficace consiste nel creare un template di Cloudformation, utilizzando un linguaggio di programmazione di alto livello come AWS CDK che supporta molti linguaggi (TypeScript, Python, e Java).

Optando per Python, che risulterà spesso una buona scelta poiché i lavori ETL saranno probabilmente scritti comunque in Python, si avrà la possibilità di utilizzare Troposphere invece di AWS CDK come framework Cloudformation, che è molto più versatile in diverse situazioni.

Inoltre le StepFunctions possono essere generate a partire dal python Step Functions Framework come mostreremo nell'esempio seguente (Troposphere + Python step function SDK).

In questo esempio molto semplice vogliamo dimostrare come creare un semplice workflow per scaricare un dataset sul Covid da un bucket AWS S3 OpenData pubblico, salvarne un piccolo sottoinsieme in un bucket S3 diverso e sottoporlo a scansione per prepararlo alle query mediante AWS Athena. Questo esempio di workflow base può essere esteso a piacimento! Ecco uno schizzo di base dell'infrastruttura:



Infrastruttura per il nostro esempio di flow

Prima di tutto procediamo installando la CLI di AWS e le librerie richieste da python:

```
pip install troposphere stepfunctions.
```

Una volta completata l'installazione, scarichiamo il codice di esempio dal nostro repository e ci ritroveremo con un file troposphere_main.py che contiene la **rappresentazione troposphere** dell'intera infrastruttura (vedi sketch) e altre cartelle contenenti il codice python delle varie funzioni Lambda (start_crawler, check_crawler status), infine un file README che spiega come eseguire il progetto. Dopodiché dovremo creare un bucket S3 come supporto per la distribuzione di Cloudformation con il nome che preferiamo.

Seguendo le istruzioni presenti nel README, possiamo semplicemente eseguire il file principale, lanciando in una console python troposphere_main.py. Eseguendo questo script, compileremo il codice troposphere in un formato JSON compatibile con Cloudformation. Fatto ciò, siamo pronti per lanciare il nuovo template di AWS Cloudformation:

aws cloudformation package --template-file troposphere_main.json --s3-bucket <YOUR CLOUDFORMATION S3 BUCKET> --s3-prefix '<THE PATH YOU PREFER>' --output-template-file troposphere_main.yml

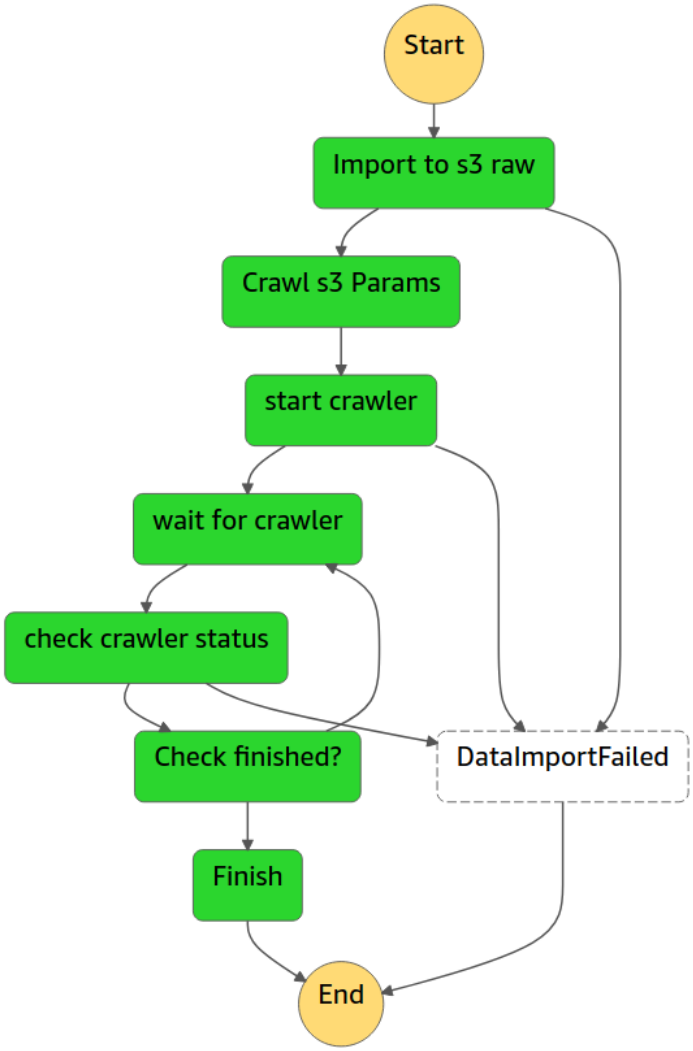
Questo comando prende come input il file JSON creato da Troposphere, carica su S3 il codice delle funzioni Glue e lambda, a cui si fa riferimento, come percorsi locali ed infine

restituisce un altro modello di Cloudformation (questa volta in YML), in cui i riferimenti ai percorsi locali sono stati modificati nei corrispondenti riferimenti su S3 (Qui ulteriori informazioni).

Finalmente siamo pronti per distribuire il modello Cloudformation utilizzando il comando:

```
aws cloudformation deploy --template-file ./troposphere_main.yml --stack-name testStepfunctionsStack --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND
```

In questo modo creeremo il "**testStepfunctionsStack**" che contiene l'infrastruttura descritta prima. Ora possiamo accedere alla console di AWS StepFunctions ed eseguire la nuova funzione (test-stepfunctions-glue), il workflow verrà eseguito e noi vedremo importati i dati Covid.



Il nostro flow di esempio completato

Sebbene questo sia solo un esempio molto basico, è importante notare che tutto il codice presentato è racchiuso nello stesso progetto e quindi facilmente estendibile a livello di flusso senza però perdere il controllo dei vari componenti: basta usare Git per il controllo di versione e Cloudformation per i deploy!

Conclusioni

Abbiamo dimostrato che le funzioni di StepFunctions sono un ottimo modo per orchestrare i flussi basati su AWS in generale e in particolare le pipeline ETL! Inoltre, abbiamo condiviso un esempio di come utilizzare Troposphere e Python StepFunctions SDK per sviluppare, in un unico progetto python, sia una funzione di StepFunctions che il codice dei suoi vari componenti.

Ed eccoci arrivati alla fine! Lasciateci un commento o contattateci per qualsiasi dubbio, domanda o idea!

Ci vediamo puntuali tra due settimane con un nuovo articolo su **#proud2becloud**!

Matteo Moroni

DevOps e Solution Architect di beSharp, mi occupo di sviluppare soluzioni Saas, Data Analysis, HPC e di progettare architetture non convenzionali a complessità divergente. Appassionato di informatica e fisica, da sempre lavoro nella prima e ho un PhD nella seconda. Parlare di tutto ciò che è tecnico e nerd mi rende felice!

Alessandro Gaggia

Head of software development di beSharp, Full-Stack developer, mi occupo di garantire lo stato dell’arte di tutta la nostra codebase. Scrivo codice in quasi ogni linguaggio, ma prediligo

Typescript. Respiro Informatica, Game design, Cinema, Fumetti e buona cucina. Disegno per passione!
