ETL ORCHESTRATION SU AWS CON AWS STEP FUNCTIONS

AWS Lambda | Data and Analytics | Serverless

beSharp | 26 Novembre 2020

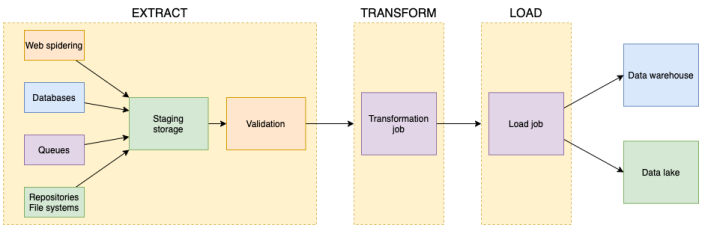
Negli ultimi anni data engineering, data governance e data analysis sono diventati importanti argomenti di discussione.

La necessità di effettuare decisioni scaturite dall'analisi dei dati, infatti, ha fatto crescere l'esigenza di collezionare e analizzare dati in diversi modi ed AWS ha dimostrato un particolare interesse in questo campo, sviluppando diversi strumenti per raggiungere questi obiettivi di business.

Prima di poter permettere a figure come i data analysts di esplorare e visualizzare i dati, è necessario eseguire un passaggio cruciale. Questo processo è generalmente chiamato **ETL** (**extract**, **transform**, and **load**) e, solitamente, è molto lontano dall'essere semplice da eseguire.

Chi svolge queste operazioni ha la responsabilità dei seguenti compiti:

- **Estrazione**: i dati arrivano solitamente da numerose ed eterogenee fonti, come database, web spidering, flussi di dati, dati semi-strutturati eccetera.
  Data la potenziale diversità delle sorgenti, una validazione dei dati che arrivano nel nostro dominio è necessaria. In questo modo si eviterà di introdurre informazioni con formati o schemi disomogenei.
- **Trasformazione**: dopo l'estrazione dei file validi in uno storage intermedio, un insieme di trasformazioni vengono generalmente applicate sui dati ricevuti. Tipicamente, questo passaggio è anche identificato come preparazione dei dati e comporta la rimozione di dati incompleti o inesatti, l'aggregazione con altri dati, la deduplicazione dei record e tutti gli step di normalizzazione e codifica.
- **Caricamento**: infine, i dati in precedenza validati e trasformati vengono salvati nei data store persistenti. Questi data store possono essere di diversa natura in base alle necessità di business. Due dei più comuni tipi di data storage per l'ETL sono i data warehouse e i data lake. I primi sono generalmente utilizzati per salvare dati con uno schema rigoroso in database relazionali come Amazon Redshift, mentre gli altri, molto utilizzati per machine learning, analisi esplorativa, analisi di big data e visualizzazione, sono comunemente formati da dati semi-strutturati. L'abbinamento di Amazon S3 (per lo storage a basso costo) ed Amazon Athena (per le veloci query sui file con tecnologia serverless), permette un eccellente sviluppo di data lake su AWS.



ETL su AWS

Nell'introduzione abbiamo già citato alcuni servizi AWS considerati importanti componenti di una infrastruttura dedicata al processo di ETL.

Oltre a quelli visti, ne esistono altri diventati lo stato dell'arte nella costruzione di pipeline di ingestion di dati.

Vediamoli insieme all'inetrno di ciascuna fase del processo:

L'estrazione dei dati

L'estrazione efficace dei dati da cui un'azienda può trarre vantaggio può avvenire con diversi ritmi e dimensioni. Da centinaia di ordini al secondo inviati da un e-commerce durante il black Friday, all'ingestion di report di business mensili. L'infrastruttura che ospita il flusso di ETL deve essere sempre pronta ad accogliere le nuove informazioni nello storage intermedio.

Alcuni servizi AWS possono aiutare ad assecondare le diverse necessità di business facendo convogliare tutti i dati in uno stesso posto, comunemente identificato con i bucket S3.

In base alla mole dei dati che ci si aspetta, è possibile incaricare diversi servizi AWS per la validazione dei file: per il miglior rapporto costo/performance in caso di event-driven e piccoli file, sceglieremo AWS Lambda. Al contrario, quando ci si aspetta che i dati da gestire possano causare il superamento dei limiti computazionali del primo, sceglieremo AWS Glue con dei batch job schedulati.

La trasformazione

La trasformazione dei dati in ingresso nella pipeline è generalmente un lavoro pesante ed è, pertanto, eseguito in batch. Per questa ragione, i migliori candidati sono le risorse di Glue. AWS Glue è basato su cluster serverless in grado di scalare i worker in maniera trasparente fino a raggiungere terabyte di RAM e migliaia di core.

Per una scalabilità ottimale è possibile far girare script Python o codice PySpark e Spark. Ricordiamo che i job di tipo PythonShell sono maggiormente indicati per carichi di grandezza medio-bassa perché non possono scalare oltre un solo worker (4 vCPU e 16 GB di RAM).

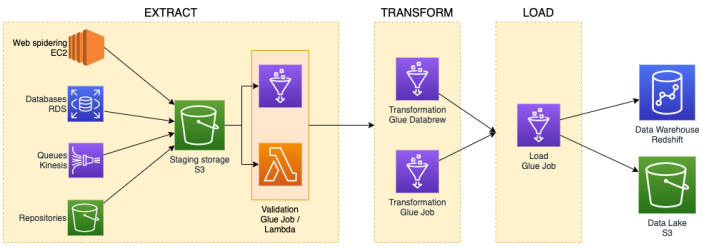
Sebbene con Spark, Glue Job, e Glue Studio sia possibile creare job di trasformazione in maniera molto meticolosa, è possibile che il nuovo servizio AWS Glue DataBrew possa soddisfare meglio queste necessità grazie anche alla sua interfaccia web completa e chiara.

È importante notare, però, che per permettere ai Glue Job di avere accesso ai dati da un singolo punto, AWS Glue incorporerà nella sua interfaccia il Data Catalog. Il Glue Data Catalog è l'archivio dei dati presenti nei nostri data store che è poi usato per l'ingestion. Per far sì che questo catalogo resti aggiornato, verrà utilizzato un altro componente di AWS Glue: il Crawler. Sarà quest'ulimo a dare la visibilità sui nuovi file e partizioni ai job che tenteranno di raggiungere i dati dalle sorgenti.

Il caricamento nelle destinazioni

Dopo il processo di trasformazione, uno specifico Glue Job, o lo stesso componente utilizzato nello step precedente, può salvare i dati validati, puliti e trasformati nelle destinazioni per analisi e visualizzazione attraverso, per esempio, Amazon QuickSight.

Per preservare la privacy sui dati sensibili che passano attraverso la pipeline, è importante effettuare il set-up di misure di sicurezza come la cifratura con KMS per i dati at rest nei bucket e database e la protezione con SSL dei trasferimenti di dati in transit. In più, è una buona pratica procedere con l'offuscazione delle informazioni personali salvate nel nostro dominio.

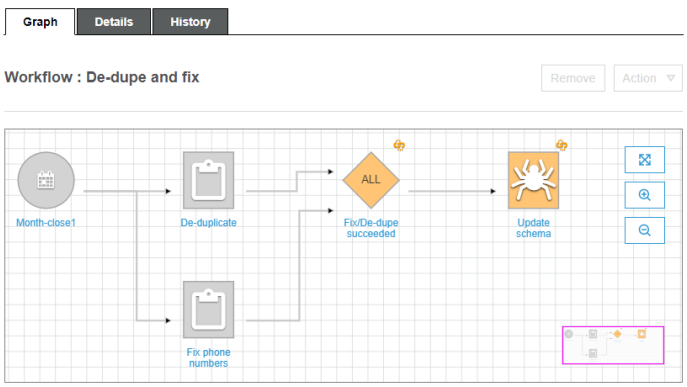


Orchestrazione del processo di ETL su AWS

La gestione dei bit e byte che scorrono nella pipeline di ETL, non è banale.

Per applicare una corretta governance sui dati prodotti dal processo, occorre effettuare dei controlli di qualità ad-hoc. È importante, infatti, controllare eventuali inadempimenti dei requisiti di business, come la mancanza di dati necessari nel data lake causati da errori di validazione.

Con gli strumenti di AWS Glue è possibile creare flussi e trigger per poter costruire delle pipeline di ingestion di dati. In questo caso, però, le soluzioni possibili che si possono raggiungere sono molto limitate dalla mancanza di direttive che permettano i cicli, retries, gestione dell'errore, e invocazione di altri servizi AWS al di fuori di AWS Glue.



Non tutto però è perduto: in AWS uno strumento che permette l'orchestrazione scrupolosa di servizi serverless esiste. Parliamo di AWS Step Functions. Questo strumento permette la gestione di

logiche di retry e gestione di errori, facilitando la reazione delle nostre applicazioni distribuite in caso di comportamenti inaspettati.

Nelle sezioni seguenti, scopriremo e utilizzeremo le Step Functions per l'orchestrazione di un caso realistico di ETL.

Cominciamo!

AWS Step Functions

AWS Step Functions permette la costruzione di macchine a stati finiti altamente scalabili che, nel caso della configurazione express, può gestire fino ad un centinaio di migliaia di cambi di stato al secondo.

Un workflow costruito con questo servizio è principalmente composto da:

- Stati: gli step per cui si passa durante il workflow.
- Direttive: branching, retries, gestione degli errori, elaborazione parallela e cicli.
- Integrazioni con altri servizi: grazie a diverse integrazioni delle Step Functions con altri servizi AWS, è possibile invocare alcuni dei tanti servizi AWS serverless. Le quasi inevitabili funzioni Lambda sono una di questi e possono agire da invoker degli altri servizi AWS che non sono direttamente integrati con le Step Functions.

Tutti questi componenti, poi, sono collegati tramite Amazon State Language, un linguaggio basato sul JSON per generare la definizione di una Step Function.

Inoltre, un aspetto molto importante di quest'ultimo è la possibilità di monitorare in tempo reale ogni esecuzione dalla console.

Il caso d'uso

Mettiamoci ora nei panni di un architetto che deve sviluppare un flusso di ETL orchestrato.

Analizziamo i requisiti di business del nostro cliente immaginario per poi creare una soluzione adatta.

Requisiti del cliente

Per il caso d'uso che andremo ad architettare, faremo finta che il cliente abbia i seguenti requisiti di business in modo da guidare le nostre decisioni:

- È necessaria una tabella di audit in modo da essere sempre a conoscenza dello stato di ogni dataset. Questa potrà essere in futuro utilizzata per la creazione di una interfaccia web utilizzata per fini di controllo.
- In questo momento, i dati in ingresso possono essere divisi in:
  - Dataset di tipo A: file giornalieri con dimensioni da 1 MB a 20 MB

- Dataset di tipo B: report mensili con dimensioni da 15 MB a 50 MB

- I file non validi devono essere spostati in un bucket dei file invalidi per permettere una conseguente analisi
- Una email di notifica deve essere inviata immediatamente in caso di fallimento di uno step
- Le fasi di trasformazione e caricamento sono gestite con codice Spark fornito dagli analisti dell'organizzazione cliente

Analisi dei requisiti

Per avere un audit trail dello stato di ogni dataset che passa nella pipeline, possiamo utilizzare una tabella DynamoDB. Questa tabella sarà poi automaticamente popolata all'inserimento di un nuovo file nel bucket di input tramite una funzione Lambda e la gestione dei Cloudwatch Event.

Grazie alle direttive permesse da Step Functions per interfacciarsi con DynamoDB, è possibile effettuare *get*, *insert*, *update* e *delete* dei record nelle tabelle dal flusso di orchestrazione. In questo modo sarà possibile aggiornare direttamente lo stato di ogni file quando viene validato, trasformato o caricato.
La tabella sarà strutturata nel seguente modo:

- Tipo di dataset (giornaliero o mensile) come partition key
- Nome del bucket e key del file come sort key
- Stato dell'ingestion impostato a NEW quando il file è appena creato
- Dimensione del file – può essere utilizzato in futuro nel caso fosse necessario gestire dataset più grandi. La nostra Step Function potrà essere aggiornata per indirizzare questi dataset verso un Glue Job piuttosto che una Lambda.

dataset_name	bucket_key	ingestion_state	file_size_in_mb
A	org.dev.staging.bucket#A/20201115.csv	NEW	8.22
A	org.dev.staging.bucket#A/20201116.csv	NEW	12.34
B	org.dev.staging.bucket#B/202010.csv	NEW	46.2

Guardando la dimensione dei due tipi di dataset che il cliente si aspetta, comunque, le funzioni Lambda sono il migliore candidato per raggiungere il miglior rapporto costo/performance.

Inoltre, le funzioni lambda possono essere adottate per reperire inizialmente la lista dei dataset che devono essere gestiti ed anche per muovere quelli scartati nella validazione nel bucket dei file invalidi.

È possibile impiegare l'integrazione delle AWS Step Functions con SNS per notificare prontamente il cliente quando un stato di errore è raggiunto a causa di un fallimento di uno degli step di ETL.

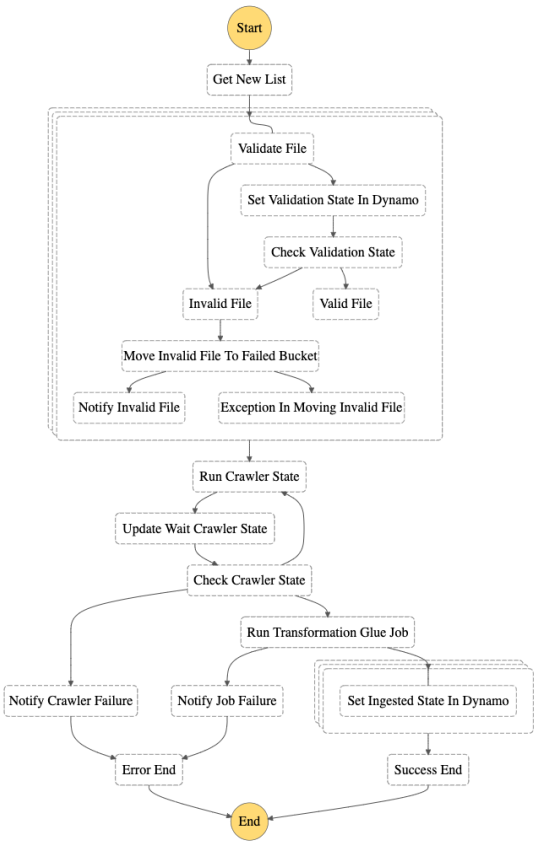
La macchina a stati finiti

È stato deciso di eseguire la Step Function in maniera schedulata, in modo da ridurre i costi di AWS Glue, gestendo i file in batch ad ogni run.

Per poter validare tutti i nuovi file arrivati dopo l'ultima esecuzione della Step Function, una funzione Lambda reperirà i record che hanno ingestion_state impostato a NEW. In questo modo, il workflow sarà capace di iterare attraverso la lista dei file appena reperita da Dynamo per eseguire le fasi di estrazione e validazione per ognuno di essi.

Come è possibile vedere dal diagramma degli stati, il risultato della lambda di validazione viene salvato nella tabella dynamo e poi usato per distinguere i file validi da quelli non validi.

Nel caso la lambda di validazione dovesse andare nello stato di errore, ma il meccanismo di retry non si verifichi, il file sarà comunque categorizzato come non valido ed automaticamente spostato nel bucket dei file non validi insieme all'invio della notifica al cliente.



Quando tutte le iterazioni sono completate e i file non validi sono stati scartati per una analisi aggiuntiva, sarà eseguito il Crawler di Glue per poter permettere l'aggiornamento del data catalog con i nuovi file.

Al momento della scrittura di questo articolo, non esiste ancora una integrazione che permetta l'esecuzione diretta del Crawler dalle Step Functions. Pertanto, spetterà ad una funzione Lambda effettuare questa operazione. Attraverso uno stato di tipo wait, è possibile specificare dopo quanto tempo verrà controllato nuovamente lo stato del crawler in modalità polling.

A questo punto, i dati sono validati e catalogati e siamo pronti per trasformarli e caricarli nel data lake.

Le Step Function sono state recentemente arricchite con la possibilità di eseguire i Glue Job grazie ad una integrazione con AWS Glue. Questo ci permetterà di eseguire gli script di trasformazione dei data analyst senza la necessità di una funzione Lambda nel mezzo per l'invocazione.

Infatti, l'esecuzione sincrona dei servizi partendo da una Step Function permette alla macchina a stati finiti di fermarsi nello stato corrente finché il servizio chiamato non ha terminato il suo lavoro. In questo modo, avremo la possibilità di distinguere i Job eseguiti con successo da quelli falliti. Nel primo caso è possibile proseguire nel workflow di ETL mentre, nel secondo, una notifica sarà mandata nuovamente al cliente.

Il prossimo passaggio è necessario solo per permettere l'auditing del processo di trasformazione e caricamento, infatti la Step Function gestirà l'aggiornamento della tabella Dynamo impostando l'ingestion_state a INGESTED per tutti i file che sono stati caricati durante questa esecuzione.

Tips and tricks

Abbiamo visto che le funzioni Lambda impiegate nelle Step Functions sono molto spesso utilizzate semplicemente per invocare altri servizi, reperire liste ed effettuare controlli sullo stato di altre risorse AWS.

Per questo motivo, infatti, è possibile mantenere una singola codebase ed una singola funzione Lambda per eseguire questi compiti piuttosto semplici, selezionando i metodi da eseguire in base allo stato con il quale la lambda è stata invocata.

Un esempio di questo comportamento è lo stato che reperisce la lista dei nuovi file dalla tabella dynamo, la quale definizione è la seguente:

```
"Get New List": {
   "Type": "Task",
   "Resource": "arn:aws:states:::lambda:invoke.sync",
   "Parameters": {
       "FunctionName": "arn:aws:lambda:eu-west-1:XXXXXXXXXXXXX:function:glue-orc-sfn-l
ambda",
       "Payload": {
           "NeededState": "NEW",
           "DatasetName.$": "$.DatasetName",
           "SFNState.$": "$$.State.Name"
       }
   },
   "ResultPath": "$.NewFilesList",
   "Next": "New Files Loop"
}
```

Questo snippet di codice Amazon State Language definisce lo stato *Get New List*. Ai parametri necessari alla funzione Lambda è stato aggiunto, però, il nome dello stato corrente della Step Function in modo da eseguire il metodo Python necessario.

```
def action_switcher(sfn_state: str) -> function:
   switcher = {
       "Get New List": get_new_list,
       "Run Crawler State": run_crawler,
       "Move Invalid File To Failed Bucket": move_file_to_bucket
   }
   return switcher.get(sfn_state, lambda: None)
```

Questo è stato possibile grazie alla capacità di accedere ai context object delle Step Function che possono essere molto utili in molte definizioni.

Conclusioni

In questo articolo abbiamo esplorato il mondo dell'ETL ed abbiamo visto come può essere complesso orchestrare una data pipeline resistente, scalabile e facilmente controllabile.

Con le Step Functions abbiamo visto, poi, come un singolo servizio sia sufficiente per architettare una soluzione resiliente permettendo il business analytics in qualsiasi scala.

E voi? Come gestite l'ETL su AWS?

Molte gemme nascoste sono ancora disponibili per lavorare ogni giorno meglio con i dati.

Continuate a seguirci per scoprirle tutte 🙂

Ci vediamo tra 14 giorni qui su **#Proud2beCloud** con un nuovo articolo!

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it