Come Creare un Data lake/DataMart su Amazon S3 con pre-ingestion flows ed ETL

24 Novembre 2022 - 7 min. read

AWS Glue | AWS Organizations | AWS Transfer Family | Data Lake | ETL | SFTP

I Big Data hanno acquisito importanza a partire da metà degli anni 2000 con lo sviluppo della metodologia MapReduce in Google e hanno continuato a crescere a un ritmo vertiginoso grazie allo sviluppo di strumenti sempre migliori: Apache Hadoop, Spark, Pandas sono stati tutti sviluppati in questo lasso di tempo.

Parallelamente, sempre più Cloud provider e service integrator hanno iniziato a offrire soluzioni gestite di Big Data/Data Lake (da Cloudera ad AWS Glue) per soddisfare la crescente domanda di aziende sempre più desiderose di analizzare i propri dati per trarne valore.

Nel frattempo il termine "Big Data" ha smesso di essere una buzzword ed è stato soppiantato in questo ruolo da parole più nuove e più accattivanti come Blockchain e Quantum computing, ma questo non ha diminuito la necessità delle aziende di sfruttare i dati per indirizzare meglio i propri clienti, ottimizzare il proprio prodotto e perfezionare i propri processi.

In questo breve articolo descriveremo una delle soluzioni che abbiamo messo a terra per rispondere a questa necessità. Vedremo come creare un **Data Lake dinamico multi-account decentralizzato su AWS, sfruttando Glue, Athena e Redshift**.

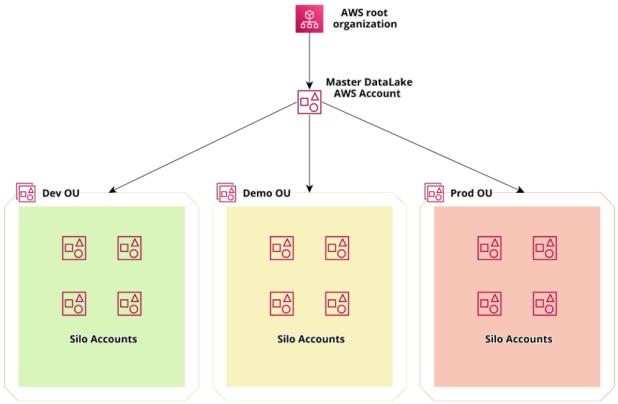
Il problema

Solitamente i data lake sono **repository di dati non strutturati** che raccolgono dati di input da fonti di dati eterogenee come database SQL legacy, database di documenti (ad esempio

MongoDB), database di tipo chiave-valore (ad esempio Cassandra) e file raw da varie fonti (server SFTP, Samba, Object storage).

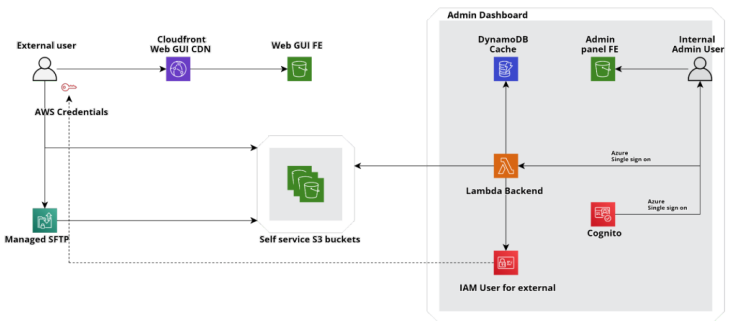
In questo caso il nostro obiettivo è quello di suddividere il database in modo che ogni progetto interno nella struttura aziendale del cliente possa accedere solo al proprio silo di dati segregato. Per farlo, imposteremo anche le operazioni ETL necessarie. Un numero selezionato di utenti dell'amministrazione generale sarà in grado di accedere ai dati da diversi silos per leggerli e aggregarli per l'analisi a livello aziendale, la business intelligence e il reporting generale.

Per soddisfare questi requisiti e garantire la massima segregazione possibile tra diversi silos, abbiamo deciso di suddividere i progetti in diversi account AWS utilizzando AWS Organizations. La struttura degli account è rappresentata nel diagramma sottostante:



Per aggiungere valore alla soluzione, un altro requisito era quello di poter creare credenziali per terze parti per inviare i dati direttamente al data lake sia con API, che con SFTP.

Ciò significa che ogni account conterrà non solo il bucket S3 con i dati e i job di Glue/Step Functions necessari per trasformarli, diversi per ogni silo, ma anche un'applicazione web di amministrazione per gestire l'accesso di terze parti tramite credenziali IAM temporanee e un frontend distribuito su Cloudfront per fornire una semplice interfaccia agli utenti per caricare i dati direttamente su S3.



Se vi state chiedendo come abbiamo gestito la parte SFTP, ecco la soluzione: abbiamo attivato il servizio AWS Transfer Family per SFTP con una Lambda personalizzata per autenticare gli utenti con le stesse credenziali IAM utilizzate per l'accesso Web app.

Pertanto, sviluppando un'applicazione Web relativamente semplice, siamo riusciti a creare un'interfaccia completamente serverless per i nostri bucket S3 in modo che gli i membri dell'azienda possano creare credenziali temporanee per consentire agli utenti esterni di accedere a una dropzone S3 dove saranno in grado di caricare nuovi file.

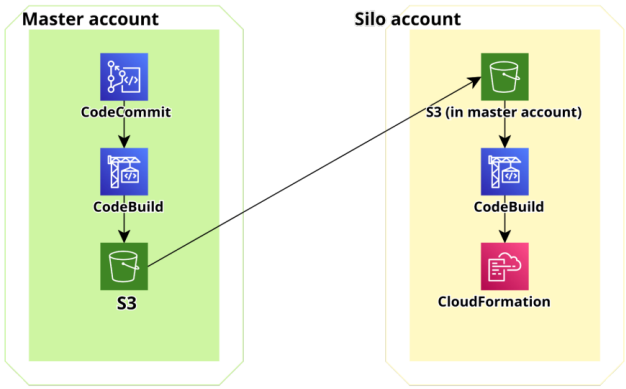
Prima di proseguire, soffermiamoci su come sia possibile far funzionare Lambda(s) Backend del diagramma sopra senza un database. La risposta è molto semplice: lo stato della nostra vending machine di credenziali è una raccolta di risorse AWS (utenti , Roles, Buckets) che creiamo direttamente con cloudformation e di cui conserviamo lo stato direttamente nel template!

Utilizzando pipeline tra account, tutti i componenti dell'infrastruttura e dell'applicazione possono essere distribuiti automaticamente su ciascun account in modalità self-service: quando viene creato un account nell'apposita AWS Organization Unit, viene creato un set di stack CloudFormation nell'account che distribuisce i componenti dell'infrastruttura di base e una codepipeline AWS per recuperare il codice dell'applicazione dall'account Master Datalake e distribuirlo nell'account Silo di destinazione.

Per rendere questo flusso scalabile a un numero arbitrario di account, la pipeline di distribuzione per la webapp è suddivisa in 2 parti: la prima parte è nell'account principale, utilizza il nostro CodeCommit Repo come sorgente, esegue test di unità e integrazione in AWS codebuild e infine crea un bundle dell'applicazione (utilizzando il pacchetto aws cloudformation). Infine comprime il modello CF e il bundle del codice e carica lo zip in un bucket S3 versionato accessibile da tutti gli account Silo figlio.

La seconda parte della pipeline si trova in ciascun account silo. È configurata per eseguire il polling del bucket per le modifiche. Quando rileva un nuovo file zip, viene avviata un'esecuzione e lo zip viene letto e distribuito direttamente utilizzando CloudFormation.

Lo schema del di pipe è mostrato di seguito:



Se anche voi lavorate molto su AWS, avrete notato che l'account Silo generico non dovrebbe essere in grado di accedere al bucket S3 in Master dove la prima parte della pipeline ha distribuito il bundle (ovviamente il bucket è privato): ciascuno di questi account è stato creato dalle organizzazioni, quindi i nuovi account si generano e muoiono e non vogliamo davvero elencarli nella policy del bucket S3 dell'account principale.

La soluzione a questo problema è piuttosto semplice: è possibile creare policy delle risorse in modo tale che solo una determinata Organization Unit (OU) possa accedere al bucket. Questo è un trucco piuttosto potente perché è applicabile anche ai Glue Catalogs e consente di condividere i dati direttamente con intere unità organizzative. Inoltre, è ancora possibile interrogarli con AWS Athena.

La policy del bucket:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action ": "s3:Get*",
            "Resource": [
                "arn:aws:s3:::bucket.code",
                "arn:aws:s3:::bucket.code/*"
            ],
            "Condition": {
                "ForAnyValue:ArnLike": {
                    "aws:PrincipalArn": "arn:aws:iam::*:role/*-codepipeli
```

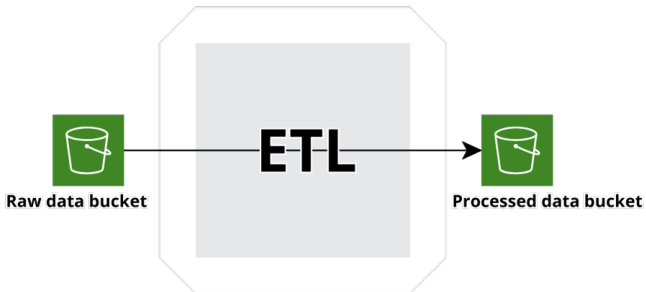
```
ne-rl"
                },
                "ForAnyValue:StringLike": { }
                    "aws:PrincipalOrgPaths": "o- ****/r-****/ou-****-****
**/ou-****-*******/*"

            }
        }
    ]
}
```

Dove "o -****/r-****/ou-****-******/ou-****-*******/*" è il percorso OU e arn :aws:s3:::bucket.codifica l'Arn del bucket.

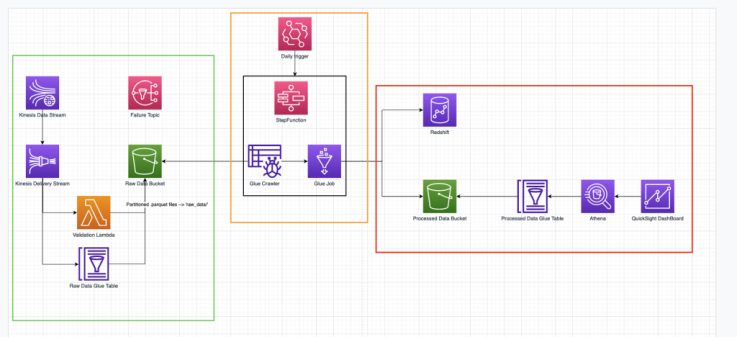
Ok: ora abbiamo tonnellate di account silo, ciascuno con la nostra Web app personalizzata che consente agli utenti di caricare nuovi file e, utilizzando criteri come quello sopra, l'account principale dell'azienda e altri account per l'analisi globale saranno in grado di accedere ai file su s3 ed eseguire query con Athena. Potremmo persino utilizzare Lakeformation cross-account per ridurre ulteriormente i permessi di accesso o aumentare la granularità (di questo parleremo presto in un articolo dedicato!)

Tuttavia, ciascun silo è separato e indipendente cosicché data scientist e data engineer che lavorano separatamente possano deployare transform jobs custom senza dove interagire con altri silos.



La tipologia di flusso sopra descritta è comune a tutti i Silos e in ogni silo account sono tipicamente presenti più flussi.

Inoltre, è possibile utilizzare anche altri servizi AWS per importare diversi tipi di dati in parallelo con l'app Web personalizzata che abbiamo sviluppato per esporre direttamente S3 ai produttori di dati. Qui presentiamo la struttura di uno dei tanti flussi implementati:



Un AWS API Gateway è stato esposto per ricevere dati da diversi sistemi.

API Gateway si integra direttamente con il servizio kinesis firehose per creare un buffer di dati configurabile per tempo e domensione, trasformare i dati nel in formato parquet e infine scriverli in S3. Il formato dei dati è definito ab initio ed esplicito nella struttura della glue table utilizzata anche da kinesis per trasformare i dati string json in parquet. I dati vengono regolarmente esportati nel bucket di output dei dati elaborati utilizzando una AWS Step Function basata su AWS Glue che prima esegue un crawler sul bucket dei dati grezzi per indicizzare i nuovi dati e quindi esegue un Glue Job per ripulire i dati e portarli nel formato concordato ; infine, i dati vengono scritti in un data warehouse di Redshift e nel bucket s3 di output dei dati elaborati. A questo punto, viene eseguito un nuovo crawler e i dati vengono letti da Athena e importati in quicksight per creare una nuova dashboard.

Conclusione

In questo articolo, abbiamo descritto come creare un data lake multiaccount dinamico su AWS e abbiamo condiviso alcune criticità interessanti che si possono incontrare.

Cosa ne pensate? Fateci sapere nei commenti!

About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

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!