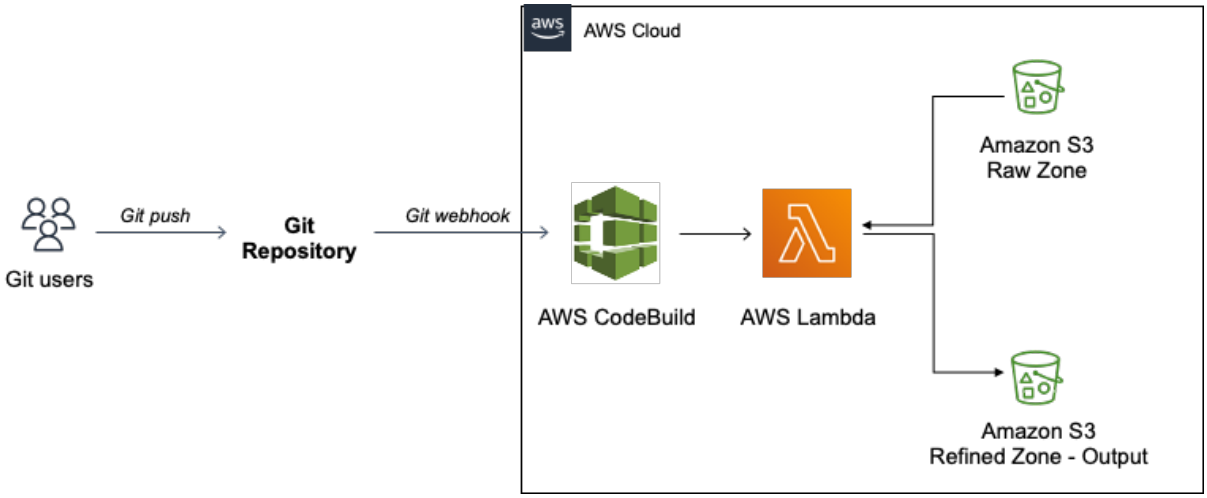
Preamble:

The goal of the exercise is to profile "hit level data" of one of Adobe Analytics' client's websites that tracks every visit/hit. The client is interested in finding the revenue generated by external Search Engines and the keywords that perform the best based on revenue.

Architecture:

Using Python for ETL, AWS S3/Lambda for serverless deployment and GitHub/AWS CodeBuild for CI/CD could be a good solution for this use case. Below is a high-level architecture of the same.



Approach:

- Profiled the source data and used AWS S3 as the raw landing zone of 'hit data'.
- Deployed a Python script (lambda function) on AWS Lambda that uses Pandas library to extract the source data from S3 (Raw Zone) as a data frame and perform aggregations to obtain the Search Engine domain, Search Keywords and Revenue generated by each Search Engine/Keywords combination.
- To ensure read/write access to/from AWS S3-Lambda, I worked on setting up new IAM roles and made necessary changes to the policies on both S3 and Lambda. Ensured optimal configuration of compute/time on Lambda for this use case. Also implemented Layers on Lambda to package required libraries.
- To ensure Continuous Integration/Continuous Delivery like implementation, I used GitHub with AWS CodeBuild. I built a YAML script that when run, installs all dependencies in the

requirements.txt file to package the required libraries. Build/Rebuilds of the lambda function are based on the buildspec.yml file.

- Enabled a webhook on CodeBuild to ensure that each time a PUSH is encountered on the GitHub repository, CodeBuild fetches all new changes and redeploys them on AWS Lambda.

- When the Lambda script is run, it fetches the source data from S3-Raw Zone and exports a new result to S3-Refined Zone which can act like a Data Lake. This file in the refined zone, answers the client's question of which search engine-keywords generate most revenue.

Improvements:

When the volume of data (>10GB) gets over a threshold the above approach may not be the best approach. Using AWS Lambda/Pandas is a serverless deployment approach that can be best used for smaller ETL operations and limited scalability. I would suggest using Spark (PySpark) and deploy the application on Amazon EMR.

Why Spark/Hive with EMR?

Spark lies on top of the Hadoop framework which is ideal for handling Big data. Spark ensures Massive Parallel Processing (MPP) of operations on large datasets. The underlying architecture ensures a distributed environment in which jobs are segregated into numerous clusters and run parallelly (typical MapReduce functionality). Once all clusters finish their jobs, the leader node aggregates the data and outputs it.
This unique architecture of Hadoop solves most performance issues with low latency, in the right use cases related to scale of volume, velocity and variety of data.