Adobe Applicant Assessment Exercise

Business Problem:

How much revenue is the client getting from external Search Engines, such as Google, Yahoo and MSN, and which keywords are performing the best based on revenue?

Development Requirements

1. Create a Python application that needs to be deployed and executed within AWS

2. The Python application needs to contain at least one class

3. The Python application needs to accept a single argument, which is the file that needs to be

processed.

Deliverable Requirements

The final output needs to be a tab delimited file with the following data points:

• Search Engine Domain (i.e., google.com)

• Search Keyword (i.e., "Laffy Taffy")

• Revenue (i.e., 12.95)

• A header row needs to be included. Use the above bulleted items for each column header,

minus the example.

• Sorted by revenue, descending, so the client can easily review which keyword is performing the

best.

• The output file should have the following naming convention:

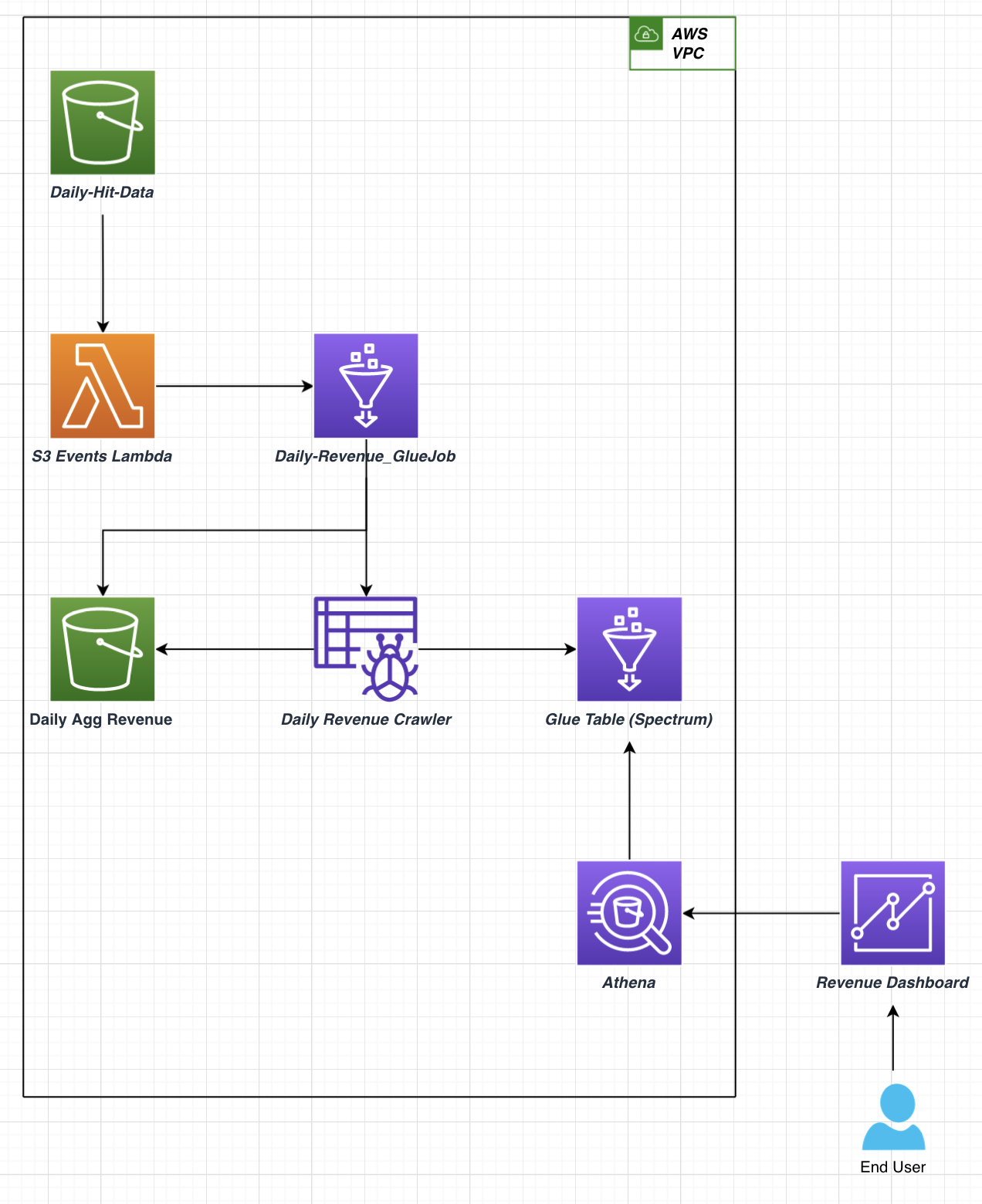
[Date]\_SearchKeywordPerformance.tab

Data Details:

1. The data file will be provided in TSV format daily.
2. The data has many columns like hit\_time\_gmt, date\_time, user\_agent, ip, geo\_city, geo\_country, geo\_region, pagename, page\_url, product\_list, referrer, event\_list
3. Product list delimited by semicolons and contains many other information field like Category, ProductName, No of Items, Revenue and Merchandizing eVar.
4. The other information which I concluded and hypothesized about the data is:
   1. Each day file will have a given day hit date data information.
   2. Daily one file will be dumped in the S3 bucket (no partitioning)

Design Aspects:

1. After initial understanding of the data and the problem, my thought is to design it as event based driven pipeline or application. We will process the data through AWS ETL Services and dump the transformed data in another S3 bucket.
2. Also, I was thinking of instead of providing the data to end customer through S3 files, we will create a QuickSight Dashboard which will contain the analysis of the Revenue data.
3. If needed, we can also provide an external table to the end users so that they can query historical data as per their business needs and analyze it.



ETL Workflow:

1. The ETL Workflow based on the above design approach in normal approach will be S3 event based ETL workflow.
2. As soon as we get the data file in the S3 bucket, the S3 event will trigger(invoke) the lambda function.
3. The Lambda function based on the S3 events configure the glue params and invoke/trigger a glue job.
4. The glue job based on lambda invoke will process the file based on the glue params and write the data to different S3 bucket.
5. Once the S3 file is written, the glue job will also trigger the associated glue crawler to create a Glue Table (External Table) and register the new file partition in the table.
6. This Glue table will be used in the Quick Sight dashboard for the revenue analysis.
7. End User/Customer will have only the visibility to the Analysis Dashboard rather than any of the internal ETL workflow.
8. Based on the User further request, we can expose the Glue table or S3 bucket files also to them for further analysis.

Lambda:

1. The Lambda is configured based on the S3 events.
2. When the file gets loaded, replaced, copied in the source s3 bucket, it will trigger the lambda function and lambda will process the s3 events and pass the corresponding parameters to the glue job so that the job can process it.
3. We are not processing and writing the data from the lambda and instead of want to do it from glue because we want to leverage Spark for scalability and performance. When the data becomes huge, lambda won’t be able to handle it.

Glue Job:

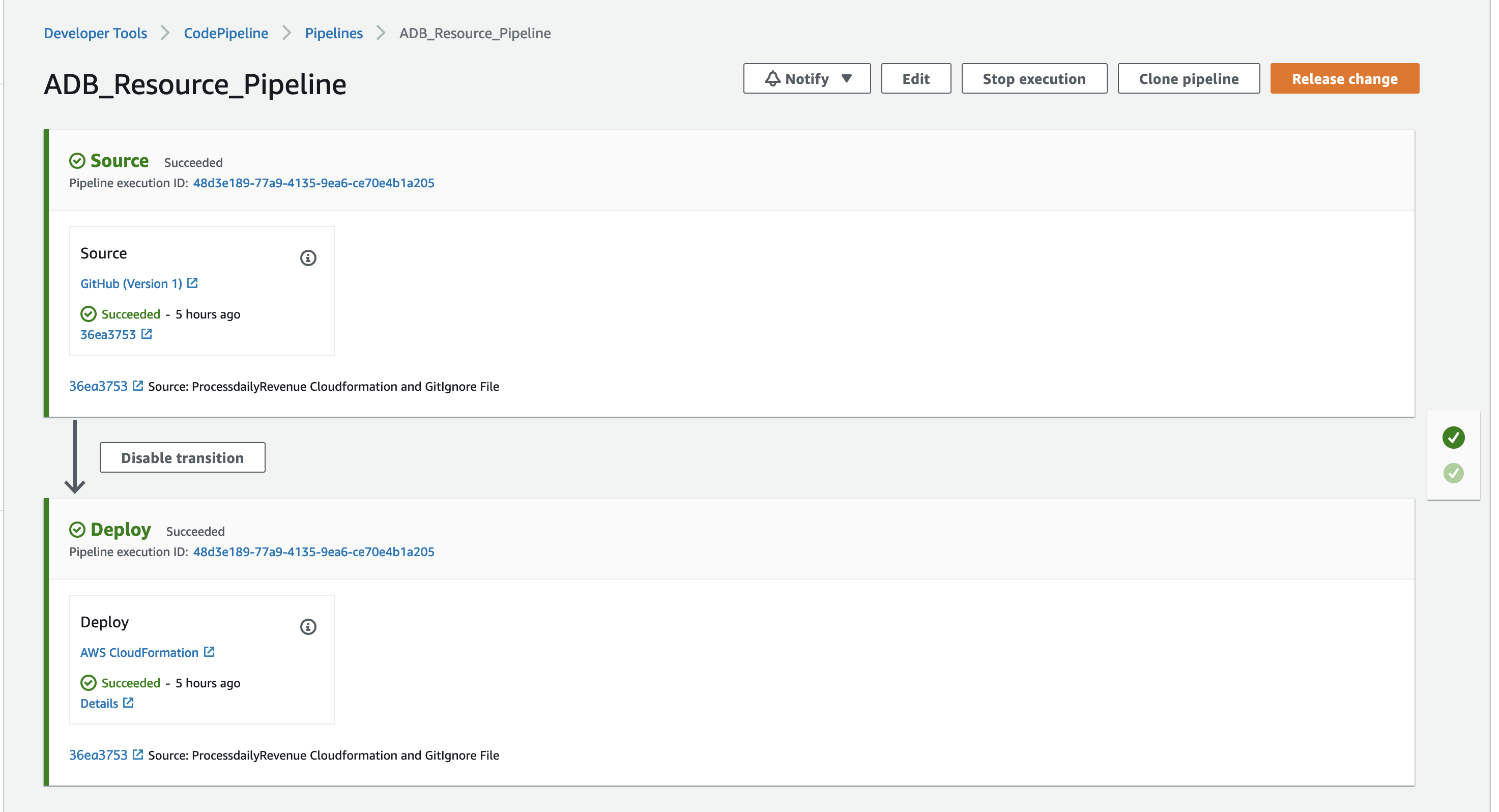
1. The Glue job gets triggered in two ways – one via Lambda and other via manually for backfill.
2. For scalability and performance, I have included the logic for backfill in the glue job which handles the scenario when we need to backfill the data due to some issue.
3. The glue job will process the data in below steps:
   1. Read the data from source s3 bucket.
   2. Parse the Product list data into different columns.
   3. Get the Host and Query from the referrer columns
   4. Clean the Revenue and the Query data.
   5. Aggregate the Revenue for the Host and Query.
4. For writing the dataframe, I am writing the dataframe into date partition rather than a single file.
5. Writing to a single would be fine now but later when the data becomes huge it won’t be feasible to write to a single file as drive memory.

Glue Crawler:

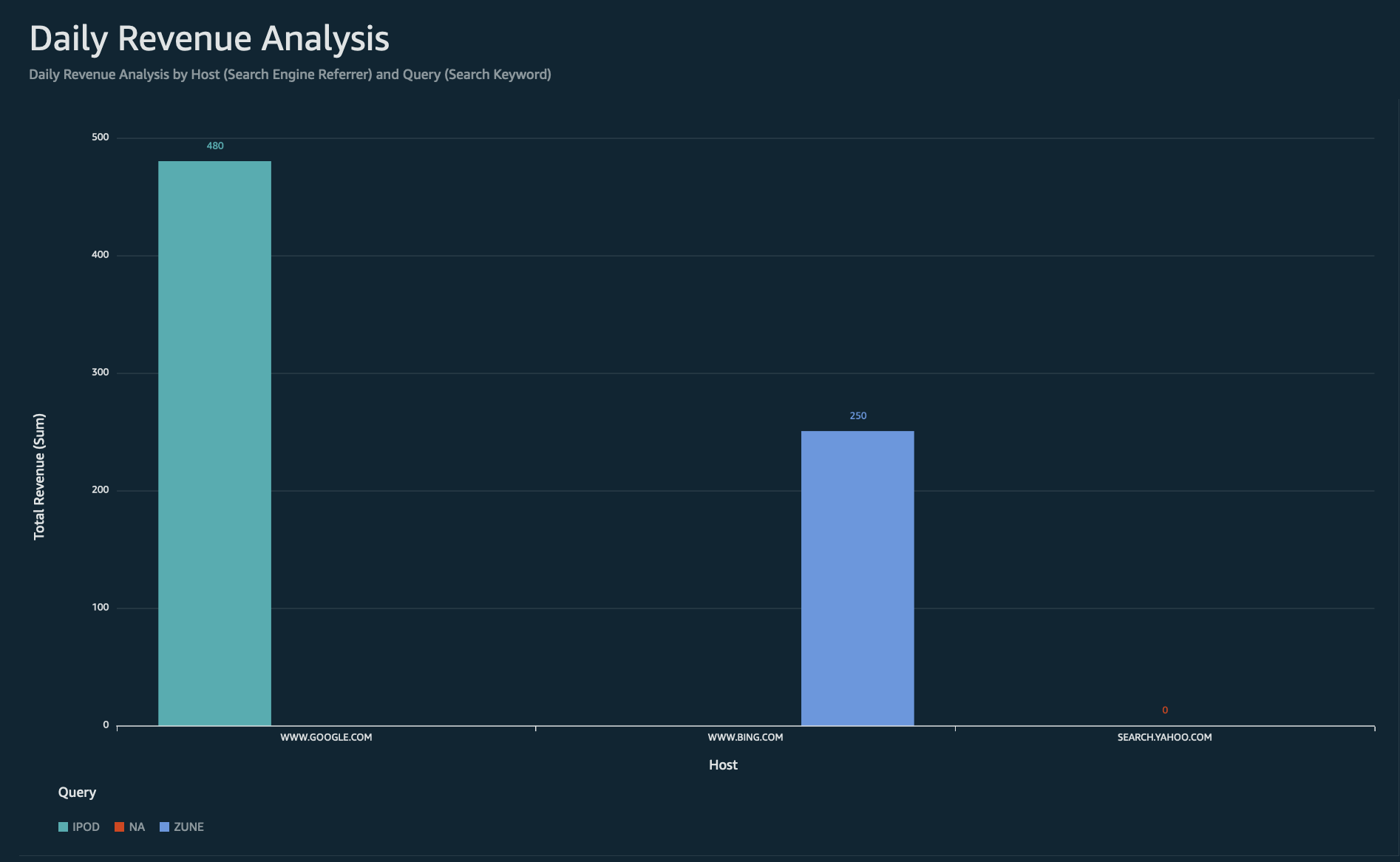
1. The glue crawler is used so that we can create the glue table which will be used in QuickSight Dashboard and also later can be provided to the user if they want to use later for historical analysis.
2. Glue crawlers are triggered at the end of glue job so that the particular partition gets registered in the Glue Table.

Deployment Scripts:

1. For deployment, I will be using CloudFormation template for creating and updating the stack.
2. Also, for Code Deploy I have created an initial Code pipeline which is based on GitHub Webhooks and gets triggered automatically when a commit occurs.
3. Below is the screenshot of the Code Pipeline, the pipeline uses separate IAM roles for one by CloudFormation for resource creation and updation and one by code pipeline for package deployment.



Quick Sight Dashboard:



1. Here the analysis based on 2021-01-04 file, suggest that we got 480$ revenue from Google for IPOD keyword search and 250$ from Bing for Zune keyword, we had some search from Yahoo also but didn’t get any revenue or items sold.
2. The QuickSight Dashboard uses SPICE query engine (free ~ 10GB) of data. (In-memory Caching). If our dataset increases by time, we can leverage the efficiency of Partition in Glue Table to scan relevant data only when filter applies but if we need all historical data, we can increase the limits.