Razer Data Engineer Technical Assignment: Part 2

Overview

This project aims to create a data processing pipeline that utilizes the docker image created from before. The pipeline will leverage AWS services such as S3, SQS, Lambda, Fargate, Kinesis and Redshift.

Architecture

Batch Ingestion Pipeline

This pipeline deals with the larger chunks of data.

1. Data Ingestion:
    - Batch files can be uploaded to a dedicated S3 bucket.
2. Triggering of batch processing:
    - We can set up an S3 PUT event trigger with a lambda function.
    - Upon detecting an event trigger, lambda will send a message to the SQS queue which in turn can be used to start a Fargate task.
3. Data processing in Fargate:
    - We can deploy the container created previously to Fargate.
    - It handles messages from the SQS queue and takes care of the processing.
4. Data loading:
    - Processed data from the Fargate task will be loaded wot Redshift.

Streaming Ingestion Pipeline

This pipeline deals real time streaming data.

1. Data Ingestion:
    - Streaming data from producers are ingested using Amazon Kinesis Data Streams
2. Triggering of stream processing:
    - Lambda takes care of processing the streaming data coming from KDS, performing any required transformations.
    - It then sends messages to the SQS queue.
3. Data processing in Fargate:
    - Similarly, Fargate will handle the messages from SQS and does the processing.
4. Data loading:
    - Processed data from the Fargate task will be loaded wot Redshift.

Failures and retries

With the given setup, the idea is to utilize a dead letter queue alongside the main SQS queue that receives messages. For messages that fail to get processed by Fargate and get sent to the DLQ, we can inspect them and find out if any bugs exist, and to retry them we could use the DLQ redrive to resend messages back to the original queue for processing.

In case it is needed, KDS is able to retain data (up to 365 days) which allows for reprocessing of data if needed.

Additionally, CloudWatch can be used to monitor for failures, and in this case we could possibly automate the DLQ Redrive process by monitoring for messages in the DLQ. If any are detected, we could have a CloudWatch alarm send an SNS to a lambda that initiates the redrive.