Problem 02:

Additional questions

Here are a few possible scenarios where the system requirements change or the new functionality is required:

1. The batch updates have started to become very large, but the requirements for their processing time are strict.
2. Code updates need to be pushed out frequently. This needs to be done without the risk of stopping a data update already being processed, nor a data response being lost.
3. For development and staging purposes, you need to start up a number of scaled-down versions of the system.

Please address at least one of the situations. Please describe:

* Which parts of the system are bottlenecks or problems that might make it incompatible with the new requirements?
* How would you restructure and scale the system to address those?

Problems with the previous solution:

It’s difficult to cater to the above requirement with previous spark solution. So, the following solution with **AWS glue** will improve the usability, high availability, and scalability. AWS Glue leverages Spark's capabilities to provide an optimized experience for ETL. So, we can migrate Spark programs to AWS Glue to take advantage of glue features.

How to cater to the following requirements

1. Code updates need to be pushed out frequently. This needs to be done without the risk of stopping a data update already being processed, nor a data response being lost.

* This can edit the code of a Glue job while the job is performing its JobRun but it'll not affect the execution of already running instance of the job. So, it'll apply new change for upcoming instance and its jobs. But this was not feasible with the previous solution with spark.

1. For development and staging purposes, you need to start up a number of scaled-down versions of the system.
   * This can be easily archive because AWS Glue mange the ETL workflow and it’s serverless. So, it’s only cost for the utilize resources. Creating multiple scaled down environments for different purposes is totally acceptable due to minimum cost and effort.

Diagram

Description automatically generated

The previously created Kubernetes back spark cluster setup can be replaced by AWS Glues setup because glue can easily execute all the sparks jobs without any changes which ran on the previous system. Using serverless with aws is cost effective rather than using on premise setup with mange infrastructure by our own.

AWS Glue calls API operations to transform data, create runtime logs, store job logic, and create notifications to help monitor job runs. The AWS Glue console connects these services into a managed application, so we can focus on creating and monitoring ETL work. The console performs administrative and job development operations on behalf of us. We should supply credentials and other properties to AWS Glue to access data sources and write to data targets. AWS Glue takes care of provisioning and managing the resources that are required to run workload. We don't need to create the infrastructure for an ETL tool because AWS Glue does it for us.