Data Pipeline DevOps considerations

Metrics and Monitoring for the running dataflows

Since the proposal stack is designed using 100% AWS components, it gets rid of it embedded CloudWatch integration. Thus, metrics are fetched since initial set ups, and it will make easy to build dashboards and alerts.

Recovering from a failed dataflow

As described a bit on architectural proposal, the ingestion layer have a retry mechanism. In case of failure an alert will be triggered and the data from requests will be stored on a bucket path designed for store error data.

For reprocess error, the reprocess layer might make use of existing raw data acquired on ingestion/collect phase.

The Redshift has a mechanism to direct load S3 files in some formats (parquet, Avro, etc.) and load then it into its tables. So, a reprocess on Redshift can happen easily.

Testing on multiple environments

The whole stack will be made with infra as code concept using AWS CloudFormation since the architecture must be made with AWS components only. Otherwise, I would personally consider Terraform.

Once there is a stack described as code, it will have variables for different environment. Then two (or more) environment could be created easily for testing and QA purposes.

Modifying Code without stopping it.

For keep stack up and running during code insertion of new features, updates or fixes AWS provides some services for this purpose. AWS CodePipeline and its satellite components (CodeDeploy, CodeCommit) will be in charge of continuous delivery and continuous integration of this solution.