

Suppose the architecture for your event-driven ETL pipeline is:

Publisher -> Kinesis Data Streams -> Kinesis Data Firehose -> S3 -> Lambda or EventBridge rule looking for arrival of files in the main data folder -> subsequent steps.

The issue with this kind of event driven processing is, suppose you continuously get records in batches of 20 records into the S3 landing area for over 12-13 minutes and based on your transformer lambda / firehose settings, the first records start coming into the S3 bucket in the 4th or 5th minute, your workflow will start immediately and there is a chance that the records that came in the 6th or 7th minute etc. do not even get picked for processing. So, your landing area and staging area counts could match but you might have dropped records in the landing layer itself.

These records could get picked up when the next set of files arrive, but this could be a big/unacceptable lag for production workloads.

Couple of ways to handle this:

1. Instead of your EventBridge source rule just looking for "Object Create" in the data folder of the landing bucket, you need to look for a specific "completion" file that is sent at the end of every batch by the source system. This completion file could be sent in a separate folder other than the data folder within the same S3 bucket, as a best practice. i.e. If the data records are sent into the "orders" folder, the batch completion files could be sent into another folder "batch-completion" under the same bucket.

So your Glue work flow will get triggered only at the end of every batch and you will not miss any records.

{

"source": ["aws.s3"],

"detail-type": ["Object Created"],

"detail": {

"bucket": {

"name": ["fo-to-dw-orders-landing-area"]

},

"object": {

"key": [{

"prefix": "batch-completion/"

}]

}

}

}

Here as well, suppose you get a "completion" file at 7:32 and another "completion" file at 7:35, your glue crawler or job for 7:32 file might still be running and the crawler / job for 7:35 might not get triggered at all and will fail because of the previous run in progress. So this approach is fine if you and the source system are sure that the batches will be spread out enough .. better the interval between 1 batch and the next batch (i.e. the interval between 1 "completion" file and the next "completion" file) is atleast the time interval for 1 complete glue workflow execution run (~15 minutes in my case).

Also, when there are multiple source systems sending data, maintaining the interval will be a challenge.

2. Create an eventbridge rule that is dependent not only on S3 completion file object create event, but also on the Glue workflow succeeded (OR condition), so that, there is continuous execution of the workflow. If there is data at the source, it will get picked up. Else, no data will get picked up. In this case, the S3 completion file object create condition is useful only for the first execution or in the case where Glue workflow is in failed status for some reason.

You must understand that this kind of a continuous loop is not a good solution unless you are expecting records throughout the day and there are strict SLA's to be met.

This is because, the resources are being continuously used whether there is data to be processed or not.

In a production scenario, if you go with this option, you must have an adhoc or a scheduled job that disables / enables the rule based on the schedule requirements, so that the workflow can be stopped when needed.

{

"$or": [{

"detail-type": ["Glue Job State Change"],

"source": ["aws.glue"],

"detail": {

"jobName": ["orders\_staging\_audit\_load"],

"state": ["SUCCEEDED"]

}

}, {

"source": ["aws.s3"],

"detail-type": ["Object Created"],

"detail": {

"bucket": {

"name": ["fo-to-dw-orders-landing-area"]

},

"object": {

"key": [{

"prefix": "batch-completion/"

}]

}

}

}]

}

Note: orders\_staging\_audit\_load is the last job in the Glue workflow.