Troubleshooting Continuous Data Ingestion using Snowpipe in Snowflake for Amazon S3

Snowpipe auto-ingest process flow for Amazon S3

In this post we will discuss some DevOps features how to troubleshoot in case you have any issues with Snowpipe implementations w/ AWS S3 as a staging.
- **Steps** how to integrate AWS S3 as external storage location with Snowflake.
- **Steps** how to set up Snowpipe for continuous data ingestion in Snowflake.
The steps to troubleshoot issues with Snowpipe that differs depending on the data files to be loaded and the approach. Trying to explain a *five thousand feet* top view of the process. Let's talk about a methodical approach, miscellaneous facts about the continuous ingestion process, how to troubleshoot such as —
- Checking the pipe status
- Viewing the copy history for a table
- Validating the data file after the loading process
- Files present in the stage but not loaded
- Files were missed or set of files not loaded
- View the records using copy history definition
- Data dups in the target tables
- Unable to reload modified data, modified data loaded unintentionally

1: Describe Pipe — Returns the result only for the pipe owner. And describes the user defined properties including the default properties when created.

```
Use Role my_rl;
Use Warehouse my_wh;
Use DataBase my_db;
Use Schema my_sch;

desc pipe pipEmpADV; -> Describes the properties specified for a pipe and default values.
//Command provides pipe properties & metadata in the following. Highlighted few definitions

created_on - Date and time when the pipe was created.
name -  Name of the pipe.
database_name - Database in which the pipe is stored.
schema_name -  Schema in which the pipe is stored.
definition - COPY statement used to load data from queued files into a Snowflake table.
owner - Name of the role that owns the pipe
notification_channel - Amazon Resource Name of the Amazon SQS queue for the stage in the DEFINITION.
```

2: Show Pipes — Listing the pipes for which you have access privileges. Used to list for a current database or schema for the session or in entire account.

```
Use Role my_rl;
Use Warehouse my_wh;
Use DataBase my_db;
Use Schema my_sch;

show pipes; -> Show all the pipes that you have privileges to view
//Command provides pipe properties & metadata in the following. Highlighted few definitions

created_on - Date and time when the pipe was created.
name -  Name of the pipe.
database_name - Database in which the pipe is stored.
schema_name -  Schema in which the pipe is stored.
definition - COPY statement used to load data from queued files into a Snowflake table.
owner - Name of the role that owns the pipe
notification_channel - Amazon Resource Name of the Amazon SQS queue for the stage in the DEFINITION.
```

3: Alter Pipe — Helps modifying limited set of properties for an existing pipe object. Supports the following operations such as:
- *Pausing* or *resuming* the pipe.
- *Refreshing* a pipe i.e. copying staged data files for loading into the target.
- *Adding/overwriting/removing* a comment for a pipe.

```
Use Role my_rl;
Use Warehouse my_wh;
Use DataBase my_db;
Use Schema my_sch;

//Alter PIPE command modifies a limited set of properties for an existing pipe object

ALTER PIPE pipEmpADV SET PIPE_EXECUTION_PAUSED = [TRUE | FALSE]
        TRUE status pauses the pipe.
        FALSE status resumes the pipe
The executionState reported by SYSTEM$PIPE_STATUS is PAUSED/ RUNNING

Refreshing a PIPE:
Alter PIPE pipEmpADV Refresh;

Only load the files staged after a specified timestamp:
Alter PIPE pipEmpADV Refresh modified_after='2018-07-30T13:56:46-07:00';
```

4: Check Pipe Status — Helps determine the current status of the pipe by using a system-defined function. **Link** online for more details.

```
Use Role my_rl;
Use Warehouse my_wh;
Use DataBase my_db;
Use Schema my_sch;

//Retrieves a JSON name/value pairs representation of the current status of a pipe.

Select system$pipe_status('my_db.my_sch.pipEmpADV');
+-------------------------------------------------+
| SYSTEM$PIPE_STATUS('MY_DB.MY_SCH.PIPEMPADV')    |
|-------------------------------------------------|
| {"executionState":"RUNNING","pendingFileCount":0} |
+-------------------------------------------------+

Where:executionState
Current execution state of the pipe, could be: RUNNING/ PAUSED/ STOPPED_STAGE_DROPPED..

pendingFileCount:Number of files queued for loading by the pipe.

AUTO_INGEST = TRUE: Files added to the cloud storage triggers an event notifications for the pipe.

oldestFileTimestamp: Earliest timestamp among data files currently queued
```

5: View Copy History — Used to query Snowflake data loading history within the last 14 days by using a system-defined function. Retrieves the activity for both copy commands and Snowpipe data loading.

```
Use Role my_rl;
Use Warehouse my_wh;
Use DataBase my_db;
Use Schema my_sch;

//Retrieve details about all loading activity in the last hour:
Select * From Table(information_schema.copy_history(table_name=>'stgEMP_RAW',
        start_time=> dateadd(hours, -1, current_timestamp())));

Returns the following columns: FILE_NAME - Name of the source file.

STAGE_LOCATION - Name of the stage where the source file is located.

LAST_LOAD_TIME - Date and time of the load record.

ROW_COUNT - Number of rows loaded from the source file.

ROW_PARSED - Number of rows parsed from the source file.

STATUS - Status: Load in progress, Loaded, Load failed, Partially loaded, or Load skipped.

PIPE_NAME - Name of the pipe defining the load parameters.

And many more....
```

6: View Load History — Displays the history of the data loaded into tables by the ***Copy*** command within the last 14 days. Displays one row for each file loaded. This view ***doesn't*** return the history of data loaded using ***Snowpipe***.

```
Use Role my_rl;
Use Warehouse my_wh;
Use DataBase my_db;
Use Schema my_sch;

//Fetch the history of data loaded into table since Oct-20-2021, assuming within previous 14 days:
Select * From information_schema.load_history
  Where schema_name=current_schema() And
  table_name='stgEMP_RAW' And
  last_load_time > 'Wed, 20 Oct 2021 12:00:00 -0800';

//Fetch 10 most recent COPY INTO commands executed:
Select * From information_schema.load_history
  Order By last_load_time Desc
  Limit 10;

View returns similar columns as discussed in COPY_HISTORY section.
```

7: Validate Data Files: If the load operation show errors in the data files, the copy history function describes the first error that occurred in each file. This table function can be used to validate the data files processed by Snowpipe within a specified time range. Returns the below details about any errors that occurred.

```
Use Role my_rl;
Use Warehouse my_wh;
Use DataBase my_db;
Use Schema my_sch;

//Validate any loads for the mypipe pipe within the previous hour:
select * from table(validate_pipe_load(
  pipe_name=>'my_db.my_sch.pipEmpADV',
  start_time=>dateadd(hour, -1, current_timestamp())));

The function returns the following columns and also returns pipe activity within the last 14 days:
ERROR - First error in the source file.

FILE - Name of the source file where the error was encountered.

LINE - Number of the line in the source file where the error was encountered.

CHARACTER - Position of the character where the error was encountered.

CATEGORY - Category of the operation when the error was produced.

CODE - ID for the error message displayed in the ERROR column.
```

8: Snowpipe Usage— Used to query the history of data loaded in Snowflake. Returns loaded data history and credit billed w/ Snowflake account only for the account admin role or any role with monitor usage global privilege.

```
Use Role my_rl;
Use Warehouse my_wh;
Use DataBase my_db;
Use Schema my_sch;

//This function returns pipe activity within the last 14 days.
//Retrieve the data load history for a 30 minute range, in your account:
Select * From table(information_schema.pipe_usage_history(
    date_range_start=>to_timestamp_tz('2017-10-24 12:00:00.000 -0700'),
    date_range_end=>to_timestamp_tz('2017-10-24 12:30:00.000 -0700')));

//Retrieve the data load history for the last 12 hours, in your account:
Select * From table(information_schema.pipe_usage_history(
    date_range_start=>dateadd('hour',-12,current_timestamp()),
    pipe_name=>'my_db.my_sch.pipEmpADV'));

//Retrieve the data load history for the last 14 days for a specified pipe in your account:
Select * From table(information_schema.pipe_usage_history(
    date_range_start=>dateadd('day',-14,current_date()),
    date_range_end=>current_date(),
    pipe_name=>'my_db.my_sch.pipEmpADV'));
```

9: Result Scan Function — A post-process output command. Returns below the result set of a previous command within 24 hours of when you executed the query. Applicable for the current sessions or any of your other sessions, including past sessions, as long as the period of 24 hours has not elapsed. Takes *query_id* or last_query_id() as arguments.

```
//Retrieve values greater than 1 from the result of your most recent query in the current session:
select $1 as value from values (1), (2);
+-------+
| VALUE |
|-------|
|     1 |
|     2 |
+-------+

select * from table(result_scan(last_query_id())) where value > 1;
+-------+
| VALUE |
|-------|
|     2 |
+-------+

//Retrieve all values from your second most recent query in the current session:
select * from table(result_scan(last_query_id(-2)));

//Retrieve the values from the c2 column in the result of the specified query:
select c2 from table(result_scan('bd6687a4-351b-4a57-a060-02b2b0f0c21c'));
```

10: Load Historical Files — Here is the option to load any backlog of data files that existed in the external stage before SQS notifications were configured. Alter pipe…refresh statement copies a set of data files staged within the previous 7 days to the Snowpipe ingest queue for loading into the target table. If you want to load data from files staged earlier, Snowflake recommends the following steps:

```
1. Load the historic data into target table by copy into <table_name> statement.

2. Configure automatic data loads using Snowpipe with event notifications. Files
that are newly staged will trigger event notifications for ingestion. Because the
historic data files don't trigger event notifications, they are not loaded twice.

3. Execute an ALTER PIPE … REFRESH statement to queue any files staged in-between
Steps 1 and 2. The statement checks the load history for both the target table and
the pipe to ensure the same files are not loaded twice.
```

11: Remove Staged Files— Pipe objects don't support the purge copy option and can't delete

staged files automatically when the data is successfully loaded into tables. To remove the staged files that no longer are needed, Snowflake recommends executing the Remove command to delete the files periodically. Or configure any lifecycle management rules by your cloud storage service provider alternatively.

```
Removes files from either an external cloud storage or internal Snowflake stage.
For internal stages, the following stage types are supported:
- Named internal stage
- Stage for a specified table
- Stage for the current user

Remove all files from the path1/sub_path2 path in a named internal stage named stgEmp:
- remove @stgEmp/path1/sub_path2;

Remove all files from the stage for the orders table:
- remove @%orders;

Remove files whose names match the pattern *jun* from the stage for the current user:
- rm @~ pattern='.*jun.*';
```

12:Troubleshooting Other Issues — Below situations are again classified in categories such as the set of files not loaded into Snowflake. Please refer below.

- Missing copy history record: Check if the copy into SQL statement in the pipe includes any pattern or regular expression and that prevents the files to load. To resolve, re-create the pipe using *Create or Replace syntax*.
- Copy history record shows an unloaded subset of files: Check if the copy history result shows a subset of files not loaded, then use A*lter Refresh* command.
- Duplicate data in the target: If there are multiple pipes pointing to the same cloud storage location in the copy command, must verify the same directory paths shouldn't overlap. Otherwise, multiple pipes could load the same set of data files into the target tables.

13: Snowpipe auto-ingest Vs Bulk data load —

	Bulk data load	Snowpipe
Load History:	Stored in the metadata of the target table for 64 days. And the same will be available upon the completion of COPY Statement	Stored in the metadata of the pipe for 14 days. And must be requested from Snowflake via REST Endpoint, SQL Table Function, or the ACCOUNT_USAGE View.
Transactions:	Loads are always performed in a single transaction. Data is inserted into table alongside any other SQL statements ran manually by users.	Loads are combined/ splited into a single/ multiple transactions based on the number and size of the rows in each data files present within it.
Compute Resources:	Requires a user specified warehouse to execute COPY statements.	Uses Snowflake Supplied compute resources.
Cost:	Billed for the amount of time each virtual warehouse is active.	Billed according to the compute resources used in the Snowpipe warehouse while loading the files.