

Experiment: Data Engineering Pipeline

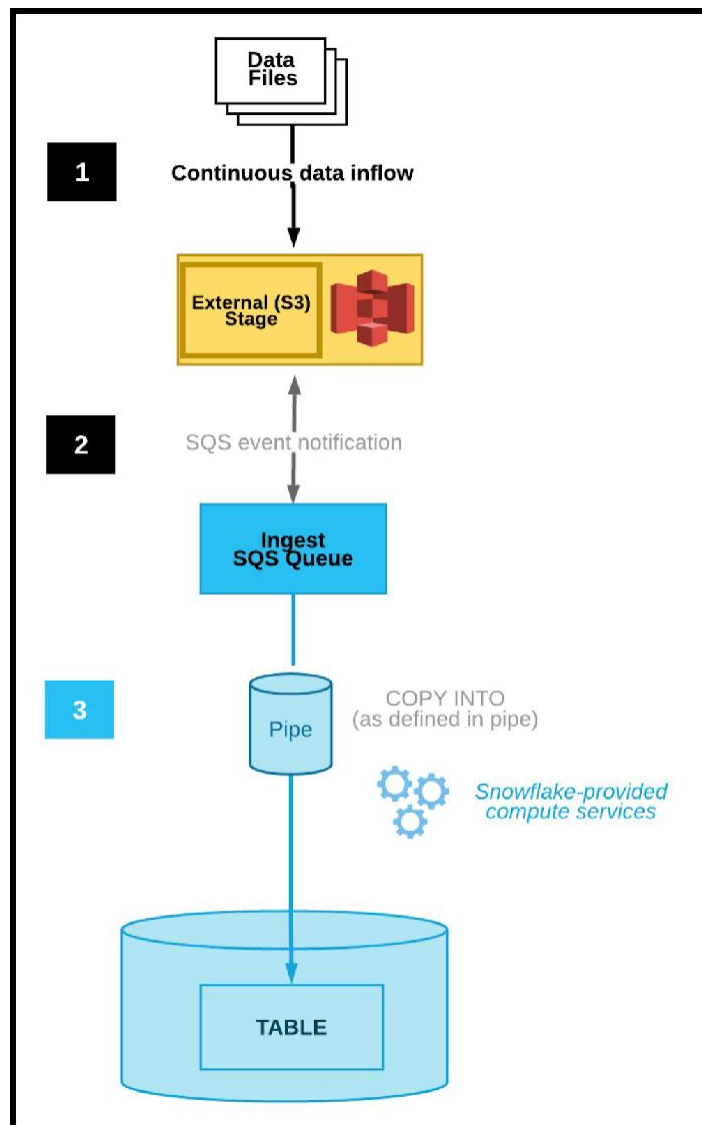
Overview

This experiment demonstrates Snowflake features specifically aligned to “Data Engineering” workload to build modern data pipelines. Data pipelines automate many of the manual steps involved in transforming and optimizing continuous data loads. Frequently, the “raw” data is first loaded temporarily into a staging table used for interim storage and then transformed using a series of SQL statements before it is inserted into the destination reporting tables. The most efficient workflow for this process involves transforming only data that is new or modified.

Experiment 22: Snowpipe

22.1 Leverage Snowpipe to Populate Table using S3 Event Notifications

This section describes the most common option for triggering Snowpipe data loads automatically using Amazon SQS (Simple Queue Service) notifications for an S3 bucket. The following diagram shows the Snowpipe auto-ingest process flow:



1. Data files are loaded in a stage.
2. An S3 event notification informs Snowpipe via an SQS queue that files are ready to load. Snowpipe copies the files into a queue.
3. A Snowflake-provided virtual warehouse loads data from the queued files into the target table based on parameters defined in the specified pipe.

22.1.1 Create a table that Snowpipe will use to write the incoming data. The data being loaded is JSON formatted so create a table using the VARIANT data type.

```
create or replace table trips_raw (v variant);
```

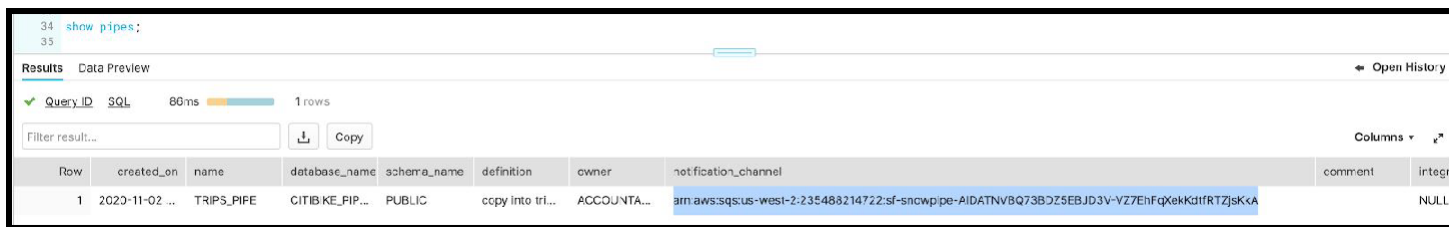
22.1.2 Create a PIPE definition to tell Snowpipe to load the JSON records in the STAGE into the table you just created.

```
create or replace pipe trips_pipe auto_ingest=true as copy into
trips_raw from @streaming_data/;
```

22.1.3 Confirm PIPE definition to tell Snowpipe to load the JSON records in the STAGE into the table you just created.

```
show pipes;
```

Make a note of the ARN of the SQS queue for the stage in the *notification_channel* column. Copy the ARN to a convenient location.

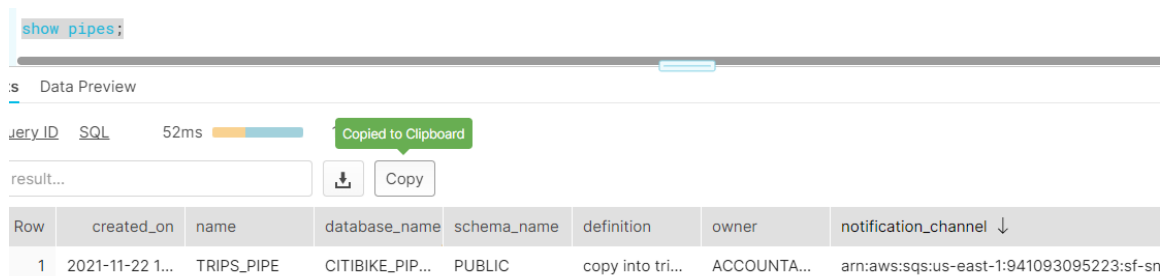


Row	created_on	name	database_name	schema_name	definition	owner	notification_channel	comment	integrations
1	2020-11-02 ...	TRIPS_PIPE	CITIBIKE_PIP...	PUBLIC	copy into tri...	ACCOUNTA...	arn:aws:sqs:us-west-2:235488214722:sf-snowpipe-AIDATNBQ73BDZ5EBJD3V-YZ7EhFqXekKd1fRTZjsKkA		NULL

22.1.4 Configuring Event Notifications. Complete the below configuration in your S3 bucket to automatically invoke Snowpipe when files are written there. This is the mechanism that is used to automatically load the streaming data files. Snowpipe relies on notifications from your S3 bucket to achieve this. Instructions for completing this configuration are found in the Snowflake documentation [link](https://docs.snowflake.com/en/user-guide/data-load-snowpipe-auto-s3.html#step-4-configure-event-notifications) below. **Perform only Step 4- Configure Event Notifications and come back to this document.**

<https://docs.snowflake.com/en/user-guide/data-load-snowpipe-auto-s3.html#step-4-configure-event-notifications>

From the Snowflake console when we execute “show pipes” it will provide the SQS queue that is the notification channel.



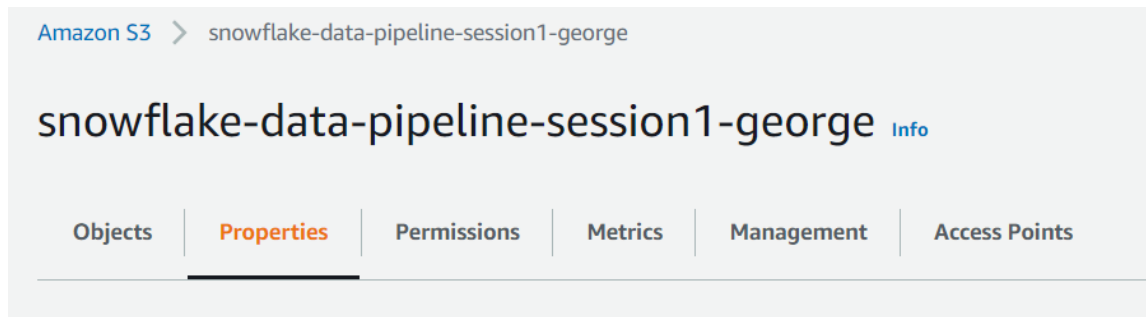
Row	created_on	name	database_name	schema_name	definition	owner	notification_channel
1	2021-11-22 1...	TRIPS_PIPE	CITIBIKE_PIP...	PUBLIC	copy into tri...	ACCOUNTA...	arn:aws:sqs:us-east-1:941093095223:sf-sn

That SQS ARN is available by clicking the Copy button on the results pane and pasting that into a text file or other editor.

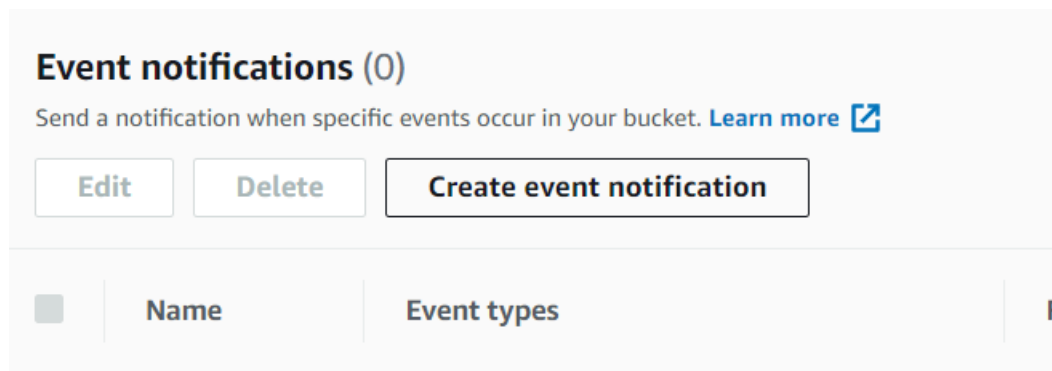
arn:aws:sqs:us-east-1:941093095223:sf-snowpipe-AIDA5WHMFA43YOTSXNQ7J--ZFTKNPCWky2PyrpnYtzsg

Sample screenshots of the S3 bucket enabling & configuring of event notifications below

View the S3 Console and choose the bucket that we created in our previous work



Choose the **Properties** tab and Scroll down to the **Event Notifications** section



No event notifications

Choose **Create event notification**

Create event notification [Info](#)

To enable notifications, you must first add a notification configuration that identifies the events to publish and the destinations where you want Amazon S3 to send the notifications.

General configuration

Event name

Event name can contain up to 255 characters.

Prefix - *optional*

Limit the notifications to objects with key starting with specified characters.

Suffix - *optional*

Limit the notifications to objects with key ending with specified characters.

Enter **Auto-ingest Snowflake** for the Event name

Event types

Specify at least one event for which you want to receive notifications. For each group, you can choose an event type or you can choose one or more individual events.

Object creation

☒ **All object create events**
s3:ObjectCreated:*

☐ **Put**
s3:ObjectCreated:Put

☐ **Post**
s3:ObjectCreated:Post

☐ **Copy**
s3:ObjectCreated:Copy

☐ **Multipart upload completed**
s3:ObjectCreated:CompleteMultipartUpload

22.2.1 Under Event Types and Object creation check the box for **All object create events**

22.2.2 Under Destination choose radio for SQS Queue and SQS Queue ARN and past the arn for the SQS queue that we retrieved with the “show pipes” result.

Destination

Destination

Choose a destination to publish the event. [Learn more](#)

- ☐ Lambda function
Run a Lambda function script based on S3 events.
- ☐ SNS topic
Send notifications to email, SMS, or an HTTP endpoint.
- ☒ SQS queue
Send notifications to an SQS queue to be read by a server.

Specify SQS queue

- ☐ Choose from your SQS queues
- ☒ Enter SQS queue ARN

SQS queue

arn:aws:sqs:us-east-1:941093095223:sf-snowpipe-AIDA5WHMFA43YOTSXNQ7J--ZFTI

Cancel

Save changes

22.2.3 Choose **Save changes**

✔ Successfully created event notification "Auto-ingest Snowflake".
Operation successfully completed.

Event notifications (1)

Send a notification when specific events occur in your bucket. [Learn more](#)

Edit

Delete

<input type="checkbox"/>	Name	Event types	Filters	Destination
<input type="checkbox"/>	Auto-ingest Snowflake	All object create events	-	SQS queue

22.2.4 Under Event Types and Object creation check the box for **All object create events**

New Worksheet

Run All Queries Saved 1 minute ago

```
47 -- 2.1.5 Call the Stored procedure that generates data (1 row)
48 call stream_data('2010-01-01', '2019-01-01');
49
50 -- 2.1.6 Verify if there are any files already in the bucket/folder
51 list @streaming_data;
52
53 -- 2.1.6 Check the status of the pipe.
54 select system$pipe_status('trips_pipe');
55
56 -- 2.1.6 Check if the data is loaded to the trips_raw table.
57 select count(*) from trips_raw;
58
59 -- 2.1.6 Review sample data from trips_raw table.
60 select * from trips_raw limit 100;
```

Results Data Preview

Query ID: SQL: 8 Rows

Copy

Row	name	size	md5	last modified
1	s3://sf-ctblike-pipeline/streaming-data/2010-01-01_0...	200B	c6459e0bb-c4eb3f50c7c548f1e11e	Sun, 8 Nov 2020 23:10:04 GMT
2	s3://sf-ctblike-pipeline/streaming-data/2018-01-01_0...	40587	f95961c3a4b577a442ca7754a69a651	Sun, 8 Nov 2020 23:10:04 GMT
3	s3://sf-ctblike-pipeline/streaming-data/2018-01-01_0...	81615	4bc7a9348a3688f6c7757bc2577b903	Sun, 8 Nov 2020 23:10:04 GMT
4	s3://sf-ctblike-pipeline/streaming-data/2018-01-01_0...	37612	1e606318bc6f5a2e66a3e322343aa12	Sun, 8 Nov 2020 23:10:04 GMT
5	s3://sf-ctblike-pipeline/streaming-data/2018-01-01_0...	36533	d2b2734b350dc023d3d0c070e5d0d7	Sun, 8 Nov 2020 23:10:04 GMT
6	s3://sf-ctblike-pipeline/streaming-data/2018-01-01_0...	37323	d6cc28d6c204c5f56c1ed4b83a00e5	Sun, 8 Nov 2020 23:10:04 GMT
7	s3://sf-ctblike-pipeline/streaming-data/2018-01-01_0...	40242	9bc24c42c33a0944c376c16dad575	Sun, 8 Nov 2020 23:10:04 GMT
8	s3://sf-ctblike-pipeline/streaming-data/2018-01-01_0...	40244	170a498170327708a1c12541a786f82a	Sun, 8 Nov 2020 23:10:04 GMT

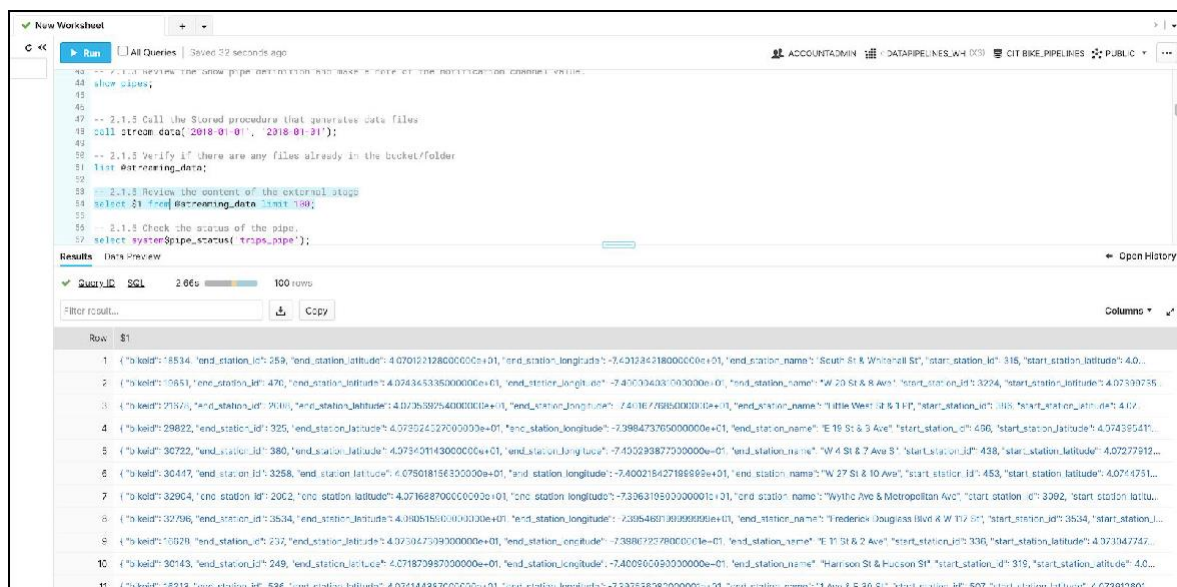
22.1.5 Confirm the setup is working by running the below steps.

Load 1 day of data to test the ingestion by calling the `STREAM_DATA` stored procedure.

```
call stream_data('2018-01-01', '2018-01-01');
```

22.1.6 Verify if there are any files already in the bucket/folder.

```
list @streaming_data;
```



The screenshot shows a Snowflake SQL query editor with a query that calls a stored procedure to load data and then lists the contents of an external stage. The results pane shows a list of files loaded into the stage.

```
-- 2.1.1 Review the snow pipe definition to make a note of the destination external stage.
show pipes;

-- 2.1.3 Call the Stored procedure that generates data files
call stream_data('2018-01-01', '2018-01-01');

-- 2.1.5 Verify if there are any files already in the bucket/folder
list @streaming_data;
```

Results Data Preview

Row	\$1
1	(["keyid": "8534", "end_station_id": 259, "end_station_latitude": 4.070121280000000e+01, "end_station_longitude": -7.431234218000000e+01, "end_station_name": "South St & Whithell St", "start_station_id": 315, "start_station_latitude": 4.0...
2	(["keyid": "10651", "end_station_id": 470, "end_station_latitude": 4.024345335000000e+01, "end_station_longitude": -7.400304331000000e+01, "end_station_name": "W 20 St & 8 Ave", "start_station_id": 3224, "start_station_latitude": 4.02590758...
3	(["keyid": "21571", "end_station_id": 2000, "end_station_latitude": 4.023539254000000e+01, "end_station_longitude": -7.401677995000000e+01, "end_station_name": "Little West St & 1 St", "start_station_id": 1115, "start_station_latitude": 4.02...
4	(["keyid": "29822", "end_station_id": 325, "end_station_latitude": 4.025224222000000e+01, "end_station_longitude": -7.398473765000000e+01, "end_station_name": "E 16 St & 3 Ave", "start_station_id": 430, "start_station_latitude": 4.024355411...
5	(["keyid": "30722", "end_station_id": 380, "end_station_latitude": 4.027401430000000e+01, "end_station_longitude": -7.400293872000000e+01, "end_station_name": "W 4 St & 7 Ave S", "start_station_id": 438, "start_station_latitude": 4.02727912...
6	(["keyid": "30447", "end_station_id": 3258, "end_station_latitude": 4.075018156100000e+01, "end_station_longitude": -7.400215427186666e+01, "end_station_name": "W 27 St & 10 Ave", "start_station_id": 453, "start_station_latitude": 4.0274751...
7	(["keyid": "32004", "end_station_id": 2002, "end_station_latitude": 4.071668700000000e+01, "end_station_longitude": -7.336319502000000e+01, "end_station_name": "Wythe Ave & Metropolitan Ave", "start_station_id": 3392, "start_station_latitude":...
8	(["keyid": "32706", "end_station_id": 3534, "end_station_latitude": 4.030515500000000e+01, "end_station_longitude": -7.395468136000000e+01, "end_station_name": "Frederick Douglas Blvd & W 17 St", "start_station_id": 3534, "start_station_latitude":...
9	(["keyid": "33529", "end_station_id": 237, "end_station_latitude": 4.025045739000000e+01, "end_station_longitude": -7.358672278000000e+01, "end_station_name": "E 17 St & 2 Ave", "start_station_id": 336, "start_station_latitude": 4.023047742...
10	(["keyid": "30143", "end_station_id": 249, "end_station_latitude": 4.021879987000000e+01, "end_station_longitude": -7.400900090000000e+01, "end_station_name": "Harrison St & Fulton St", "start_station_id": 319, "start_station_latitude": 4.0...
11	(["keyid": "3313", "end_station_id": 536, "end_station_latitude": 4.021144887000000e+01, "end_station_longitude": -7.332733828000000e+01, "end_station_name": "11 Ave & E 30 St", "start_station_id": 507, "start_station_latitude": 4.023012801...

22.1.7 Review the contents of the external stage

```
select $1 from @streaming_data limit 100;
```



```
select system$pipe_status('trips_pipe');
```

Snowpipe copies the data into our raw table. Review the table data.

```
select count(*) from
trips_raw; select * from
trips_raw limit 100;
```

Clean up the stage by calling the purge_files function.

The screenshot shows the Snowflake SQL Editor interface. At the top, the title bar says "Details". Below it, the "Run" button is highlighted. The query editor contains the following SQL code:

```
-- select * from streaming_data limit 100;
55
56 -- 2.1.5 Check the status of the pipe.
57 select system$pipe_status('trips_pipe');
58
59 -- 2.1.5 Check if the data is loaded to the trips_raw table.
60 select count(*) from trips_raw;
61
62 -- 2.1.5 Review sample data from trips_raw table.
63 select * from trips_raw limit 100;
64
65 -- 2.1.5 Clean up the stage by calling the purge_files function.
66 call purge_files('trips_raw', '@streaming_data/');
67
68
```

Below the query editor, the "Results" tab is active, showing a "Data Preview" of the query results. The results table has one column, "PURGE_FILES", and one row with the value "0".

Row	PURGE_FILES
1	0

A "Done" button is located at the bottom right of the interface.

```
call purge_files('trips_raw', '@streaming_data/');
```

Truncate table.

```
truncate table trips_raw;
```

22.2 Troubleshooting Snowpipe

If data isn't written to the table you can use the following commands to see where the process is broken.

Verify data was in fact written to your STAGE by the stored procedure.

```
list @streaming_data;
```

Next check the status of the PIPE using the [system\\$pipe_status](#) call. Make sure the result shows status 'RUNNING' and there are no errors present.

Details

```
1 {"executionState":"RUNNING","pendingFileCount":0,"notificationChannelName":"arn:aws:sqs:us-west-2:235488214722:sf-snowpipe-AIDATNVBQ73BDZ5EBJD3V-VZ7EhFqXekKdtfRTZjsKkA","numOutstandingMessagesOnChannel":0,"lastReceivedMessageTimestamp":"2020-11-08T23:10:22.319Z","lastForwardedMessageTimestamp":"2020-11-08T23:10:22.389Z"}
```

Done

```
select system$pipe_status('trips_pipe');
```

Lastly, check the `information_schema.copy_history` to see if there were any problems encountered while writing records to the `trips_raw` table. Parsing errors or permissions errors could accidentally affect your load jobs. You can find the details for each of the files you are loading. Play with the `dateadd` function to adjust the size of the window when looking at processed files.

```
select *
from
table(information_schema.copy_history(
table_name=>'citibike_pipelines.public.trips_raw', start_time=>dateadd(hour, -1,
current_timestamp)));
```

Note: the above commands aren't strictly used for troubleshooting purposes. You can use these anytime to gain insight into the status or jobs and track the velocity of arriving data.