

StreamSets Quick-Start Examples

Table of Contents

Introduction	1
Fragments	2
Read Files from S3 Fragment	2
Snowflake (Direct) Fragment	3
Snowflake (File Uploader) Fragment	5
S3 Files to Snowflake (Direct) pipeline	7
Create an S3 Files to Snowflake (Direct) pipeline	7
Running the S3 Files to Snowflake (Direct) pipeline	10
S3 Files to Snowflake (File Uploader) pipeline	14
Create an S3 Files to Snowflake (File Uploader) pipeline	14
Running the S3 Files to Snowflake (File Uploader) pipeline	15
Creating and Using Sample Pipelines	20

Introduction

This project includes a set of quick-start examples to allow users to quickly assemble pipelines from [Fragments](#) and to use [Sample Pipelines](#).

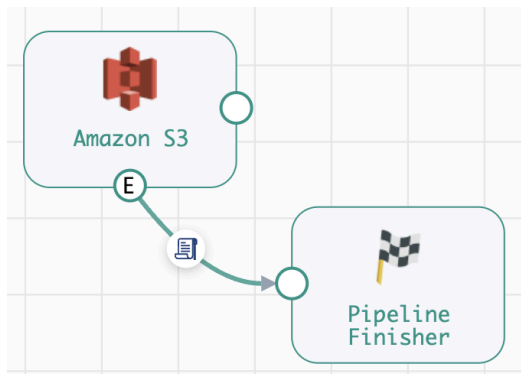
You can download all of the artifacts for this project from [here](#).

Fragments

Download the example Fragments from [here](#) and import them into Control Hub's `Build > Fragments` page

Read Files from S3 Fragment

This Fragment is comprised of two stages; an [Amazon S3 Origin](#) and a [Pipeline Finisher Executor](#):



This Fragment is configured by default with the following assumptions:

- An Amazon S3 [Connection](#) exists and is set in the Amazon S3 Origin
- The files read from S3 are in JSON format and will be parsed by the pipeline that uses the Fragment. If one wants to use Whole File Data Format, one can set that as a parameter as described below.
- The JSON files read from the origin contain multiple JSON objects (i.e. multiple newline delimited JSON objects, for example in the included sample data file [here](#)).
- The Pipeline Finisher does not automatically reset the origin, so if you want to read data repeatedly, use the "Reset Origin" option when starting a pipeline

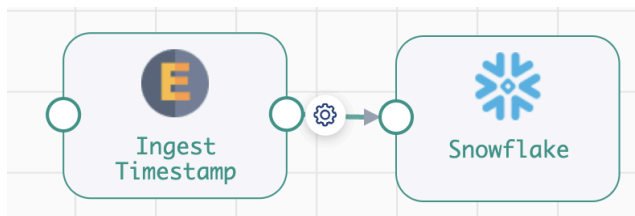
To configure defaults for the Fragment, edit the Fragment and set values for the `S3_BUCKET` and `S3_DATA_FORMAT` (to either `JSON` or `WHOLE_FILE`) . If you are using `JSON` format, use a destination Fragment like `Snowflake (Direct)`, or if you are using `WHOLE_FILE` format, use a destination Fragment like `Snowflake (File Uploader)` .

The `DIRECTORY` and `FILE_PICKUP_PATTERN` parameter will be set in the pipeline that uses this Fragment. Note also that any of the parameter values can be overridden by the pipeline that uses the Fragment.

General	Parameters	Test Origin
Parameters		
		<div>BUCKET : 146bucket</div> <div>DATA_FORMAT : JSON</div> <div>DIRECTORY : Enter Value</div> <div>FILE_PICKUP_PATTERN : Enter Value</div>

Snowflake (Direct) Fragment

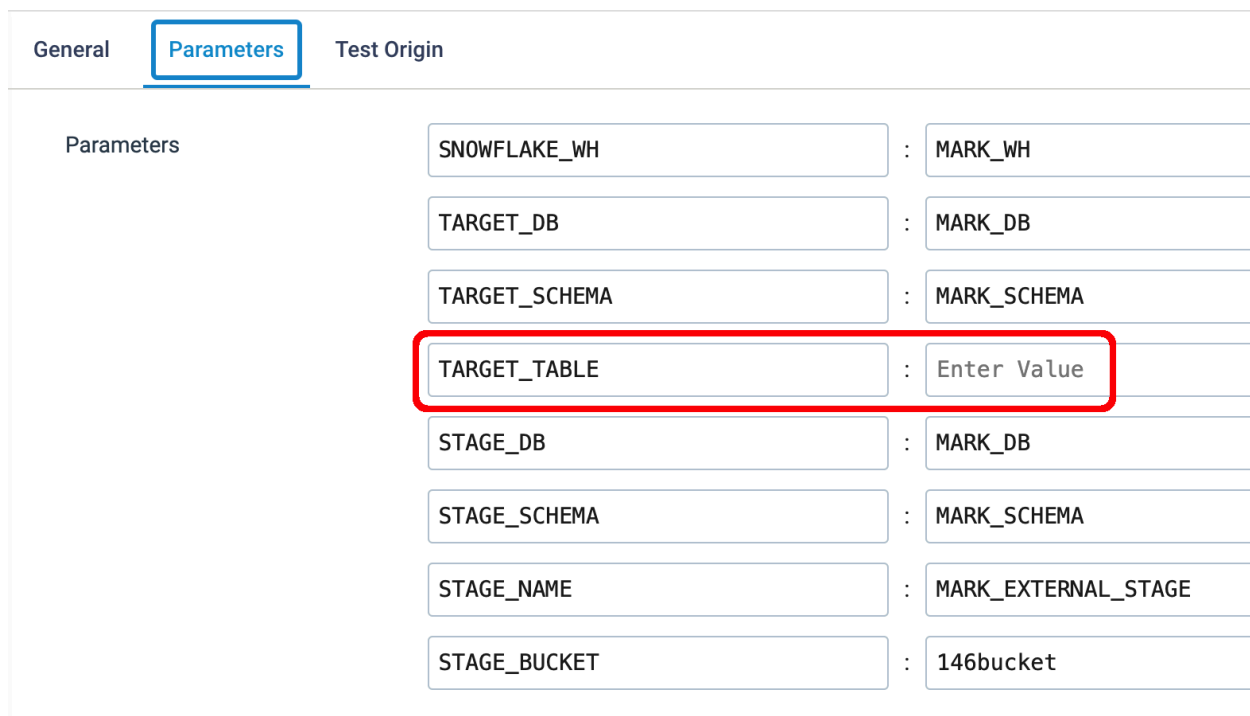
This Fragment is comprised of two stages; an [Expression Evaluator](#) that appends an `ingest_timestamp` field to every record, and a [Snowflake Destination](#):



This Fragment is configured by default with the following assumptions:

- A Snowflake [Connection](#) exists and is set in the Snowflake Destination
- An External Stage already exists in Snowflake. You could also modify the Snowflake Destination's *Staging* tab to use an internal stage if desired.
- Source data in pipelines that use this Fragment must be in a parsable format, such as JSON or Delimited; [Whole File Data Format](#) is not supported by this Fragment. (To use Whole File Data Format, use the Snowflake (File Uploader) Fragment instead, as described below.
- Data Collector's auto-generated Snowflake File Format will correctly handle the source data. You can set your own file format options in the Snowflake stage's config or specify your own File Format if needed.

To configure defaults for the Fragment, edit the Fragment and set values for all of the Parameters except the `TARGET_TABLE`. The `TARGET_TABLE` parameter will be set in the pipeline that uses this Fragment. The Fragment is configured to auto-create the target table if it does not exist in advance. Note that any of these parameter values can be overridden by the pipeline that uses the Fragment

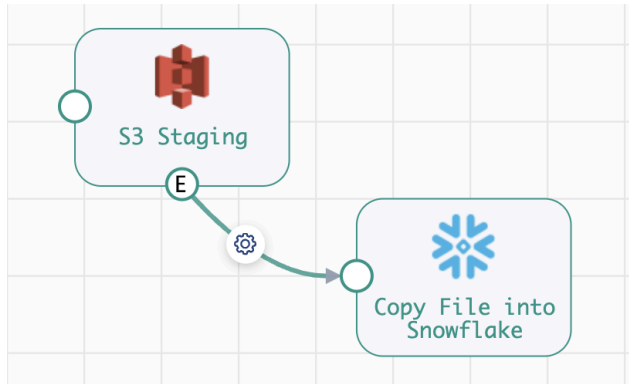


General	Parameters	Test Origin
Parameters		
SNOWFLAKE_WH	:	MARK_WH
TARGET_DB	:	MARK_DB
TARGET_SCHEMA	:	MARK_SCHEMA
TARGET_TABLE	:	Enter Value
STAGE_DB	:	MARK_DB
STAGE_SCHEMA	:	MARK_SCHEMA
STAGE_NAME	:	MARK_EXTERNAL_STAGE
STAGE_BUCKET	:	146bucket

If the parameter list appears too long, feel free to hard-code any of the appropriate parameterized properties in the stage config itself, and delete any of the no-longer-needed parameters.

Snowflake (File Uploader) Fragment

This Fragment is comprised of two stages; an [Amazon S3 Destination](#) configured to use Whole File Format and to emit events, and a [Snowflake Executor](#) configured to execute a COPY INTO command when a file is written to the external stage location:



This Fragment is configured by default with the following assumptions:

- An S3 [Connection](#) exists and is set in the S3 Destination
- A Snowflake Connection exists and is set in the Snowflake Executor
- The Snowflake target table must exist in advance
- An External Stage already exists in Snowflake at a known directory within an S3 bucket.
- A Snowflake File Format exists that can parse the source data

To configure defaults for the Fragment, edit the Fragment and set values for all of the Parameters except the Snowflake `TARGET_TABLE` and `FILE_FORMAT_NAME`

General Parameters Test Origin

Parameters

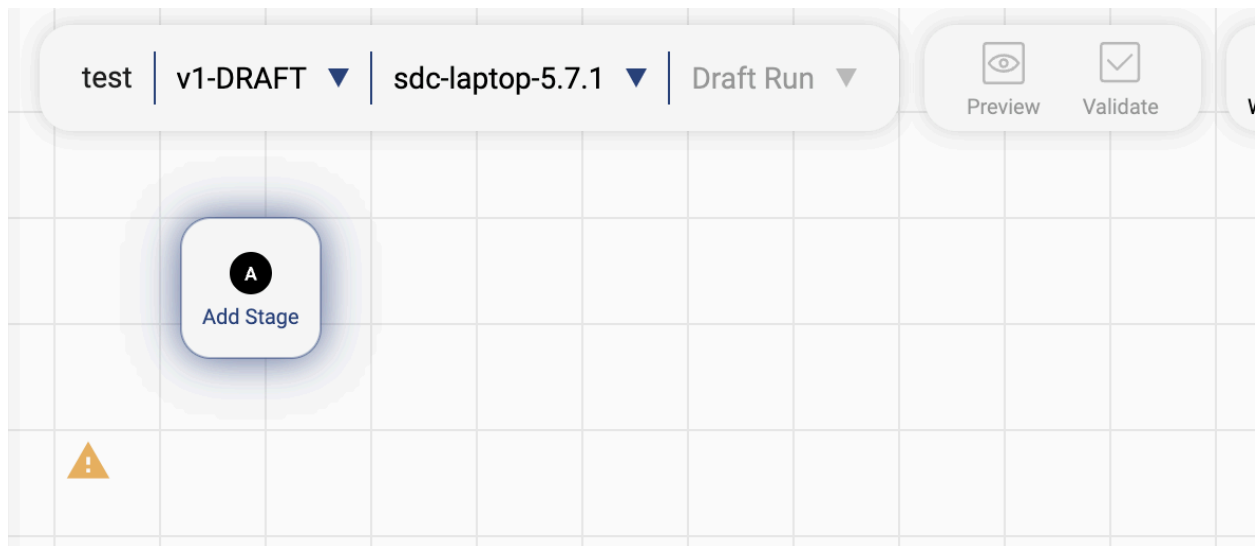
S3_STAGE_BASE_DIR	:	snowflake_external_stage
S3_STAGE_BUCKET	:	146bucket
SNOWFLAKE_WH	:	MARK_WH
TARGET_DB	:	MARK_DB
TARGET_SCHEMA	:	MARK_SCHEMA
TARGET_TABLE	:	Enter Value
STAGE_DB	:	MARK_DB
STAGE_SCHEMA	:	MARK_SCHEMA
STAGE_NAME	:	MARK_EXTERNAL_STAGE
FILE_FORMAT_DB	:	MARK_DB
FILE_FORMAT_SCHEMA	:	MARK_SCHEMA
FILE_FORMAT_NAME	:	Enter Value

S3 Files to Snowflake (Direct) pipeline

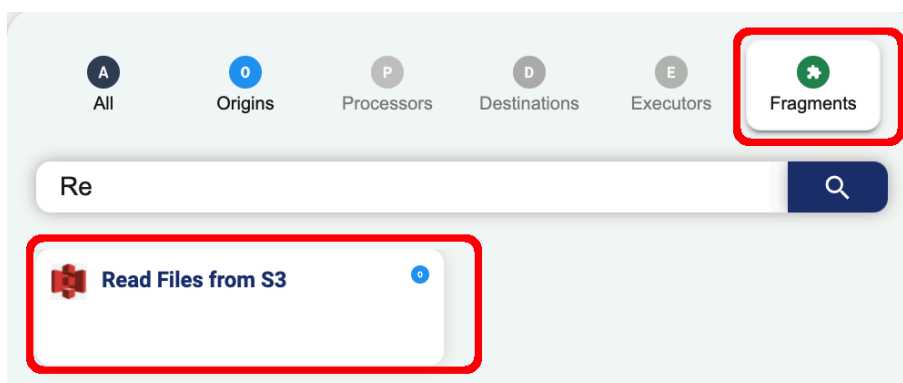
Create an S3 Files to Snowflake (Direct) pipeline

Here are step by step instructions for how to build a pipeline that uses the `Read Files from S3` and `Snowflake (Direct)` Fragments:

- Create a new pipeline:



Click on `Add Stage` and select the `Read Files from S3` Fragment



When prompted, set a Fragment parameter prefix like `S3_` :

Fragment Parameter

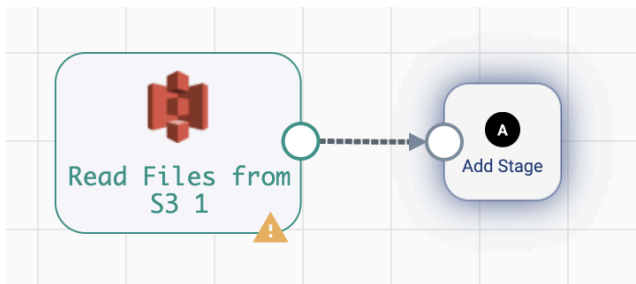
Parameter Name Prefix

S3_

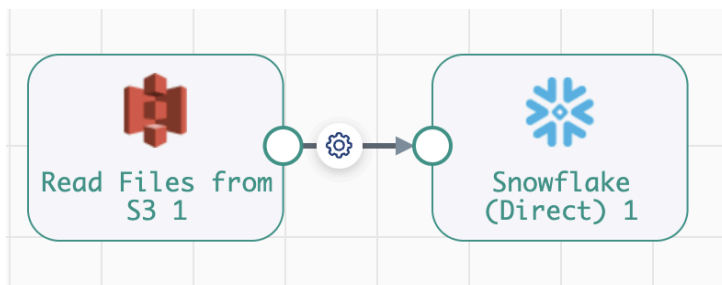
Cancel

Done

Click Done and the Fragment is added to the Pipeline:



Click on `Add Stage` and select the Fragment `Snowflake (Direct)`, set its parameter prefix to `SF_` and the pipeline should now look like this:



View the pipeline's parameters and you should see the combined parameters from both Fragments, most of which are already set with the Fragments' default values. In this case, set only the SF_TARGET_TABLE (for the Snowflake table to be written to), and the S3_DIRECTORY and S3_FILE_PICKUP_PATTERN file pickup pattern. You can also override any of the default values inherited from the Fragment:

General

Parameters

Notifications

Error Records

Advanced

Test Origin

Hide Advanced Options ^

Parameters

SF_SNOWFLAKE_WH	:	MARK_WH
SF_TARGET_DB	:	MARK_DB
SF_TARGET_SCHEMA	:	MARK_SCHEMA
SF_TARGET_TABLE	:	Enter Value
SF_STAGE_DB	:	MARK_DB
SF_STAGE_SCHEMA	:	MARK_SCHEMA
SF_STAGE_NAME	:	MARK_EXTERNAL_STAGE
SF_STAGE_BUCKET	:	146bucket
S3_BUCKET	:	146bucket
S3_DATA_FORMAT	:	JSON
S3_DIRECTORY	:	Enter Value
S3_FILE_PICKUP_PATTERN	:	Enter Value

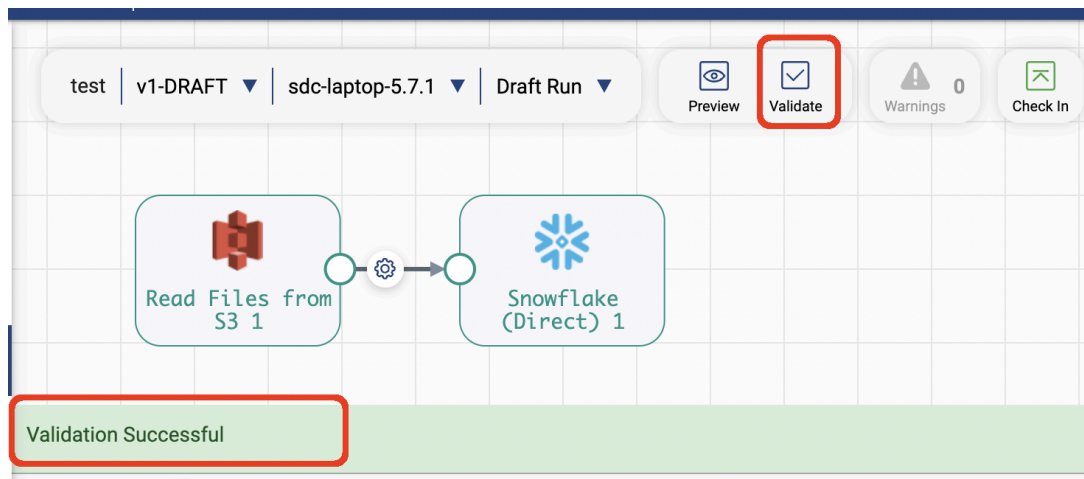
Running the S3 Files to Snowflake (Direct) pipeline

To run the S3 Files to Snowflake (Direct) pipeline I will copy the file `retail.json` provided [here](#) to my S3 bucket named `146bucket` to the S3 directory `sample-data`.

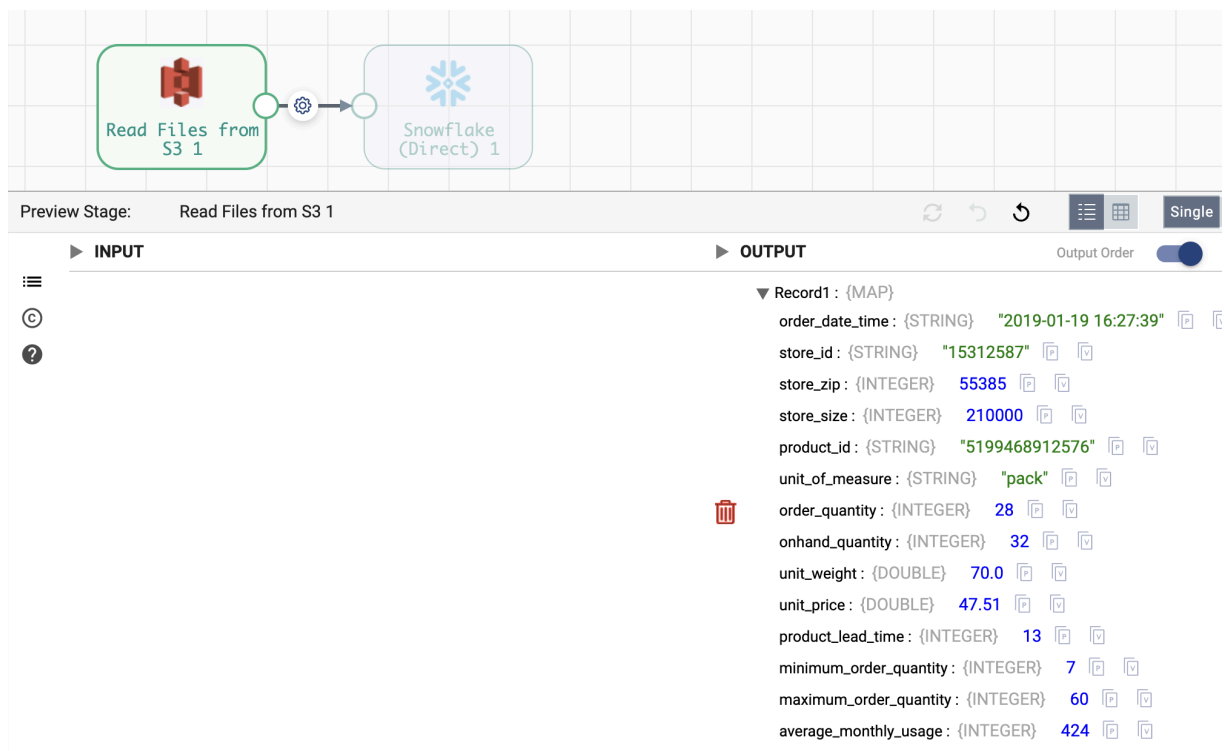
I will set the highlighted pipeline parameters:

General	Parameters	Notifications	Error Records	Advanced	Test Origin
Hide Advanced Options ^					
Parameters					
SF_SNOWFLAKE_WH		:	MARK_WH		
SF_TARGET_DB		:	MARK_DB		
SF_TARGET_SCHEMA		:	MARK_SCHEMA		
SF_TARGET_TABLE		:	ORDERS		
SF_STAGE_DB		:	MARK_DB		
SF_STAGE_SCHEMA		:	MARK_SCHEMA		
SF_STAGE_NAME		:	MARK_EXTERNAL_STAGE		
SF_STAGE_BUCKET		:	146bucket		
S3_BUCKET		:	146bucket		
S3_DATA_FORMAT		:	JSON		
S3_DIRECTORY		:	sample-data		
S3_FILE_PICKUP_PATTERN		:	*.json		

Validate the pipeline to confirm connectivity is good:



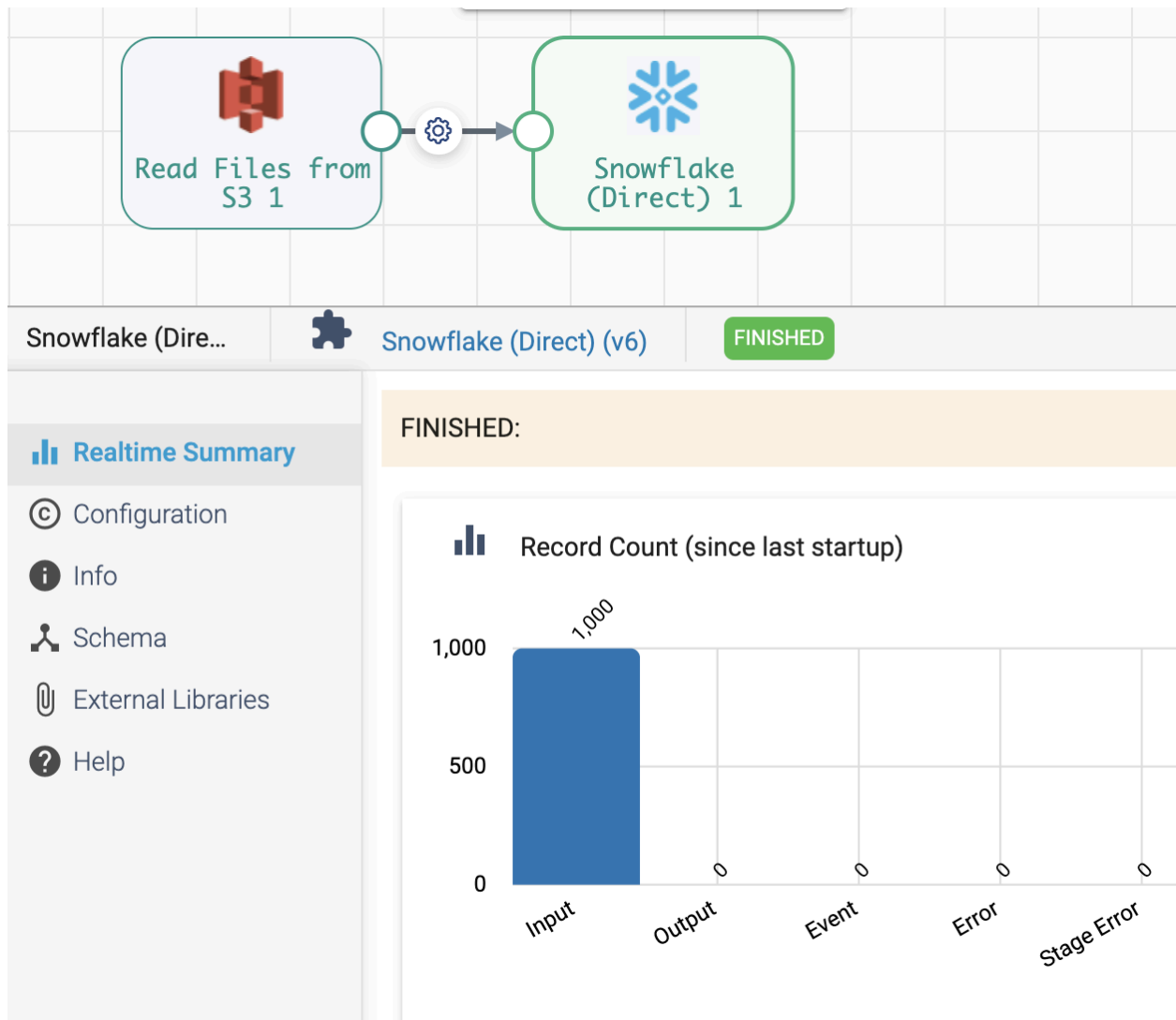
Preview the pipeline to confirm the JSON file is found and can be parsed:



Note the JSON has been parsed into strongly typed fields.

Close the Preview.

Select Draft Run > Reset Origin & Start and you should see 1000 records are read from S3 and written to Snowflake:



Confirm the ORDERS table was created and populated in Snowflake:

```

1  create or replace TABLE MARK_DB.MARK_SCHEMA.ORDERS (
2      ORDER_DATE_TIME VARCHAR(16777216),
3      STORE_ID VARCHAR(16777216),
4      STORE_ZIP NUMBER(38,0),
5      STORE_SIZE NUMBER(38,0),
6      PRODUCT_ID VARCHAR(16777216),
7      UNIT_OF_MEASURE VARCHAR(16777216),
8      ORDER_QUANTITY NUMBER(38,0),
9      ONHAND_QUANTITY NUMBER(38,0),
10     UNIT_WEIGHT FLOAT,
11     UNIT_PRICE FLOAT,
12     PRODUCT_LEAD_TIME NUMBER(38,0),
13     MINIMUM_ORDER_QUANTITY NUMBER(38,0),
14     MAXIMUM_ORDER_QUANTITY NUMBER(38,0),
15     AVERAGE_MONTHLY_USAGE NUMBER(38,0),
16     INGEST_TIMESTAMP TIMESTAMP_NTZ(9),
17     PACKAGE_WEIGHT FLOAT
18 );

```

Data Preview

	ORDER_DATE_TIME	STORE_ID	STORE_ZIP	STORE_SIZE	PRODUCT_ID	UNIT_OF_MEASURE	
1	2019-01-19 16:27:39	15312587	55385	210000	5199468912576	pack	
2	2019-01-14 04:29:33	82525545	55401	200000	3420285437020	barrel	
3	2019-01-15 17:09:38	32713442	55380	210000	5626682225224	barrel	
4	2019-01-21 05:25:27	31065115	55381	250000	1284741567186	barrel	
5	2019-01-28 08:09:39	55508308	55358	220000	3224337794433	bag	
6	2019-01-31 06:40:07	61975378	55348	180000	0155491151769	box	
7	2019-01-05 13:41:52	38950827	55337	230000	5544405043798	bag	
8	2019-01-21 03:31:09	61975378	55348	180000	2581928565720	bag	
9	2019-01-19 07:16:42	49354836	55369	250000	6536405196498	box	
10	2019-01-10 12:51:55	95002811	55401	210000	7073704866909	pallet	
11	2019-01-16 18:51:37	71501642	55336	220000	5205790709361	box	
12	2019-01-09 11:19:11	98419777	55330	220000	3272781936565	pallet	
13	2019-01-16 23:35:25	25442045	55336	250000	6593784065593	bag	
14	2019-01-09 19:17:35	98107452	55387	220000	7642776226166	box	

S3 Files to Snowflake (File Uploader) pipeline

Create an S3 Files to Snowflake (File Uploader) pipeline

To create a pipeline that uses the `Snowflake (File Uploader)` Fragment, follow the same steps as the previous pipeline example, but use the `Snowflake (File Uploader)` Fragment rather than the `Snowflake (Direct)` Fragment.

Set the `S3_DATA_FORMAT` parameter to `WHOLE_FILE`

This new pipeline's parameter list should look like this:

General	Parameters	Notifications	Error Records	Advanced	Test Origin
Parameters					
S3_BUCKET		:	146bucket		
S3_DATA_FORMAT		:	WHOLE_FILE		
S3_DIRECTORY		:	Enter Value		
S3_FILE_PICKUP_PATTERN		:	Enter Value		
SF_S3_STAGE_BASE_DIR		:	snowflake_external_stage		
SF_S3_STAGE_BUCKET		:	146bucket		
SF_SNOWFLAKE_WH		:	MARK_WH		
SF_TARGET_DB		:	MARK_DB		
SF_TARGET_SCHEMA		:	MARK_SCHEMA		
SF_TARGET_TABLE		:	Enter Value		
SF_STAGE_DB		:	MARK_DB		
SF_STAGE_SCHEMA		:	MARK_SCHEMA		
SF_STAGE_NAME		:	MARK_EXTERNAL_STAGE		
SF_FILE_FORMAT_DB		:	MARK_DB		
SF_FILE_FORMAT_SCHEMA		:	MARK_SCHEMA		
SF_FILE_FORMAT_NAME		:	Enter Value		

At runtime, the user will need to provide values for the four highlighted empty parameters. As mentioned earlier, if this parameter list appears too long, you can hard-code some values within the Fragments and trim the parameter list.

Running the S3 Files to Snowflake (File Uploader) pipeline

To run the S3 Files to Snowflake (File Uploader) pipeline I will copy the file `employees.csv` provided [here](#) to my S3 bucket named `146bucket` to the S3 directory `sample-data`.

I will create a Snowflake File Format named `MARK_QUOTED_CSV_FORMAT` using this command:

```
CREATE OR REPLACE FILE FORMAT MARK_QUOTED_CSV_FORMAT
  TYPE = 'CSV'
  FIELD_DELIMITER = ','
  FIELD_OPTIONALLY_ENCLOSED_BY='\"';
```

I will create the target Snowflake table in advance:

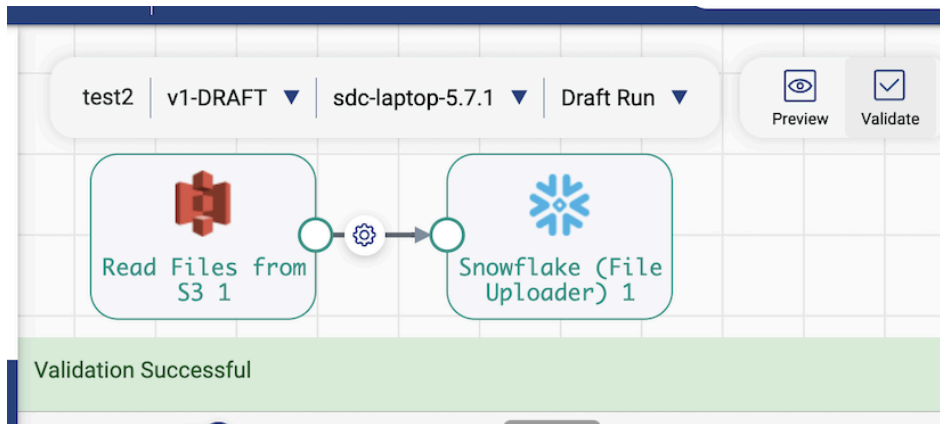
```
CREATE TABLE MARK_DB.MARK_SCHEMA.EMPLOYEES (
  ID BIGINT,
  NAME STRING,
  ADDRESS STRING);
```

I will set the highlighted pipeline parameters:

General	Parameters	Notifications	Error Records	Advanced	Test Origin
---------	------------	---------------	---------------	----------	-------------

Parameters	S3_BUCKET	:	146bucket
	S3_DATA_FORMAT	:	WHOLE_FILE
	S3_DIRECTORY	:	sample-data
	S3_FILE_PICKUP_PATTERN	:	*.csv
	SF_S3_STAGE_BASE_DIR	:	snowflake_external_stage
	SF_S3_STAGE_BUCKET	:	146bucket
	SF_SNOWFLAKE_WH	:	MARK_WH
	SF_TARGET_DB	:	MARK_DB
	SF_TARGET_SCHEMA	:	MARK_SCHEMA
	SF_TARGET_TABLE	:	EMPLOYEES
	SF_STAGE_DB	:	MARK_DB
	SF_STAGE_SCHEMA	:	MARK_SCHEMA
	SF_STAGE_NAME	:	MARK_EXTERNAL_STAGE
	SF_FILE_FORMAT_DB	:	MARK_DB
	SF_FILE_FORMAT_SCHEMA	:	MARK_SCHEMA
	SF_FILE_FORMAT_NAME	:	MARK_QUOTED_CSV_FORMAT

Validate the pipeline to confirm connectivity is good:



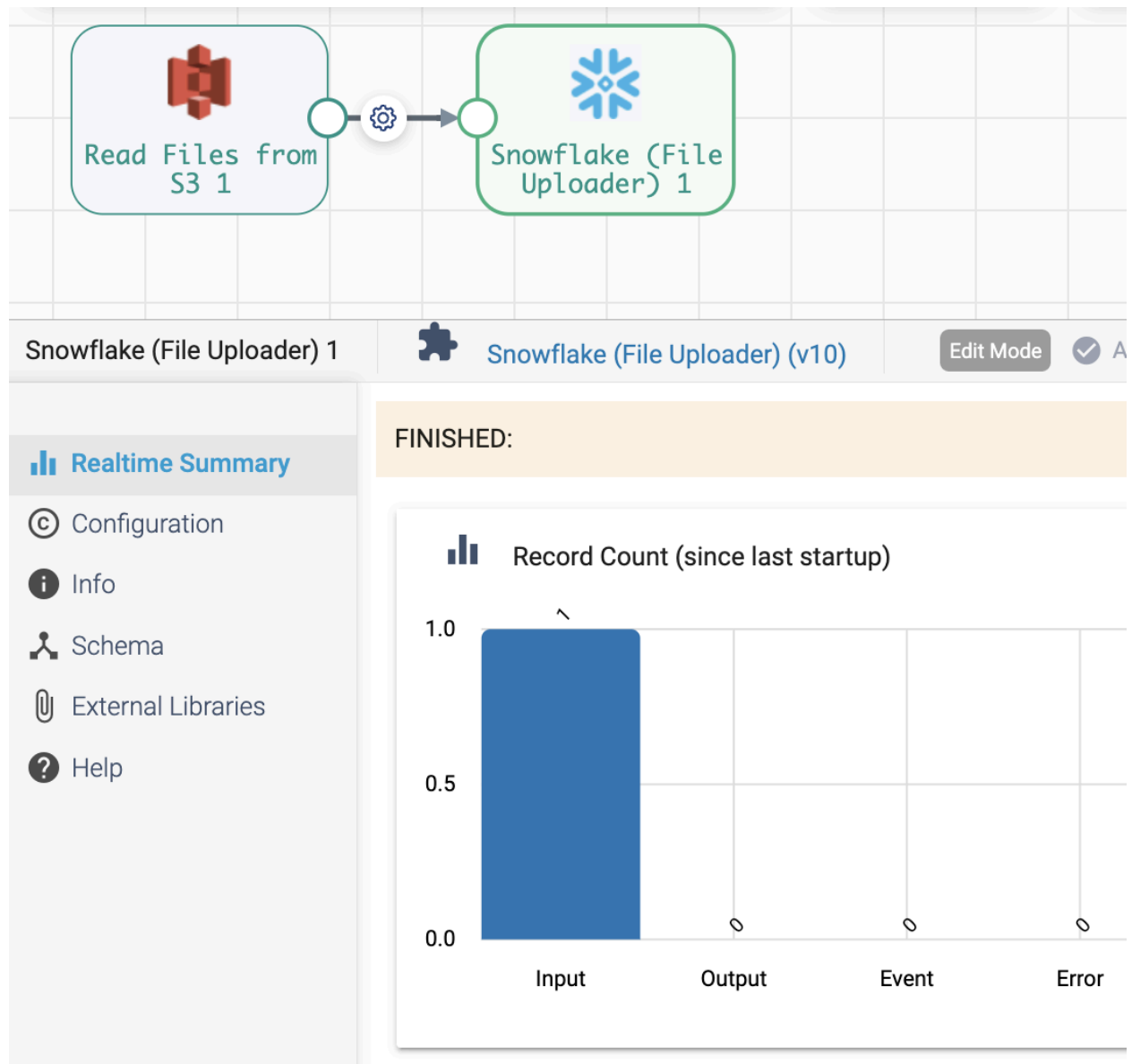
Preview the pipeline to see the file that is picked up from S3:

The screenshot shows the pipeline preview interface. At the top, there are tabs for 'test2', 'View Logs', and 'Close Preview'. The main area displays the pipeline diagram with the 'Read Files from S3 1' stage highlighted by a red box. Below the diagram, the 'Preview Stage: Read Files from S3 1' is shown. The 'INPUT' section is empty, and the 'OUTPUT' section shows a single record with the following fields:

Field	Value
fileRef	{FILE_REF} [object Object]
fileInfo	{MAP}
owner	{STRING} "S3Owner [name=markaws,id=5a1f27b1d88:278d772ed8c2bf66a04f46a25f8ea95de4a]"
Content-Range	{LONG} 17751839
Last-Modified	{DATETIME} Feb 29, 2024 6:58:03 PM
Content-MD5	{STRING}
bucket	{STRING} "146bucket"
filename	{STRING} "sample-data/employees.csv"

Close the Preview.

Select Draft Run > Reset Origin & Start and you should see 1 file is read from S3 and written to Snowflake:



Confirm the EMPLOYEES table was populated in Snowflake:

Data Preview

	ID	NAME	ADDRESS
1	8192986676757054530	Josiah Rau	Suite 078 4039 Breitenberg Rapids, Lake Eusebiafort, ND 894
2	7057449624894708419	Ronny Fahey	Suite 897 465 O'Conner Lights, Robelshire, DE 03368-4733
3	3249338004677987968	Adam Kuphal	7539 Garret Drive, Feilside, NY 58349
4	4300472919448150184	Lynwood Jakubowski	Suite 998 2864 Cummings Islands, North Trey, MI 49253
5	6803173852271527073	Jonie Ankunding Sr.	1461 Damon Islands, West Gilbertport, MI 47926
6	2959618324222561874	Ayanna Towne	477 Andy Knoll, Langworthborough, MA 44393-6303
7	7029179294379784194	Denae Franecki	8314 Bins Parkway, West Tamalaside, WY 38897-2161
8	3642468504132086688	Kris Buckridge Sr.	556 Oberbrunner Stream, Welchburgh, MI 01150
9	5157659581415794065	Winford D'Amore	307 Dooley Trail, South Juliannaland, VT 55490-0431
10	1695833149951710204	Jeffery Halvorson	5457 Barney Port, North Christinia, OR 38947-4399
11	5422555966301214725	Marshall Torphy	Apt. 523 7801 Dare Island, DuBuqueland, MD 15898-2864

Creating and Using Sample Pipelines

Add the label `templates` to the pipelines and check in the pipeline, and you should see that they are added to the User Samples when creating new pipelines.

Create a new pipeline and start with a Sample Pipeline:

New Pipeline

1

Define Pipeline

Define the pipeline name, the type of engine for the pipeline, and whether to start v sample pipeline. [Learn more](#)

Name:

My JSON Load to Snowflake

Description:

Engine Type:

☒ Data Collector - Runs data ingestion pipelines that perform transformations in streaming, CDC, or batch modes

☐ Transformer - Runs data processing pipelines on Spark that transformations such as joins, aggregates, and sorts on the

☐ Transformer for Snowflake - Runs data processing pipelines

Start with:

☐ Blank Pipeline

☒ Sample Pipeline

Cancel

Next

Select a sample pipeline

Sample Pipelines (8)			
Name ↑	Tutorial	Origins	Destinations
<input type="radio"/> Field -Level Transforms	Click to view		
<input type="radio"/> jdbc to s3	Click to view		
<input checked="" type="radio"/> S3 Files to Snowflake (Direct)	Click to view		
<input type="radio"/> S3 Files to Snowflake (File Up...	Click to view		
<input type="radio"/> SFTP to Snowflake (using Frag...	Click to view		
<input type="radio"/> SFTP to Snowflake (using Frag...	Click to view		
<input type="radio"/> Solution - Lab 1 Basic Ingestion	Click to view	JDBC Multitable Consumer	Snowflake
<input type="radio"/> Solution Lab 2 - Streaming Pip...	Click to view		Snowflake

A new pipeline duplicated from the sample will be created.

The user can now edit any parameters or configs, can add or delete stages, etc....