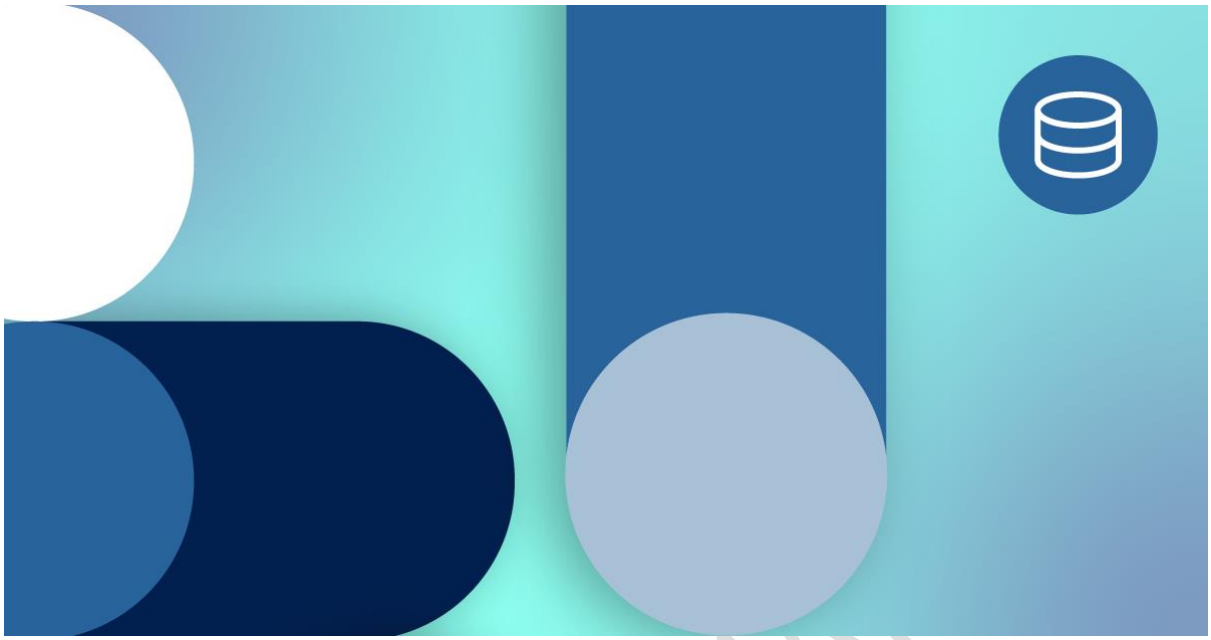
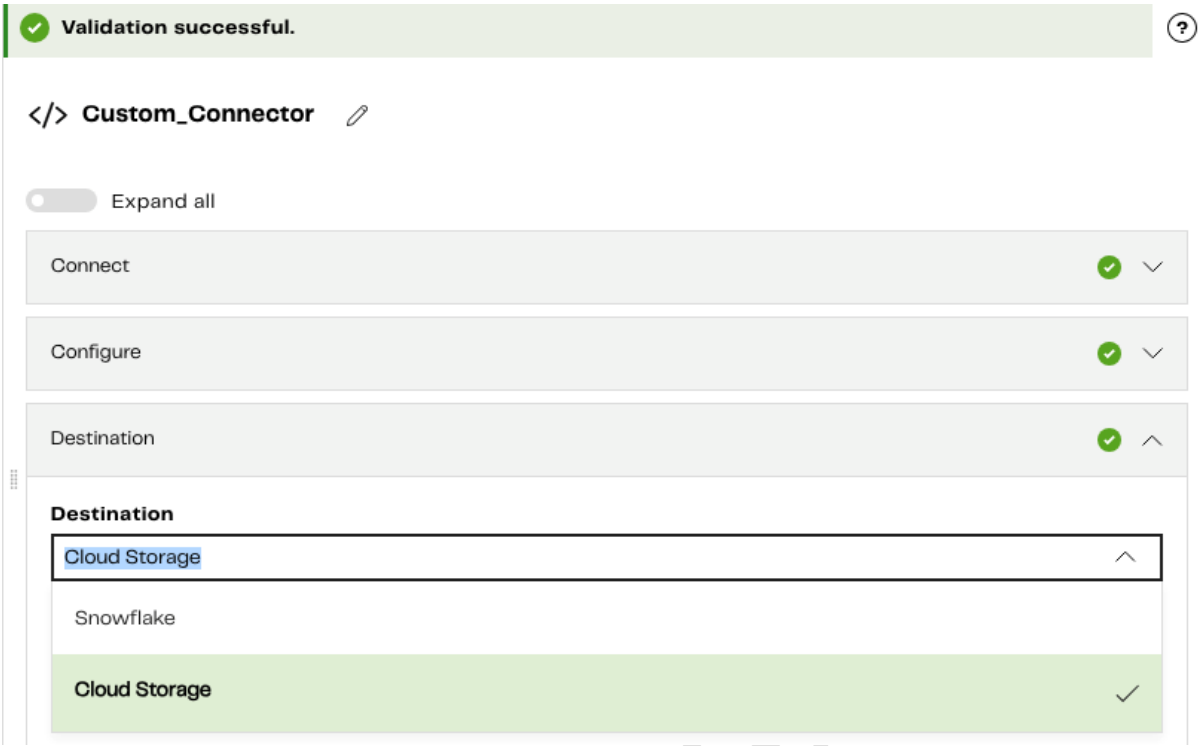
Creating external tables in Snowflake from Custom Connector response data in cloud storage



Within the Data Productivity Cloud, we can build and make use of Custom Connectors to make custom API calls, and load the data into Snowflake by referencing the custom connector in a pipeline in Designer. There are a few options on how to interact with the responses from the API calls that the Custom Connector will make and receive:

1. We can load the response(s) directly into a table in Snowflake without the need for any cloud storage. This is the typical use for Custom Connector components in Designer
2. We can take the responses and upload them to cloud storage such as an S3 bucket or Blob containers

These options are controlled by the 'Destination' properties on the Custom Connector component:



After some researching and thinking, I thought it may be a cool idea to take the responses from the API calls I am making with the Custom Connector and upload them to an S3 bucket in AWS and make use of external tables in Snowflake to read the response files in the S3 bucket and populate a table in Snowflake that way.

To do this, there are a few prerequisite steps that need to be taken before we can make full use of this scenario:

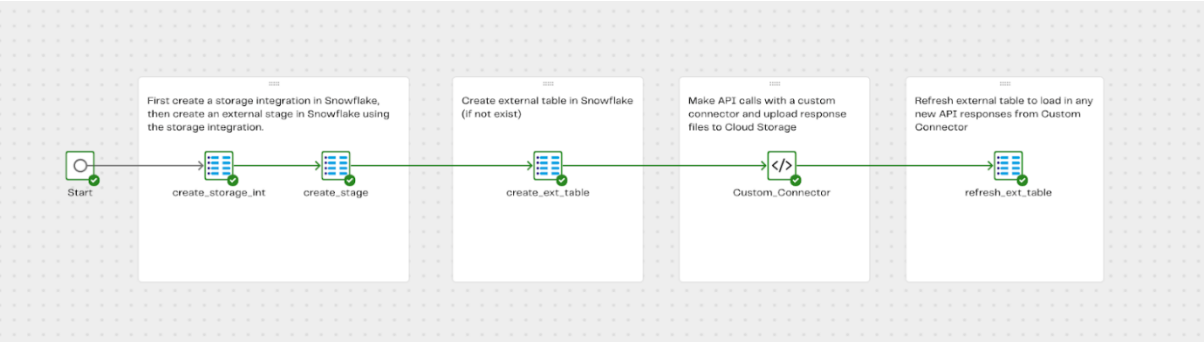
1. We need to have a storage integration in Snowflake created, so there is permissible interaction between Snowflake and the S3 bucket being referenced in the Custom Connector component in Designer
2. We also need to create an external stage in Snowflake that will leverage the storage integration and ultimately be used by the External table to populate table data

Once these prerequisite steps are completed, we can leverage Designer and the SQL Script orchestration component to create the external table in Snowflake, which will read the responses uploaded from the Custom Connector and populate our external table. Overtime as more response files are uploaded to cloud storage, we can build in a mechanism to refresh the external table to contain the new response files, or even explore refreshing the table automatically.

Overall, as a simple design pattern, this would look like:

1. Use the SQL Script to create the external table (recommended to use the parameter of IF NOT EXISTS in the create table script as this will keep the script future-proof for further executions)
2. Make API Calls with the Custom Connector component and configure it to upload response files to cloud storage

3. Use the SQL Script component to refresh the external table so it will include any new response files uploaded from the Custom Connector component



SQL code to create a Snowflake Storage Integration and External Stage

Creating a storage integration:

```
CREATE STORAGE INTEGRATION <Storage_integration_name>
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = 'S3'
  STORAGE_AWS_ROLE_ARN = '<AWS_ROLE_ARN>'
  ENABLED = TRUE
  STORAGE_ALLOWED_LOCATIONS = ('<S3_location>');
```

Creating an External Stage

```
  CREATE STAGE <External_stage_name>
  STORAGE_INTEGRATION = <Storage_integration_name>
  URL = '<s3_location>'
```

Pros and Cons of External Tables:

Pros:

1. Data does not have to be loaded into Snowflake, and external tables can populate table data from a cloud storage location
2. External tables can be automatically refreshed so when new files land in cloud storage, the table data reflects new data
3. External tables support query and join operations
4. Views can be created from external tables

Cons:

1. External tables are read-only, so no DML (Data manipulation language) operations can occur against External tables
2. The creation of external tables and required dependencies (stages, storage integrations) can require some technical familiarity with Snowflake to configure and set up.
3. Data files can build up in cloud storage over time, as external tables are populated directly by files stored in cloud storage, so they must be present
4. There can be a few different variations of setting up External tables, such as the creation of storage integrations and stages (technically, external tables can be created and populate data without the use of a storage integration and only with the use of a stage, but the stage will need to be set up slightly differently)

Conclusion:

As we can see, there are a few ways to interact with and populate tables in Snowflake from API responses made by Custom Connectors in Designer.

Matillion is the data pipeline platform that empowers data teams to build and manage pipelines faster for AI and Analytics at scale. Its flexibility is exemplified by its capability to interact with and populate tables in Snowflake in a number of different ways. Matillion's versatile UI, pre-built components, and code options including SQL, Python, and DBT, allow you to seamlessly integrate external tables, helping to support all your AI and analytics workloads.