

Integrate.io – Setup and Configuration

Written by: [Hristo Hristov](#) | August 26, 2024

Test Environment Setup

To simulate a real-time transactional system, I have imported the [Wide World Importers database](#) from Microsoft. Then, I defined this simple query which produces a consolidated result of orders and order lines:

```
SELECT o.[OrderID]
      ,[CustomerID]
      ,[SalespersonPersonID]
      ,[ContactPersonID]
      ,[OrderDate]
      ,[ExpectedDeliveryDate]
      ,[CustomerPurchaseOrderNumber]
      ,[IsUndersupplyBackordered]
      ,ol.[OrderLineID]
      ,ol.[OrderID]
      ,[StockItemID]
      ,[Description]
      ,[Quantity]
      ,[UnitPrice]
      ,[TaxRate]
      ,(Quantity * UnitPrice) * (1 + TaxRate/100.0) SaleTotal
FROM [WideWorldImporters].[Sales].[Orders] o
JOIN [Sales].[OrderLines] ol ON o.OrderID = ol.OrderID
ORDER BY o.OrderID
```

Here is how the result set looks:

The screenshot displays a SQL query window with the following text:

```
SELECT o.[OrderID],
       [CustomerID],
       [SalespersonPersonID],
       [ContactPersonID],
       [OrderDate],
       [ExpectedDeliveryDate],
       [CustomerPurchaseOrderNumber],
       [IsUndersupplyBackordered],
       [OrderLineID],
       ol.[OrderID],
       [StockItemID],
       [Description],
       [Quantity],
       [UnitPrice],
       [TaxRate],
       (Quantity * UnitPrice) * (1 + TaxRate/100.0) SaleTotal
FROM [WideWorldImporters].[Sales].[Orders] o
JOIN [Sales].[OrderLines] ol ON o.OrderID = ol.OrderID
ORDER BY o.OrderID
```

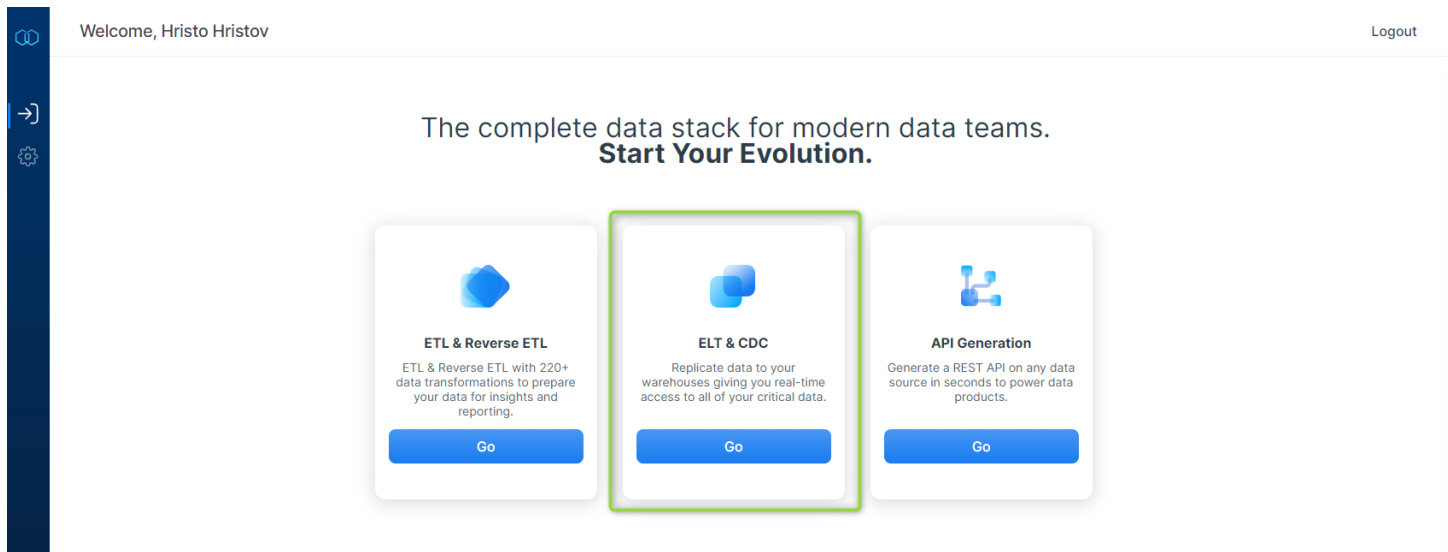
Below the query, the 'Results' tab shows a table with 14 columns: OrderID, CustomerID, SalespersonPersonID, ContactPersonID, OrderDate, ExpectedDeliveryDate, CustomerPurchaseOrderNumber, IsUndersupplyBackordered, OrderLineID, OrderID, StockItemID, and Description. The table contains 44 rows of data, with the first row being highlighted. The status bar at the bottom indicates 'Query executed successfully.' and '231,412 rows'.

Next, I wrote a python script that runs this query every 5 seconds, takes every next row and inserts it into a “staging” table *LiveOrdersData* on an Azure SQL Server. This process simulates the arrival of new sales data in an OLTP system at regular five-second intervals and will serve as the source data that we will plug into Integrate.io for CDC and replication. With this scenario in mind, let us see how to leverage the platform’s capabilities to make it happen.

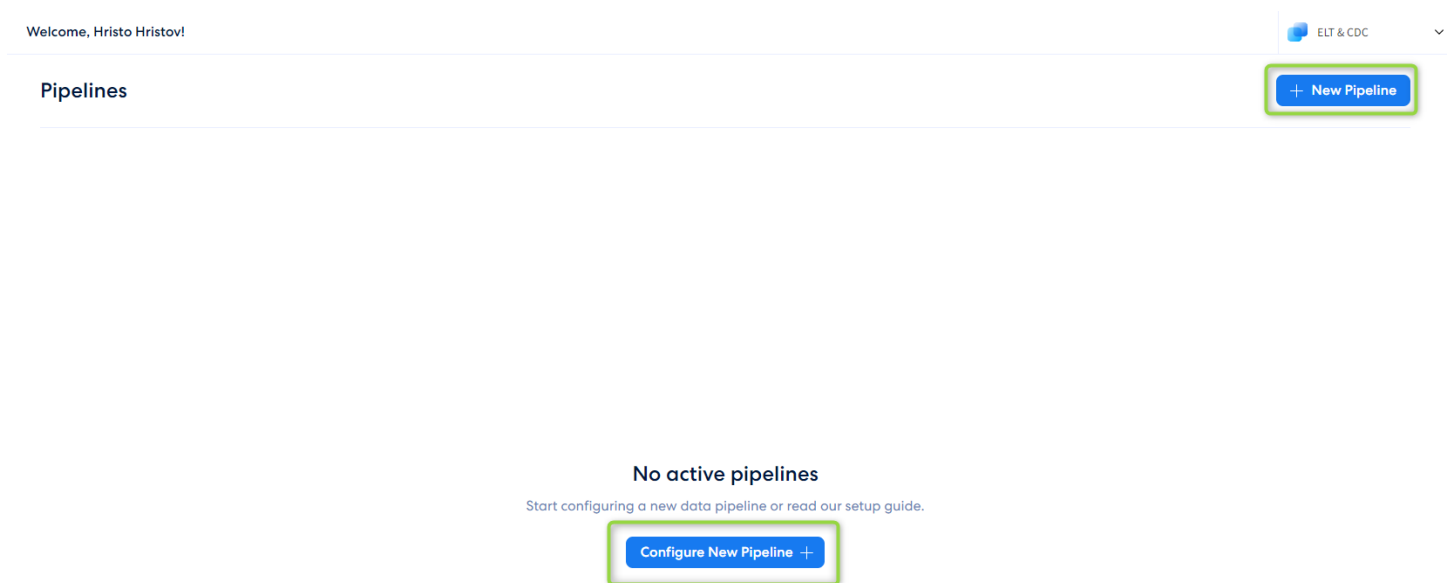
Set up the source

Once you have signed up for Integrate.io, go ahead and can sign in. Then you will be welcomed by the following screen. Click *Go* below the ELT & CDC card. This action will take you to your list of pipelines.

Create a pipeline



This will be the first pipeline so click *New Pipeline*:



You will get a pop-up asking you to choose a name and data processing location. For the data processing location pick a region that is the closest to your data source and finally click on *Start configuration*:

Create pipeline ×

Pipeline name

Add a pipeline name that's easy to recognize.

Data processing location

 ▼

This is where ELT & CDC will operate and run computation on data. Please whitelist the ELT & CDC IPs in your firewall as required.

Select a data source

The first step in the process of creating a real-time CDC pipeline is to choose and configure a data source. In our case this is Azure SQL so we can pick MSSQL Server CDC:

< Real-time Sales Data

01 Configure Source > 02 Configure Destination > 03 Schema Settings > 04 Sync Settings

Select source

MySQL CDC

PostgreSQL CDC

Microsoft SQL Server CDC

Then specify the variant:

Select source

< Select Microsoft SQL Server CDC variant

RDS SQL Server

Azure SQL Database

Others

At this point the main configuration menu will unfold before you. Here we see a well-organized list of steps that we are required to follow to set up the data source correctly. The first tab gives us an overview of the requirements. Most importantly, CDC must be enabled on the database and the table:

01

Configure Source >

02

Configure Destination >

03

Schema Settings >

04

Sync Settings

[« Select an existing source](#)

Setting up Microsoft SQL Server CDC for ELT & CDC

Before you begin to add a Microsoft SQL Server Database as source, please make sure that it is configured to work with Integrate.io Sync by following the steps below.

01 Overview

ELT & CDC uses change data capture (CDC) for Microsoft SQL Server.

Requirements

- Microsoft SQL Server 2008 (10.0) version or above.
- Sysadmin permissions are required for SQL Server or Azure SQL Managed.
- The db_owner role is required to enable change data capture for Azure SQL Database.
- CDC must be enabled for the database.
- CDC must be enabled for the needed tables.

[Continue](#)

02 Create role for sync

03 Grant necessary privileges

04 Enable CDC for the database

05 Enable CDC for the needed tables

Create role for sync

Let us move to step two, which is the first configuration step. While this is not required, Integrate.io is mindful by suggesting we need a dedicated db login, which is a good practice for such scenarios:

02 Create role for sync

Create a sync user for ELT & CDC by executing,

```
CREATE LOGIN integrateio WITH PASSWORD = [REDACTED];
```

Copy

Continue

Grant necessary privileges

Having a new login, we must assign it to the correct roles. These are db_ddladmin, db_datareader, db_datawriter, db_securityadmin, and bulkadmin. You copy the script and execute it in SSMS or Azure Data Studio:

03 Grant necessary privileges

Specify the database and schema to be synced below,

Database

source-data

Schema

dbo

Grant the privileges for the database and schema by running the following queries,

```
USE source-data
GO
CREATE USER integrateio FOR LOGIN integrateio WITH DEFAULT_SCHEMA=dbo
GO
CREATE SCHEMA dbo AUTHORIZATION integrateio
GO
EXEC sp_addrolemember 'db_ddladmin', 'integrateio';
EXEC sp_addrolemember 'db_datareader', 'integrateio';
EXEC sp_addrolemember 'db_datawriter', 'integrateio';
EXEC sp_addrolemember 'db_securityadmin', 'integrateio';
EXEC sp_addsrvrolemember 'integrateio', 'bulkadmin';
GO
```

Copy

Continue

Note the last stored procedure call may fail as it appears `bulkadmin` does not exist in some Azure SQL versions. You may have to double check the documentation. In any case, this authorization level is not required for this scenario.

Enable CDC for the database

Next, we must enable the change data capture functionality. Just like previously copy the suggested script and run it in SSMS or Azure Data Studio:

04 Enable CDC for the database

1. Go to Azure dashboard
2. Enable change data capture by running,

```
EXEC sys.rds_cdc_enable_db  
GO
```

Copy

Continue

Note that running this statement will cause an error if your Azure SQL database is on the Free, Basic or Standard Single Database (S0,S1,S2) tier. To have CDC enabled you must have eDTUs > 100 or max vCore >= 1. The most budget-friendly option supporting CDC is Standard SE with 100 DTUs ([resource limits reference](#)). After running the stored procedure, I can also check to ensure CDC has been enabled:

```
15 EXEC sys.sp_cdc_enable_db  
16 GO  
17  
18 SELECT is_cdc_enabled  
19        ,name  
20 FROM sys.databases  
21
```

Results		Messages	
	is_cdc_enabled		name
1	0		master
2	1		source-data

Enable CDC for the table

The final step five that Integrate.io guides us to is to enable CDC on the table itself. Copy the script and run it:

05 Enable CDC for the needed tables

1. Go to Azure dashboard
2. Enable change data capture by running,

```
EXEC sys.sp_cdc_enable_table
@source_schema = N'MySchema',
@source_name   = N'MyTable',
@role_name     = N'integrateio',
@supports_net_changes = 1
GO
```

Copy

With these steps done, we are ready to go to the next step: configure the source connection.

Configure the source

Source details

We are now coming to the end of the configuration process for the source. After configuring our database and table now we need to ensure the connections work. The process starts with giving the source a name:

[« Select an existing source](#)

Configure new Microsoft SQL Server CDC source

01 Source Details

Source Name *

Continue

Source credentials

Then fill in the connection credentials that Integrate.io generated for us previously (or the ones you prefer):

02 Source Credentials

Host *	Port *
<input type="text" value="...database.windows.net"/>	<input type="text" value="1433"/>
Database User *	Database Password *
<input type="text" value="integrateio"/>	<input type="password" value="*****"/>
Database *	Schema *
<input type="text" value="source-data"/>	<input type="text" value="dbo"/>

03 Connection options

04 Custom SSL options

Connection options

Clicking *Continue* after filling in the source credentials will open the connection options section:

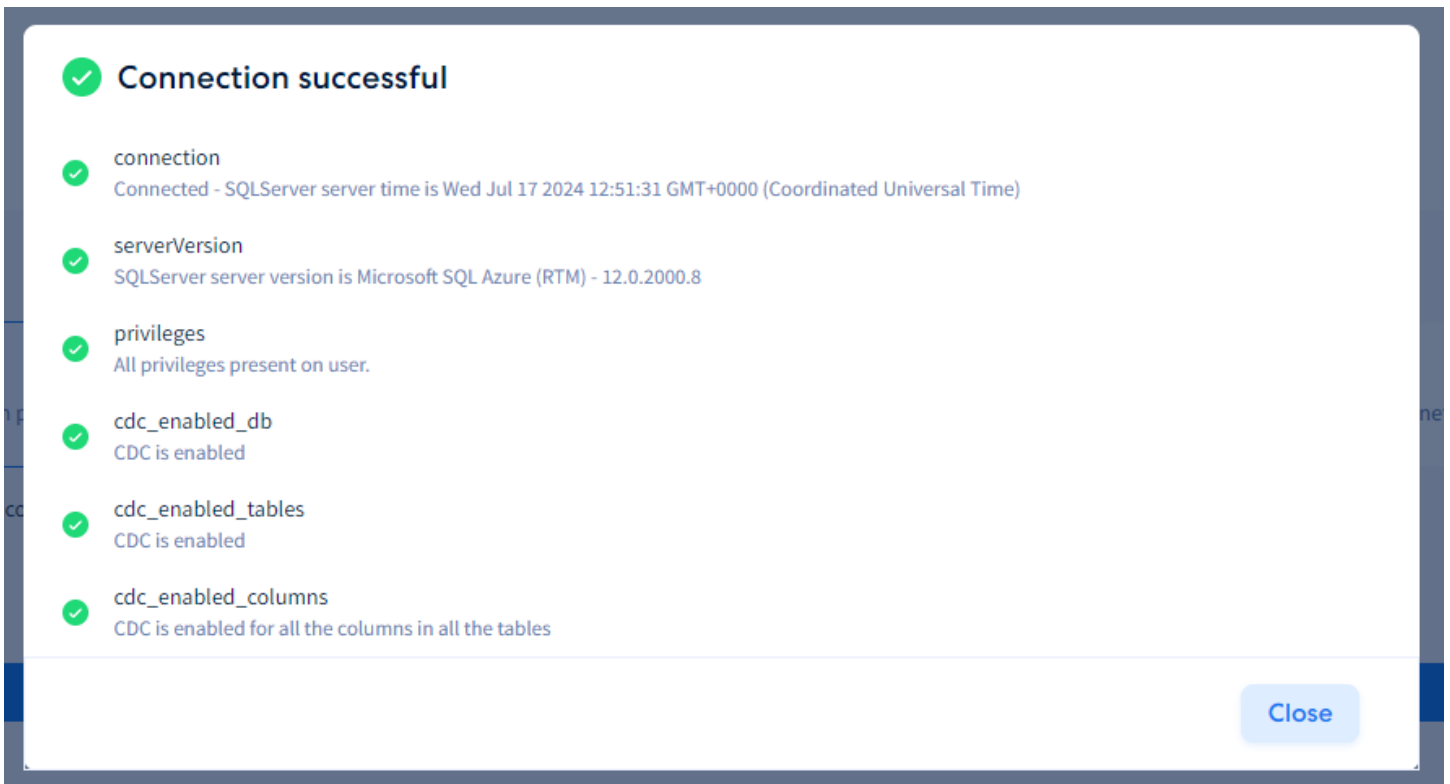
03 Connection options

Connect directly Choose if your database is available on public address.	Connect via secure tunnel Choose if your source is NOT publicly accessible or in a private network.
--	---

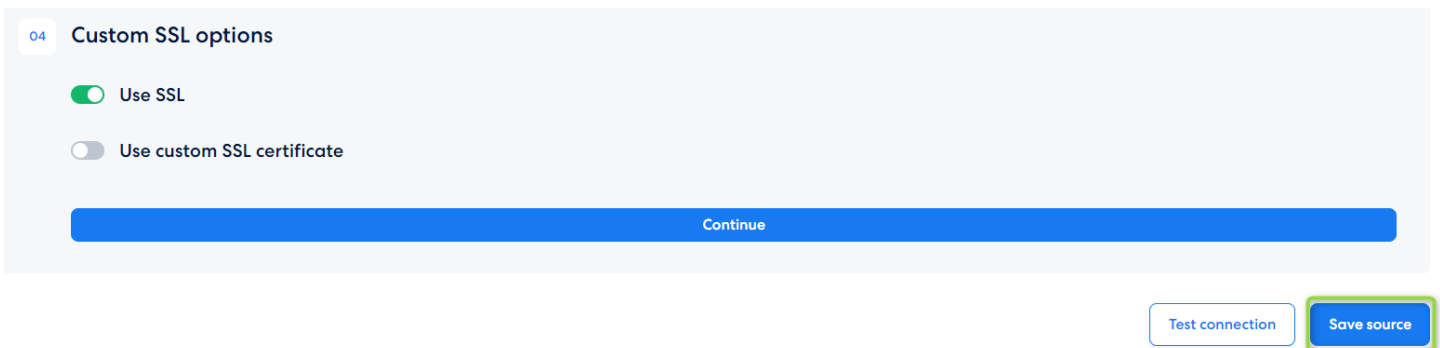
Please whitelist the following IPs to allow connections to your source.

04 Custom SSL options

In the direct connection case, you must whitelist the IPs in the MSSQL server firewall. After having the IPs whitelisted, click *Test Connection*. The expected result is a dialog with a summary stating all prerequisites have been fulfilled:



Finally, check the SSL options and click *Save source*:



At this point our source configuration is ready! Integrate.io has guided as in a well-structured and thoroughly documented process to configure our CDC-enabled Azure SQL data source. In case new tables appear, they will be automatically detected and synchronized as well.


With our source connection ready, let us proceed with setting up the destination data connection.


Set up the destination

For this scenario, I will be using a Snowflake data warehouse destination. Snowflake is a powerful data-as-a-service (DaaS) platform that can efficiently store vast amounts of data. Using Integrate.io to create your data pipeline makes tapping into data stores quick and easy, even for non-technical users. After signing up for [Snowflake](#), you can proceed with setting up the connection from Integrate.io.

Select destination





First select the destination type:

<  Real-time Sales Data

 Configure Source > **02** Configure Destination > 03 Schema Settings > 04 Sync Settings

Select destination

Q Search for destination

 BigQuery  Amazon Redshift  Amazon S3  Snowflake

Overview

Note it is required that you are an admin to execute the setup steps:

[« Select an existing destination](#)

Setting up Snowflake for ELT & CDC

Before you begin to add Snowflake as destination, please make sure that it is configured to work with Integrate.io Sync by following the steps below.

01 Overview

ELT & CDC uses Snowflake compute resource in auto suspend and auto-resume mode to run queries to load data from an external stage. You have an option to either run a ELT & CDC dedicated compute warehouse and provide a database for ELT & CDC to load data or choose an existing warehouse and database to load.

Using an existing warehouse is recommended to minimise compute resources usage and cost for loading data from any source to Snowflake

Requirements

- Any one of the `ACCOUNTADMIN`, `SYSADMIN` or `SECURITYADMIN` access for running the prerequisite/setup steps.

Continue

User setup

Next, we must create an integration-level user that Integrate.io will use to write to the data warehouse:

02 User setup

Provide the username and password for this connection.

Username	Password	Warehouse
<input type="text" value="integrateio"/>	<input type="password" value="....."/>	<input type="text" value="COMPUTE_WH"/>

If the user doesn't exist, run the below script to create the user

```
-- Create user
USE ROLE SECURITYADMIN;

CREATE USER IF NOT EXISTS integrateio
PASSWORD = .....
DEFAULT_WAREHOUSE = 'COMPUTE_WH';
```

[Copy](#)

[Continue](#)

Role setup

Next, we must set up the role. Specify name of the role, database and schema:

03 Role setup

Provide the role name, database and schema for this connection.

Role name

IIO_ROLE

Database

SALES_DW

Schema

SALES

Run the below script to create the role and grant warehouse, database and schema access.

```
-- Create role
USE ROLE SECURITYADMIN;
CREATE ROLE IF NOT EXISTS IIO_ROLE;

-- Grant role user and warehouse access
GRANT ROLE IIO_ROLE TO ROLE SYSADMIN;
GRANT ROLE IIO_ROLE TO USER integrateio;
GRANT USAGE ON WAREHOUSE SALES_DW TO ROLE IIO_ROLE;

-- Create the database and schema if not exist.
-- Snowflake advice not to use ACCOUNTADMIN when creating objects.
-- If database exists, make sure it is under the SYSADMIN role.
USE ROLE SYSADMIN;
CREATE DATABASE IF NOT EXISTS SALES_DW;
CREATE SCHEMA IF NOT EXISTS SALES_DW.SALES;

-- Grant stage, database and schema access to role
USE ROLE ACCOUNTADMIN;
GRANT CREATE STAGE ON ALL SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
GRANT CREATE SCHEMA, MONITOR, USAGE ON DATABASE SALES_DW TO ROLE IIO_ROLE;
GRANT ALL ON ALL SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
GRANT ALL ON FUTURE SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
```

Copy

Continue

Here is the raw script:

```
-- Create role
USE ROLE SECURITYADMIN;
CREATE ROLE IF NOT EXISTS IIO_ROLE;

-- Grant role user and warehouse access
GRANT ROLE IIO_ROLE TO ROLE SYSADMIN;
GRANT ROLE IIO_ROLE TO USER integrateio;
GRANT USAGE ON WAREHOUSE COMPUTE_WH TO ROLE IIO_ROLE;

-- Create the database and schema if not exist.
-- Snowflake advice not to use ACCOUNTADMIN when creating objects.
-- If database exists, make sure it is under the SYSADMIN role.
USE ROLE SYSADMIN;
CREATE DATABASE IF NOT EXISTS SALES_DW;
CREATE SCHEMA IF NOT EXISTS SALES_DW.SALES;

-- Grant stage, database and schema access to role
USE ROLE ACCOUNTADMIN;
GRANT CREATE STAGE ON ALL SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
GRANT CREATE SCHEMA, MONITOR, USAGE ON DATABASE SALES_DW TO ROLE IIO_ROLE;
GRANT ALL ON ALL SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
GRANT ALL ON FUTURE SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
```

To execute this statement successfully, ensure you have the default COMPUTE_WH selected:

```
1  -- Create role
2  USE ROLE SECURITYADMIN;
3  CREATE ROLE IF NOT EXISTS IIO_ROLE;
4
5  -- Grant role user and warehouse access
6  GRANT ROLE IIO_ROLE TO ROLE SYSADMIN;
7  GRANT ROLE IIO_ROLE TO USER integrateio;
8  GRANT USAGE ON WAREHOUSE COMPUTE_WH TO ROLE IIO_ROLE;
9
10 -- Create the database and schema if not exist.
11 -- Snowflake advice not to use ACCOUNTADMIN when creating objects.
12 -- If database exists, make sure it is under the SYSADMIN role.
13 USE ROLE SYSADMIN;
14 CREATE DATABASE IF NOT EXISTS SALES_DW;
15 CREATE SCHEMA IF NOT EXISTS SALES_DW.SALES;
16
17 -- Grant stage, database and schema access to role
18 USE ROLE ACCOUNTADMIN;
19 GRANT CREATE STAGE ON ALL SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
20 GRANT CREATE SCHEMA, MONITOR, USAGE ON DATABASE SALES_DW TO ROLE IIO_ROLE;
21 GRANT ALL ON ALL SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
22 GRANT ALL ON FUTURE SCHEMAS IN DATABASE SALES_DW TO ROLE IIO_ROLE;
23
```

status
1 Statement executed successfully.

Query Details

Query duration 635ms

Rows 1

Query ID 01b5baed-0304-2e96-0...

Show more

status

100% filled

Storage integration setup

The final fourth step is the storage integration setup:

04 Storage integration setup

Create storage integration for Integrate.io.

Storage Integration Name

IIO_STORAGE_INTEGRATION_DJPGRCCLD

Run the below script to create the role and grant warehouse, database and schema access.

```
-- Create storage integration
USE ROLE ACCOUNTADMIN;
CREATE STORAGE INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCCLD
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE

-- These will be updated by Integrate.io upon test connection
STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::0000:role/dummy-role'
STORAGE_ALLOWED_LOCATIONS = ('s3://dummy-bucket');

-- Grant storage integration access to role
GRANT CREATE INTEGRATION ON ACCOUNT TO ROLE IIO_ROLE;
GRANT OWNERSHIP ON INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCCLD TO ROLE IIO_ROLE;
GRANT USAGE ON INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCCLD TO ROLE IIO_ROLE;
```

Copy

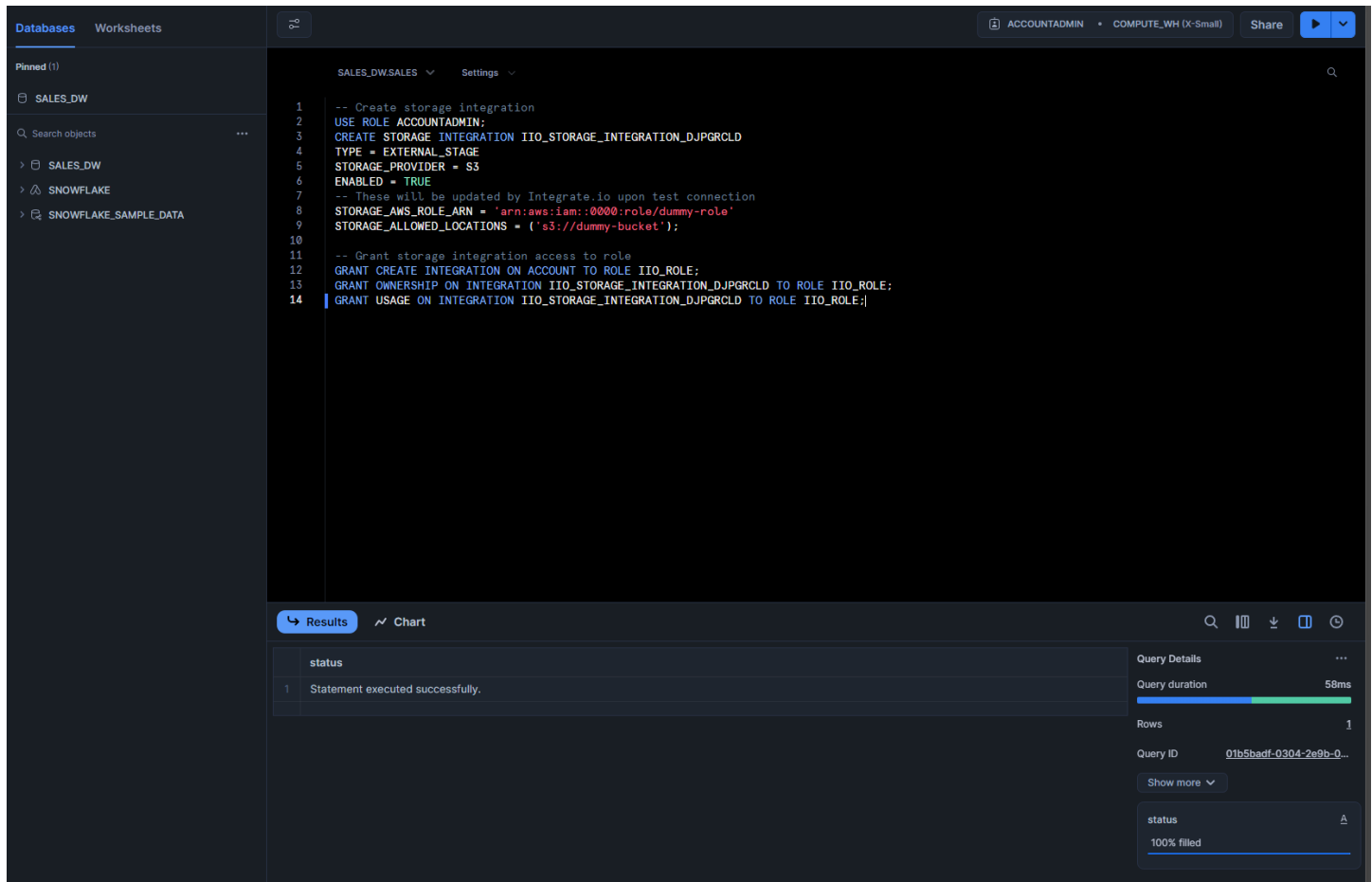
Continue

```
-- Create storage integration
USE ROLE ACCOUNTADMIN;
CREATE STORAGE INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCCLD
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE

-- These will be updated by Integrate.io upon test connection
STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::0000:role/dummy-role'
STORAGE_ALLOWED_LOCATIONS = ('s3://dummy-bucket');

-- Grant storage integration access to role
GRANT CREATE INTEGRATION ON ACCOUNT TO ROLE IIO_ROLE;
GRANT OWNERSHIP ON INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCCLD TO ROLE IIO_ROLE;
GRANT USAGE ON INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCCLD TO ROLE IIO_ROLE;
```

This is how it looks like executed in Snowflake:



The screenshot shows the Snowflake SQL Editor interface. The top navigation bar includes 'Databases' and 'Worksheets'. The left sidebar shows a tree view with 'Pinned (1)', 'SALES_DW', and search options. The main editor area contains the following SQL code:

```
1 -- Create storage integration
2 USE ROLE ACCOUNTADMIN;
3 CREATE STORAGE INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCLD
4 TYPE = EXTERNAL_STAGE
5 STORAGE_PROVIDER = S3
6 ENABLED = TRUE
7 -- These will be updated by Integrate.io upon test connection
8 STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::0000:role/dummy-role'
9 STORAGE_ALLOWED_LOCATIONS = ('s3://dummy-bucket');
10
11 -- Grant storage integration access to role
12 GRANT CREATE INTEGRATION ON ACCOUNT TO ROLE IIO_ROLE;
13 GRANT OWNERSHIP ON INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCLD TO ROLE IIO_ROLE;
14 GRANT USAGE ON INTEGRATION IIO_STORAGE_INTEGRATION_DJPGRCLD TO ROLE IIO_ROLE;
```

The bottom panel shows the 'Results' tab with a single row: 'Statement executed successfully.' The 'Query Details' panel on the right shows 'Query duration: 58ms', 'Rows: 1', and 'Query ID: 01b5badf-0304-2e9b-0...'. A progress bar indicates '100% filled'.

This important configuration step is required because under the covers, Integrate.io uses Amazon S3 storage as a buffer between the source and the destination. From Azure SQL Integrate.io will serialize the data to an avro format and store it in S3, which is a setup particularly suitable for streaming analytics, data replication and big data processing scenarios like this one. Integrate.io will assign the value for `STORAGE_AWS_ROLE_ARN` and `STORAGE_ALLOWED_LOCATIONS` at run-time so there is nothing extra the user must do to make the pipeline work.

Applying transformations

There is also an option to apply certain transformations on the source data before it reaches the destination. Specifically, Integrate.io supports replacement of values in a column. To apply this rule, you must contact the support team and provide a Regex pattern for the string to find and a replacement string. Other transformations are not supported due to following the best practices of the typical architecture of data replication to a data warehouse. In such an architecture, more extensive transformations are implemented in the destination only after the data arrives, e.g. when creating a curated data set out of a staged data set.

With these steps done, we are ready to go to the next step: configure the destination connection.

Configure the destination

Destination details

Integrate.io is now guiding us through the final configuration steps for the destination. We must provide a name for the destination:

01 Configure Source > 02 Configure Destination > 03 Schema Settings > 04 Sync Settings

[← Select an existing destination](#)

Configure new Snowflake destination

01 Destination Details

Destination Name *

sales-dw

Continue

02 Destination Credentials

Test connection

Save destination

Destination credentials

Fill in the fields with the connection attributes that we now have, such as username, password, role name, storage integration name, database, and schema:

02 Destination Credentials

Snowflake Account Identifier *

.west-europe.azure

You can find this in your account URL. For example, if zx73881.us-east-2.aws.snowflakecomputing.com is the URL, then zx73881.us-east-2.aws is the account identifier

Warehouse *

COMPUTE_WH

Username *

integrateio

Password *

.....

User Role *

IIO_ROLE

Storage Integration Name *

IIO_STORAGE_INTEGRATION_DJPGRCLD

Database *

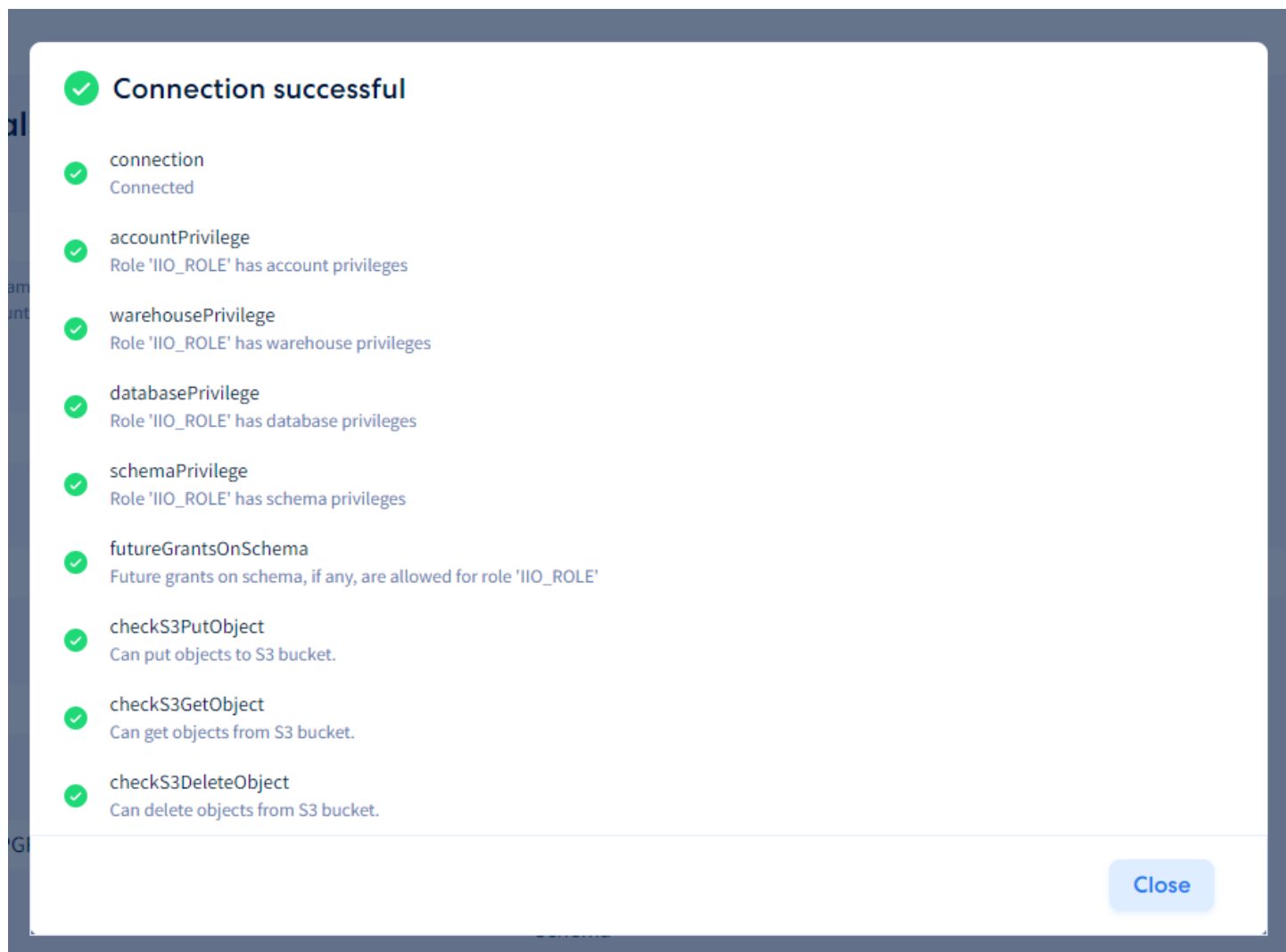
SALES_DW

Schema *

SALES

Continue

Testing the connection should display the following pop-up:



Configure the pipeline

At this point we have successfully configured the source and the destination as well as tested the connectivity. We are now at the schema settings stage. Integrate.io gives us a list of the available tables for syncing:

Comma separated list of tables Select

[Refresh table list](#)

<input checked="" type="checkbox"/> Select All	Table Name	Mode	<input type="checkbox"/> Append only	Modify Columns
<input checked="" type="checkbox"/>	dbo.LiveOrdersData	Incremental Sync	<input type="checkbox"/>	Select columns to exclude
<input type="checkbox"/>	dbo.systranschemas	Incremental Sync		
<input type="checkbox"/>	dbo.IntVect	Incremental Sync		

Showing 1 to 3 of 3 results.

Continue to Sync Settings

Here I will just select the table I need without applying any additional settings: it is that easy to get started with real-time CDC! We have the option to “append only” to avoid having deletes replicate which is not relevant in this scenario. From this interface we can also modify the columns that we want to exclude from syncing. Here I will exclude the ContactPersonID:

Search and add columns to exclude from syncing

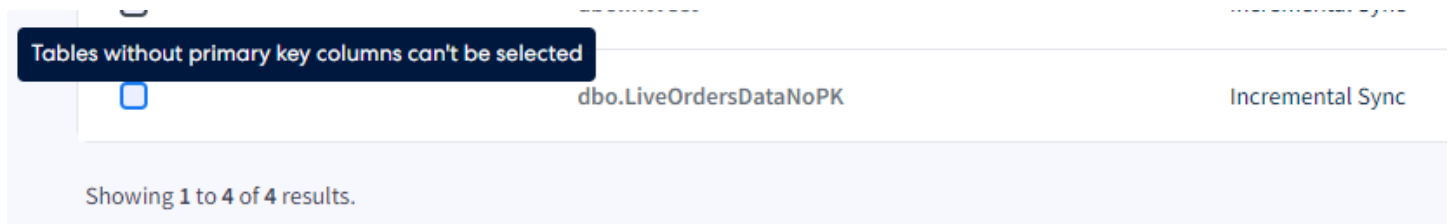
Search

ContactPersonID

Cancel Save

Continue to Sync Settings

You will notice the column selection dropdown menu will not contain the primary key column. Therefore, in the table list also tables without a primary key will not be available for selection for CDC replication. Tables that do have a unique index, but no primary key will be blanked out as well, although CDC may be enabled for them:



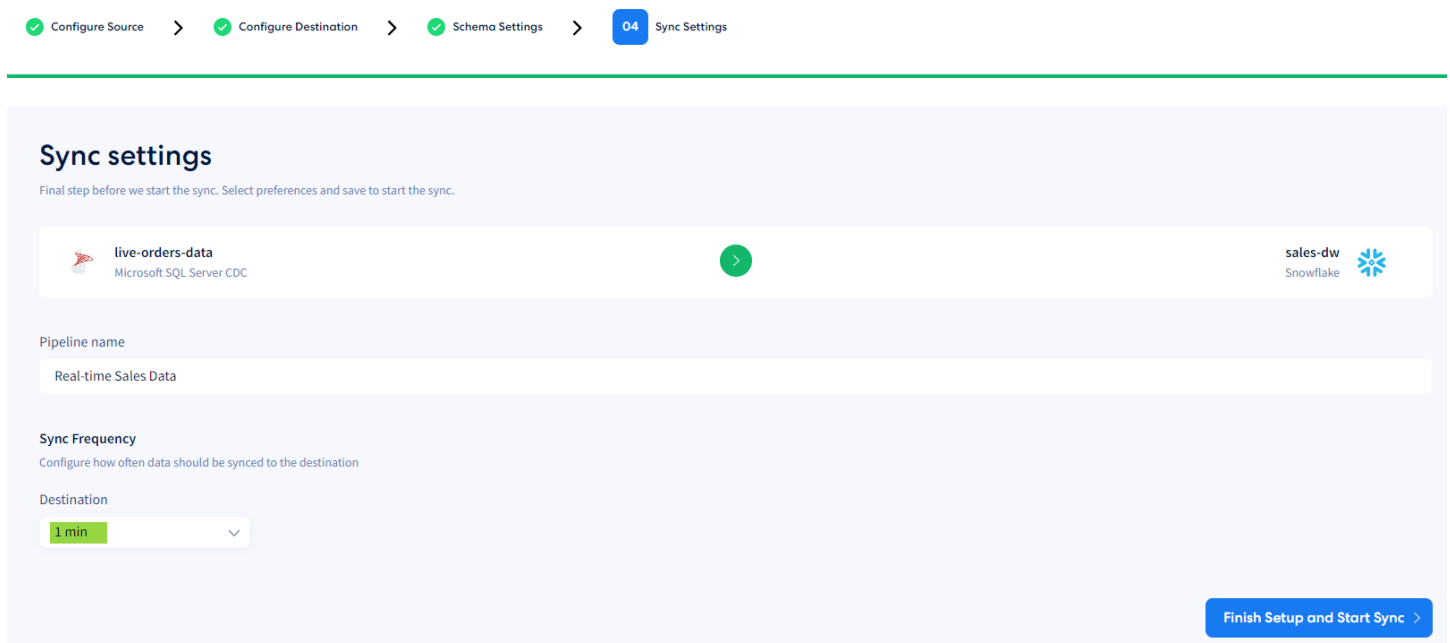
Tables without primary key columns can't be selected

<input type="checkbox"/>	dbo.LiveOrdersDataNoPK	Incremental Sync
--------------------------	------------------------	------------------

Showing 1 to 4 of 4 results.

Sync settings

The final stage of the pipeline setup is the sync frequency setting. To demonstrate the capabilities of the platform we will use the highest frequency of one minute. This setting will make the pipeline run and check for changes every minute.



Configure Source > Configure Destination > Schema Settings > 04 Sync Settings

Sync settings

Final step before we start the sync. Select preferences and save to start the sync.

live-orders-data
Microsoft SQL Server CDC

sales-dw
Snowflake

Pipeline name
Real-time Sales Data

Sync Frequency
Configure how often data should be synced to the destination

Destination
1 min

Finish Setup and Start Sync >

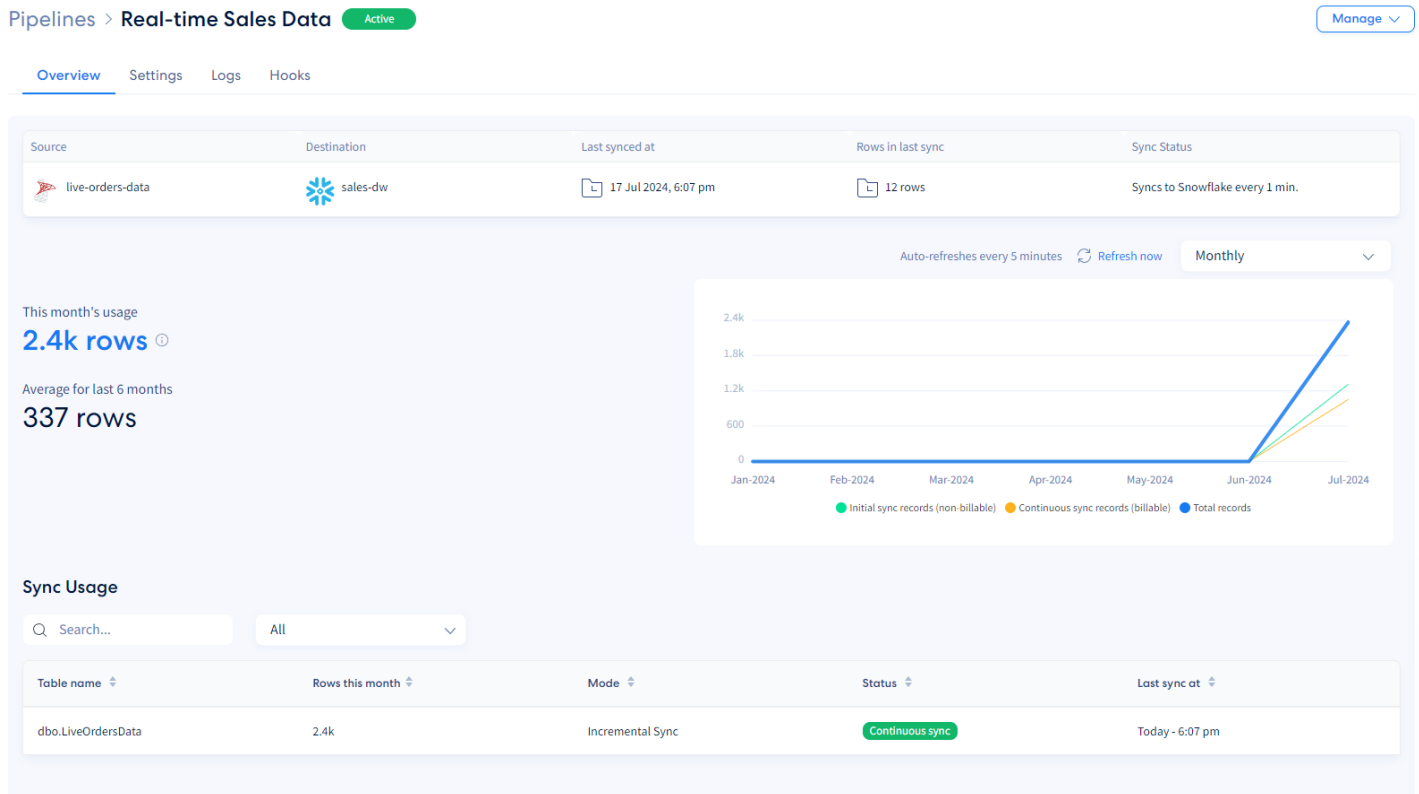
Once done, click on *Finish Setup and Start Sync*. The initial sync will commence:



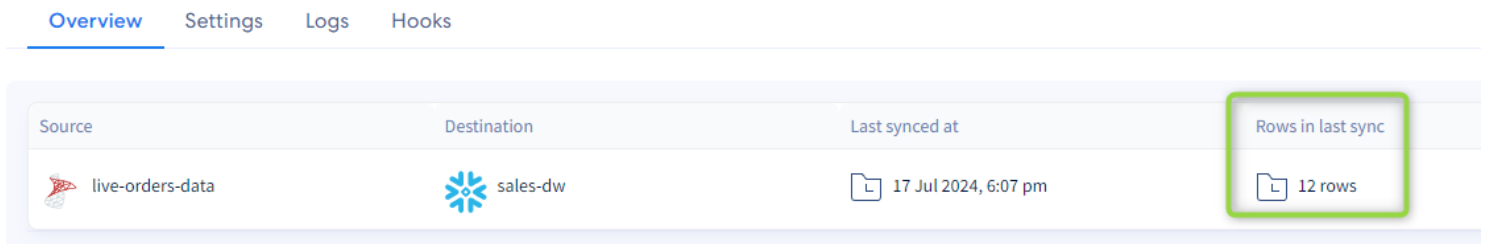
Starting initial sync...

Pipeline created. Redirecting to pipeline dashboard shortly.

After the initial sync completes, the pipeline will execute every minute to sync only the new rows. Once the pipeline is running, we get access to a handy overview. Here we see the main pipeline metrics such as last sync and number of rows that were synced:



Since we generate new data every 5 seconds, we should be getting about 12 new rows every minute ($60 / 5 = 12$):

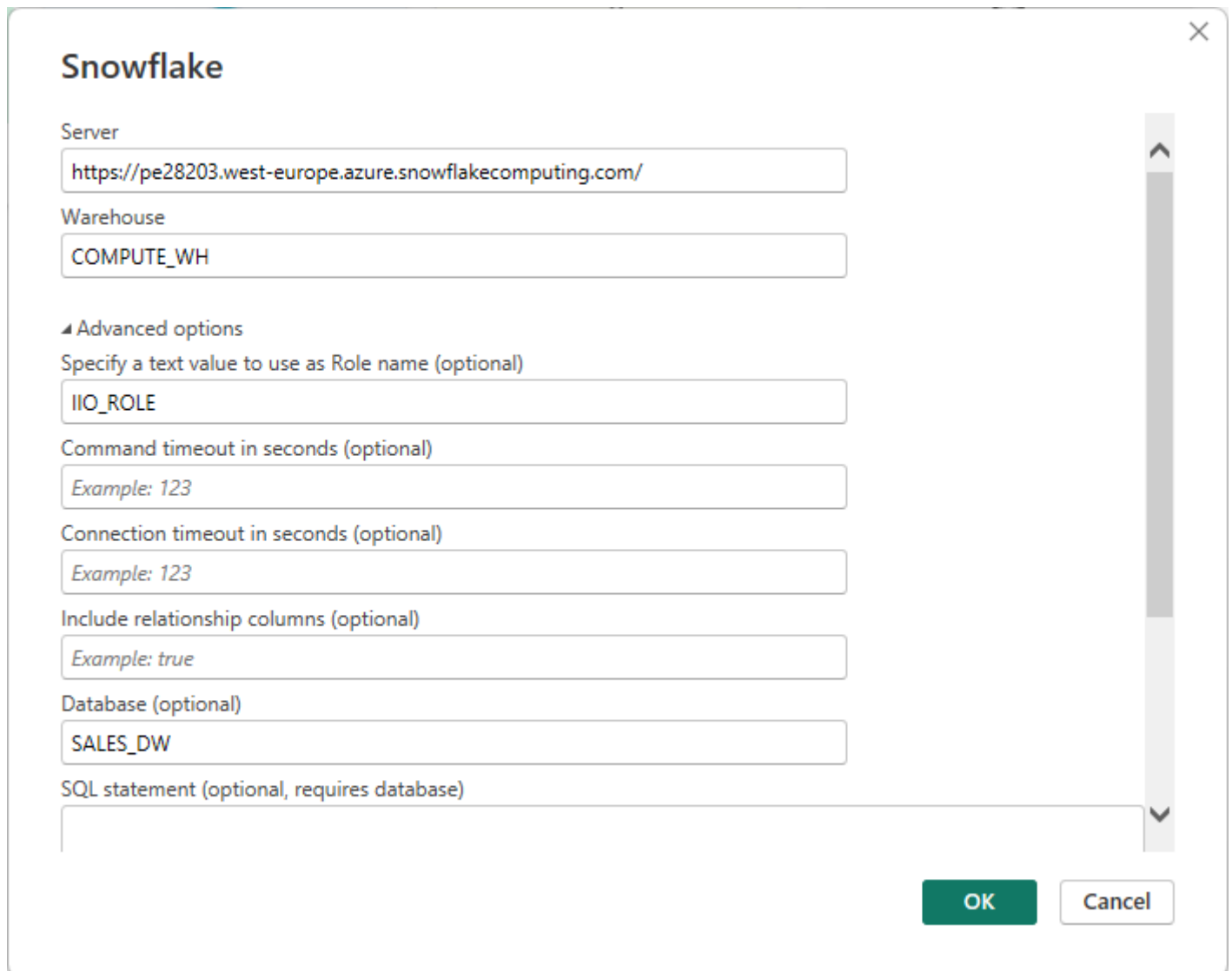


Leverage the data

Having the one-minute pipeline all set and running, Integrate.io will now have the data available in Snowflake either for analytics or other scenarios. For example, application integration, fueling other line of business apps with real-time data or building curated data products. For the purposes of this demo, let us just examine the data by building a Power BI report.

Power BI data source

To start, configure the Snowflake data source in Power BI and login with the previously configured user as suggested by Integrate.io. Next, from the familiar Power BI data navigator I can pick the table I need:

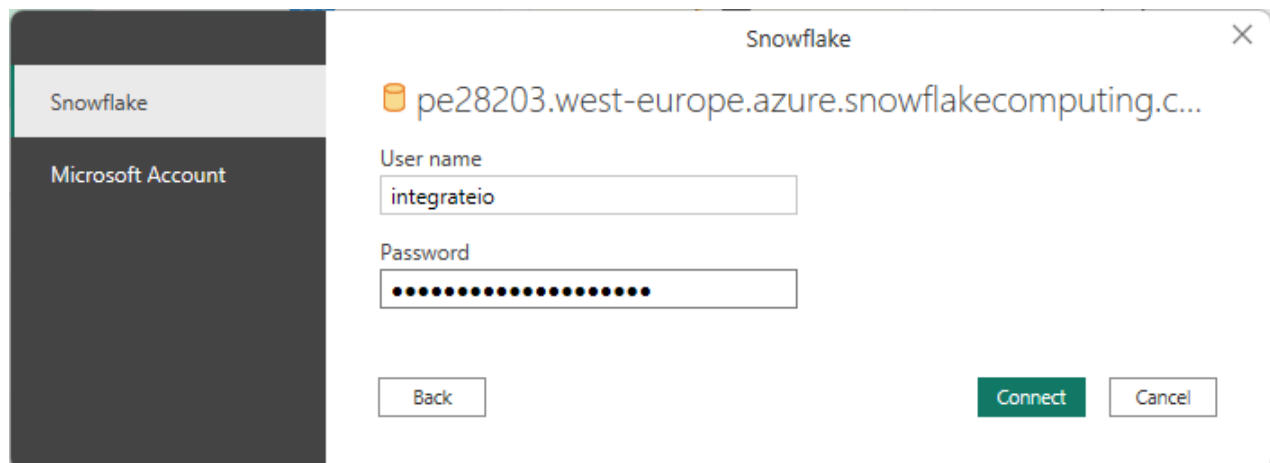


The screenshot shows a 'Snowflake' configuration dialog box. It contains several input fields for configuration:

- Server:** `https://pe28203.west-europe.azure.snowflakecomputing.com/`
- Warehouse:** `COMPUTE_WH`
- Advanced options:**
 - Specify a text value to use as Role name (optional):** `IIO_ROLE`
 - Command timeout in seconds (optional):** `Example: 123`
 - Connection timeout in seconds (optional):** `Example: 123`
 - Include relationship columns (optional):** `Example: true`
 - Database (optional):** `SALES_DW`
 - SQL statement (optional, requires database):** (Empty field)

At the bottom right, there are two buttons: **OK** (green) and **Cancel** (white).

Then we can log in with the previously configured user as suggested by Integrate.io:



The screenshot shows a 'Snowflake' login dialog box. It features a sidebar on the left with 'Snowflake' and 'Microsoft Account' options. The main area displays the server name and login fields:

- Server:** `pe28203.west-europe.azure.snowflakecomputing.c...`
- User name:** `integrateio`
- Password:** (Masked with dots)

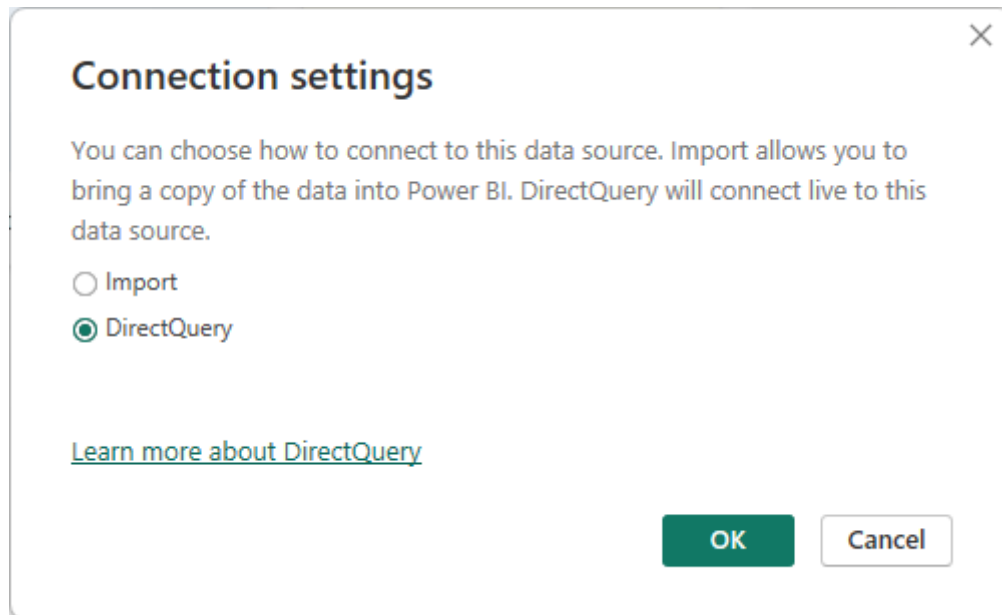
At the bottom, there are three buttons: **Back** (white), **Connect** (green), and **Cancel** (white).

Next, from the familiar Power BI data navigator I can pick the table I need:

The screenshot shows the Power BI Navigator interface. On the left, a tree view shows the hierarchy: 'pe28203.west-europe.azure.snowflakecomputing...' > 'SALES_DW [2]' > 'PUBLIC' > 'SALES [1]' > 'LiveOrdersData'. The 'LiveOrdersData' table is selected. On the right, the table data is displayed in a grid view with columns: SaleID, OrderID, CustomerID, SalespersonPersonID, and ContactPersonID. The data rows are numbered 1 through 23. At the bottom of the window, there are buttons for 'Select Related Tables', 'Load', 'Transform Data', and 'Cancel'.

SaleID	OrderID	CustomerID	SalespersonPersonID	ContactPersonID
1	1	832		2
2	2	803		8
3	2	803		8
4	3	105		7
5	4	57		16
6	4	57		16
7	4	57		16
8	5	905		3
9	5	905		3
10	5	905		3
11	6	976		13
12	6	976		13
13	6	976		13
14	7	575		8
15	7	575		8
16	7	575		8
17	7	575		8
18	8	964		7
19	8	964		7
20	8	964		7
21	9	77		7
22	9	77		7
23	10	191		20

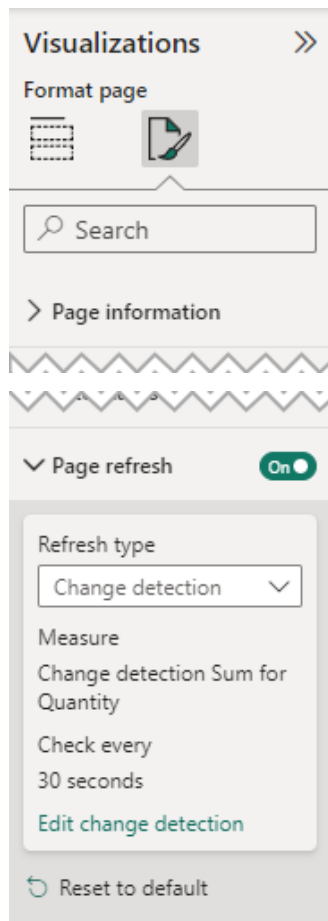
Importantly, we will utilize the DirectQuery connection mode to showcase the data arriving every minute:



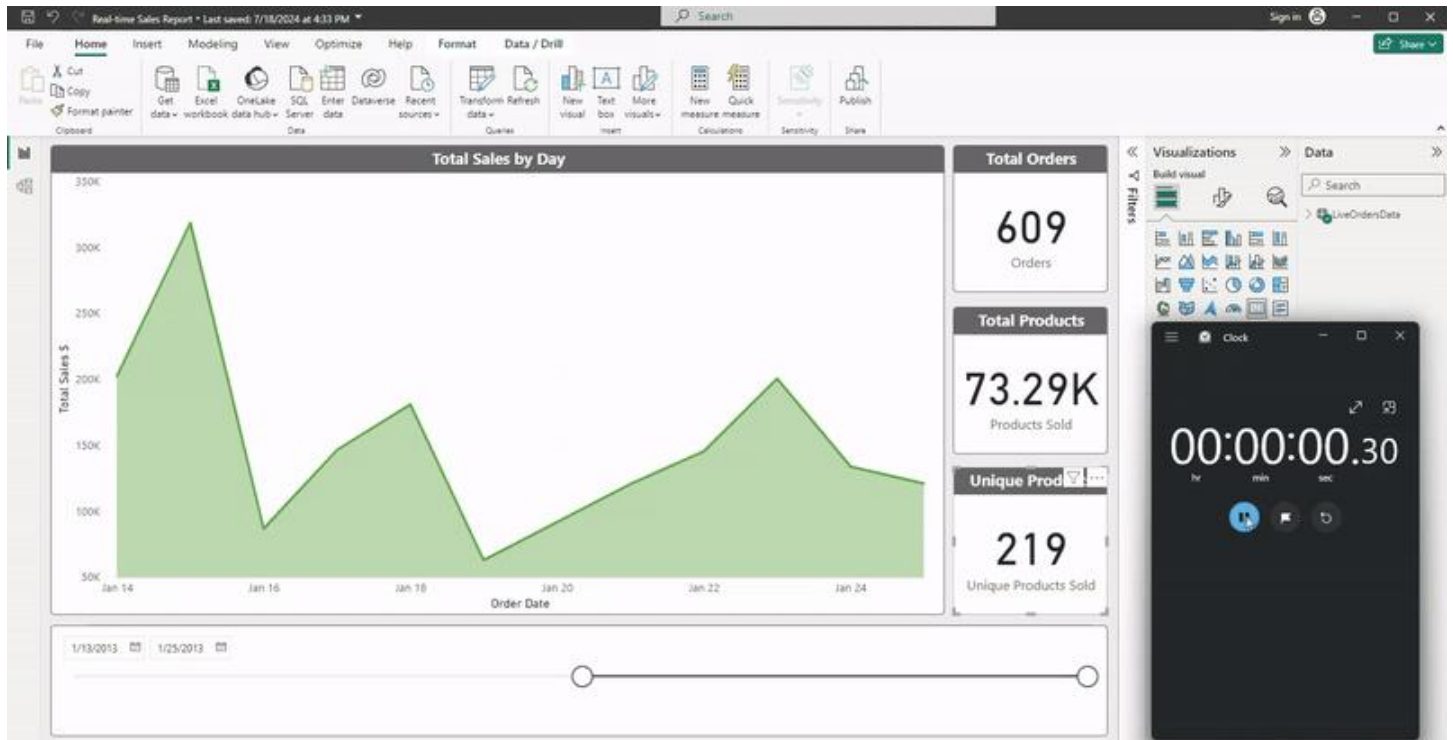
Next, click on the blank report canvas, open the Visualizations pane, and expand the Page refresh tab. Set it to *On* and configure it like this:

- Change detection of the total sum for Quantity (i.e., overall number of products sold)
- Check every thirty seconds.

This configuration instructs Power BI to refresh the page every 30 seconds so new data is immediately reflected on the report visuals as Integrate.io continuously syncs to Snowflake:



We can observe the result of the pipeline run in the following animated screenshot. Notice how the total sales quantities and sum of total sales in USD evolve as the new data come in (speed has been increased 3x):



At this point we have the pipeline running and we utilize the incoming live data in a Power BI report. There is one last point left: monitoring the pipeline to ensure the CDC runs smoothly.

Monitor and re-configuring the pipeline

Once the pipeline is saved and starts running, we can monitor its execution. From the list of pipelines, you can access your pipeline:

Welcome, Hristo Hristov! ELT & CDC

Pipelines

[+ New Pipeline](#)

1 Pipeline

Search: 20

Pipeline Name	Source	Destination	Last Sync	Rows this month	Status
Real-time Sales Data	live-orders-data	sales-dw	Today - 2:28 pm	5.3k rows	Active

Showing 1 to 1 of 1 results.

In the pipeline overview you have four tabs: Overview, Settings, Logs and Hooks:

Overview Settings Logs Hooks

Source	Destination	Last synced at	Rows in last sync	Sync Status
live-orders-data	sales-dw	22 Jul 2024, 2:29 pm	—	Syncs to Snowflake every 1 min.

Auto-refreshes every 5 minutes Refresh now Daily

This day's usage
5.3k rows

Average for last 6 days
762 rows

Sync Usage

Search... All

Table name	Rows this day	Mode	Status	Last sync at
dbo.LiveOrdersData	5.3k	Incremental Sync	Continuous sync	Last Thursday - 4:34 pm

The overview provides a summary of how many rows have been synced on daily, weekly, monthly or yearly basis. You can see the last sync, the rows synced during the last sync session and which tables are being synced. From the settings tab you can change the sync frequency and select more tables for syncing:

Sync Settings

Sync Frequency

Configure how often data should be synced to the destination

Destination

1 min ▾

Reset to default

Save changes

Configuration

Select tables and columns to sync

All columns are synced by default. Click on a table name to select specific columns to exclude. [Learn more](#)

Comma separated list of tables Select

Refresh table list

<input checked="" type="checkbox"/> Select All	Table Name	Mode	<input type="checkbox"/> Append only	Modify Columns
<input checked="" type="checkbox"/>	dbo.LiveOrdersData	Incremental Sync	<input type="checkbox"/>	Select columns to exclude
<input type="checkbox"/>	dbo.systranschemas	Incremental Sync		
<input type="checkbox"/>	dbo.IntVect	Incremental Sync		

Showing 1 to 3 of 3 results.

From the Logs tab we can access some high-level logs describing the execution state of the pipeline. In case there are any errors with the pipeline execution, they will pop up here:

20 ▾

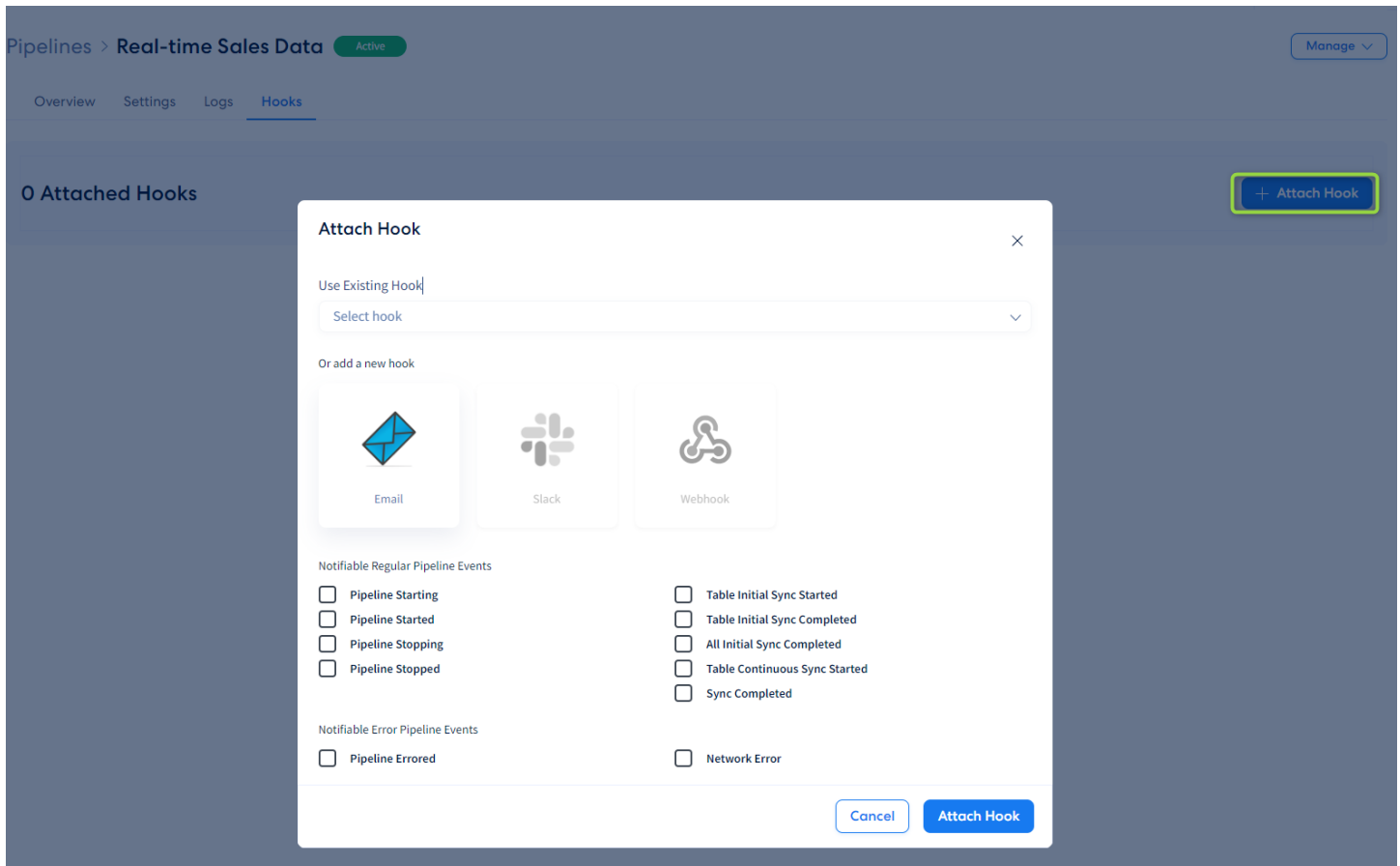
Created at	Action
Today at 01:17 pm	Pipeline Started
Today at 01:17 pm	Continuous Sync Started
Today at 01:17 pm	Pipeline Starting
Today at 01:16 pm	Pipeline Started
Today at 01:16 pm	Continuous Sync Started

```

{
  id:"51822290",
  created_at:"2024-07-22T11:17:42.338584+00:00",
  action:"pipeline_started",
  message:"Pipeline started",
  data:{
    extra_request_data:{
      pipeline:"pl-8521-QGuvzjQLrj6ji3swGPsYoA",
      host_name:"b366d2dd56a2",
      created_at:"2024-07-22T11:17:42.062Z",
      source_name:"pl-8521-QGuvzjQLrj6ji3swGPsYoA",
      source_type:"pipeline"
    }
  }
}

```

Finally, from the Hooks tab you can create a hook, or a messaging automation related to pipeline events. Click on *Attach Hook* and pick a preference for messaging:



With the messaging automation, Integrate.io ensures you remain updated at every data movement and pipeline event, including if any errors occur.

Conclusion

With Integrate.io we created a real-time data replication and CDC pipeline in a straightforward and reliable way. We were able to quickly expose transactional data to a cloud data warehouse so our business users can observe the sales evolution in real-time. Using the clear, step-by-step instructions from the ELT & CDC wizard, we can develop a scalable and secure data replication pipeline facilitating immediate data insights with little effort. This is how you can bring visible business value to your scenario. Interested in finding out more?

Head over to [Integrate.io's trial page](#) to sign up now.

Next Steps

- [Integrate.io ELT and CDC knowledge base](#)
- [Transforms – replacement rules](#)
- [Azure SQL CDC](#)
- [Automatic page refresh in Power BI](#)