

Data platforms: Snowflake, BigQuery and Databricks

From the SourceLace manual: How to connect Snowflake, Google BigQuery and Databricks for read-only SQL. Each person signs in with their own login, so the platform's own permissions apply to every query and schema lookup.

SourceLace version 0.1.0 · Made October 4, 2026

The latest version is at <https://sourcelace.onrender.com/docs>

Data platforms: Snowflake, BigQuery and Databricks

How to connect Snowflake, Google BigQuery and Databricks for read-only SQL. Each person signs in with their own login, so the platform's own permissions apply to every query and schema lookup.

Status: in testing. Built against each vendor's documented REST APIs and covered by automated tests against simulated services; not yet used with a customer's live account.

What people can do

- `search_schema` lists tables and views the person can see; `describe_object` lists a table's columns and types.
- `query` (language `sql`) takes **one** `SELECT` or `WITH ... SELECT` statement in the platform's own SQL. Anything that could change data (`INSERT`, `UPDATE`, `DELETE`, `MERGE`, `CREATE`, `DROP`, `ALTER`, `SELECT ... INTO`, `CALL` and so on), and anything with more than one statement, is refused **before** it is sent. A few functions that reach outside SQL are refused too (Snowflake `SYSTEM$...` functions, Databricks `reflect`).
- Results are capped at 2,000 rows (50 shown at a time), and SourceLace stops reading once it has enough. Results are held in memory for 30 minutes and never written to a database.
- Every query has a time limit of about a minute. A query that runs longer is cancelled on the platform, and the person is asked to narrow it.
- These sources never accept changes. `get_record` is not used: warehouse tables have no record ids, so people query with a `WHERE` clause.
- SourceLace never uses a shared service account for these sources.

The redirect URL for all three is:

<https://sourcelace.onrender.com/connect/callback>

Snowflake

Who sets it up: your Snowflake admin (someone with `ACCOUNTADMIN`, or a role with `CREATE INTEGRATION`), once per Snowflake account.

1. Create the security integration (Snowflake admin)

In a Snowflake worksheet, run:

```
USE ROLE ACCOUNTADMIN;

CREATE SECURITY INTEGRATION SOURCELACE
  TYPE = OAUTH
  ENABLED = TRUE
  OAUTH_CLIENT = CUSTOM
  OAUTH_CLIENT_TYPE = 'PUBLIC'
  OAUTH_REDIRECT_URI = 'https://sourcelace.onrender.com/connect/callback'
  OAUTH_ENFORCE_PKCE = TRUE
  OAUTH_ISSUE_REFRESH_TOKENS = TRUE
  OAUTH_REFRESH_TOKEN_VALIDITY = 7776000; -- 90 days, then people sign in again
```

Why these settings:

- `OAuth_Client_Type = 'PUBLIC'` with `OAuth_Enforce_Pkce = TRUE`: SourceLace proves each sign-in with PKCE instead of a client secret, so there is no Snowflake secret for anyone to copy or for SourceLace to store.
- `OAuth_Issue_Refresh_Tokens`: people stay connected until the refresh token expires, instead of signing in every 10 minutes.

Snowflake blocks ACCOUNTADMIN, ORGADMIN and SECURITYADMIN from OAuth sign-ins by default. Keep it that way: people should use their everyday roles.

If the account has a **network policy**, it must allow SourceLace's outbound IP addresses; ask support@sourcelace.com for the current list.

2. Copy the client id

```
SELECT SYSTEM$SHOW_OAUTH_CLIENT_SECRETS('SOURCELACE');
```

Copy only `OAuth_Client_ID` from the result. SourceLace does not need either client secret: leave them where they are.

3. Add the source (SourceLace admin)

Kind: **Snowflake** (snowflake).

Option	Type	Default	Example	What it is
account	Text	(required)	corvanta-xy12345.snowflakecomputing.com	The account URL. The account identifier alone (corvanta-xy12345) works too. Only *.snowflakecomputing.com addresses are accepted.
client_id	Text	(required)	AbCdEf123...	OAuth_Client_ID from step 2.
warehouse	Name	each person's default	REPORTING_WH	The warehouse queries run on.
role	Name	each person's default	ANALYST	If set, people sign in with this role only (Snowflake shows it on the consent screen).
database	Name	(none)	SALES	Default database for queries. With it, search_schema looks in that database; without it, across the account.
schema	Name	(none)	PUBLIC	Default schema for queries.

How SourceLace talks to Snowflake: sign-in at `https://<account>/oauth/authorize`; queries through Snowflake's SQL API with a 60-second statement timeout, `MULTI_STATEMENT_COUNT = 1` (Snowflake itself refuses a second statement), and the query tag `sourcelace`, so your Query History shows exactly what SourceLace ran and for whom.

Google BigQuery

Who sets it up: SourceLace provides the Google app people sign in with. Your Google Cloud admin gives people the roles below, and your SourceLace admin adds the source with your project id.

Sign-in asks only for `https://www.googleapis.com/auth/bigquery.readonly`, plus `openid` and `email` (to show who is connected). If your Google Workspace restricts third-party apps, allow SourceLace's app under **Google Admin console** → **Security** → **Access and data control** → **API controls**.

What each person needs in Google Cloud

- **BigQuery Job User** (`roles/bigquery.jobUser`) on the project in the project option, so they can run queries there. Queries are billed to that project.
- **BigQuery Data Viewer** (`roles/bigquery.dataViewer`) on the datasets they should read, as they would for the BigQuery console.

Add the source (SourceLace admin)

Kind: **Google BigQuery** (`bigquery`).

Option	Type	Default	Example	What it is
project	Text	(required)	corvanta-data	The Google Cloud project id that runs, and pays for, queries.
location	Text	(none)	US, EU, europe-west2	Where the data lives.
max_bytes_billed	Number of bytes	10000000000 (10 GB)	50000000000	The most one query may scan.

Cost protection: every query is first sent as a **dry run**, which is free. If it would scan more than `max_bytes_billed`, it is refused with the size it would have scanned, before anything is billed. The real query is then sent with `maximumBytesBilled` set, so BigQuery enforces the cap too, and with the label `source=sourcelace`. Tables are named `dataset.table` (or ``project.dataset.table``).

Databricks

Who sets it up: your Databricks **account** admin, once per Databricks account. Works on AWS (`*.cloud.databricks.com`), Azure (`*.azuredatabricks.net`) and Google Cloud (`*.gcp.databricks.com`).

1. Create the app connection (Databricks account admin)

1. Open the account console: `accounts.cloud.databricks.com` (AWS), `accounts.azuredatabricks.net` (Azure) or `accounts.gcp.databricks.com` (Google Cloud).
2. **Settings** → **App connections** → **Add connection**.
3. **Name:** SourceLace. **Redirect URLs:** the redirect URL above.
4. **Access scopes: All APIs.** SourceLace itself only calls the SQL statement and Unity Catalog read endpoints; each person's own grants still decide what they can see. Sign-in asks for `all-apis` and `offline_access`.

5. **Untick "Generate a client secret"**. SourceLace signs in as a public client with PKCE, so there is no secret to store.
6. Leave the token lifetimes as they are (or set the refresh token to 90 days). Click **Add**, then copy the **Client ID**.

2. Pick the SQL warehouse

In the workspace, **SQL Warehouses → (the warehouse) → Overview**: copy the warehouse **ID** (16 characters, such as 1234567890abcdef). Each person needs **Can use** on it, plus the usual Unity Catalog grants (USE CATALOG, USE SCHEMA, SELECT) on what they should read. A stopped serverless warehouse starts on the first query, which can take a few seconds.

3. Add the source (SourceLace admin)

Kind: **Databricks** (databricks).

Option	Type	Default	Example	What it is
host	Text	(required)	corvanta.cloud.databricks.com	The workspace address. Only Databricks addresses are accepted.
client_id	Text	(required)	1a2b3c4d-...	The app connection's Client ID from step 1.
warehouse_id	Text	(required)	1234567890abcdef	The SQL warehouse ID from step 2.
catalog	Name	(none)	main	Default catalog. With it, search_schema looks only in that catalog.
schema	Name	(none)	sales	Default schema.

Tables are named `catalog.schema.table`. A query still running after 50 seconds is cancelled rather than left running.

When something goes wrong

What you see	What to do
"Set the Snowflake client_id option: the OAUTH_CLIENT_ID of the account's SourceLace security integration"	Fill in <code>client_id</code> (step 2).
"The Snowflake account option must be a host name such as corvanta-xy12345.snowflakecomputing.com."	Fix account.
"The Snowflake warehouse option must be a plain name such as ANALYTICS." (or role, database, schema)	Use a plain name, without quotes or dots.
"Snowflake sign-in failed: ..."	Snowflake's own reason follows. Check the redirect URI in the security integration matches exactly, and that the person is not signing in with a blocked role such as ACCOUNTADMIN.
"Snowflake took too long to run this query, so it was stopped. Try a narrower query."	Narrow the query.

What you see	What to do
"Google did not grant read access to BigQuery. Connect again and tick the BigQuery box on Google's consent screen."	Connect again and tick the box.
"Set the BigQuery project option to a Google Cloud project id, such as corvanta-data."	Fill in project.
"This query would scan ..., more than this source's limit of Select fewer columns, filter on the partition column, or ask an admin to raise max_bytes_billed."	Narrow the query, or raise max_bytes_billed.
"BigQuery: Access Denied: ..."	The person lacks BigQuery Job User on the project or Data Viewer on the dataset.
"Set the Databricks client_id option ..." or "Set the Databricks warehouse_id option ..."	Fill in the option from the steps above.
"The Databricks host option must be a host name such as corvanta.cloud.databricks.com."	Fix host.
"Databricks took too long to run this query, so it was stopped. Try a narrower query, or check that the SQL warehouse is running."	Narrow the query, or start the warehouse.
"'...' is not allowed in a read-only ... query." or "SELECT ... INTO is not allowed in a read-only query."	Only plain reads are allowed.