Capture and view data lineage using Unity Catalog

You can use Unity Catalog to capture runtime data lineage across queries run on Azure Databricks. Lineage is supported for all languages and is captured down to the column level. Lineage data includes notebooks, workflows, and dashboards related to the query. Lineage can be visualized in Catalog Explorer in near real-time and retrieved with the Databricks REST API.

Note

You can also view and query lineage data using the lineage system tables (Public Preview).

Lineage is aggregated across all workspaces attached to a Unity Catalog metastore. This means that lineage captured in one workspace is visible in any other workspace sharing that metastore. Users must have the correct permissions to view the lineage data. Lineage data is retained for 1 year.

Requirements

The following are required to capture data lineage using Unity Catalog:

- The workspace must have Unity Catalog enabled.
- Tables must be registered in a Unity Catalog metastore.
- Queries must use the Spark DataFrame (for example, Spark SQL functions that return a DataFrame) or Databricks SQL interfaces
- To view the lineage of a table or view, users must have at least the BROWSE privilege on the table's or view's parent catalog.
- To view lineage information for notebooks, workflows, or dashboards, users must have permissions on these objects as defined by the access control settings in the workspace.
- To view lineage for a Unity Catalog-enabled pipeline, you must have CAN_VIEW permissions on the pipeline.
- You might need to update your outbound firewall rules to allow for connectivity to the Event Hub endpoint in the Azure Databricks control plane. Typically this applies if your Azure Databricks workspace is deployed in your own VNet (also known as VNet injection).

Limitations

- Streaming between Delta tables is supported only in Databricks Runtime 11.3 LTS or above.

- Because lineage is computed on a 1-year rolling window, lineage collected more than 1 year ago is not displayed. For example, if a job or query reads data from table A and writes to table B, the link between table A and table B is displayed for only 1 year.

You can filter lineage data by time frame. When “All lineage” is selected, lineage data collected starting in June 2023 is displayed.

- Workflows that use the Jobs API runs submit request are unavailable when viewing lineage. Table and column level lineage is still captured when using the runs submit request, but the link to the run is not captured.
- Unity Catalog captures lineage to the column level as much as possible. However, there are some cases where column-level lineage cannot be captured.
- Column lineage is only supported when both the source and target are referenced by table name (Example: select * from <catalog>.<schema>.<table>). Column lineage cannot be captured if the source or the target are addressed by path (Example: select * from delta."s3://<bucket>/<path>").
- Data lineage doesn’t capture reads on tables with column masks.
- If a table is renamed, lineage is not captured for the renamed table.
- If you use Spark SQL dataset checkpointing, lineage is not captured.
- Unity Catalog captures lineage from Delta Live Tables pipelines in most cases. However, in some instances, complete lineage coverage cannot be guaranteed, such as when pipelines use the APPLY CHANGES API or TEMPORARY tables.

Examples

Note

- The following examples use the catalog name lineage_data and the schema name lineagedemo. To use a different catalog and schema, change the names used in the examples.
- To complete this example, you must have CREATE and USE SCHEMA privileges on a schema. A metastore admin, catalog owner, or schema owner can grant these privileges. For example, to give all users in the group ‘data_engineers’ permission to create tables in the lineagedemo schema in the lineage_data catalog, a user with one of the above privileges or roles can run the following queries:

SQL

CREATE SCHEMA lineage_data.lineagedemo;

```
GRANT USE SCHEMA, CREATE on SCHEMA lineage_data.lineagedemo to
`data_engineers`;
```

Capture and explore lineage

To capture lineage data, use the following steps:

1. Go to your Azure Databricks landing page, click ⊕ **New** in the sidebar, and select **Notebook** from the menu.
2. Enter a name for the notebook and select **SQL** in **Default Language**.
3. In **Cluster**, select a cluster with access to Unity Catalog.
4. Click **Create**.
5. In the first notebook cell, enter the following queries:

SQL

```
CREATE TABLE IF NOT EXISTS
 lineage_data.lineagedemo.menu (
  recipe_id INT,
  app string,
  main string,
  dessert string
 );

INSERT INTO lineage_data.lineagedemo.menu
  (recipe_id, app, main, dessert)
VALUES
  (1,"Ceviche", "Tacos", "Flan"),
  (2,"Tomato Soup", "Souffle", "Creme Brulee"),
  (3,"Chips","Grilled Cheese","Cheesecake");

CREATE TABLE
 lineage_data.lineagedemo.dinner
```

```
AS SELECT
 recipe_id, concat(app," + ", main," + ",dessert)
AS
 full_menu
FROM
 lineage_data.lineagedemo.menu
```

6. To run the queries, click in the cell and press **shift+enter** or click and select **Run Cell**.

To use Catalog Explorer to view the lineage generated by these queries, use the following steps:

1. In the **Search** box in the top bar of the Azure Databricks workspace, enter lineage_data.lineagedemo.dinner and click **Search lineage_data.lineagedemo.dinner in Databricks**.
2. Under **Tables**, click the dinner table.
3. Select the **Lineage** tab. The lineage panel appears and displays related tables (for this example it's the menu table).
4. To view an interactive graph of the data lineage, click **See Lineage Graph**. By default, one level is displayed in the graph. You can click on the icon on a node to reveal more connections if they are available.
5. Click on an arrow connecting nodes in the lineage graph to open the **Lineage connection** panel. The **Lineage connection** panel shows details about the connection, including source and target tables, notebooks, and workflows.
6. To show the notebook associated with the dinner table, select the notebook in the **Lineage connection** panel or close the lineage graph and click **Notebooks**. To open the notebook in a new tab, click on the notebook name.
7. To view the column-level lineage, click on a column in the graph to show links to related columns. For example, clicking on the 'full_menu' column shows the upstream columns the column was derived from:

To demonstrate creating and viewing lineage with a different language, for example, Python, use the following steps:

1. Open the notebook you created previously, create a new cell, and enter the following Python code:

Python

```python
%python

from pyspark.sql.functions import rand, round

df = spark.range(3).withColumn("price", round(10*rand(seed=42),2)).withColumnRenamed("id","recipe_id")


df.write.mode("overwrite").saveAsTable("lineage_data.lineagedemo.price")


dinner = spark.read.table("lineage_data.lineagedemo.dinner")

price = spark.read.table("lineage_data.lineagedemo.price")


dinner_price = dinner.join(price, on="recipe_id")

dinner_price.write.mode("overwrite").saveAsTable("lineage_data.lineagedemo.dinner_price")
```

2. Run the cell by clicking in the cell and pressing **shift+enter** or clicking ▶▾ and selecting **Run Cell**.
3. In the **Search** box in the top bar of the Azure Databricks workspace, enter lineage_data.lineagedemo.price and click **Search lineage_data.lineagedemo.price in Databricks**.
4. Under **Tables**, click the price table.

5. Select the **Lineage** tab and click **See Lineage Graph**. Click on the icons to explore the data lineage generated by the SQL and Python queries.

6. Click on an arrow connecting nodes in the lineage graph to open the **Lineage connection** panel. The **Lineage connection** panel shows details about the connection, including source and target tables, notebooks, and workflows.

Capture and view workflow lineage

Lineage is also captured for any workflow that reads or writes to Unity Catalog. To demonstrate viewing lineage for an Azure Databricks workflow, use the following steps:

1. Click ⊕ **New** in the sidebar and select **Notebook** from the menu.
2. Enter a name for the notebook and select **SQL** in **Default Language**.
3. Click **Create**.
4. In the first notebook cell, enter the following query:

SQLCopy

```
SELECT * FROM lineage_data.lineagedemo.menu
```

5. Click **Schedule** in the top bar. In the schedule dialog, select **Manual**, select a cluster with access to Unity Catalog, and click **Create**.
6. Click **Run now**.
7. In the **Search** box in the top bar of the Azure Databricks workspace, enter lineage_data.lineagedemo.menu and click **Search lineage_data.lineagedemo.menu in Databricks**.
8. Under **Tables View all tables**, click the menu table.
9. Select the **Lineage** tab, click **Workflows**, and select the **Downstream** tab. The job name appears under **Job Name** as a consumer of the menu table.

Capture and view dashboard lineage

To demonstrate viewing lineage for a SQL dashboard, use the following steps:

1. Go to your Azure Databricks landing page and open Catalog Explorer by clicking **Catalog** in the sidebar.
2. Click on the catalog name, click **lineagedemo**, and select the menu table. You can also use the **Search tables** text box in the top bar to search for the menu table.
3. Click **Actions > Create a quick dashboard**.
4. Select columns to add to the dashboard and click **Create**.
5. In the **Search** box in the top bar of the Azure Databricks workspace, enter lineage_data.lineagedemo.menu and click **Search lineage_data.lineagedemo.menu in Databricks**.
6. Under **Tables View all tables**, click the menu table.

7. Select the **Lineage** tab and click **Dashboards**. The dashboard name appears under **Dashboard Name** as a consumer of the menu table.

Lineage permissions

Lineage graphs share the same permission model as Unity Catalog. If a user does not have the BROWSE or SELECT privilege on a table, they cannot explore the lineage. Additionally, users can only see notebooks, workflows, and dashboards that they have permission to view. For example, if you run the following commands for a non-admin user userA:

SQL

GRANT USE SCHEMA on lineage_data.lineagedemo to ` userA@company.com` ;

GRANT SELECT on lineage_data.lineagedemo.menu to ` userA@company.com` ;

When userA views the lineage graph for the lineage_data.lineagedemo.menu table, they will see the menu table. They will not be able to see information about associated tables, such as the downstream lineage_data.lineagedemo.dinner table.
The dinner table is displayed as a masked node in the display to userA,
and userA cannot expand the graph to reveal downstream tables from tables they do not have permission to access.

If you run the following command to grant the BROWSE permission to a non-admin user userB:

SQL

GRANT BROWSE on lineage_data to ` userA@company.com` ;

userB can now view the lineage graph for any table in the lineage_data schema.

Delete lineage data

Warning

The following instructions delete all objects stored in Unity Catalog. Use these instructions only if necessary. For example, to meet compliance requirements.

To delete lineage data, you must delete the metastore managing the Unity Catalog objects. Data will be deleted within 90 days.

Data lineage API

The data lineage API allows you to retrieve table and column lineage.

Important

To access Databricks REST APIs, you must **authenticate**.

Retrieve table lineage

This example retrieves lineage data for the dinner table.

Request

Bash

```
curl --netrc -X GET \
-H 'Content-Type: application/json' \
https://<databricks-instance/api/2.0/lineage-tracking/table-lineage \
-d '{"table_name": "lineage_data.lineagedemo.dinner", "include_entity_lineage": true}}'
```

Replace <get-workspace-instance>.

This example uses a .netrc file.

Response

JSON

```
{
 "upstreams": [
  {
   "tableInfo": {
    "name": "menu",
    "catalog_name": "lineage_data",
    "schema_name": "lineagedemo",
    "table_type": "TABLE"
   },
   "notebookInfos": [
    {
     "workspace_id": 4169371664718798,
     "notebook_id": 1111169262439324
    }
   ]
  }
 ],
 "downstreams": [
  {
   "notebookInfos": [
    {
     "workspace_id": 4169371664718798,
     "notebook_id": 1111169262439324
    }
   ]
  },
  {
   "tableInfo": {
    "name": "dinner_price",
    "catalog_name": "lineage_data",
    "schema_name": "lineagedemo",
    "table_type": "TABLE"
   },
   "notebookInfos": [
    {
     "workspace_id": 4169371664718798,
     "notebook_id": 1111169262439324
    }
   ]
  }
 ]
}
```

Retrieve column lineage

This example retrieves column data for the dinner table.

Request

Bash

```
curl --netrc -X GET \
-H 'Content-Type: application/json' \
https://<databricks-instance/api/2.0/lineage-tracking/column-lineage \
-d '{"table_name": "lineage_data.lineagedemo.dinner", "column_name": "dessert"}}'
```

Replace <get-workspace-instance>.

This example uses a .netrc file.

Response

JSON

```
{
 "upstream_cols": [
  {
   "name": "dessert",
   "catalog_name": "lineage_data",
   "schema_name": "lineagedemo",
   "table_name": "menu",
   "table_type": "TABLE"
  },
  {
   "name": "main",
   "catalog_name": "lineage_data",
   "schema_name": "lineagedemo",
   "table_name": "menu",
   "table_type": "TABLE"
  },
  {
   "name": "app",
   "catalog_name": "lineage_data",
   "schema_name": "lineagedemo",
   "table_name": "menu",
   "table_type": "TABLE"
  }
 ],
 "downstream_cols": [
  {
   "name": "full_menu",
   "catalog_name": "lineage_data",
   "schema_name": "lineagedemo",
   "table_name": "dinner_price",
   "table_type": "TABLE"
  }
 ]
}
```