Writing to GCP(google cloud) lakehouse runtime catalog iceberg tables with dbt

Masaya Kameyama

2026-08-24

Introduction

AI . dbt lakehouse runtime catalog 3 .

, GCP lakehouse runtime catalog snowflake . 2 , bigquery lakehouse runtime catalog , . , bigquery DDL/DML lakehouse runtime catalog iceberg table .

, dbt bigquery native table , (gold) lakehouse runtime catalog iceberg table . , dbt , . , , . , 2026 8 .

bigquery lakehouse runtime catalog

lakehouse runtime catalog bigquery , **4** . managed iceberg table , :

```
CREATE TABLE `PROJECT_ID.CATALOG_ID.NAMESPACE.TABLE_NAME` (id int, data string);
INSERT INTO `PROJECT_ID.CATALOG_ID.NAMESPACE.TABLE_NAME` VALUES (1, "foo");
```

managed iceberg table `WITH CONNECTION OPTIONS(file_format, table_format, storage_uri)` . , `CATALOG_ID` ID . `catalog_type CATALOG_TYPE_GCS_BUCKET` . `UPDATE` / `DELETE` / `MERGE` .

`gcp.biglake.bigquery-dml.enabled` . bigquery DDL , bigquery DML . spark OSS , bigquery `SELECT` , DML

```
DML statements are only supported over tables that have data stored in BigQuery.
```

. `TBLPROPERTIES` `ALTER TABLE` bigquery DML . iceberg table , .

, , set up bigquery DDL/DML . , .

terraform

namespace terraform . bigquery `CREATE TABLE` namespace , namespace :

```
resource "google_storage_bucket" "lakehouse" {
  name                        = "my-lakehouse-bucket"
  location                    = "us-central1"
  force_destroy               = true
  uniform_bucket_level_access = true
}

resource "google_biglake_iceberg_catalog" "main" {
  name             = google_storage_bucket.lakehouse.name
  catalog_type     = "CATALOG_TYPE_GCS_BUCKET"
  credential_mode  = "CREDENTIAL_MODE_VENDED_CREDENTIALS"
  primary_location = "us-central1"
}

resource "google_biglake_iceberg_namespace" "gold" {
  catalog      = google_biglake_iceberg_catalog.main.name
  namespace_id = "gold"
}

# vended credentials
#     GCS metadata/data     ,
resource "google_storage_bucket_iam_member" "catalog_agent" {
  bucket = google_storage_bucket.lakehouse.name
  role   = "roles/storage.objectAdmin"
  member = "serviceAccount:${google_biglake_iceberg_catalog.main.biglake_service_account}"
}
```

IAM , `storage.objects.create denied` 403 . vended credentials GCS , GCS .

dbt

dbt 1.10 `catalogs.yml` , dbt-bigquery `catalog_type: biglake_metastore` . **bigquery managed iceberg table** . , iceberg `table_format` /

`file_format` / `storage_uri` OPTIONS , 3 `project.dataset.table` . catalog integration `biglake_metastore` 1 , `managed_iceberg` .

, dbt core Relation `database.schema.identifier` 3 . lakehouse runtime catalog project, catalog, namespace, table 4 , .

Relation , config 4 DDL/DML . :

```
{% macro lakehouse_iceberg_merge_sql(table_ref, source_sql, column_names, unique_keys) %}
  {%- set update_columns = column_names | reject('in', unique_keys) | list -%}
  merge into {{ table_ref }} as dbt_dest
  using (
    {{ source_sql }}
  ) as dbt_src
  on
    {%- for key in unique_keys %}
    {{ 'and' if not loop.first }} dbt_dest.`{ key }` = dbt_src.`{ key }`
    {%- endfor %}
  {% if update_columns | length > 0 -%}
  when matched then update set
    {%- for col in update_columns %}
    `{ col }` = dbt_src.`{ col }`{{ ',' if not loop.last }}
    {%- endfor %}
  {% endif -%}
  when not matched then insert
    ({% for col in column_names %}`{ col }`{{ ', ' if not loop.last }}{% endfor %})
  values
    ({% for col in column_names %}dbt_src.`{ col }`{{ ', ' if not loop.last }}{% endfor %})
{% endmacro %}


{% materialization lakehouse_iceberg, adapter='bigquery' %}

  {%- set identifier = model['alias'] -%}
  {%- set catalog = var('iceberg_catalog') -%}
  {%- set namespace = var('iceberg_namespace') -%}
  {%- set table_ref = '`' ~ target.project ~ '.' ~ catalog ~ '.' ~ namespace ~ '.' ~ identif

  {%- set unique_key = config.get('unique_key') -%}
  {%- set unique_keys = [unique_key] if unique_key is string else (unique_key or []) -%}
  {%- set full_refresh_mode = should_full_refresh() -%}
```

```
  {{ run_hooks(pre_hooks) }}

  {#- 4    Relation    get_columns_in_relation    .
      compiled SQL            -#}
  {%- set columns = adapter.get_column_schema_from_query(get_empty_subquery_sql(sql)) -%}
  {%- set column_names = columns | map(attribute='name') | list -%}
  {%- set column_ddl = [] -%}
  {%- for c in columns -%}
    {%- do column_ddl.append('`' ~ c.name ~ '` ' ~ c.data_type) -%}
  {%- endfor -%}

  {% if full_refresh_mode %}
    {% do run_query('drop table if exists ' ~ table_ref) %}
  {% endif %}

  {% do run_query('create table if not exists ' ~ table_ref ~ ' (' ~ column_ddl | join(', ')

  {% call statement('main') %}
    {% if unique_keys | length > 0 and not full_refresh_mode %}
      {{ lakehouse_iceberg_merge_sql(table_ref, sql, column_names, unique_keys) }}
    {% else %}
      insert into {{ table_ref }}
        ({% for col in column_names %}`{ col }`{{ ', ' if not loop.last }}{% endfor %})
      select {% for col in column_names %}`{ col }`{{ ', ' if not loop.last }}{% endfor %}
      from (
        {{ sql }}
      )
    {% endif %}
  {% endcall %}

  {{ run_hooks(post_hooks) }}

  {#- 4  iceberg table relation cache        -#}
  {{ return({'relations': []}) }}

{% endmaterialization %}
```

3 .

. 4 Relation `adapter.get_columns_in_relation` . `get_empty_subquery_sql` compiled SQL `where false limit 0` , `adapter.get_column_schema_from_query` bigquery . dbt contract .

. run `CREATE TABLE IF NOT EXISTS` `MERGE` . INSERT , 2 . `--full-refresh` `DROP` → `CREATE` → `INSERT ... SELECT` .

`return({'relations': []})` . dbt relation cache , 3 relation dbt .
.

:

```
{{ config(
    materialized='lakehouse_iceberg',
    schema='gold',
    unique_key=['event_date', 'tenant_id']
) }}

select
    event_date,
    tenant_id,
    impressions,
    clicks,
    current_timestamp() as loaded_at
from {{ ref('int_events_daily') }}
```

`CREATE TABLE` `INSERT ... VALUES` , dbt . , :

`CREATE TABLE( )`
`INSERT ... VALUES`
`INSERT ... SELECT`(native table → iceberg table)
`MERGE`(native table USING)
`CREATE TABLE AS SELECT`

native table iceberg table `INSERT ... SELECT` `MERGE` , . `INSERT ... VALUES` , dbt .

dbt . `loaded_at` `current_timestamp()` :

- 1 run: `CREATE TABLE` + `MERGE`
- 2 run: `loaded_at` . MERGE
- `--full-refresh`: dbt `MERGE` `INSERT` , GCS ID

lakehouse runtime catalog , 3 .

1. gold bigquery dataset . managed iceberg table dataset
2. iceberg REST API

```
GET https://biglake.googleapis.com/iceberg/v1/restcatalog/v1/projects/PROJECT_ID/catalogs/CA

{ "identifiers": [ { "namespace": ["gold"], "name": "gold_events_daily" } ] }
```

3. iceberg

pyiceberg . spark trino :

```python
import google.auth
import google.auth.transport.requests
from pyiceberg.catalog.rest import RestCatalog

creds, _ = google.auth.default(scopes=["https://www.googleapis.com/auth/cloud-platform"])
creds.refresh(google.auth.transport.requests.Request())

catalog = RestCatalog("main", **{
    "uri": "https://biglake.googleapis.com/iceberg/v1/restcatalog",
    #       400
    "warehouse": "gs://my-lakehouse-bucket",
    "token": creds.token,
    "header.x-goog-user-project": "PROJECT_ID",
    "header.X-Iceberg-Access-Delegation": "vended-credentials",
    # RestCatalog  OAuth    ADC
    "rest.auth.type": "noop",
})

t = catalog.load_table("gold.gold_events_daily")
print(t.current_snapshot())
print(t.scan().to_arrow())
```

`pyiceberg[gcsfs]` FileIO `load_table` , `pyiceberg[pyarrow,gcsfs]` [1].

`loaded_at` dbt run . `EXPORT TABLE METADATA` , dbt bigquery .
export , .

3

. ,

- . relation cache , `ref()` . iceberg table dbt , -
  gold

[1] , ADC `quota_project_id` , API . ,
`UserProjectAccountProblem` 403 . `gcloud config set project` `gcloud auth application-default set-quota-project` .

- `dbt test` `dbt docs` . 3 relation . silver , iceberg table 4
- . `CREATE TABLE IF NOT EXISTS` , `--full-refresh`

, ,

- `DROP TABLE` **GCS** . , metadata parquet . `--full-refresh` ,
- **native table** **iceberg table** . bigquery US us-central1 , . silver dataset , `INSERT ... SELECT`
- . 3 . GA

, dbt . dbt-bigquery Relation 4 , `catalogs.yml` lakehouse runtime catalog catalog type , . .

Summary

- 2 bigquery lakehouse runtime catalog , . `` `PROJECT_ID.CATALOG_ID.NAMESP `` `WITH CONNECTION` `OPTIONS` DDL/DML .
- `gcp.biglake.bigquery-dml.enabled` , bigquery , OS-S .
- dbt-bigquery `catalogs.yml`(`catalog_type: biglake_metastore`) managed iceberg table , dbt core Relation 3 , lakehouse runtime catalog .
- 4 , bigquery native table , lakehouse runtime catalog iceberg table . `get_column_schema_from_query` `get_empty_subquery_sql` , run `MERGE`, `--full-refresh` `DROP` → `CREATE` → `INSERT ... SELECT` .
- `INSERT ... SELECT`, `MERGE`(native source USING), `CREATE TABLE AS SELECT` . bigquery dataset , iceberg REST API pyiceberg . `EXPORT TABLE METADATA` .
- iceberg table (`ref()`) , `dbt test` / `dbt docs` . `DROP TABLE` GCS , native table .