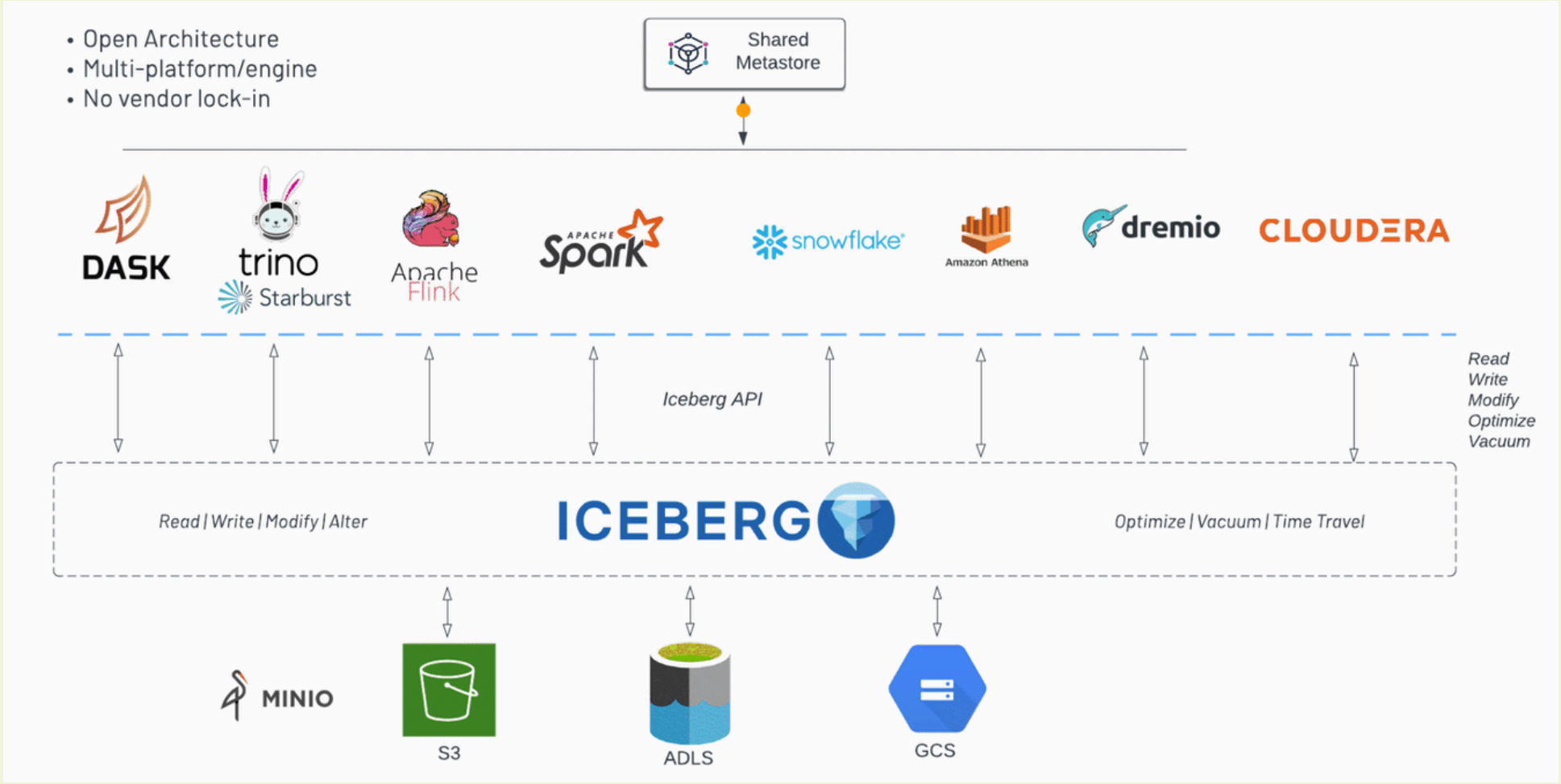
# Using iceberg tables with Databricks

Ganesh R

Azure Data Engineer

As per official documentation: Databricks iceberg tables store their data and metadata files in an external storage (like S3, Google Cloud Storage, or Azure Storage). That means, you are responsible for management of the external storage, e.g. data protection, data recovery etc. Databricks does not provide Fail-safe storage for Iceberg tables

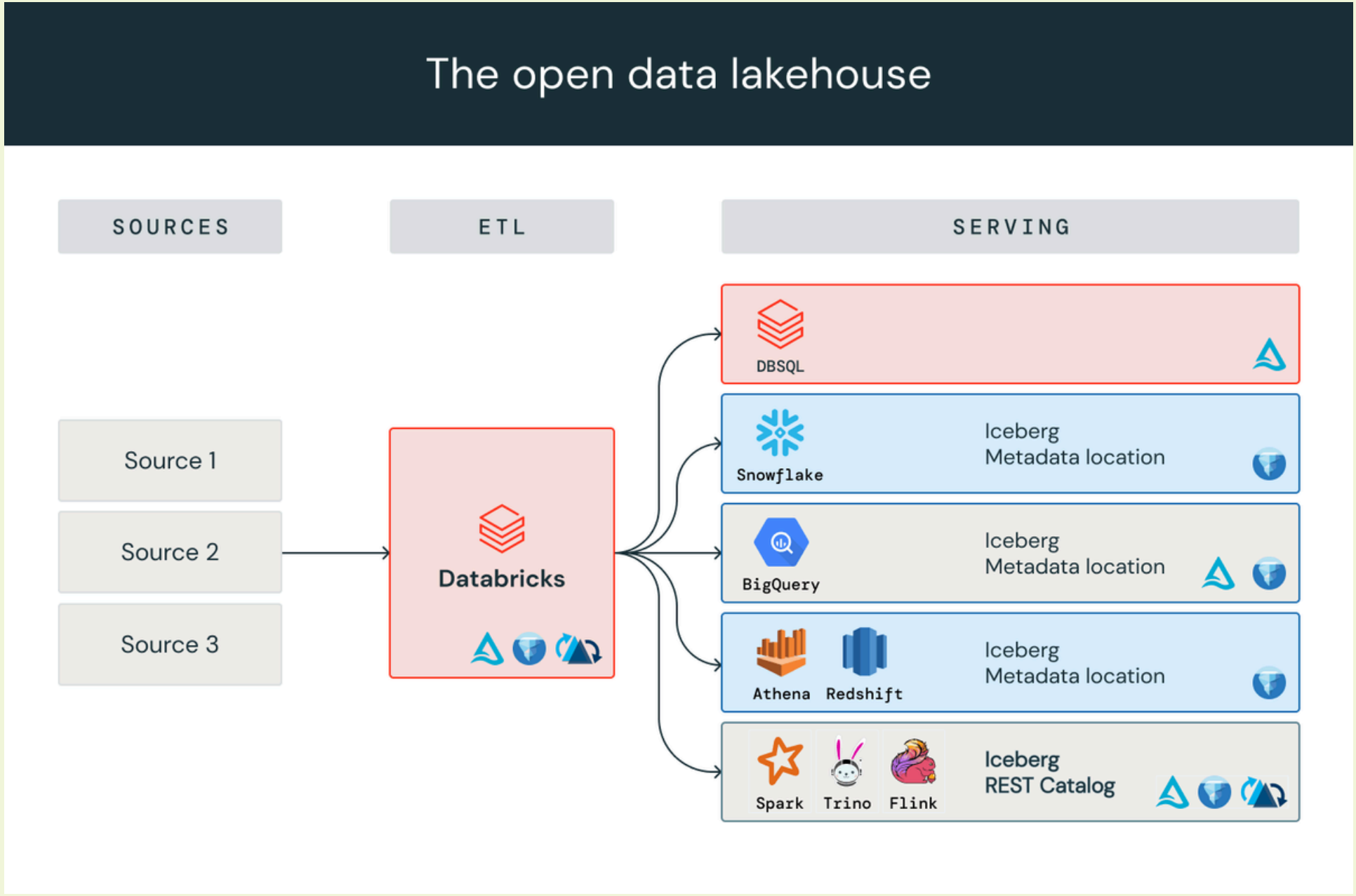


Why Use Apache Iceberg with Databricks?

Apache Iceberg has emerged as a strong contender in the table format space, offering unique advantages over more traditional options like Delta Lake. Apache Iceberg, is an open standard, allowing for greater flexibility and interoperability across different platforms.

Some key advantages of using Apache Iceberg include:

- Vendor Neutrality: Iceberg allows you to avoid vendor lock-in, ensuring that your data solutions remain flexible and adaptable to changing business needs.
- Ecosystem Integration: With support from a wide range of tools like Dremio, BigQuery, Apache Drill, and Snowflake, Apache Iceberg offers a rich ecosystem that goes beyond what Delta Lake can provide.
- Advanced Features: Apache Iceberg includes unique capabilities like partition evolution and hidden partitioning, which simplify data management and improve performance.


The open data lakehouse
SOURCES
ETL
SERVING
Source 1
Source 2
Source 3
Databricks
DBSQL
Snowflake
Iceberg
Metadata location
BigQuery
Iceberg
Metadata location
Athena
Redshift
Iceberg
Metadata location
Spark
Trino
Flink
Iceberg
REST Catalog

Let’s discuss aobut key benefits of Iceberg Tables:

1. Lower Storage Costs

Storing Iceberg tables in S3 can be more cost-efficient than Databrick’s managed storage. Cloud providers like AWS, GCP, and Azure typically offer lower storage costs than Snowflake, making Iceberg a budget-friendly alternative for massive datasets. Note that for small data it may still not make sene to save pennies, & spend dollars on iceberg table management.

2. Reduced Compute Costs

Iceberg supports partition pruning and metadata caching, leading to faster queries with less compute overhead.
However these features are available in Snowflake too and Snowflake can be more efficient in this based on type of data.

3. Multi-Engine Compatibility (No Vendor Lock-in)
One of Iceberg’s biggest advantages is its interoperability with multiple compute engines:

Spark, Flink, Trino, Presto, and more can query Iceberg tables directly.

This flexibility allows organizations to use different engines without being locked into Snowflake

4. No Ingestion Cost:

In Databrick's, external tables require extra steps to refresh tables using COPY INTO or workflow, which are costly operations.

With Iceberg, Databrick's can query the tables directly, eliminating the need for additional ingestion steps, reducing both complexity and cost.

5. Schema Evolution

Iceberg supports schema evolution similar to Databrick's but with additional benefits.
It tracks column history and integrates with external catalogs like AWS Glue, Hive Metastore, and Nessie, offering more flexibility for schema changes.

6. ACID Transactions:
Iceberg ensures ACID compliance using snapshot-based Multi-Version Concurrency Control (MVCC). Instead of modifying data files directly, each update creates a new metadata snapshot that tracks:

- Newly added files
- Deleted or modified files
- Previous versions of the table

For concurrent reads and writes, Iceberg uses Optimistic Concurrency Control (OCC):

1. Each transaction starts with the latest table snapshot.

2. Writers generate a new snapshot and attempt to commit it.

3. Before committing, Iceberg checks for conflicts with the latest snapshot.

4. If unchanged, the new snapshot is committed successfully.

5. If changed, the transaction retries with the updated snapshot to resolve conflicts.

However, Iceberg is not the silver bullet if your data is not in Petabyte or Terabyte volume. So do a POC before going ahead with Iceberg tables with Databricks. So if I have to make a rule on when to use Iceberg tables in Databricks and when not?

1. If your data volume is really large (TBs or PBs), Iceberg's cheap storage cost can add to your saving.

2. You have frequent or costly ingestion job to ingest data in snowflake Icebergs no ingestion required can add to your saving.,

But if your data is small or medium you may not see performance or cost benefit if you compare between Iceberg table and Snowflake tables.

Also if your data is dynamic and you make use of Databricks automatic clustering to keep your data organized for faster query, you may want to stick to Databricks because there is no automatic clustering with Iceberg.

## Enough of Theory, Let's Div into Practicals

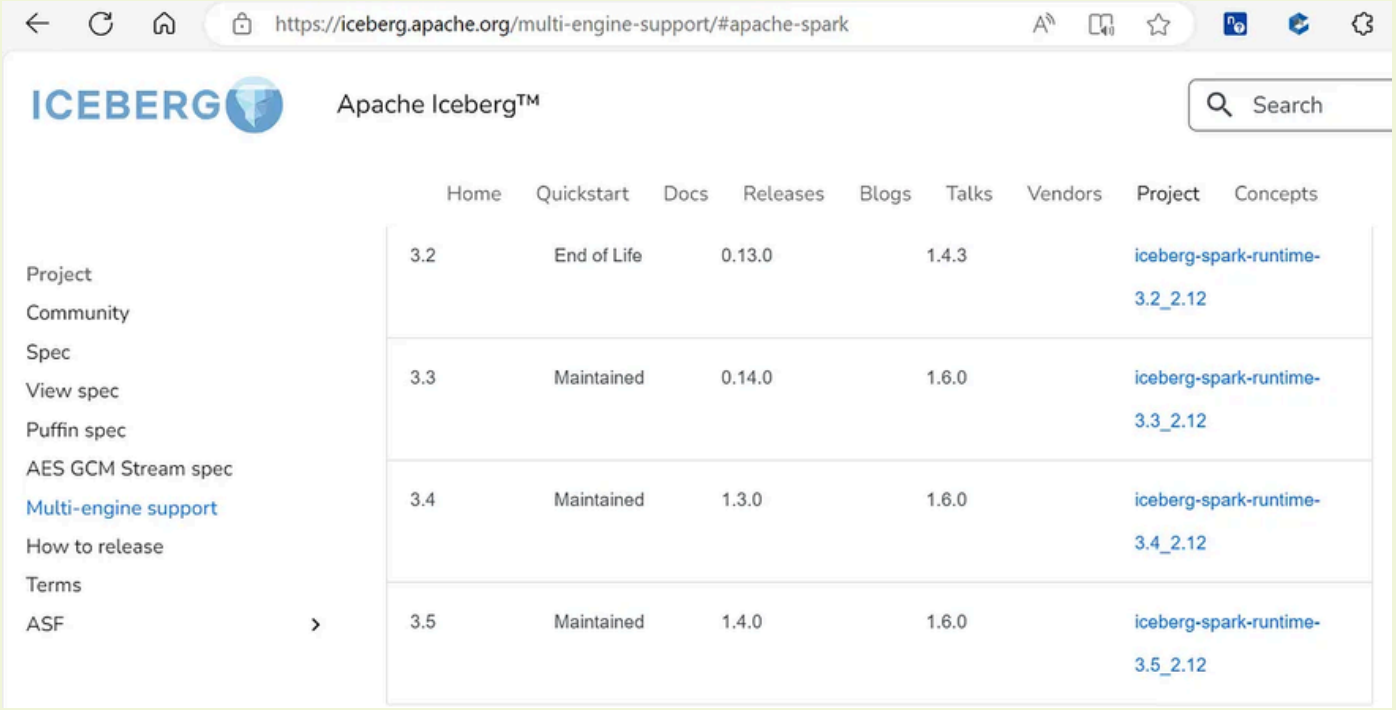
Working with Iceberg Table: PyIceberg PyIceberg is a lightweight python package that uses Iceberg APIs to perform all operations like: Creating Table, Reading and Writing Data etc.

Download the Iceberg Jar File

First, you'll need to download the appropriate Iceberg runtime jar file that matches your Databricks Spark version. You can find the Spark version under Compute > Cluster > Configuration: Databricks Runtime Version. Make sure to select the Iceberg runtime jar that corresponds to this version.

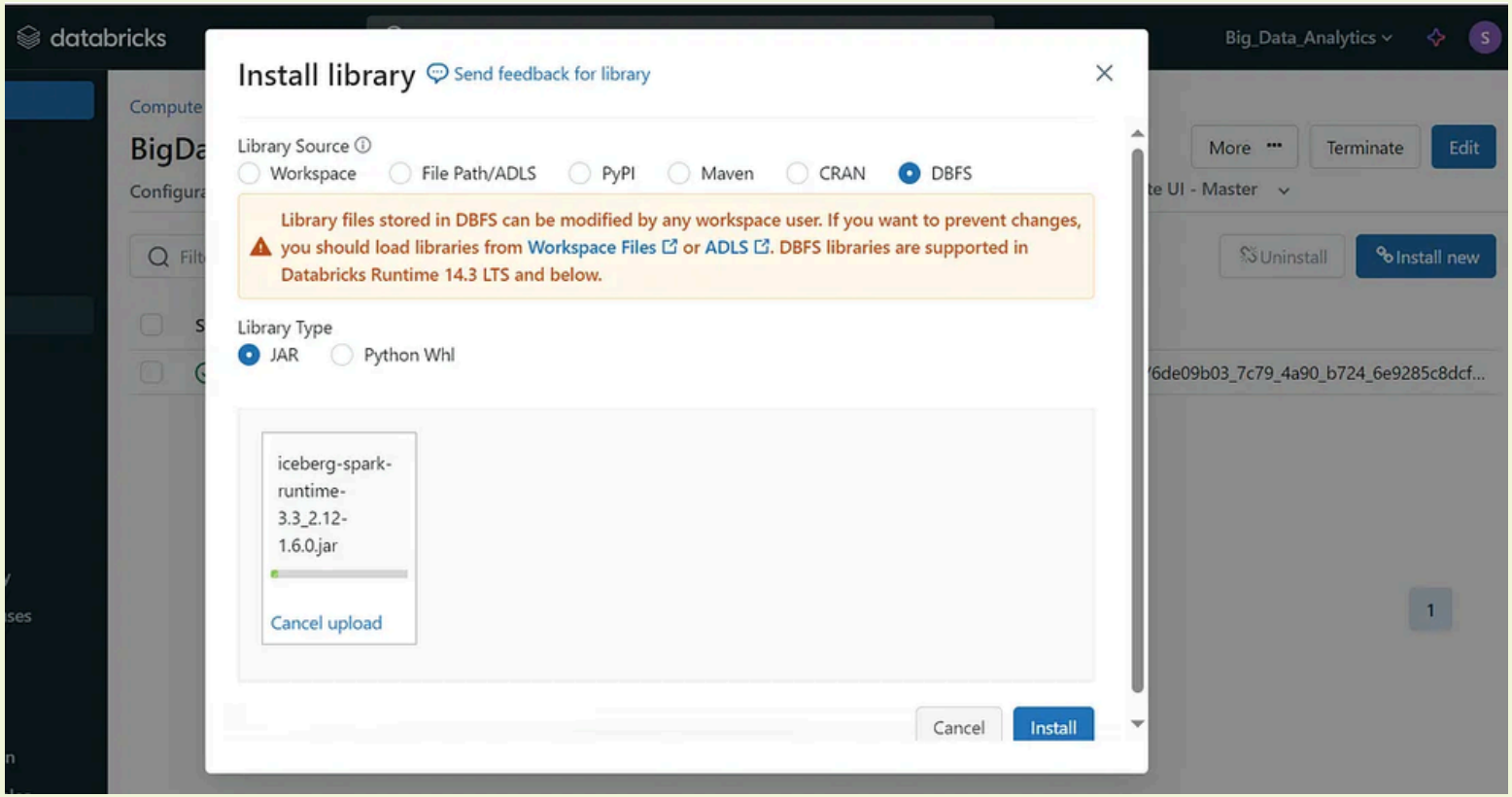
Search compatible version here

For example, if you're using Databricks Runtime 10.4 (Spark 3.2), download the "iceberg-spark-runtime-3.2_2.12" jar file.



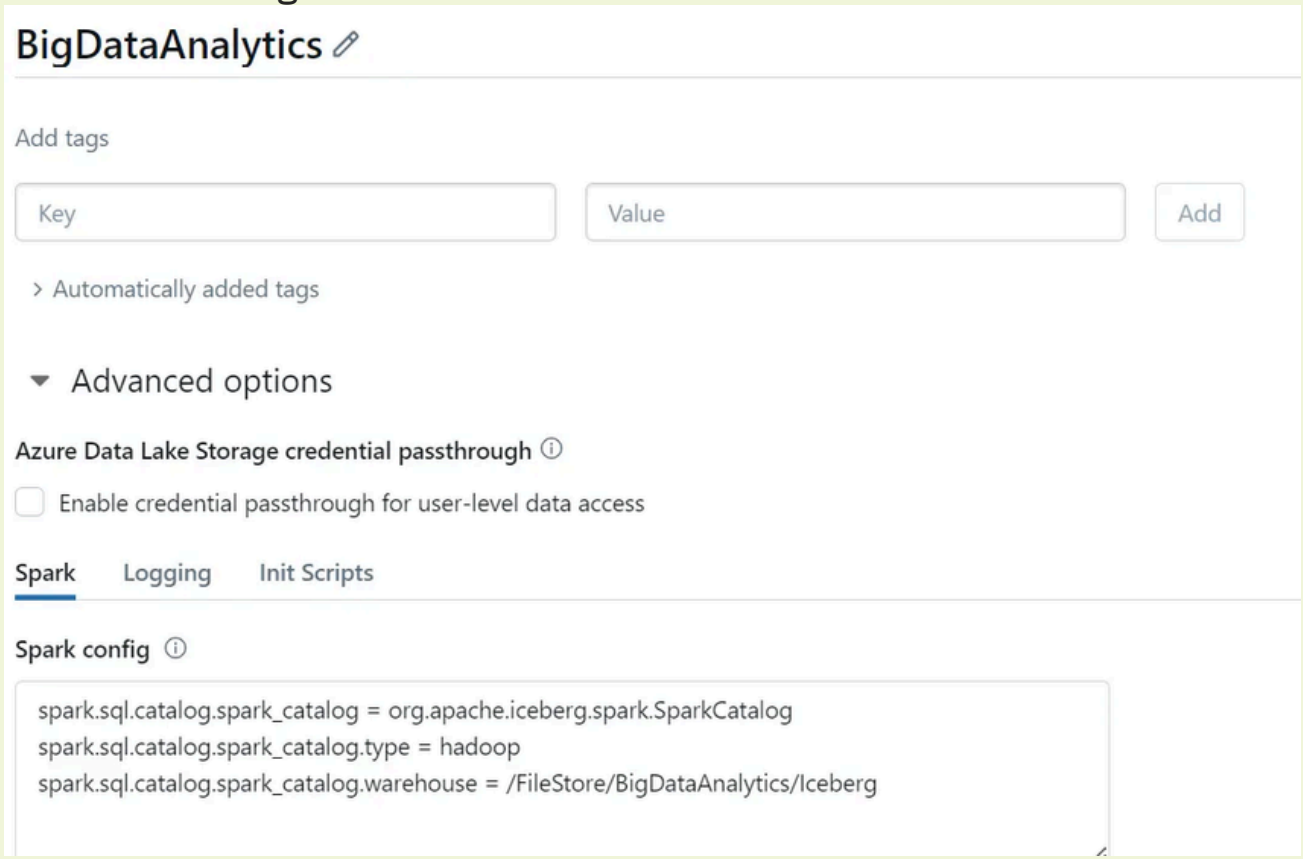
Import the Iceberg Jar File

Next, navigate to Compute > Cluster > Libraries in your Databricks workspace. Click on the Install new button to add the Iceberg runtime jar file you downloaded.



Amend Cluster Configuration to Enable Iceberg

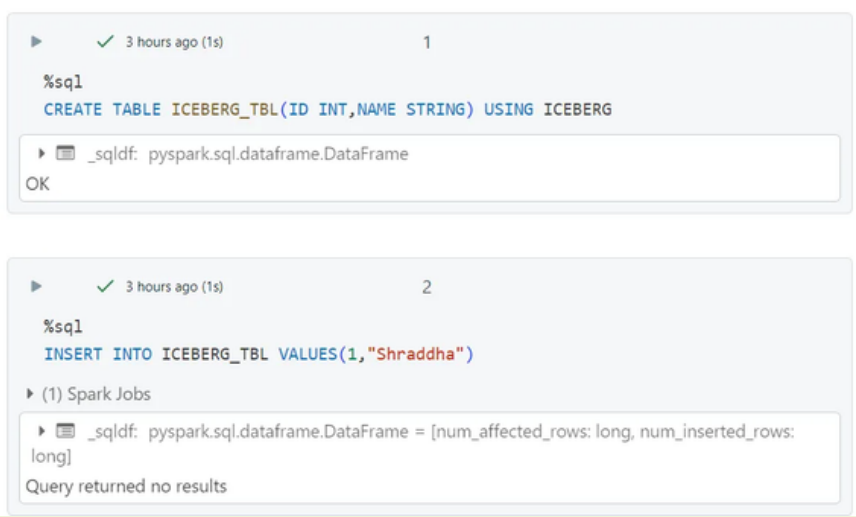
To enable Iceberg, you'll need to modify your cluster configuration. Go to Configuration > Advanced Options > Spark Config and add the following lines:

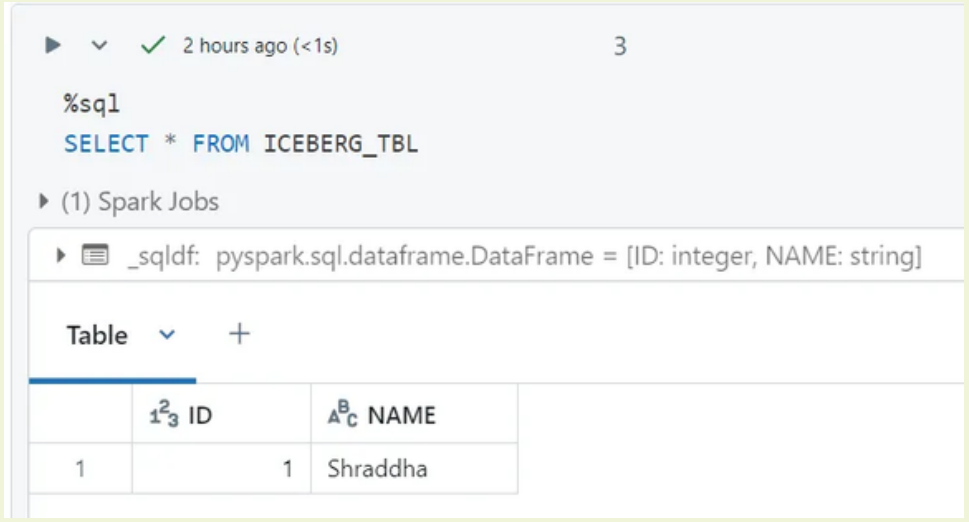


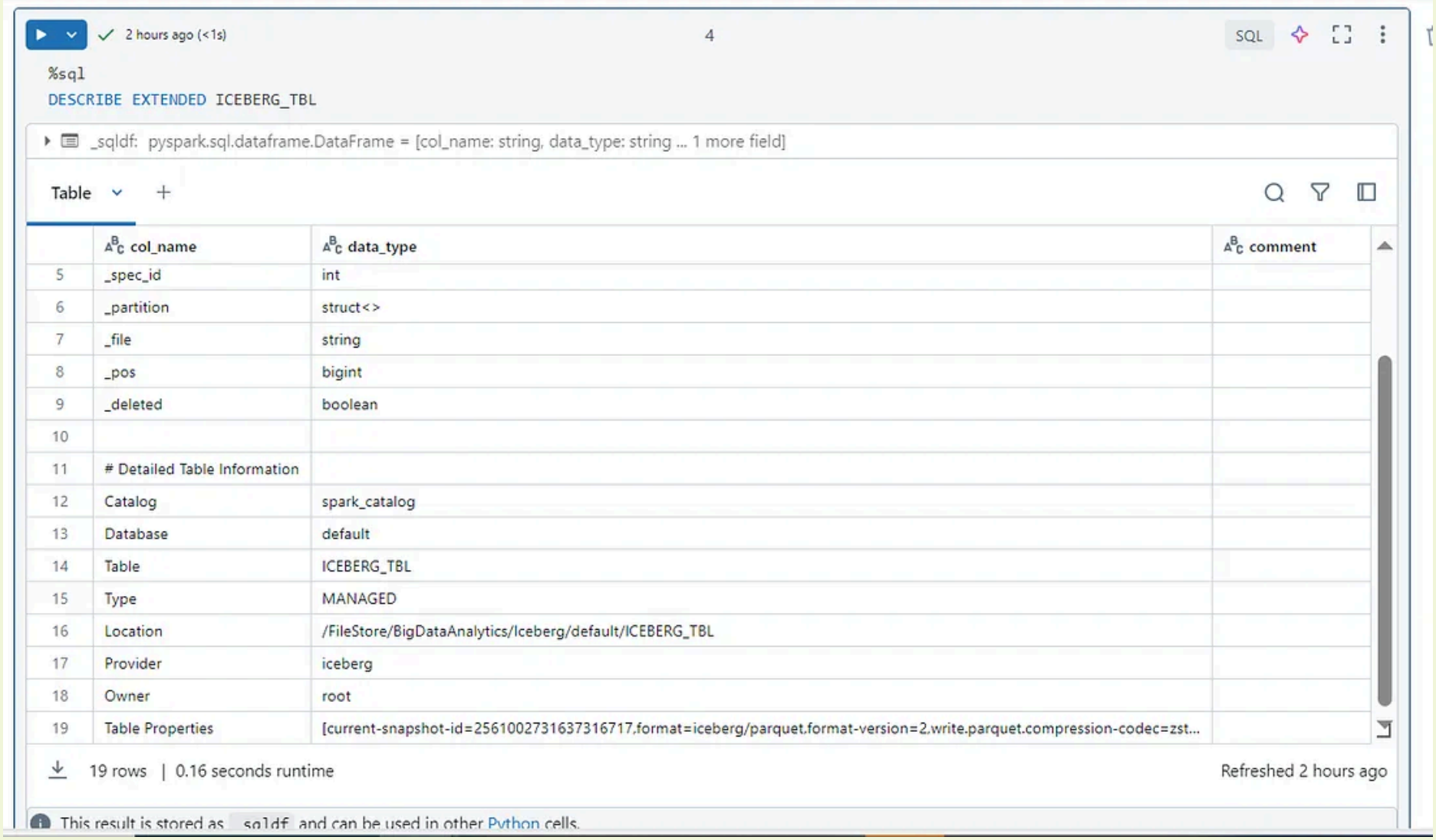

These settings create the necessary catalog for working with Iceberg tables. The `spark.sql.catalog.spark_catalog.warehouse` path defines where the table will be created. Initially, this location is in DBFS, which we'll change later to use Azure Data Lake Storage (ADLS).

Create and Use Iceberg Tables in Databricks

You can now create Iceberg tables within Databricks. Create a new notebook and run the following SQL commands:




2 hours ago (<1s)
4
SQL
%sql
DESCRIBE EXTENDED ICEBERG_TBL
_sqldf: pyspark.sql.dataframe.DataFrame = [col_name: string, data_type: string ... 1 more field]
Table
col_name
data_type
comment
5 _spec_id int
6 _partition struct<>
7 _file string
8 _pos bigint
9 _deleted boolean
10
11 # Detailed Table Information
12 Catalog spark_catalog
13 Database default
14 Table ICEBERG_TBL
15 Type MANAGED
16 Location /FileStore/BigDataAnalytics/Iceberg/default/ICEBERG_TBL
17 Provider iceberg
18 Owner root
19 Table Properties [current-snapshot-id=2561002731637316717,format=iceberg/parquet.format-version=2,write.parquet.compression-codec=zst...
19 rows | 0.16 seconds runtime
Refreshed 2 hours ago
This result is stored as _sqldf and can be used in other Python cells.

Configure Spark to Store Data in an External Data Lake

To allow access to the Iceberg table outside of Databricks, configure Spark to store data in an Azure Data Lake Storage (ADLS) account. Add the following key to your cluster's Spark Config:

fs.azure.account.key.<Your ADLS Storage Account Name>.blob.core.windows.net <Your ADLS Access Key>

Then, update the `spark.sql.catalog.spark_catalog.warehouse` path to point to your ADLS location:

spark.sql.catalog.spark_catalog.warehouse wasbs://<Your ADLS Container Name>@<Your ADLS Storage Account Name>.blob.core.windows.net/<Optional Folder Location>/

Restart your cluster to apply the changes.

# Follow for more content like this Azure Cloud for Data Engineering

# Ganesh R

## Azure Data Engineer

https://www.linkedin.com/in/rganesh0203/