White Paper

Power Multicloud Data Analytics and AI using Dell Object Storage and Databricks

Activate your data in Dell object storage using Databricks Lakehouse Platform multicloud analytics

May 2023

H19607

Abstract

This white paper describes how to enable multicloud analytics and AI using Databricks Lakehouse Platform for your Dell ECS investment on-premises or in cloud-adjacent data centers.

Copyright

Contents

Executive summary

Overview

Enterprise data is increasingly becoming distributed across core, cloud, and edge. The strategy regarding “where to place the data” is important and is influenced by factors such as data gravity, architectural considerations, security, governance, choice of tools, cost economics, decoupling of compute and storage, separation of workloads, and many more.

The data industry has also been disrupted by several new architecture paradigms. Two significant innovations are available:

- **Open data formats** - This innovation includes open file formats (Parquet, AVRO, and ORC) and open table formats (Delta Lake, Iceberg, and Hudi), which enable data interoperability.
- **Data Lakehouse** - This innovation combines the best of both worlds, which is the performance and reliability of data warehouses with the flexibility and multi-workload support of data lakes.

Customers have limited options to plug in their existing on-premises data stores into these modern architecture constructs without investing significant effort and cost.

This white paper describes how customers can use the power of the Databricks Lakehouse Platform, a unified, open and scalable cloud-based modern data platform to query, process, and share data that is stored in Dell ECS object storage on-premises or in a cloud-adjacent data center such as Faction. Additionally, customers can comply with data localization requirements, build multicloud resiliency, and prevent the creation of multiple copies of data by connecting the same data source to any public cloud. They can also use the Databricks Delta Sharing, an open-source solution, to share data in Dell ECS inside and outside their organization.

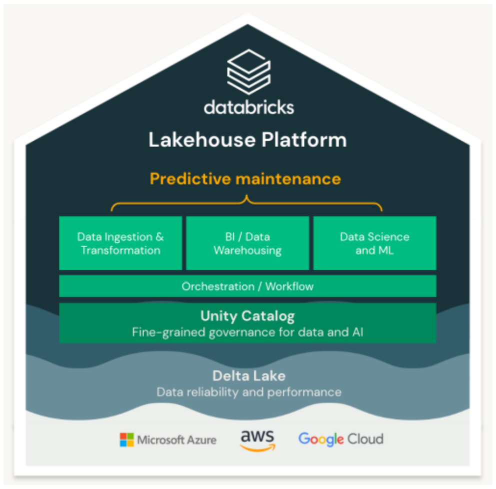
About Dell Technologies

Dell Technologies helps organizations and individuals build their digital future and transform how they work, live, and play. The company provides customers with the industry’s broadest and most innovative technology and services portfolio for the data era.

Dell ECS is the leading object storage platform from Dell Technologies, with unmatched scalability, performance, resilience, and economics. ECS delivers rich S3-compatibility on a globally distributed architecture, empowering organizations to support enterprise workloads such as cloud-native, archive, IoT, AI, and big data analytics applications at scale. For more information, see Dell Enterprise Object Storage.

About Databricks

Databricks was founded in 2013 by the original creators of Apache Spark™, Delta Lake and MLflow. As the only unified lakehouse platform in the cloud, Databricks combines the best of data warehouses and data lakes to offer a simple, open, multicloud platform for data and Artificial Intelligence (AI).



About Faction

Faction is the leader in cloud data services which enable one copy of data to be presented to applications, and in cloud-native and third-party data services such as AI, Machine Learning (ML), and Business Analytics across multiple public clouds, simultaneously. Faction's premier cloud data service product, Cloud Control Volumes, reflects a data-first architecture for storing and concurrently attaching data to multiple public clouds. Faction delivers cloud data agility, performance, scalability, security, and data sovereignty as a service, enabling data-driven transformation.

Audience

This white paper is for organizations wanting to leverage Databricks compute and data sharing services for data residing in Dell ECS Object Storage on-premises or in a co-location facility.

Revision history

Publication date	Part number	Description
May 2023		Initial release

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on this document. Contact the Dell Technologies team by email.

Authors:

- Dell Technologies: Chetan Somaiah and Ranjeetha Raja
- Databricks: Ron Joy

Note: For links to related documentation, see the Dell Technologies ECS Info Hub.

Solution Overview

Implementing this solution requires setting up a Dell ECS object storage either on-premises or in a cloud-adjacent data center such as Faction. When connected directly on-premises, the connectivity happens over public Internet. In this case, the ECS S3 endpoint must be publicly accessible. The Databricks instance is hosted either in AWS or Azure public cloud. This white paper elaborates on the case where Dell ECS is set up in Faction.

Dell ECS is used as an external storage location for Databricks. Data is stored in ECS using the Delta Lake open table format. Spark programmatically reads and writes data from and into ECS. Because the external data is not known to Databricks metastore, the data does not reflect in the Unity Catalog automatically, however. This solution leverages certain practices for ease of demonstration, such as inputting s3a access and secret keys into notebook code. For at-scale production use cases, you can leverage other mechanisms. Similarly, you may opt to use bucket-specific configurations in your core-site.xml (for example, fs.s3a.bucket.BUCKETNAME.endpoint).

Dell has partnered with Faction, Inc. to deliver turnkey cloud data services that combine Dell's robust storage portfolio with Faction's cloud data enablement services, including access to the Faction Internetwork eXchange™ (FIX), Faction's proprietary network fabric. For more information, see Faction. Through FIX, data that is stored on the dell-powered object cloud data service can be accessed by applications and services in any cloud within the region. This is how Databricks connects to Dell ECS object storage hosted in Faction.

Databricks, Faction, and Dell ECS use Transport Security Protocol (TLS) 1.2 for data in transit. All data traffic is encrypted using TLS, preventing vulnerable access points from cyberattacks.

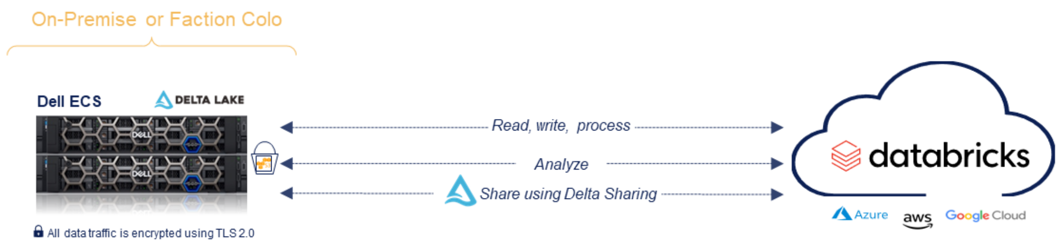

On-Premise or Faction Colo
Dell ECS
DELTA LAKE
Read, write, process
Analyze
Share using Delta Sharing
databricks
Azure
aws
Google Cloud
All data traffic is encrypted using TLS 2.0

Figure. Connecting Dell ECS object storage to Databricks

Databricks Architecture

The key architectural components of Databricks are:

- **Delta Lake** – Delta Lake provides the most unified lakehouse foundation for all data types, allowing you to maintain one copy of your data for all workloads, and its 100% open source. Delta Lake is the default storage format for all operations on Databricks.
- **Unity Catalog** – Unity Catalog is a governance solution purpose-built for the lakehouse to simplify security and governance across both structured and unstructured data, as well as all data, analytics and AI products. .
- **Delta Sharing** – Delta Sharing enables open, secure data sharing without any vendor dependence. Databricks builds Delta Sharing into its Unity Catalog data governance platform, enabling a Databricks user, called a data provider, to share data with a person or group outside their organization, called a data recipient. .

Environment Setup

Connect to Dell ECS from Databricks

Open a workspace and create a new notebook in Databricks.

Set the following S3 parameters to connect to Dell ECS storage:

```
sc._jsc.hadoopConfiguration().set("fs.s3a.access.key", "<s3_access_key>")
sc._jsc.hadoopConfiguration().set("fs.s3a.secret.key", "<s3_secret_key>")

# If you are using Auto Loader file notification mode to load files,
provide the AWS Region ID.
sc._jsc.hadoopConfiguration().set("fs.s3a.endpoint", <end point URL>)
```

Delta Sharing

Set up *open Delta Sharing* to share data in Dell ECS with recipients who do not have a Databricks account:

1. Install Delta Sharing module in a VM. See Delta Sharing.

2. Create a Delta sharing config file to specify what data you want to share with others.

```
# The format version of this config file
version: 1
# Config shares/schemas/tables to share
shares:
- name: "share1"
  schemas:
  - name: "schema1"
    tables:
    - name: "table1"
      # S3. See https://github.com/delta-io/delta-sharing#s3 for how to
config the credentials
      # location: "s3a://<bucket-name>/<the-table-path>"
      location: "s3a://nick-poc-data/delta_sharing/test1"

# Set the host name that the server will use
host: "localhost"
# Set the port that the server will listen on. Note: using ports below
1024
# may require a privileged user in some operating systems.
# port: 8080
port: 8999
# Set the url prefix for the REST APIs
endpoint: "/delta-sharing"
# Set the timeout of S3 presigned url in seconds
preSignedUrlTimeoutSeconds: 3600
# How many tables to cache in the server
deltaTableCacheSize: 10
# Whether we can accept working with a stale version of the table. This
is useful when sharing
# static tables that will never be changed.
stalenessAcceptable: false
# Whether to evaluate user provided `predicateHints`
evaluatePredicateHints: false
```

3. Create a core-site.xml and specify the Dell ECS storage connection information to let the Delta sharing service know where data is stored by running the following commands:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
```

```
  <property>
    <name>fs.s3a.endpoint</name>
    <description>AWS S3 endpoint to connect to. An up-to-date list
is provided in the AWS Documentation: regions and endpoints. Without
this property, the standard region (s3.amazonaws.com) is assumed.
    </description>
    <value>{endpoint_url}</value>
  </property>

  <property>
    <name>fs.s3a.access.key</name>
    <value>{access_key}</value>
  </property>

  <property>
    <name>fs.s3a.secret.key</name>
    <value>{secret_key}</value>
  </property>

</configuration>
```

4. Start a docker version of a Delta sharing service using the configurations you created in the preceding steps by running the following commands:

```
# Start docker
docker run -p 8999:8999 \

# Mount delta sharing config file from Step 1 above
  --mount type=bind,source=/root/nick/pro/delta-sharing-server-
0.6.0/conf/delta-sharing-server.yaml,target=/opt/docker/conf/delta-
sharing-server.yaml \

# Mount core site xml file from Step 2 above
  --mount type=bind,source=/root/nick/pro/delta-sharing-server-
0.6.0/conf/core-site.xml,target=/opt/docker/conf/core-site.xml\
  deltaio/delta-sharing-server:0.5.2 -- --config /opt/docker/conf/delta-
sharing-server.yaml
```

5. Use a python script to get the shared data from the Delta sharing service. Create a JSON file named *local_share.json* to specify the connection information:

```
{
    "shareCredentialsVersion": 1,
    "endpoint": "http://localhost:8999/delta-sharing",
    "bearerToken": ""
}
```

External Metastore

You can use an external Apache Hive Metastore to manage Delta Lake tables in ECS buckets. To do this, install Hive in a VM.

1. Update Hive configuration with ECS details

To register tables that are stored in ECS S3 as external tables in Hive, it is necessary to change the Hive configurations.

The following code shows a sample extract from **Hive-site.xml**. Specify the Dell ECS S3 connection information such as *endpoint*, *access_key*, and *secret_key*:

```
<configuration>

  <property>
    <name>hive.input.format</name>
    <value>io.delta.hive.HiveInputFormat</value>
  </property>

  <property>
    <name>hive.tez.input.format</name>
    <value>io.delta.hive.HiveInputFormat</value>
  </property>

  <property>
    <name>hive.aux.jars.path</name>
    <value>/opt/auxlib/delta-hive-assembly_2.11-
0.6.0.jar</value>
  </property>

  <property>
    <name>fs.s3a.endpoint</name>
    <value>{endpoint}</value>
  </property>

  <property>
    <name>fs.s3a.access.key</name>
    <value>{access_key}</value>
  </property>

  <property>
    <name>fs.s3a.secret.key</name>
    <value>{secret_key}</value>
  </property>

</configuration>
```

2. Register a Delta Lake table as an external table on Hive.

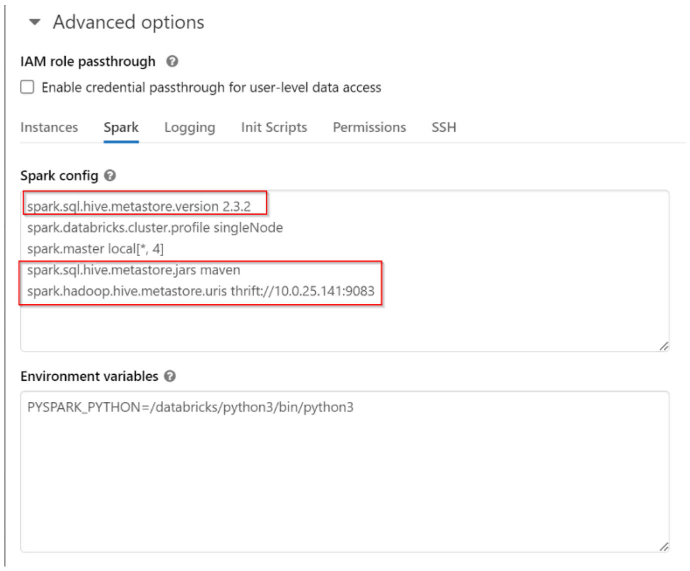
   Register a Delta Lake table that is stored in Dell ECS S3 as an external table in Hive. An external table is an s3a link pointing to a Delta table in ECS S3.

   Create a Hive CLI and load the Delta-Hive connector which is already downloaded and stored in docker images. Then run the following command in your Hive CLI to create an external table on a Hive service:

   ```
   ADD JAR /opt/auxlib/delta-hive-assembly_2.13-0.6.0.jar;
   CREATE EXTERNAL TABLE delta_test(col1 STRING)
   STORED BY 'io.delta.hive.DeltaStorageHandler'
   LOCATION 's3a://<bucket>/<folder>/<table>';
   ```

3. Register a remote Hive in Databricks.

   Configure the Databricks Spark environment by specifying the Hive version and the host and port of the Hive metastore using the Thrift protocol:

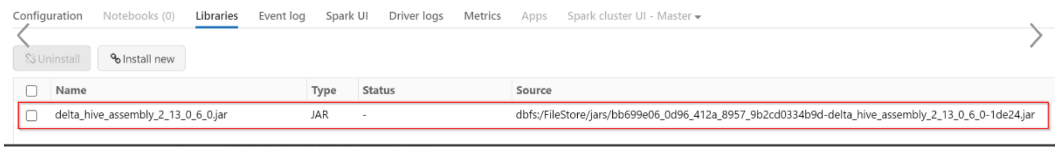


```
spark.sql.hive.metastore.version 2.3.2
spark.databricks.cluster.profile singleNode
spark.master local[*, 4]
spark.sql.hive.metastore.jars maven
spark.hadoop.hive.metastore.uris thrift://10.0.25.141:9083
```

```
PYSPARK_PYTHON=/databricks/python3/bin/python3
```

Advanced options

4. Upload the Delta-hive connector jar to the Databricks workspace and install it on the cluster.



Name	Type	Status	Source
delta_hive_assembly_2_13_0_6_0.jar	JAR	-	dbfs:/FileStore/jars/bb699e06_0d96_412a_8957_9b2cd0334b9d-delta_hive_assembly_2_13_0_6_0-1de24.jar

5. Create a Databricks notebook and run `cmd` to check the connectivity to Hive thrift.

```
%sh nc -zv 10.0.25.141 10000
```

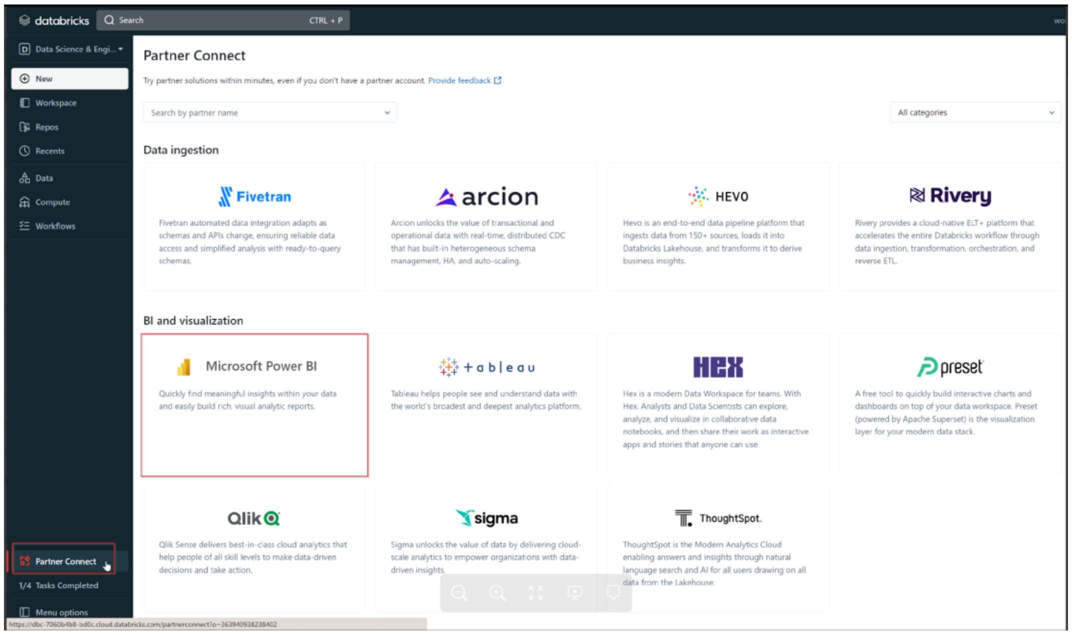
```
Connection to 10.0.25.141 10000 port [tcp/webmin] succeeded!
```

Microsoft Power BI Setup

To set up Microsoft Power BI connectivity with Databricks:

Generate a connection file to Power BI.

Click "Partner Connect in the Databricks workspace, and then click the Microsoft Power BI to open the configuration page.



Partner Connect

Try partner solutions within minutes, even if you don't have a partner account. Provide feedback

Search by partner name

All categories

Data ingestion

Fivetran

Fivetran automated data integration adapts as schemas and APIs change, ensuring reliable data access and simplified analysis with ready-to-query schemas.

arcion

Arcion unlocks the value of transactional and operational data with real-time, distributed CDC that has built-in heterogeneous schema management, HA, and auto-scaling.

HEVO

Hevo is an end-to-end data pipeline platform that ingests data from 150+ sources, loads it into Databricks Lakehouse, and transforms it to derive business insights.

Rivery

Rivery provides a cloud-native ELT+ platform that accelerates the entire Databricks workflow through data ingestion, transformation, orchestration, and reverse ETL.

BI and visualization

Microsoft Power BI

Quickly find meaningful insights within your data and easily build rich, visual analytic reports.

tableau

Tableau helps people see and understand data with the world's broadest and deepest analytics platform.

HEX

Hex is a modern Data Workspace for teams. With Hex, Analysts and Data Scientists can explore, analyze, and visualize in collaborative data notebooks, and then share their work as interactive apps and stories that anyone can use.

preset

A free tool to quickly build interactive charts and dashboards on top of your data workspace. Preset (powered by Apache Superset) is the visualization layer for your modern data stack.

Qlik

Qlik Sense delivers best-in-class cloud analytics that help people of all skill levels to make data-driven decisions and take action.

sigma

Sigma unlocks the value of data by delivering cloud-scale analytics to empower organizations with data-driven insights.

ThoughtSpot.

ThoughtSpot is the Modern Analytics Cloud enabling answers and insights through natural language search and AI for all users drawing on all data from the Lakehouse.

https://dbc-7060b4b8-bd0c.cloud.databricks.com/partnerconnect?o=263940938238402

Select the Databricks cluster that you want to share with others to process or analyze the data.

Click the **Power BI** tile.

In the **Connect to partner** dialog, for **Compute**, choose the name of the Databricks compute resource that you want to connect.

Choose **Download connection file**. Here is a sample connection file:

```
{
  "version": "0.1",
  "connections": [
    {
      "details": {
        "protocol": "databricks-sql",
        "address": {
          "host": "dbc-7060b4b8-bd0c.cloud.databricks.com",
          "path": "sql/protocolv1/o/263940938238402/0315-104711-
xqbiixvq"
        },
        "authentication": null,
        "query": null
      },
      "options": {
        "Catalog": "",
        "Database": ""
      },
      "mode": "DirectQuery"
    }
  ]
}
```

Open the downloaded connection file to launch Power BI Desktop.

Use Cases

Enterprise data remains distributed across on-premises or co-located sites, public clouds, and the edge. The choice of where data is stored depends on where:

- It is created
- It can be processed based on workloads
- It is to be consumed by different users and apps

Keeping data on-premises only limits access to rich cloud-based services and tools. It also restricts the ability to combine on-premise data with other data sets in the cloud. On the other hand, keeping all data in the cloud may run counter to existing data gravity on-premise, leading to related cost and data security issues. Enterprises should build a multicloud data architecture by design.

This paper describes several use cases whereby you can connect the data that is stored in Dell ECS object storage on-premises or in a cloud- adjacent data center such as Faction with a cloud-native Databricks Lakehouse Platform.

This paper describes five use cases:

- **Read data** stored in Dell ECS, **process** it using Databricks , and then **write** it back into Dell ECS.
- **Read data** stored in AWS S3, **process** it using the Databricks engine on cloud, and then **write** it into Dell ECS on-premises or Faction.
- Use Databricks Delta Sharing capability to **share data** residing in Dell ECS without physically moving or copying data.
- Register an external metastore in Databricks to **access data** in external tables in Dell ECS.
- **Analyze and visualize data that is stored** in Dell ECS using Databricks and Microsoft Power BI.

Use Case 1 – Read, write, and process data in Dell ECS

Read data stored in Dell ECS, process it using Databricks engine on cloud, and write it back into Dell ECS.

Open a Databricks workspace and create a notebook. Use Spark S3 modules to read data from Dell ECS and load it into a Data Frame. Use the Spark libraries to process this data and apply transformations. Finally, the data is written back into Dell ECS.

Prerequisites: See Connect to ECS from Databricks

Benefits:

- Data residing on-premises or in a co-located service such as Faction can be read and processed using Databricks cloud compute services on-demand. The processed data can then be written back into the on-premises or co-located store.
- Data does not have to physically move out of the on-premises or co-located service, enabling compliance with data localization laws,
- Rapid collaboration is enabled across Databricks environments including multiple clouds, as along with access to data from multiple cloud services, which can be useful for feature engineering and other use cases.

- ECS Storage is accessible from Databricks Serverless compute as well as classic compute enabling customers to reduce wait times, simplify network management, limit operational and monitoring overhead, and reduce total cost of ownership.

Use Case 2 – Read from AWS S3, process, and write into Dell ECS

Read data stored in AWS S3 storage, process it using Databricks engine on cloud, and then write it back into Dell ECS.

Open a Databricks workspace and create a notebook. Read data from AWS S3 storage and load it into a Data Frame. Use the Spark libraries to process this data and apply transformations. Finally, the data is written into Dell ECS.

Prerequisites: See Connect to ECS from Databricks

Benefits:

- Data residing in cloud storage can be read and processed using Databricks cloud compute services on-demand. The processed data can then be written into the on-premises or cloud adjacent data centers.
- When, for regulatory reasons, the data needs to be stored on-premises or in a co-location, this capability can be used to move raw/ processed data from the cloud to on-premises.

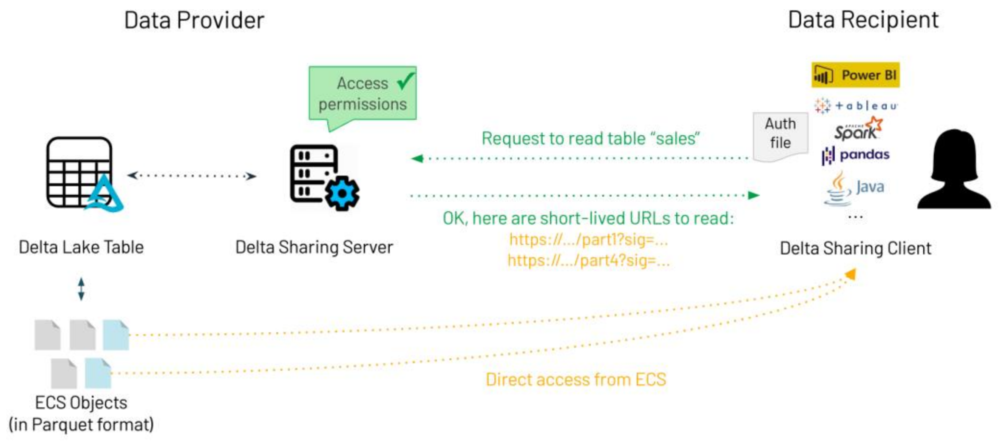
Use Case 3 – Share data from Dell ECS using Delta sharing

Use Databricks Delta Sharing to share data residing in Dell ECS without physically moving or copying data.

In this scenario a Databricks data provider has data stored on-premises in Dell ECS, and the provider wants to share this data with recipients who may not have a Databricks account. Delta sharing's simple REST protocol, and the fact that Databricks has open-sourced a reference Delta sharing server, makes it extremely simple to implement Delta sharing either as a data distributor or data consumer & recipient.

As a data provider, you generate a token and share it securely with the recipient. The recipient uses the token to authenticate and obtain read-access to the tables that are included in the shares to which you have given them access. The recipient can access the shared data using many computing tools and platforms, including Databricks, Spark, Pandas, and Power BI.

Pre-requisite: See Delta Sharing Setup



Benefits:

- If a customer wants to share data residing in an on-premises / co-location storage with recipients who may/ not have Databricks account, they can do this using open Delta Sharing.

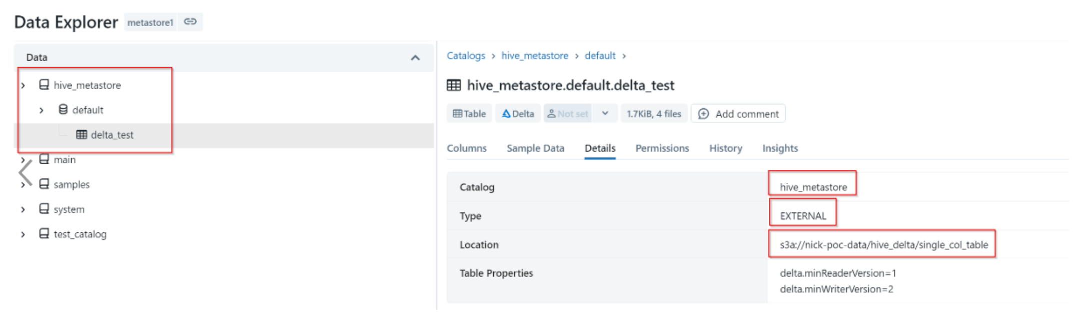
- Customers can now monetize on-premises data by sharing with willing third parties securely and easily without first moving the data to the cloud.

Use Case 4 – Register external metastore in Databricks

Use an external Hive metastore to manage Delta tables in Dell ECS. This data can be accessed from Databricks by registering the external metastore in Databricks.

Pre-requisite: See External Metastore Setup

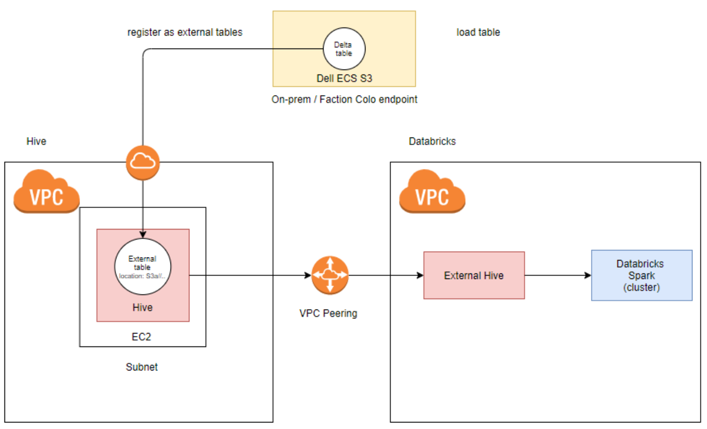
After the initial setup is complete, Hive data is viewable in Databricks Data Explorer. You can run queries on this dataset from Databricks.



hive_metastore.default.delta_test

Catalog	hive_metastore
Type	EXTERNAL
Location	s3a://nick-poc-data/hive_delta/single_col_table

Table Properties	delta.minReaderVersion=1
	delta.minWriterVersion=2


register as external tables
Delta table
Dell ECS S3
On-prem / Faction Colo endpoint
load table
Hive
VPC
External table location: S3a://...
Hive
EC2
Subnet
VPC Peering
Databricks
VPC
External Hive
Databricks Spark (cluster)

Benefits:

- Customers can continue to use existing their external metastore with Databricks without having to migrate the metadata.
- Customers can see a single, unified catalog of their data across all external locations including on-premises, simplifying data access and exploration.

Use Case 5 – Analyze and visualize data using Microsoft Power BI

Visualize data that is stored in Dell ECS using Databricks and Microsoft Power BI.

Data stored in Dell ECS can be accessed using Databricks and visualized using Microsoft Power BI. This enables hybrid operational reporting involving data situated on-premises or co-located and in the cloud. To do this, connect to an external metastore, as explained in Use Case 4.

Launch Microsoft Power BI Desktop and enter your authentication credentials:

- Personal Access Token: Enter your Databricks personal access token.
- Username/Password: Enter your Databricks username (typically your email address) and password

Click **Connect**. Select the Databricks data to query from the Power BI Navigator.

Pre-requisite: See Microsoft Power BI Setup

Benefits:

- The Databricks query engine can read data from Dell ECS store on-premises or co-located and visualized using the Power BI dashboards.
- This use case can also be extended to combine data in ECS with additional data in native Databricks storage. The final output can be visualized using Power BI dashboards.
- End users of the Power BI dashboards are abstracted from the complexity of data location and formats across on-premises and cloud.

Benefits

The above use cases unlock several benefits for various data user personas:

- **Compute on demand** – Data engineers, data analysts, and data scientists can access compute on demand with Databricks and use it to process data that is located on-premises or in a cloud-adjacent data center such as Faction.
- **Data sharing** – Users can share data inside and outside the organization using Databricks Delta Sharing, without the need to physically make copies of the data.
- **Reduced data movement complexity** – Data platform administrators and data engineers can reduce the costs and complexity of moving data between on-premises and cloud and maintaining multiple copies of data.
- **Simplify data governance** – Data stewards and privacy experts can enforce data localization policies by having tighter control on movement of data across premises and regions. The shared responsibility model brings the best of both worlds – on-premises or co-located sites and cloud.
- **Adopt open architecture and avoid vendor lock-in** – CDOs and CIOs can accelerate adoption of open architecture with open data formats, , avoiding vendor lock in and decoupling storage and compute.
- **Build multicloud resiliency** – Businesses can rely on a common store in on-premises or co-located sites that will seamlessly integrate with multiple cloud providers or regions of any given cloud provider, thereby avoiding service downtimes if there are outages in any given cloud or region, with a single copy of data.

Conclusion

It is common to maintain large on-premises data repositories for a range of business purposes. Historically, the data in these repositories was only available to the applications and infrastructures running on-premises. Today, these repositories can be accessed by applications running natively in the cloud through high-speed cloud-adjacent connectivity, and the data can be used for business insights or long-term research. This represents a major change to how businesses have traditionally thought about using their data. It is a key value-add to organizations thinking about implementing multicloud architectures to simplify their workflows and optimize their data investments while delivering critical business insights.

Get a demo of this solution

The Dell Technologies Customer Solution Center (CSC) is an important supporting organization for your analytics implementation. The CSC uses this joint solution and many other solutions to accelerate achieving your goals and realize your digital future.

Contact your Dell Technologies Sales Representative today to schedule a customized briefing or solutions engagement for the Dell ECS with Databricks solution described in this paper.

- **Proof of Concept** – Validate that your preferred solution meets your needs with a custom Proof of Concept. Dell Technologies solution architects enable practical, firsthand implementation based on your test cases.
- **Design Session** – Collaborate with Dell Technologies experts to design a solution framework. Brainstorm with Dell experts to explore your current IT environment, your future objectives, and potential solutions.
- **Technical Deep Dive** – Dive into the technical solution details that you are considering for your organization. Learn from live product demonstrations and solution-focused discussions with Dell Technologies subject matter experts.

Why Dell Technologies

Dell Technologies has been a leader in the big data and advanced analytics space for more than a decade, with proven products, solutions, and expertise. Dell Technologies has teams of experts specializing in applications, infrastructure, and data management who are dedicated to staying at the forefront of latest technologies and tuning solutions for your applications to help you keep pace with this constantly evolving landscape.

Dell Technologies is building a broad ecosystem of partners in the data space to bring the necessary experts, resources, and capabilities to customers and accelerate their data strategy. Dell Technologies believes that customers should be able to innovate using data irrespective of where it resides across on-premises, public cloud, and the edge. By partnering with Databricks, an industry leader in data management and analytics, Dell Technologies is creating optimized solutions for customers. Dell Technologies uniquely provides an extensive portfolio of technologies to deliver the advanced infrastructures that underpin successful data implementations. With years of experience and an ecosystem of curated technology and service partners, Dell Technologies provides innovative solutions, servers, networking, storage, workstations, and services that reduce complexity and enable you to capitalize on a universe of data.

References

Dell Technologies documentation

The following Dell Technologies documentation provides information related to the solution that this document describes. Access depends on your login credentials. If you do not have access to a document, contact your Dell Technologies representative.

- Dell ECS Info Hub

Databricks documentation

The following Databricks documentation provides detailed information that is relevant to this solution:

- What is Delta Lake?
- Unity Catalog
- Delta Sharing
- External tables
- External Apache Hive metastore (legacy)

Faction Inc. documentation

The following website provides information about Faction's cloud data services with Dell Technologies:

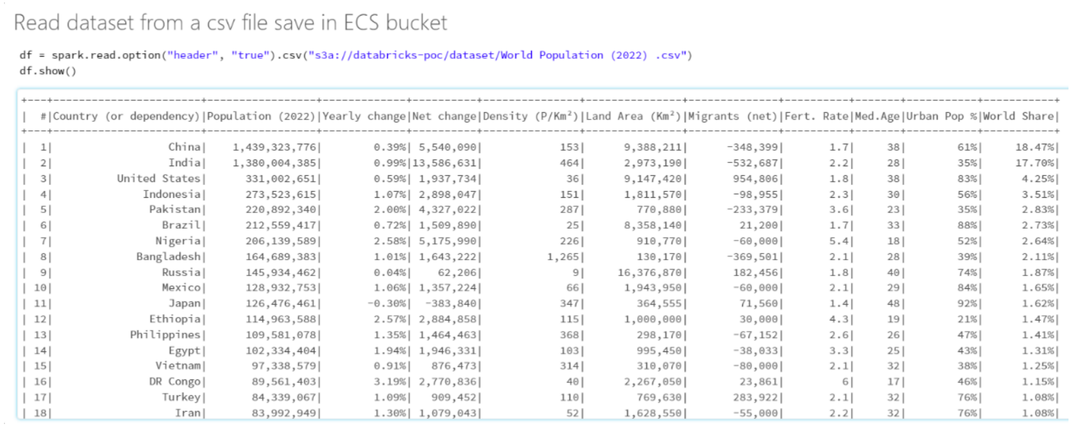
- Faction

Appendix

This section provides sample code snippets and the output generated for each of the use cases mentioned above, for quick reference.

Use Case 1

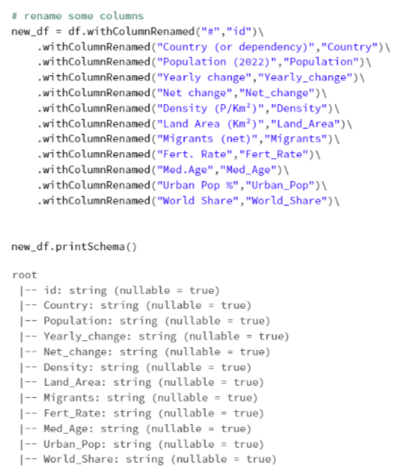
Read data stored in Dell ECS, process it using Databricks engine on cloud, and then write it back into Dell ECS.



Read dataset from a csv file save in ECS bucket

```
df = spark.read.option("header", "true").csv("s3a://databricks-poc/dataset/World Population (2022) .csv")
df.show()
```

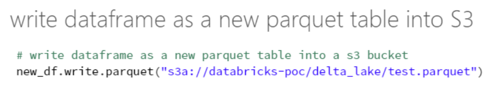
```
+---+-----------------------+-----------------+-------------+----------+---------------+---------------+--------------+----------+-------+-----------+-----------+
|  #|Country (or dependency)|Population (2022)|Yearly change|Net change|Density (P/Km²)|Land Area (Km²)|Migrants (net)|Fert. Rate|Med.Age|Urban Pop %|World Share|
+---+-----------------------+-----------------+-------------+----------+---------------+---------------+--------------+----------+-------+-----------+-----------+
|  1|                  China|    1,439,323,776|        0.39%| 5,540,090|            153|      9,388,211|      -348,399|       1.7|     38|        61%|     18.47%|
|  2|                  India|    1,380,004,385|        0.99%|13,586,631|            464|      2,973,190|      -532,687|       2.2|     28|        35%|     17.70%|
|  3|          United States|      331,002,651|        0.59%| 1,937,734|             36|      9,147,420|       954,806|       1.8|     38|        83%|      4.25%|
|  4|              Indonesia|      273,523,615|        1.07%| 2,898,047|            151|      1,811,570|       -98,955|       2.3|     30|        56%|      3.51%|
|  5|               Pakistan|      220,892,340|        2.00%| 4,327,022|            287|        770,880|      -233,379|       3.6|     23|        35%|      2.83%|
|  6|                 Brazil|      212,559,417|        0.72%| 1,509,890|             25|      8,358,140|        21,200|       1.7|     33|        88%|      2.73%|
|  7|                Nigeria|      206,139,589|        2.58%| 5,175,990|            226|        910,770|       -60,000|       5.4|     18|        52%|      2.64%|
|  8|             Bangladesh|      164,689,383|        1.01%| 1,643,222|          1,265|        130,170|      -369,501|       2.1|     28|        39%|      2.11%|
|  9|                 Russia|      145,934,462|        0.04%|    62,206|              9|     16,376,870|       182,456|       1.8|     40|        74%|      1.87%|
| 10|                 Mexico|      128,932,753|        1.06%| 1,357,224|             66|      1,943,950|       -60,000|       2.1|     29|        84%|      1.65%|
| 11|                  Japan|      126,476,461|       -0.30%|  -383,840|            347|        364,555|        71,560|       1.4|     48|        92%|      1.62%|
| 12|               Ethiopia|      114,963,588|        2.57%| 2,884,858|            115|      1,000,000|        30,000|       4.3|     19|        21%|      1.47%|
| 13|            Philippines|      109,581,078|        1.35%| 1,464,463|            368|        298,170|       -67,152|       2.6|     26|        47%|      1.41%|
| 14|                  Egypt|      102,334,404|        1.94%| 1,946,331|            103|        995,450|       -38,033|       3.3|     25|        43%|      1.31%|
| 15|                Vietnam|       97,338,579|        0.91%|   876,473|            314|        310,070|       -80,000|       2.1|     32|        38%|      1.25%|
| 16|               DR Congo|       89,561,403|        3.19%| 2,770,836|             40|      2,267,050|        23,861|         6|     17|        46%|      1.15%|
| 17|                 Turkey|       84,339,067|        1.09%|   909,452|            110|        769,630|       283,922|       2.1|     32|        76%|      1.08%|
| 18|                   Iran|       83,992,949|        1.30%| 1,079,043|             52|      1,628,550|       -55,000|       2.2|     32|        76%|      1.08%|
```



```
# rename some columns
new_df = df.withColumnRenamed("#","id")\
    .withColumnRenamed("Country (or dependency)","Country")\
    .withColumnRenamed("Population (2022)","Population")\
    .withColumnRenamed("Yearly change","Yearly_change")\
    .withColumnRenamed("Net change","Net_change")\
    .withColumnRenamed("Density (P/Km²)","Density")\
    .withColumnRenamed("Land Area (Km²)","Land_Area")\
    .withColumnRenamed("Migrants (net)","Migrants")\
    .withColumnRenamed("Fert. Rate","Fert_Rate")\
    .withColumnRenamed("Med.Age","Med_Age")\
    .withColumnRenamed("Urban Pop %","Urban_Pop")\
    .withColumnRenamed("World Share","World_Share")\
```

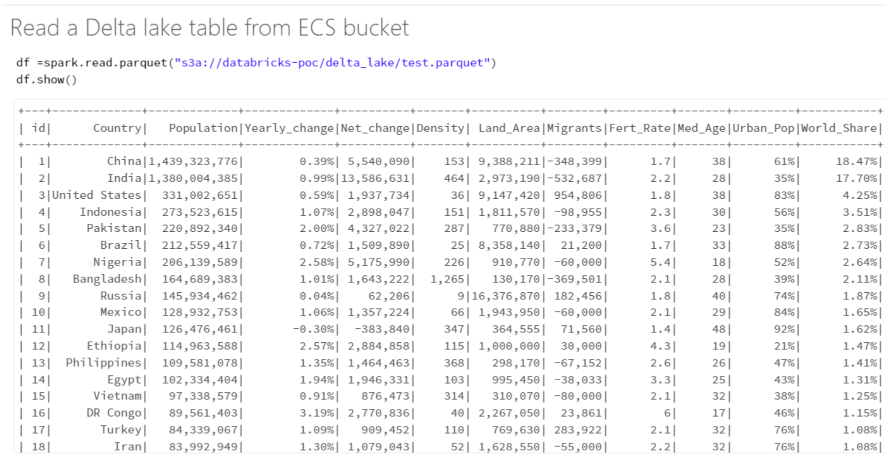
```
new_df.printSchema()
```

```
root
 |-- id: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Population: string (nullable = true)
 |-- Yearly_change: string (nullable = true)
 |-- Net_change: string (nullable = true)
 |-- Density: string (nullable = true)
 |-- Land_Area: string (nullable = true)
 |-- Migrants: string (nullable = true)
 |-- Fert_Rate: string (nullable = true)
 |-- Med_Age: string (nullable = true)
 |-- Urban_Pop: string (nullable = true)
 |-- World_Share: string (nullable = true)
```



write dataframe as a new parquet table into S3

```
# write dataframe as a new parquet table into a s3 bucket
new_df.write.parquet("s3a://databricks-poc/delta_lake/test.parquet")
```



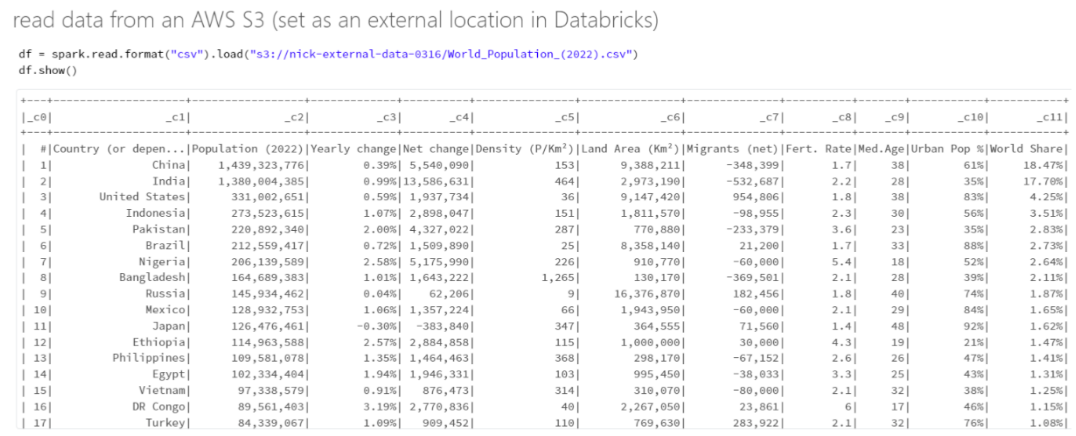
Read a Delta lake table from ECS bucket

```
df =spark.read.parquet("s3a://databricks-poc/delta_lake/test.parquet")
df.show()
```

```
+---+-------------+-------------+-------------+----------+-------+----------+--------+---------+-------+---------+-----------+
| id|      Country|   Population|Yearly_change|Net_change|Density| Land_Area|Migrants|Fert_Rate|Med_Age|Urban_Pop|World_Share|
+---+-------------+-------------+-------------+----------+-------+----------+--------+---------+-------+---------+-----------+
|  1|        China|1,439,323,776|        0.39%| 5,540,090|    153| 9,388,211|-348,399|      1.7|     38|      61%|     18.47%|
|  2|        India|1,380,004,385|        0.99%|13,586,631|    464| 2,973,190|-532,687|      2.2|     28|      35%|     17.70%|
|  3|United States|  331,002,651|        0.59%| 1,937,734|     36| 9,147,420| 954,806|      1.8|     38|      83%|      4.25%|
|  4|    Indonesia|  273,523,615|        1.07%| 2,898,047|    151| 1,811,570| -98,955|      2.3|     30|      56%|      3.51%|
|  5|     Pakistan|  220,892,340|        2.00%| 4,327,022|    287|   770,880|-233,379|      3.6|     23|      35%|      2.83%|
|  6|       Brazil|  212,559,417|        0.72%| 1,509,890|     25| 8,358,140|  21,200|      1.7|     33|      88%|      2.73%|
|  7|      Nigeria|  206,139,589|        2.58%| 5,175,990|    226|   910,770| -60,000|      5.4|     18|      52%|      2.64%|
|  8|   Bangladesh|  164,689,383|        1.01%| 1,643,222|  1,265|   130,170|-369,501|      2.1|     28|      39%|      2.11%|
|  9|       Russia|  145,934,462|        0.04%|    62,206|      9|16,376,870| 182,456|      1.8|     40|      74%|      1.87%|
| 10|       Mexico|  128,932,753|        1.06%| 1,357,224|     66| 1,943,950| -60,000|      2.1|     29|      84%|      1.65%|
| 11|        Japan|  126,476,461|       -0.30%|  -383,840|    347|   364,555|  71,560|      1.4|     48|      92%|      1.62%|
| 12|     Ethiopia|  114,963,588|        2.57%| 2,884,858|    115| 1,000,000|  30,000|      4.3|     19|      21%|      1.47%|
| 13|  Philippines|  109,581,078|        1.35%| 1,464,463|    368|   298,170| -67,152|      2.6|     26|      47%|      1.41%|
| 14|        Egypt|  102,334,404|        1.94%| 1,946,331|    103|   995,450| -38,033|      3.3|     25|      43%|      1.31%|
| 15|      Vietnam|   97,338,579|        0.91%|   876,473|    314|   310,070| -80,000|      2.1|     32|      38%|      1.25%|
| 16|     DR Congo|   89,561,403|        3.19%| 2,770,836|     40| 2,267,050|  23,861|        6|     17|      46%|      1.15%|
| 17|       Turkey|   84,339,067|        1.09%|   909,452|    110|   769,630| 283,922|      2.1|     32|      76%|      1.08%|
| 18|         Iran|   83,992,949|        1.30%| 1,079,043|     52| 1,628,550| -55,000|      2.2|     32|      76%|      1.08%|
```

Use Case 2

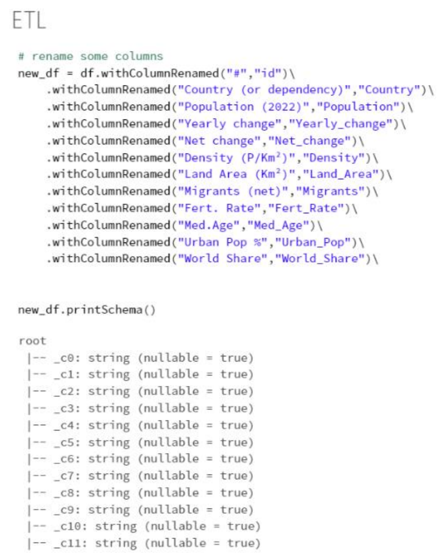
Read data stored in AWS S3 storage, process it using the Databricks engine on cloud, and then write it back into Dell ECS.



read data from an AWS S3 (set as an external location in Databricks)

```
df = spark.read.format("csv").load("s3://nick-external-data-0316/World_Population_(2022).csv")
df.show()
```

```
+---+--------------------+-----------------+-------------+----------+---------------+---------------+--------------+----------+-------+-----------+-----------+
|_c0|                 _c1|              _c2|          _c3|       _c4|            _c5|            _c6|           _c7|       _c8|    _c9|       _c10|       _c11|
+---+--------------------+-----------------+-------------+----------+---------------+---------------+--------------+----------+-------+-----------+-----------+
|  #|Country (or depen...|Population (2022)|Yearly change|Net change|Density (P/Km²)|Land Area (Km²)|Migrants (net)|Fert. Rate|Med.Age|Urban Pop %|World Share|
|  1|               China|    1,439,323,776|        0.39%| 5,540,090|            153|      9,388,211|      -348,399|       1.7|     38|        61%|     18.47%|
|  2|               India|    1,380,004,385|        0.99%|13,586,631|            464|      2,973,190|      -532,687|       2.2|     28|        35%|     17.70%|
|  3|       United States|      331,002,651|        0.59%| 1,937,734|             36|      9,147,420|       954,806|       1.8|     38|        83%|      4.25%|
|  4|           Indonesia|      273,523,615|        1.07%| 2,898,047|            151|      1,811,570|       -98,955|       2.3|     30|        56%|      3.51%|
|  5|            Pakistan|      220,892,340|        2.00%| 4,327,022|            287|        770,880|      -233,379|       3.6|     23|        35%|      2.83%|
|  6|              Brazil|      212,559,417|        0.72%| 1,509,890|             25|      8,358,140|        21,200|       1.7|     33|        88%|      2.73%|
|  7|             Nigeria|      206,139,589|        2.58%| 5,175,990|            226|        910,770|       -60,000|       5.4|     18|        52%|      2.64%|
|  8|          Bangladesh|      164,689,383|        1.01%| 1,643,222|          1,265|        130,170|      -369,501|       2.1|     28|        39%|      2.11%|
|  9|              Russia|      145,934,462|        0.04%|    62,206|              9|     16,376,870|       182,456|       1.8|     40|        74%|      1.87%|
| 10|              Mexico|      128,932,753|        1.06%| 1,357,224|             66|      1,943,950|       -60,000|       2.1|     29|        84%|      1.65%|
| 11|               Japan|      126,476,461|       -0.30%|  -383,840|            347|        364,555|        71,560|       1.4|     48|        92%|      1.62%|
| 12|            Ethiopia|      114,963,588|        2.57%| 2,884,858|            115|      1,000,000|        30,000|       4.3|     19|        21%|      1.47%|
| 13|         Philippines|      109,581,078|        1.35%| 1,464,463|            368|        298,170|       -67,152|       2.6|     26|        47%|      1.41%|
| 14|               Egypt|      102,334,404|        1.94%| 1,946,331|            103|        995,450|       -38,033|       3.3|     25|        43%|      1.31%|
| 15|             Vietnam|       97,338,579|        0.91%|   876,473|            314|        310,070|       -80,000|       2.1|     32|        38%|      1.25%|
| 16|            DR Congo|       89,561,403|        3.19%| 2,770,836|             40|      2,267,050|        23,861|         6|     17|        46%|      1.15%|
| 17|              Turkey|       84,339,067|        1.09%|   909,452|            110|        769,630|       283,922|       2.1|     32|        76%|      1.08%|
```

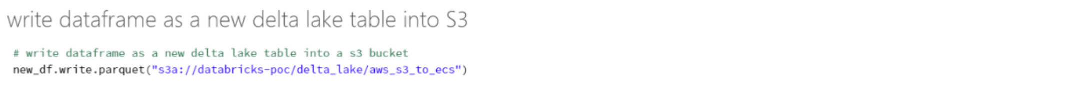


ETL

```
# rename some columns
new_df = df.withColumnRenamed("#","id")\
    .withColumnRenamed("Country (or dependency)","Country")\
    .withColumnRenamed("Population (2022)","Population")\
    .withColumnRenamed("Yearly change","Yearly_change")\
    .withColumnRenamed("Net change","Net_change")\
    .withColumnRenamed("Density (P/Km²)","Density")\
    .withColumnRenamed("Land Area (Km²)","Land_Area")\
    .withColumnRenamed("Migrants (net)","Migrants")\
    .withColumnRenamed("Fert. Rate","Fert_Rate")\
    .withColumnRenamed("Med.Age","Med_Age")\
    .withColumnRenamed("Urban Pop %","Urban_Pop")\
    .withColumnRenamed("World Share","World_Share")\
```

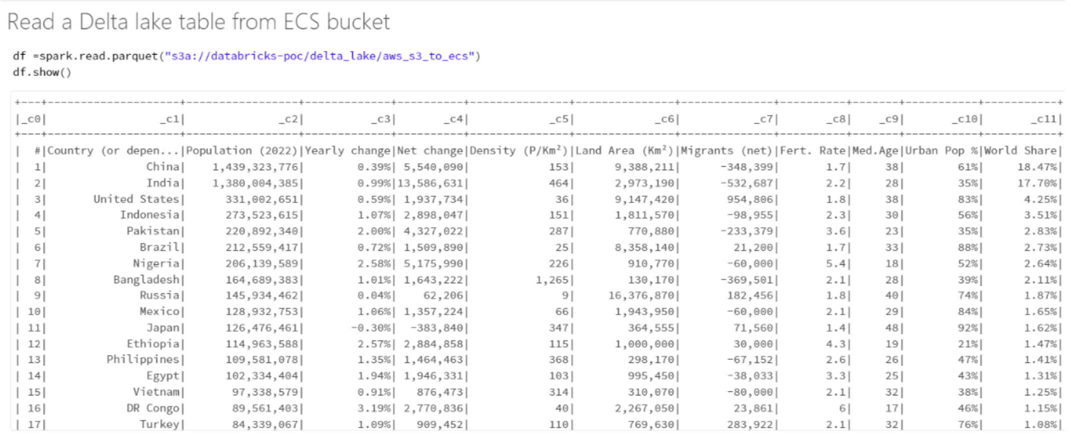
```
new_df.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: string (nullable = true)
 |-- _c10: string (nullable = true)
 |-- _c11: string (nullable = true)
```



write dataframe as a new delta lake table into S3

```
# write dataframe as a new delta lake table into a s3 bucket
new_df.write.parquet("s3a://databricks-poc/delta_lake/aws_s3_to_ecs")
```



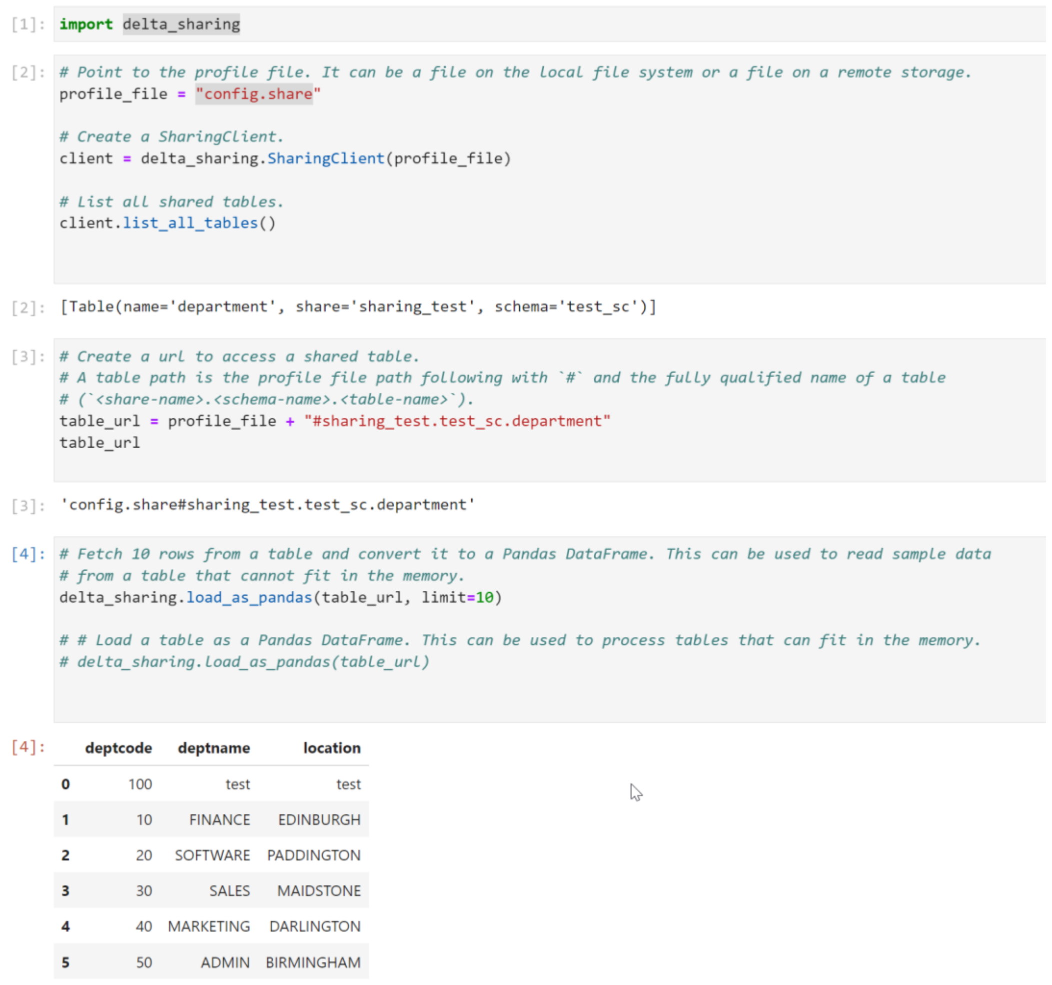
Read a Delta lake table from ECS bucket

```
df =spark.read.parquet("s3a://databricks-poc/delta_lake/aws_s3_to_ecs")
df.show()
```

```
+---+--------------------+-----------------+-------------+----------+---------------+---------------+--------------+----------+-------+-----------+-----------+
|_c0|                 _c1|              _c2|          _c3|       _c4|            _c5|            _c6|           _c7|       _c8|    _c9|       _c10|       _c11|
+---+--------------------+-----------------+-------------+----------+---------------+---------------+--------------+----------+-------+-----------+-----------+
|  #|Country (or depen...|Population (2022)|Yearly change|Net change|Density (P/Km²)|Land Area (Km²)|Migrants (net)|Fert. Rate|Med.Age|Urban Pop %|World Share|
|  1|               China|    1,439,323,776|        0.39%| 5,540,090|            153|      9,388,211|      -348,399|       1.7|     38|        61%|     18.47%|
|  2|               India|    1,380,004,385|        0.99%|13,586,631|            464|      2,973,190|      -532,687|       2.2|     28|        35%|     17.70%|
|  3|       United States|      331,002,651|        0.59%| 1,937,734|             36|      9,147,420|       954,806|       1.8|     38|        83%|      4.25%|
|  4|           Indonesia|      273,523,615|        1.07%| 2,898,047|            151|      1,811,570|       -98,955|       2.3|     30|        56%|      3.51%|
|  5|            Pakistan|      220,892,340|        2.00%| 4,327,022|            287|        770,880|      -233,379|       3.6|     23|        35%|      2.83%|
|  6|              Brazil|      212,559,417|        0.72%| 1,509,890|             25|      8,358,140|        21,200|       1.7|     33|        88%|      2.73%|
|  7|             Nigeria|      206,139,589|        2.58%| 5,175,990|            226|        910,770|       -60,000|       5.4|     18|        52%|      2.64%|
|  8|          Bangladesh|      164,689,383|        1.01%| 1,643,222|          1,265|        130,170|      -369,501|       2.1|     28|        39%|      2.11%|
|  9|              Russia|      145,934,462|        0.04%|    62,206|              9|     16,376,870|       182,456|       1.8|     40|        74%|      1.87%|
| 10|              Mexico|      128,932,753|        1.06%| 1,357,224|             66|      1,943,950|       -60,000|       2.1|     29|        84%|      1.65%|
| 11|               Japan|      126,476,461|       -0.30%|  -383,840|            347|        364,555|        71,560|       1.4|     48|        92%|      1.62%|
| 12|            Ethiopia|      114,963,588|        2.57%| 2,884,858|            115|      1,000,000|        30,000|       4.3|     19|        21%|      1.47%|
| 13|         Philippines|      109,581,078|        1.35%| 1,464,463|            368|        298,170|       -67,152|       2.6|     26|        47%|      1.41%|
| 14|               Egypt|      102,334,404|        1.94%| 1,946,331|            103|        995,450|       -38,033|       3.3|     25|        43%|      1.31%|
| 15|             Vietnam|       97,338,579|        0.91%|   876,473|            314|        310,070|       -80,000|       2.1|     32|        38%|      1.25%|
| 16|            DR Congo|       89,561,403|        3.19%| 2,770,836|             40|      2,267,050|        23,861|         6|     17|        46%|      1.15%|
| 17|              Turkey|       84,339,067|        1.09%|   909,452|            110|        769,630|       283,922|       2.1|     32|        76%|      1.08%|
```

Use Case 3

Use Databricks Delta Sharing to share data residing in Dell ECS without physically moving or copying data:



```
[1]: import delta_sharing
```

```
[2]: # Point to the profile file. It can be a file on the local file system or a file on a remote storage.
     profile_file = "config.share"

     # Create a SharingClient.
     client = delta_sharing.SharingClient(profile_file)

     # List all shared tables.
     client.list_all_tables()
```

```
[2]: [Table(name='department', share='sharing_test', schema='test_sc')]
```

```
[3]: # Create a url to access a shared table.
     # A table path is the profile file path following with `#` and the fully qualified name of a table
     # (`<share-name>.<schema-name>.<table-name>`).
     table_url = profile_file + "#sharing_test.test_sc.department"
     table_url
```

```
[3]: 'config.share#sharing_test.test_sc.department'
```

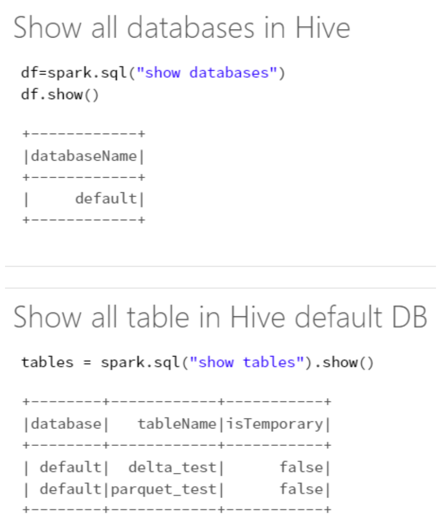
```
[4]: # Fetch 10 rows from a table and convert it to a Pandas DataFrame. This can be used to read sample data
     # from a table that cannot fit in the memory.
     delta_sharing.load_as_pandas(table_url, limit=10)

     # # Load a table as a Pandas DataFrame. This can be used to process tables that can fit in the memory.
     # delta_sharing.load_as_pandas(table_url)
```

[4]:

	deptcode	deptname	location
0	100	test	test
1	10	FINANCE	EDINBURGH
2	20	SOFTWARE	PADDINGTON
3	30	SALES	MAIDSTONE
4	40	MARKETING	DARLINGTON
5	50	ADMIN	BIRMINGHAM

Use Case 4

Access data in Dell ECS with external Hive metastore registered in Databricks

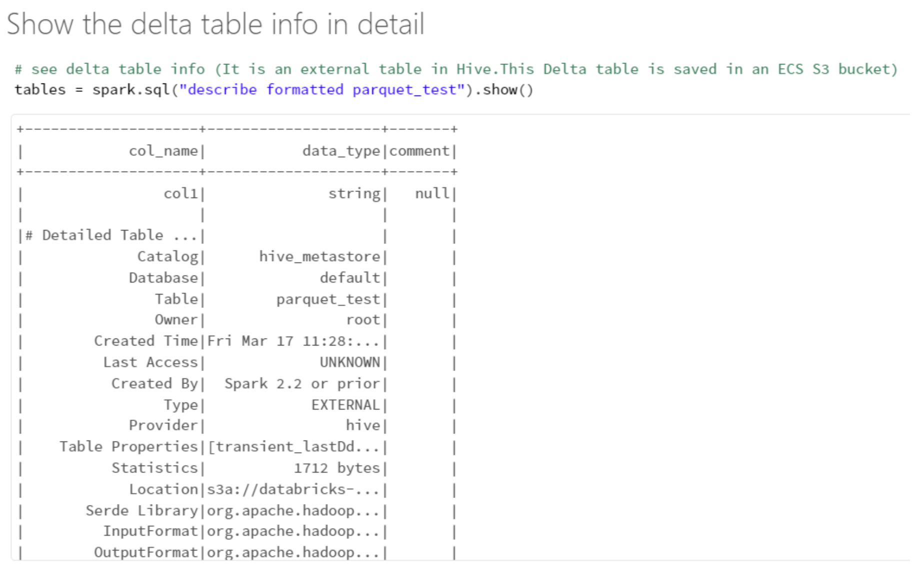


Show all databases in Hive

```
df=spark.sql("show databases")
df.show()

+------------+
|databaseName|
+------------+
|     default|
+------------+
```

Show all table in Hive default DB

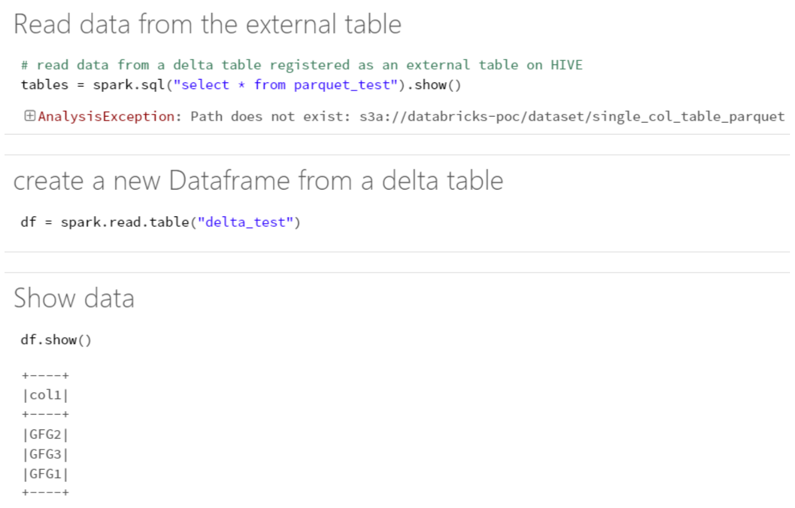
```
tables = spark.sql("show tables").show()

+--------+------------+-----------+
|database|   tableName|isTemporary|
+--------+------------+-----------+
| default|  delta_test|      false|
| default|parquet_test|      false|
+--------+------------+-----------+
```



Show the delta table info in detail

```
# see delta table info (It is an external table in Hive.This Delta table is saved in an ECS S3 bucket)
tables = spark.sql("describe formatted parquet_test").show()
```

```
+--------------------+--------------------+-------+
|            col_name|           data_type|comment|
+--------------------+--------------------+-------+
|                col1|              string|   null|
|                    |                    |       |
|# Detailed Table ...|                    |       |
|             Catalog|      hive_metastore|       |
|            Database|             default|       |
|               Table|        parquet_test|       |
|               Owner|                root|       |
|        Created Time|Fri Mar 17 11:28:...|       |
|         Last Access|             UNKNOWN|       |
|          Created By|  Spark 2.2 or prior|       |
|                Type|            EXTERNAL|       |
|            Provider|                hive|       |
|    Table Properties|[transient_lastDd...|       |
|          Statistics|          1712 bytes|       |
|            Location|s3a://databricks-...|       |
|       Serde Library|org.apache.hadoop...|       |
|         InputFormat|org.apache.hadoop...|       |
|        OutputFormat|org.apache.hadoop...|       |
```



Read data from the external table

```
# read data from a delta table registered as an external table on HIVE
tables = spark.sql("select * from parquet_test").show()
```

```
⊞AnalysisException: Path does not exist: s3a://databricks-poc/dataset/single_col_table_parquet
```

create a new Dataframe from a delta table

```
df = spark.read.table("delta_test")
```

Show data

```
df.show()
```

```
+----+
|col1|
+----+
|GFG2|
|GFG3|
|GFG1|
+----+
```

Databricks

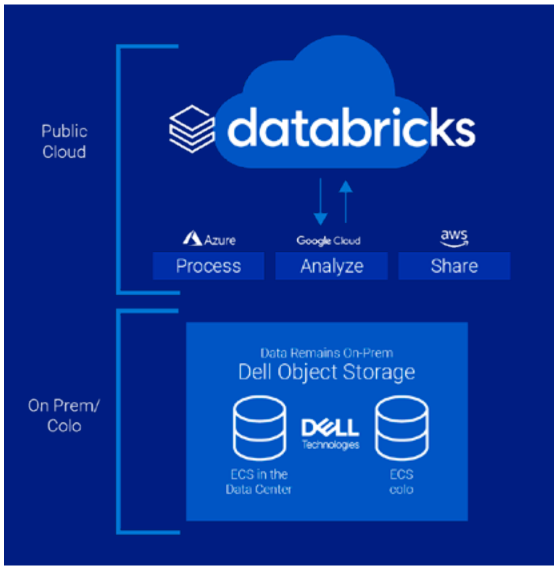
Databricks, the data and AI company, was founded in 2013 by the original creators of Apache Spark™, Delta Lake and MLflow. The Databricks Lakehouse Platform combines the best elements of data lakes and data warehouses to deliver the reliability, strong governance and performance of data warehouses with the openness, flexibility and machine learning support of data lakes. As the world's first and only lakehouse platform in the cloud, Databricks offers a simple, open and multi-cloud platform to unify customer's for data.

Delta Lake

Delta Lake is an open format storage layer that delivers reliability, security and performance on your data lake — for both streaming and batch operations. By replacing data silos with a single home for structured, semi-structured and unstructured data, Delta Lake is the foundation of a cost-effective, highly scalable lakehouse.

Delta Sharing

Delta Sharing provides an open solution to securely share live data from Databricks Lakehouse Platform to any computing platform.



Dell Technologies

Dell Technologies helps organizations and individuals build their digital future and transform how they work, live and play. The company provides customers with the industry's broadest and most innovative technology and services portfolio for the data era.

Dell ECS Enterprise Object Storage

ECS is the leading object storage platform from Dell Technologies, with unmatched scalability, performance, resilience, and economics. ECS delivers rich S3-compatibility on a globally distributed architecture, empowering organizations to support enterprise workloads such as cloud-native, archive, IoT, AI, and big data analytics applications at scale.

Dell Data Management

To provide the necessary flexibility, choice and interoperability across tools in the data space, Dell has curated a diverse ecosystem of leading technologies so that you never get locked in while you continue to innovate with your data. As a trusted partner for essential infrastructure, Dell can help activate data and accelerate time to insights by bridging data silos across multi-cloud and edge. By moving analytics closer to the point of data creation through innovative edge data solutions, we enable enterprises to be more agile.

Learn more about Dell Data Management solutions

Contact a Dell Technologies Expert

View more resources

Join the conversation with #DellKnowsData