DATA+AI SUMMIT BY databricks

Architecting Delta Sharing for large Scale

Bilal Obeidat, Bhavin Kukadia
June 2024

Presenters

Bilal Obeidat

Principal Solution Architect

- Brickester since 2017.
- Work closely with field eng, clients and product team
- Live in seattle and father of 3 kids
- Work hard and have fun

Bhavin Kukadia

Principal Solution Architect

- Brickster since 2017
- Hands on experience all things platform security across all of databricks public cloud deployments
- Long walks, bourbon and craft beer’s
- Work hard, have fun, make money

Key Objectives

- Architecting Delta Sharing for Success
- Understanding the Security Internals of Delta Sharing
- Storage Configuration of Delta Sharing
- Knowledge of D2O: The Platform-Agnostic Capability

Sharing and collaboration is a critical imperative for all customers

R&D and Clinical Research

Advertising & Marketing

Commercialization

Financial Markets

Supply Chain & Operations

Regulatory & Reporting

HR

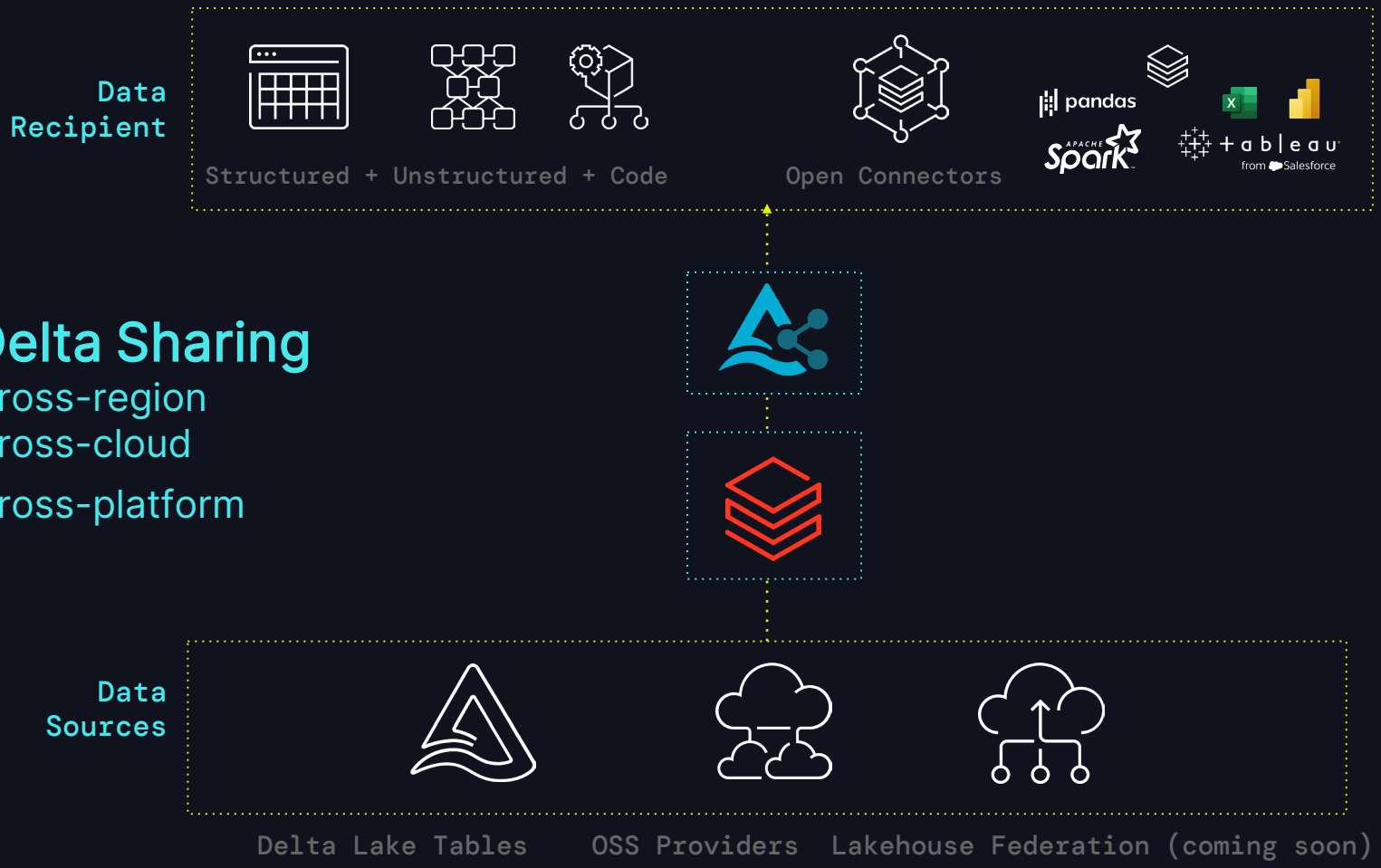
Customers want to access a wide ecosystem of data partners…

- Data Licensing from data vendors
- SaaS Platform Zero-Copy Bi-Directional Sharing
- Peer-to-Peer Sharing & Collaboration
- Internal sharing across Business Units

…but face important challenges

Every vendor/partner has a different delivery method	Data must be moved and copied around	Data fragmented across different platforms	Can’t share non-structured data assets
↓	↓	↓	↓
Slow onboarding of new data	**Data duplication, latency, and cost**	**Platform silos limit innovation**	**Can’t unlock AI use-cases**

An open approach to sharing is critical to maximize reach and impact



Delta Sharing Ecosystem is Growing Fast

16K+

data recipients using Delta Sharing

+300% YoY

growth in active Delta Shares

40%

active Delta Shares use open connectors

New Sharing and Marketplace Partners

Industry-leading partners adopting Databricks for collaboration

Databricks' Open Sharing Partner Ecosystem

Broadest ecosystem without any vendor or platform limitations

Architecting Data Sharing Strategies

Architecting for different use cases

Accessing Telemetric data

ML team in a separate region

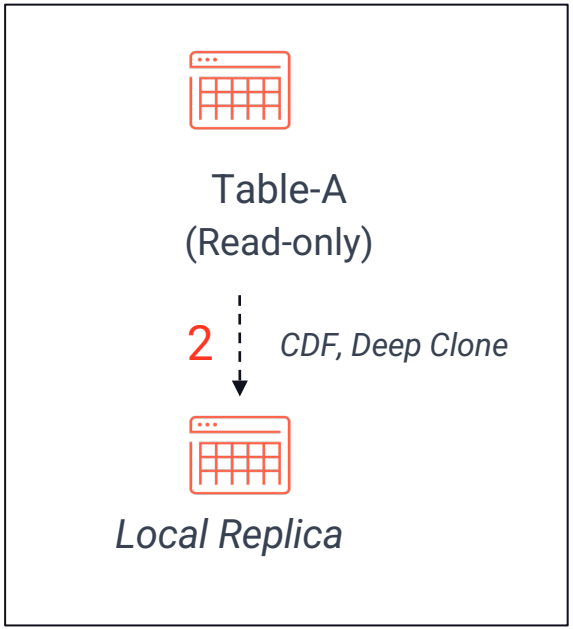
Live Sharing vs. Data Replication

Provider

Live Sharing

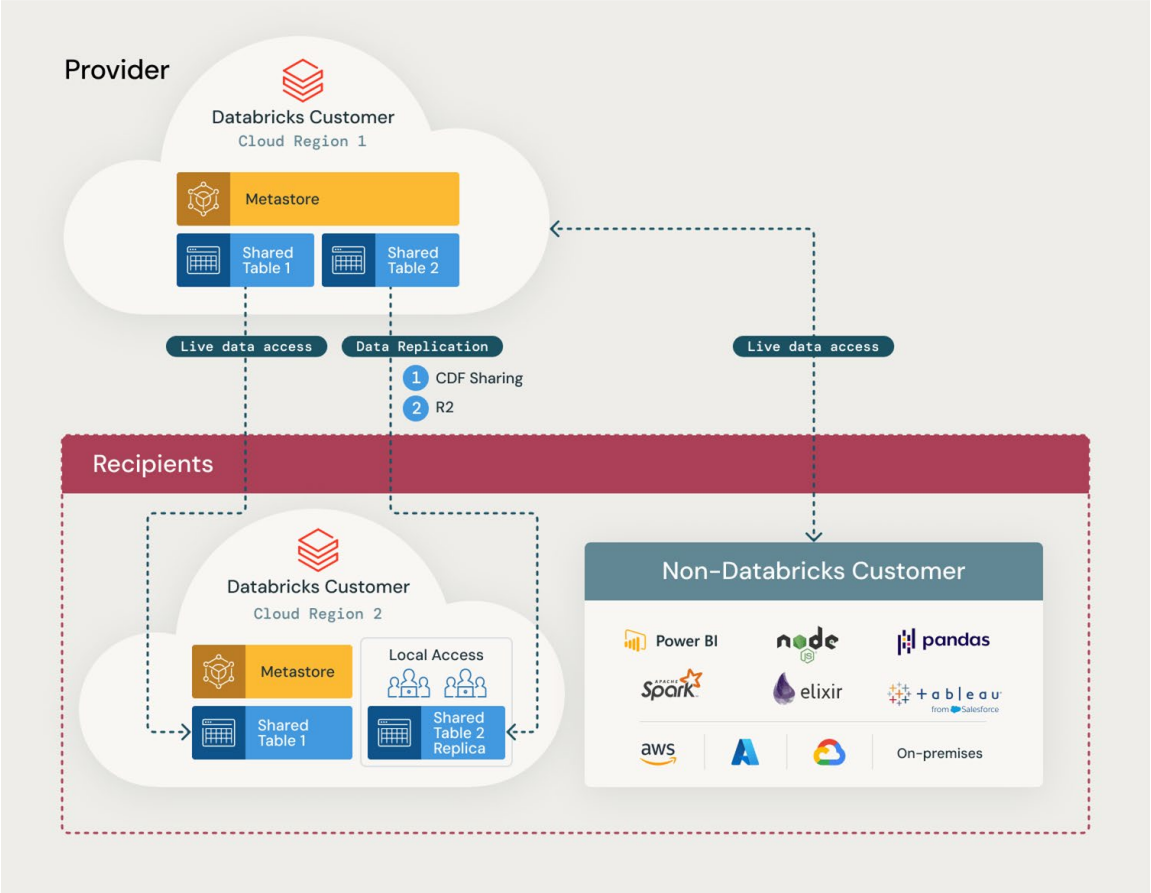
Trade-offs:

- Egress Cost
- Implementation effort of creating a local copy
- Performance
- Data Timeliness



Recipient

Global Data Sharing



1. Change Data Feed (CDF)

Configuration

- Recipient incrementally constructs a local copy using CDF
- Requires sharing table history.
- Enable Change Data Feed (CDF) explicitly in setup code.
- Set table property delta.enableChangeDataFeed = true using Create/Alter table commands.

Manage Egress, enhance performance

```
ALTER SHARE flights_data_share
ADD TABLE db_flights.flights
AS db_flights.flights_with_cdf
WITH CHANGE DATA FEED;
```

Once Data is added or updated, Changes can be accessed as in this example

```
-- View changes as of version 1
SELECT * FROM
  table_changes('db_flights.flights', 1)
```

1. Change Data Feed (CDF)

Data Refresh

- When adding the table to the Share, ensure it is done with the CDF option
- On the recipient side, changes can be accessed and merged into a local copy of the data , notebook

```
On the recipient side
    - example
CREATE OR REPLACE TEMPORARY VIEW
silverTable_latest_version as
SELECT *
  FROM
      (SELECT *, rank() over (partition by Country
order by _commit_version desc) as rank
       FROM table_changes('silverTable', 2, 5)
       WHERE _change_type !='update_preimage')
  WHERE rank=1
    - Followed by a Merge command
```

2. Deep Clone a Shared table

One Click Replication

Manage Egress, enhance performance, Ease of implementation

- Easy ‘one click’ replication
- Used for sharing within the same Databricks cloud account.
- Copies both source table data and metadata to the clone target.
- Identifies new data and refreshes the target table accordingly.

Syntax for Deep Cloning:

```
CREATE TABLE [IF NOT EXISTS]
table_name DEEP CLONE
source_table_name [TBLPROPERTIES
clause] [LOCATION path]
```

Used on the recipient side to create a local copy (`table_name`) of the shared table (`source_table_name`).

2. Deep Clone a Shared table

Data Refresh

- Leverages Cloud Tokens instead of pre-signed URLs

Incremental Data Refresh: A Databricks Workflows job can be scheduled using the command

```
CREATE OR REPLACE TABLE table_name
DEEP CLONE
```

3. Materialized View on a shared table

Auto Refresh vs Complete Refresh

Manage Egress, enhance performance, Ease of implementation

- Create materialized view on a shared table with history
- Auto refresh with incremental updates
- Intra -account sharing

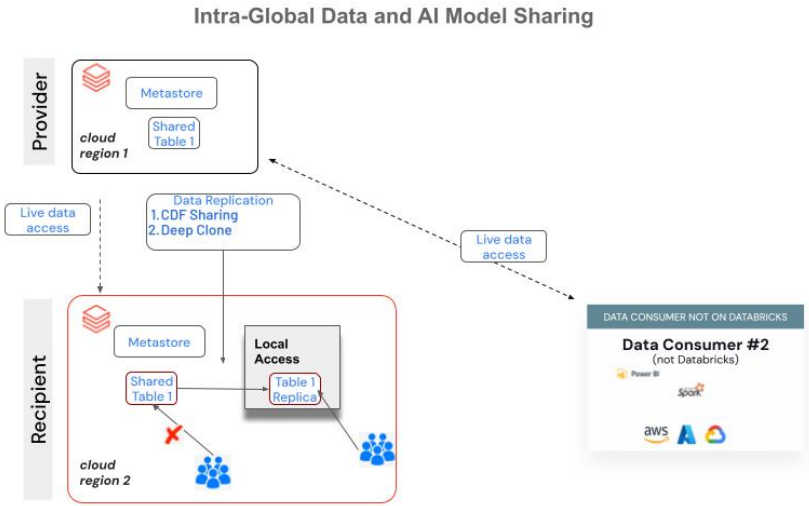
```
CREATE MATERIALIZED VIEW IF NOT EXISTS
newcatlg.default.mv_acme_customers as
        select user_id,email from
acmecatg.default.acme_customers_copy;

select count(*) from
newcatlg.default.mv_acme_customers;

REFRESH MATERIALIZED VIEW
newcatlg.default.mv_acme_customers;
```

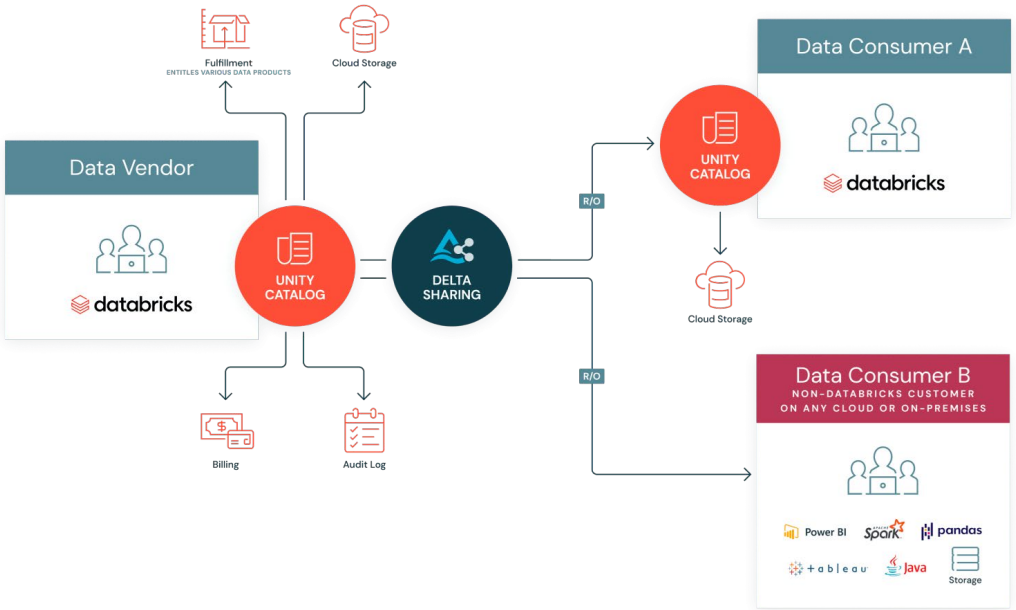
Access Control in Data Replication:

1. Recipient should restrict access to the shared table to prevent unintentional cross-cloud queries
2. The recipient's user group can access the local replica table like any Delta table



Data Aggregators - Hub and Spoke Model

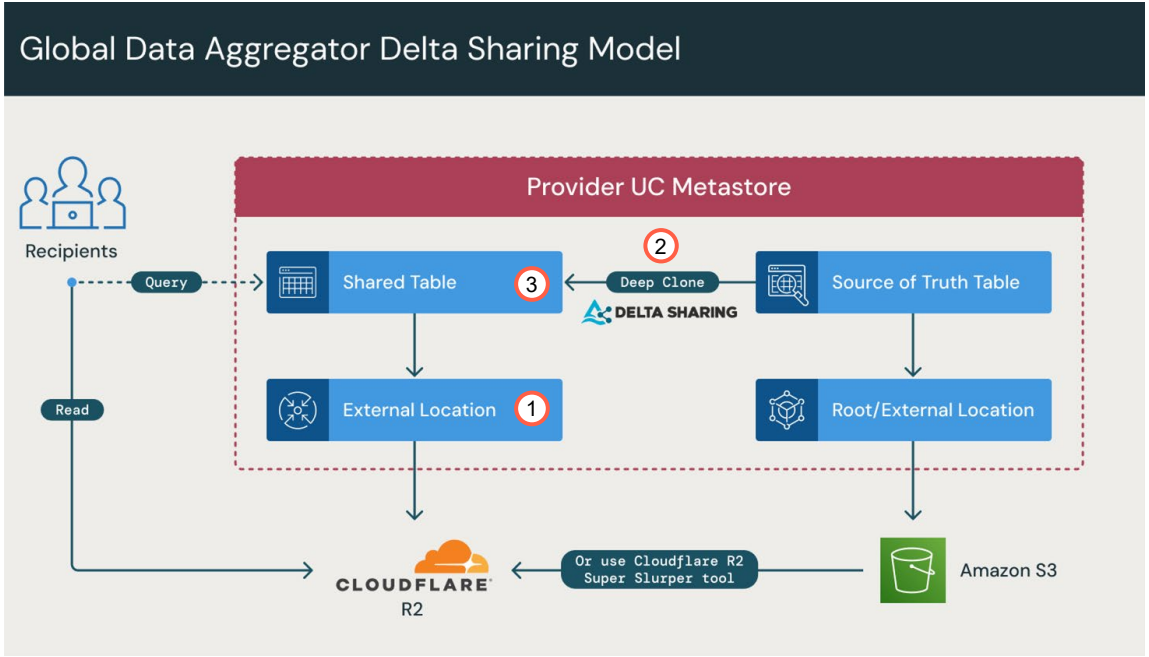
- Centralizes data collection and distribution to clients, merging data from diverse sources.
- Connects recipients across clouds and platforms.
- **Main challenge:** scale to a high number of cross-region recipients, with a predictable cost.
  - Economies of scale: the more recipients, the lower the marginal cost.



Zero-egress sharing from Cloudflare R2

No need to migrate all data to R2, simply replicate the data to be shared to R2 and then share:

1. Mount an R2 bucket as an external storage location
2. Create new tables in R2, syncing data incrementally from S3/ADLS using ‘deep clone’
3. Create a Delta Share, as usual, on the R2 table



Summary of data replication options

Data Replication Tool	Key highlights	Recommendation
Change data feed on a shared table	• It works within and across accounts • CDF needs to be enabled on the table • Requires coding to propagate the CDC changes on the destination table • The process can be orchestrated via Databricks workflows	Use for external Sharing with partners/clients across regions
Delta Deep Clone & MV	• It works within the same account • Minimum coding • Incremental refresh via Databricks workflows	Recommended when sharing internally across regions
Cloudflare R2 with Databricks	• Cloudflare account required • Ideal for large-scale data sharing across multiple regions and cloud platforms • Utilize delta deep clone or R2 super slurper for efficient data creation and refreshing in R2	Strongly recommended for large scale Delta Sharing in terms of number of regions and data sharing size

Takeaways

Identify and Refine

Considering Data replication

- Adopt live sharing v.s. data replication based on your use case
- Manage Egress cost effectively
- Select the right data replication mechanism
- Consider no egress cost solution

Balancing the Key factors

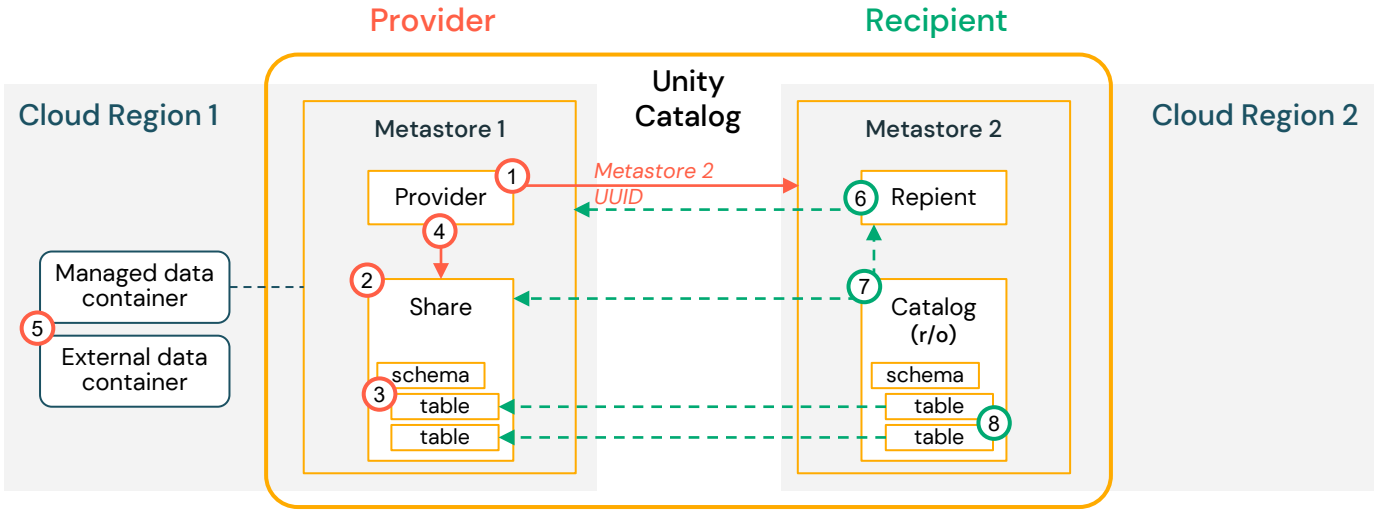
- Implementation Costs
- Egress Fees
- Data Access Performance
- Data Freshness/Timeliness

Control & Monitor

- Control access to the Delta share asset
- Monitor data assets

Delta Sharing Security Architecture

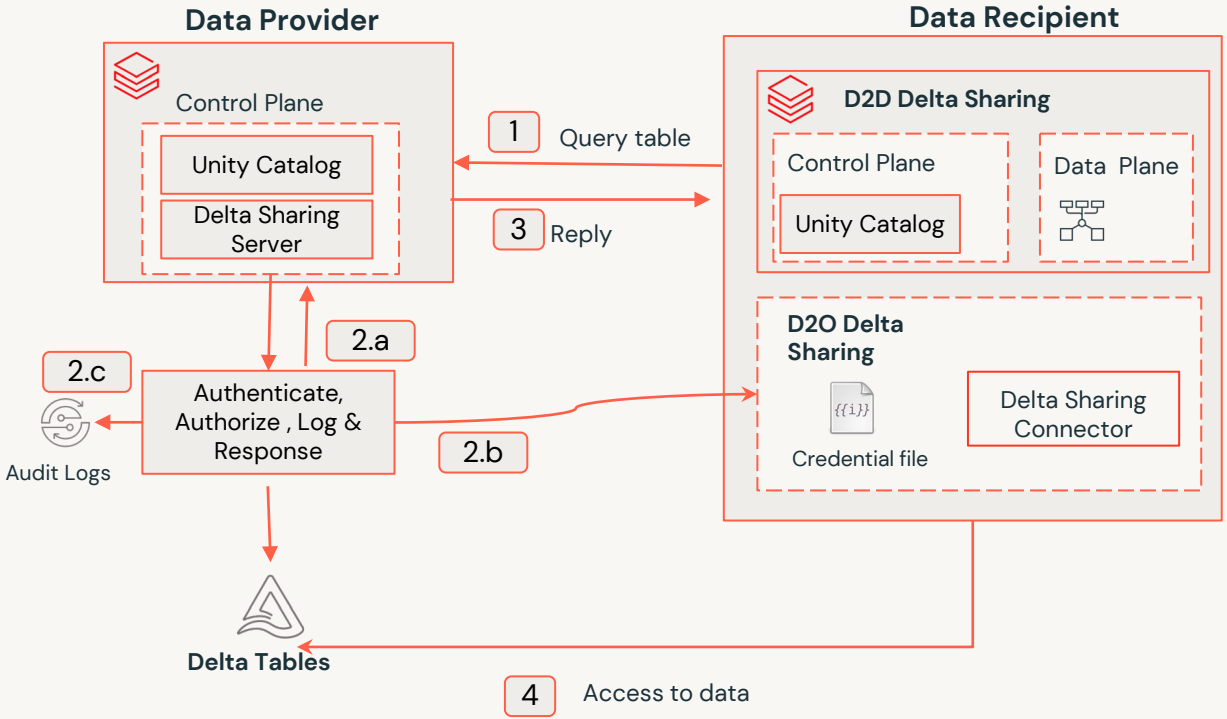
Databricks to Databricks Delta Sharing



1. *create <recipient>*
2. *create <share>*
3. *add Assets(tables,..) to <share>*
4. *grant "select" on <share> to <recipient>*
5. *enable network access for consumers in cloud region 2*
6. *List <providers> and <shares> (automatically available after recipient creation)*
7. *Create catalog for <provider> and <share>*
8. *Grant access to "Metastore 2" users on schemas/tables*

Delta Sharing

Data Access



1 Read Data

2.a

Authentication: Confirm valid token, confirm consumer access
Authorization: Confirm access to requested table
Response: Return short lived(1 hour) pre-signed URL to access data or in the intra-account case return a temporary read access to the table's root directory for efficient query processing via a cloud token.
IP list , fine-grained access

2.b D2O, generate a list of temp URL, signing happened locally

2.c Log activity

3 D2D, Reply with temporary signed URL list or use a cloud token

4 Access to Data via URLs
end-to-end TLS encryption between client and server and between client and cloud storage

Revocation

Credential Leak Mitigation: In case of a security breach due to a leaked activation URL, immediate actions are required.

- Rotate Recipient Credentials: Invalidate the compromised URL and token using unity-catalog rotate-recipient-token, and issue a new activation URL.
- Revoke Read Permissions:
  Use SHOW GRANT and REVOKE SELECT commands to remove the problematic recipient's access to the shared data.
- Delete Recipient for Extreme Cases:
  Employ DROP RECIPIENT and CREATE RECIPIENT commands to invalidate all activation URLs and tokens for the compromised recipient and create a new one.

API Input Validation

All API inputs are Databricks-generated identifiers (e.g. UUIDs) which are validated to be well formed and to exist. Malformed inputs are properly handled and do not leak any relevant information.

Transport Security

Internal/privileged traffic from recipient-CP to provider-CP with service-level authentication, inaccessible to external agents
All communication is encrypted with TLS 1.2.

User Authentication and Authorization

Data provider users: They must be users on Databricks with Databricks account and they use personal access tokens to access Databricks APIs to manage and share their data on Unity Catalog. The token usually has a limited lifetime (varying from hours to days). *

Data Recipient users: They can be 3rd party users without a Databricks account. Access to shared data is granted using recipient bearer tokens. These bearer tokens are issued by accessing a one-time activation URL shared by the data provider. The bearer token lifespan is configurable by the data provider and can have infinite lifetime (never expires).

Storage Access

Delta Sharing Access Essentials

- Network and storage access is only needed when the provider restricts access to the storage using a firewall or a private endpoint
- The recipient must have access to the provider's storage, where the shared data is located
- Creation and access granting to shares can proceed even without recipient's access to provider storage

Intra-Cloud and Cross-Cloud Delta Sharing

Two common data sharing paradigms:

❖ Intra-Cloud Sharing

Data shared within the same cloud platform (Azure, AWS, GCP) by both provider and recipient.

Provider & Recipient
Azure
AWS
GCP

❖ Cross-Cloud Sharing

Sharing between different clouds (Azure, AWS, GCP) by provider and recipient

Provider	Recipient
Azure	AWS, GCP
AWS	Azure, GCP
GCP	Azure, AWS

Provider Network Security Options

Common network security implementation

Azure

Storage account

- Firewall.

  List of IP ranges/Vnets that are allowed to access the storage account

- Private endpoint

  Private endpoint to establish a private link to the storage account

AWS

- Provider secures S3 using S3 bucket policies
- VPC S3 endpoint

GCP

- VPC Service Control

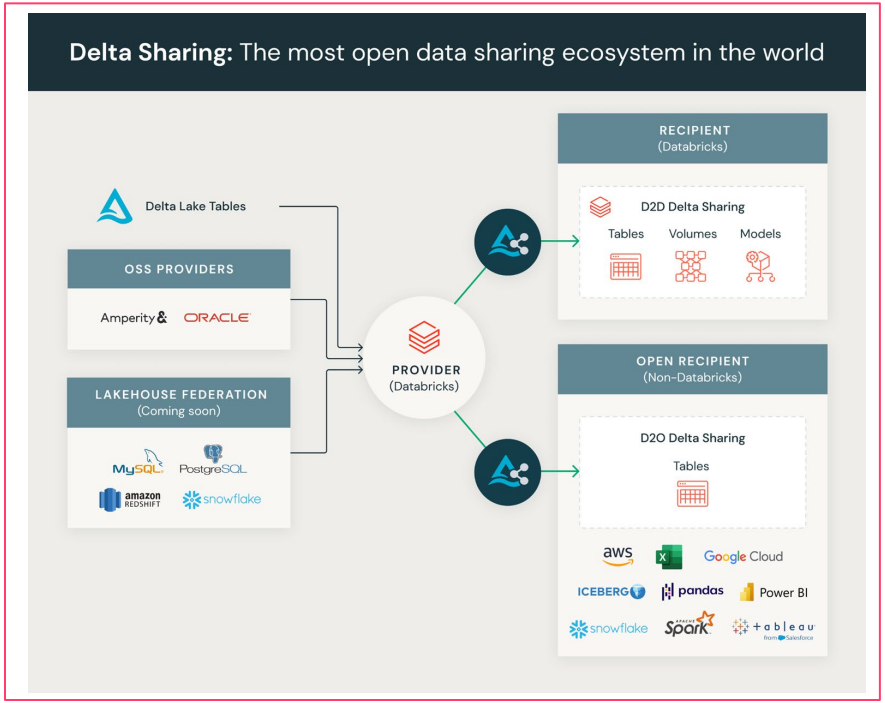
Provider secures GCP project where the GCP storage account (GCS) resides

Data Sharing Demo

- Databricks to Databricks - Cross cloud
- Databricks to Open

Open Data Sharing

Open Data Sharing Ecosystem



- Ability to Consume Data on Non-Databricks Platforms
- Delta Sharing Native Connectors
- Expanding Beyond Native Connectors with
  - Python and PySpark
  - Foreign Catalog

Delta Sharing Connectors

Some of the common connectors

Connector	GitHub Link	Description	Major Features
Python	delta-io/delta-sharing	Python/PySpark sharing client	Query metadata, Query version,Get latest snapshot, CDF,
Spark	delta-io/delta-sharing	Apache Spark sharing client	Query metadata, Query version,Get latest snapshot, CDF, Streaming
PowerBI	databricks/powerbi-delta-sharing-connector	Documentation	Get latest snapshot
Excel Add-in	Exponam Excel Add-in	Excel add-in for Delta Sharing and writing Delta tables	Query metadata, Query version,Get the latest snapshot
Tableau	https://www.tableau.com/blog/tableau-databricks-delta-sharing-connector	Tableau Connector	Query metadata, Query version,Get latest snapshot

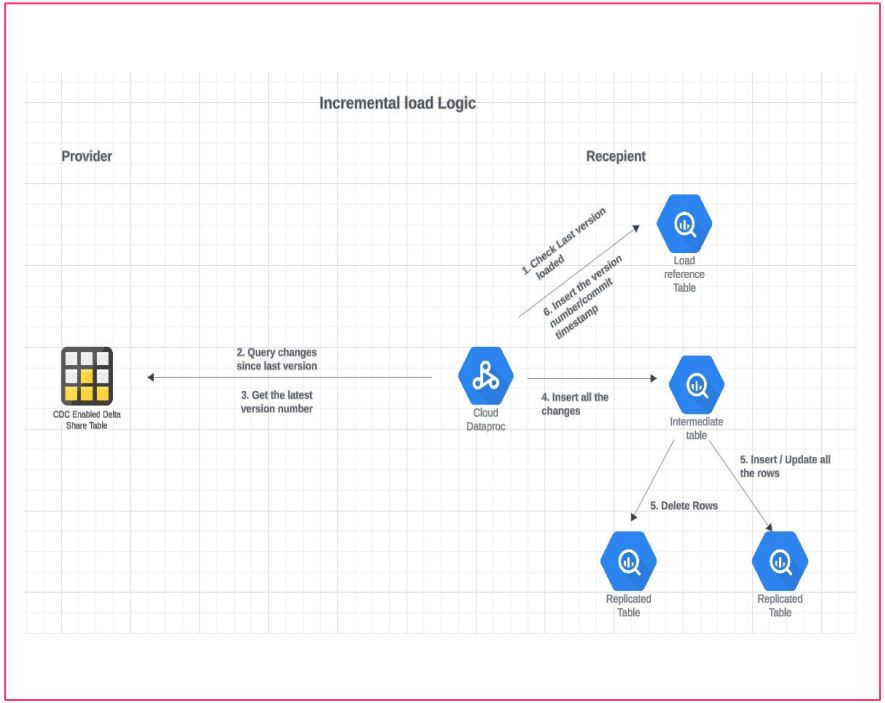
Python Panda & PySpark Connector

- Creating a new recipient generates an activation token.
- The recipient uses this activation token to download a one-time credential file.
- The credential file contains a token and a Delta Sharing endpoint.
- The code or connectors reference this credential file when querying the data

```
pip install delta-sharing
import delta_sharing
import pandas as pd
# Path to the Delta Sharing profile JSON file
profile_file = "path/to/your/profile.delta-sharing.json"
# Load the profile
client = delta_sharing.SharingClient(profile_file)
# Load a specific table into a DataFrame
table_url = "delta-sharing://<profile>#schema_name.table_name"
df = delta_sharing.load_as_pandas(table_url)
```

Sharing Data with BigQuery

- Leverage the Delta sharing PySpark package
- Cloud Dataproc to run Pyspark
- Leverage a staging table
- Implement an incremental refresh
- Blog
- Notebook



Q & A

bhavin@databricks.com
bilal@databricks.com