Unity Catalog Upgrade

Jobs and Table Upgrades

OBJECTIVES

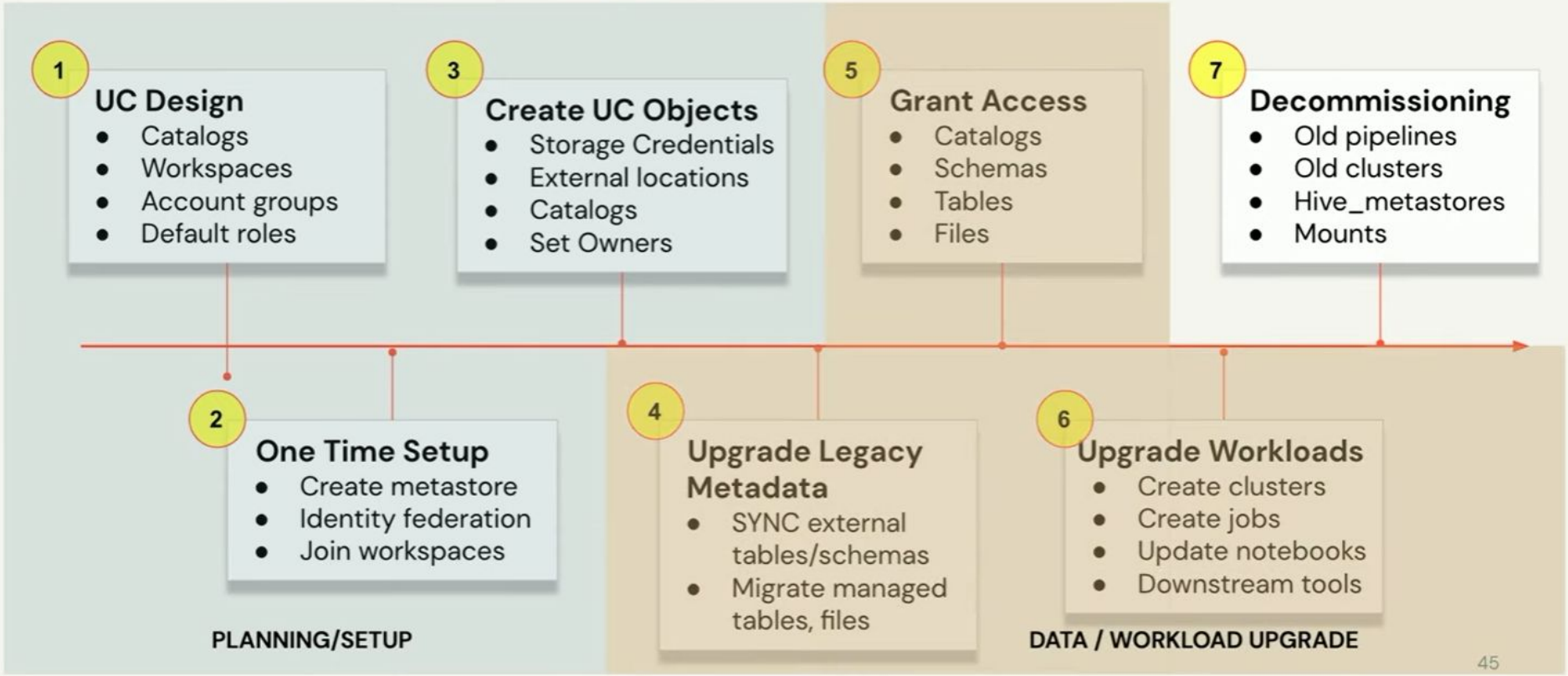
The purpose of today's discussion is to go over best practices around upgrading your Data, Code, and Compute and plan the following:

- Databricks compute (clusters, workflows, pools, etc.)
- Tables
- Views
- Functions
- ML Models
- DB SQL artifacts
- BI Consumers of Databricks
- Notebooks and Code
- Plan to migrate the above without downtime or loss of continuity.

Record the decisions made and incorporate into diagrams and design documents for the project deliverables

High Level Roadmap to Unity Catalog

Steps to consider for a full upgrade

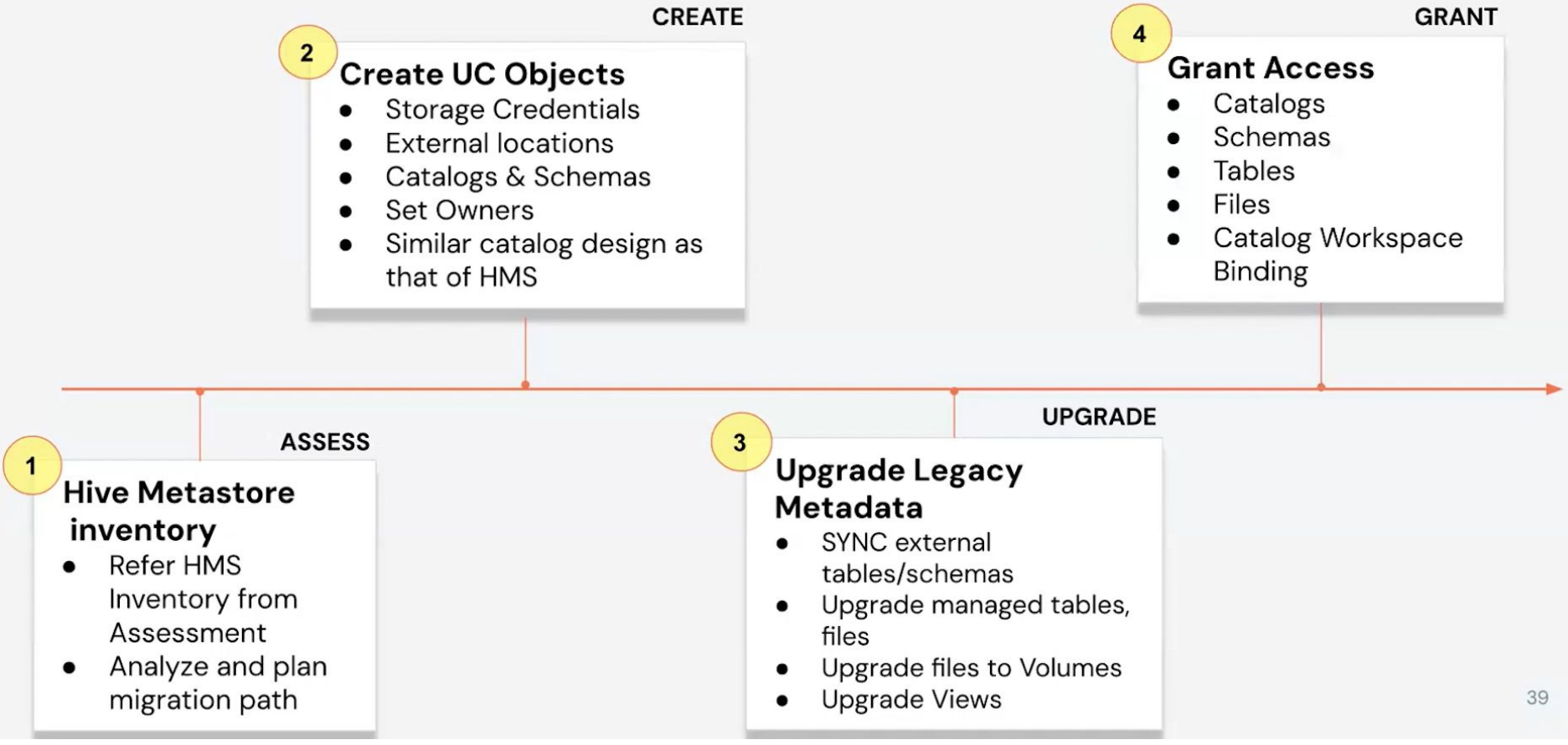


Upgrading Data

- Managed vs. External Tables
- SYNC or Upgrade
- Table Access Controls
- Upgrading Views and User Defined Functions

Upgrade Data to Unity Catalog

Steps to consider for upgrading Data



[UCX][adam.edgeworth] UCX Assessment

28	300	0
Total Databases	Total Tables	Metastore Crawl Failures

Table Types

database	name	type	format	table_view	storage	is_delta	location
adamdlt	notebook_log_raw	EXTERNAL	DELTA	TABLE	DBFS ROOT	No	dbfs:/AdamDLT/tables/notebook_log_raw
adamdlt	notebook_log_silver	EXTERNAL	DELTA	TABLE	DBFS ROOT	No	dbfs:/AdamDLT/tables/notebook_log_silver
bookstore_eng_pro	bronze	MANAGED	DELTA	TABLE	DBFS ROOT	No	dbfs:/user/hive/warehouse/bookstore_eng_pro.db/bronze
bronze	cars_fiat	EXTERNAL	JSON	TABLE	EXTERNAL	No	s3a://koantek-ascend-s3/raw/internal/cars/fiat/-1
bronze	cars_fiat_new	MANAGED	DELTA	TABLE	DBFS ROOT	No	dbfs:/user/hive/warehouse/bronze.db/cars_fiat_new

1 2 3 4 5 ··· 12 >

Database Summary

database	tables	views	unsupported	dbfs_root	delta_tables	upgrade
adamdlt	2	0	0	2	2	Asset Replication Required
bookstore_eng_pro	1	0	0	1	1	Asset Replication Required
bronze	142	0	0	70	72	Asset Replication Required
cars	5	0	0	0	0	Some Non Delta Assets
dbacademy_abhishek	1	0	0	1	1	Asset Replication Required
dbacademy_adam_edgeworth_koantek_com_spark_programming_asp_1_3___spark_sql	4	0	0	0	0	Some Non Delta Assets
dbacademy_adam_edgeworth_koantek_com_spark_programming_asp_1_4___reader___writer	1	0	0	1	1	Asset Replication Required
default	23	2	0	18	21	Asset Replication Required

1 2 >

External Locations

location
s3://koantek-ascend-uc-s3/
s3a://databricks-corp-training/common/ecommerce/
s3://test-ec2-blt-s3-1/hive/default/
s3a://koantek-ascend-s3/
s3://uc-upgrade-test/
s3://koantek-ascend-s3/raw/orchestration/

Mount Points

name	source
/mnt/training	s3a://databricks-corp-training/common
/databricks-datasets	databricks-datasets
/mnt/Testing	abfss://input@shankmainworksa1.dfs.core.windows.net/
/mnt//mnt/uc-audit-log/konatek-ascend	s3n://uc-audit-log/konatek-ascend
/Volumes	UnityCatalogVolumes
/mnt/uc-audit-log/konatek-ascend	s3n://uc-audit-log
/databricks/mlflow-tracking	databricks/mlflow-tracking
/databricks-results	databricks-results

1 2 >

Clusters

cluster_id	cluster_name	creator	compatible	failures
0925-084101-7g854toc	Koantek-Ascend-Default	pankaj.wani@koantek.com	Compatible	[]
0908-073550-2ud78mv2	AWS-Ascend-Default-Cluster	pankaj.wani@koantek.com	Compatible	[]

Jobs

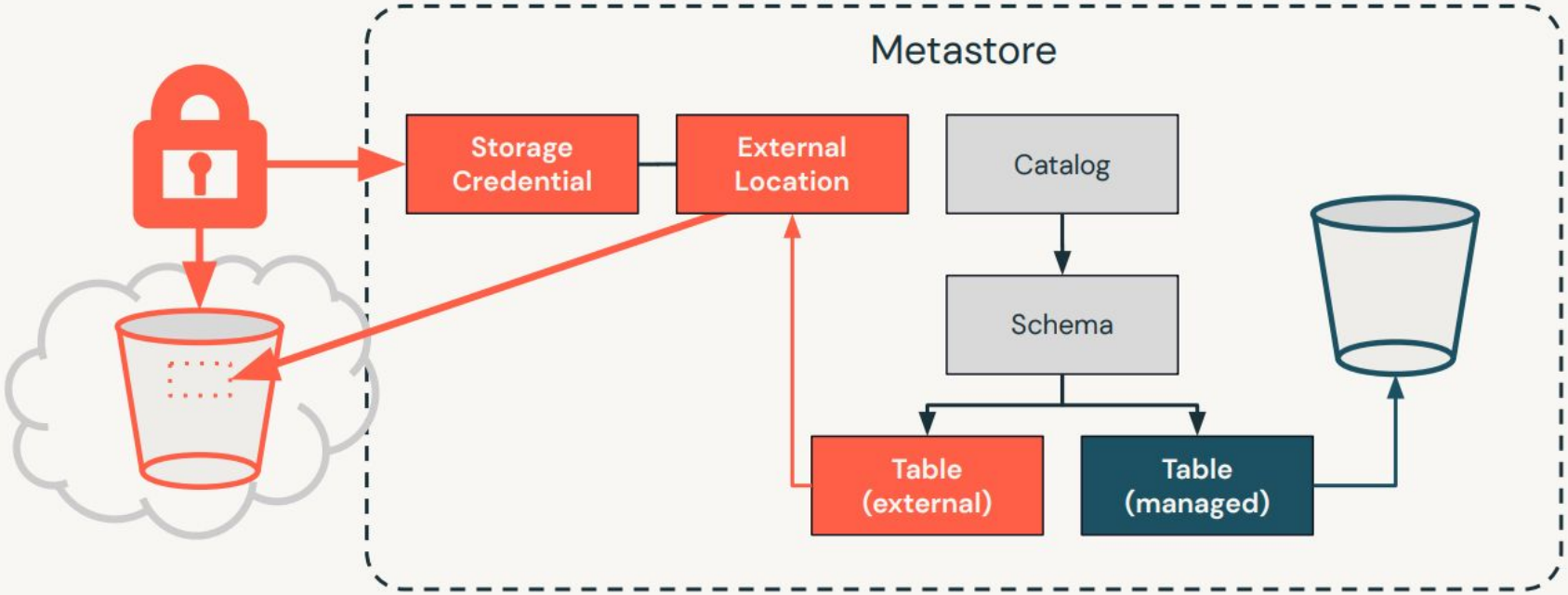
job_id	job_name	creator	compatible	failures
617748696280381	UAT_CSV_Update	dadasaheb.hinge@koantek.com	Compatible	[]
578902941770627	UAT_CSV	debjani.choudhury@koantek.com	Compatible	[]
737932726065701	UAT_CSV	sadhana.s.reddy@koantek.com	Compatible	[]
847642380145278	UAT_CSV	vineesha.chebrolu@koantek.com	Compatible	[]
1083186830106532	UAT_CSV	shubham.raut@koantek.com	Compatible	[]
191167170957240	UAT_CSV	dadasaheb.hinge@koantek.com	Compatible	[]
816975473793374	Parquet_UAT_Check	rohit.b.kumar@koantek.com	Compatible	[]
240222309766392	XML_UAT_Check	rohit.b.kumar@koantek.com	Compatible	[]

1 2 >

Pipelines

pipeline_id	pipeline_name	creator_name	Azure_SPN_Present	failures
6be140f1-ef4a-4326-b74f-21a2e7a6e926	dbdemos_dlt_lakehouse_churn_hemavathy_rathinam	hemavathy.rathinam@koantek.com	No	[]
74adefec-d169-42d6-b9f4-55c958e1e5dc	Adam DLT	edward.edgeworth@koantek.com	No	[]
90525aac-3098-47ca-9505-cddb4c08e14c	DLT Quickstart Pipeline	adam.edgeworth@koantek.com	No	[]
9f289504-544b-4faf-8faf-e61f0e1c84e8	Demo_pipiline	ratnesh.kumar@koantek.com	No	[]
c8c008f4-6e3c-4d6a-a4e4-0ac5c5e52754	AWS_DLT	sommireddy.komma@koantek.com	No	[]
2b0ccc7e-40a4-4fc6-b750-1d15de4f7584	demo_bookstore	abhishek.kumar@koantek.com	No	[]
3da5e63d-c4ff-43bf-bf73-ff49b6ce233b	Sayyam_test_pipeline	sayyam.jain@koantek.com	No	[]
4a5e7430-8d62-464a-8fb3-aa77a5d28049	aws_final	ratnesh.kumar@koantek.com	No	[]
4c699d33-4efc-456e-8677-07b5a83b96f2	databricks_sql_aws_log_collection	pankaj.wani@koantek.com	No	[]
51232099-34b6-4940-b128-3c1974ccb47f	DLT to S3 TEST	adam.edgeworth@koantek.com	No	[]

Managed and External Tables



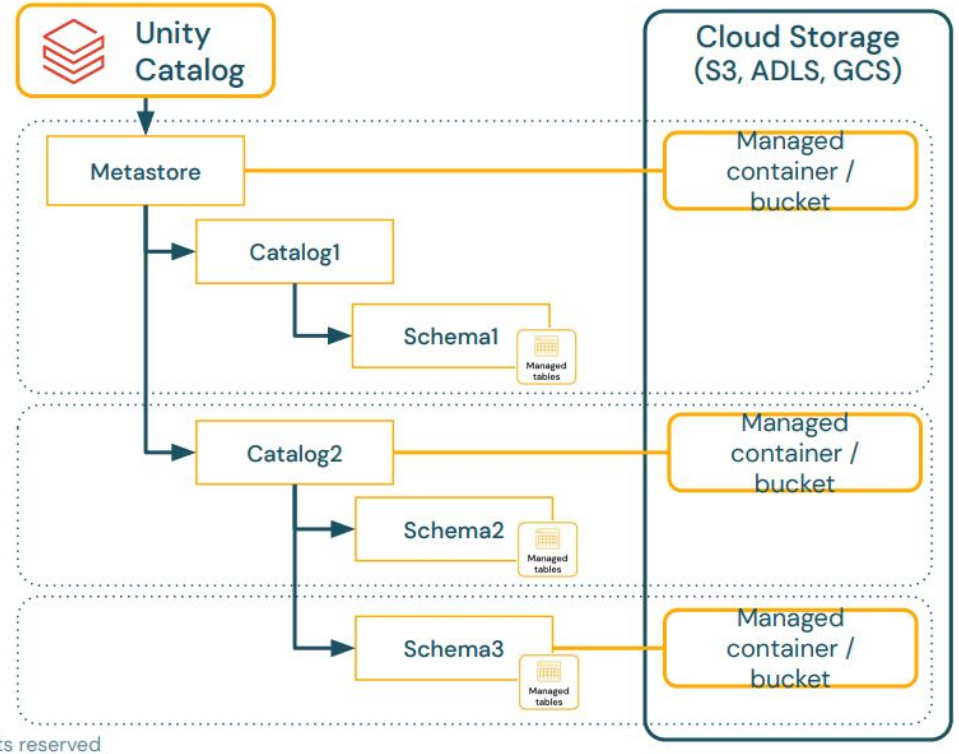
Default Storage Location for Catalogs and Schemas

Fine grained data isolation by storage locations for managed tables

1) Store at the **metastore**

2) Store at the **catalog**

3) Store at the **schema**



Note:
The lower location supersedes the higher: If, for example, both catalog and schema of a table have a managed location, then the table data is stored in the schema location

Managed and External Tables

Comparison

Characteristic	Managed Tables	External Tables
Syntax	Simpler syntax, no overhead, user doesn't need to know any locations or URLs: • CREATE TABLE <name> • df.write.saveAsTable("<name>")	The table location must be known and specified (full 'abfss://' URL as UC does not support mounts): • CREATE TABLE <name> LOCATION 'abfss://...' • df.write.option("path", "abfss://...").saveAsTable("<name>")

Data Location	Data lives in the metastore / catalog / schema managed storage location:
	• Uses GUID style folders which are less human-readable: abfss://data@storageaccount.dfs.core.windows.net/myschema/.../8d408932-2926-4c5c-9403-0fa7dd2e485d
	• There are multiple ways to retrieve the table location:
	  ○ DESCRIBE DETAIL <table_name>
	  ○ UC REST API: /api/2.1/unity-catalog/tables/{full_name}
	  ○ Directly via the HMS compatible layer
	• Every time a managed table is dropped and recreated, it will use a different GUID folder
	  ○ If using CREATE OR REPLACE instead of DROP/CREATE then the location is preserved
		Data lives in a user-provided storage location
		• In UC, this is an External Location and users must also have the correct privileges (CREATE EXTERNAL TABLE) to be able to create a table on an External Location
		• Specified with LOCATION or .option("path")
		• However, it must be a full 'abfss://' URL as UC does not support mounts

DROP behaviour	DROP discards metadata and underlying data is deleted from the storage account within 30 days • UNDROP can be used to restore the table • If recreating the table then it's better to use CREATE OR REPLACE instead of DROP/CREATE	DROP discards metadata only • UNDROP can also be used • If data needs to be deleted then this must be managed separately (e.g. dbutils.fs.rm)

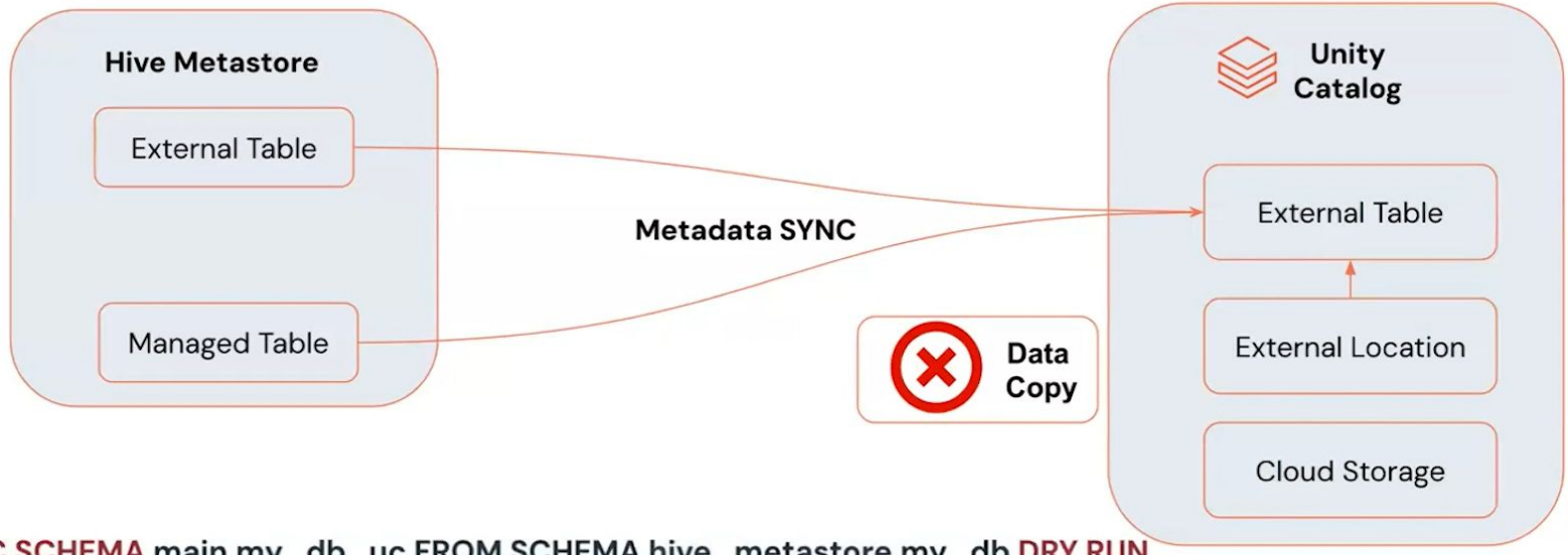
Managed and External Tables

Comparison cont.

Characteristic	Managed Tables	External Tables
Supported formats	DELTA only	DELTA, PARQUET, ORC ,CSV, JSON, AVRO, TEXT
Auto Maintenance and Auto Tuning	Coming to Managed tables in a first phase	Coming to External tables in a second phase (typically 1-2 quarters after Managed tables)
New product features	Supported first • For example - Liquid tables (partition-less auto clustering tables)	Might be supported later, no guarantees • For example - Liquid tables are not planned at the moment (but could change in the future)
External readers	External readers could read from UC Managed tables, however the folder path is less human-readable • Recommended to use the HMS compatibility layer • If not possible, there are multiple ways to retrieve the table location: ○ DESCRIBE DETAIL <table_name> ○ UC REST API: /api/2.1/unity-catalog/tables/{full_name}	External tables are most recommended for when an external reader needs to access the data using the location and doesn't support an HMS. • However, this will bypass the govername and access control enforced by UC (permissions, lineage, audit, etc.)
Migration from Hive	Currently, migrating to a Managed UC table can only happen with data copy (DEEP CLONE or CTAS) • New functionality to permit Managed to Managed migration is being developed	Simpler, either with the Data Explorer UI or with the SYNC command. • No data migration, only the metadata is copied
DR	Currently very difficult to do, solution in development	Simple as it can reference the same path (if data replication is handled separately)
Best for	User created Delta tables (Gold/Platinum)	• Pipelines created Delta tables (Bronze, Silver) • Tables over folders shared with other systems • Non-Delta tables

Upgrade Data to Unity Catalog

SYNC Means **no data copy**




```
SYNC SCHEMA main.my_db_uc FROM SCHEMA hive_metastore.my_db DRY RUN

SYNC TABLE main.my_db_uc.my_tbl FROM TABLE hive_metastore.my_db.my_tbl
```

Data upgrade scenarios

Upgrade Methodology Matrix

HMS Table Type	Target UC Table Type	Data Storage Location	Methodology
External or Managed	External or Managed	DBFS Root	CTAS/Deep Clone
External or Managed	External	DBFS Mounted Storage or Cloud Storage	SYNC
External or Managed	Managed	DBFS Mounted Storage or Cloud Storage	CTAS/Deep Clone
Hive SerDe Interface			
External or Managed	External or Managed	NA	CTAS/Deep Clone

Sync External Tables

1. Create External locations (as advised by the assessment step)
2. SYNC the external and managed tables that are not using the DBFS root storage.
   a. SYNC now supports Managed tables
   b. SYNC can be run on a whole database/schema
   c. SYNC converts managed tables to external
   d. For large migrations (1000s of tables) use the migration wizard
   e. HOW OFTEN SHOULD WE SYNC?

One Time Sync:

- Delta or Parquet Tables
- Static Metadata
- HMS to UC Switchover

Ongoing Sync:

- Non Parquet/Delta Tables
- Constant Metadata Updates

Challenges with External Tables

What works what doesn't

1. Performance challenges on Parquet Tables with many partitions (Partition Support **Q3**)
2. When upgrading a Managed Table in HMS to an External Table in UC, DROP semantics do not enforce file deletion. (Unifying Table Types, switch External -> Managed **Q3**)
3. We don't encourage writing from HMS + UC together. Pick a single writer.
4. Hive SerDe support is not available. (HMS Federation **Q4**)

Upgrading Hive tables to Unity

Managed & External tables - use SYNC command

- Run multiple times to pull changes from the hive/glue database into Unity over time
  - Use a job for long term synchronization
- Use the DRY RUN option to test the sync without making any changes to the target table.
- Works on Hive Managed Tables where schema locations are defined.

```
SYNC SCHEMA hive_metastore.my_db TO SCHEMA main.my_db_uc DRY RUN

SYNC TABLE hive_metastore.my_db.my_tbl TO TABLE main.my_db_uc.my_tbl
```

Moving Managed Hive tables to Unity

Optional or if in DBFS root - CTAS/CLONE

```
1  // A. Managed Delta -> Managed Delta
2  CREATE TABLE <new_catalog>.<new_schema>.<new_table> CLONE
3  hive_metastore.<old_schema>.<old_table>;
4  // B. Managed non-Delta -> External non-Delta
5  CREATE TABLE <new_catalog>.<new_schema>.<new_table> LOCATION <..> AS SELECT * FROM
6  hive_metastore.<old_schema>.<old_table>;
7  // A+B. Once fully upgraded and tested, drop hive table
8  DROP TABLE hive_metastore.<old_schema>.<old_table>;
```

Upgrade Data to Unity Catalog

Challenges with migrating Data to UC

Certain migration scenarios are not yet supported by SYNC:

1. Tables created on DBFS Root or Tables using Hive SerDe interface
2. Customer wants to only move to Managed tables in UC
3. Migrating views and User Defined Functions
4. Migrating Table ACLs
5. Migrating IAM Roles or ADLS Gen 2 Security to UC ACLs

Other Challenges with UC data upgrade:

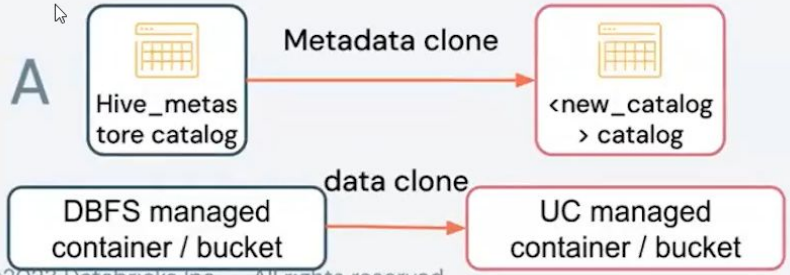
6. Number of tables to be migrated and/or amount of data replication involved
7. NON DELTA TABLE PERFORMANCE Limitations

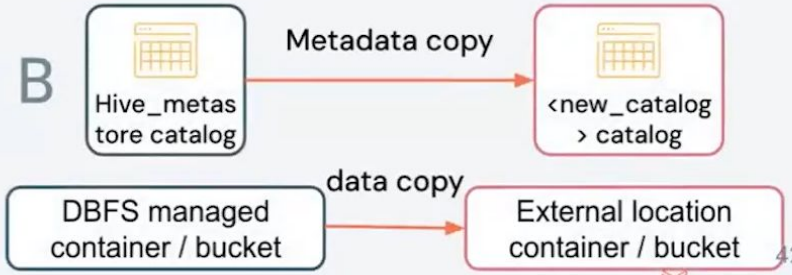
Upgrade Data to Unity Catalog

Challenges with migrating Data to UC

1. **Tables created on DBFS Root or Tables using Hive SerDe interface - Use CTAS/CLONE**
2. **Customer wants to only move to Managed tables in UC - Use CTAS/CLONE**

```
// A. Managed -> Managed
CREATE TABLE <new_catalog>.<new_schema>.<new_table> CLONE
hive_metastore.<old_schema>.<old_table>;
// B. Hive SerDe -> Managed Delta or External Delta or External non-Delta
CREATE TABLE <new_catalog>.<new_schema>.<new_table> [LOCATION <..>] AS SELECT * FROM
hive_metastore.<old_schema>.<old_table>;
// A+B. Once fully upgraded and tested, drop hive table
DROP TABLE hive_metastore.<old_schema>.<old_table>;
```



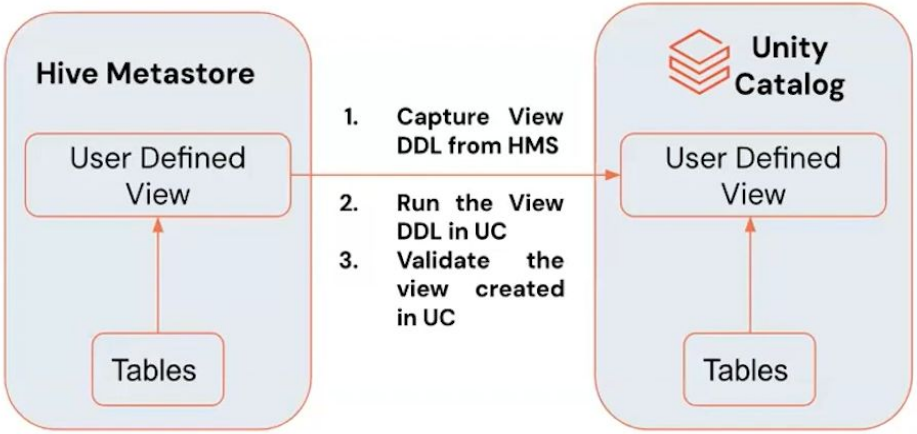


Upgrade Data to Unity Catalog

Challenges with migrating Data to UC

3. Migrating views and User Defined Functions

Prereq: Upgrade all of a view's referenced tables to the Unity Catalog from Hive Metastore



User defined Functions (UDFs) in UC

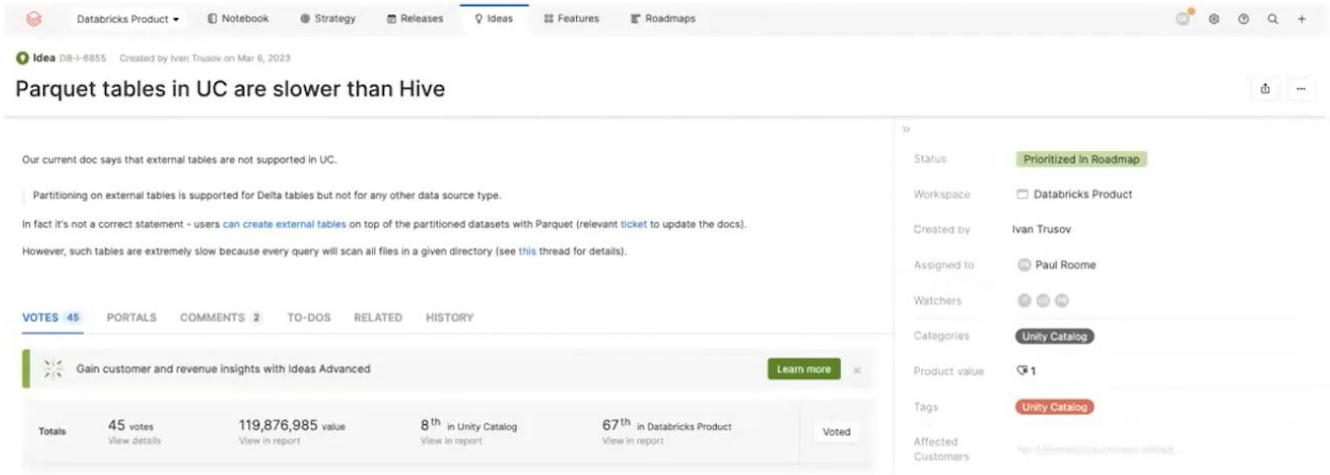
- Python/Scalar Panda/Pyspark UDF
  - DBR 13.2+ All Access Modes
- Python/Scalar Panda
  - DBR 13.2+ All Access Modes
  - Only Serverless or Pro SQL WH
  - In Public Preview
- SQL UDF
  - DBR 9.1+ All Access Modes
  - All SQL WH
  - GA
- Scala UDF
  - Soon in Private Preview (Q3, 2023)

Upgrade Data to Unity Catalog

Challenges with migrating Data to UC

6. NON DELTA TABLE PERFORMANCE Limitations

- AHA – Currently Prioritized In Roadmap – By end of Q3. Check details in go/uc/matrix



Upgrading Compute

- Upgrading Clusters
- Cluster Security Modes
- Upgrading SQL Warehouses

Upgrade Clusters to Unity Catalog

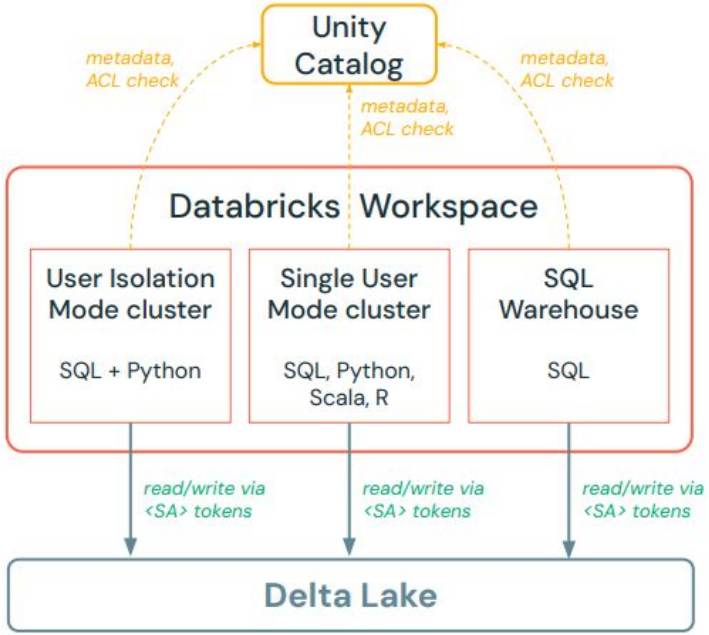
Why?

- To allow clusters delegate authorization activity to centralized UC
- Utilize storage credentials and ACL based authorization
- To enable seamless interaction with metadata across workspaces
- To let go instance profiles after migration

How?

- Update cluster policies to use the proper UC enabled cluster type.
- Terraform databricks_cluster resource
- Workspace console

Clusters/endpoints with Unity Catalog



<SA> System Account
(Service Principal or Managed Identity for Azure, IAM Role for AWS, Service Account for GCP)

Standard clusters with User Isolation mode

- Use User isolation mode for general workloads (ETL, data exploration, ...) using SQL and Python
- Multiple users can work on the same cluster

Standard clusters with Single User mode

- Use Single User mode for Scala users and for Data Scientists
- ML Runtime with MLflow is supported
- Only the owner can execute code on this cluster, so for notebook collaboration, co-workers can see everything, but they cannot execute cells.
- Limitations
  - Access to views requires access to the underlying table
  - Dynamic views (e.g. for row- /column-level security) are not supported

SQL Warehouse

- Use SQL Warehouses for Business Analysts either using Databricks SQL Editor or external BI tools like Power BI, Tableau, ...

Note: Multi-user support for ML Runtimes and MLFlow is under development

Clusters

Security mode feature matrix

Security mode	Supported languages	Legacy table ACL	Credential passthrough	Shareable	RDD API	DBFS Fuse mounts	Init scripts and libraries	Dynamic views	Machine learning
None	All			●	●	●	●		●
Single user	**All**				●	●	●		●
User isolation	**SQL Python**	●		●				●	
Legacy table ACL	SQL Python	●		●			●		
Legacy Passthrough	SQL Python		●	●	●	●	●		

Challenges with Upgrading Compute

What works what doesn't

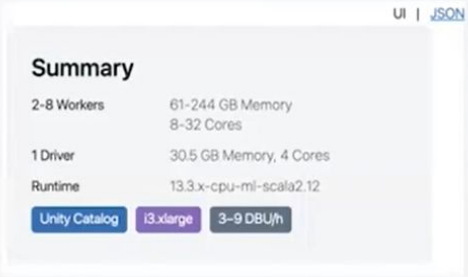
- Shared clusters
  - Don't support DBR ML runtime. (Preview)
  - Init script support for shared clusters.
  - Support for Scala, R. (Preview)
- Single user clusters
  - Don't support RLF and CLM
  - Don't support Dynamic Views
- DLT External UC tables accessibility.

Upgrade Clusters to Unity Catalog

Cluster Options

Clusters running DBR less than 11.1

- Drop and re-create Unity Catalog compatible cluster
- Cluster creation summary depicts Unity Catalog support
-



Clusters running DBR 11.1+

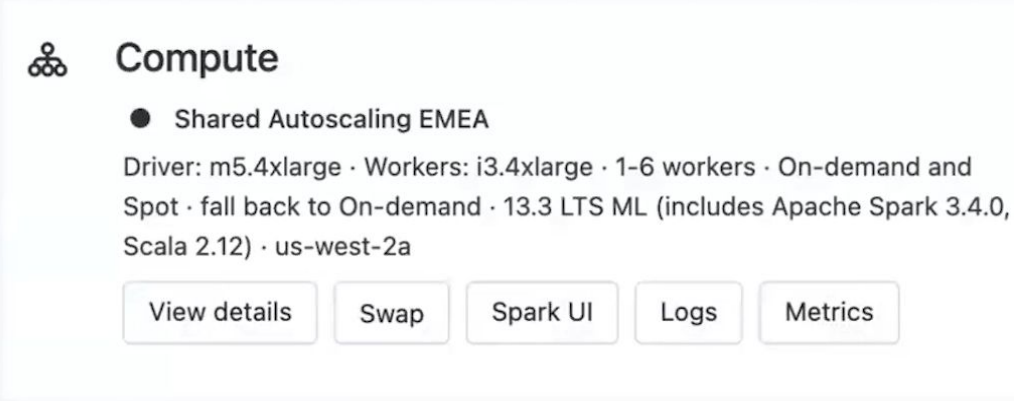
- Edit existing cluster using Access Modes
  - Single User
  - Shared
  - No isolation shared
- DBR 14 Beta is available
- Recommend 13.x LTS

Upgrade Clusters to Unity Catalog

Cluster Options

Job Clusters

- Job Details
- Configure Compute
- Select Access Mode
  - Single User
  - Shared
  - No isolation shared



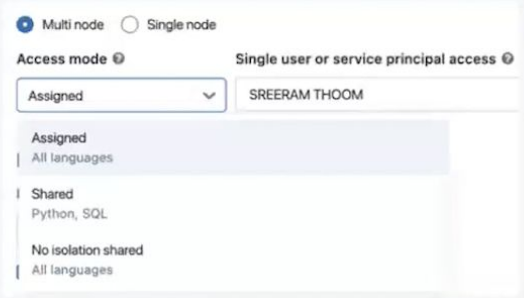
Upgrade Clusters to Unity Catalog

Challenges

- Shared clusters don't support DBR ML runtime, Distributed ML and GPUs.
- Jars/whl files require Shared Cluster Preview
  - Scala UDFs are not supported
  - Limited support for Python UDFs
  - Python UDFs are not accessible from SQL Warehouses
- DLT generated UC tables are not accessible using Single User/Job clusters.
- Jobs will not automatically work on a UC enabled cluster.
- Queries & Dashboards may not automatically work on a UC enabled SQL Warehouse

Upgrade Clusters to Unity Catalog

Cluster Options



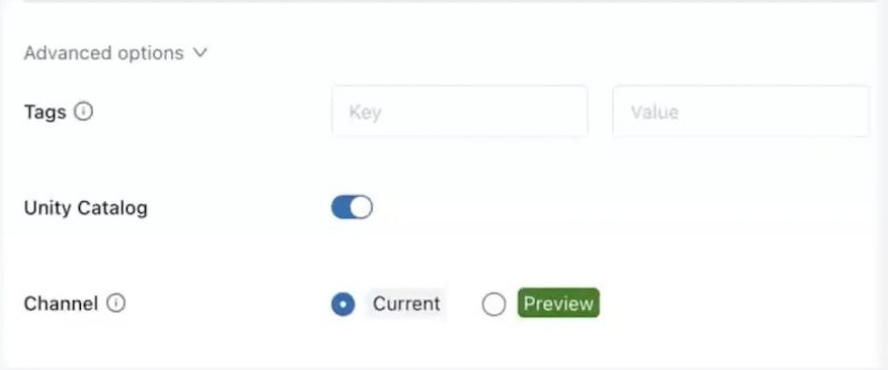
- **Single user**
  - This mode allows you to Run SQL, Python, R and Scala workloads as a single user, with access to data secured in Unity Catalog.
  - Limitations:
    - Can not access DLT tables
    - Dynamic Views not supported
- **Shared**
  - Multiple users can share the cluster to run SQL and Python workloads on data secured in Unity Catalog.
  - Improved shared Cluster is in PrPr to increase parity between Single User and Shared.
  - Limitations:
    - ML Runtime not supported
- **No isolation shared**
  - This mode allows clusters to run without UC.

Upgrade Clusters to Unity Catalog

Cluster Options

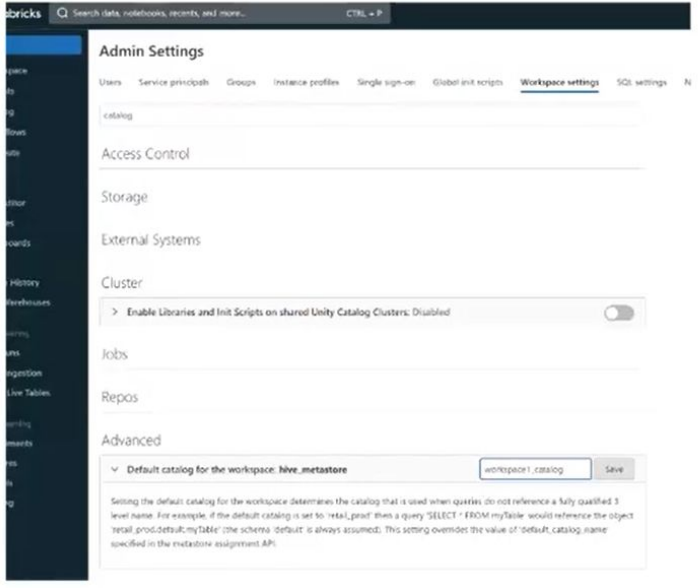
SQL Warehouse Clusters

- Under Advanced options
- Toggle Unity Catalog option
- Use workspace level default catalog configuration to work for both DBR clusters and SQL Warehouses.



Step 4 Upgrade the DBSQL Warehouses

1. Switch warehouses to UC Enabled Or Create new ones
2. Set a default catalog for the workspace
3. Validate SQL scripts and dashboards



Upgrading Workflows and Jobs

Upgrading Jobs are not trivial tasks

Code changes, config changes, DBR changes oh my

Libraries, mounts, configuration

- Usage of custom libraries
  - Requires shared clusters preview
  - Still may not work when tested
  - Framework code may need to be changed
- Interacting w. mounted paths
  - Should use volumes
- Streaming jobs
  - Checkpoint locations, no support for continuous processing mode
- MLR Limitations
  - Currently require changing operating pattern

Spark Commands and external libs

- Code that uses AWS or Azure SDKs
  - Identity no longer comes from the user, so service principal, or other auth forms are required
- Code that performs spark operation that reads or writes to:
  - HMS Table -> 3 L NS
  - Mount Point -> Volume
  - S3/abfss/gcs path -> ExtLoc
  - A sub-partition of a registered table

DBR versions and ACLs

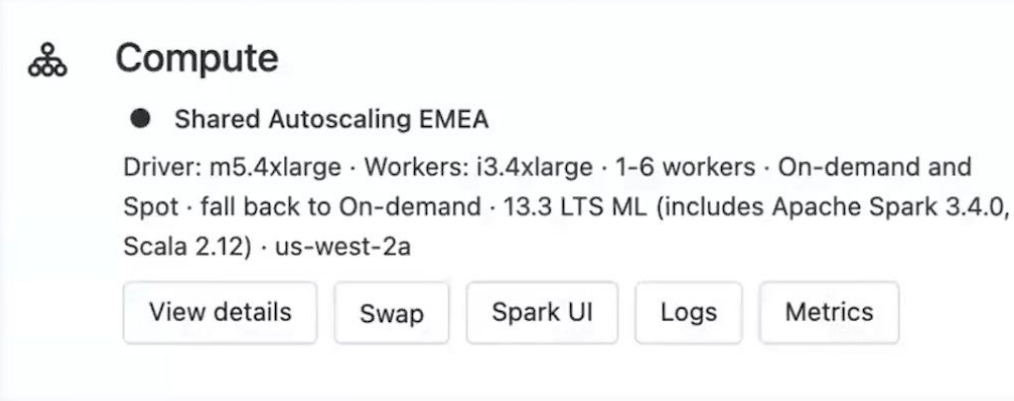
- DBR version lower than 11.3
  - Should use the latest version if possible
- ACLs need to be applied to service principals to have permissions so jobs run
  - Can get this from HMS Lineage if turned on
- DLT jobs
  - Requires DBR 13.1
  - Data has to be migrated

Upgrade Clusters to Unity Catalog

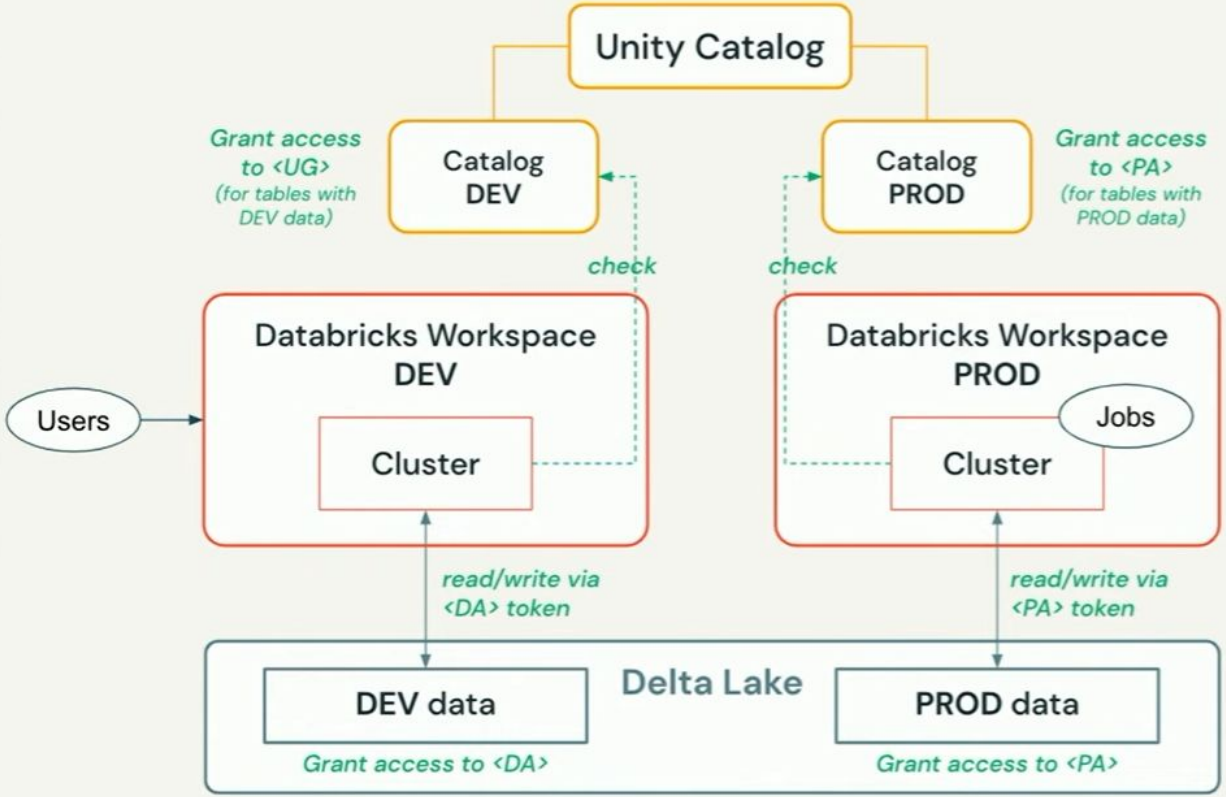
Cluster Options

Job Clusters

- Job Details
- Configure Compute
- Select Access Mode
  - Single User
  - Shared
  - No isolation shared



Software Development Lifecycle setup w/ UC



Note:

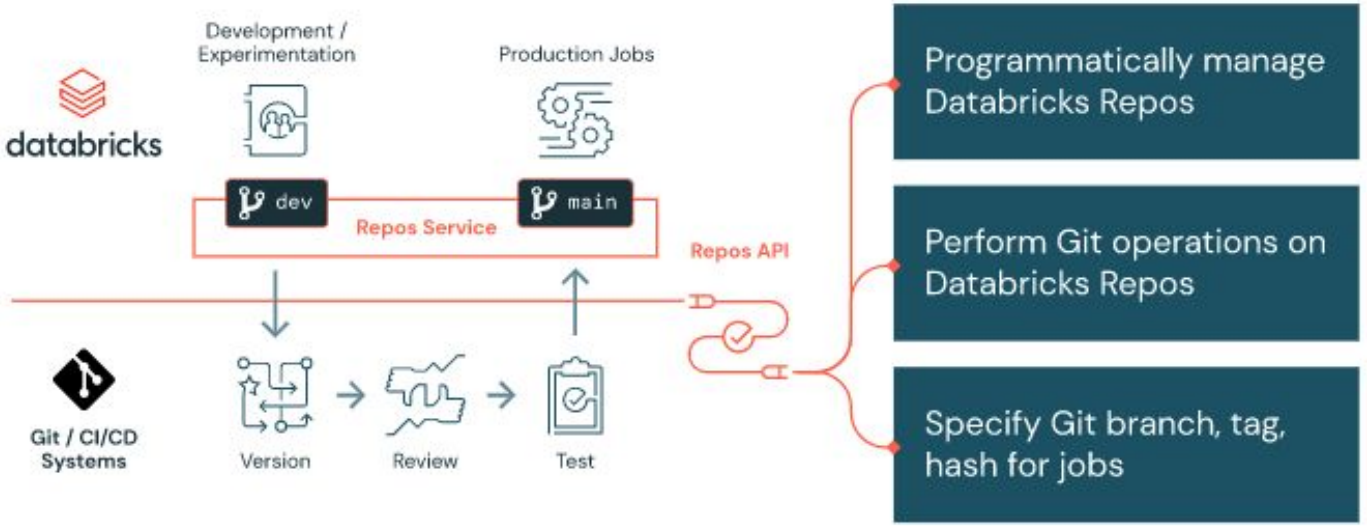
- One of the reasons to have different Workspaces for DEV and PROD is that they could reside in different VNets/VPCs. This is independent of UC, but leads to a setup as it is shown here.

<DA> DEV System Account (Service Principal, Instance Profile, Service Account)

<PA> PROD System Account (Service Principal, Instance Profile, Service Account)

<UG> User Group (Developers, Data Engineers, Data Scientists)

Source Control



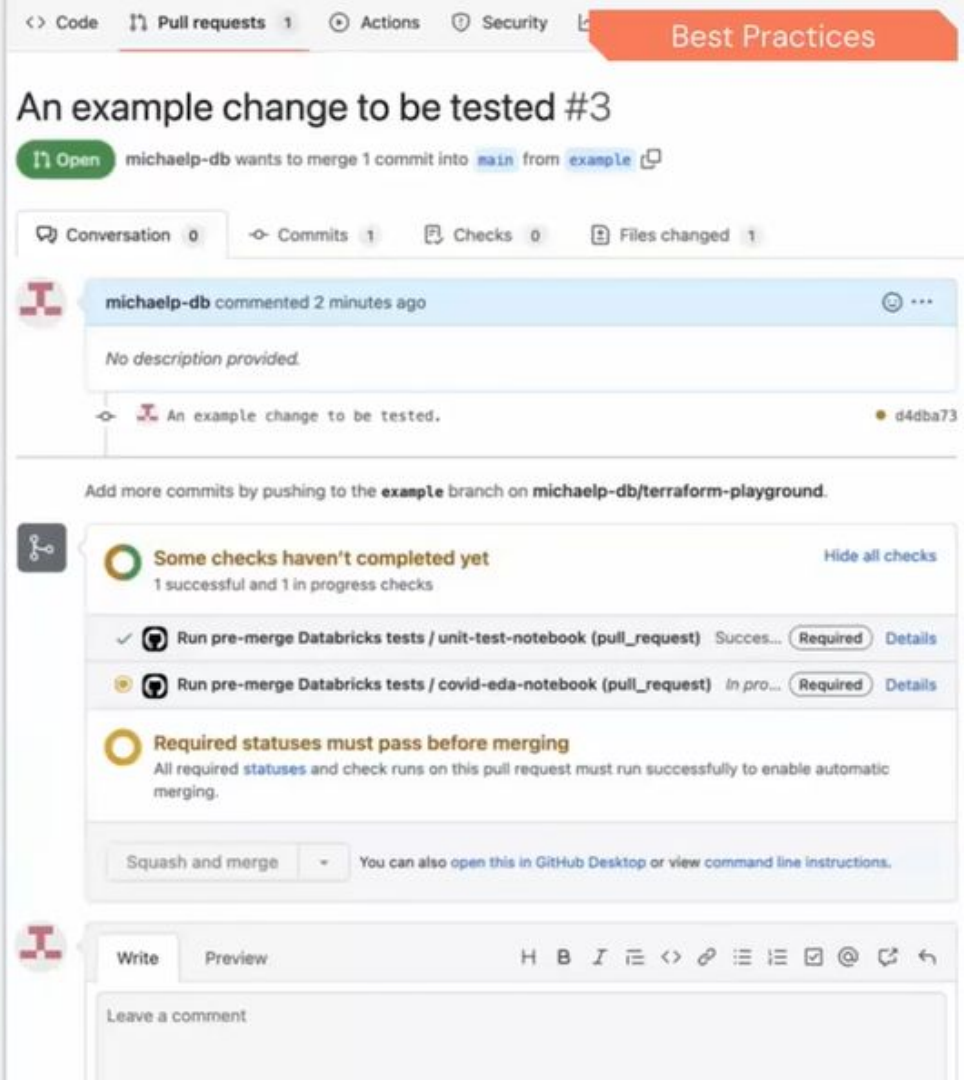
GitHub GitLab Azure DevOps AWS CodeCommit Bitbucket

Best practices: CI/CD

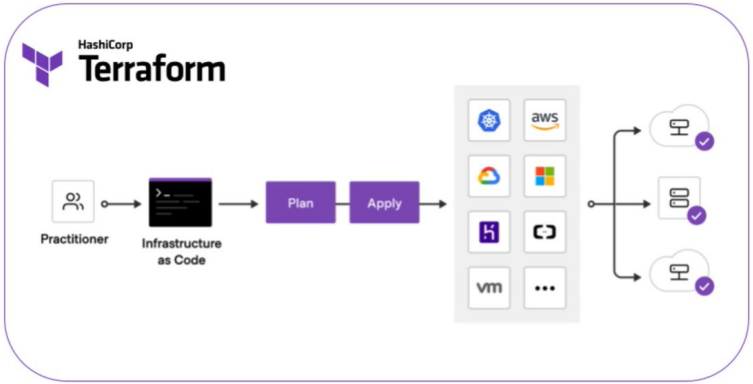
Automate code testing and deployment

Trigger continuous integration testing using the **Repos API** and support for git provider webhooks

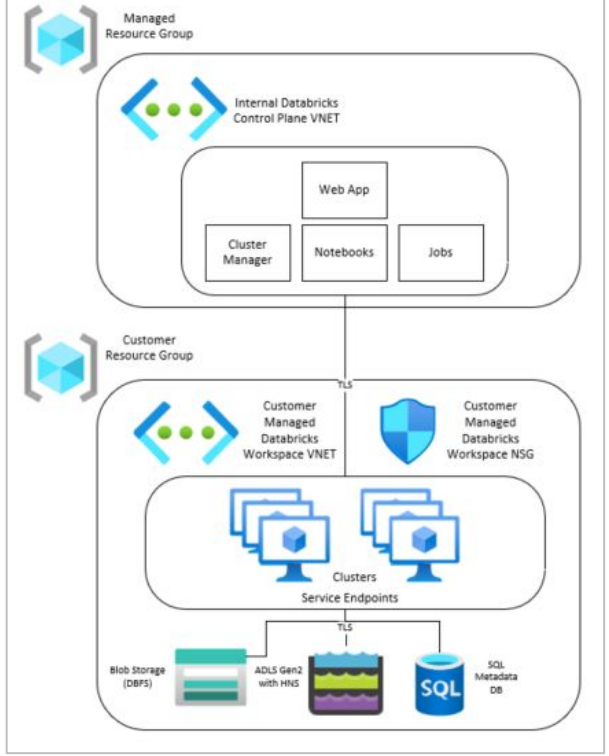
Continuously **deploy the latest production code using DABs** in your CI/CD pipeline, or the **Workflows + git source** integration



Infrastructure (As Code)



- Automate the creation of production-grade infrastructure
- Composable (expressed as JSON)
- Testable
- Releasable
- Reusable
- Repeatable
- Declarative (Get Projects done faster)
- Open Source (Wide adoption)
- Cloud Agnostic



DABs: A Summary

Public Preview

Databricks Asset Bundles (DABs) are YAML files that specify the artifacts, resources and configurations of a Databricks project.

Databricks Project

Code: Notebooks, Python .whl, JARs, dbt, etc.

Resources: Databricks Workflows, MLFlow, Delta Live Table Pipelines, etc.

Execution Environment: Databricks Workspace, Compute configurations,

Databricks Asset Bundle
bundle.yml

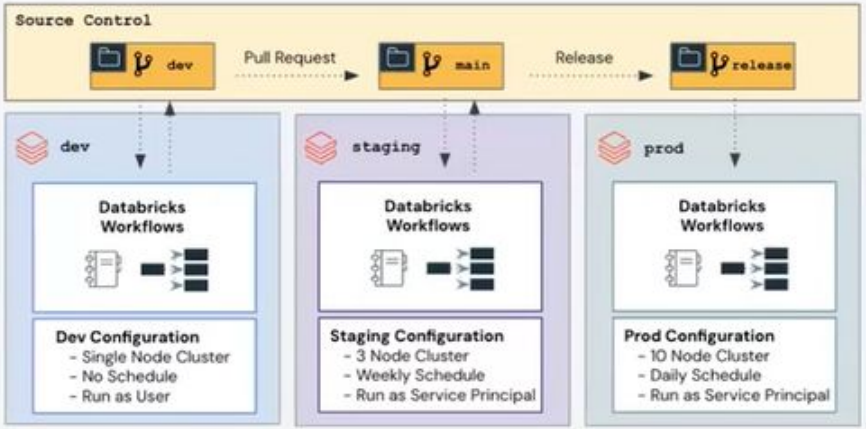
Workspace: Name and default workspace

Resource configurations : Jobs, DLT Pipelines, MLFlow, etc. Additional parameters follows REST API Schema

Environment-based specs: Control project behaviour in different environments

You can deploy to multiple workspaces, run your projects, tweak configs, and test changes from your IDEs, Terminal or from within Databricks!

You can **install the Databricks CLI** and begin to **deploy your bundles on your CI/CD server**; you can also **trigger automatically with Git Actions**




```
$ databricks bundle deploy -e "dev"
$ databricks bundle run pipeline -refresh-all -e "dev"
```

Example Project Structure

Deployment & Configurations

Source Code

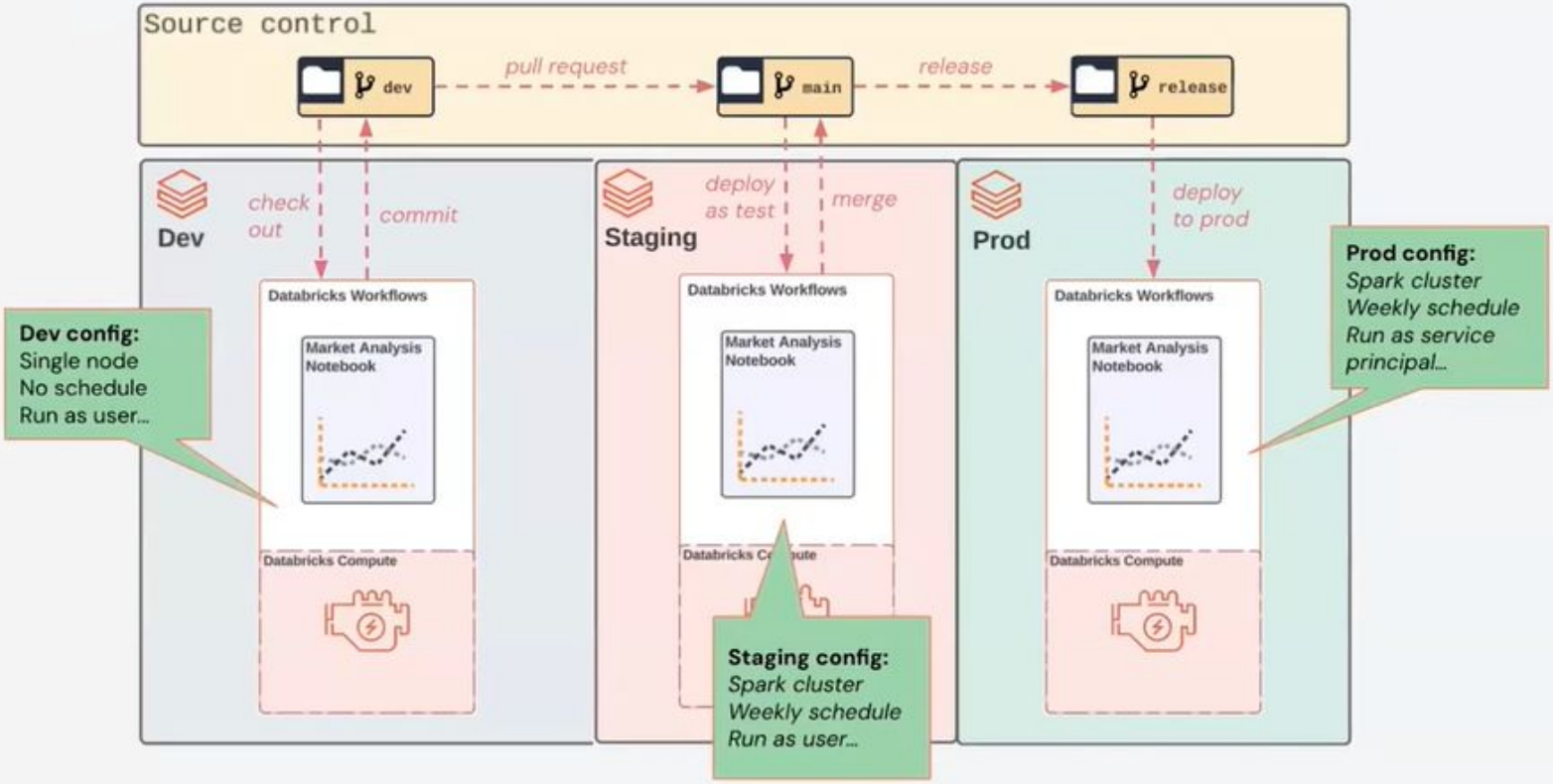
Tests

Dependencies

Basic Python with DABs

```
├── resources/
│   ├── job.yml
│   ├── dlt_pipeline.yml
├── databricks.yml
├── .gitignore
├── README.md
├── notebooks/
│   ├── dlt_pipeline.ipynb
│   ├── notebook.ipynb
├── customer_unification/
│   ├── load_config.py
│   └── cleaning.py
├── tests/
│   ├── main_test.py
├── fixtures/
│   ├── test_data.csv
├── requirements.txt
└── test-requirements.txt
```

The journey to production



Upgrade Strategies

- Upgrade Readers and Gold Zone first
- Upgrade ML
- Upgrade DBSQL
- Upgrade ETL

Demo Scenario

HighTech Company - analyzing app user access patterns

John
Data Engineer
Owns pipeline, wants lineage
Upgrade readers, then writers

→ Upgrade, Lineage

Alice
BI Analyst
Wants to analyze data across platforms
Join with federated query to Snowflake

→ Federation

Emily
Unified Governance and Security
Protect PII, comply with GDPR and Reg
Apply RLS/CLS

→ RLS / CLS

Marc
Platform Owner
Cares about cost trends and chargeback
Leverage System tables

→ System Tables

Scenario 1 - Customer with Jobs against HMS

Applies to almost every customer, low barrier to entry

Problem

- Customer has a data engineering team that produces data using Workflows or another orchestration tool.
- The customer has still reliance on non databricks data producers (EMR)
- Customer has an analyst team that consumes data and creates their own derivative assets.
- Customer has 10s of jobs creating 100s of data assets.
- Customer has 25 Analysts consuming in 6 different groups
- Customer's Analysts, typically, store data in DBFS

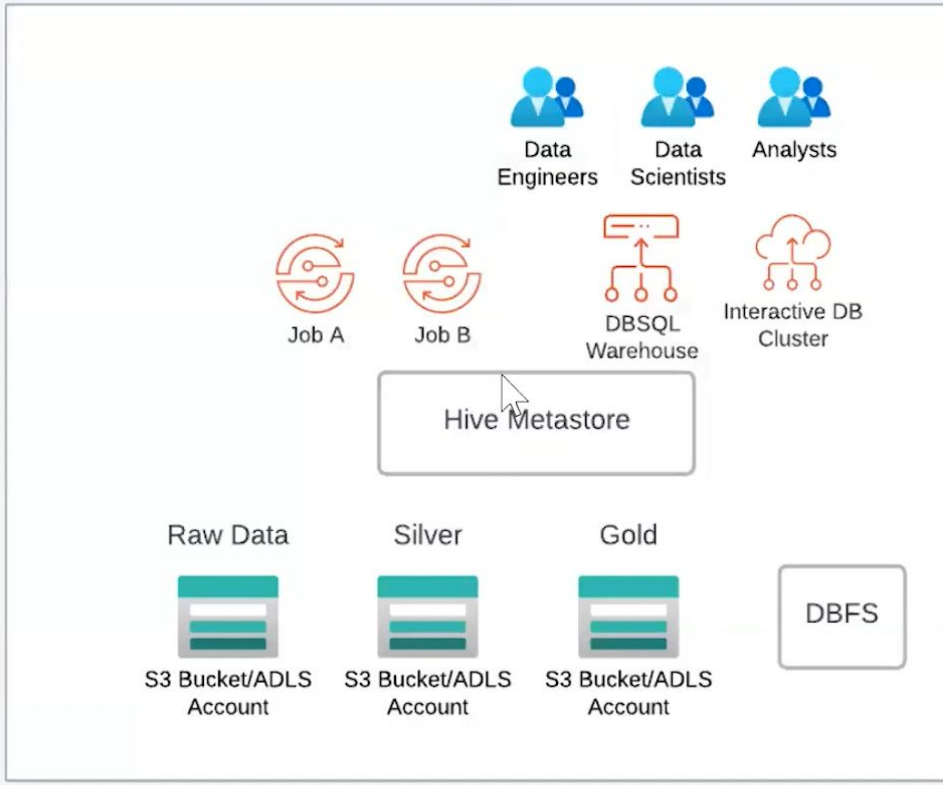
Solution

- Leave Jobs running against Hive Metastore
- Create a new Job, that can be called from orchestration, to SYNC assets after every Job is complete
- Upgrade compute, give analysts access to only shared compute.
- Introduce Serverless SQL Warehouses
- Migrate DBFS tables using CTAS statements
- Set up ACLS on data assets

Scenario 1

Before Architecture

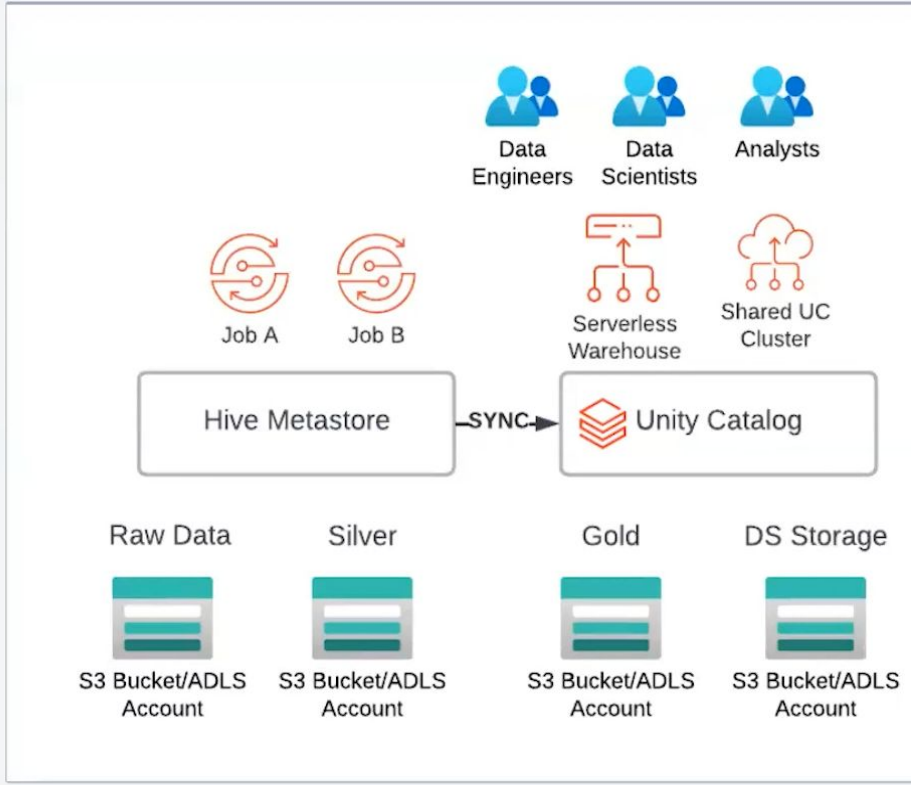
1. Data ingestion into cloud storage
2. All the metadata in HMS
3. Ingestion is performed by Jobs
4. Ingestion by external tools
5. Data Engineers save data to DBFS
6. Analysts use DB SQL Warehouses



Scenario 1

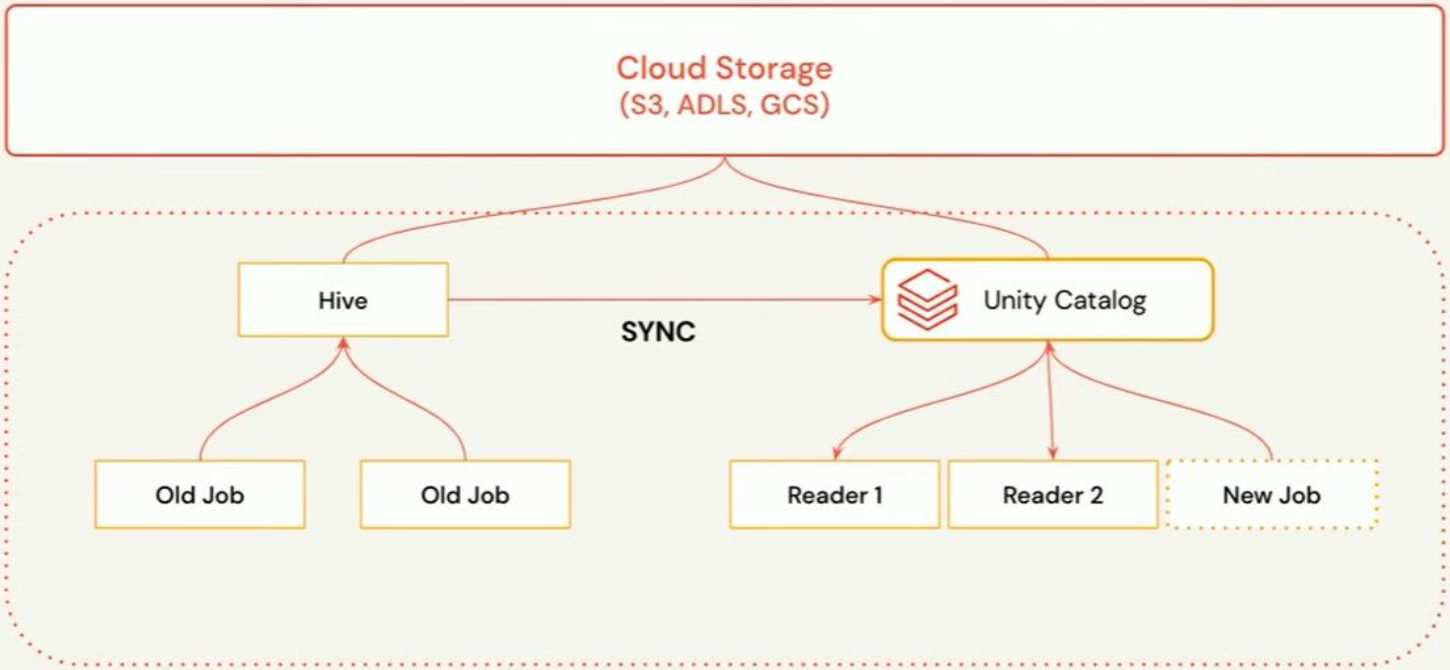
After Architecture

1. Metadata is SYNC'd to UC
2. ACLs are set on the Data
3. All data is consumed through UC
4. Data from DBFS is now in Cloud Storage
5. HMS federation will address similar usercases



Keep your jobs

Bring your readers



Scenario 2 - Customer with ML Workloads

Applies to many of our customers

Problem

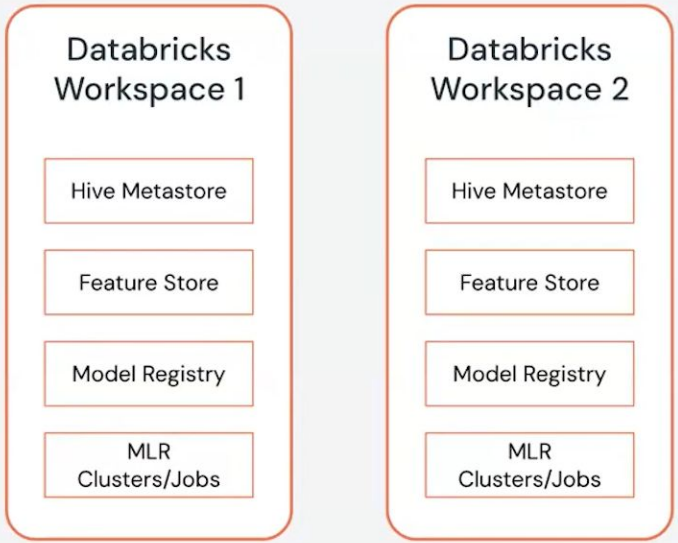
- Customer has a data engineering team that produces data using Workflows or another orchestration tool.
- Customer has an ML Engineering Team that produces Features
- Customer has a Data Science Team that produces machine learning/AI models.
- Customer has X number of jobs creating Y number of data assets.
- Customer has A number of Data Scientists consuming Y number of data assets and producing Z number of machine learning models.

Solution

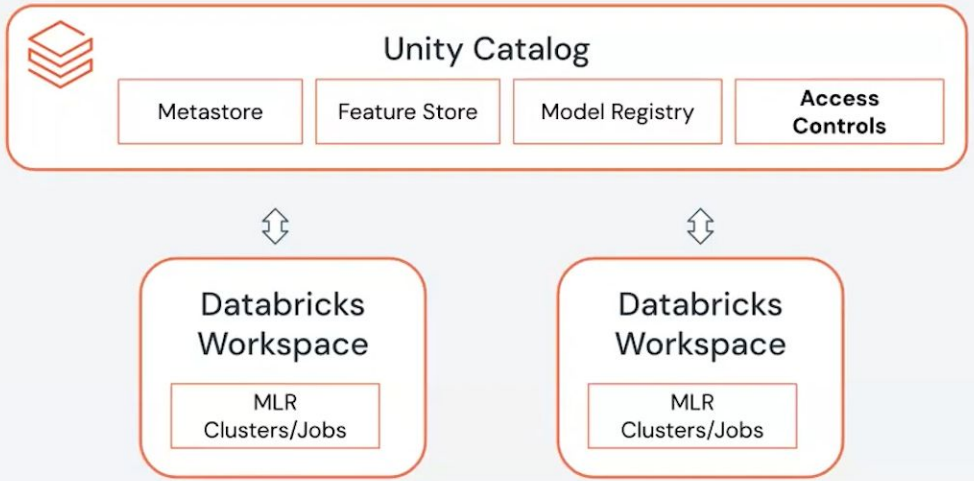
- Leave Jobs running against Hive Metastore
- Create a new Job, that can be called from orchestration, to SYNC assets after every Job is complete
- Upgrade compute, give Data Scientists access to shared compute so they can create features and prep training data.
- Store training data in UC Volumes.
- Store features in UC feature store
- Allow data scientists to access single user compute to train their models and store in the UC feature store.

UC + ML Operating Model

Without Unity Catalog



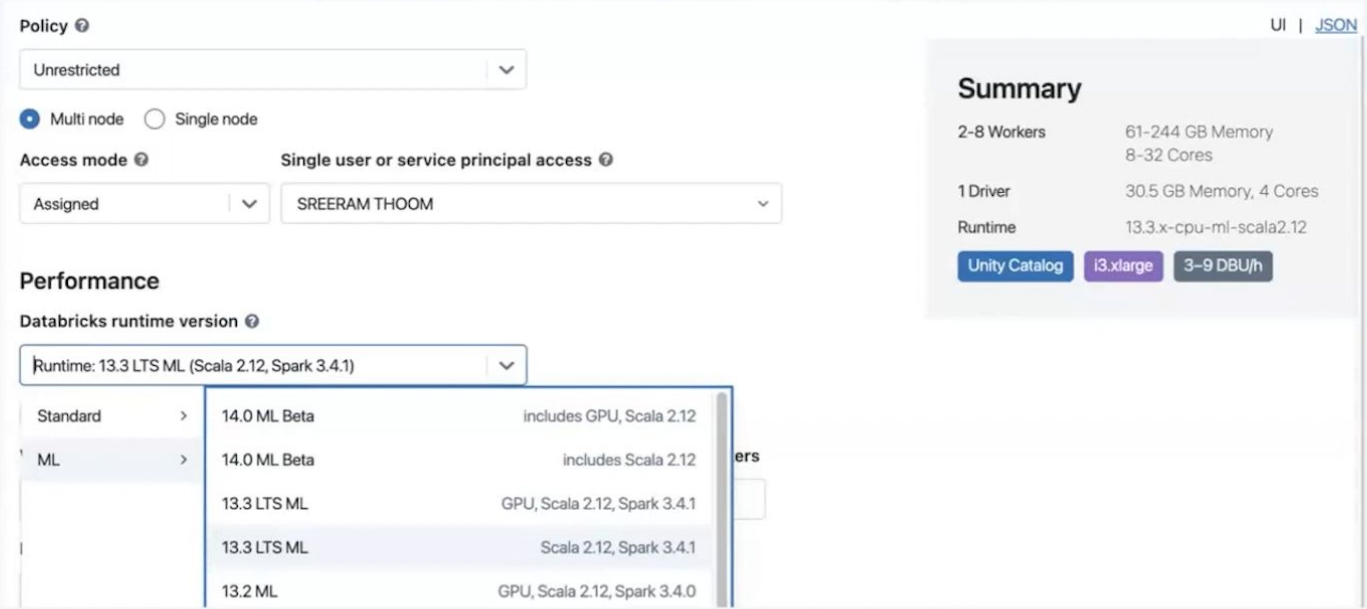
With Unity Catalog



Upgrade ML Workloads to Unity Catalog

ML Runtime

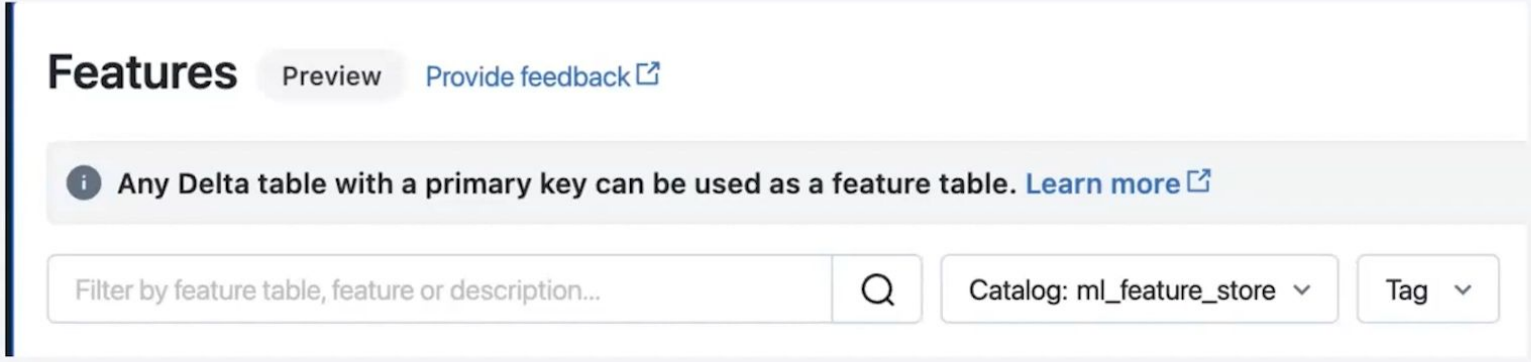
- Choose ML Databricks runtimes
- Currently UC supports Single User access mode clusters
- In future, Improved Shared clusters will support MLR



Upgrade ML Workloads to Unity Catalog

Unity Catalog Feature Store

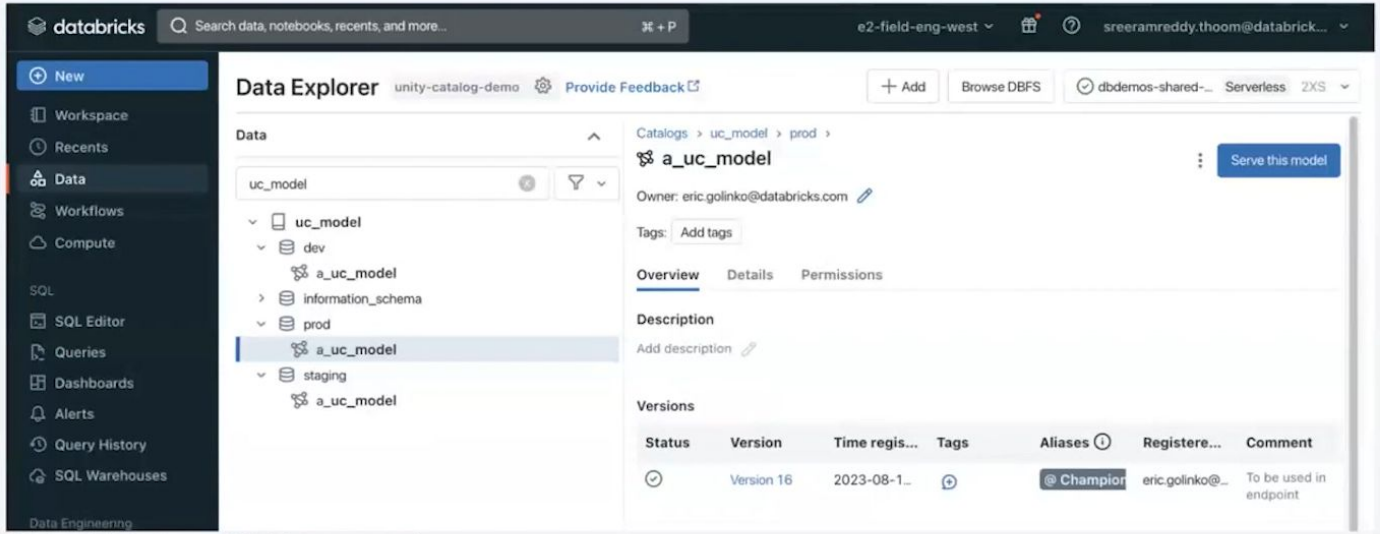
- Search / view feature tables in Feature Store UI
- Manage feature tables in Data Explorer
- Share feature table across workspaces
- Unified Permission Control over feature data and the metadata
- Existing Delta table in Unity Catalog as a feature table by declaring a Primary Key Constraint on the table
- Tagging



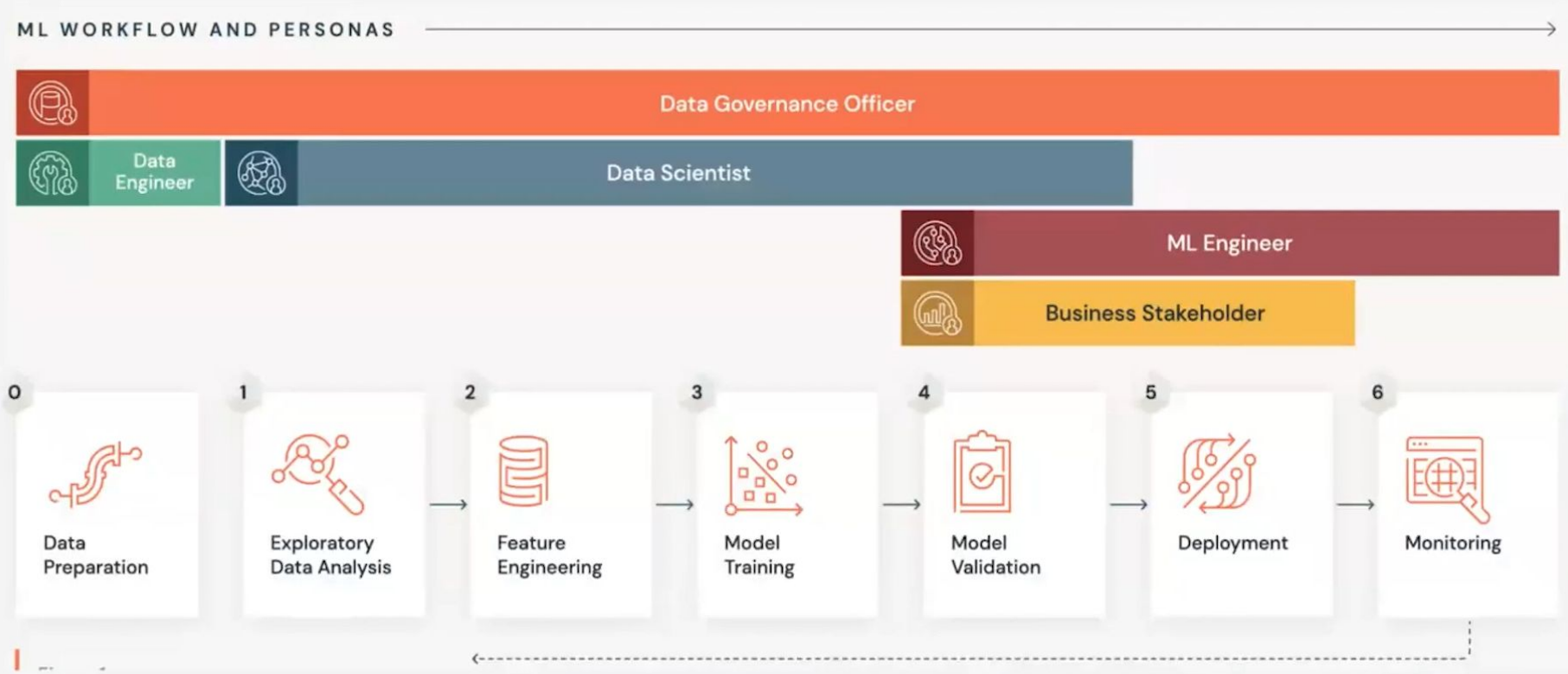
Upgrade ML Workloads to Unity Catalog

Unity Catalog Model Registry

- Centralized Model registry shared across workspaces
- Model aliases
- UC Lineage, Auditing and ACLs over Models
- MLFlow client configuration mlflow.set_registry_uri("databricks-uc")

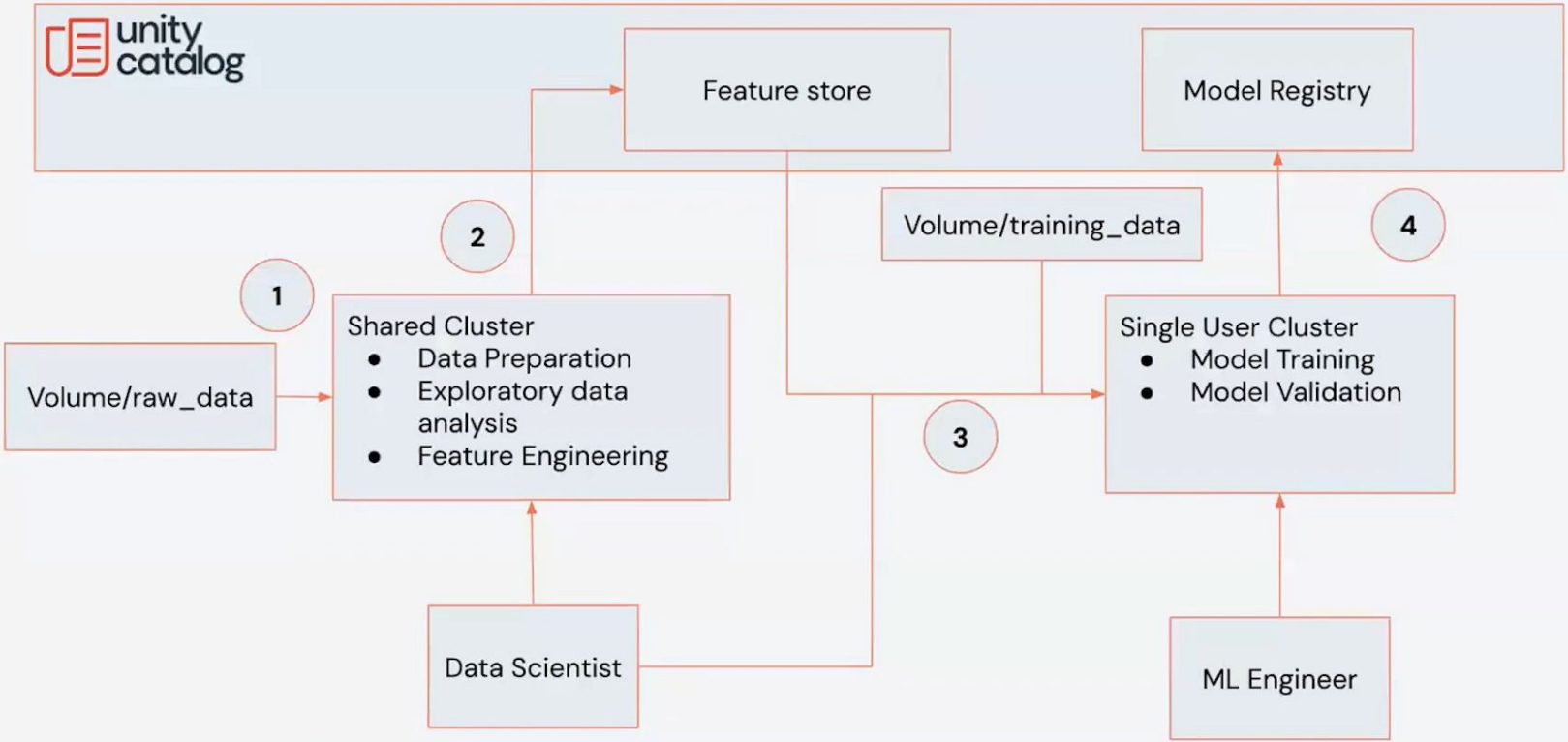


UC + ML Operating Model



UC + ML Operating Model

Features and ML Models



Scenario 3 - Customer with DBSQL Workloads

Applies to almost as many customers

Problem

- Customer has a data team that produces data using tools like DBT or fivetran.
- Customer analyst team also consumes data and creates their own derivative assets.
- Customer has X number of DBT/fivetran workflows creating Y number of data assets.
- Customer has A number of Analysts consuming Y number of data assets.
- Customer's Analysts store data in DBFS

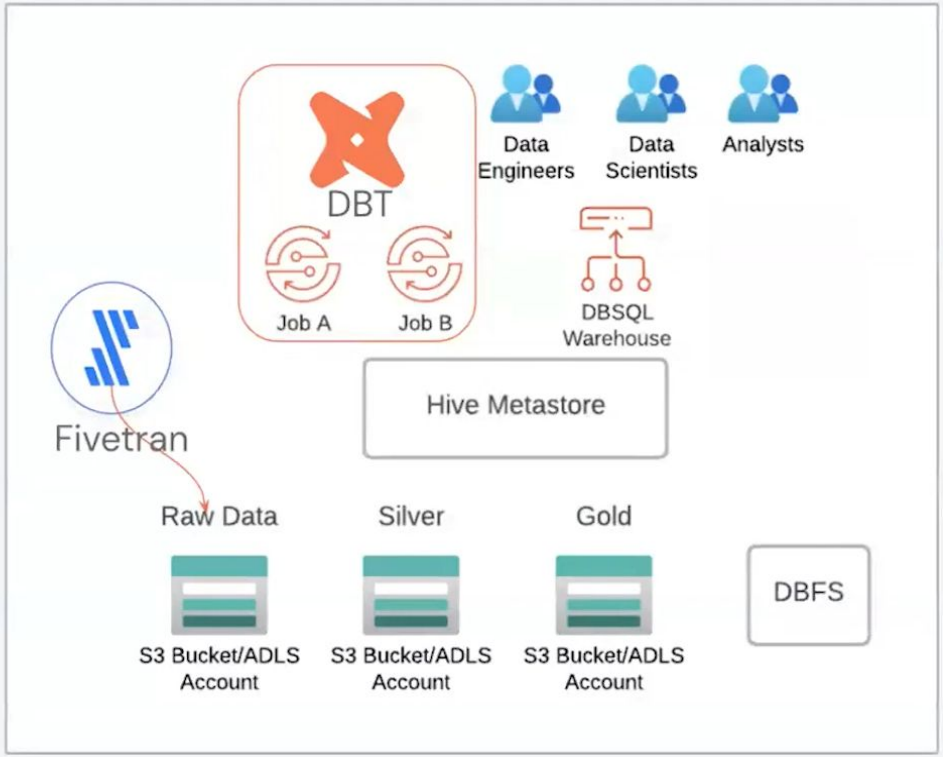
Solution

- Leave DBT/Fivetran running against Hive Metastore
- Create a new Job, that can be called from orchestration, to SYNC assets after every Job is complete
- Leave DBT/Fivetran SQL Warehouses alone, do not turn UC on.
- Create new warehouses for users to query data from Unity Catalog

Scenario 3

Before Architecture

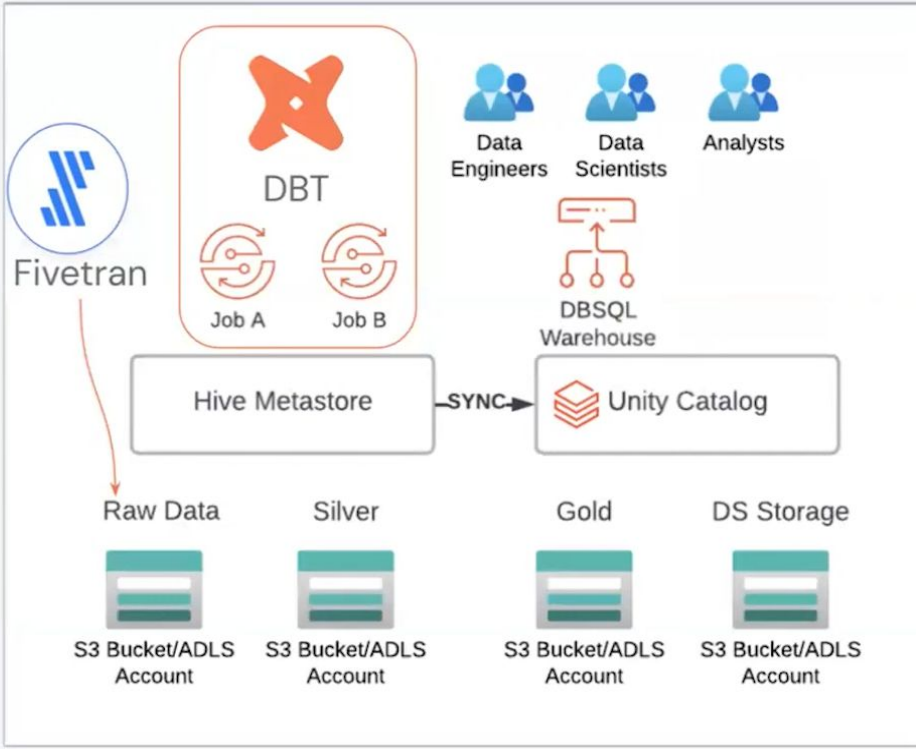
1. Data ingestion into cloud storage via Fivetran
2. All the metadata in HMS
3. Ingestion is performed by Fivetran
4. Data Engineers use DBT and save data to Cloud Storage
5. Analysts use DB SQL Warehouses and stores data in DBFS



Scenario 3

After Architecture

1. Metadata is SYNC'd to UC
2. ACLs are set on the Data
3. All data is consumed through UC
4. Data Analysts now stores data in Cloud Storage
5. Set a default catalog for UC either workspace level or with your connection URL while accessing data from external systems



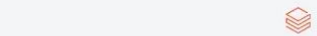
What if…

Tooling could use this to help you auto-magically?

UC Job Upgrade Assistant

- Connect a workspace
- Downloads jobs and notebook definitions
- Runs assessments on those notebooks
- Provides you a guided workflow UI with suggestions
- Can automatically upgrade a notebook if it has enough information
- Push changes to a new job in the workspace, cutover when ready.

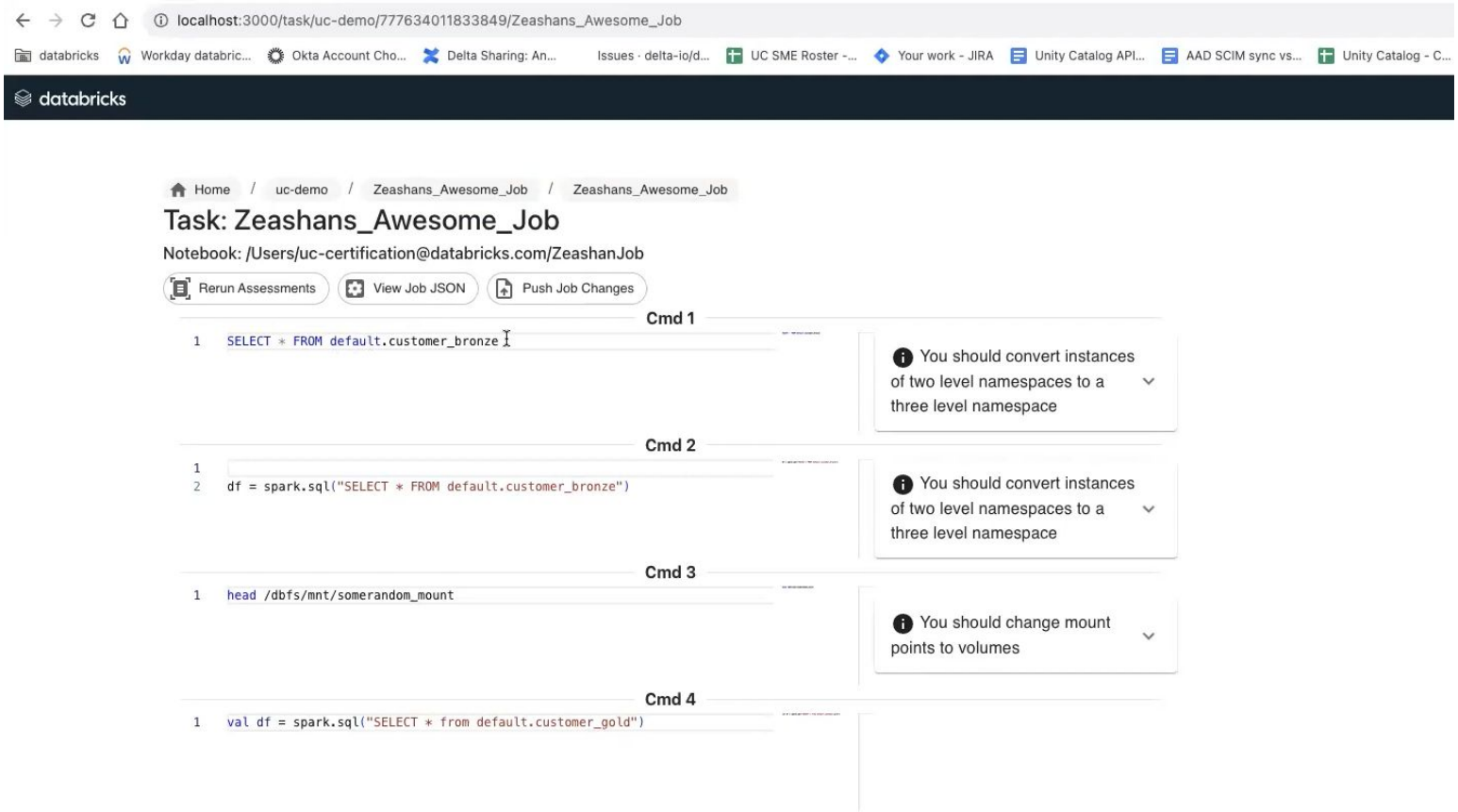
- **More Details**
  - Uses output from Assessment Profiler and HMS Lineage
  - Parses Python and SQL notebooks into ASTs that are evaluated to produce display output (highlights and tool tips and suggestions)
  - Completely extensible, register your own functions
  - Code highlighting in cells
  - Only needs Python to run



HMS Lineage to the rescue

Turn it on and get lineage results to help you upgrade

- In Private Preview, opt your customer in
- Works back to DBR 9.1, works **best** on 11.3 LTS or higher
  - Table to Table Lineage
  - Table to View Lineage
  - Filesystem to Table Lineage
  - Table to Notebook Lineage
  - Who created a table or view


localhost:3000/task/uc-demo/777634011833849/Zeashans_Awesome_Job
databricks
Workday databric...
Okta Account Cho...
Delta Sharing: An...
Issues · delta-io/d...
UC SME Roster -...
Your work - JIRA
Unity Catalog API...
AAD SCIM sync vs...
Unity Catalog - C...
databricks
Home / uc-demo / Zeashans_Awesome_Job / Zeashans_Awesome_Job
Task: Zeashans_Awesome_Job
Notebook: /Users/uc-certification@databricks.com/ZeashanJob
Rerun Assessments
View Job JSON
Push Job Changes
Cmd 1
1 SELECT * FROM default.customer_bronze
You should convert instances of two level namespaces to a three level namespace
Cmd 2
1
2 df = spark.sql("SELECT * FROM default.customer_bronze")
You should convert instances of two level namespaces to a three level namespace
Cmd 3
1 head /dbfs/mnt/somerandom_mount
You should change mount points to volumes
Cmd 4
1 val df = spark.sql("SELECT * from default.customer_gold")

Default Unity Catalog Limits

These can be raised - require an exception

- 1 Metastore per Cloud Region
- 1,000 Catalogs / Metastore
- 200 Storage Credentials / Metastore
- 500 External Locations / Metastore
- 1,000 Functions / Schema
- 10,000 Volumes / Schema
- 100,000 Tables / Metastore
- 100,000 Volumes / Metastore
- 1,000 Shares/Metastore
- 1,000 Tables/Share
- 1,000 Volumes/Share
- 5,000 Account Groups
- 10,000 Account Users

Current Previews

Opt your customer in to get the best of Unity Catalog

1. Volumes (Public Preview)
2. Row Level Filtering/Column Level Masking (Private Preview, Public Soon)
3. Purview Integration (Microsoft Preview)
4. System Tables (Public Preview)
5. CMK Header (Private Preview)
6. HMS Interface (Private Preview)
7. Lakehouse Monitoring (Gated Preview)
8. Browse Permission (Private Preview)
9. Scala UDF on shared clusters (Private Preview)
10. Cross Cloud Storage Location (Private Preview)

Upcoming Q3 Features

All of these are planned P0/P1 Priorities

- UC by Default
- Metastore Replication for DR (Cross Region)
- Dynamic Table Types (External to Managed Switching)
- Purpose Based Access Control
- Parquet Performance Enhancement
- HMS Federation
- HMS Interface Enhancements

UC Upgrade Recommendations (So far)

- Create Environmental workspaces (e.g. Dev, QA, Prod) in West US (and perhaps East US too).
    - We recommend creating these via Terraform and CI/CD. Existing workspaces will be "prod", we will create new lower environment workspaces.
- Originate all code and Databricks jobs from Source Control and CI/CD, and push them through the above environments.
    - All notebooks will be exported and added to git.
    - Jobs will be updated to be git based instead of workspace based.
- Set up SSO to Databricks and organize Azure AD with an "All Databricks Users" Group that will be synced via SCIM to Databricks.
    - Dimitry and team will slice this group up into nested groups within Databricks as desired.
- Enable UC by performing all Account level UC Configuration: Metastores, Catalogs, Credentials, SCIM, Groups, Admin Roles, Ownership, Identity Federation, External Locations etc.
- Next Steps: Plan Table, Cluster, Job upgrade strategy