UNITY CATALOG :-

- Allows us to define our data access rules only once, and then apply them to multiple workspaces, clouds, languages and use cases.

- Provides access controls for all Managed Data Assets, including files, tables, rows and columns within tables, views.

- Access control rules are defined as a property of a workspace, in the past, making it a challenge to apply on multiple workspaces.
  But, Unity Catalog lives outside of the workspace, thus spans multiple/all workspaces, and clouds.

- Data can freely roam from system to system, with all the access rules being applicable on it everywhere, only by defining the rules just once.

- Unity Catalog integrates seamlessly with other catalogs, such as Hive Metastore.

- UC enables doing fine-grained auditing of all queries performed.

⇒ Data lineage of all tables & columns are collected and visualised by UC.

⇒ Key Concepts :-

⇒ **Metastore** (top-level logical container of Unity Catalog) Contructs that represents the metadata, as well as the access-control lists.

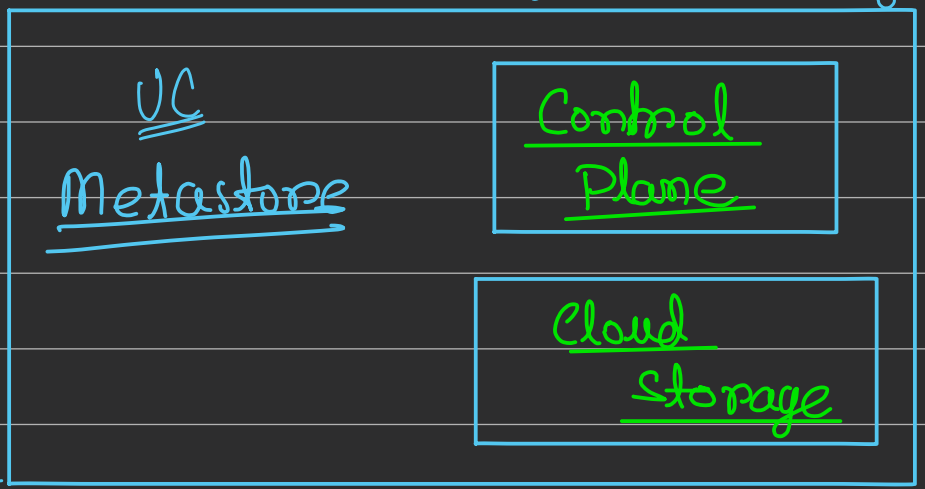
Unity Catalog v/s Hive Metastore :-

⇒ Hive metastore is the traditional default metastore of Databricks, linked to each Databricks workspace

⇒ Unity Catalog metastore is central and is not linked to any individual workspace.

⇒ Metadata of the table objects, and the ACLs are stored in the Control Plane [This ties metastore to a cloud region]

⇒ UC metastore is a logical construct for organising all the data and their metadata.
(UC metastore is not a physical container.)



This ties a metastore to a Cloud Storage, and thus, we need to select a region, while configuring the Unity Catalog metastore.

UC catalog metastore data is stored externally in a cloud storage, and thus, it lives outside of the databricks workspaces.

⇒ Catalog:

⇒ Topmost container for data objects.
⇒ 1st part of the 3 level namespace.

⇒ One metastore can have many Catalogs.
⇒ Can only contain Schemas.

⇒ Storage Credentials:

⇒ As all data lives in the Cloud, Unity Catalog needs a way to authenticate with the Cloud Storage Containers.

⇒ Storage Credentials encapsulates an authentication method to access cloud storage.
Each metastore has one such built-in authentication method, to access its own internal cloud storage.

⇒ Additional methods can be added, in case of accessing data, external tables, etc.

⇒ Storage Credentials are not very convinient, to access external cloud storage location, as it accesses the entire container.

⇒ Architecture :-

⇒ Previously, User/Group Management was a function of the workspace.
This created problem, as Metastores were localised.

So, with Unity Calalog, all security-related elements are factored out of the workspace.

⇒ Users, Groups, Metastores for the Unity Catalog are managed through the account console manually, and then assigned to one or more workspaces, enabling multiple workspaces to share same ACLs.

Compute Resources can directly connect to Unity Catalog, and thus, no seperate configs needed.

⇒ Any changes in policies in the Metastore, are immediately propagated to assigned workspaces

⇒ Thus, UC moves the Security Related Aspects out of the Workspace.

⇒ Prerequisite to create a metastore :-

i) Cloud storage Container

ii) Cloud credential that allows Databricks to access the container.

⇒ Metadata is maintained in the Control Plane and must align geographically with the workspaces, to which metastore will be aligned.

⇒ There can be one metastore per region. If desired region is not available, it is possible that a metastore with that region is already there.

The container that we pass while creating metastore, is the place where managed data files will be stored.

⇒ Whenever we assign a Unity Catalog metastore to any Workspace, the Data Explorer of that workspace gets updated / completely changed, to show new view of Data, Catalogs, etc.

From Data Explorer itself, then, we can view and manage permissions to database object.

Show granted permissions :-

```
SHOW GRANTS ON SCHEMA main.default;
```

main → catalog name

default → schema name

Updated Points :-

⇒ Once a workspace is connected to Unity Catalog through assignment by Unity Catalog Metastore, UC then presents the existing Hive metastore, as a special catalog named as hive_metastore.

Assets existing inside the Hive metastore can then be accessed using hive_metastore catalog name, as the 1st part of the 3-level naming convention.

⇒ Tables = Metadata + Data

(Relational Entity.)

(Column names & data types, properties, owner, etc.)

Files

The data that appear as records

Decoupling tables into metadata & datafiles is important, and forms the basis b/w categorisation of Managed & External Tables.

⇒ External Location :-

⇒ External Locations are build on a concept of Storage Credential., and is extended by a Storage Path to the cloud storage container.

These allow users to subdivide container into smaller pieces, and exercise and manage seperate controls over them.

⇒ External cloud container is directly not accessible to users, that contains multiple directories and sub-directories.
The whole container may contain data related to different aspects, and then, any specific individual getting acess to entire container will be meaningless.

Thus, external locations are made to point specific directories and subdirectories within the container, and using that, only that specific path/paths will be accessible.

⇒ External locations can be used to support & manage External Tables.
Can also be used to govern direct access to files stored in cloud storage.

⇒ SHARES & RECIPIENT :-

⇒ Both are related to Delta Sharing. (secured protocol developed by Databricks for sharing data objects).

SHARES :- Read-only logical collection of tables.

RECIPIENTS :- Data reader outside the organisation, that has read permissions on SHARES.