Unity Catalog Quick Setup Guide

UberEats Data Intelligence Platform

This document provides step-by-step instructions for setting up the Unity Catalog in the UberEats project's development and production Databricks environments.

Environment Details

Workspaces

- **Development:** `ubereats-dev-workspace`
- **Production:** `ubereats-prod-workspace`

Storage Accounts

- **Development:** `adlsubereatsdev`
- **Production:** `adlsubereatsprod`

Containers (in each storage account)

- `landing`
- `bronze`
- `silver`
- `gold`

Setup Instructions

Step 1: Create Resource Groups

First, create resource groups to organize your Azure resources:

For metastore
Resource Group Name: rg-unity-catalog-metastore

For development environment
Resource Group Name: rg-ubereats-dev

For production environment
Resource Group Name: rg-ubereats-prod

Using Azure CLI:

```
# Login to Azure
az login

# Create resource groups
az group create --name rg-unity-catalog-metastore --location eastus2
az group create --name rg-ubereats-dev --location eastus2
az group create --name rg-ubereats-prod --location eastus2
```

Step 2: Create Metastore Storage Account

Create a dedicated storage account for the Unity Catalog metastore:

Resource Group: rg-unity-catalog-metastore
Storage Account Name: ucmetastoreubereats
Container Name: metastore

Using Azure CLI:

```
# Create storage account
az storage account create \
  --name ucmetastoreubereats \
  --resource-group rg-unity-catalog-metastore \
  --location eastus2 \
```

```
  --sku Standard_LRS \
  --kind StorageV2 \
  --enable-hierarchical-namespace true

# Create container
az storage container create \
  --name metastore \
  --account-name ucmetastoreubereats
```

Best Practice: Using a separate storage account for the metastore provides better security and access control.

Step 3: Create Metastore in Account Console

1. Navigate to Databricks Account Console
2. Go to **Data** → **Metastores** → **Create Metastore**
3. Fill in the details:
   - **Name:** `ubereats-metastore`
   - **Region:** Same as your workspaces (e.g., East US 2)
   - **Storage Location:** `abfss://metastore@ucmetastoreubereats.dfs.core.windows.net/`
4. Under **Storage Credentials**:
   - Click **Create new credential**
   - **Name:** `ucmetastore-credential`
   - **Authentication Type:** Access Key
   - Enter the access key for `ucmetastoreubereats`
5. Click **Create**

Step 4: Assign Metastore to Workspaces

1. In the Account Console, find your metastore
2. Click **Assign**
3. Select the development workspace: `ubereats-dev-workspace`
4. Enable Unity Catalog
5. Click **Assign**
6. Repeat for production workspace: `ubereats-prod-workspace`

Step 5: Create Storage Credentials in Each Workspace

#First, make sure your development and production storage accounts are created:

```
# Create dev storage account
az storage account create \
   --name adlsubereatsdev \
   --resource-group rg-ubereats-dev \
   --location eastus2 \
   --sku Standard_LRS \
   --kind StorageV2 \
   --enable-hierarchical-namespace true

# Create containers in dev storage
for container in landing bronze silver gold; do
   az storage container create \
      --name $container \
      --account-name adlsubereatsdev
done

# Create prod storage account
az storage account create \
   --name adlsubereatsprod \
   --resource-group rg-ubereats-prod \
   --location eastus2 \
   --sku Standard_LRS \
   --kind StorageV2 \
   --enable-hierarchical-namespace true

# Create containers in prod storage
for container in landing bronze silver gold; do
   az storage container create \
      --name $container \
      --account-name adlsubereatsprod
done
```

Then create storage credentials in each workspace:

Development Workspace:

1. Go to the development workspace: `ubereats-dev-workspace`
2. Navigate to **Data → Create → Storage Credential**
3. Fill in the details:
    - **Name:** `adlsubereatsdev-credential`
    - **Authentication Type:** Access Key
    - Enter the access key for `adlsubereatsdev`
4. Click **Create**

Production Workspace:

1. Go to the production workspace: `ubereats-prod-workspace`
2. Navigate to **Data → Create → Storage Credential**
3. Fill in the details:
    - **Name:** `adlsubereatsprod-credential`
    - **Authentication Type:** Access Key
    - Enter the access key for `adlsubereatsprod`
4. Click **Create**

Step 6: Create Catalogs

In each workspace, create the following catalogs:

1. Navigate to **Data → Create → Catalog**
2. Create each of these catalogs:
    - `ubereats_operations` (comment: "Operations data domain")
    - `ubereats_finance` (comment: "Finance data domain")
    - `ubereats_ml` (comment: "Machine Learning data domain")

> **Best Practice:** Organize catalogs by business domain to maintain a logical separation of data.

Step 7: Create Schemas for Each Catalog

For each catalog, create the following schemas:

1. Select the catalog → **Create → Schema**
2. Create these schemas in each catalog:
    - `bronze` (comment: "Raw data layer")
    - `silver` (comment: "Validated and transformed data")
    - `gold` (comment: "Business-ready data")

Step 8: Create External Locations

Development Workspace:

1. Navigate to **Data → Create → External Location**

Create the following external locations:

Name: adlsubereatsdev-landing
URL: abfss://landing@adlsubereatsdev.dfs.core.windows.net/

Credential: adlsubereatsdev-credential
Comment: Landing zone for raw files
Name: adlsubereatsdev-bronze
URL: abfss://bronze@adlsubereatsdev.dfs.core.windows.net/
Credential: adlsubereatsdev-credential
Comment: Bronze layer storage
Name: adlsubereatsdev-silver
URL: abfss://silver@adlsubereatsdev.dfs.core.windows.net/
Credential: adlsubereatsdev-credential
Comment: Silver layer storage
Name: adlsubereatsdev-gold
URL: abfss://gold@adlsubereatsdev.dfs.core.windows.net/
Credential: adlsubereatsdev-credential
Comment: Gold layer storage

2.

Production Workspace:

Repeat the same process with the production storage account `adlsubereatsprod`.

Step 9: Create Volumes

For each catalog and schema combination, create corresponding volumes:

1. Select a catalog (e.g., `ubereats_operations`)
2. Go to **Create → Volume**

Fill in the details for each volume:

Name: landing
Volume Type: External
External Location: adlsubereatsdev-landing (or prod equivalent)
Comment: Landing data for operations

3.
4. Repeat for `bronze`, `silver`, and `gold` in each catalog

Note: In production, use the production external locations.

Step 10: Set Up Access Control

Create Groups:

1. Go to **Admin Settings → User Management → Groups**
2. Create the following groups:
    - `data-engineers`
    - `data-scientists`
    - `analysts`
3. Add appropriate users to each group

Set Permissions:

1. Navigate to **Data**
2. For each catalog, click **Permissions**
3. Grant the appropriate permissions:
    - `data-engineers`: `USAGE`, `CREATE`, `MODIFY`
    - `data-scientists`: `USAGE`, `SELECT`
    - `analysts`: `USAGE`, `SELECT`
4. Click **Save**
5. Repeat for schemas and external locations as needed

> **Best Practice:** Follow the principle of least privilege - grant only the permissions necessary for each role.

Step 11: Verify Setup

Create a test notebook and run the following commands:

```
# List all catalogs
spark.sql("SHOW CATALOGS").display()

# List schemas in operations catalog
spark.sql("SHOW SCHEMAS IN ubereats_operations").display()

# Check access to volumes
dbutils.fs.ls("/Volumes/ubereats_operations/bronze/")

# Try creating a test table
spark.sql("""
CREATE TABLE ubereats_operations.bronze.test_table (
  id INT,
  name STRING,
  timestamp TIMESTAMP
)
```

""")

```
# Insert test data
spark.sql("""
INSERT INTO ubereats_operations.bronze.test_table
VALUES (1, 'Test Record', current_timestamp())
""")

# Query the data
spark.sql("SELECT * FROM ubereats_operations.bronze.test_table").display()
```

Best Practices Summary

1. **Separation of Concerns:**
    - Use separate storage accounts for metastore and data
    - Maintain distinct dev and prod environments
2. **Consistent Naming:**
    - Follow a consistent naming pattern across environments
    - Use descriptive names for catalogs, schemas, and volumes
3. **Security:**
    - Implement role-based access control using groups
    - Apply the principle of least privilege
    - Regularly audit permissions
4. **Organization:**
    - Organize catalogs by business domain
    - Follow the medallion architecture (bronze, silver, gold)
    - Use external volumes for integration with existing storage
5. **Governance:**
    - Enable audit logging from the start
    - Document data lineage and dependencies
    - Regularly review and maintain the catalog structure

Troubleshooting

Issue	Resolution
"Access denied"	Check group membership and permissions
Can't see Unity Catalog	Verify workspace is assigned to metastore
Can't create objects	Ensure user has CREATE permission
Can't access volume	Verify correct external location and permissions
Storage errors	Check storage credential and access key validity

---

Document Version: 1.0
Last Updated: April 8, 2025