LAB 01: Platform & Workspace Setup

Duration: ~40 min
Day: 1
After module: M01: Platform & Workspace
Difficulty: Beginner

Scenario

> *“It’s your first day on the RetailHub data team. Before you can start building the analytics platform, you need to set up your workspace: create a compute cluster, explore Unity Catalog, and upload the raw data files that will be used throughout the training.”*

Objectives

After completing this lab you will be able to: - Create and configure a Databricks compute cluster - Navigate Unity Catalog hierarchy (Catalog > Schema > Table/Volume) - Create and use Unity Catalog Volumes for raw file storage - Upload dataset files to a Volume - Run basic `dbutils` commands

---

Prerequisites

- Access to Databricks workspace (URL + credentials provided by trainer)
- Dataset files available in the Git repository (`dataset/` folder)

Part 1: Create Compute Cluster (~10 min)

Step 1: Navigate to Compute

1. In the left sidebar, click **Compute**
2. Click **Create compute**

<screen = Databricks left sidebar with “Compute” menu item highlighted>

Step 2: Configure Cluster

Use the following settings:

Setting	Value
Cluster name	`training_{your_name}`
Policy	Personal Compute (or as provided by trainer)
Access mode	Single User
Databricks Runtime	Latest LTS (e.g., 15.4 LTS)
Node type	As provided by trainer
Terminate after	60 minutes of inactivity

<screen = Compute creation form with fields filled in as described above>

Step 3: Start the Cluster

1. Click **Create compute**
2. Wait for the cluster status to change to **Running** (green circle)
3. This may take 2-5 minutes

<screen = Compute page showing cluster in “Running” state with green indicator>

> **Exam Tip:** Know the difference between **All-Purpose clusters** (interactive, development) and **Job clusters** (automated, ephemeral). Serverless compute is the default for SQL Warehouses and is becoming the default for notebooks.

Part 2: Explore Unity Catalog (~10 min)

Step 4: Open Catalog Explorer

1. In the left sidebar, click **Catalog**
2. You should see your catalog: `retailhub_{your_name}`

<screen = Catalog Explorer showing the catalog tree with retailhub_username catalog expanded>

Step 5: Explore the Catalog Hierarchy

Navigate through the hierarchy:

```
retailhub_{your_name}          -- CATALOG (top-level namespace)
  ├── bronze                   -- SCHEMA (database)
  │     ├── (tables)           -- TABLES
  │     └── (volumes)          -- VOLUMES
  ├── silver                   -- SCHEMA
  ├── gold                     -- SCHEMA
  └── default                  -- SCHEMA (auto-created)
```

1. Click on your catalog name to expand it
2. Click on `bronze` schema
3. Note the tabs: **Tables**, **Volumes**, **Functions**

<screen = Catalog Explorer showing expanded catalog with bronze/silver/gold schemas visible>

> **Exam Tip:** Unity Catalog uses a **3-level namespace**: `catalog.schema.object`. Always use fully qualified names in production code.

Step 6: Create a Volume

1. Click on `default` schema
2. Click **Create** > **Volume**
3. Configure:
    - **Name:** `datasets`
    - **Volume type:** Managed
4. Click **Create**

<screen = Create Volume dialog with name “datasets” and type “Managed” selected>

> **Exam Tip: Managed volumes** store data in the metastore-managed location. **External volumes** point to existing cloud storage paths. Both are governed by Unity Catalog permissions.

Part 3: Explore External Connections (~5 min)

Step 7: Navigate to External Data

1. In the left sidebar, click **Catalog**
2. In the top navigation, click **External Data** (or look in the left panel under your catalog)
3. Click **Connections**

<screen = Catalog Explorer with “External Data” > “Connections” view open>

Step 8: Understand External Connections

External Connections define how Databricks connects to external data sources (cloud storage, databases, APIs).

Review the Connections page and note: - A connection stores **credentials** and **endpoint** information - Connections are used by **External Locations** (cloud storage) and **Foreign Connections** (databases like PostgreSQL, MySQL, SQL Server) - To create a connection: **Create connection** > choose type > provide credentials

Connection Type	Use Case	Example
Storage	Access files on cloud storage	S3 bucket, ADLS container, GCS bucket
Database	Query external databases via Lakehouse Federation	PostgreSQL, MySQL, SQL Server

> **Note:** Creating actual connections requires cloud credentials. In this training, we explore the UI to understand the concept. Your trainer may demonstrate a live connection.

> **Exam Tip:** External Connections are part of Unity Catalog governance. They require `CREATE CONNECTION` privilege and are managed at the **metastore level**. Know the difference between **External Locations** (storage paths) and **Foreign Connections** (database endpoints for Lakehouse Federation).

Part 4: Upload Dataset Files (~10 min)

Step 9: Upload Files to the Volume

1. Navigate to **Catalog** > `retailhub_{your_name}` > `default` > **Volumes** > `datasets`
2. Click **Upload to this volume**
3. Upload the following files from the `dataset/` folder:
   - `customers/customers.csv`
   - `products/products.csv`
   - `orders/orders_batch.json`

<screen = Volume file browser showing uploaded CSV and JSON files with file sizes>

Step 10: Verify Files with dbutils

Open the lab notebook (`LAB_01_code.ipynb`) and complete the tasks there.

Part 5: Notebook Tasks

Open **`LAB_01_code.ipynb`** and complete all `# TODO` cells.

Tasks in the notebook: 1. Run the setup cell (`%run ../setup/00_setup`) 2. List files in your Volume using `dbutils.fs.ls()` 3. Read the customers CSV using `spark.read` 4. Display the first 5 rows 5. Check the catalog/schema context with `SELECT current_catalog(), current_schema()`

Summary

In this lab you: - Created a compute cluster configured for the training - Explored the Unity Catalog 3-level namespace - Created a managed Volume for raw file storage - Uploaded dataset files and verified access via `dbutils`

> **What’s next:** In LAB 02 you will use these files to build your first ELT ingestion pipeline — reading CSV/JSON, transforming data, and writing Delta tables.