



## Setup Guide

# Onehouse Catalog Sync for Databricks Unity Catalog

Jan 2025

## Contents

<b>Introduction</b>	<b>2</b>
<b>Prerequisites</b>	<b>2</b>
<b>Create a Databricks workspace</b>	<b>3</b>
<b>Create a Unity Catalog</b>	<b>5</b>
Create an S3 Bucket for the metastore	5
Create an IAM role to access the storage location	5
Create a Unity Catalog metastore	8
Configure the Unity Catalog for CORS	11
<b>Connect the Workspace to the Onehouse data storage location</b>	<b>12</b>
Create and attach IAM policy for Onehouse customer storage access	12
Create a Databricks Storage Credential	13
Create a Databricks External Location	15
Gather the external catalog information in Databricks	17
<b>Create the Metadata Catalog Sync in Onehouse</b>	<b>19</b>
Create the Databricks metadata catalog in Onehouse	19
Attach the Onehouse Databricks catalog to one or more Stream Captures.	20

## Introduction

This guide provides a step-by-step recipe for setting up a Onehouse metadata catalog sync to [Databrick's Unity Catalog](#).

This provides a unified governance solution for managing and securing data assets across cloud environments. With [Onehouse's OneTable](#) support, users can ingest, and store data in Delta Lake format, and these tables automatically sync to Unity Catalog. This ensures seamless management of metadata, permissions, and lineage in a centralized, secure catalog.

The setup for Databricks Unity Catalog sync setup in Onehouse is relatively easy – the majority of this guide describes how to set up the AWS IAM roles and policies, and Databricks external catalog location.

## Prerequisites

You will need the following:

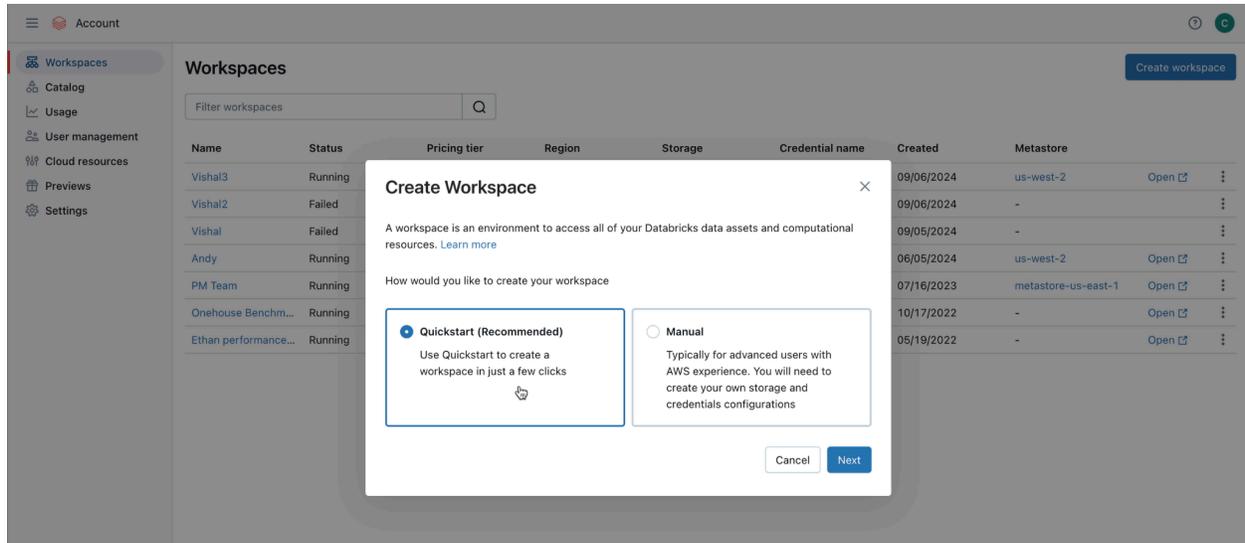
- A Onehouse account with with Admin role, and a provisioned Onehouse project.
- At least one Onehouse table that has Delta Lake metadata. Databricks requires Delta Lake table metadata to work with Onehouse tables. To meet this requirement, be sure you have created a OneTable catalog with Delta Lake enabled, and have attached it to the relevant Stream Capture.
- Databricks 'Account admin' role on your account in order to be able to create a new Databricks Workspace and Catalog.
- Amazon AWS account administrator privileges to be able to create a VPC and IAM roles.

## Create a Databricks workspace

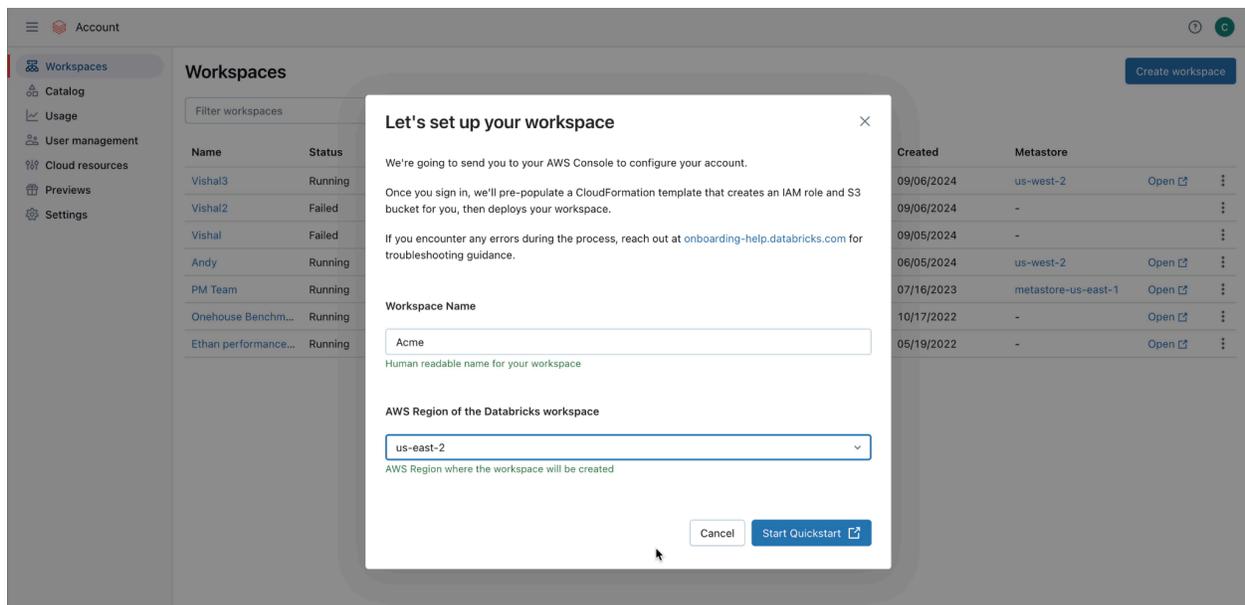
Refer to the Databricks documentation for workspace creation here:

<https://docs.databricks.com/en/admin/workspace/quick-start.html#create-a-workspace-using-the-aws-quick-start-recommended>

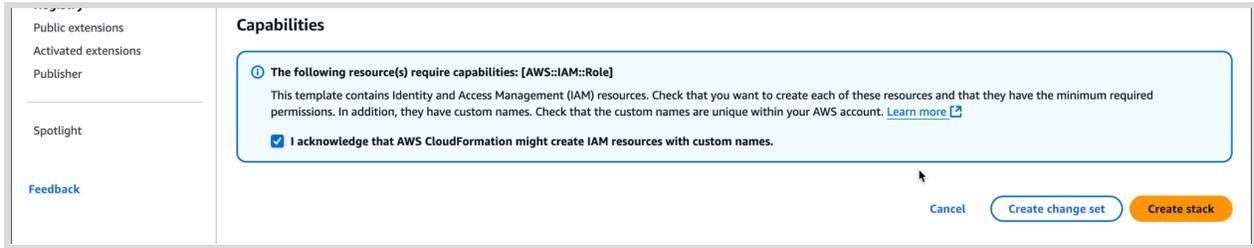
- If you don't already have a Databricks workspace or wish to start with a new one, create a new Databricks workspace using the AWS quickstart method.



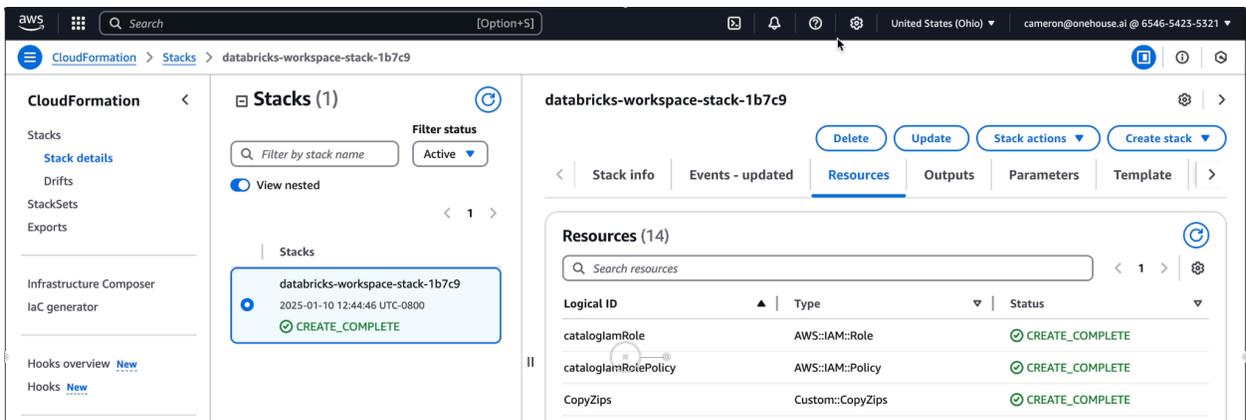
Note that if you intend to create a new Unity catalog, this will need to be done in an AWS region that has not already been used to host a Unity catalog.



In AWS, you'll need to acknowledge that roles will be created in your account. (The roles all start with 'databricks-workspace-stack-'.)



You'll see several CloudFormation events scroll by. When the stack shows 'COMPLETE', you can continue.



Go back to Databricks, and refresh the browser page to see the new workspace in the list.

## Create a Unity Catalog

You will also need to enable Unity Catalog metastore in the workspace.

Refer to the Databricks documentation 'Managing Unity Catalog' at:

<https://docs.databricks.com/en/data-governance/unity-catalog/get-started.html>

Also, refer 'Create a Unity Catalog' at:

<https://docs.databricks.com/en/data-governance/unity-catalog/create-metastore.html>

## Create an S3 Bucket for the metastore

- Name the S3 bucket something unique, but identifying it as a metastore.

The screenshot shows the AWS console interface for creating a new S3 bucket. The breadcrumb navigation indicates the path: Amazon S3 > Buckets > Create bucket. The main heading is 'Create bucket' with an 'Info' link. Below this, a sub-heading states 'Buckets are containers for data stored in S3.' The configuration is divided into two main sections: 'General configuration' and 'Object Ownership'. In the 'General configuration' section, the 'AWS Region' is set to 'US East (Ohio) us-east-2'. The 'Bucket type' is set to 'General purpose' (selected with a radio button), with a note that it is recommended for most use cases. The 'Directory' option is unselected. The 'Bucket name' is 'databricks-unity-catalog-acme', with a note that it must be unique within the global namespace. Below the name field is a 'Copy settings from existing bucket - optional' section with a 'Choose bucket' button. The 'Object Ownership' section has 'ACLs disabled (recommended)' selected, with a note that all objects are owned by this account. The 'ACLs enabled' option is unselected. At the bottom of the console, there is a footer with '© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

## Create an IAM role to access the storage location

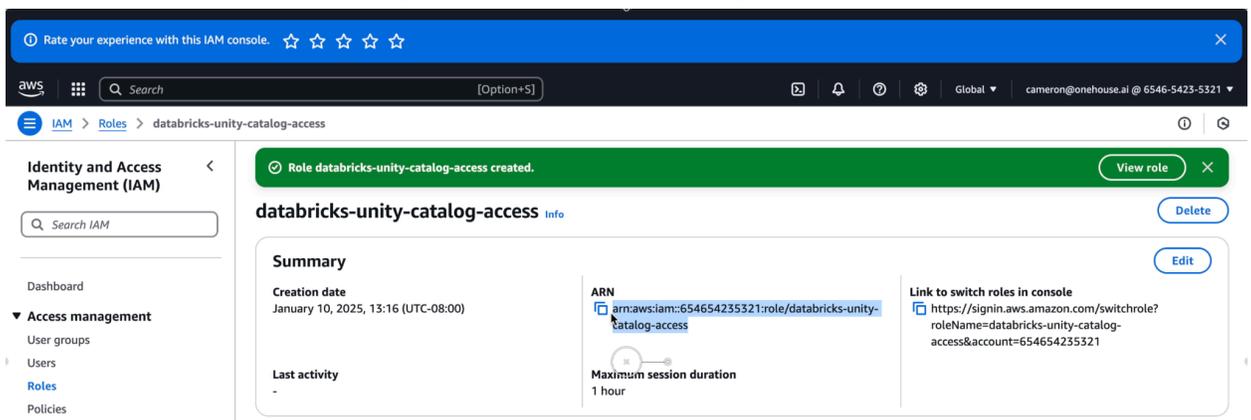
- In the Databricks account console, click on your username in the upper right corner, and copy the **Account ID** value.
- In AWS IAM, create a new role.
  - Type: Custom Trust policy
  - Paste in the following policy, substituting the Databricks Account #.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::414351767826:role/unity-catalog-prod-UCMasterRole-14S5ZJVKOTYTL"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<DATABRICKS_ACCT_ID>"
        }
      }
    }
  ]
}

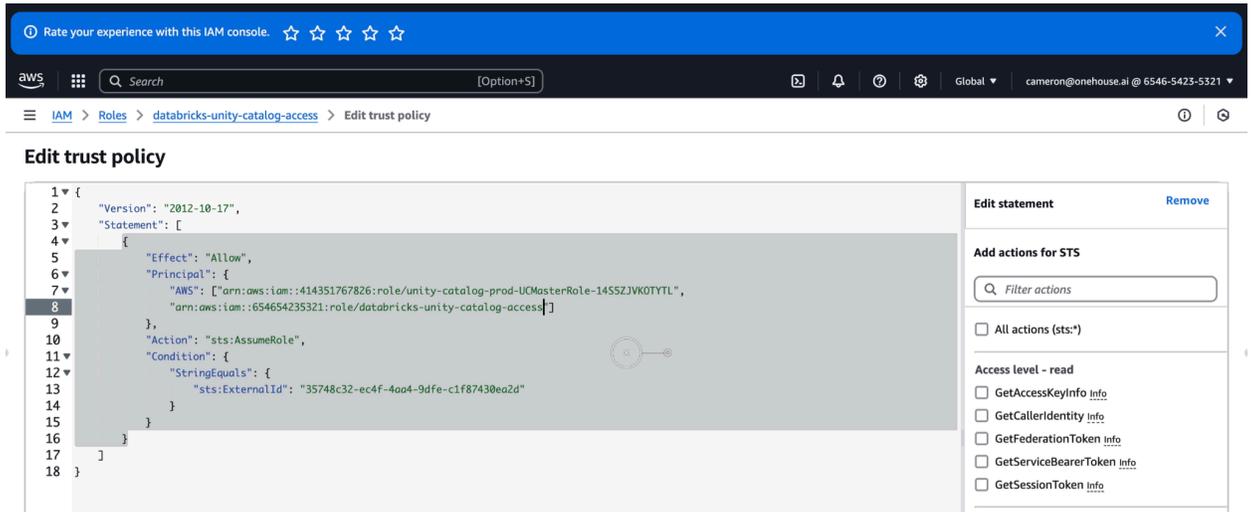
```

- Click 'Next', and 'Next' and name the role something similar to `databricks-unity-catalog-access`
- Click 'Create Role'
- Click into the new role, and copy the ARN for the role you just created:



- Scroll down and click the 'Trust relationships' tab, then click 'Edit trust policy'

- Add the ARN for the this role to the list of Principals. (This makes the role self-assumable.)

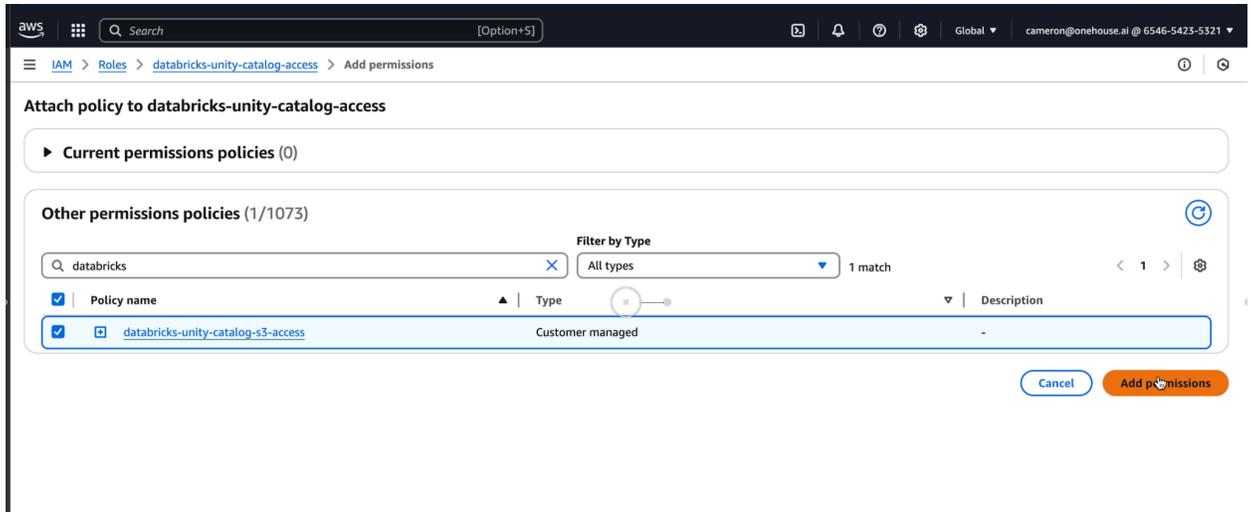


- Click 'Update Policy'
- Create a new IAM policy to allow access to the S3 metastore bucket created earlier. Be sure to replace the bucket name and also the assumable role. Name the policy something like: `databricks-unity-catalog-s3-access`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::databricks-unity-catalog-acme/*",
        "arn:aws:s3:::databricks-unity-catalog-acme"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::654654235321:role/databricks-unity-catalog-access"
      ],
    }
  ]
}
```

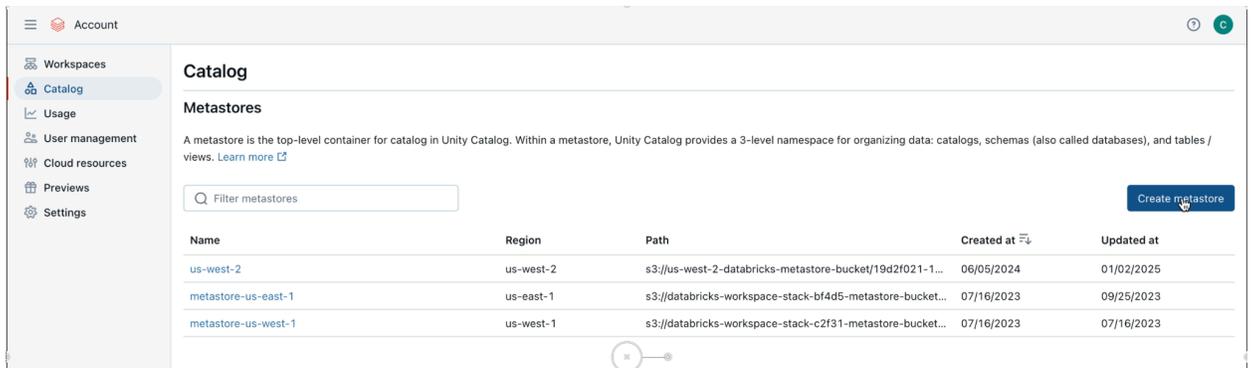
```
}
  ]
}
"Effect": "Allow"
```

- Navigate back to the `databricks-unity-catalog-access` role you created and scroll down to the Permissions tab, and under 'Permissions policies, click 'Add permissions'.
- Attach the `databricks-unity-catalog-s3-access` policy you created to the role.

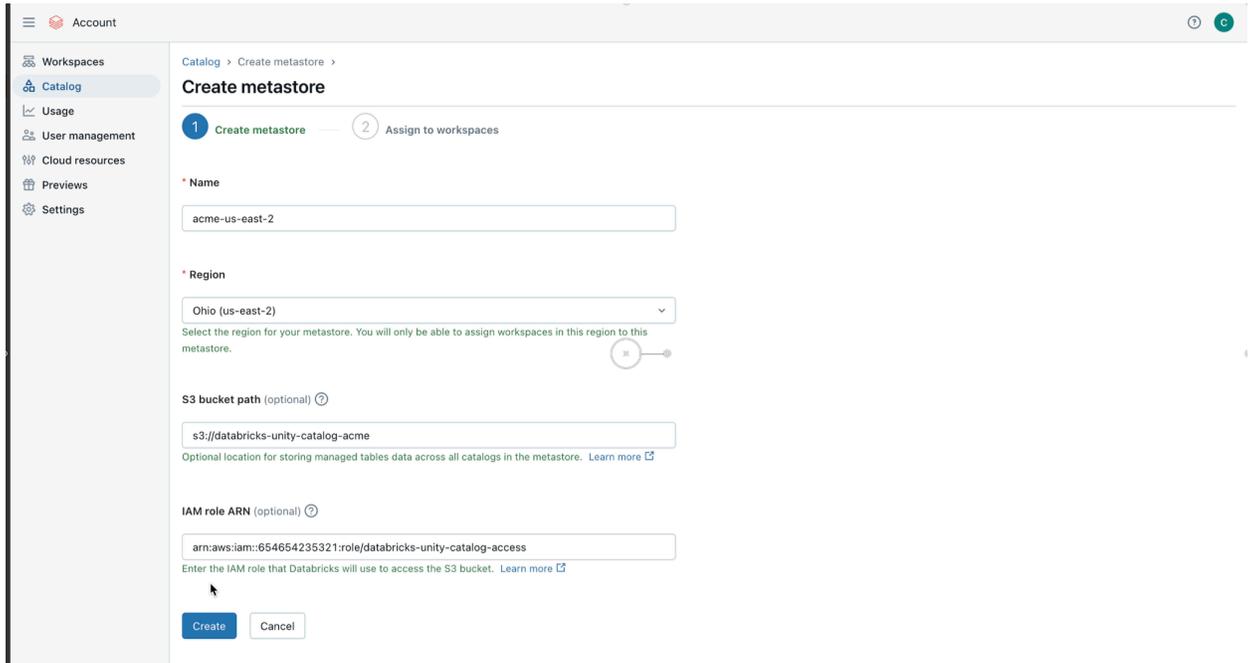


## Create a Unity Catalog metastore

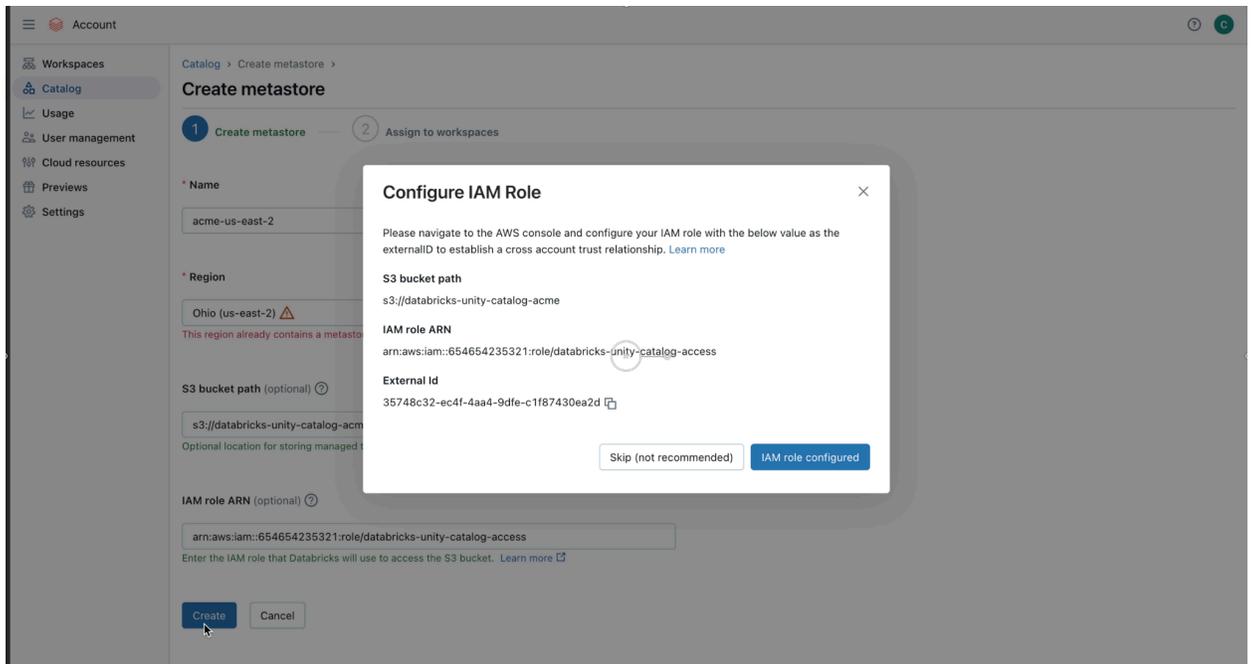
- In the Databricks Account console, click on 'Catalog' in the left navigation area, and then click 'Create metastore'



- ❑ Choose an AWS region that has not already been used to host a Unity catalog. Paste in the S3 metastore bucket URI. Paste in the IAM role ARN. Click 'Create'.



- ❑ When you are asked to 'Configure the IAM Role', note that you have already created the role, but double-check it to be sure that it meets the stated requirements.



Select the workspace to be associated with the new metadata store.

The screenshot shows the 'Create metastore' page in the 'Assign to workspaces' step. The left sidebar contains navigation options: Workspaces, Catalog, Usage, User management, Cloud resources, Previews, and Settings. The main content area has a breadcrumb 'Catalog > Create metastore >' and a progress indicator with 'Create metastore' and 'Assign to workspaces'. Below this is the heading 'Assign acme-us-east-2 to workspaces' and a search box for workspaces. A table lists available workspaces with columns for Name, Status, Pricing tier, Region, Created, and Metastore. The 'Acme' workspace is selected with a blue checkmark. At the bottom, it says '1 workspace selected' and has 'Assign' and 'Skip' buttons.

Name	Status	Pricing tier	Region	Created	Metastore
<input checked="" type="checkbox"/> Acme	Running	Enterprise	us-east-2	today at 12:45 PM	-
<input type="checkbox"/> Vishal3	Running	Enterprise	us-west-2 ⓘ	09/06/2024	us-west-2 ⓘ ⚠
<input type="checkbox"/> Vishal2	Failed	Enterprise	us-west-2 ⓘ	09/06/2024	-
<input type="checkbox"/> Vishal	Failed	Enterprise	us-west-2 ⓘ	09/05/2024	-
<input type="checkbox"/> Andy	Running	Enterprise	us-west-2 ⓘ	06/05/2024	us-west-2 ⓘ ⚠
<input type="checkbox"/> PM Team	Running	Enterprise	us-east-1 ⓘ	07/16/2023	metastore-us-east-1 ⓘ ⚠
<input type="checkbox"/> Onehouse Benchmarks	Running	Enterprise	us-east-2	10/17/2022	-
<input type="checkbox"/> Ethan performance benchmar...	Running	Enterprise	us-west-1 ⓘ	05/19/2022	-

The screenshot shows the same 'Create metastore' page, but with a modal dialog box titled 'Enable Unity Catalog?' overlaid. The dialog contains the following text:

Assigning the metastore will update workspaces to use Unity Catalog, meaning that:

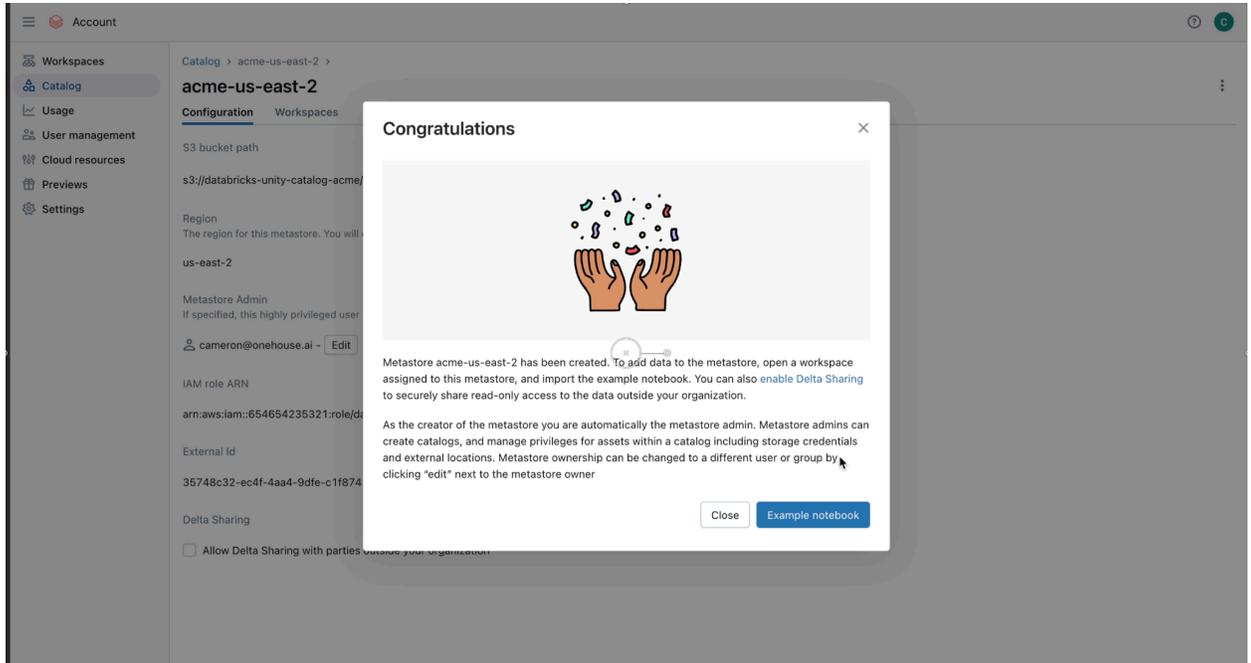
- ✓ Data can be governed and accessed across workspaces
- ✓ Data access and lineage is captured automatically
- ✓ Identities are managed centrally at the account level (cannot be reversed)

Before enabling Unity Catalog, consider these readiness checks:

- ✓ Understand the privileges of workspace admins in Unity Catalog and review existing workspace admin designations
- ✓ Update any automation for principal/group management, such as SCIM, Okta and Microsoft Entra connectors, and Terraform to reference account endpoints instead of workspace endpoints

[Learn more](#)

Buttons: Back, Enable



## Configure the Unity Catalog for CORS

- Refer to the Databricks documentation to configure the metastore storage for CORS here:

<https://docs.databricks.com/en/data-governance/unity-catalog/storage-cors.html>

## Connect the Workspace to the Onehouse data storage location

To sync your Onehouse tables with Databricks Unity Catalog, you must grant Databricks access to the storage location where your Onehouse tables are stored. This involves creating a Databricks Storage Credential and External Location.

Refer to the Databricks documentation: Create a storage credential for connecting to AWS S3 at: <https://docs.databricks.com/en/connect/unity-catalog/cloud-storage/storage-credentials.html>

### Create and attach IAM policy for Onehouse customer storage access

- Create a new AWS IAM Policy. This will be to allow access to the Onehouse customer (data lake) storage. Replace the bucket name with your Onehouse S3 customer bucket location, and the 'Resource' with the ARN for the role you created earlier.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::BUCKET_NAME",
        "arn:aws:s3:::BUCKET_NAME/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "<DATABRICKS_ACCESS_ROLE_ARN"
    }
  ]
}
```

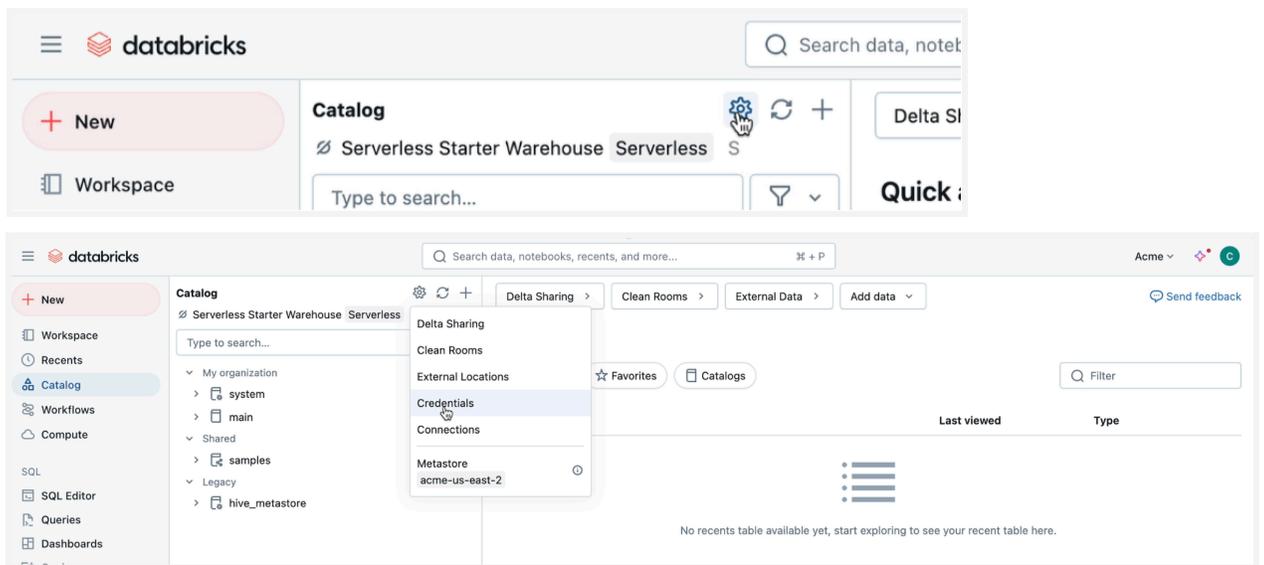
- Click 'Next'. Name the new policy something like: `databricks-onehouse-S3-access-ajax`
- Click 'Create Policy'.

Note: We will reuse the AWS Role created earlier for Unity Catalog access, and simply attach the new policy for Onehouse S3 customer bucket access.

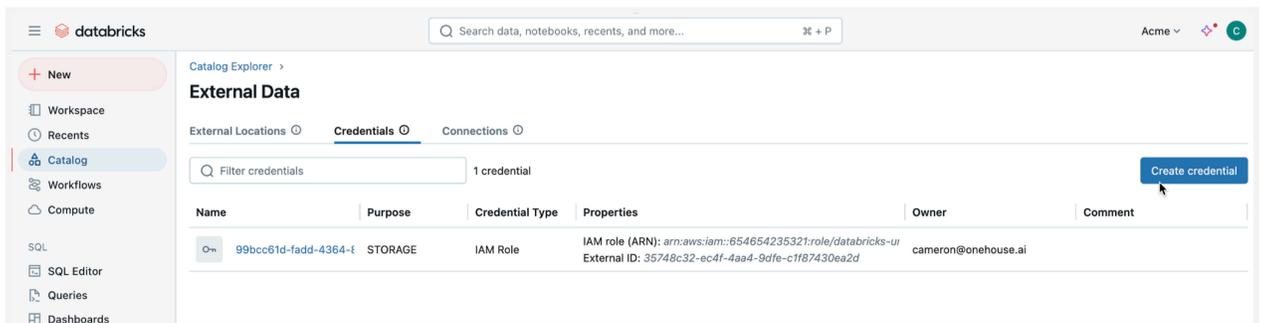
- Back in the `databricks-unity-catalog-access` role, click 'Add permissions' and 'Attach policies'.
- Select the `databricks-onehouse-S3-access-ajax` policy that was just created and click 'Add permissions'.

## Create a Databricks Storage Credential

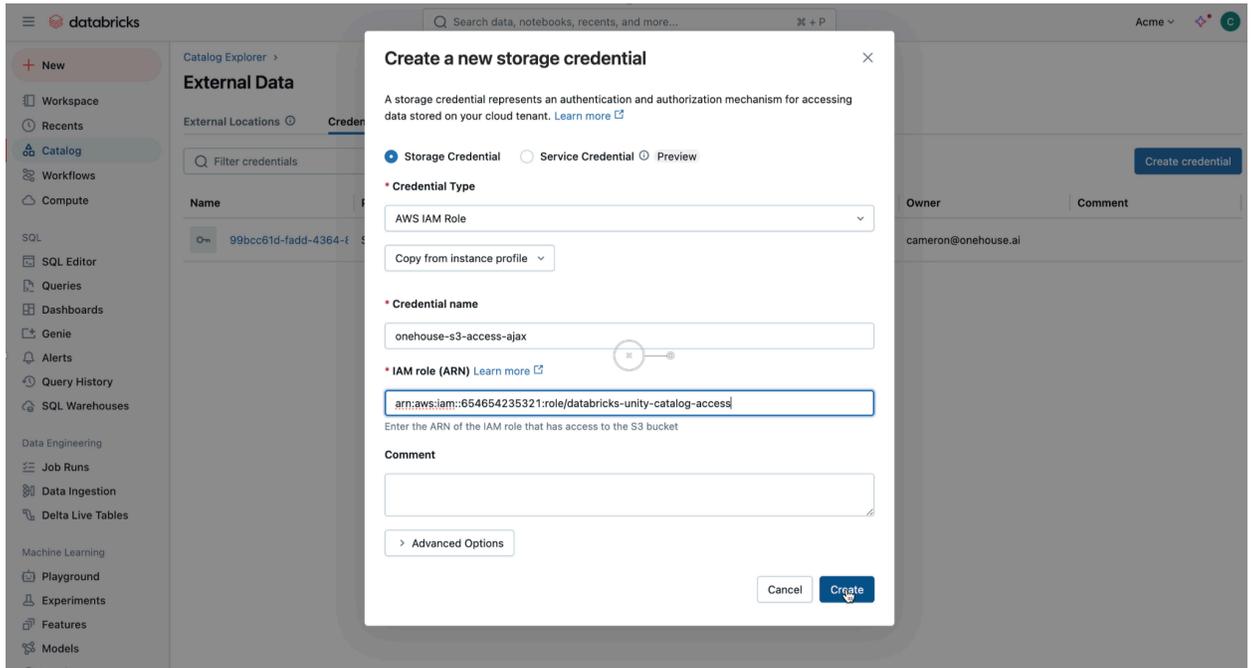
- In the Databricks workspace console, click 'Catalog', then the settings gear icon, then 'Credentials'.



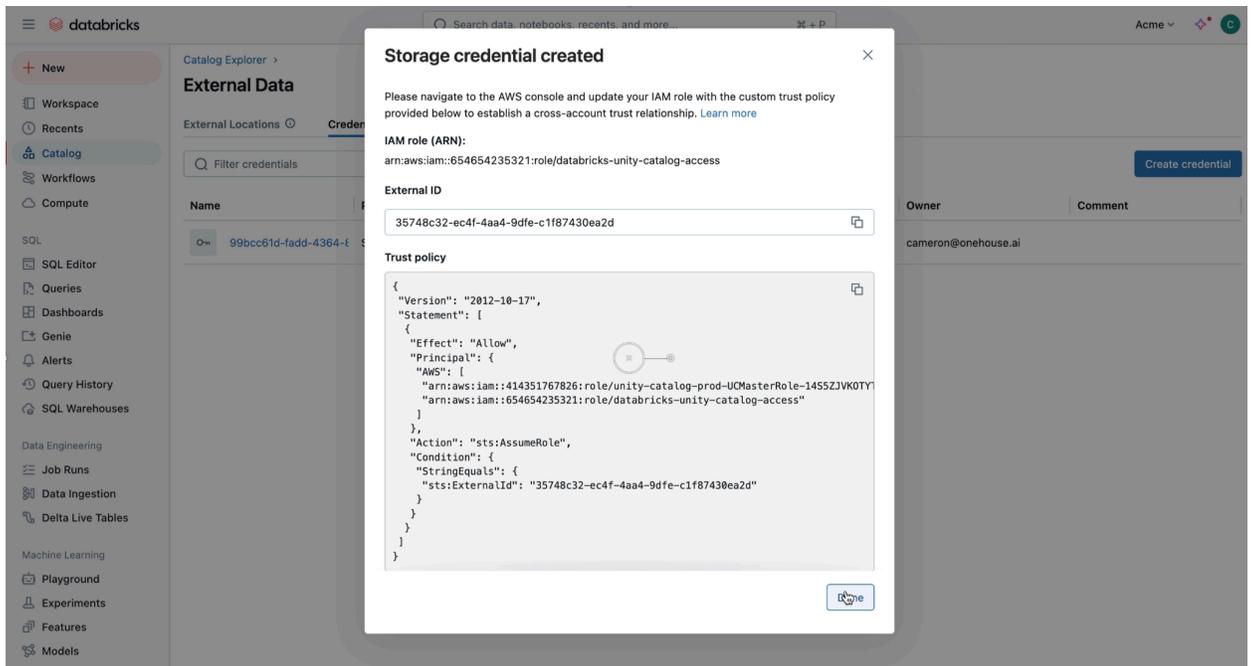
- Click 'Create credential'



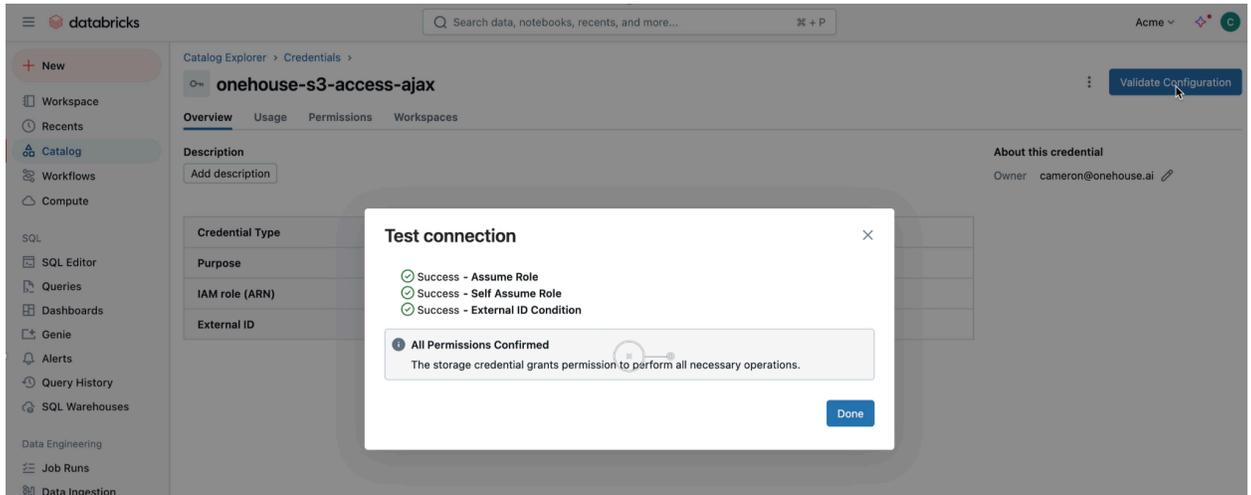
- For the storage credential name, choose something like `onehouse-s3-access`. Paste in the AWS ARN of the role you created earlier. Click 'Create'.



- You can confirm that the role trust policy matches what is offered in the next dialog, but you should have already set this up.



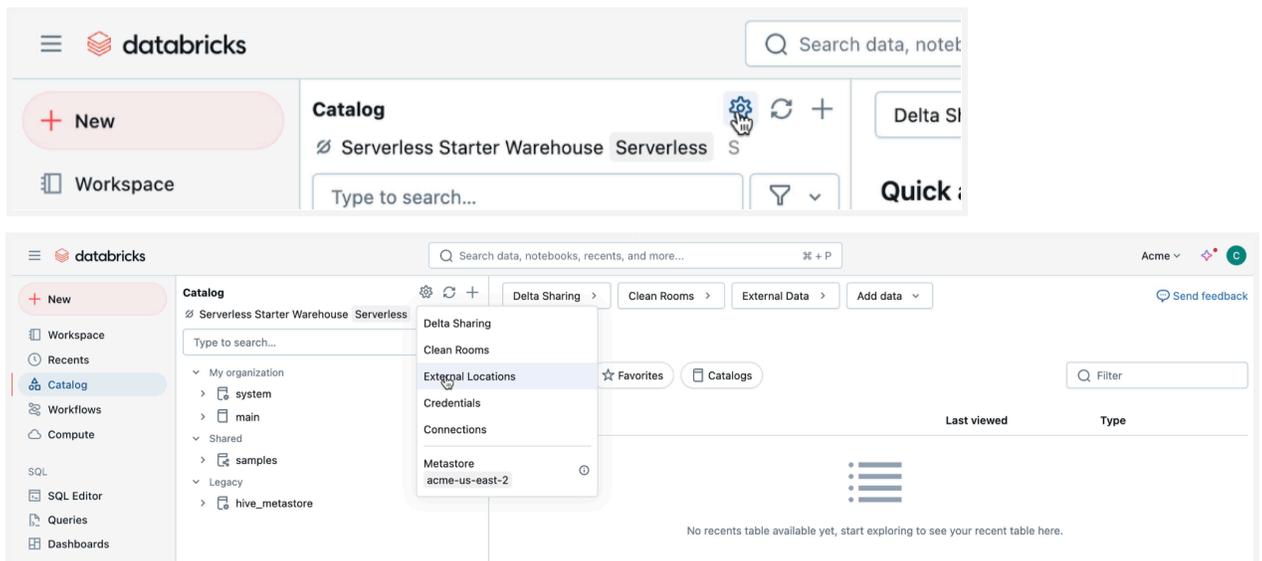
- ❑ Click 'Validate Configuration' and ensure that all the permissions are confirmed good.



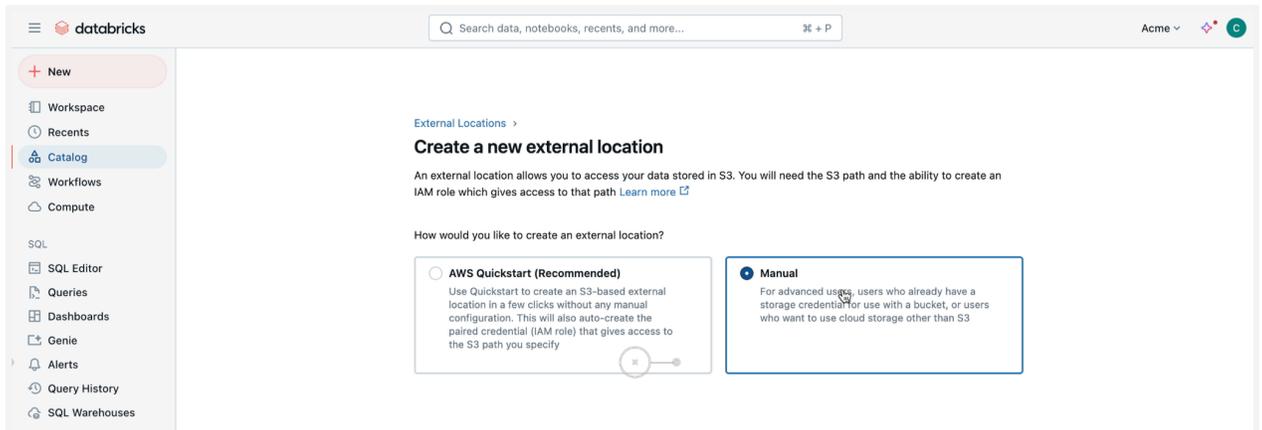
## Create a Databricks External Location

Since we are referencing an existing Onehouse S3 bucket, refer to the Databricks documentation: 'Creating an external location manually using Catalog Explorer' at: <https://docs.databricks.com/en/connect/unity-catalog/cloud-storage/external-locations.html#create-an-external-location-manually-using-catalog-explorer>

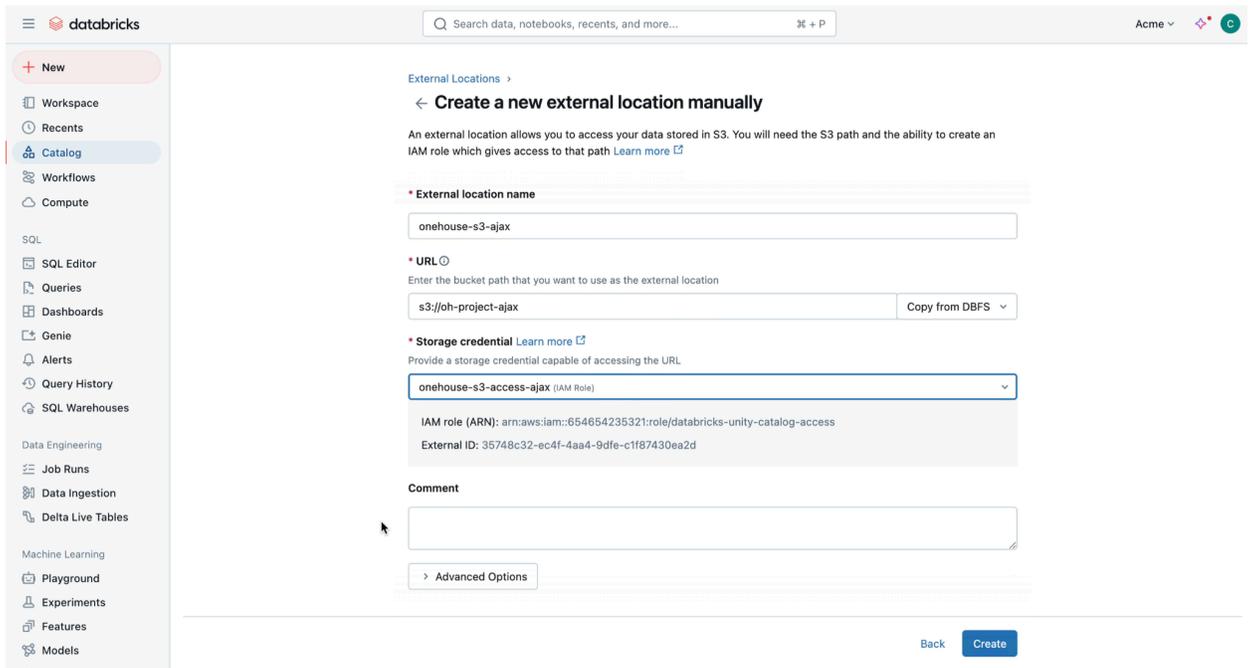
- ❑ In the Databricks workspace console, click 'Catalog', then the settings gear icon, then 'External Locations'.



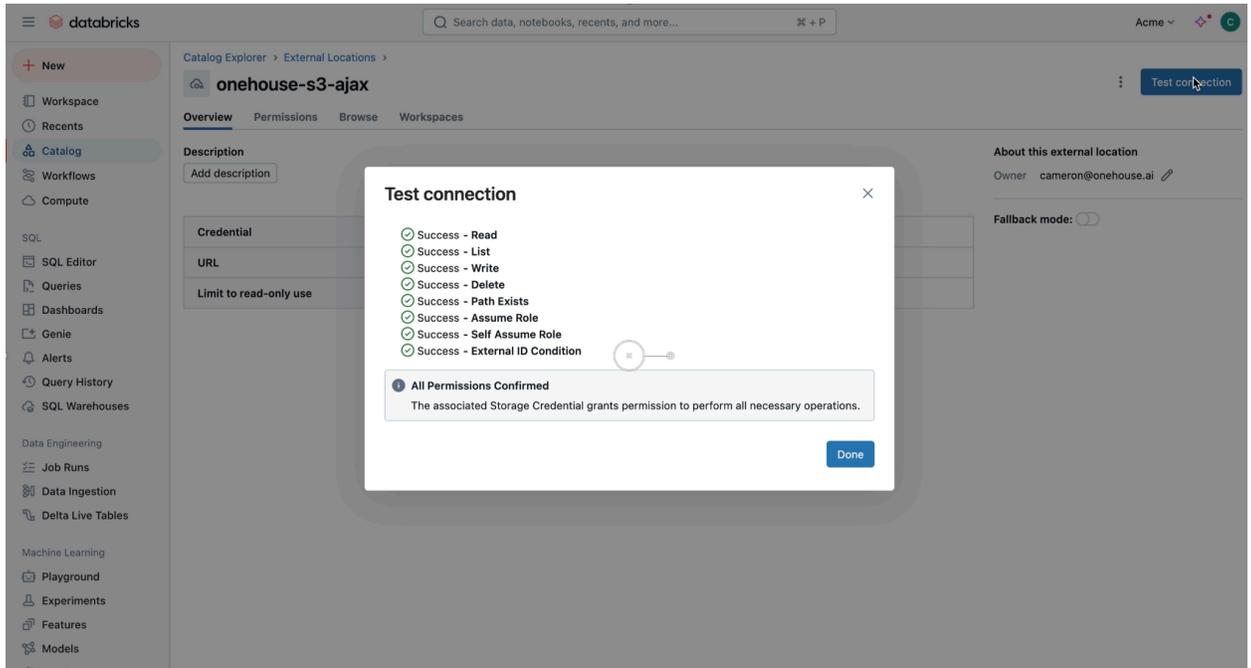
- Choose 'Manual', then click 'Next'.



- Provide a name such as onehouse-s3-projectname. Paste in the S3 URL for the existing Onehouse customer data bucket. Choose the storage credential that we created earlier. Click 'Create'.



- Click 'Test connection' and ensure that all permissions are confirmed good.



## Gather the external catalog information in Databricks

There are two ways that Onehouse can authenticate to Databricks to synchronize the metadata catalog: OAuth, and Access Token.

- For the Access Token Auth Type, refer to the Databricks documentation for personal access token generation at: <https://docs.databricks.com/en/integrations/jdbc/authentication.html#databricks-personal-access-token>
- For the OAuth access type, refer to the Databricks documentation for service principal creation and retrieving credentials at: <https://docs.databricks.com/en/integrations/jdbc/authentication.html#oauth-machine-to-machine-m2m-authentication>
- In the Databricks workspace console, select 'SQL Warehouses', and then select the 'Serverless Starter Warehouse' that should have been created with the new workspace, or select the SQL Warehouse that you plan to use for catalog synchronization.

□ Make note of the Server hostname and HTTP path.

The screenshot shows the Databricks interface for a 'Serverless Starter Warehouse'. The left sidebar contains navigation options like 'Workspace', 'Recents', 'Catalog', 'Workflows', 'Compute', 'SQL Editor', 'Queries', 'Dashboards', 'Genie', 'Alerts', 'Query History', 'SQL Warehouses', 'Data Engineering', 'Job Runs', 'Data Ingestion', 'Delta Live Tables', 'Machine Learning', 'Playground', 'Experiments', 'Features', and 'Models'. The main content area is titled 'Serverless Starter Warehouse' and includes tabs for 'Overview', 'Connecting details', and 'Monitoring'. Below the tabs, there are instructions to 'Use these details to connect to this warehouse' and a 'Create a personal access token' button. A row of icons represents various tools: Tableau, Power BI, dbt, Python, Java, Node.js, Go, and More tools. The configuration details are as follows:

- Server hostname:** dbc-6a470af3-aced.cloud.databricks.com
- HTTP path:** /sql/1.0/warehouses/8887c0454516e2c5
- JDBC URL:** 2.6.25 or later
- JDBC URL (expanded):** jdbc:databricks://dbc-6a470af3-aced.cloud.databricks.com:443/default;transportMode=http;ssl=1;AuthMech=3;httpPath=/sql/1.0/warehouses/8887c0454516e2c5;
- OAuth URL:** https://dbc-6a470af3-aced.cloud.databricks.cr

Additional text includes: 'Databricks supports drivers released within the last two years. Download drivers here'.

## Create the Metadata Catalog Sync in Onehouse

Finally! We are ready to create the metadata sync in Onehouse and attach it to a data lakehouse table.

### Create the Databricks metadata catalog in Onehouse

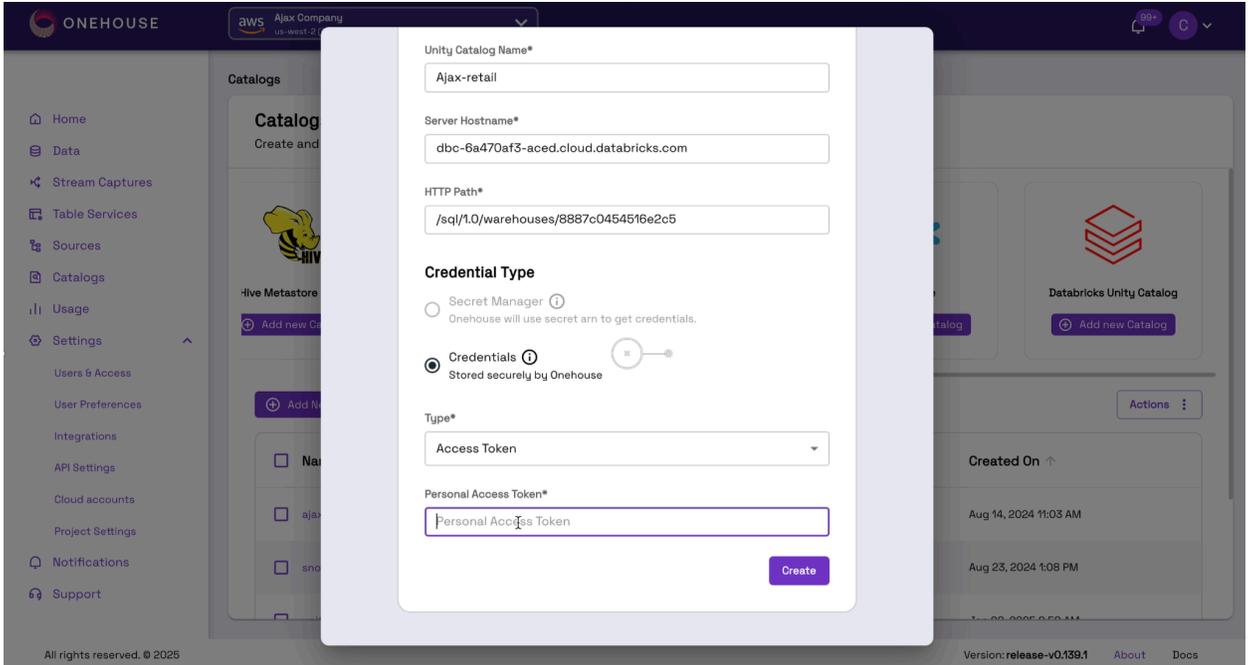
- In Onehouse, click 'Catalogs' in the left navigation panel.
- Scroll over to the Databricks Unity Catalog icon and choose 'Add new Catalog'.
- Fill in the new catalog form:
  - Enter a Onehouse catalog name such as databricks-catalog
  - The Unity Catalog Name is the location in the Unity catalog where the tables will be synced. If the catalog location doesn't already exist, it will be created.
  - Paste in the Server Hostname from Databricks.
  - Paste in the server HTTP Path from Databricks.

The screenshot shows the 'New catalog' form in the Onehouse interface. The form is titled 'New catalog' and includes the following fields and options:

- Name\***: databricks-catalog
- Type\***: Databricks Unity Catalog
- Unity Catalog Name\***: Ajax-retail
- Server Hostname\***: dbc-6a470af3-aced.cloud.databricks.com
- HTTP Path\***: /sql/1.0/warehouses/8887c0454516e2c5
- Credential Type**:
  - Secret Manager (Onehouse will use secret arn to get credentials.)
  - Credentials (Stored securely by Onehouse)

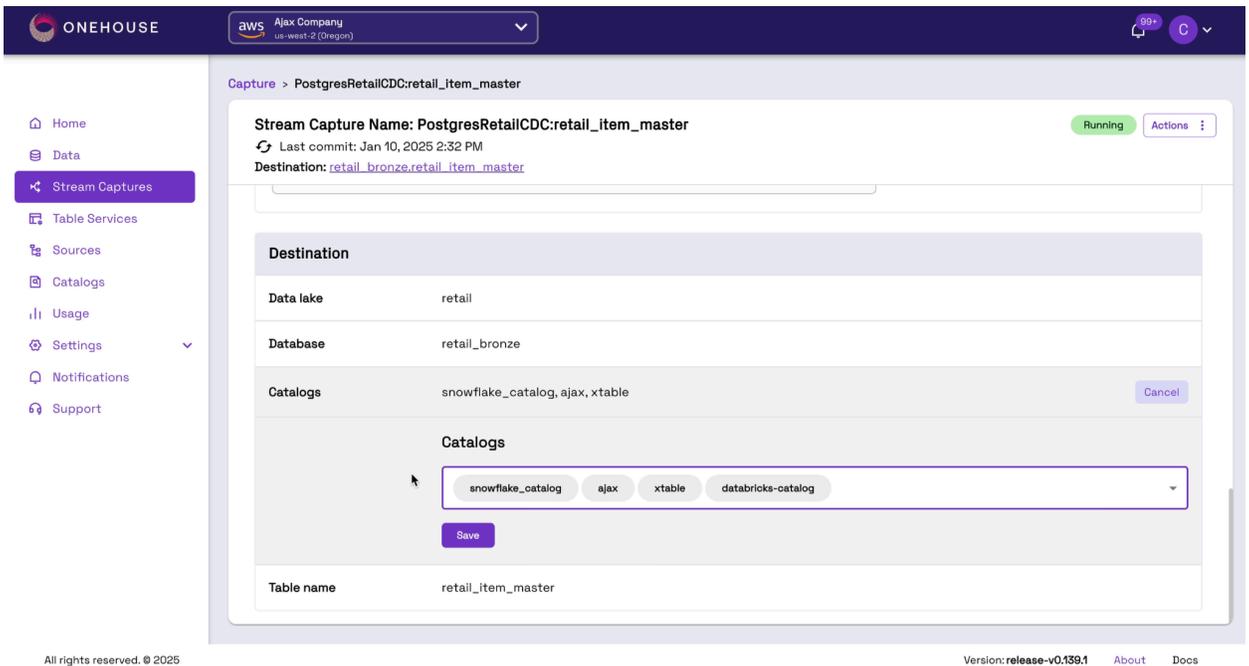
The background shows the Onehouse interface with the 'Catalogs' section visible. The 'Databricks Unity Catalog' icon is highlighted, and the 'Add new Catalog' button is visible. The 'Created On' section shows two dates: Aug 14, 2024 11:03 AM and Aug 23, 2024 1:08 PM.

- Choose the Credential type.
  - For OAuth type, enter the service principal credentials 'client-id' and 'client-secret' from your workspace.
  - For Access Token type, enter the 'personal-access-token' for your workspace user.



Attach the Onehouse Databricks catalog to one or more Stream Captures.

- Edit or create a stream capture that uses the Databricks catalog sync.



- After a time, the table metadata will be synchronized to the Databricks Unity Catalog. It can then be referenced in queries.

The screenshot shows the Databricks SQL Editor interface. On the left is a navigation sidebar with options like 'New', 'Workspace', 'Recents', 'Catalog', 'Workflows', 'Compute', 'SQL', 'SQL Editor', 'Queries', 'Dashboards', 'Genie', 'Alerts', 'Query History', 'SQL Warehouses', 'Data Engineering', and 'Job Runs'. The main 'Catalog' pane shows a tree view of the database structure. Under 'My organization' > 'acme' > 'default' > 'information\_schema', the 'retail\_bronze' schema is expanded, and its tables ('retail\_item\_categories', 'retail\_item\_master', 'retail\_scans', 'retail\_stores') are highlighted with a red box. The SQL Editor contains the following query:

```
1 SELECT category_code, item_id, item_price, item_upc
2 from acme.retail_bronze.retail_item_master
3 LIMIT 10
```

The 'Run selected (1000)' button is visible above the query. Below the query, the 'Raw results' section shows a table with 5 columns: 'category\_code', 'item\_id', 'item\_price', and 'item\_upc'. The first two rows of data are displayed:

	category_code	item_id	item_price	item_upc
1	20100	152967	1	2855988
2	20100	152957	1.06	2134552

###