ONEHOUSE

Setup Guide

Onehouse Catalog Sync for Databricks Unity Catalog

Jan 2025

Contents

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

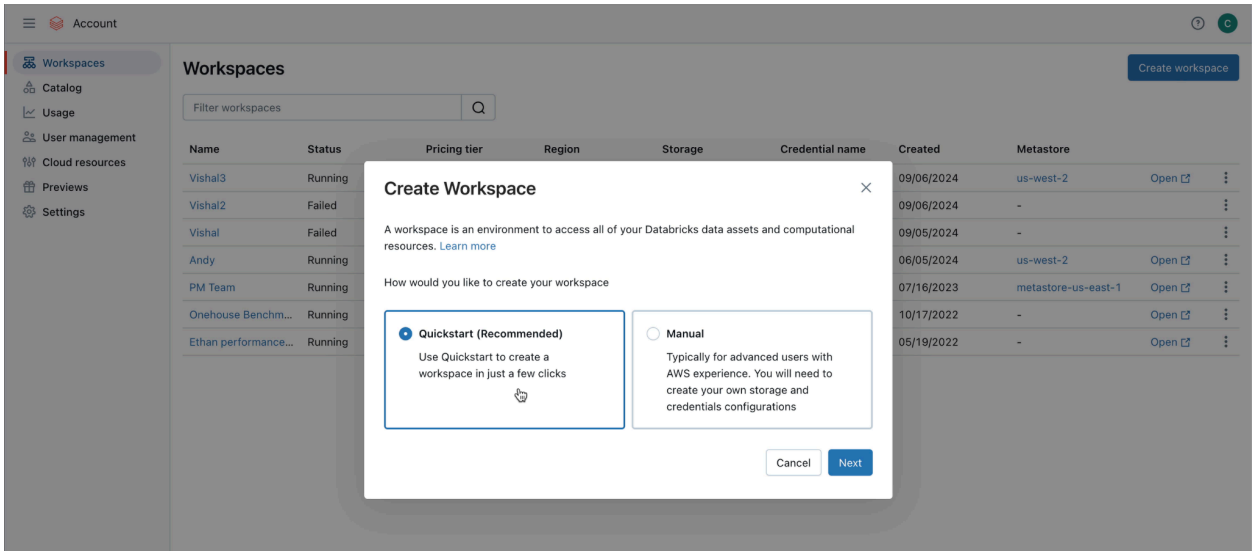
You will need the following:

- ☐ 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.

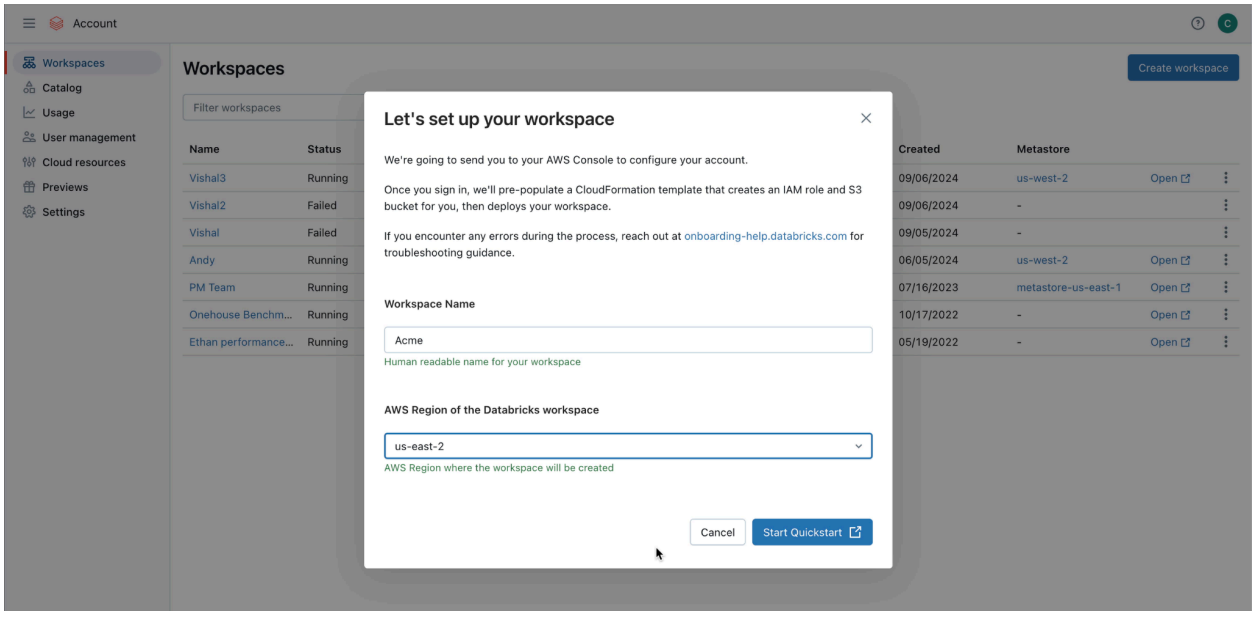
Create a Databricks workspace

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

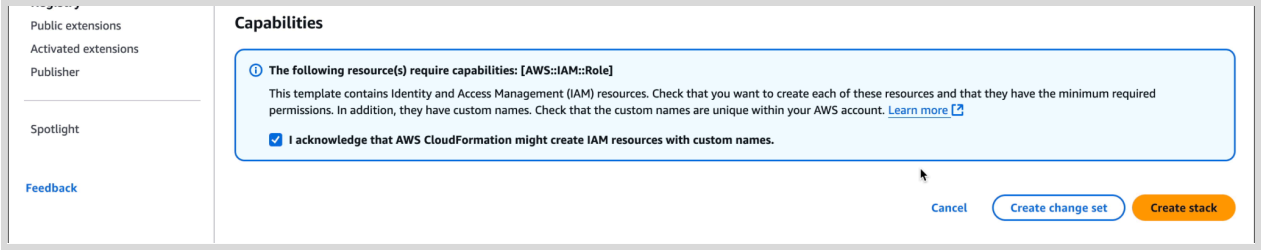
- [ ] If you don't already have a Databricks workspace or wish to start with a new one, create a new Databricks workspace using the AWS quickstart method.



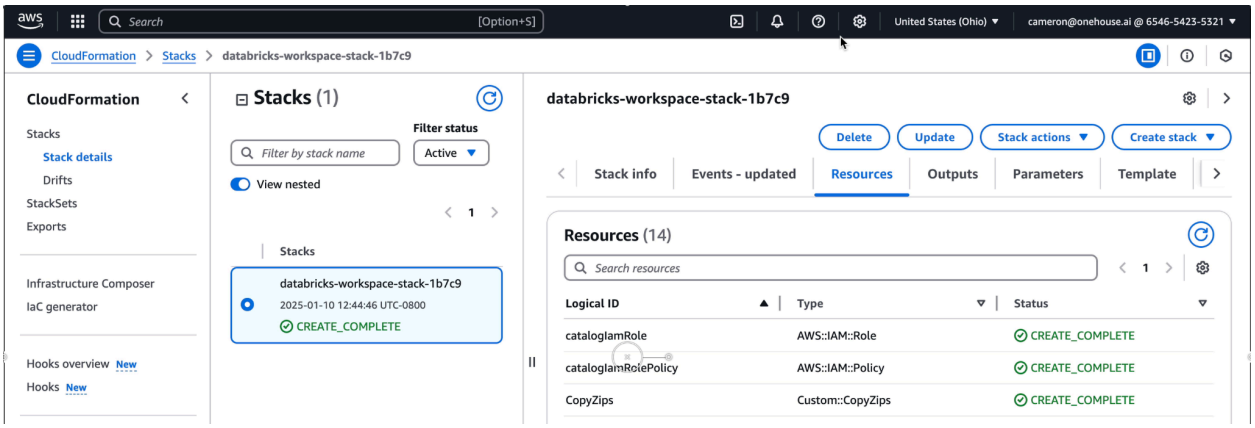
Note that if you intend to create a new Unity catalog, this will need to be done in an AWS region that has not already been used to host a Unity catalog.



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

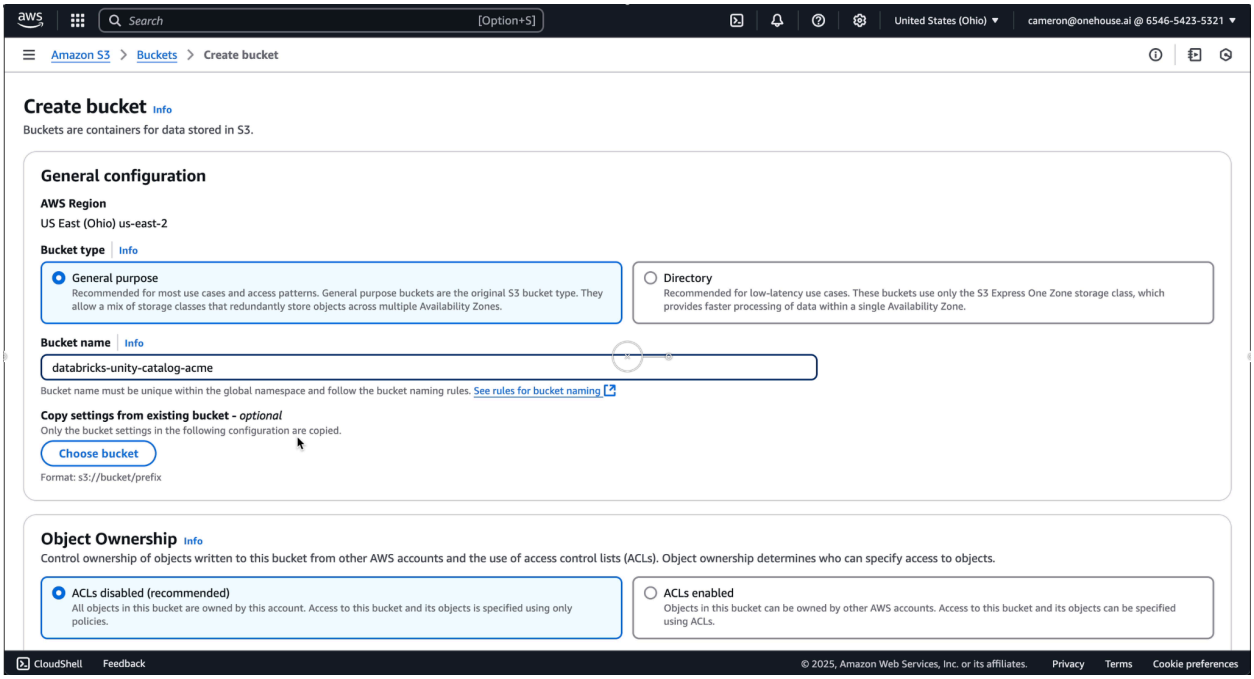
You will also need to enable Unity Catalog metastore in the workspace.

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.

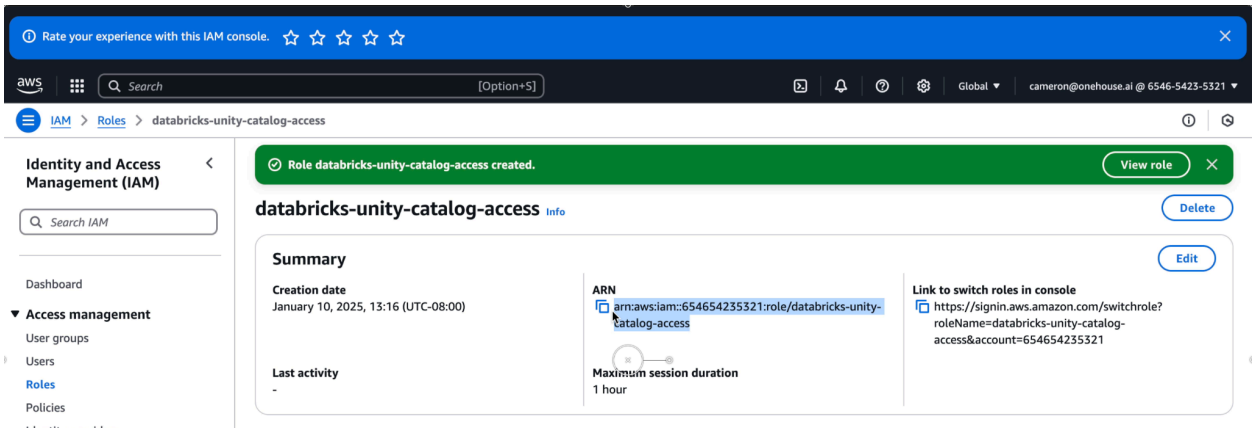


Create an IAM role to access the storage location

- [ ] 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 #.

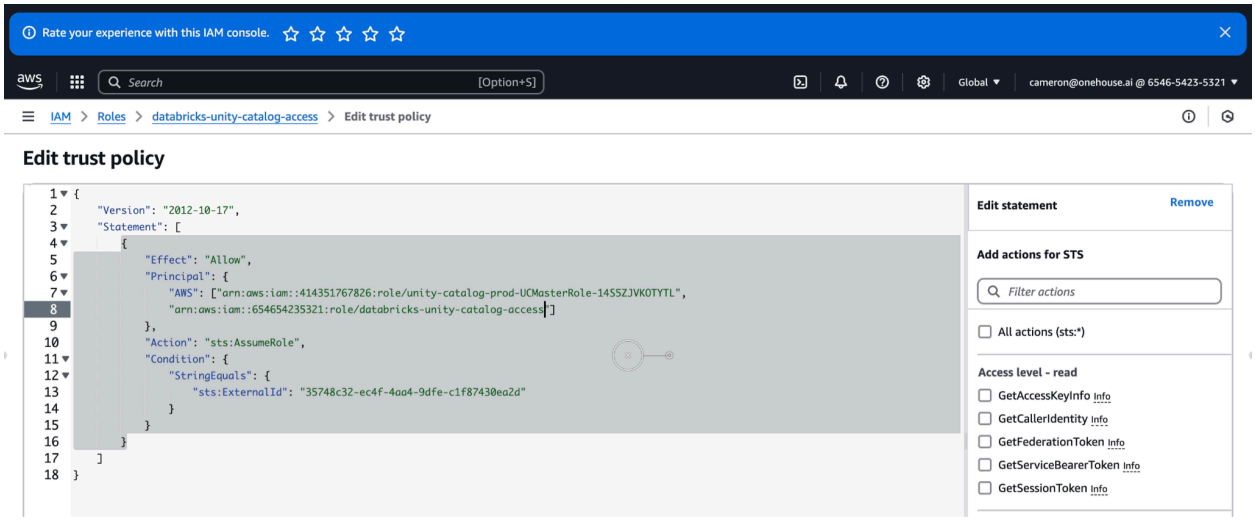
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
"arn:aws:iam::414351767826:role/unity-catalog-prod-UCMasterRole-14S5ZJ
VKOTYTL"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "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.)



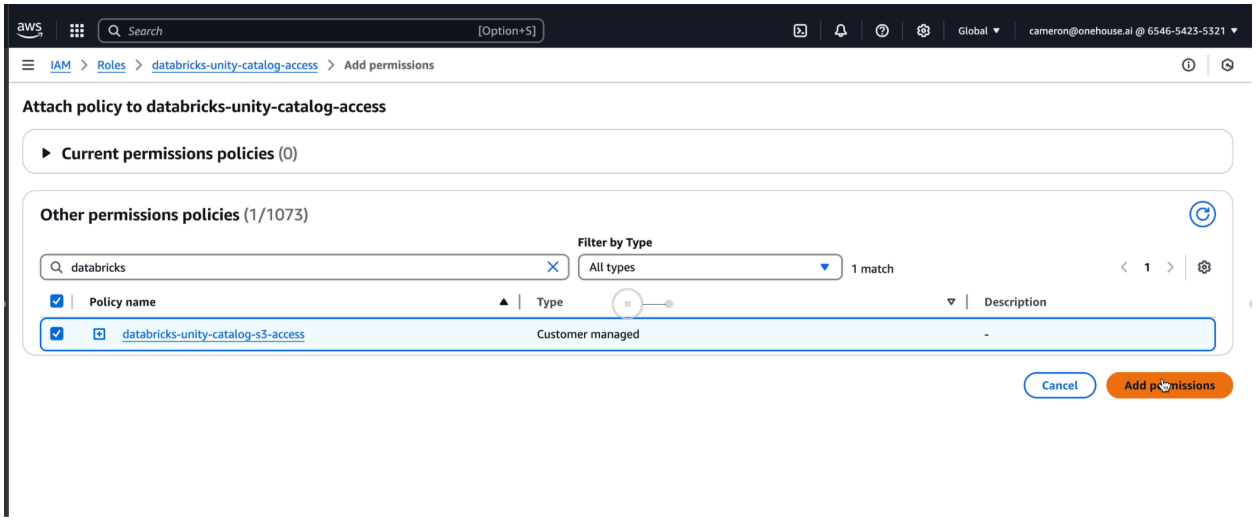
- [ ] Click 'Update Policy'
- [ ] Create a new IAM policy to allow access to the S3 metastore bucket created earlier. Be sure to replace the bucket name and also the assumable role. Name the policy something like: `databricks-unity-catalog-s3-access`

```
{
    "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"
            ],
```

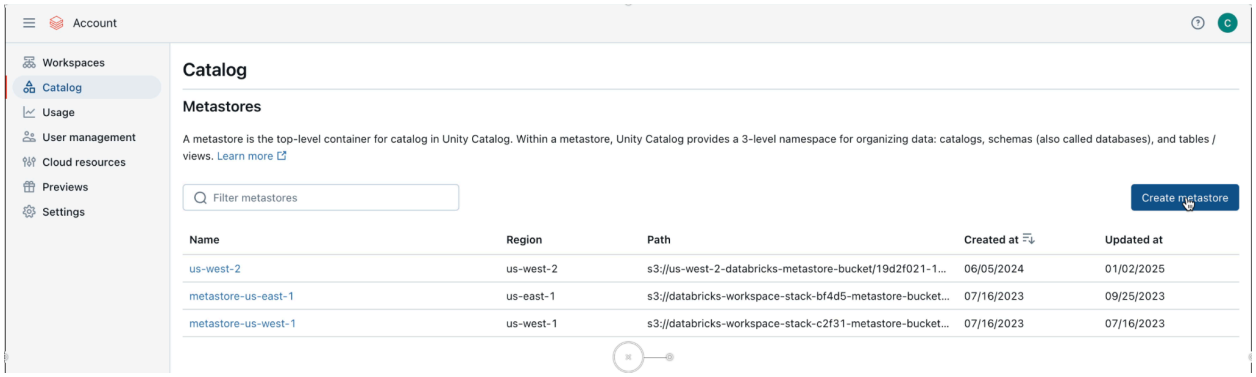
```
                "Effect": "Allow"
            }
        ]
}
```

- [ ] 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.

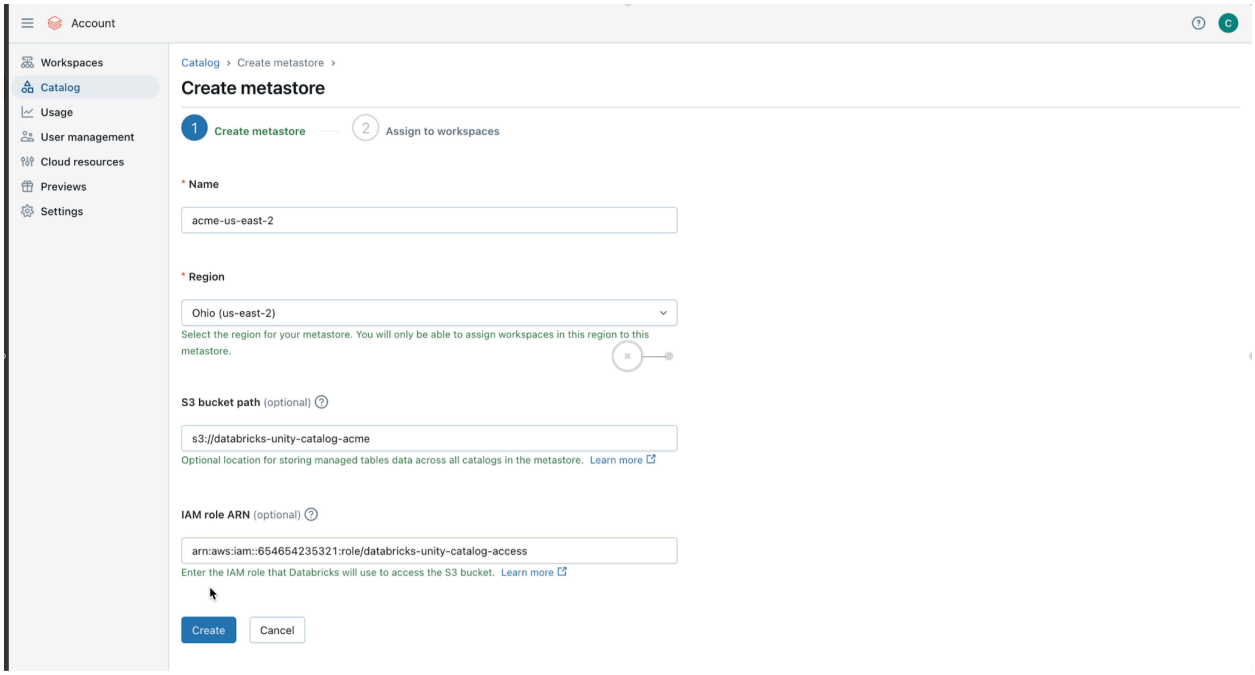


Create a Unity Catalog metastore

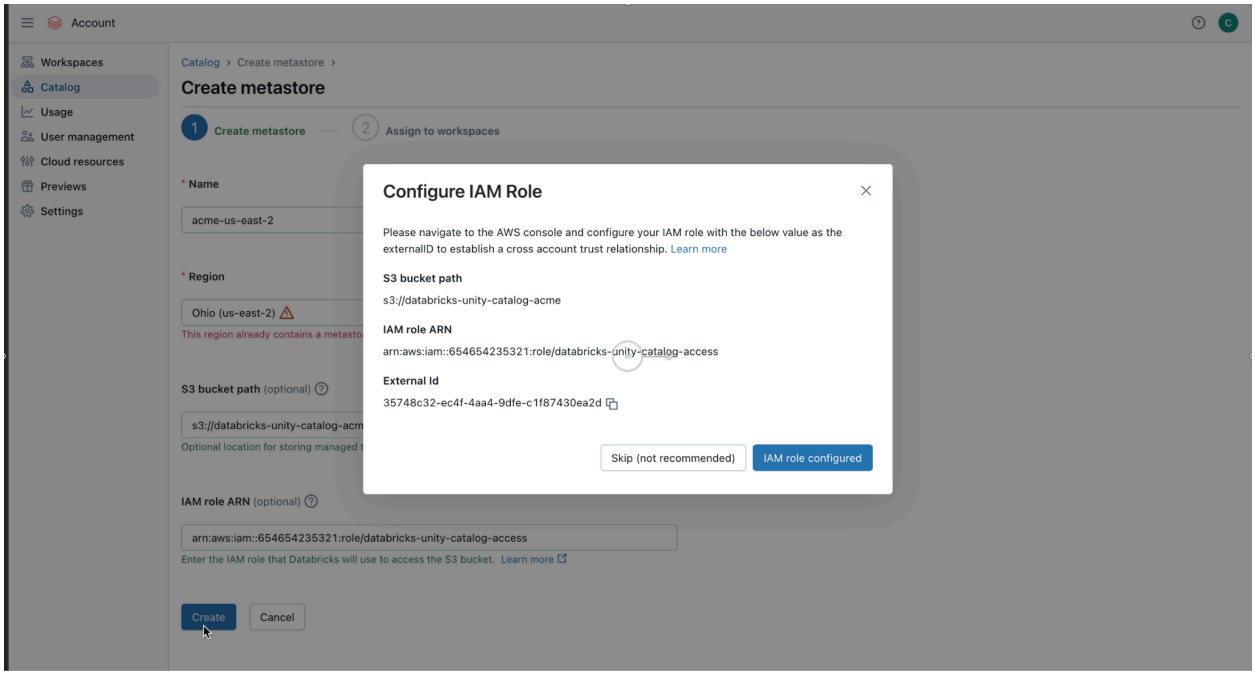
- [ ] In the Databricks Account console, click on 'Catalog' in the left navigation area, and then click 'Create metastore'



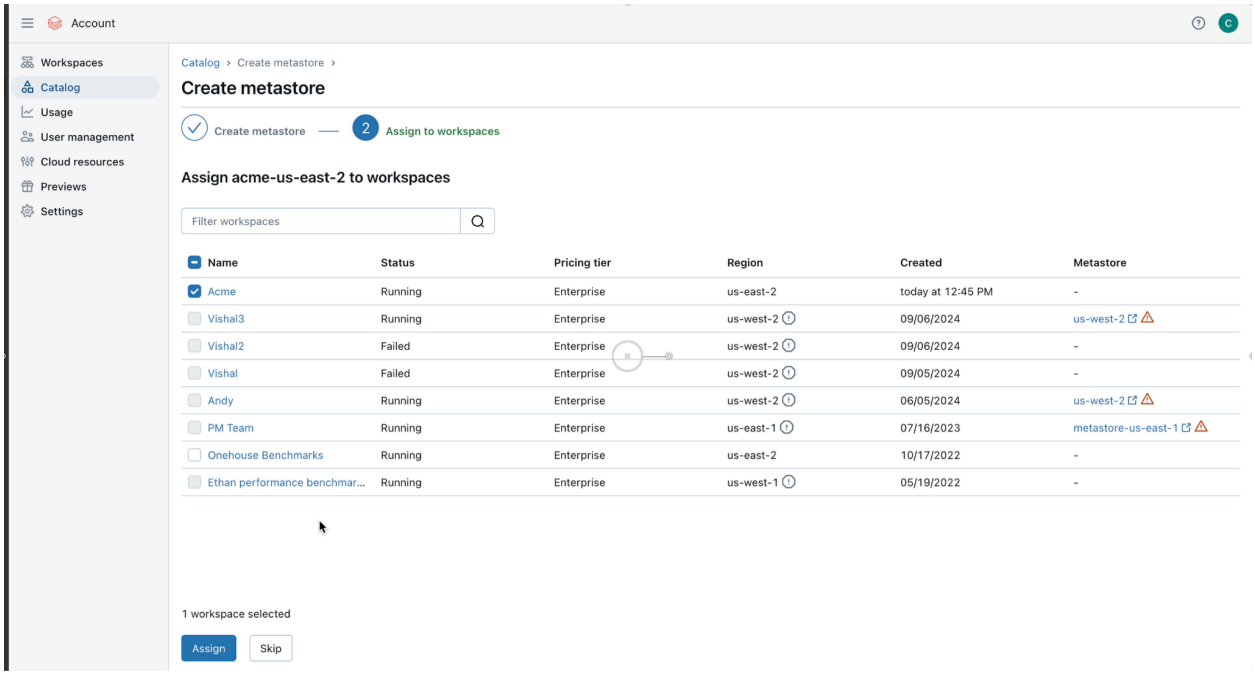
- [ ] Choose an AWS region that has not already been used to host a Unity catalog. Paste in the S3 metastore bucket URI. Paste in the IAM role ARN. Click ‘Create’.

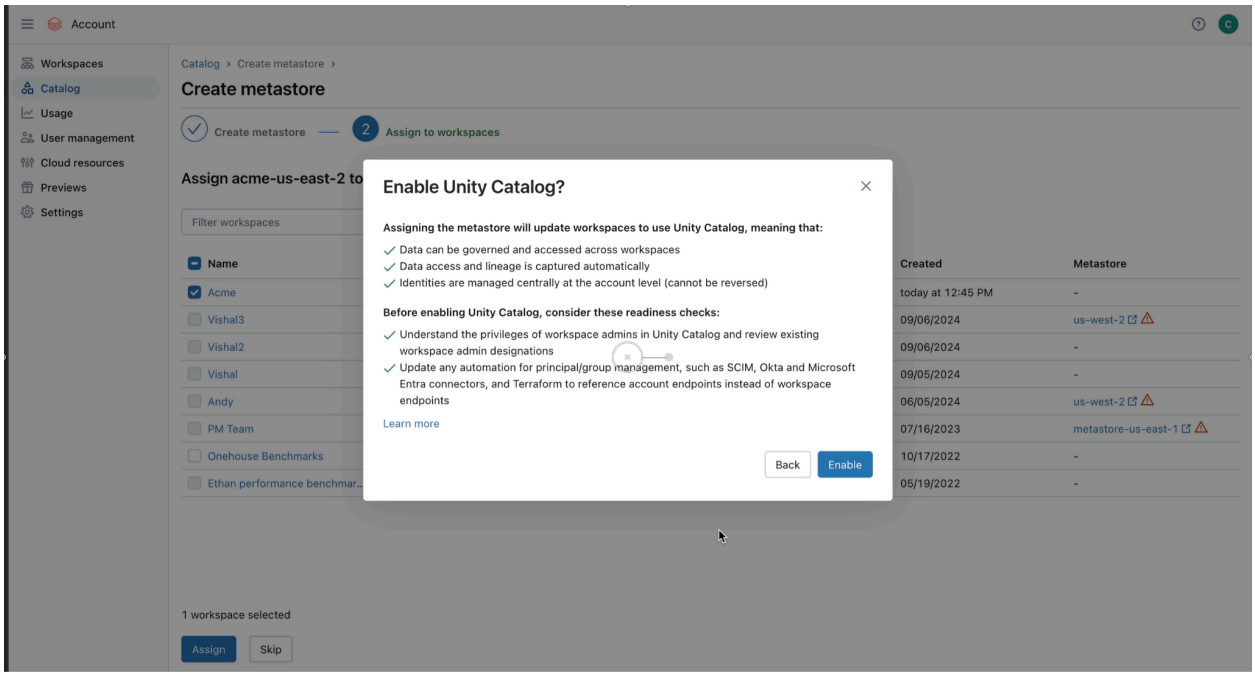


- [ ] When you are asked to ‘Configure the IAM Role’, note that you have already created the role, but double-check it to be sure that it meets the stated requirements.

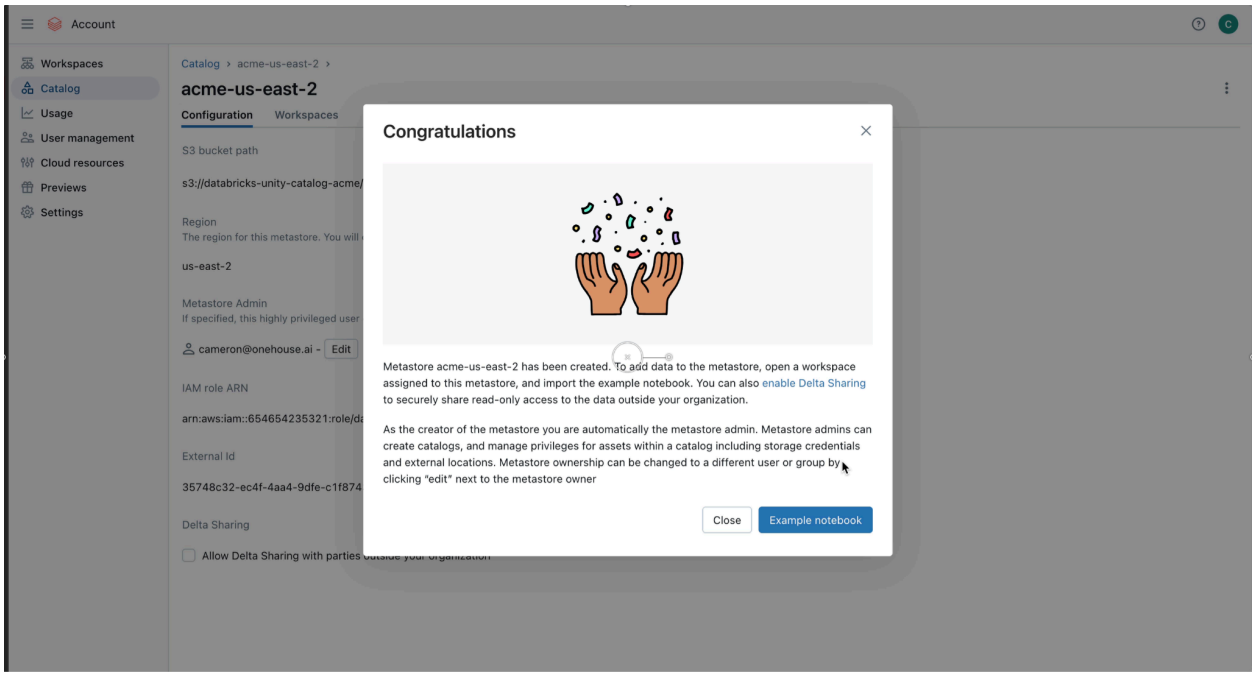


- [ ] Select the workspace to be associated with the new metadata store.





- [ ]



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`

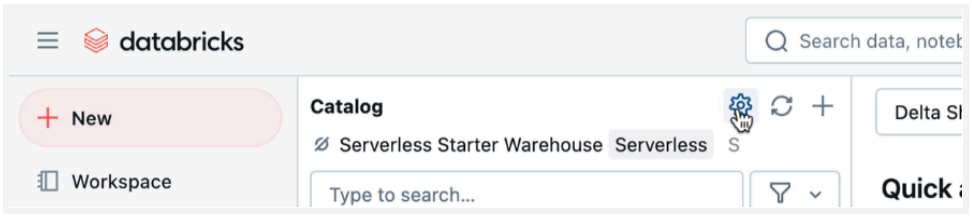
- [ ] Click 'Create Policy'.

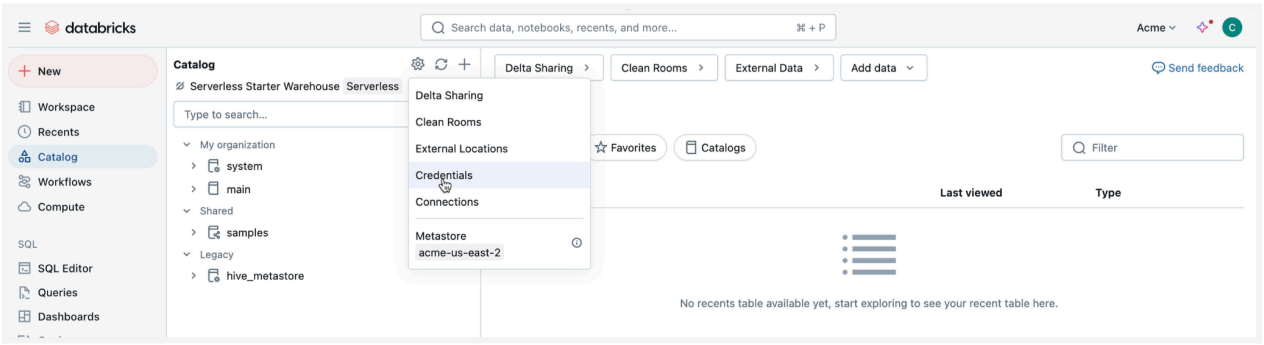
Note: We will reuse the AWS Role created earlier for Unity Catalog access, and simply attach the new policy for Onehouse S3 customer bucket access.

- [ ] 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'.

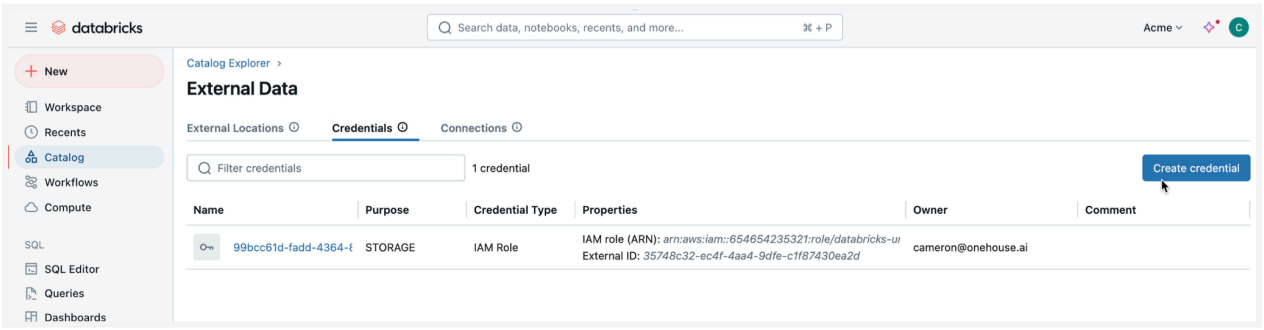
Create a Databricks Storage Credential

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

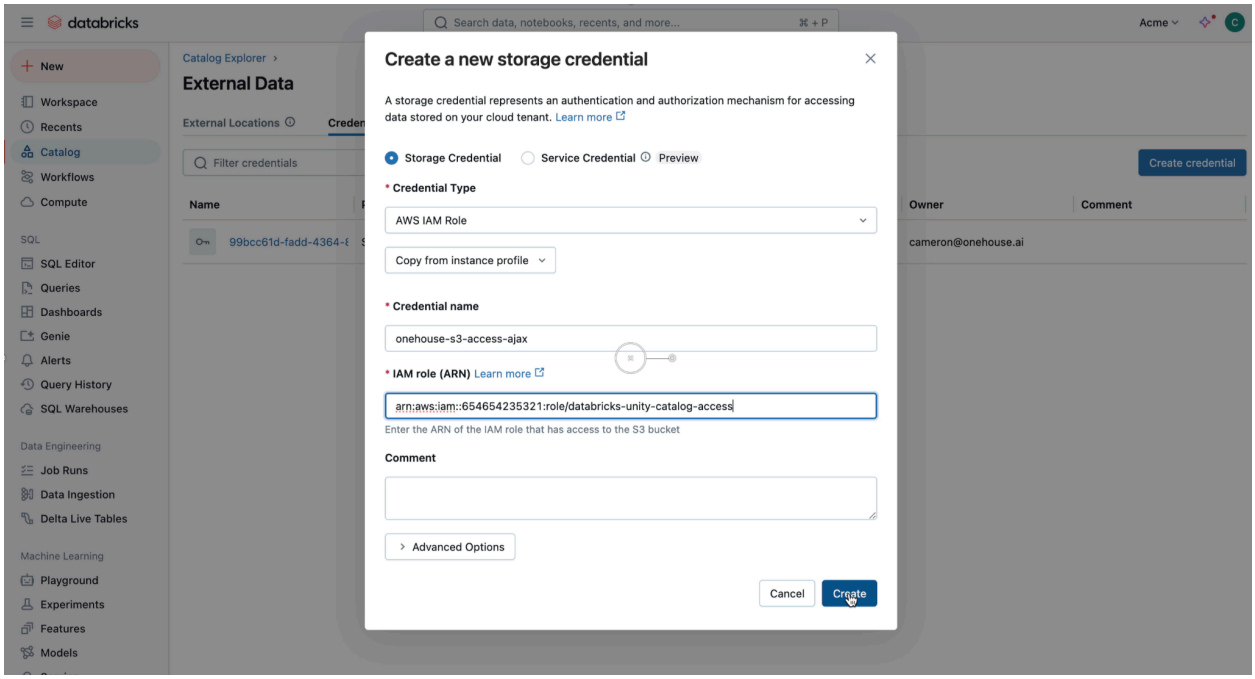




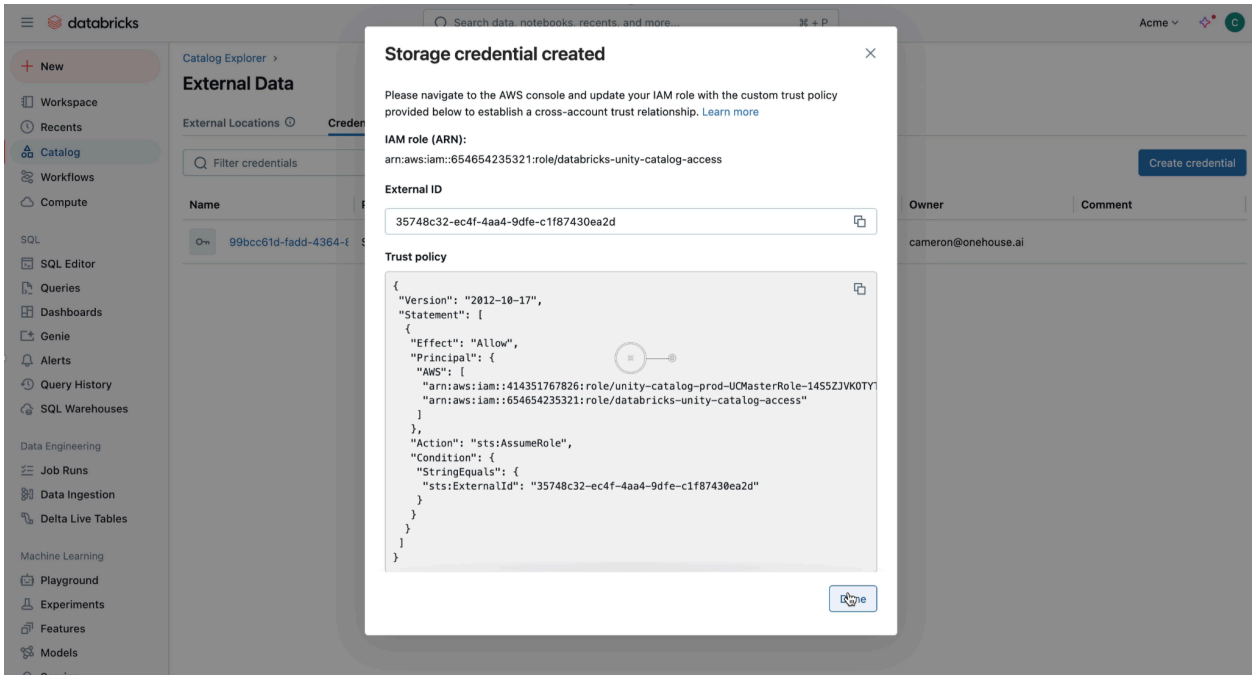
- [ ] Click 'Create credential'



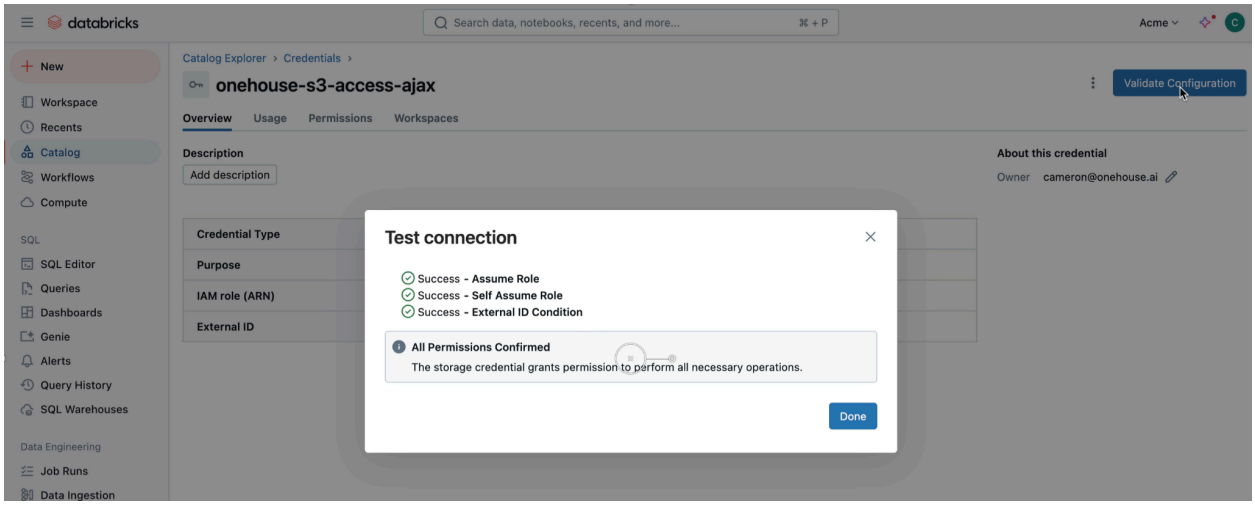
- [ ] For the storage credential name, choose something like `onehouse-s3-access`. Paste in the AWS ARN of the role you created earlier. Click ‘Create’.



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



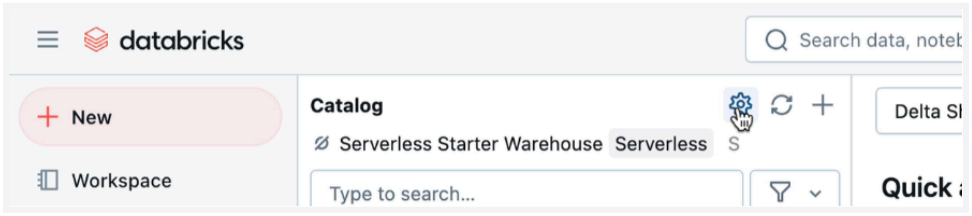
- [ ] Click ‘Validate Configuration’ and ensure that all the permissions are confirmed good.

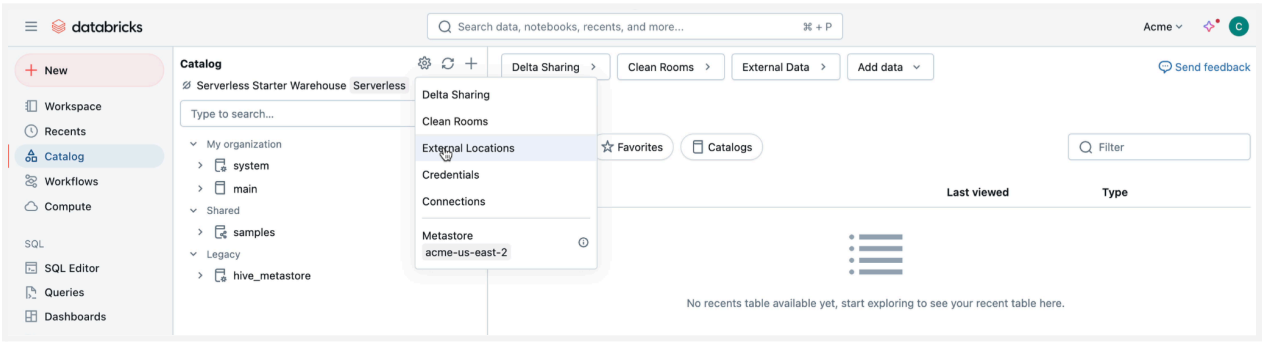


Create a Databricks External Location

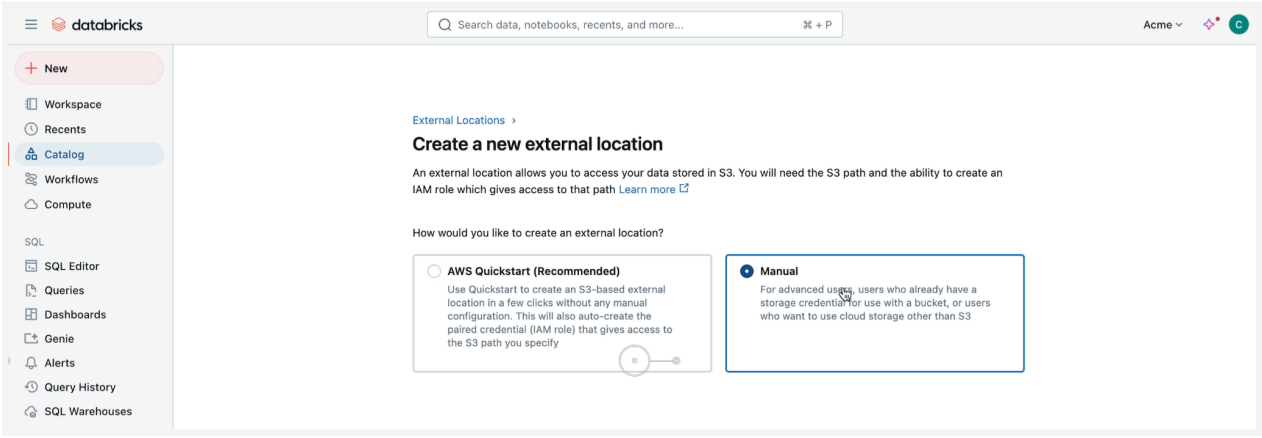
Since we are referencing an existing Onehouse S3 bucket, refer to the Databricks documentation: ‘Creating an external location manually using Catalog Explorer’ at: https://docs.databricks.com/en/connect/unity-catalog/cloud-storage/external-locations.html#create-an-external-location-manually-using-catalog-explorer

- [ ] In the Databricks workspace console, click ‘Catalog’, then the settings gear icon, then ‘External Locations’.

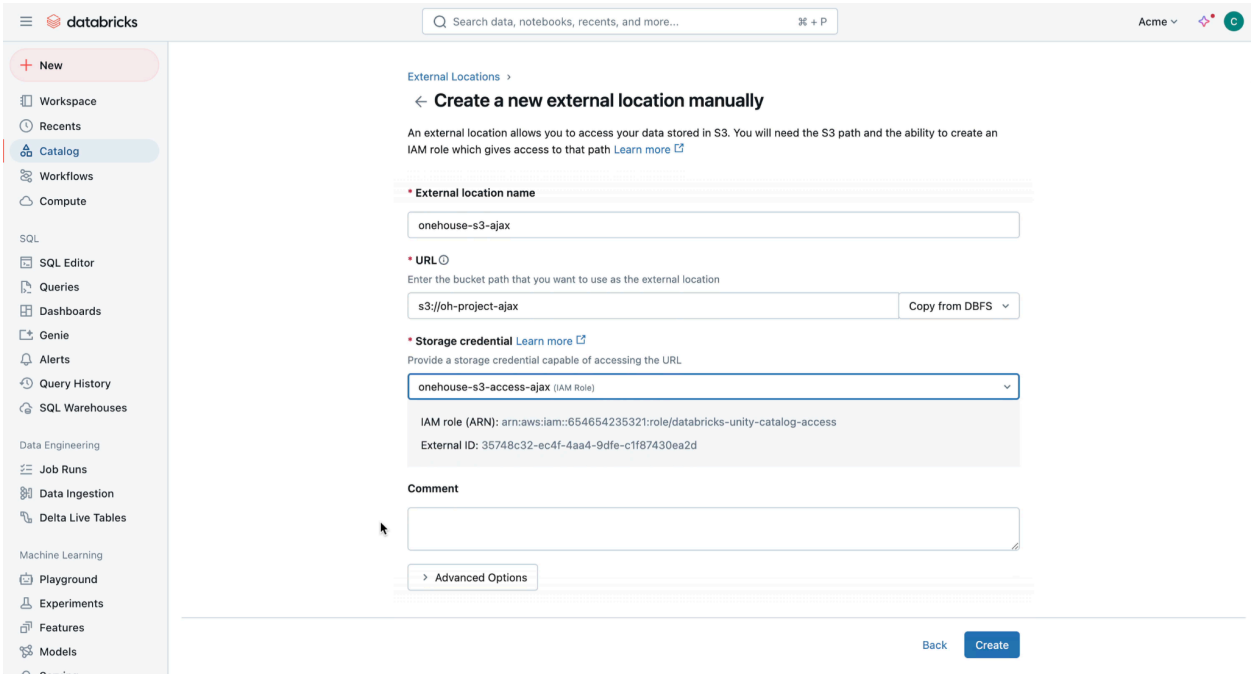




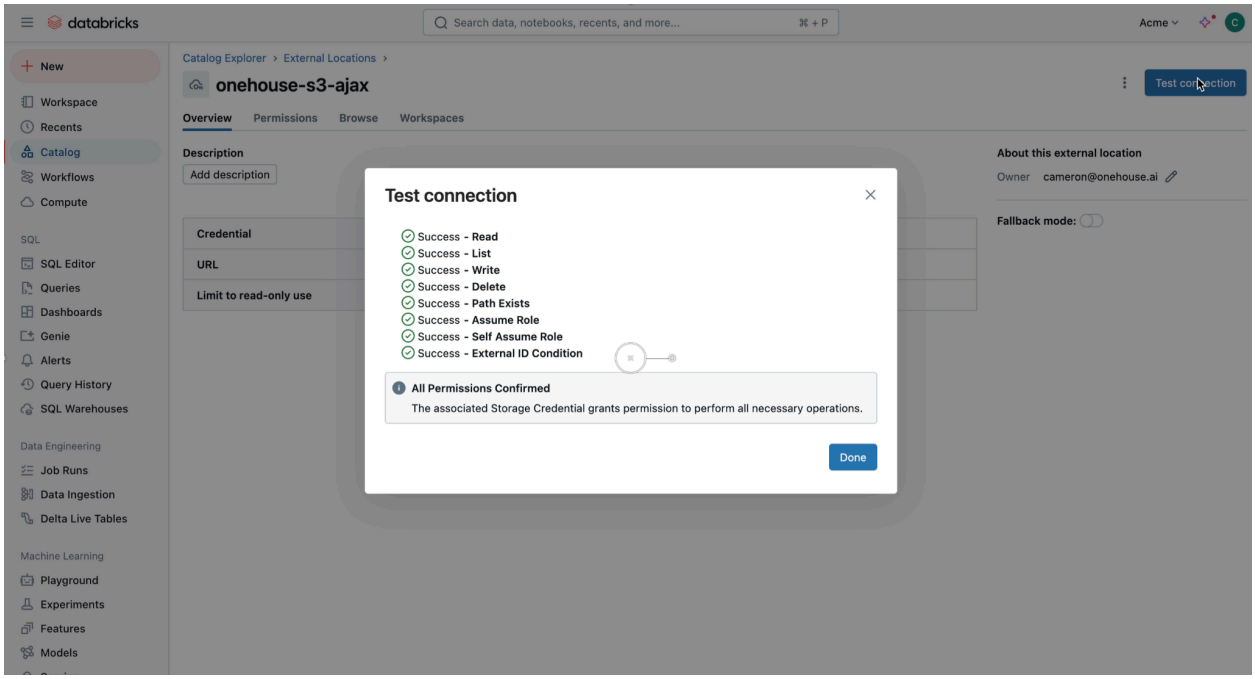
- [ ] Choose ‘Manual’, then click ‘Next’.



- [ ] 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.

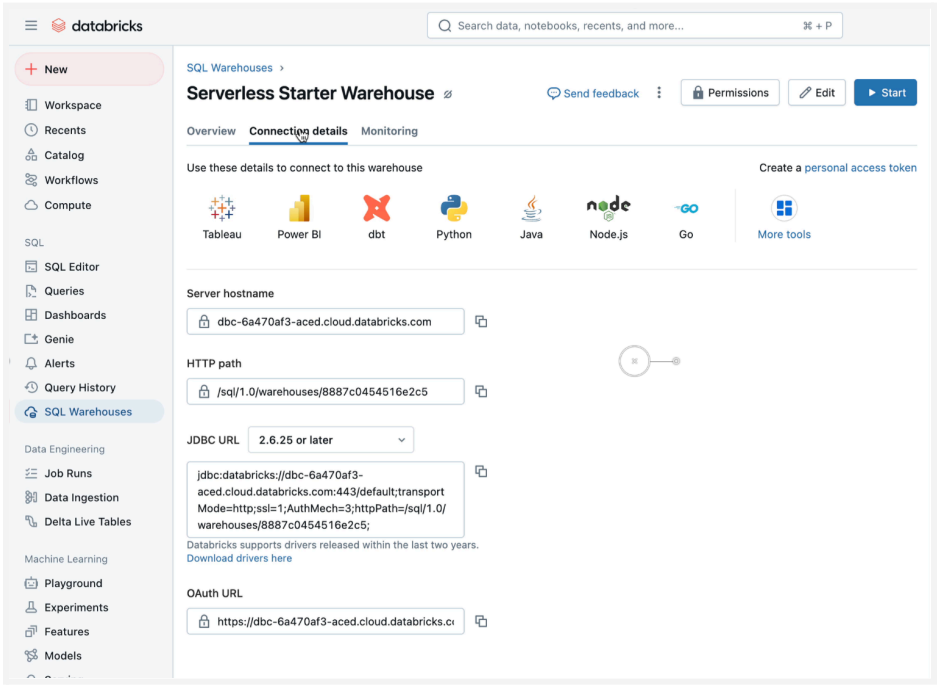


Gather the external catalog information in Databricks

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`.

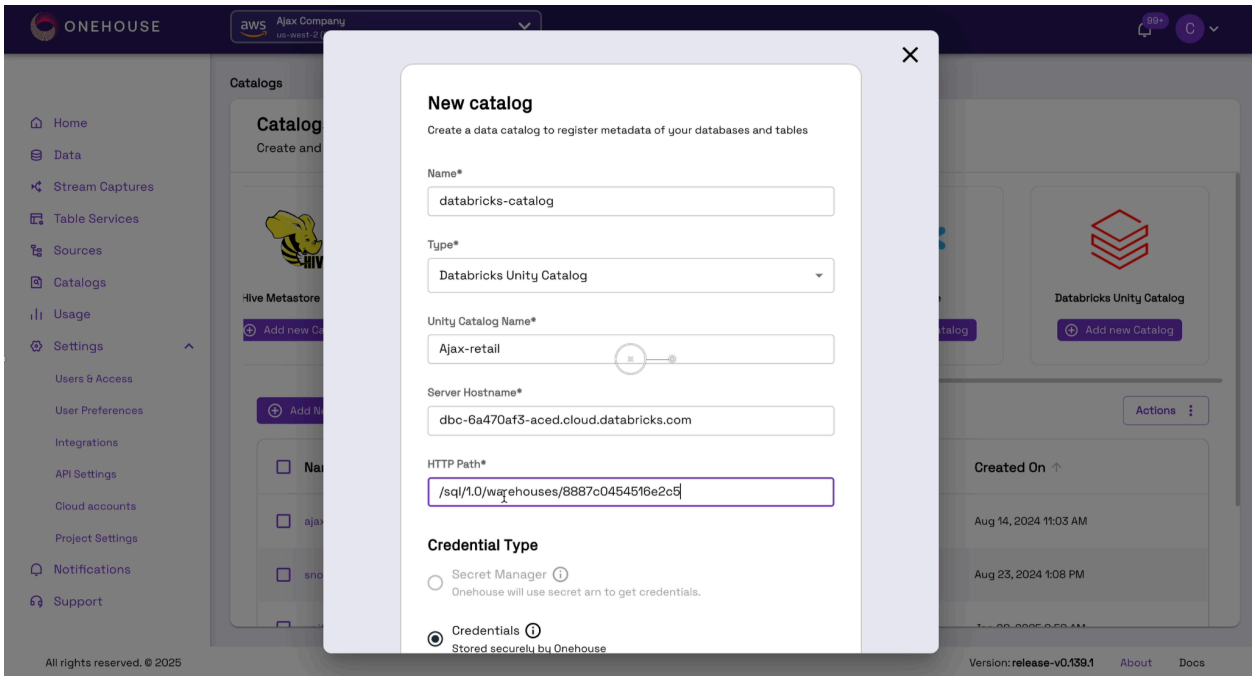


Create the Metadata Catalog Sync in Onehouse

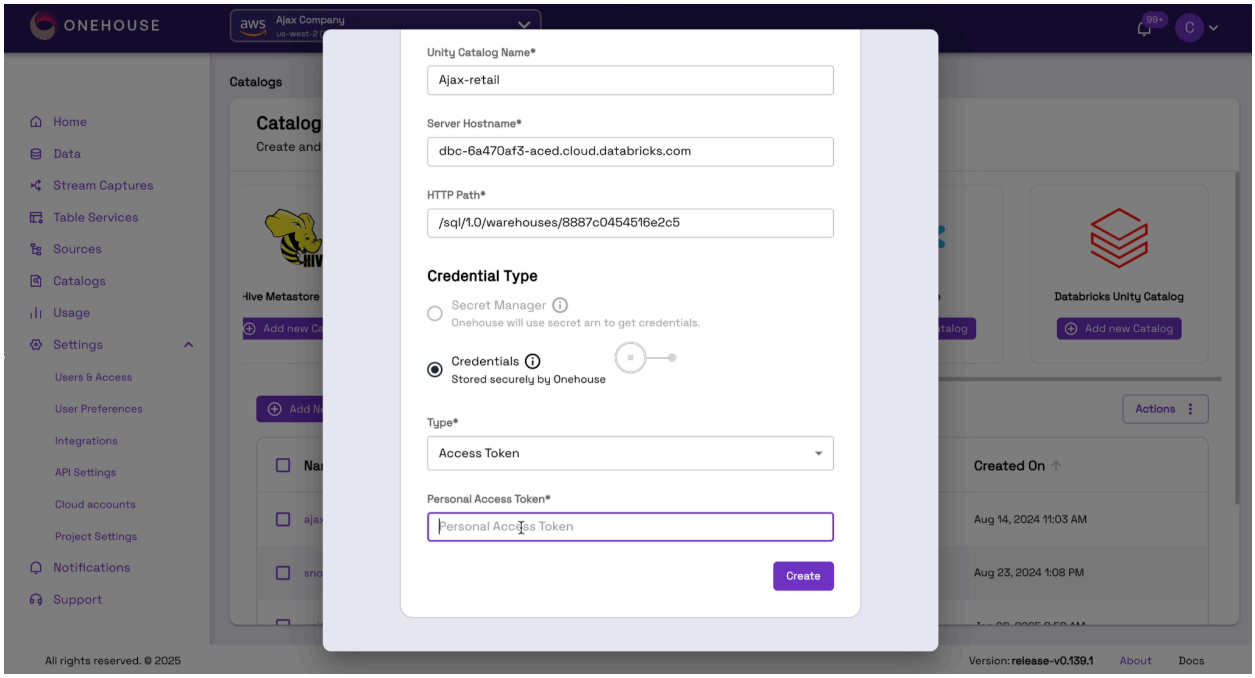
Finally! We are ready to create the metadata sync in Onehouse and attach it to a data lakehouse table.

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.

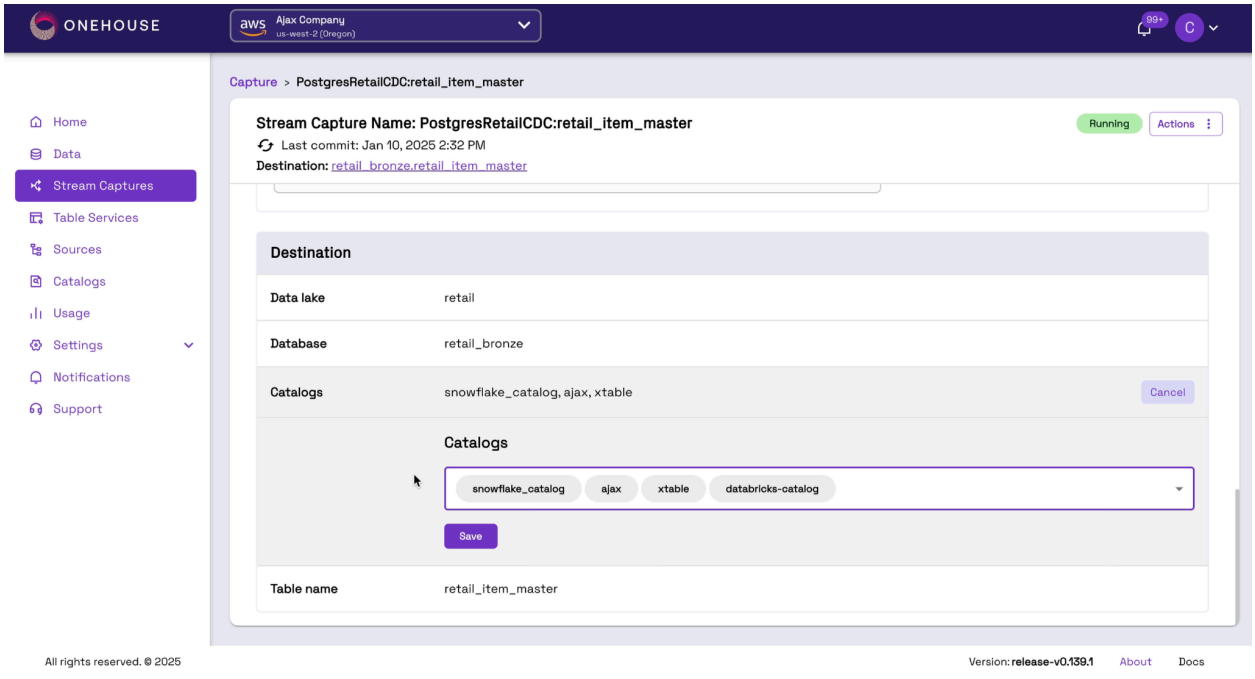


- ☐ Choose the Credential type.
    - For OAuth type, enter the service principal credentials ‘client-id’ and ‘client-secret’ from your workspace.
    - For Access Token type, enter the ‘personal-access-token’ for your workspace user.

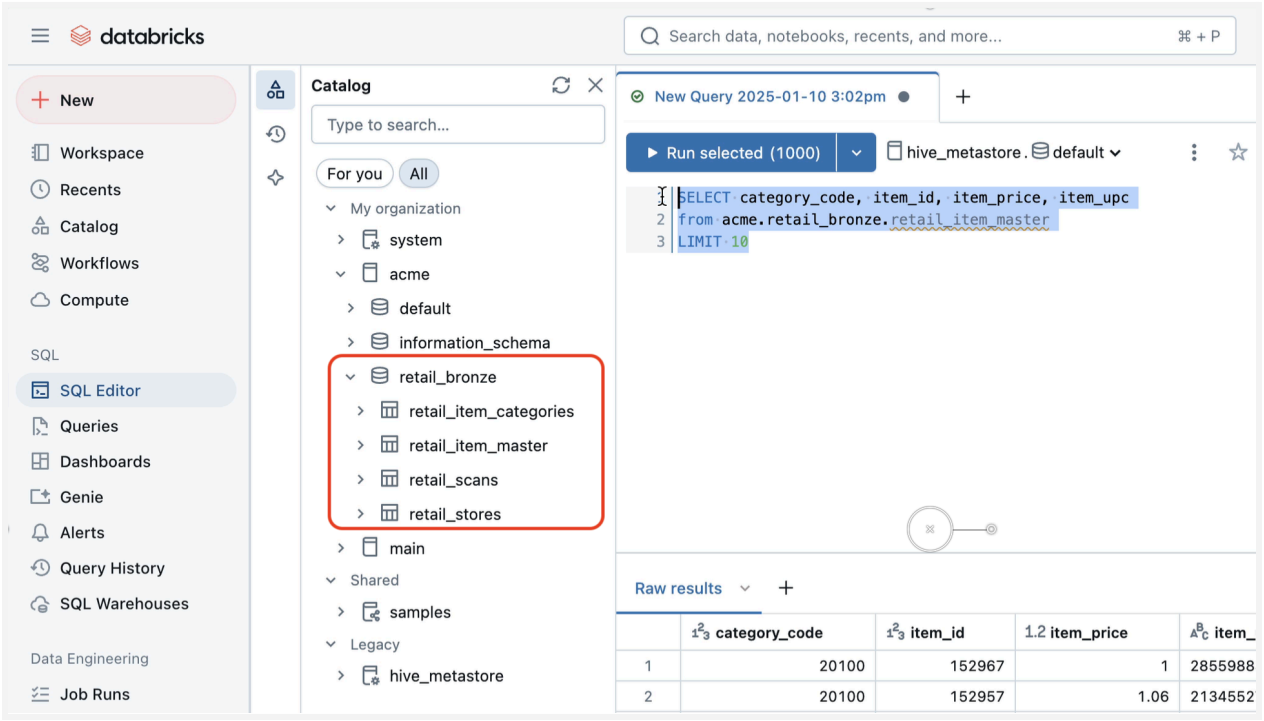


Attach the Onehouse Databricks catalog to one or more Stream Captures.

- [ ] 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.



###