To configure **Databricks** to use the **AWS Glue Data Catalog** as its metastore within the **same AWS account**, you need to set up appropriate IAM permissions, configure the Databricks cluster, and ensure connectivity to the Glue Data Catalog. Since the Databricks workspace and Glue Data Catalog are in the same AWS account, the setup is simpler than a cross-account configuration, as it avoids the need for cross-account IAM role assumptions. Below is a step-by-step guide, with the configuration details provided in an <xaiArtifact> tag as requested.

**Steps to Configure Databricks to Use AWS Glue Data Catalog as Metastore (Same AWS Account)**

**Prerequisites**

1. **AWS Admin Access**: Ensure you have administrator access to IAM roles and policies in the AWS account hosting both Databricks and the Glue Data Catalog.
2. **Databricks Workspace**: An active Databricks workspace in AWS.
3. **S3 and Glue Setup**: An S3 bucket for data storage and an active Glue Data Catalog in the same AWS account.
4. **IAM Role for Databricks**: An existing IAM role assigned to the Databricks cluster with permissions to access S3 for data storage.

**Configuration Steps**

1. **Update the Databricks IAM Role for Glue Access**:
   * In the AWS Management Console, navigate to **IAM** > **Roles** and locate the IAM role used by your Databricks cluster (e.g., databricks-cluster-role).
   * Attach a policy to this role that grants permissions to access the AWS Glue Data Catalog. The policy should include the following permissions:

json

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"glue:BatchCreatePartition",

"glue:BatchDeletePartition",

"glue:BatchGetPartition",

"glue:CreateDatabase",

"glue:CreateTable",

"glue:CreateUserDefinedFunction",

"glue:DeleteDatabase",

"glue:DeletePartition",

"glue:DeleteTable",

"glue:DeleteUserDefinedFunction",

"glue:GetDatabase",

"glue:GetDatabases",

"glue:GetPartition",

"glue:GetPartitions",

"glue:GetTable",

"glue:GetTables",

"glue:GetUserDefinedFunction",

"glue:GetUserDefinedFunctions",

"glue:UpdateDatabase",

"glue:UpdatePartition",

"glue:UpdateTable",

"glue:UpdateUserDefinedFunction"

],

"Resource": [

"arn:aws:glue:<AWS\_Region>:<AWS\_Account\_ID>:catalog",

"arn:aws:glue:<AWS\_Region>:<AWS\_Account\_ID>:database/\*",

"arn:aws:glue:<AWS\_Region>:<AWS\_Account\_ID>:table/\*/\*"

]

},

{

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:PutObject",

"s3:DeleteObject",

"s3:ListBucket"

],

"Resource": [

"arn:aws:s3:::<S3\_Bucket\_Name>/\*",

"arn:aws:s3:::<S3\_Bucket\_Name>"

]

}

]

}

* + - Replace <AWS\_Region> with your AWS region (e.g., us-east-1).
    - Replace <AWS\_Account\_ID> with your AWS account ID.
    - Replace <S3\_Bucket\_Name> with the S3 bucket used for table data storage (e.g., my-databricks-bucket).

1. **Optional: Add a Glue Data Catalog Resource Policy** (if fine-grained control is needed):
   * In the AWS Management Console, navigate to **AWS Glue** > **Data Catalog** > **Catalog Settings**.
   * Add a resource policy to explicitly allow the Databricks IAM role to access the Glue Data Catalog (optional, as the IAM role policy above is often sufficient in the same account):

json

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "DatabricksGlueAccess",

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::<AWS\_Account\_ID>:role/<Databricks\_Role\_Name>"

},

"Action": [

"glue:BatchCreatePartition",

"glue:BatchDeletePartition",

"glue:BatchGetPartition",

"glue:CreateDatabase",

"glue:CreateTable",

"glue:CreateUserDefinedFunction",

"glue:DeleteDatabase",

"glue:DeletePartition",

"glue:DeleteTable",

"glue:DeleteUserDefinedFunction",

"glue:GetDatabase",

"glue:GetDatabases",

"glue:GetPartition",

"glue:GetPartitions",

"glue:GetTable",

"glue:GetTables",

"glue:GetUserDefinedFunction",

"glue:GetUserDefinedFunctions",

"glue:UpdateDatabase",

"glue:UpdatePartition",

"glue:UpdateTable",

"glue:UpdateUserDefinedFunction"

],

"Resource": [

"arn:aws:glue:<AWS\_Region>:<AWS\_Account\_ID>:catalog",

"arn:aws:glue:<AWS\_Region>:<AWS\_Account\_ID>:database/\*",

"arn:aws:glue:<AWS\_Region>:<AWS\_Account\_ID>:table/\*/\*"

]

}

]

}

* + - Replace <AWS\_Account\_ID>, <Databricks\_Role\_Name>, and <AWS\_Region> with your actual values.

1. **Configure the Databricks Cluster**:
   * In your Databricks workspace, navigate to **Clusters** and select or create the cluster you want to configure.
   * Under **Advanced Options** > **Spark** tab, add the following Spark configurations to enable the Glue Data Catalog as the metastore:

plaintext

spark.databricks.hive.metastore.glueCatalog.enabled true

spark.hadoop.aws.region <AWS\_Region>

* + - Replace <AWS\_Region> with your AWS region (e.g., us-east-1).
  + If you plan to use **Apache Iceberg** tables with the Glue Data Catalog, add the following additional configurations:

plaintext

spark.sql.extensions org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions

spark.sql.catalog.glue org.apache.iceberg.spark.SparkCatalog

spark.sql.catalog.glue.catalog-impl org.apache.iceberg.aws.glue.GlueCatalog

spark.sql.catalog.glue.warehouse s3://<S3\_Bucket\_Name>/warehouse/

spark.sql.catalog.glue.io-impl org.apache.iceberg.aws.s3.S3FileIO

spark.jars.packages org.apache.iceberg:iceberg-spark-runtime-3.5\_2.12:1.7.0

* + - Replace <S3\_Bucket\_Name> with the S3 bucket used for your data warehouse (e.g., my-databricks-bucket).
    - Ensure the Databricks Runtime version supports Iceberg (e.g., Databricks Runtime 16.0 with Spark 3.5.0).

1. **Restart the Cluster**:
   * Save the cluster configuration and restart the cluster to apply the changes.
2. **Verify the Configuration**:
   * Run the following Spark SQL commands in a Databricks notebook to confirm that the Glue Data Catalog is being used:

sql

SHOW CATALOGS;

* + - You should see glue listed as a catalog.

sql

SHOW DATABASES IN glue;

SHOW TABLES IN glue.<database\_name>;

SELECT \* FROM glue.<database\_name>.<table\_name>;

* + - Replace <database\_name> and <table\_name> with actual values from your Glue Data Catalog.
  + Create a test table to verify write operations:

sql

CREATE TABLE glue.default.test\_table (id INT, name STRING)

USING PARQUET

LOCATION 's3://<S3\_Bucket\_Name>/test\_table/';

INSERT INTO glue.default.test\_table VALUES (1, 'Alice'), (2, 'Bob');

SELECT \* FROM glue.default.test\_table;

* + - Replace <S3\_Bucket\_Name> with your S3 bucket.

1. **Using FileStore with Glue Data Catalog**:
   * If you want to store data in **FileStore** (dbfs:/FileStore) and register it in the Glue Data Catalog:
     + Upload a file to FileStore (e.g., via UI or dbutils.fs.put):

python

dbutils.fs.put("dbfs:/FileStore/test\_data.csv", "id,name\n1,Alice\n2,Bob", overwrite=True)

* + - Register the file as a table in the Glue Data Catalog:

sql

CREATE TABLE glue.default.filestore\_table

USING CSV

OPTIONS (header 'true', inferSchema 'true')

LOCATION 'dbfs:/FileStore/test\_data.csv';

* + - Query the table:

sql

SELECT \* FROM glue.default.filestore\_table;

* + - Note: The actual file in FileStore is stored in the underlying S3 bucket associated with your DBFS root (e.g., s3://<databricks-root-bucket>/FileStore/test\_data.csv). Ensure the Databricks IAM role has access to this S3 bucket.

**Example Artifact: Spark Configuration for Databricks Cluster**

spark\_config.txt

plain

•

**Notes**

* **IAM Permissions**: Ensure the Databricks IAM role has permissions to access both the Glue Data Catalog and the S3 bucket(s) where table data is stored. The S3 permissions in the IAM policy above are critical for reading/writing data.
* **FileStore Integration**: When using FileStore with Glue, the file paths (dbfs:/FileStore/) resolve to the DBFS root S3 bucket. Ensure the IAM role has access to this bucket.
* **Iceberg Support**: If using Iceberg tables, verify that the Iceberg runtime JAR version matches your Databricks Runtime (e.g., iceberg-spark-runtime-3.5\_2.12:1.7.0 for Databricks Runtime 16.0).
* **Unity Catalog Alternative**: Databricks recommends **Unity Catalog** for modern metadata management, as it offers fine-grained access control and cross-workspace sharing. You can federate Glue Data Catalog with Unity Catalog if needed (see [Databricks Documentation](https://docs.databricks.com)).
* **Troubleshooting**:
  + If the glue catalog does not appear in SHOW CATALOGS, verify the Spark configurations and IAM permissions.
  + For errors like AccessDeniedException, check the IAM role and Glue resource policy for missing permissions.
  + For Iceberg-specific issues (e.g., java.lang.InstantiationException), ensure the correct Iceberg runtime JAR is included.