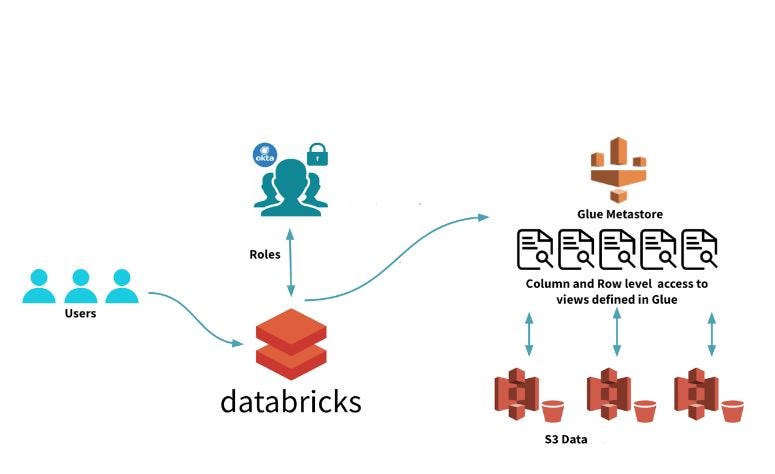
**Databricks with AWS Glue**



**Pre-requisites:**

* An AWS account with AWS Glue Data Catalog enabled.
* An existing Databricks workspace.

**Assumption:**

* Databricks Workspace and Glue Catalog are deployed in the same AWS Account.
* If not cross-account accesses can be set up with slight changes in configs. As mentioned here: [Policy for the target Glue Catalog](https://docs.databricks.com/archive/external-metastores/aws-glue-metastore.html#step-2-create-a-policy-for-the-target-glue-catalog)

**Process:**

**Step 1: Set Up AWS Glue Data Catalog**

* Log in to the AWS Management Console.
* Open the AWS Glue service and create a Data Catalog if you don’t already have a pre-configured one.
* Configure your databases, tables, and metadata in the Glue Data Catalog, if not already exist.

**Step 2: Configure IAM Role Permissions**

* Create an IAM role with the necessary permissions to access the Glue Data Catalog and interact with Databricks. Preferably add the following policy to the preconfigured EC2 Instance profiles.

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Sid": "GrantCatalogAccessToGlue",  
 "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:region:account-id:catalog",  
 "arn:aws:glue:region:account-id:database/database name",  
 "arn:aws:glue:region:account-id:table/database name/\*"  
 ]  
 }  
 ]  
}

* Note: You may add further limit tables if required and also add other glue resources like user-defined functions as detailed here: [Glue Specifying resource ARNs](https://docs.aws.amazon.com/glue/latest/dg/glue-specifying-resource-arns.html)
* If you’re using CFN to deploy the EC2 role this might help, its a template but you can specify details and deploy.

{  
 "AWSTemplateFormatVersion": "2010-09-09",  
 "Description": "Provision IAM Role Databricks Instance Profile",  
 "Parameters" : {  
 "RoleName" : {  
 "Type" : "String",  
 "Default" : "databricks-ec2-role",  
 "Description" : "Enter rolename for you Instance profile"  
 }  
 },  
 "Resources": {  
 "DatabricksEC2InstanceRole": {  
 "Type": "AWS::IAM::Role",  
 "Properties": {  
 "RoleName": { "Ref" : "RoleName" },  
 "AssumeRolePolicyDocument": {  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Effect": "Allow",  
 "Principal": {  
 "Service": "ec2.amazonaws.com"  
 },  
 "Action": "sts:AssumeRole"  
 }  
 ]  
 },  
 "Policies": [  
 {  
 "PolicyName": {"Fn::Join": [ "",[ { "Ref" : "RoleName" },"-s3Access"] ] },  
 "PolicyDocument": {  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Effect": "Allow",  
 "Action": [  
 "s3:List\*",  
 "s3:Get\*"  
 ],  
 "Resource": "\*"  
 },  
 {  
 "Effect": "Allow",  
 "Action": [  
 "s3:\*"  
 ],  
 "Resource": [  
 "<RootS3ARN>",  
 "<RootS3ARN>/\*"  
 ]  
 },  
 {  
 "Action": [  
 "s3:List\*",  
 "s3:Get\*",  
 "s3:Put\*",  
 "s3:DeleteObject"  
 ],  
 "Resource": [  
 "arn:aws:s3:::<TableS3ARN>",  
 "arn:aws:s3:::<TableS3ARN>/\*"  
 ],  
 "Effect": "Allow"  
 }  
 ]  
 }  
 },  
 {  
 "PolicyName": {"Fn::Join": [ "",[ { "Ref" : "RoleName" },"-glueAccess"] ] },,  
 "PolicyDocument": {  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Sid": "GrantCatalogAccessToGlue",  
 "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:region-id:aws-account-id:catalog",  
 "arn:aws:glue:region-id:aws-account-id:database/database-name",  
 "arn:aws:glue:region-id:aws-account-id:table/database-name/\*",  
 "arn:aws:glue:region-id:aws-account-id:userDefinedFunction/database-name/\*",  
 "arn:aws:glue:region-id:aws-account-id:connection/\*",  
 "arn:aws:glue:region-id:aws-account-id:database/default",  
 "arn:aws:glue:region-id:aws-account-id:table/default/\*"  
 ]  
 }  
 ]  
 }  
 }  
 ]  
 }  
 },  
 "InstanceProfile": {  
 "Type": "AWS::IAM::InstanceProfile",  
 "Properties": {  
 "Path": "/",  
 "Roles": [  
 {  
 "Ref": "DatabricksEC2InstanceRole"  
 }  
 ]  
 }  
 }  
 }  
}

* Ensure the IAM role has the necessary permissions to read/write metadata from the Glue Data Catalog. This includes underlying S3 bucket access too. If it’s an instance profile it should be able to access the S3 bucket the Glue metadata points to.
* Import the Instance profile to Databricks Workspace if not already enrolled. Your Databricks workspace deployment role should be able to pass this new IAM role for enrollment to be successful. For more details: [Look up the IAM role used to create the Databricks deployment](https://docs.databricks.com/en/connect/storage/tutorial-s3-instance-profile.html#step-4-locate-the-iam-role-that-created-the-databricks-deployment)

**Step 3: Configure Cluster Policy Settings**

* In your Databricks workspace, navigate to the “Clusters” section.
* Select the cluster on which you want to enable the Glue Metastore.
* Under the “Advanced Options” section, locate the “Spark” tab.
* In the “Spark” tab, scroll down to the “Metastore” section.

spark.sql.hive.convertMetastoreParquet false  
mapreduce.input.fileinputformat.input.dir.recursive true  
spark.databricks.hive.metastore.glueCatalog.enabled true

## If Glue catalog in a different account THEN Also Addspark.hadoop.hive.metastore.glue.catalogid <aws-account-id-for-glue-catalog>  
spark.hadoop.aws.region <aws-region-for-glue-catalog>

* I personally prefer creating a separate cluster policy for Glue access and adding the following would translate to automatically adding the above Spark Confs in the cluster:

"spark\_conf.spark.sql.hive.convertMetastoreParquet": {  
 "type": "fixed",  
 "value": "false"  
 },  
 "spark\_conf.mapreduce.input.fileinputformat.input.dir.recursive": {  
 "type": "fixed",  
 "value": "true"  
 },  
"spark\_conf.spark.databricks.hive.metastore.glueCatalog.enabled": {  
 "type": "fixed",  
 "value": "true",  
 "hidden": false  
 }

**Step 4: Apply Changes and Start the Cluster**

* Review the cluster configuration and ensure that all settings are accurate.
* Click “Create Cluster” or “Edit” (if you are modifying an existing cluster) to save the changes.
* Start the cluster to enable the Glue Metastore integration.

**Step 5: Verify Glue Metastore Integration**

* Create a new notebook or open an existing one in the Databricks workspace.
* Write Spark SQL queries or use DataFrame API operations to interact with the tables and metadata stored in the Glue Data Catalog.
* Validate that the data and metadata operations are successfully utilizing the Glue Metastore.