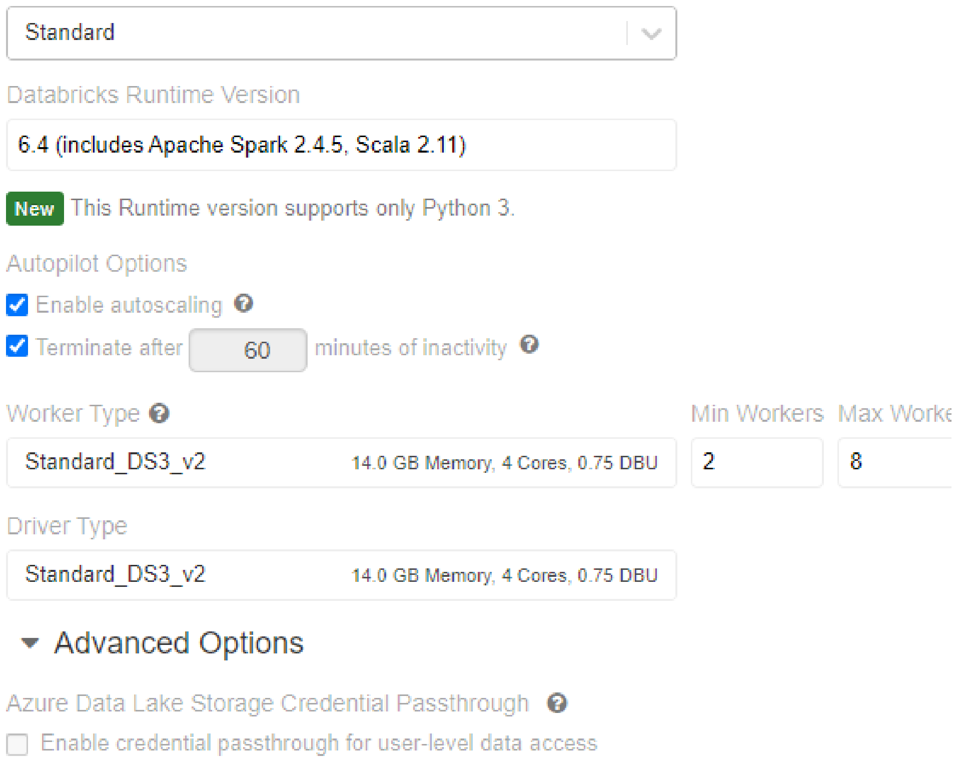
Databricks Setup /Hive Metastore

Databricks Configure clusters

Databricks runtime 6.4 was setup in standard mode with auto scaling enabled



Standard

Databricks Runtime Version

6.4 (includes Apache Spark 2.4.5, Scala 2.11)

New This Runtime version supports only Python 3.

Autopilot Options

☑ Enable autoscaling

☑ Terminate after 60 minutes of inactivity

Worker Type		Min Workers	Max Worke
Standard_DS3_v2	14.0 GB Memory, 4 Cores, 0.75 DBU	2	8

Driver Type

Standard_DS3_v2 14.0 GB Memory, 4 Cores, 0.75 DBU

Advanced Options

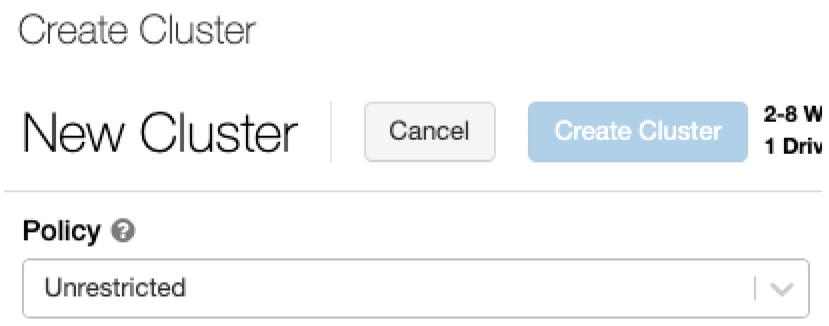
Azure Data Lake Storage Credential Passthrough

☐ Enable credential passthrough for user-level data access

Cluster policy

A cluster policy limits the ability to configure clusters based on a set of rules. The policy rules limit the attributes or attribute values available for cluster creation. Cluster policies have ACLs that limit their use to specific users and groups and thus limit which policies you can select when you create a cluster.

To configure a cluster policy, select the cluster policy in the **Policy** drop-down. The policy was set to unrestricted:



Policy

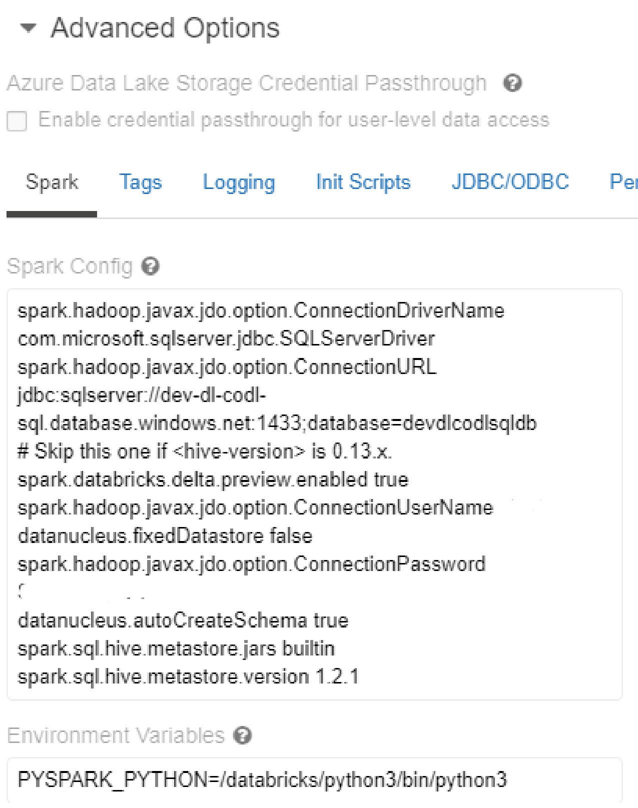
Unrestricted

Connecting to an external metastore using the UI

To set up an external metastore using the Databricks UI:

1. Click the **Clusters** button on the sidebar.

2. Click **Create Cluster**.
3. Enter the following Spark configuration options under Advance option:


Advanced Options
Azure Data Lake Storage Credential Passthrough
Enable credential passthrough for user-level data access
Spark
Tags
Logging
Init Scripts
JDBC/ODBC
Permissions
Spark Config
Environment Variables

```
spark.hadoop.javax.jdo.option.ConnectionDriverName
com.microsoft.sqlserver.jdbc.SQLServerDriver
spark.hadoop.javax.jdo.option.ConnectionURL
jdbc:sqlserver://dev-dl-codl-
sql.database.windows.net:1433;database=devdlcodlsqldb
# Skip this one if <hive-version> is 0.13.x.
spark.databricks.delta.preview.enabled true
spark.hadoop.javax.jdo.option.ConnectionUserName [redacted]
datanucleus.fixedDatastore false
spark.hadoop.javax.jdo.option.ConnectionPassword
[redacted]
datanucleus.autoCreateSchema true
spark.sql.hive.metastore.jars builtin
spark.sql.hive.metastore.version 1.2.1
```

```
PYSPARK_PYTHON=/databricks/python3/bin/python3
```

What is a Hive Metastore

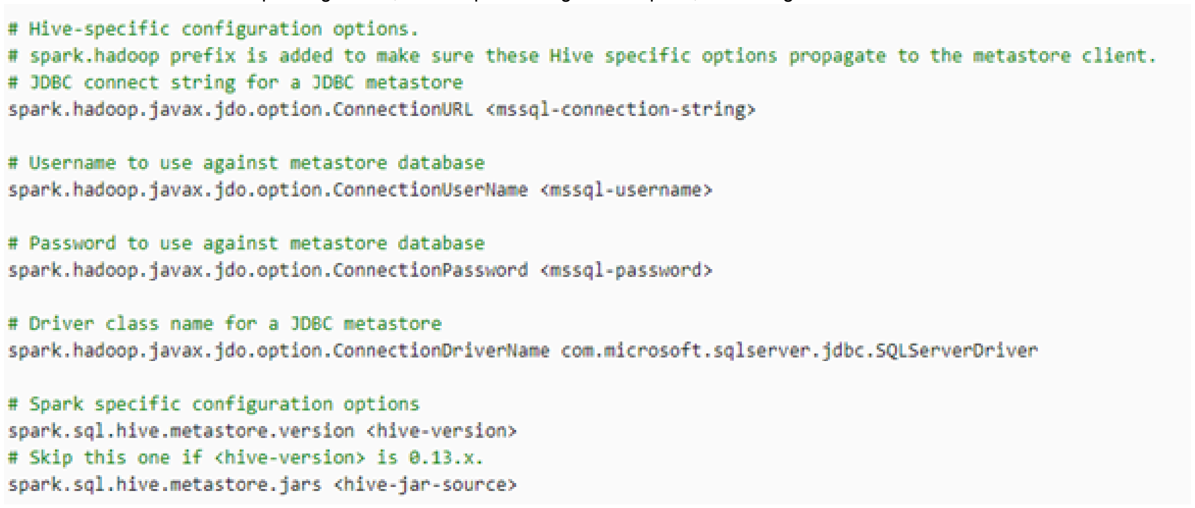
The Hive Metastore is a relational database repository containing metadata about objects you create in Hive. When a Hive table is created, the table definition (column names, data types, comments, etc.) are stored in the Hive Metastore. This is automatic and simply part of the Hive architecture.The reason why the Hive Metastore is critical is because it acts as a central schema repository which can be used by other access tools like Spark

Database diagram of Hive metastore:

External Hive metastore

- Every Azure Databricks deployment has a central Hive metastore accessible by all clusters to persist table metadata, including table and column names as well as storage location.
- By default, the metastore is managed by Azure in the shared Databricks control plane.
- Instead of using the default, you have the option to use your existing external Hive metastore instance.
- Any database with JDBC connectivity can be used as an external hive metastore.
- Azure Databricks can also initialize an empty database as a metastore, by setting specific options in the Spark configuration.
- Setting up the external hive metastore

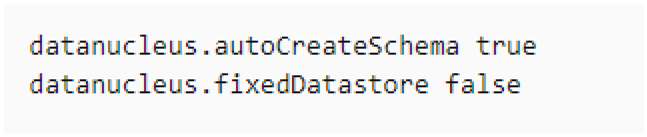
In the Databricks cluster setup configuration, under Spark configuration option, this configuration was used:



```
# Hive-specific configuration options.
# spark.hadoop prefix is added to make sure these Hive specific options propagate to the metastore client.
# JDBC connect string for a JDBC metastore
spark.hadoop.javax.jdo.option.ConnectionURL <mssql-connection-string>

# Username to use against metastore database
spark.hadoop.javax.jdo.option.ConnectionUserName <mssql-username>

# Password to use against metastore database
spark.hadoop.javax.jdo.option.ConnectionPassword <mssql-password>

# Driver class name for a JDBC metastore
spark.hadoop.javax.jdo.option.ConnectionDriverName com.microsoft.sqlserver.jdbc.SQLServerDriver

# Spark specific configuration options
spark.sql.hive.metastore.version <hive-version>
# Skip this one if <hive-version> is 0.13.x.
spark.sql.hive.metastore.jars <hive-jar-source>
```

<mssql-connection-string> is the JDBC connection string (which you can get in the Azure portal)

<mssql-username> and <mssql-password> specify the username and password of your Azure SQL Database account that has read/write access to the database
Depending on what hive version is used spark.sql.hive.metastore.jars can be easily configured by setting it to builtin. Hive 1.2.0 are built in Databricks 6.x and hive 2.3 is built in Databricks 7.x.
For all other Hive versions, Azure Databricks recommends that you download the metastore JARs and set the configuration spark.sql.hive.metastore.jars to point to the downloaded JARs

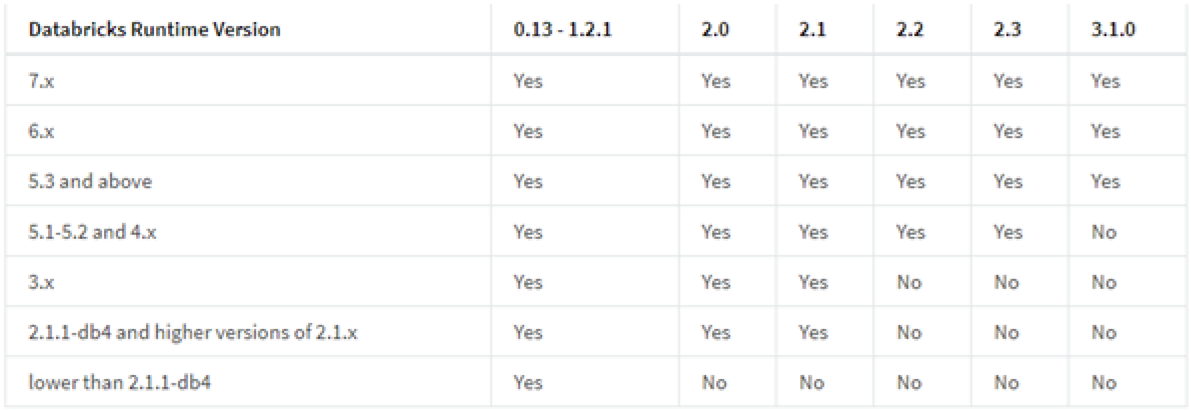


```
datanucleus.autoCreateSchema true
datanucleus.fixedDatastore false
```

The script also contains the above two lines of code; the Hive client library will try to create and initialize tables in the metastore database automatically. What it does is to run a “hive-schema-1.2.0.mssql.sql” to set up the Hive metastore database.

Hive metastore version and compatibility

The current initialized version of hive metastore is 1.2.1, which is compatible with all Databricks runtime and Immuta is compatible with all version of Hive metastore :

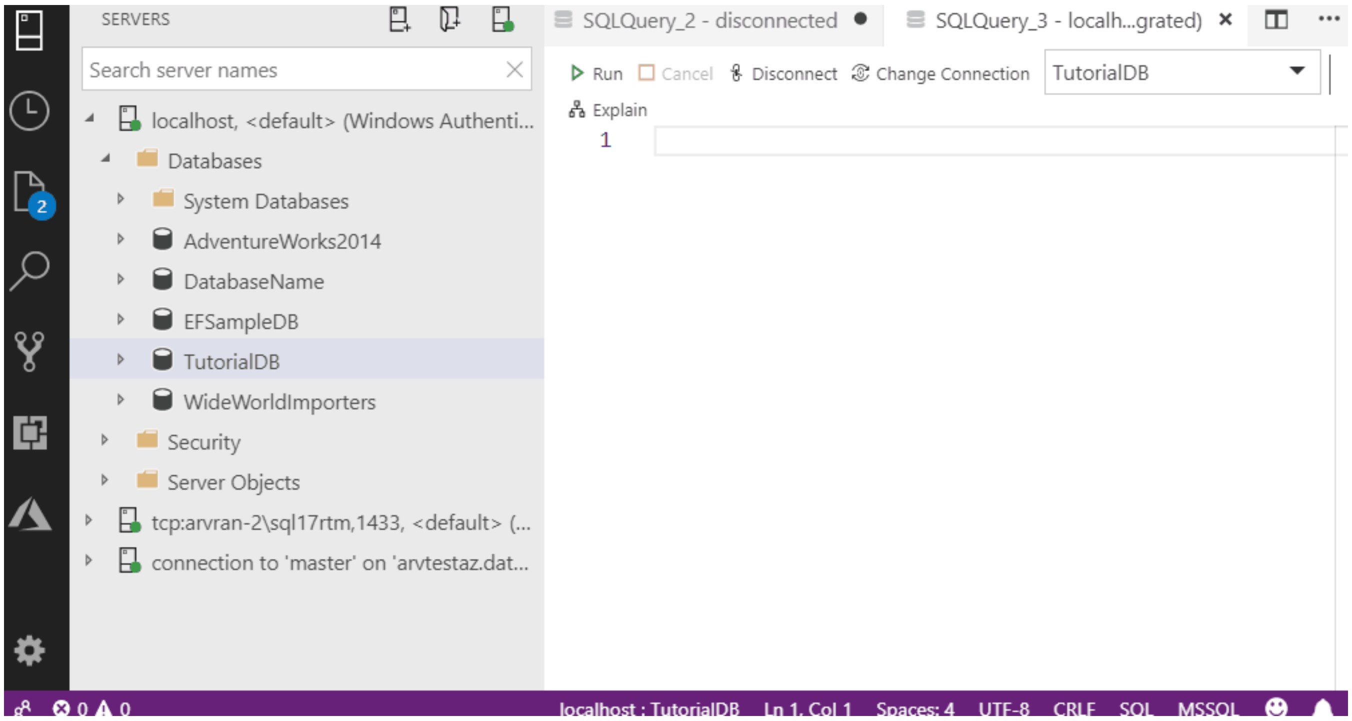


Databricks Runtime Version	0.13 - 1.2.1	2.0	2.1	2.2	2.3	3.1.0
7.x	Yes	Yes	Yes	Yes	Yes	Yes
6.x	Yes	Yes	Yes	Yes	Yes	Yes
5.3 and above	Yes	Yes	Yes	Yes	Yes	Yes
5.1-5.2 and 4.x	Yes	Yes	Yes	Yes	Yes	No
3.x	Yes	Yes	Yes	No	No	No
2.1.1-db4 and higher versions of 2.1.x	Yes	Yes	Yes	No	No	No
lower than 2.1.1-db4	Yes	No	No	No	No	No

The latest version of hive metastore is 3.x which is supposedly is significantly different from version 2.x and 1.2.x, which can cause compatibility uses with older external services or services that does not update frequently.

SQL Server dacpac/bacpac

An alternative way and now the current way of setting up a hive metastore is by using bacpac file. This is utilizing an extension in Azure Data studio. This approach is simple and straightforward,
the only thing we must do is backup a current hive metastore into a bacpac file that can later be deployed to an empty database.This can be accomplished by utilizing a feuture in Azure Data Studio:



this same Data-Tier application wizard can later be used to deploy the same bacpac file we created into a new Database. The current version of hive metastore we are using is 2.3.7.
One additional benefit to this approach, besides backing up hive metastore, we can also backup and deploy previously created tables, views, and databases and deploy them with ease.