AdminManual Metastore 3.0 Administration

Metastore 3.0 Administration

Version Note

This document applies only to the Metastore in Hive 3.0 and later releases. For Hive 0, 1, and 2 releases please see the Metastore Administration document.

Introduction

The definition of Hive objects such as databases, tables, and functions are stored in the Metastore. Depending on how the system is configured, statistics and authorization records may also be stored there. Hive, and other execution engines, use this data at runtime to determine how to parse, authorize, and efficiently execute user queries.

The Metastore persists the object definitions to a relational database (RDBMS) via DataNucleus, a Java JDO based Object Relational Mapping (ORM) layer. See Supported RDBMSs below for a list of supported RDBMSs that can be used.

The Metastore can be configured to embed the Apache Derby RDBMS or connect to a external RDBMS. The Metastore itself can be embedded entirely in a user process or run as a service for other processes to connect to. Each of these options will be discussed in turn below.

Changes From Hive 2 to Hive 3

Beginning in Hive 3.0, the Metastore can be run without the rest of Hive being installed. It is provided as a separate release in order to allow non-Hive systems to easily integrate with it. (It is, however, still included in the Hive release for convenience.) Making the Metastore a standalone service involved changing a number of configuration parameter names and tool names. All of the old configuration parameters and tools still work, in order to maximize backwards compatibility. This document will cover both the old and new names. As new functionality is added old, Hive style names will not be added.

For details on using the Metastore without Hive, see Running the Metastore Without Hive below.

General Configuration

The metastore reads its configuration from the file `metastore-site.xml`. It expects to find this file in `$METASTORE_HOME/conf` where `$METASTORE_HOME` is an environment variable. For backwards compatibility it will also read any `hive-site.xml` or `hive-metastoresite.xml` files found in `HIVE_HOME/conf`. Configuration options can also be defined on the command line (see Starting and Stopping the Service below).

Configuration values specific to running the Metastore with various RDBMSs, embedded or as a service, and without Hive are discussed in the relevant sections. The following configuration values apply to the Metastore regardless of how it is being run. This table covers only commonly customized configuration values. For less commonly changed configuration values see Less Commonly Changed Configuration Parameters.

Parameter	Hive 2 Parameter	Default Value	Description
metastore.warehouse.dir	hive.metastore.warehouse.dir		URI of the default location for tables in the default catalog and database.
datanucleus.schema.autoCreateAll	datanucleus.schema.autoCreateAll	false	Auto creates the necessary schema in the RDBMS at startup if one does not exist. Set this to false after creating it once. To enable auto create also set hive.metastore.schema.verification=false. **Auto creation is not recommended in production**; run `schematool` instead.

metastore.schema.verification	hive.metastore.schema.verification	true	Enforce metastore schema version consistency. When set to true: verify that version information stored in the RDBMS is compatible with the version of the Metastore jar. Also disable automatic schema migration. Users are required to manually migrate the schema after upgrade, which ensures proper schema migration. This setting is strongly recommended in production. When set to false: warn if the version information stored in RDBMS doesn't match the version of the Metastore jar and allow auto schema migration.
metastore.hmshandler.retry.attempts	hive.hmshandler.retry.attempts	10	The number of times to retry a call to the meastore when there is a connection error.
metastore.hmshandler.retry.interval	hive.hmshandler.retry.interval	2 sec	Time between retry attempts.
metastore.log4j.file	hive.log4j.file	none	Log4j configuration file. If unset will look for `metastore-log4j2.properties` in `$METASTORE_HOME/conf`
metastore.stats.autogather	hive.stats.autogather	true	Whether to automatically gather basic statistics during insert commands.

RDBMS

Option 1: Embedding Derby

The metastore can be run with Apache Derby embedded. This is the default configuration. However, it is not intended for use beyond simple testing. In this configuration only one client can use the Metastore and any changes are not durable beyond the life of the client (since it uses an in memory version of Derby).

Option 2: External RDBMS

For any durable, multi-user installation, an external RDBMS should be used to store Metastore objects. The Metastore connects to an external RDBMS via JDBC. Any jars required by the JDBC driver for your RDBMS should be placed in `METASTORE_HOME/lib` or explicilty passed on the command line. The following values need to be configured to connect the Metastore to an RDBMS. (Note: these configuration parameters did not change between Hive 2 and 3.)

Configuration Parameter	Comment
javax.jdo.option.ConnectionURL	Connection URL for the JDBC driver
javax.jdo.option.ConnectionDriverName	JDBC driver class
javax.jdo.option.ConnectionUserName	User name to connect to the RDBMS with
javax.jdo.option.ConnectionPassword	Password to connect to the RDBMS with. The Metastore uses Hadoop's CredentialProvider API so this does not have to be stored in clear text in your configuration file.

Supported RDBMSs

As the Metastore uses DataNucleus to communicate with the RDBMS, theoretically any storage option supported by DataNucleus would work with the Metastore. However, we only test and recommend the following:

RDBMS	Minimum Version	javax.jdo.option.ConnectionURL	javax.jdo.option.ConnectionDriverName
MS SQL Server	2008 R2	jdbc:sqlserver://<HOST>:<PORT>;DatabaseName=<SCHEMA>	`com.microsoft.sqlserver.jdbc.SQLServerDriver`
MySQL	5.6.17	jdbc:mysql://<HOST>:<PORT>/<SCHEMA>	`com.mysql.jdbc.Driver`
MariaDB	5.5	jdbc:mysql://<HOST>:<PORT>/<SCHEMA>	`org.mariadb.jdbc.Driver`
Oracle*	11g	jdbc:oracle:thin:@//<HOST>:<PORT>/xe	`oracle.jdbc.OracleDriver`
Postgres	9.1.13	jdbc:postgresql://<HOST>:<PORT>/<SCHEMA>	`org.postgresql.Driver`

<HOST> = The host the RDBMS is on.

<PORT> = Port the RDBMS is listening for JDBC connections on

<SCHEMA> = The schema (or database) that the Metastore stores its tables in.

*The Oracle values shown are for Oracle's thin JDBC client. If you are using a different client the ConnectionURL and ConnectionDriverName values will differ.

Special Note: When using Postgres you should set the configuration parameter `metastore.try.direct.sql.ddl` (previously `hive.metastore.try.direct.sql.ddl`) to false, to avoid failures in certain operations.

Installing and Upgrading the Metastore Schema

The Metastore provides the `schematool` utility to work with the Metastore schema in the RDBMS. For a full list of options see the `-help` option of the tool. The following summarizes what the tool can do. In most cases `schematool` can read the configuration from the `metastore-site.xml` file, though the configuration can also be passed as options on the command line.

- `-initSchema`: install a new schema. This should be used when first setting up a Metastore.
- `-upgradeSchema`: upgrade to the newly installed version. For 3.0, upgrades can be done from 1.2, 2.0, 2.1, 2.2, and 2.3 to 3.0. If you need to upgrade from before 1.2, use an older version of Hive's `schematool` to first upgrade your schema to 1.2, then use the current Metastore version to upgrade to 3.0.
- `-createUser`: create the Metastore user and schema. This does not install the tables, it just creates the database user and schema. This likely will not work in a production environment because you likely will not have permissions to create users and schemas. You will likely need your DBA to do this for you.
- `-validate`: check that your Metastore schema is correct for its recorded version

Running the Metastore

Embedded Mode

The Metastore can be embedded directly into a process as a library. This is often done with HiveServer2 to avoid an additional network hop for metadata operations. It can also be done when using the Hive CLI or any other process. This mode is the default and will be used anytime the configuration parameter `metastore.uris` is not set.

Except in the case of HiveServer2, using this mode raises a few concerns. First, having many clients will put a burden on the backing RDBMS since each client will have its own set of connections. Second, every client must have read/write access to the RDBMS. This makes it hard to properly secure the RDBMS. Therefore embedded mode is not recommended in production with the exception of HiveServer2.

Metastore Server

To run the Metastore as a service, you must first configure it with a URL.

Configured On	Parameter	Hive 2 Parameter	Format	Default Value	Comment
Client	metastore.thrift.uris	hive.metastore.uris	thrift://<HOST>:<PORT>[, thrift://<HOST>:<PORT>...]	none	HOST = hostname, PORT = should be set to match metastore.thrift.port on the server (which defaults to 9083. You can provide multiple servers in a comma separate list.
Server	metastore.thrift.port	hive.metastore.port	integer	9083	Port Thrift will listen on.

Once you have configured your clients, you can start the Metastore on a server using the `start-metastore` utility. See the `-help` option of that utility for available options. There is no stop-metastore script. You must locate the process id for the metastore and kill that process.

High Availability

The Metastore service is stateless. This allows you to start multiple instances of the service to provide for high availability. It also allows you to configure some clients to embed the metastore (e.g. HiveServer2) while still running a Metastore service for other clients. If you are running multiple Metastore services you can put all their URIs into your client's `metastore.thrift.uris` value and then set `metastore.thrift.uri.selection` (in Hive 2 `hive.metastore.uri.selection`) to `RANDOM` or `SEQUENTIAL`. `RANDOM` will cause your client to randomly select one of the servers in the list, while `SEQUENTIAL` will cause it to start at the beginning of the list and attempt to connect to each server in order.

Securing the Service

TODO: Need to fill in details for setting up with Kerberos, SSL, etc.

CLIENT_KERBEROS_PRINCIPAL, KERBEROS_*, SSL*, USE_SSL, USE_THRIFT_SASL

Running the Metastore Without Hive

Beginning in Hive 3.0, the Metastore is released as a separate package and can be run without the rest of Hive. This is referred to as standalone mode.

By default the Metastore is configured for use with Hive, so a few configuration parameters have to be changed in this configuration.

Configuration Parameter	Set to for Standalone Mode
metastore.task.threads.always	org.apache.hadoop.hive.metastore.events.EventCleanerTask,org.apache.hadoop.hive.metastore.MaterializationsCacheCleanerTask
metastore.expression.proxy	org.apache.hadoop.hive.metastore.DefaultPartitionExpressionProxy

Currently the following features have not been tested or are known not to work with the Metastore in standalone mode:

- The compactor (for use with ACID tables) cannot be run without Hive. ACID tables can be read and written to, but they cannot compacted.
- Replication has not been tested outside of Hive.

Performance Optimizations

CachedStore

Prior to Hive 3.0 there was only a single implementation of the MetaStore API (called `ObjectStore`). HIVE-16520 introduced a second implementation that can cache objects from the database in memory. This can save a significant amount of time for round trips to the database. It can be used by changing the parameter `metastore.rawstore.impl` to `org.apache.hadoop.hive.metastore.cache.CachedStore`.

The cache is automatically updated with new data when changes are made through this MetaStore. In a scenario where there are multiple MetaStore servers the caches can be out of date on some of them. To prevent this the CachedStore automatically refreshes the cache in a configurable frequency (default: 1 minute).

Details about all properties for the CachedStore can be found on Configuration Properties (Prefix: `metastore.cached`).

Less Commonly Changed Configuration Parameters

BATCHED_RETRIEVE_*, CLIENT_CONNECT_RETRY_DELAY, FILTER_HOOK, SERDES_USING_METASTORE_FOR_SCHEMA, SERVER_*_THREADS,

`THREAD_POOL_SIZE`

Security: EXECUTE_SET_UGI, metastore.authorization.storage.checks

Setting up Caching: CACHED*, CATALOGS_TO_CACHE & AGGREGATE_STATS_CACHE*

Transactions: MAX_OPEN_TXNS, TXNS_*