**Setting up hive 😊**

**Metastore :**

[https://hive.apache.org/docs/latest/adminmanual-metastore-3-administration\_75978150/#rdbms](https://hive.apache.org/docs/latest/adminmanual-metastore-3-administration_75978150/%23rdbms)

<https://hive.apache.org/docs/latest/adminmanual-metastore-administration_27362076/>

A screenshot of a computer

AI-generated content may be incorrect.

**Available RDBMS:**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Data Nucleus??**
* DataNucleus is a Java-based persistence framework that helps applications store Java objects into databases — both traditional relational databases (RDBMS) and NoSQL stores.
* In simple terms:
* It acts as a middle layer between Java objects and your actual database.
* It handles the mapping, storage, and retrieval of those objects transparently.
* Why is DataNucleus used in Hive Metastore?
* Hive’s Metastore is designed to store metadata (like databases, tables, schemas) as Java objects. These objects need to be persisted into a real database.
* DataNucleus helps with:
* Mapping Java objects (e.g. Table, Database, Partition)
* Generating SQL and managing CRUD operations
* Connecting to JDBC-compatible databases (like PostgreSQL, MySQL, etc.)
* Handling the object-relational mapping (ORM)
* How it fits in Hive Metastore architecture:

Hive Metastore

|

|---> Java Metadata Objects

|

|--- (Managed by) ---> DataNucleus

|

|--- (Connects via JDBC) ---> RDBMS (e.g., PostgreSQL)

* So why is this important?
* When you configure the Hive Metastore with something like:

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:postgresql://postgres:5432/metastore</value>

</property>

* You’re actually configuring DataNucleus (via JDO – Java Data Objects) to connect to your database
* DataNucleus is the ORM layer used by Hive to store metadata into databases.It translates Hive’s internal Java objects into SQL and stores them in your configured RDBMS,It works via JDO API, which is standard in Java for object persistence.

**There Are 2 Modes to Run the Metastore:**

* Embedded Mode (Default if you don’t set metastore URIs)

1. **What is it?**  
   The Metastore runs *inside* the same process as HiveServer2 or Hive CLI.
2. **When does it happen?**  
   If you don't set the parameter metastore.thrift.uris, Hive assumes you're using this mode.
3. **Example use case:**  
   HiveServer2 wants to access metadata, so instead of sending a network request to a metastore service, it uses an internal connection (like calling a function).

* Metastore Server Mode (Recommended for Production**)**

1. **What is it?**  
   You run a dedicated Hive Metastore server as a separate process or container, and HiveServer2 or other apps connect to it over Thrift protocol.
2. **Configuration Required:**

* You need to set this on the Hive clients (e.g. hive-server2):

<property>

<name>hive.metastore.uris</name>

<value>thrift://hive-metastore:9083</value>

</property>

* On the Metastore server container (the one that runs hive --service metastore), set:

<property>

<name>metastore.thrift.port</name>

<value>9083</value>

</property>

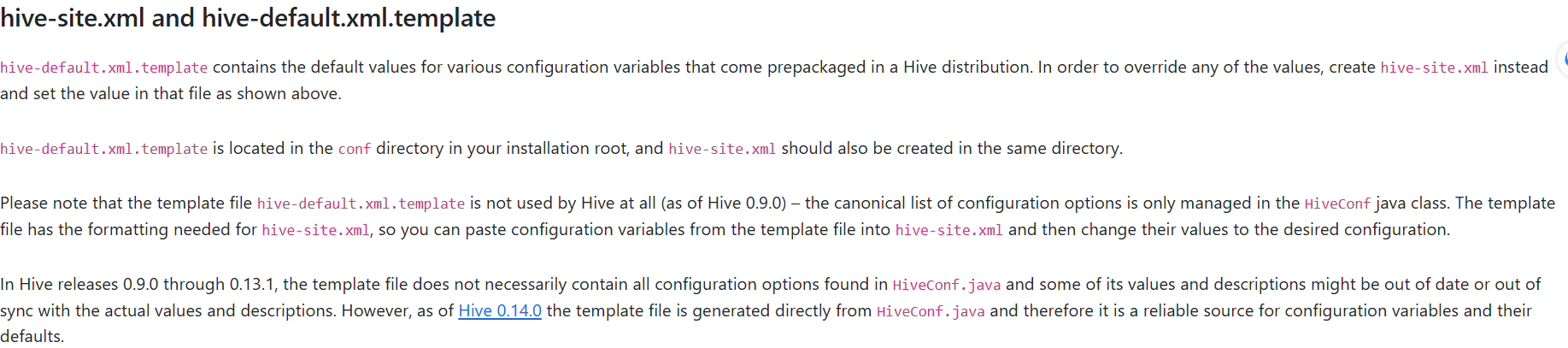
**Hiveserver2:**

* The order of precedence of the config files is as follows (later one overwrites)

*hive-site.xml ----> hivemetastore-site.xml ----> hiveserver2-site.xml ----> -hiveconf [commandline]*

* A number of configuration variables in Hive can be used by the administrator to change the behavior for their installations and user sessions. These variables can be configured in any of the following ways, shown in the order of preference:

1. Using the set command in the CLI or Beeline for setting session level values for the configuration variable for all statements subsequent to the set command
2. Using the --hiveconf option of the [hive] command or [beeline] command for the entire session.
3. In hive-site.xml. This is used for setting values for the entire Hive configuration



**Steps** :

1. You need first after deciding the right deployment method and the database used to creatre a DB container 🡪 postgres container
2. Download and Install Hive on Hadoop client container / move it and give it premisions:

COPY ./apache-hive-4.0.1-bin.tar.gz /tmp

RUN tar -xvf /tmp/apache-hive-4.0.1-bin.tar.gz -C /usr/local/ && \

    mv /usr/local/apache-hive-4.0.1-bin /usr/local/hive && \

    rm -f /tmp/apache-hive-4.0.1-bin.tar.gz && \

    chown -R hadoop:hadoop /usr/local/hive && \

    chmod -R 777 /usr/local/hive

1. Set Environment Variables (in ~/.bashrc or /etc/profile.d/hive.sh)

#Environment variables

ENV HIVE\_HOME=/usr/local/hive \

    tez\_home=/usr/local/tez \

    path=$PATH:/usr/local/hive/bin:/usr/local/tez/bin \

    HIVE\_CONF\_DIR=/usr/local/hive/conf \

 HADOOP\_CLASSPATH=/usr/local/hadoop/etc/hadoop:/usr/local/hive/lib/\*:/usr/local/tez/lib/\*

1. Create hive-site.xml and but it on /usr/local/hive/conf/ {&HIVE\_HOME }
2. Put the right configuration suitable for your setup
3. Download the write JDBC Driver 🡪 PostgreSQL JDBC and put it on the libraries inside hive:

#install postgresql driver

ADD https://jdbc.postgresql.org/download/postgresql-42.6.0.jar  /usr/local/hive/lib/

RUN chown hadoop:hadoop /usr/local/hive/lib/postgresql-42.6.0.jar && \

    chmod 777 /usr/local/hive/lib/postgresql-42.6.0.jar

1. Start services on each container accordingly

* Metastore

1. schematool -dbType postgres -initSchema
2. hive --service metastore
3. schematool -dbType postgres -validate

* Hiveserver2

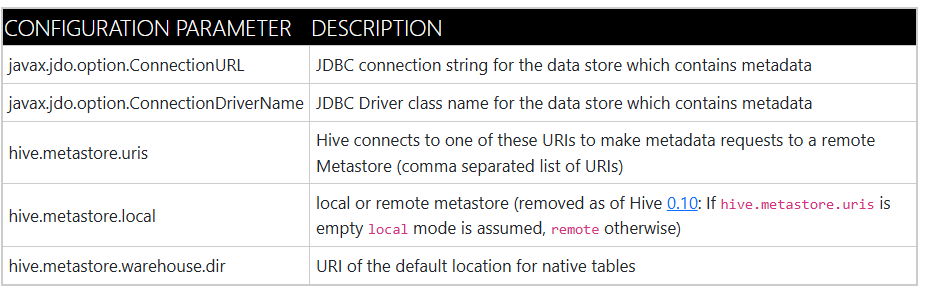
11.hive --service hiveserver2

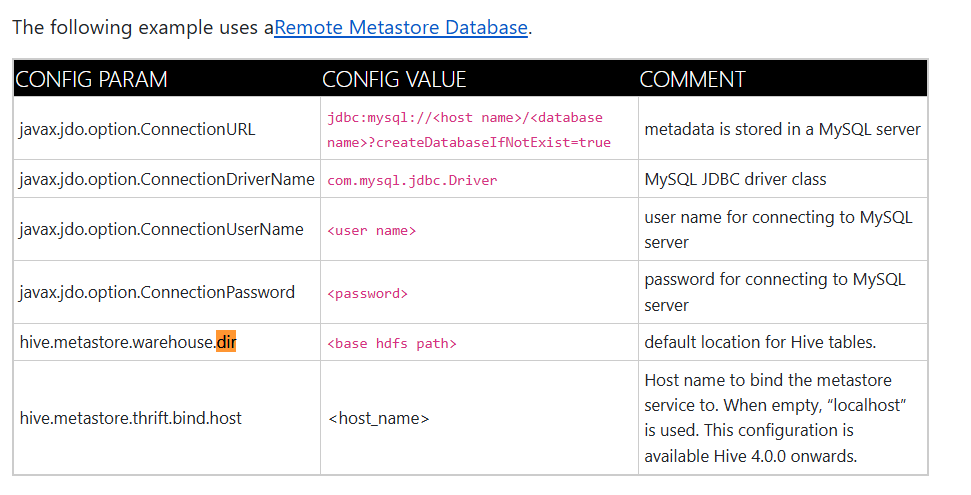
12. Test Hive Connection:

beeline -u jdbc:hive2://<hive-server2-host>:10000

(or)

Using Dbvisualizer hive connection

…

****

<https://hive.apache.org/docs/latest/adminmanual-metastore-administration_27362076/>

* **Hive Metastore & JDBC Configuration**

1. **hive.metastore.db.type**

* Sets the database type for the metastore, Tells Hive which RDBMS to expect for metastore storage.
* **Value:** postgres

1. **javax.jdo.option.ConnectionURL**

* JDBC connection URL for connecting to the external RDBMS, Locates the PostgreSQL database for the metastore
* **Value:** jdbc:postgresql://postgres-metastore:5432/metastore

1. **javax.jdo.option.ConnectionDriverName**

* Fully qualified JDBC driver class name.
* **Value:** org.postgresql.Driver

1. **javax.jdo.option.ConnectionUserName**

* Username for connecting to the RDBMS.
* **Value:** hive

1. **javax.jdo.option.ConnectionPassword**

* Password for the above user.
* **Value:** hive
* **Schema & Storage Configuration**

1. **hive.metastore.schema.verification**

* version check, Disables strict schema validation; helpful in dev/testing environments.
* **Value:** false

1. **hive.metastore.warehouse.dir**

* Path in HDFS where Hive will store database and table data,( storage location)
* **Value:** /user/hive/warehouse
* **Metastore Access & Events**

1. **hive.metastore.admins**

* Admin users for Hive metastore.
* **Value:** hive

1. **hive.metastore.event.db.notification.enable**

* Enables database-level event notifications, used for hive replicas
* **Value:** false

1. **hive.metastore.event.db.notification.api.auth**

* Enables authentication for metastore events API
* **Value:** false
* **HiveServer2 Configuration**

1. **hive.server2.thrift.port**

* Port on which HiveServer2 listens, **:** Required for clients like Beeline to connect.
* **Value:** 10000

1. **hive.server2.thrift.bind.host**

* Host binding for HiveServer2, Makes HiveServer2 accessible from outside the container
* **Value:** 0.0.0.0

1. **hive.server2.enable.doAs**

* HiveServer2 should impersonate as Hive user for simpler container management
* **Value:** false
* **Query Execution Engine**

1. **hive.execution.engine**

* The execution engine used for queries.
* **Value:** tez

Setting up tez 😊:

After making sure hive is setup well :>:>:>:>:>:>

1. Install tezz and put it on a directory /change the premisions

ADD https://archive.apache.org/dist/tez/0.10.4/apache-tez-0.10.4-bin.tar.gz /tmp

RUN tar -xvf /tmp/apache-tez-0.10.4-bin.tar.gz -C /usr/local/ && \

    mv /usr/local/apache-tez-0.10.4-bin /usr/local/tez && \

    rm -f /tmp/apache-tez-0.10.4-bin.tar.gz && \

    chown -R hadoop:hadoop /usr/local/tez && \

    chmod -R 777 /usr/local/tez

1. Set Environment Variables

ENV HIVE\_HOME=/usr/local/hive \

    tez\_home=/usr/local/tez \

    path=$PATH:/usr/local/hive/bin:/usr/local/tez/bin \

    HIVE\_CONF\_DIR=/usr/local/hive/conf \

HADOOP\_CLASSPATH=/usr/local/hadoop/etc/hadoop:/usr/local/hive/lib/\*:/usr/

local/tez/lib/\*

1. Copy Tez JARs to HDFS :

hdfs dfs -mkdir -p /apps/tez

hdfs dfs -put usr/local/tezz/\* /apps/tez/

1. Check configurations from :

https://tez.apache.org/install.html

A close-up of a white card

AI-generated content may be incorrect.





**(tez.lib.uris )**

is a configuration property in tez-site.xml that tells Tez **where to find its libraries** (JARs). These libraries need to be present **on HDFS**, so they can be distributed to the YARN containers where Tez will run jobs.

It supports different values >>:

* Simple file --- Will be copied and added to classpath
* Directory --- All **first-level** files are included (not recursive)
* Compressed archive --- Tez will **extract it**, but (define the classpath separately)

**(tez.lib.uris.classpath)**

if u use the compressed (.tar.gz, .tgz, .zip)

first Tez extracts them into the container's working dir but tez doesn't know how the internal folders are structured ,so you must manually tell it where to find the classes adding this property

**Two Main Modes to Deploy Tez:**

A screenshot of a computer error

AI-generated content may be incorrect.

<https://tez.apache.org/install.html>

>:>:> use *tez-x.y.z-minimal.tar.gz* downloaded with tez (no hadoop jars inside) ,tez relies on hadoop already installed on the cluster +(easy to use and maintain)

----------------------------------------------------------------------------------------------------------

A screenshot of a computer

AI-generated content may be incorrect.

>:>:>:> This mode will support rolling upgrades. It is the user's responsibility to ensure that the versions of Tez and Hadoop being used are compatible. To do this configuration, we need to

* Upload both tez-minimal and hadoop tarball to hdfs
* the text immediately following the ‘#’ is a fragment that refers to a symlink that will be created for the archive. If no fragment is given, the symlink will be set to the name of the archive