Apache Hadoop

Hive 2.1.0 install on Ubuntu 16.04

- **C'est quoi Hive**

Apache Hive est considéré comme le standard de facto pour les requêtes SQL interactives sur des pétaoctets de données dans Hadoop.
Hadoop a été conçu pour organiser et stocker des quantités énormes de données de toutes formes, tailles et formats. En raison de l'architecture "schema on read" de Hadoop, un cluster Hadoop constitue un réservoir parfait de données hétérogènes, structurées et non structurées, provenant d'une multitude de sources

Les analystes de données utilisent Hive pour interroger, résumer, explorer et analyser ces données, puis les transformer en informations commerciales exploitables.

Hive fournit également un mécanisme pour projeter la structure sur ces données et les interroger à l'aide d'un langage de type SQL appelé HiveQL.

HiveQL permet également aux programmeurs de map / reduce traditionnels de brancher leurs mappeurs et réducteurs personnalisés.

- **Installer Hive**

```
hduser@Hadoop:~$ wget https://archive.apache.org/dist/hive/hive-
2.1.0/apache-hive-2.1.0-bin.tar.gz
```

```
hduser@Hadoop:~$ sudo tar xvzf apache-hive-2.1.0-bin.tar.gz -C /usr/local
```

Ouvrez **~/.bashrc** et définissez la variable d'environnement HIVE_HOME pour qu'elle pointe vers le répertoire d'installation et PATH:

```
export HIVE_HOME=/usr/local/apache-hive-2.1.0-bin
export HIVE_CONF_DIR=/usr/local/apache-hive-2.1.0-bin/conf
export PATH=$HIVE_HOME/bin:$PATH
export CLASSPATH=$CLASSPATH:/usr/local/hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/usr/local/apache-hive-2.1.0-bin/lib/*:.
```

Sourcer le fichier bashrc

```
hduser@Hadoop:~$ source ~/.bashrc
```

- **Création du répertoire Hive warehouse**

Hive utilise Hadoop, nous devons donc avoir Hadoop dans notre path:

```
$ echo $HADOOP_HOME
```

```
/usr/local/hadoop
```

De plus, nous devons utiliser les commandes HDFS ci-dessous pour créer /tmp et /user/hive/warehouse (alias **hive.metastore.warehouse.dir**) et les définir chmod g + w avant de pouvoir créer une table dans Hive.

```
hduser@Hadoop:~$ hdfs dfs -mkdir -p /user/hive/warehouse
hduser@Hadoop:~$ hdfs dfs -mkdir /user/hduser
hduser@Hadoop:~$ hdfs dfs -mkdir /tmp
hduser@Hadoop:~$ hdfs dfs -chmod g+w /tmp
hduser@Hadoop:~$ hdfs dfs -chmod g+w /user/hive/warehouse
hduser@Hadoop:~$ hdfs dfs -chmod g+w /user/hduser
```

```
hduser@Hadoop:~$ hdfs dfs -ls /
18/10/18 22:53:22 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 3 items
drwxr-xr-x   - hduser supergroup          0 2018-10-18 17:43 /hbase
drwxrwxr-x   - hduser supergroup          0 2018-10-18 22:52 /tmp
drwxr-xr-x   - hduser supergroup          0 2018-10-18 22:51 /user

hduser@Hadoop:~$ hdfs dfs -ls /user
18/10/18 22:58:05 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 2 items
drwxrwxr-x   - hduser supergroup          0 2018-10-18 22:56 /user/hduser
drwxr-xr-x   - hduser supergroup          0 2018-10-18 22:51 /user/hive
```

le directory **warehouse** est l'emplacement où les tables ou les données liées à Hive seront stockées et le répertoire temporaire **tmp** est l'emplacement temporaire pour stocker le résultat intermédiaire du traitement.

- **Configurer Hive**

Pour configurer Hive avec Hadoop, nous devons éditer le fichier **hive-env.sh**, qui est placé dans le répertoire $HIVE_HOME/conf.
Faites les commandes suivantes :

```
hduser@Hadoop:~$ cd $HIVE_HOME/conf
hduser@Hadoop:/usr/local/apache-hive-2.1.0-bin/conf$ sudo cp hive-
env.sh.template hive-env.sh
```

Editez le fichier hive-env.sh en ajoutant la ligne suivante:
```
export HADOOP_HOME=/usr/local/hadoop
```

L'installation de Hive est terminée avec succès. Nous avons maintenant besoin d'un serveur de base de données externe pour configurer **Metastore**. Nous utilisons la base de données **Apache Derby**.

- **Downloading Apache Derby**

```
hduser@Hadoop:/usr/local/apache-hive-2.1.0-bin/conf$ cd
```

```
hduser@Hadoop:~$ wget http://archive.apache.org/dist/db/derby/db-derby-10.13.1.1/db-derby-10.13.1.1-bin.tar.gz

hduser@Hadoop:~$ sudo tar xvzf db-derby-10.13.1.1-bin.tar.gz -C /usr/local
```

Configurons l'environnement Derby en ajoutant les lignes suivantes au fichier ~/.bashrc:

```
export DERBY_HOME=/usr/local/db-derby-10.13.1.1-bin
export PATH=$PATH:$DERBY_HOME/bin
export CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar
```

Nous devons créer un répertoire nommé **data** dans le répertoire **$DERBY_HOME** pour stocker les données **Metastore**.

hduser@Hadoop:~$ sudo mkdir $DERBY_HOME/data

Nous avons maintenant terminé l'installation de Derby et la configuration environnementale.

- **Configuration de Hive Metastore**

Configurer Metastore signifie spécifier à Hive où la base de données est stockée. Nous allons le faire en modifiant le fichier **hive-site.xml**, qui se trouve dans le répertoire **$HIVE_HOME/conf**.
Copions le fichier template en utilisant la commande suivante:

hduser@Hadoop:~$ cd $HIVE_HOME/conf
hduser@Hadoop:/usr/local/apache-hive-2.1.0-bin/conf$ sudo cp hive-default.xml.template hive-site.xml

Assurez-vous que les lignes suivantes se trouvent entre les balises <configuration> et </configuration> de **hive-site.xml**:

```
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:derby:;databaseName=metastore_db;create=true</value>
    <description>
      JDBC connect string for a JDBC metastore.
      To use SSL to encrypt/authenticate the connection, provide database-specific
SSL flag in the connection URL.
      For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
    </description>
  </property>
```

Créez un fichier nommé **jpox.properties** et ajoutez-y les lignes suivantes:

```
javax.jdo.PersistenceManagerFactoryClass =

org.jpox.PersistenceManagerFactoryImpl
org.jpox.autoCreateSchema = false
org.jpox.validateTables = false
org.jpox.validateColumns = false
org.jpox.validateConstraints = false
org.jpox.storeManagerType = rdbms
org.jpox.autoCreateSchema = true
```

```
org.jpox.autoStartMechanismMode = checked
org.jpox.transactionIsolation = read_committed
javax.jdo.option.DetachAllOnCommit = true
javax.jdo.option.NontransactionalRead = true
javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver
javax.jdo.option.ConnectionURL =
jdbc:derby://hadoop1:1527/metastore_db;create = true
javax.jdo.option.ConnectionUserName = APP
javax.jdo.option.ConnectionPassword = mine
```

Nous devons définir **l'autorisation** sur le dossier **Hive**:

```
hduser@Hadoop:/usr/local/apache-hive-2.1.0-bin/conf$ cd /usr/local
hduser@Hadoop:/usr/local$ sudo chown -R hduser:hadoop apache-hive-2.1.0-bin
```

- **Metastore schema initialization**

À partir de Hive 2.1, nous devons exécuter la commande **schematool** ci-dessous comme étape d'initialisation. Dans notre cas, nous utilisons derby comme type de base de données:

```
hduser@Hadoop:/usr/local$ cd /usr/local/apache-hive-2.1.0-bin/bin
hduser@Hadoop:/usr/local/apache-hive-2.1.0-bin/bin$ schematool -dbType
derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-2.1.0-
bin/lib/log4j-slf4j-impl-
2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in
[jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:
jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :   org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:      APP
Starting metastore schema initialization to 2.1.0
Initialization script hive-schema-2.1.0.derby.sql
Initialization script completed
schemaTool completed
```

- **Vérification de l'installation de Hive en exécutant Hive CLI**

Pour utiliser l'interface de ligne de commande Hive à partir du shell, exécutez la commande **bin/hive** pour vérifier Hive.

```
hduser@Hadoop:/usr/local/apache-hive-2.1.0-bin/bin$ cd

hduser@Hadoop:~$ echo $HIVE_HOME
/usr/local/apache-hive-2.1.0-bin

hduser@Hadoop:~$ $HIVE_HOME/bin/hive
```

Quelques erreurs peuvent survenir lorsque nous essayons de démarrer hive via la commande bin/hive. Les erreurs et les correctifs correspondants sont les suivants:

Editer le fichier **hive-site.xml**:

```
  <property>
    <name>hive.downloaded.resources.dir</name>
    <!--
    <value>${system:java.io.tmpdir}/${hive.session.id}_resources</value>
    -->
    <value>/home/hduser/hive/tmp/${hive.session.id}_resources</value>
    <description>Temporary local directory for added resources in the
remote file system.</description>
  </property>

  <property>
    <name>hive.exec.local.scratchdir</name>
    <!--
    <value>${system:java.io.tmpdir}/${system:user.name}</value>
    -->
    <value>/tmp/mydir</value>
    <description>Local scratch space for Hive jobs</description>
  </property>
```

Maintenant que nous avons corrigé les erreurs, démarrons Hive CLI:

hduser@Hadoop:/usr/local/apache-hive-2.1.0-bin/conf$ cd /usr/local/apache-hive-2.1.0-bin/bin
hduser@Hadoop:/usr/local/apache-hive-2.1.0-bin/bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/usr/local/apache-hive-2.1.0-bin/lib/hive-common-2.1.0.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive>

Pour afficher toutes les tables:

```
hive> show tables;
OK
Time taken: 1.813 seconds
hive>
```

Pour sortir de Hive il faut taper **exit**

```
hive> exit ;
```