# HDP 2.3.2

# Mongo 2.6.11

**install MongoDB service as per https://github.com/nikunjness/mongo-ambari**

# IMPORTANT

make sure you change directory to home after completing the mongo-ambari service install

**cd**

# install gradle

**wget https://services.gradle.org/distributions/gradle-2.7-bin.zip**

**unzip gradle-2.7-bin.zip**

**mv gradle-2.7 /opt/**

**export GRADLE\_HOME=/opt/gradle-2.7/bin/**

# download mongo-hadoop

**wget https://github.com/mongodb/mongo-hadoop/archive/master.zip**

**unzip master.zip**

**cd mongo-hadoop-master/**

# compile the connectors, should take between 2-10min

**./gradlew jar**

# copy drivers to one directory

**mkdir ~/drivers**

**cd ~/drivers**

# download mongodb java drivers or build your own

# http://mongodb.github.io/mongo-java-driver/3.0/driver/getting-started/installation-guide/

**wget** [**https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongodb-driver/3.0.4/mongodb-driver-3.0.4.jar**](https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongodb-driver/3.0.4/mongodb-driver-3.0.4.jar)

# or build using this pom

[**https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongodb-driver/3.0.4/mongodb-driver-3.0.4.pom**](https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongodb-driver/3.0.4/mongodb-driver-3.0.4.pom)

**cp ~/mongo-hadoop-master/core/build/libs/mongo-hadoop-core-1.5.0-SNAPSHOT.jar ~/drivers**

**cp ~/mongo-hadoop-master/pig/build/libs/mongo-hadoop-pig-1.5.0-SNAPSHOT.jar ~/drivers**

**cp ~/mongo-hadoop-master/hive/build/libs/mongo-hadoop-hive-1.5.0-SNAPSHOT.jar ~/drivers**

**cp ~/mongo-hadoop-master/spark/build/libs/mongo-hadoop-spark-1.5.0-SNAPSHOT.jar ~/drivers**

**cp ~/mongo-hadoop-master/flume/build/libs/flume-1.5.0-SNAPSHOT.jar ~/drivers**

# copy drivers to hdp libs, needs these on the classpath

**cp -r ~/drivers/\* /usr/hdp/current/hadoop-client/lib/**

# restart services in Ambari

# create local user

**cd**

**sudo -u hdfs hdfs dfs -mkdir /user/root**

**sudo -u hdfs hdfs dfs -chown -R root:hdfs /user/root**

### HIVE ###

# https://www.mongodb.com/blog/post/using-mongodb-hadoop-spark-part-1-introduction-setup

**wget http://www.barchartmarketdata.com/data-samples/mstf.csv**

# load data into mongo

**mongoimport mstf.csv --type csv --headerline -d marketdata -c minibars**

# check data is in mongo

[root@sandbox mongo-tutorial]# mongo

MongoDB shell version: 2.6.11

connecting to: test

> **use marketdata**

switched to db marketdata

> **db.minibars.findOne()**

{

"\_id" : ObjectId("564359756336db32f2b4e8ce"),

"Symbol" : "MSFT",

"Timestamp" : "2009-08-24 09:30",

"Day" : 24,

"Open" : 24.41,

"High" : 24.42,

"Low" : 24.31,

"Close" : 24.31,

"Volume" : 683713

}

> **exit**

# login to beeline

# if you get error jdbc:hive2://localhost:10000 (closed)> Error: Failed to open new session: java.lang.RuntimeException: java.lang.RuntimeException: org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.security.authorize.AuthorizationException): User: hive is not allowed to impersonate root (state=,code=0)

# go to core-site and replace "users" with "\*" for proxyusers for hive group

# make sure jars are copied to hdp libs otherwise will get the error in the jira below <https://jira.mongodb.org/browse/HADOOP-224>

**hdfs dfs -put drivers/\* /tmp/udfs**

**beeline**

**!connect jdbc:hive2://localhost:10000 “” ””**

# create\_mongo\_mapped\_hive\_table.hql

**add jar hdfs://sandbox.hortonworks.com:8020/tmp/udfs/mongo-hadoop-hive-1.5.0-SNAPSHOT.jar;**

**add jar hdfs://sandbox.hortonworks.com:8020/tmp/udfs/mongo-hadoop-core-1.5.0-SNAPSHOT.jar;**

**add jar hdfs://sandbox.hortonworks.com:8020/tmp/udfs/mongodb-driver-3.0.4.jar;**

**DROP TABLE IF EXISTS bars;**

**CREATE EXTERNAL TABLE bars**

**(**

**objectid STRING,**

**Symbol STRING,**

**TS STRING,**

**Day INT,**

**Open DOUBLE,**

**High DOUBLE,**

**Low DOUBLE,**

**Close DOUBLE,**

**Volume INT**

**)**

**STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'**

**WITH SERDEPROPERTIES('mongo.columns.mapping'='{"objectid":"\_id",**

**"Symbol":"Symbol", "TS":"Timestamp", "Day":"Day", "Open":"Open", "High":"High", "Low":"Low", "Close":"Close", "Volume":"Volume"}')**

**TBLPROPERTIES('mongo.uri'='mongodb://localhost:27017/marketdata.minibars');**

# if error Error: Error while processing statement: FAILED: Hive Internal Error: com.sun.jersey.api.client.ClientHandlerException(java.io.IOException: java.net.ConnectException: Connection refused) (state=08S01,code=12)

**shut down all services and restart the Sandbox, hive metastore ports most likely conflicting**

# query the table

# select\_from\_mongo\_mapped\_hive\_table.hql

select \* from bars where bars.volume > 5000000 and bars.volume < 10000000;

+---------------------------+--------------+-------------------+-----------+------------+------------+-----------+-------------+--------------+--+

|       bars.objectid       | bars.symbol  |      bars.ts      | bars.day  | bars.open  | bars.high  | bars.low  | bars.close  | bars.volume  |

+---------------------------+--------------+-------------------+-----------+------------+------------+-----------+-------------+--------------+--+

| 564359756336db32f2b4f1f7  | MSFT         | 2009-08-31 16:00  | 31        | 24.64      | 24.65      | 24.64     | 24.65       | 5209285      |

| 564359756336db32f2b4ff6f  | MSFT         | 2009-09-14 16:00  | 14        | 25.0       | 25.0       | 24.99     | 25.0        | 9574088      |

| 564359756336db32f2b5027d  | MSFT         | 2009-09-16 16:00  | 16        | 25.21      | 25.22      | 25.18     | 25.2        | 7920502      |

| 564359756336db32f2b50eb5  | MSFT         | 2009-09-28 16:00  | 28        | 25.85      | 25.89      | 25.83     | 25.83       | 5487064      |

| 564359756336db32f2b5210a  | MSFT         | 2009-10-16 09:30  | 16        | 26.45      | 26.6       | 26.45     | 26.48       | 5092072      |

| 564359756336db32f2b52902  | MSFT         | 2009-10-23 10:55  | 23        | 28.55      | 28.56      | 28.3      | 28.35       | 5941372      |

| 564359766336db32f2b54721  | MSFT         | 2009-11-20 09:30  | 20        | 29.66      | 29.72      | 29.62     | 29.63       | 6859911      |

| 564359766336db32f2b59cba  | MSFT         | 2010-02-12 16:00  | 12        | 27.94      | 27.94      | 27.93     | 27.93       | 5076037      |

| 564359766336db32f2b5c14f  | MSFT         | 2010-03-19 16:00  | 19        | 29.6       | 29.61      | 29.58     | 29.59       | 8826314      |

| 564359766336db32f2b5cd17  | MSFT         | 2010-03-31 14:08  | 31        | 29.45      | 29.46      | 29.4      | 29.46       | 5314205      |

| 564359766336db32f2b5dccc  | MSFT         | 2010-04-15 16:00  | 15        | 30.87      | 30.87      | 30.87     | 30.87       | 5228182      |

| 564359766336db32f2b5dccd  | MSFT         | 2010-04-16 09:30  | 16        | 30.79      | 30.88      | 30.75     | 30.86       | 6267858      |

| 564359766336db32f2b5de53  | MSFT         | 2010-04-16 16:00  | 16        | 30.68      | 30.7       | 30.67     | 30.67       | 5014677      |

| 564359766336db32f2b5e77d  | MSFT         | 2010-04-26 16:00  | 26        | 31.1       | 31.11      | 31.09     | 31.11       | 5338985      |

| 564359776336db32f2b5fcd0  | MSFT         | 2010-05-14 16:00  | 14        | 28.93      | 28.93      | 28.93     | 28.93       | 5318496      |

| 564359776336db32f2b613b9  | MSFT         | 2010-06-07 16:00  | 7         | 25.3       | 25.31      | 25.29     | 25.29       | 6956406      |

| 564359776336db32f2b616c7  | MSFT         | 2010-06-09 16:00  | 9         | 24.79      | 24.81      | 24.78     | 24.79       | 7953364      |

# order by or any select into won’t work, check status of https://jira.mongodb.org/browse/HADOOP-101

### SPARK ###

# https://databricks.com/blog/2015/03/20/using-mongodb-with-spark.html

**pyspark --jars drivers/mongo-hadoop-spark-1.5.0-SNAPSHOT.jar**

# Paste the following in PySpark shell

# spark\_mongo\_integration.py

# set up parameters for reading from MongoDB via Hadoop input format

config = {"mongo.input.uri": "mongodb://localhost:27017/marketdata.minibars"}

inputFormatClassName = "com.mongodb.hadoop.MongoInputFormat"

# these values worked but others might as well

keyClassName = "org.apache.hadoop.io.Text"

valueClassName = "org.apache.hadoop.io.MapWritable"

# read the 1-minute bars from MongoDB into Spark RDD format

minBarRawRDD = sc.newAPIHadoopRDD(inputFormatClassName, keyClassName, valueClassName, None, None, config)

# configuration for output to MongoDB

config["mongo.output.uri"] = "mongodb://localhost:27017/marketdata.fiveminutebars"

outputFormatClassName = "com.mongodb.hadoop.MongoOutputFormat"

# takes the verbose raw structure (with extra metadata) and strips down to just the pricing data

minBarRDD = minBarRawRDD.values()

minBarRDD.saveAsTextFile("hdfs://sandbox.hortonworks.com:8020/user/root/spark-mongo-output3")

# cat the file in hdfs

**hdfs dfs -cat spark-mongo-output3/part-00000 | head -n 5**

### PIG

# download enron dataset

**wget https://s3.amazonaws.com/mongodb-enron-email/enron\_mongo.tar.bz2**

**bzip2 -d enron\_mongo.tar.bz2**

**tar -xvf enron\_mongo.tar**

# restore database

**mongorestore dump/enron\_mail/messages.bson**

# add user

**mongo**

**use enron\_mail**

**db.createUser(**

**{**

**user: "reportsUser",**

**pwd: "12345678",**

**roles: [**

**{ role: "readWrite", db: "enron\_mail" },**

**{ role: "readWrite", db: "enron\_processed" }**

**]**

**}**

**)**

# query mongodb, select all rows

**use enron\_mail**

**db.messages.find()**

# create new mongodb database

**use enron\_processed**

**db.createUser(**

**{**

**user: "writesUser",**

**pwd: "12345678",**

**roles: [**

**{ role: "readWrite", db: "enron\_processed" }**

**]**

**}**

**)**

> **exit**

# load\_store\_mongodb.pig, MAKE SURE YOU RUN WITH TEZ, WITH MR it’s OVER 15min

**pig -x tez load\_store\_mongodb.pig**

--take data from a mongo database and insert into a new database

**REGISTER drivers/mongodb-driver-3.0.4.jar;**

**REGISTER drivers/mongo-hadoop-core-1.5.0-SNAPSHOT.jar**

**REGISTER drivers/mongo-hadoop-pig-1.5.0-SNAPSHOT.jar**

**set default\_parallel 5**

**set mapred.map.tasks.speculative.execution false**

**set mapred.reduce.tasks.speculative.execution false**

**a = LOAD 'mongodb://reportsUser:12345678@sandbox.hortonworks.com:27017/enron\_mail.messages'**

**USING com.mongodb.hadoop.pig.MongoLoader('id, body, headers:[]', 'id');**

**b = limit a 100;**

**c = filter b by $1 is not null;**

**--dump c;**

**STORE c INTO 'mongodb://writesUser:12345678@sandbox.hortonworks.com:27017/enron\_processed.messages'**

**USING com.mongodb.hadoop.pig.MongoInsertStorage('', '' );**

###################

# load\_store\_bson.pig

# make sure you run this in tez\_local mode, we’re not working with HDFS here.

**pig -x tez\_local load\_store\_bson.pig**

--load mongodb binary messages and output to a new binary mongodb compliant output file on --local filesystem

**REGISTER drivers/mongodb-driver-3.0.4.jar;**

**REGISTER drivers/mongo-hadoop-core-1.5.0-SNAPSHOT.jar**

**REGISTER drivers/mongo-hadoop-pig-1.5.0-SNAPSHOT.jar**

**raw = LOAD 'file:///root/dump/enron\_mail/messages.bson' using com.mongodb.hadoop.pig.BSONLoader('','headers:[]') ;**

**send\_recip = FOREACH raw GENERATE $0#'From' as from, $0#'To' as to;**

**send\_recip\_filtered = FILTER send\_recip BY to IS NOT NULL;**

**send\_recip\_split = FOREACH send\_recip\_filtered GENERATE from as from, FLATTEN(TOKENIZE(to)) as to;**

**send\_recip\_split\_trimmed = FOREACH send\_recip\_split GENERATE from as from, TRIM(to) as to;**

**send\_recip\_grouped = GROUP send\_recip\_split\_trimmed BY (from, to);**

**send\_recip\_counted = FOREACH send\_recip\_grouped GENERATE group, COUNT($1) as count;**

**STORE send\_recip\_counted INTO 'file:///tmp/enron\_result.bson' using com.mongodb.hadoop.pig.BSONStorage;**

# review the output

**head -n 5 /tmp/enron\_result.bson/part-v001-o000-r-00000.bson**

# load messages in mongo format and store in Pig format

**hdfs dfs -put dump/enron\_mail/messages.bson /tmp/**

**pig -x tez load\_store\_bson\_hdfs.pig**

--load messages in mongo format and store in Pig format

**REGISTER drivers/mongodb-driver-3.0.4.jar;**

**REGISTER drivers/mongo-hadoop-core-1.5.0-SNAPSHOT.jar**

**REGISTER drivers/mongo-hadoop-pig-1.5.0-SNAPSHOT.jar**

**raw = LOAD '/tmp/messages.bson' using com.mongodb.hadoop.pig.BSONLoader('','headers:[]') ;**

**send\_recip = FOREACH raw GENERATE $0#'From' as from, $0#'To' as to;**

**send\_recip\_filtered = FILTER send\_recip BY to IS NOT NULL;**

**send\_recip\_split = FOREACH send\_recip\_filtered GENERATE from as from, FLATTEN(TOKENIZE(to)) as to;**

**send\_recip\_split\_trimmed = FOREACH send\_recip\_split GENERATE from as from, TRIM(to) as to;**

**send\_recip\_grouped = GROUP send\_recip\_split\_trimmed BY (from, to);**

**send\_recip\_counted = FOREACH send\_recip\_grouped GENERATE group, COUNT($1) as count;**

**STORE send\_recip\_counted INTO 'hdfs://sandbox.hortonworks.com:8020/tmp/enronoutputpig' using PigStorage();**

# check output

**hdfs dfs -cat /tmp/enronoutputpig/part-v001-o000-r-00000 | head -n 5**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# cleanup

**hdfs dfs -rm -r spark-mongo-output\***

**hdfs dfs -rm -r /tmp/messages.bson**

**hdfs dfs -rm -r /tmp/enronoutputpig**

Code is available at: https://github.com/dbist/hdp-mongo-tutorial