10: Docker Tutorial: Hadoop Big Data services & folders on Cloudera quickstart

Posted on May 26, 2019

You can also install it on VMWare as illustrated on the ⏯ Getting started with BigData on Cloudera.

If you are not familiar with Docker get some hands-on experience at a series of step by step Docker tutorials with Java & Springboot examples.

This tutorial is based on 09: Docker Tutorial: Cloudera on Docker via DockerHub, where Cloudera

300+ Java Interview FAQs

300+ Java FAQs 🔥

16+ Java Key Areas Q&As

150+ Java Architect FAQs

80+ Java Code Quality Q&As

150+ Java Coding Q&As

300+ Big Data Interview FAQs

300+ Big Data FAQs 🔥

Tutorials - Big Data

TUT - ⏯ Starting Big Data

TUT - Starting Spark & Scala

Quickstart gets installed on Docker for **learning purpose**.

```
~/projects/docker-hadoop]$ docker run --hostname=qu
 --privileged=true -t -i -v /Users/arulkumarankumar
p 8888:8888 -p 80:80 -p 7180:7180 cloudera/quickst
```

When the above command runs, you will see all the services that gets started:

zookeeper, journalnode, datanode, namenode, and secondary namenode

```
Starting zookeeper ... STARTED
starting datanode, logging to /var/log/hadoop-hdfs/
Started Hadoop datanode (hadoop-hdfs-datanode):
starting journalnode, logging to /var/log/hadoop-hd
Started Hadoop journalnode:        [  OK  ]
starting namenode, logging to /var/log/hadoop-hdfs/
Started Hadoop namenode:
starting secondarynamenode, logging to /var/log/had
Started Hadoop secondarynamenode:
```

historyserver, nodemanager, resourcemanager,HBase master, rest, thrift, Hive Metastore, Hive Server2, and Sqoop Server

```
Started Hadoop historyserver:
starting nodemanager, logging to /var/log/hadoop-y
Started Hadoop nodemanager:
starting resourcemanager, logging to /var/log/hado
Started Hadoop resourcemanager:
starting master, logging to /var/log/hbase/hbase-h
Started HBase master daemon (hbase-master):
starting rest, logging to /var/log/hbase/hbase-hba
Started HBase rest daemon (hbase-rest):
starting thrift, logging to /var/log/hbase/hbase-h
Started HBase thrift daemon (hbase-thrift):
Starting Hive Metastore (hive-metastore):
Started Hive Server2 (hive-server2):
Starting Sqoop Server:
Sqoop home directory: /usr/lib/sqoop2
Setting SQOOP_HTTP_PORT:     12000
Setting SQOOP_ADMIN_PORT:      12001
```

Spark history-server, HBase regionserver, hue, and Impala

```
Starting Spark history-server (spark-history-serv
Starting Hadoop HBase regionserver daemon: starti
hbase-regionserver.
Starting hue:
Started Impala State Store Server (statestored):

Setting OOZIE_HOME:          /usr/lib/oozie
Sourcing:                    /usr/lib/oozie/bin/o
  setting JAVA_LIBRARY_PATH="$JAVA_LIBRARY_PATH:/

......
Starting Solr server daemon:
Started Impala Catalog Server (catalogd) :
Started Impala Server (impalad):

```

All these services can be viewed via the "**Cloudera Manager**" admin console.

/etc/init.d services

init.d is the sub-directory of **/etc** directory in Linux file system. init.d basically contains the bunch of start/stop/reload/restart/status scripts which are used to control the Hadoop ecosystem daemons whilst the system is running or during boot. If you look at /etc/init.d then you will notice all the scripts for different services

```
[root@quickstart /]# cd /etc/init.d
[root@quickstart init.d]# ls
atd                             hadoop-httpfs
cloudera-quickstart-init        hadoop-mapreduce-h
cloudera-scm-agent              hadoop-yarn-nodema
cloudera-scm-server             hadoop-yarn-proxys
crond                           hadoop-yarn-resour
flume-ng-agent                  halt
```

```
functions                      hbase-master
hadoop-hdfs-datanode           hbase-regionserver
hadoop-hdfs-journalnode        hbase-rest
hadoop-hdfs-namenode           hbase-solr-indexer
hadoop-hdfs-secondarynamenode  hbase-thrift
[root@quickstart init.d]# service hbase-master st
HBase master daemon is running
[root@quickstart init.d]# service mysqld status
mysqld (pid  169) is running...
[root@quickstart init.d]# service impala-server s
Impala Server is running

```

netstat -anp to find ports

mysql runs on port 3306.

```
[root@quickstart init.d]# netstat -anp | grep mys
tcp        0      0 0.0.0.0:3306                0
tcp        0      0 127.0.0.1:3306              1
tcp        0      0 172.17.0.2:3306             1
tcp        0      0 172.17.0.2:3306             1
tcp        0      0 127.0.0.1:3306              1
tcp        0      0 127.0.0.1:3306              1
tcp        0      0 127.0.0.1:3306              1
tcp        0      0 172.17.0.2:3306             1
tcp        0      0 172.17.0.2:3306             1
unix  2      [ ACC ]     STREAM     LISTENING
[root@quickstart init.d]#

```

impalad runs on multiple ports, and **21050 for impalad front-end**, 21000 for impalad impala-shell, 22000 is for back-end, and so on. You can check the Cloudera documentation for further details.

```
[root@quickstart init.d]# netstat -anp | grep imp
tcp        0      0 0.0.0.0:23000               0
tcp        0      0 0.0.0.0:21050               0
tcp        0      0 0.0.0.0:21000               0
tcp        0      0 0.0.0.0:25000               0
tcp        0      0 0.0.0.0:22000               0
tcp        0      0 172.17.0.2:23000            1
tcp        0      0 172.17.0.2:23000            1
tcp        0      0 127.0.0.1:45732             1
unix  3      [ ]         STREAM     CONNECTED
unix  3      [ ]         STREAM     CONNECTED
unix  3      [ ]         STREAM     CONNECTED
unix  3      [ ]         STREAM     CONNECTED
unix  3      [ ]         STREAM     CONNECTED
unix  3      [ ]         STREAM     CONNECTED
unix  3      [ ]         STREAM     CONNECTED
unix  3      [ ]         STREAM     CONNECTED
unix  3      [ ]         STREAM     CONNECTED
unix  2      [ ]         STREAM     CONNECTED

```

ps auxwww to find service run details

```
[root@quickstart init.d]# ps auxwww | grep hive-se
hive      2084  0.4  2.4 793748 234952 ?       Sl
r/log/hive -Dhive.log.file=hive-server2.log -Dhive
.log -Dhadoop.home.dir=/usr/lib/hadoop -Dhadoop.id
tive -Dhadoop.policy.file=hadoop-policy.xml -Djava
doop.util.RunJar /usr/lib/hive/lib/hive-service-1.
root      5616  0.0  0.0 103300  1988 pts/0    S+
[root@quickstart init.d]#

```

Hbase starts a number of services like master, region server, etc.

```
[root@quickstart init.d]# ps auxwww | grep hbase
hbase     1329  0.0  0.0   9256  2424 ?        S
f foreground_start master
hbase     1343  0.5  2.0 3145200 198992 ?      Sl
moryError=kill -9 %p -XX:+UseConcMarkSweepGC -XX:
hbase-hbase-master-quickstart.cloudera.log -Dhbase
library.path=/usr/lib/hadoop/lib/native:/usr/lib/
.hbase.master.HMaster start
hbase     1472  0.0  0.0   9256  2396 ?        S
f foreground_start rest
hbase     1486  0.2  1.3 2955260 128120 ?      Sl
ryError=kill -9 %p -XX:+UseConcMarkSweepGC -Dhbase
-Dhbase.home.dir=/usr/lib/hbase -Dhbase.id.str=hb
r/lib/hbase/lib/native/Linux-amd64-64 -Dhbase.secu
hbase     1695  0.0  0.0   9256  2452 ?        S
f foreground_start thrift
hbase     1711  0.2  1.5 2963200 148116 ?      Sl
moryError=kill -9 %p -XX:+UseConcMarkSweepGC -Dhb
log -Dhbase.home.dir=/usr/lib/hbase -Dhbase.id.st
:/usr/lib/hbase/lib/native/Linux-amd64-64 -Dhbase
hbase     2659  0.0  0.0   9256  2420 ?        S
f foreground_start regionserver
```

```
hbase     2673  0.5  1.9 3129228 189668 ?      Sl
utOfMemoryError=kill -9 %p -XX:+UseConcMarkSweepG
root      5644  0.0  0.0 103300  2032 pts/0    S+
[root@quickstart init.d]#

```

/var/log folder

This where the log files go.

```
[root@quickstart /]# cd /var/log
[root@quickstart log]# ls
btmp                   flume-ng                 hadoo
cloudera-scm-agent     hadoop-0.20-mapreduce    hadoo
cloudera-scm-server    hadoop-hdfs              hbase
dedup.log              hadoop-httpfs            hbase
dracut.log             hadoop-kms               hive
[root@quickstart log]# cd hive
[root@quickstart hive]# ls -ltr
total 16
-rw-r--r-- 1 hive hive  198 May 26 01:38 hive-met
-rw-r--r-- 1 hive hive  167 May 26 01:38 hive-ser
-rw-r--r-- 1 hive hive  439 May 26 01:38 hive-met
-rw-r--r-- 1 hive hive 1128 May 26 01:39 hive-ser
[root@quickstart hive]#

```

Java & Python versions

```
[root@quickstart /]# java -version
java version "1.7.0_67"
Java(TM) SE Runtime Environment (build 1.7.0_67-b
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b0
[root@quickstart /]# which java
/usr/bin/java
[root@quickstart /]# python --version
Python 2.6.6
[root@quickstart cloudera]# which python
/usr/bin/python

```

/usr/bin folder

```
[root@quickstart lib]# ls -ltr /usr/bin | grep imp
-rwxr-xr-x    1 root root       1856 Mar 23  2016 im
```

```
-rwxr-xr-x   1 root root     11030 Mar 23  2016 imp
[root@quickstart lib]#

```

/usr/lib folder

```
[root@quickstart lib]# ls -ltr /usr/lib
....
[root@quickstart lib]# ls -ltr /usr/lib/hive/lib
.....
lrwxrwxrwx  1 root root      38 Apr  6  2016 accumu
lrwxrwxrwx  1 root root      37 Apr  6  2016 accumu
lrwxrwxrwx  1 root root      37 Apr  6  2016 accumu
lrwxrwxrwx  1 root root      40 Apr  6  2016 mysql-

```

/usr/jars folder

All the jars used above in "**/usr/lib/…**"

```
[root@quickstart lib]# ls -ltr /usr/jars
```

/var/run/ or /run folder

Run-time variable data. You can get the "**pid**" (i.e process id). You will also know what services are running.

```
[root@quickstart cloudera]# ls -ltr /var/run/hadoop
total 16
-rw-r--r-- 1 hdfs hdfs 4 May 26 09:08 hadoop-hdfs-
srw-rw-rw- 1 hdfs hdfs 0 May 26 09:08 dn.50010
-rw-r--r-- 1 hdfs hdfs 4 May 26 09:08 hadoop-hdfs-
-rw-r--r-- 1 hdfs hdfs 4 May 26 09:09 hadoop-hdfs-
-rw-r--r-- 1 hdfs hdfs 4 May 26 09:09 hadoop-hdfs-

```

```
[root@quickstart cloudera]# ls -ltr /var/run/impala
total 12
-rw-r--r-- 1 impala impala 5 May 26 09:10 statesto
-rw-r--r-- 1 impala impala 5 May 26 09:10 catalogd
-rw-r--r-- 1 impala impala 5 May 26 09:10 impalad-
```

```
[root@quickstart cloudera]#

```

Examples from Cloudera quickstart

The jar shown below has a number of examples, and you can test your environment by running the MapReduce job as shown.

```
[root@quickstart /]# find / -name hadoop-mapreduce
/usr/lib/hadoop-mapreduce/hadoop-mapreduce-example
```

The jobs that are available in hadoop-mapreduce-example.jar:

```
 .......
     pgd.addClass("wordcount", WordCount.class,
                  "A map/reduce program that cou
     pgd.addClass("wordmean", WordMean.class,
                  "A map/reduce program that cou
     pgd.addClass("wordmedian", WordMedian.class
                  "A map/reduce program that cou
     pgd.addClass("wordstandarddeviation", WordS
                  "A map/reduce program that cou
     pgd.addClass("aggregatewordcount", Aggregat
                  "An Aggregate based map/reduce
     pgd.addClass("aggregatewordhist", Aggregate
                  "An Aggregate based map/reduce
     pgd.addClass("grep", Grep.class,
                  "A map/reduce program that cou
     pgd.addClass("randomwriter", RandomWriter.c
                  "A map/reduce program that wri
     pgd.addClass("randomtextwriter", RandomText
     "A map/reduce program that writes 10GB of r
     pgd.addClass("sort", Sort.class, "A map/red

     pgd.addClass("pi", QuasiMonteCarlo.class, Q
     pgd.addClass("bbp", BaileyBorweinPlouffe.cl
     pgd.addClass("distbbp", DistBbp.class, Dist

     pgd.addClass("pentomino", DistributedPentom
     "A map/reduce tile laying program to find s
     pgd.addClass("secondarysort", SecondarySort
                  "An example defining a seconda
     pgd.addClass("sudoku", Sudoku.class, "A sud
     pgd.addClass("join", Join.class, "A job tha
      pgd.addClass("multifilewc", MultiFileWordCou
      pgd.addClass("dbcount", DBCountPageView.clas
      pgd.addClass("teragen", TeraGen.class, "Gene
      pgd.addClass("terasort", TeraSort.class, "Ru
      pgd.addClass("teravalidate", TeraValidate.cl
      exitCode = pgd.run(argv);
......

```

Run a mapreduce job

Running the “pi” example **MapReduce** job:

```
[root@quickstart /]# sudo -u hdfs hadoop jar \
> /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examp
> pi 10 100
Number of Maps  = 10
Samples per Map = 100
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Wrote input for Map #5
Wrote input for Map #6
Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
Starting Job
19/05/26 03:27:00 INFO client.RMProxy: Connecting
19/05/26 03:27:01 INFO input.FileInputFormat: Tota
19/05/26 03:27:01 INFO mapreduce.JobSubmitter: num
19/05/26 03:27:01 INFO mapreduce.JobSubmitter: Sub
19/05/26 03:27:01 INFO impl.YarnClientImpl: Submit
19/05/26 03:27:01 INFO mapreduce.Job: The url to t
/
19/05/26 03:27:01 INFO mapreduce.Job: Running job
19/05/26 03:27:08 INFO mapreduce.Job: Job job_1558
19/05/26 03:27:08 INFO mapreduce.Job:  map 0% redu
19/05/26 03:27:14 INFO mapreduce.Job:  map 10% red
19/05/26 03:27:15 INFO mapreduce.Job:  map 20% red
19/05/26 03:27:16 INFO mapreduce.Job:  map 30% red
19/05/26 03:27:18 INFO mapreduce.Job:  map 40% red
19/05/26 03:27:19 INFO mapreduce.Job:  map 50% red
19/05/26 03:27:20 INFO mapreduce.Job:  map 60% red
19/05/26 03:27:21 INFO mapreduce.Job:  map 70% red
19/05/26 03:27:23 INFO mapreduce.Job:  map 80% red
19/05/26 03:27:24 INFO mapreduce.Job:  map 100% re
19/05/26 03:27:25 INFO mapreduce.Job:  map 100% re
19/05/26 03:27:25 INFO mapreduce.Job: Job job_1558
19/05/26 03:27:25 INFO mapreduce.Job: Counters: 49
```

```
        File System Counters
                FILE: Number of bytes read=226
                FILE: Number of bytes written=125
                FILE: Number of read operations=0
                FILE: Number of large read operat
                FILE: Number of write operations=
                HDFS: Number of bytes read=2730
                HDFS: Number of bytes written=215
                HDFS: Number of read operations=4
                HDFS: Number of large read operat
                HDFS: Number of write operations=
        Job Counters
                Launched map tasks=10
                Launched reduce tasks=1
                Data-local map tasks=10
                Total time spent by all maps in o
                Total time spent by all reduces i
                Total time spent by all map tasks
                Total time spent by all reduce ta
                Total vcore-seconds taken by all
                Total vcore-seconds taken by all
                Total megabyte-seconds taken by a
                Total megabyte-seconds taken by a
        Map-Reduce Framework
                Map input records=10
                Map output records=20
                Map output bytes=180
                Map output materialized bytes=280
                Input split bytes=1550
                Combine input records=0
                Combine output records=0
                Reduce input groups=2
                Reduce shuffle bytes=280
                Reduce input records=20
                Reduce output records=0
                Spilled Records=40
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=308
                CPU time spent (ms)=4690
                Physical memory (bytes) snapshot=
                Virtual memory (bytes) snapshot=1
                Total committed heap usage (bytes
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1180
        File Output Format Counters
                Bytes Written=97
```

```
Job Finished in 24.374 seconds
Estimated value of Pi is 3.14800000000000000000
[root@quickstart /]#

```

Do you want to open multiple terminal windows?

```
$ docker ps
CONTAINER ID        IMAGE                   COMMAND
bf645c6a2930        cloudera/quickstart     "/usr/bi

```

```
$ docker exec -it bf645c6a2930 /bin/bash
[root@quickstart /]#

```

How to stop the conatiner?

```
$ docker ps
CONTAINER ID        IMAGE                   COMMAND
bf645c6a2930        cloudera/quickstart     "/usr/bi

```

```
$ docker stop bf645c6a2930

```

What is next?

In the next post let's look at the CLIs like impala-shell, hdfs, hive, spark-shell, pyspark, etc.

‹ 09: Docker Tutorial: Getting started with Hadoop Big Data on Cloudera quickstart

11: Docker Tutorial: Hadoop Big Data CLIs on Cloudera quickstart ›

Disclaimer