**Installing CDH (Cloudera Distribution Including Apache Hadoop) 5.5 on Mac OSX**

1. **Dependencies**
2. Install the Java version that is supported for the CDH version you are installing.
3. Enable ssh on your mac by turning on remote login. You can find this option under your toolbar’s System Preferences > Sharing.
4. Check the box for Remote Login to enable the service.
5. Allow access for: “Only these users: Administrators”

c. Enable password-less ssh login to localhost for MRv1 and HBase.

1. Generate an rsa or dsa key.
   1. ssh-keygen -t rsa -P ""
   2. Continue through the key generator prompts (use default options).
2. Test: ssh localhost

**2. CDH**

The CDH tarballs are very nicely packaged and easily [downloadable](http://blog.cloudera.com//download) from Cloudera’s repository. Download and explode the tarballs in a lib directory where you can manage latest versions with a simple symlink as the following. Although Mac OSX’s “Make Alias” feature is bi-directional, do not use it, but instead use your command-line ln -s command, such as ln -s source\_file target\_file.

* /Users/jordanh/cloudera/
* cdh5.1/
* hadoop -> /Users/jordanh/cloudera/lib/hadoop-2.3.0-cdh5.1.0
* hbase -> /Users/jordanh/cloudera/lib/hbase-0.98.1-cdh5.1.0
* hive -> /Users/jordanh/cloudera/lib/hive-0.12.0-cdh5.1.0
* zookeeper -> /Users/jordanh/cloudera/lib/zookeeper-3.4.5-cdh4.7.0
* ops/
* dn
* logs/hadoop, logs/hbase, logs/yarn
* nn/
* pids
* tmp/
* zk/

Now, set your environment properties according to the paths where you’ve exploded your tarballs.

CDH="cdh5.1"

export HADOOP\_HOME="/Users/jordanh/cloudera/${CDH}/hadoop"

export HBASE\_HOME="/Users/jordanh/cloudera/${CDH}/hbase"

export HIVE\_HOME="/Users/jordanh/cloudera/${CDH}/hive"

export HCAT\_HOME="/Users/jordanh/cloudera/${CDH}/hive/hcatalog"

export PATH=${JAVA\_HOME}/bin:${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin:${ZK\_HOME}/bin:${HBASE\_HOME}/bin:${HIVE\_HOME}/bin:${HCAT\_HOME}/bin:${M2\_HOME}/bin:${ANT\_HOME}/bin:${PATH}

Update your main Hadoop configuration files, as shown in the sample files below.

1. $HADOOP\_HOME/etc/hadoop/core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:8020</value>

<description>The name of the default file system. A URI whose

scheme and authority determine the FileSystem implementation. The

uri's scheme determines the config property (fs.SCHEME.impl) naming

the FileSystem implementation class. The uri's authority is used to

determine the host, port, etc. for a filesystem.</description>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/Users/jordanh/cloudera/ops/tmp/hadoop-${user.name}</value>

<description>A base for other temporary directories.</description>

</property>

<property>

<name>io.compression.codecs</name>

<value>org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.BZip2Codec,org.apache.hadoop.io.compress.SnappyCodec</value>

<description>A comma-separated list of the compression codec classes that can

be used for compression/decompression. In addition to any classes specified

with this property (which take precedence), codec classes on the classpath

are discovered using a Java ServiceLoader.</description>

</property>

</configuration>

1. $HADOOP\_HOME/etc/hadoop/hdfs-site.xml

<configuration>

<property>

<name>dfs.namenode.name.dir</name>

<value>/Users/jordanh/cloudera/ops/nn</value>

<description>Determines where on the local filesystem the DFS name node

should store the name table(fsimage). If this is a comma-delimited list

of directories then the name table is replicated in all of the

directories, for redundancy. </description>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/Users/jordanh/cloudera/ops/dn/</value>

<description>Determines where on the local filesystem an DFS data node

should store its blocks. If this is a comma-delimited

list of directories, then data will be stored in all named

directories, typically on different devices.

Directories that do not exist are ignored.

</description>

</property>

<property>

<name>dfs.datanode.http.address</name>

<value>localhost:50075</value>

<description>

The datanode http server address and port.

If the port is 0 then the server will start on a free port.

</description>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication.

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

</description>

</property>

</configuration>

3. $HADOOP\_HOME/etc/hadoop/yarn-site.xml

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

<description>the valid service name should only contain a-zA-Z0-9\_ and can not start with numbers</description>

</property>

<property>

<name>yarn.log-aggregation-enable</name>

<value>true</value>

<description>Whether to enable log aggregation</description>

</property>

<property>

<name>yarn.nodemanager.remote-app-log-dir</name>

<value>hdfs://localhost:8020/tmp/yarn-logs</value>

<description>Where to aggregate logs to.</description>

</property>

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>8192</value>

<description>Amount of physical memory, in MB, that can be allocated

for containers.</description>

</property>

<property>

<name>yarn.nodemanager.resource.cpu-vcores</name>

<value>4</value>

<description>Number of CPU cores that can be allocated

for containers.</description>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-mb</name>

<value>1024</value>

<description>The minimum allocation for every container request at the RM,

in MBs. Memory requests lower than this won't take effect,

and the specified value will get allocated at minimum.</description>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb</name>

<value>2048</value>

<description>The maximum allocation for every container request at the RM,

in MBs. Memory requests higher than this won't take effect,

and will get capped to this value.</description>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-vcores</name>

<value>1</value>

<description>The minimum allocation for every container request at the RM,

in terms of virtual CPU cores. Requests lower than this won't take effect,

and the specified value will get allocated the minimum.</description>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-vcores</name>

<value>2</value>

<description>The maximum allocation for every container request at the RM,

in terms of virtual CPU cores. Requests higher than this won't take effect,

and will get capped to this value.</description>

</property>

</configuration>

4. $HADOOP\_HOME/etc/hadoop/mapred-site.xml

<configuration>

<property>

<name>mapreduce.jobtracker.address</name>

<value>localhost:8021</value>

</property>

<property>

<name>mapreduce.jobhistory.done-dir</name>

<value>/tmp/job-history/</value>

<description></description>

</property>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

<description>The runtime framework for executing MapReduce jobs.

Can be one of local, classic or yarn.

</description>

</property>

<property>

<name>mapreduce.map.cpu.vcores</name>

<value>1</value>

<description>

The number of virtual cores required for each map task.

</description>

</property>

<property>

<name>mapreduce.reduce.cpu.vcores</name>

<value>1</value>

<description>

The number of virtual cores required for each reduce task.

</description>

</property>

<property>

<name>mapreduce.map.memory.mb</name>

<value>1024</value>

<description>Larger resource limit for maps.</description>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>1024</value>

<description>Larger resource limit for reduces.</description>

</property>

<property>

<name>mapreduce.map.java.opts</name>

<value>-Xmx768m</value>

<description>Heap-size for child jvms of maps.</description>

</property>

<property>

<name>mapreduce.reduce.java.opts</name>

<value>-Xmx768m</value>

<description>Heap-size for child jvms of reduces.</description>

</property>

<property>

<name>yarn.app.mapreduce.am.resource.mb</name>

<value>1024</value>

<description>The amount of memory the MR AppMaster needs.</description>

</property>

</configuration>

5. $HADOOP\_HOME/etc/hadoop/hadoop-env.sh (indicated properties only)

# Where log files are stored. $HADOOP\_HOME/logs by default.

export HADOOP\_LOG\_DIR="/Users/jordanh/cloudera/ops/logs/hadoop"

export YARN\_LOG\_DIR="/Users/jordanh/cloudera/ops/logs/yarn"

# The directory where pid files are stored when processes run as daemons. /tmp by default.

export HADOOP\_PID\_DIR="/Users/jordanh/cloudera/ops/pids"

export YARN\_PID\_DIR=${HADOOP\_PID\_DIR}

6. $HBASE\_HOME/conf/hbase-site.xml

<configuration>

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

<description>The mode the cluster will be in. Possible values are

false for standalone mode and true for distributed mode. If

false, startup will run all HBase and ZooKeeper daemons together

in the one JVM.

</description>

</property>

<property>

<name>hbase.tmp.dir</name>

<value>/Users/jordanh/cloudera/ops/tmp/hbase-${user.name}</value>

<description>Temporary directory on the local filesystem.

Change this setting to point to a location more permanent

than '/tmp' (The '/tmp' directory is often cleared on

machine restart).

</description>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/Users/jordanh/cloudera/ops/zk</value>

<description>Property from ZooKeeper's config zoo.cfg.

The directory where the snapshot is stored.

</description>

</property>

s<property>

<name>hbase.rootdir</name>

<value>hdfs://localhost:8020/hbase</value>

<description>The directory shared by region servers and into

which HBase persists. The URL should be 'fully-qualified'

to include the filesystem scheme. For example, to specify the

HDFS directory '/hbase' where the HDFS instance's namenode is

running at namenode.example.org on port 9000, set this value to:

hdfs://namenode.example.org:9000/hbase. By default HBase writes

into /tmp. Change this configuration else all data will be lost

on machine restart.

</description>

</property>

</configuration>

7. $HBASE\_HOME/conf/hbase-env.sh

# Where log files are stored. $HBASE\_HOME/logs by default.

# Where log files are stored. $HBASE\_HOME/logs by default.

export HBASE\_LOG\_DIR="/Users/jordanh/cloudera/ops/logs/hbase"

# The directory where pid files are stored. /tmp by default.

export HBASE\_PID\_DIR="/Users/jordanh/cloudera/ops/pids"

# Tell HBase whether it should manage its own instance of Zookeeper or not.

export HBASE\_MANAGES\_ZK=true

**3. Putting it all together**

By now, we should have accomplished setting up HDFS, YARN, and HBase.

These are the bare essentials for getting your local machine ready for running MapReduce jobs and building applications on HBase. In the next few steps, we will start/stop the services and provide examples to ensure each service is operating correctly. The steps are listed in the specific order for initialization in order to adhere to dependencies. The order could be reversed for halting the services.

##### **Service HDFSs**

*NameNode*

format: hdfs namenode -format

start: hdfs namenode

stop: Ctrl-C

url: <http://localhost:50070/dfshealth.html>

*DataNode*

start: hdfs datanode

stop: Ctrl-C

url: <http://localhost:50075/browseDirectory.jsp?dir=%2F&nnaddr=127.0.0.1:8020>

*Test*

hadoop fs -mkdir /tmp

hadoop fs -put /path/to/local/file.txt /tmp/

hadoop fs -cat /tmp/file.txt

##### **Service YARN**

*ResourceManager*

start: yarn resourcemanager

stop: Ctrl-C

url: <http://localhost:8088/cluster>

*NodeManager*

start: yarn nodemanager

stop: Ctrl-C

url: <http://localhost:8042/node>

**4. Running the program on Hadoop**

In terminal, we use “hadoop jar” command to run our jar on the hadoop system.

For this, we create an “input” folder using hdfs dfs -mkdir input command where we put our input data file.

To run the jar file we do the following:

hadoop jar /Users/najus/MUM/Big\ Data/workspace/Crystalball/target/crystalball-0.0.1-SNAPSHOT.jar edu.mum.wordcount.WordCount /input/sample.txt /output