For running a fully-distributed operation on more than one host, make the following configurations. In hbase-site.xml, add the property hbase.cluster.distributed and set it to true and point the HBase hbase.rootdir at the appropriate HDFS NameNode and location in HDFS where you would like HBase to write data. For example, if you namenode were running at namenode.example.org on port 8020 and you wanted to home your HBase in HDFS at /hbase, make the following configuration.

```
<configuration>
  ...
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://namenode.example.org:8020/hbase</value>
    <description>The directory shared by RegionServers.
    </description>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
    <description>The mode the cluster will be in. Possible values are
      false: standalone and pseudo-distributed setups with managed Zookeeper
      true: fully-distributed with unmanaged Zookeeper Quorum (see hbase-env.sh)
    </description>
  </property>
  ...
</configuration>
```

In addition, a fully-distributed mode requires that you modify conf/regionservers. The Section 2.7.1.2, “regionservers” file lists all hosts that you would have running HRegionServers, one host per line (This file in HBase is like the Hadoop slaves file). All servers listed in this file will be started and stopped when HBase cluster start or stop is run.

Of note, if you have made *HDFS client configuration* on your Hadoop cluster -- i.e. configuration you want HDFS clients to use as opposed to server-side configurations -- HBase will not see this configuration unless you do one of the following:

- Add a pointer to your HADOOP_CONF_DIR to the HBASE_CLASSPATH environment variable in hbase-env.sh.
- Add a copy of hdfs-site.xml (or hadoop-site.xml) or, better, symlinks, under ${HBASE_HOME}/conf, or
- if only a small set of HDFS client configurations, add them to hbase-site.xml.

An example of such an HDFS client configuration is dfs.replication. If for example, you want to run with a replication factor of 5, hbase will create files with the default of 3 unless you do the above to make the configuration available to HBase.

Make sure HDFS is running first. Start and stop the Hadoop HDFS daemons by running bin/start-hdfs.sh over in theHADOOP_HOME directory. You can ensure it started properly by testing the **put** and **get** of files into the Hadoop filesystem. HBase does not normally use the mapreduce daemons. These do not need to be started.
If you are managing your own ZooKeeper, start it and confirm its running else, HBase will start up ZooKeeper for you as part of its start process.
Start HBase with the following command:

```
bin/start-hbase.sh
```

Run the above from the HBASE_HOME directory.
You should now have a running HBase instance. HBase logs can be found in the logs subdirectory. Check them out especially if HBase had trouble starting.
HBase also puts up a UI listing vital attributes. By default its deployed on the Master host at port 60010 (HBase RegionServers listen on port 60020 by default and put up an informational http server at 60030). If the Master were running on a host namedmaster.example.org on the default port, to see the Master's homepage you'd point your browser athttp://master.example.org:60010.

A distributed HBase depends on a running ZooKeeper cluster. All participating nodes and clients need to be able to access the running ZooKeeper ensemble. HBase by default manages a ZooKeeper "cluster" for you. It will start and stop the ZooKeeper ensemble as part of the HBase start/stop process. You can also manage the ZooKeeper ensemble independent of HBase and just point HBase at the cluster it should use. To toggle HBase management of ZooKeeper, use the HBASE_MANAGES_ZK variable inconf/hbase-env.sh. This variable, which defaults to true, tells HBase whether to start/stop the ZooKeeper ensemble servers as part of HBase start/stop.

When HBase manages the ZooKeeper ensemble, you can specify ZooKeeper configuration using its native zoo.cfg file, or, the easier option is to just specify ZooKeeper options directly in conf/hbase-site.xml. A ZooKeeper configuration option can be set as a property in the HBase hbase-site.xml XML configuration file by prefacing the ZooKeeper option name withhbase.zookeeper.property. For example, the clientPort setting in ZooKeeper can be changed by setting thehbase.zookeeper.property.clientPort property. For all default values used by HBase, including ZooKeeper configuration, see Section 2.6.1.1, “HBase Default Configuration”. Look for the hbase.zookeeper.property prefix [13]

You must at least list the ensemble servers in hbase-site.xml using the hbase.zookeeper.quorum property. This property defaults to a single ensemble member at localhost which is not suitable for a fully distributed HBase. (It binds to the local machine only and remote clients will not be able to connect).

For example, to have HBase manage a ZooKeeper quorum on nodes *rs{1,2,3,4,5}.example.com*, bound to port 2222 (the default is 2181) ensure HBASE_MANAGE_ZK is commented out or set to true in conf/hbase-env.sh and then edit conf/hbase-site.xmland set hbase.zookeeper.property.clientPort and hbase.zookeeper.quorum. You should also sethbase.zookeeper.property.dataDir to other than the default as the default has ZooKeeper persist data under /tmp which is often cleared on system restart. In the example below we have ZooKeeper persist to /user/local/zookeeper.

```
<configuration>
  ...
  <property>
    <name>hbase.zookeeper.property.clientPort</name>
    <value>2222</value>
    <description>Property from ZooKeeper's config zoo.cfg.
    The port at which the clients will connect.
    </description>
  </property>
  <property>
    <name>hbase.zookeeper.quorum</name>
```

```
<value>rs1.example.com,rs2.example.com,rs3.example.com,rs4.example.com,rs5.example.com</value>
    <description>Comma separated list of servers in the ZooKeeper Quorum.
    For example, "host1.mydomain.com,host2.mydomain.com,host3.mydomain.com".
    By default this is set to localhost for local and pseudo-distributed modes
    of operation. For a fully-distributed setup, this should be set to a full
    list of ZooKeeper quorum servers. If HBASE_MANAGES_ZK is set in hbase-env.sh
    this is the list of servers which we will start/stop ZooKeeper on.
    </description>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/usr/local/zookeeper</value>
    <description>Property from ZooKeeper's config zoo.cfg.
    The directory where the snapshot is stored.
    </description>
  </property>
  ...
</configuration>
```

1. Create a configuration file. This file can be called anything. Use the following settings as a starting point:
2.
```
tickTime=2000
dataDir=/var/lib/zookeeper/
clientPort=2181
initLimit=5
syncLimit=2
server.1=zoo1:2888:3888
server.2=zoo2:2888:3888
server.3=zoo3:2888:3888
```
3. You can find the meanings of these and other configuration settings in the section Configuration Parameters. A word though about a few here:
4. Every machine that is part of the ZooKeeper ensemble should know about every other machine in the ensemble. You accomplish this with the series of lines of the form **server.id=host:port:port**. The parameters **host** and **port** are straightforward. You attribute the server id to each machine by creating a file named `myid`, one for each server, which resides in that server's data directory, as specified by the configuration file parameter **dataDir**.
5. The myid file consists of a single line containing only the text of that machine's id. So `myid` of server 1 would contain the text "1" and nothing else. The id must be unique within the ensemble and should have a value between 1 and 255.
1. If your configuration file is set up, you can start a ZooKeeper server:

2. `$ java -cp zookeeper.jar:lib/slf4j-api-1.6.1.jar:lib/slf4j-log4j12-1.6.1.jar:lib/log4j-1.2.15.jar:conf \ org.apache.zookeeper.server.quorum.QuorumPeerMain zoo.cfg`
3. QuorumPeerMain starts a ZooKeeper server, JMX management beans are also registered which allows management through a JMX management console. The ZooKeeper JMX document contains details on managing ZooKeeper with JMX.
4. See the script *bin/zkServer.sh*, which is included in the release, for an example of starting server instances.
5. Test your deployment by connecting to the hosts:
   In Java, you can run the following command to execute simple operations:
   `$ java -cp zookeeper.jar:lib/slf4j-api-1.6.1.jar:lib/slf4j-log4j12-1.6.1.jar:lib/log4j-1.2.15.jar:conf:src/java/lib/jline-0.9.94.jar \ org.apache.zookeeper.ZooKeeperMain -server 127.0.0.1:2181`
   In C, you can compile either the single threaded client or the multithreaded client: or n the c subdirectory in the ZooKeeper sources. This compiles the single threaded client:
   `$ make cli_st`
   And this compiles the mulithreaded client:
   `$ make cli_mt`
6. Running either program gives you a shell in which to execute simple file-system-like operations. To connect to ZooKeeper with the multithreaded client, for example, you would run:
7. `$ cli_mt 127.0.0.1:2181`