# AN INTRODUCTION TO HADOOP ADMINISTRATION



Here we explain some of the most common Hadoop administrative tasks. There are many, so we only talk about some of the main ones. The reader is encouraged to consult the Apache Hadoop documentation to dig more deeply into each topic.

As you work through some admin commands and tasks, you should know that each version of Hadoop is slightly different. They tend to change some of the command script names. In this example we are using Hadoop 2.7.3.

You will need a Hadoop cluster setup to work through this material. Follow our instructions here on how to set up a cluster. It is not enough to run a local-only Hadoop installation if you want to learn some admin tasks.

*(This article is part of our Hadoop Guide. Use the right-hand menu to navigate.)*

## Common admin tasks

Here are some of common admin tasks:

- Monitor health of cluster
- Add new data nodes as needed
- Optionally turn on security
- Optionally turn on encryption
- Recommended, but optional, to turn on high availability
- Optional to turn on MapReduce Job History Tracking Server
- Fix corrupt data blocks when necessary

- Tune performance

We discuss some of these tasks below.

# Turn on security

By default Hadoop is set up with no security. To run Hadoop in secure mode each user and service authentications with Kerberos. Kerberos is built into Windows and is easily added to Linux.

As for Hadoop itself, the nodes uses RPC (remote procedure calls) to execute commands on other servers. You can set dfs.encrypt.data.transfer and hadoop.rpc.protection to encrypt data transfer and remote procedure calls.

To encrypt data at rest the admin would need to set up an Encryption Key, HDFS Encryption Zone, and Ranger Key Manager Services together with setting up users and roles.

# Hadoop web interface URLs

The most common URLs you use with Hadoop are:

| | |
|---|---|
| NameNode | http://localhost:50070 |
| Yarn Resource Manager | http://localhost:8088 |
| MapReduce JobHistory Server | http://localhost:19888 |

These screens are shown below.

## NameNode Main Screen

# Overview 'hadoop-master:9000' (active)

| | |
|---|---|
| **Started:** | Wed Apr 12 11:10:49 CLST 2017 |
| **Version:** | 2.7.3, rbaa91f7c6bc9cb92be5982de4719c1c8af91ccff |
| **Compiled:** | 2016-08-18T01:41Z by root from branch-2.7.3 |
| **Cluster ID:** | CID-5338ba98-48b4-4147-b251-172a075cb629 |
| **Block Pool ID:** | BP-1020999087-127.0.0.1-1490389530681 |

# Summary

Security is off.

Safe mode is ON. The reported blocks 5 needs additional 48 blocks to reach the threshold 0.9990 of total blocks 53. The number of live datanodes 2 has reached the minimum number 0. Safe mode will be turned off automatically once the thresholds have been reached.

134 files and directories, 53 blocks = 187 total filesystem object(s).

Heap Memory used 42 MB of 253.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 41.89 MB of 42.53 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

| | |
|---|---|
| **Configured Capacity:** | 172.24 GB |
| **DFS Used:** | 6.09 MB (0%) |
| **Non DFS Used:** | 13.42 GB |
| **DFS Remaining:** | 158.81 GB (92.2%) |
| **Block Pool Used:** | 6.09 MB (0%) |
| **DataNodes usages% (Min/Median/Max/stdDev):** | 0.00% / 0.01% / 0.01% / 0.00% |
| **Live Nodes** | 2 (Decommissioned: 0) |
| **Dead Nodes** | 0 (Decommissioned: 0) |
| **Decommissioning Nodes** | 0 |
| **Total Datanode Volume Failures** | 0 (0 B) |
| **Number of Under-Replicated Blocks** | 0 |
| **Number of Blocks Pending Deletion** | 0 |
| **Block Deletion Start Time** | 4/12/2017, 11:10:49 AM |

# NameNode Journal Status

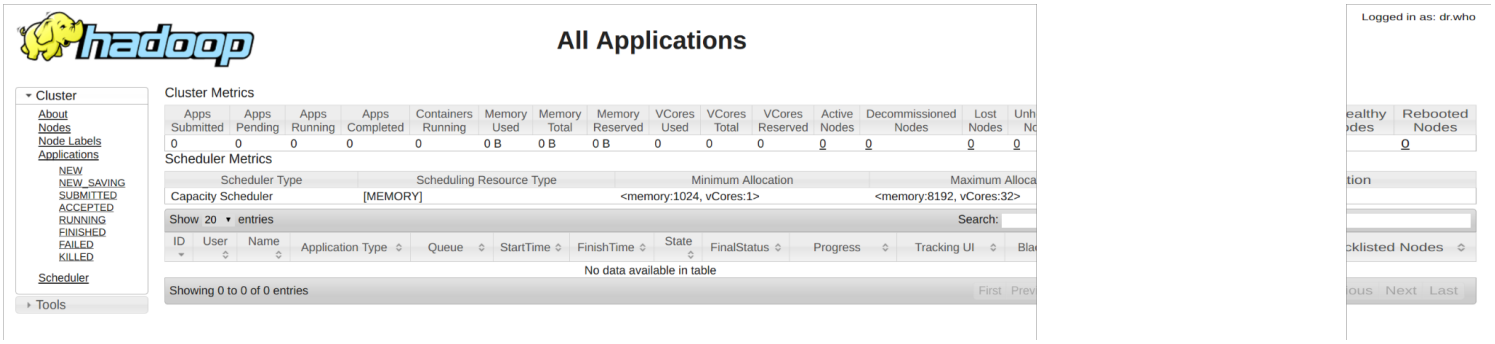
**Current transaction ID:** 6152

| Journal Manager | State |
|---|---|
| FileJournalManager(root=/usr/local/hadoop) | EditLogFileOutputStream(/usr/local/hadoop/current/edits_inprogress_0000000000000006152) |

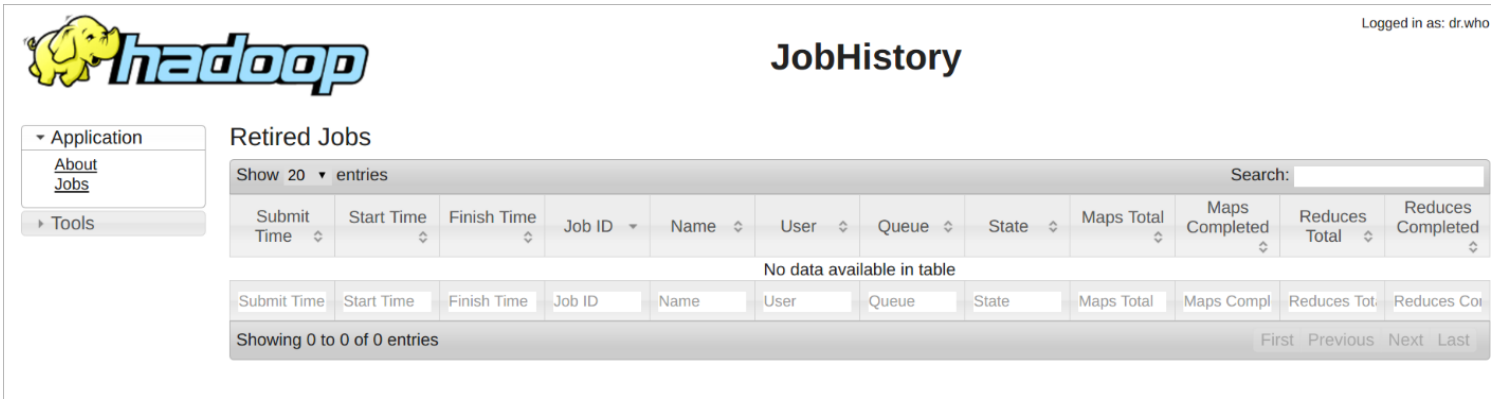
# NameNode Storage

| Storage Directory | Type | State |
|---|---|---|
| /usr/local/hadoop | IMAGE_AND_EDITS | Active |

# Yarn Resource Manager




# MapReduce Job History Server




# Configure high availability

High Availability sets two two redundant NameNodes in an Active/Passive configuration with a hot standby. Without this, if the NameNode crashes the the cluster cannot be used until the NameNode is recovered. With HA the administrator can fail over to the 2nd NameNode in the case of a failure.

Note that the SecondaryNameNode that runs on the cluster master is not a HA NameNode server. The primary and secondary NameNode servers work together, so the secondary cannot be used as a failover mechanism.

The set up HA you set dfs.nameservices and dfs.ha.namenodes. in hdfs-site.xml as well as their IP address and port an mount an NFS directory between the machines so that they can share a common folder.

You run administrative commands using the CLI:

```
hdfs haadmin
```

# MapReduce job history server

The job history MapReduce server is not installed by default. The configuration and how to start it is shown below.

```
cat /usr/local/hadoop/hadoop-2.7.3//etc/hadoop/mapred-site.xml

<configuration>
<property>
```

```
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>
<property>
<name>mapreduce.jobhistory.address</name>
<value>localhost:10020</value>
</property>
<property>
<name>mapreduce.jobhistory.webapp.address</name>
<value>localhost:19888</value>
</property>
</configuration>
```

Start the MapReduce job history server with the following command:

```
$HADOOP_HOME/sbin/mr-jobhistory-daemon.sh --config
$HADOOP_HOME/etc/hadoop start historyserver
starting historyserver, logging to
/usr/local/hadoop/hadoop-2.7.3//logs/mapred-hadoop-historyserver-hp.out
```

And query it like this:

```
curl http://localhost:19888/ws/v1/history/info

{"historyInfo":{"startedOn":1492004745263,"hadoopVersion":"2.7.3","hadoopBuil
dVersion":"2.7.3 from baa91f7c6bc9cb92be5982de4719c1c8af91ccff by root source
checksum
2e4ce5f957ea4db193bce3734ff29ff4","hadoopVersionBuiltOn":"2016-08-18T01:41Z"}
}
```

Or just login to the webpage.

# Add datanode

You can add a datanode without having to stop Hadoop.

The basic steps are to create the Hadoop user and then configure ssh keys with no passcode so that the user can ssh from one server to another without having to enter a password. Update the /etc/hosts files to add the hostname to all the machines in the cluster. Then you zip up and copy the entire $HADOOP_HOME directory on the master to the same target machine in the same directory.

Then you add the new datanode to $HADOOP_HOME/etc/hadoop/slaves.

Then run this command on the new datanode:

```
hadoop-daemon.sh --config $HADOOP_CONF_DIR --script hdfs start datanode
```

Now you should be able to see it show up when you print the topology:

```
hdfs dfsadmin -printTopology
```

```
192.168.1.83:50010 (hadoop-slave-1)
192.168.1.85:50010 (hadoop-slave-2)
```

# Run Pig Mapreduce job

Here is a Pig script you can run to generate a MapReduce job so that you can have a job to track. (If you do not have pig installed you can refer to https://www.bmc.com/blogs/hadoop-apache-pig/)

First create this file sales.csv

```
Dallas,Jane,20000
Houston,Jim,75000
Houston,Bob,65000
New York,Earl,40000
Dallas,Fred,40000
Dallas,Jane,20000
Houston,Jim,75000
```

You can copy the file onto itself multiple times to create a very large file so you will have a job that will run for a few minutes.

Then copy it from the local file system to Hadoop.

```
hadoop fs -copyFromLocal /data
```

Check that it is there:

```
hadoop fs -cat /user/sales.csv
```

Run Pig. Pig with no command line options runs Pig in cluster (aka MapReduce) mode.

Paste in this script:

```
a = LOAD '/data/sales.csv' USING PigStorage(',') AS
(shop:chararray,employee:chararray,sales:int);

Dump a
Describe a

b = group a by employee;

results = FOREACH b generate SUM(a.sales) as sum, a.employee;
```

Then you can check the different screens for job data.

# Common CLI commands

| | |
|---|---|
| Stop and start Hadoop. | `start_dfs.sh start_yarn.sh` |
| Format HDFS. | `$HADOOP_HOME/bin/hdfs namenode -format` |
| Turn off safe mode. | `hadoop dfsadmin -safemode leave` |

| | |
|---|---|
| List processes. | `jps`<br>`Namenode jobs:`<br>`26961 RunJar`<br>`28916 SecondaryNameNode`<br>`24121 JobHistoryServer`<br>`29403 Jps`<br>`28687 NameNode`<br>`29135 ResourceManager`Datanode jobs;<br>4231 Jps<br>3929 DataNode<br>4077 NodeManager |
| Corrupt data blocks. Find missing blocks. | `hdfs fsck /` |

# Monitoring health of nodemanagers

| | |
|---|---|
| `yarn.nodemanager.health-checker.script.path` | Script path and filename. |
| `yarn.nodemanager.health-checker.script.opts` | Command line options. |
| `Command line options.` | Run frequency |
| `checker.script.interval-ms` | |
| `checker.script.interval-ms` | Timeout. |

# Other common admin tasks

Setup log aggregation.

Configure rack awareness.

Configure load balancing between datanodes.

Upgrade to newer version.

Use cacheadmin to manage Hadoop centralized cache.

Take snapshots.

Configure user permissions and access control.

# Common problems

It is not recommended to use localhost as the URL for the Hadoop file system on the localhost. That will cause it to bind to 127.0.0.1 instead of the machine's routable IP address. Then in Pig you will get this error:

```
pig java.net.connectexception connection refused localhost:9000
```

So set the bind IP address to 0.0.0.0 in etc/hadoop/core-site.xml:

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://0.0.0.0:9000/</value>
```

```
</property>
```

# WebAppProxy server

Setting up the WebAppProxy server is a security issue. You can use it to set up a proxy server between masters and slaves. It blocks users from using the Yarn URL for hacking. The Yarn user has elevated privileges, which is why that is a risk. It throws up a warning is someone accesses it plus it strips cookies that could be used in an attack.

# Where to go from here

The user is encouraged to read further the topics mentioned in this doc and in particular in the Other Common Admin Tasks section as that is where they are going to find tuning and maintenance tools and issues that will certain become issues as they work to maintain a production system and fix all the associated problems.