* [Hadoop Cluster Setup](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Hadoop_Cluster_Setup)
  + [Purpose](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Purpose)
  + [Prerequisites](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Prerequisites)
  + [Installation](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Installation)
  + [Configuring Hadoop in Non-Secure Mode](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Configuring_Hadoop_in_Non-Secure_Mode)
    - [Configuring Environment of Hadoop Daemons](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Configuring_Environment_of_Hadoop_Daemons)
    - [Configuring the Hadoop Daemons](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Configuring_the_Hadoop_Daemons)
  + [Monitoring Health of NodeManagers](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Monitoring_Health_of_NodeManagers)
  + [Slaves File](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Slaves_File)
  + [Hadoop Rack Awareness](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Hadoop_Rack_Awareness)
  + [Logging](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Logging)
  + [Operating the Hadoop Cluster](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Operating_the_Hadoop_Cluster)
    - [Hadoop Startup](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Hadoop_Startup)
    - [Hadoop Shutdown](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Hadoop_Shutdown)
  + [Web Interfaces](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html" \l "Web_Interfaces)

# Hadoop Cluster Setup

## Purpose

This document describes how to install and configure Hadoop clusters ranging from a few nodes to extremely large clusters with thousands of nodes. To play with Hadoop, you may first want to install it on a single machine (see [Single Node Setup](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html)).

This document does not cover advanced topics such as [Security](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SecureMode.html) or High Availability.

## Prerequisites

* Install Java. See the [Hadoop Wiki](http://wiki.apache.org/hadoop/HadoopJavaVersions) for known good versions.
* Download a stable version of Hadoop from Apache mirrors.

## Installation

Installing a Hadoop cluster typically involves unpacking the software on all the machines in the cluster or installing it via a packaging system as appropriate for your operating system. It is important to divide up the hardware into functions.

Typically one machine in the cluster is designated as the NameNode and another machine the as ResourceManager, exclusively. These are the masters. Other services (such as Web App Proxy Server and MapReduce Job History server) are usually run either on dedicated hardware or on shared infrastrucutre, depending upon the load.

The rest of the machines in the cluster act as both DataNode and NodeManager. These are the slaves.

## Configuring Hadoop in Non-Secure Mode

Hadoop’s Java configuration is driven by two types of important configuration files:

Read-only default configuration - core-default.xml, hdfs-default.xml, yarn-default.xml and mapred-default.xml.

Site-specific configuration - etc/hadoop/core-site.xml, etc/hadoop/hdfs-site.xml, etc/hadoop/yarn-site.xml and etc/hadoop/mapred-site.xml.

Additionally, you can control the Hadoop scripts found in the bin/ directory of the distribution, by setting site-specific values via the etc/hadoop/hadoop-env.sh and etc/hadoop/yarn-env.sh.

To configure the Hadoop cluster you will need to configure the environment in which the Hadoop daemons execute as well as the configuration parameters for the Hadoop daemons.

HDFS daemons are NameNode, SecondaryNameNode, and DataNode. YARN damones are ResourceManager, NodeManager, and WebAppProxy. If MapReduce is to be used, then the MapReduce Job History Server will also be running. For large installations, these are generally running on separate hosts.

### **Configuring Environment of Hadoop Daemons**

Administrators should use the etc/hadoop/hadoop-env.sh and optionally the etc/hadoop/mapred-env.sh and etc/hadoop/yarn-env.sh scripts to do site-specific customization of the Hadoop daemons’ process environment.

At the very least, you must specify the JAVA\_HOME so that it is correctly defined on each remote node.

Administrators can configure individual daemons using the configuration options shown below in the table:

| **Daemon** | **Environment Variable** |
| --- | --- |
| NameNode | HADOOP\_NAMENODE\_OPTS |
| DataNode | HADOOP\_DATANODE\_OPTS |
| Secondary NameNode | HADOOP\_SECONDARYNAMENODE\_OPTS |
| ResourceManager | YARN\_RESOURCEMANAGER\_OPTS |
| NodeManager | YARN\_NODEMANAGER\_OPTS |
| WebAppProxy | YARN\_PROXYSERVER\_OPTS |
| Map Reduce Job History Server | HADOOP\_JOB\_HISTORYSERVER\_OPTS |

For example, To configure Namenode to use parallelGC, the following statement should be added in hadoop-env.sh :

export HADOOP\_NAMENODE\_OPTS="-XX:+UseParallelGC"

See etc/hadoop/hadoop-env.sh for other examples.

Other useful configuration parameters that you can customize include:

* HADOOP\_PID\_DIR - The directory where the daemons’ process id files are stored.
* HADOOP\_LOG\_DIR - The directory where the daemons’ log files are stored. Log files are automatically created if they don’t exist.
* HADOOP\_HEAPSIZE / YARN\_HEAPSIZE - The maximum amount of heapsize to use, in MB e.g. if the varibale is set to 1000 the heap will be set to 1000MB. This is used to configure the heap size for the daemon. By default, the value is 1000. If you want to configure the values separately for each deamon you can use.

In most cases, you should specify the HADOOP\_PID\_DIR and HADOOP\_LOG\_DIR directories such that they can only be written to by the users that are going to run the hadoop daemons. Otherwise there is the potential for a symlink attack.

It is also traditional to configure HADOOP\_PREFIX in the system-wide shell environment configuration. For example, a simple script inside /etc/profile.d:

HADOOP\_PREFIX=/path/to/hadoop

export HADOOP\_PREFIX

| **Daemon** | **Environment Variable** |
| --- | --- |
| ResourceManager | YARN\_RESOURCEMANAGER\_HEAPSIZE |
| NodeManager | YARN\_NODEMANAGER\_HEAPSIZE |
| WebAppProxy | YARN\_PROXYSERVER\_HEAPSIZE |
| Map Reduce Job History Server | HADOOP\_JOB\_HISTORYSERVER\_HEAPSIZE |

### **Configuring the Hadoop Daemons**

This section deals with important parameters to be specified in the given configuration files:

* etc/hadoop/core-site.xml

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| fs.defaultFS | NameNode URI | <hdfs://host:port/> |
| io.file.buffer.size | 131072 | Size of read/write buffer used in SequenceFiles. |

etc/hadoop/hdfs-site.xml

Configurations for NameNode:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| dfs.namenode.name.dir | Path on the local filesystem where the NameNode stores the namespace and transactions logs persistently. | If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy. |
| dfs.hosts /dfs.hosts.exclude | List of permitted/excluded DataNodes. | If necessary, use these files to control the list of allowable datanodes. |
| dfs.blocksize | 268435456 | HDFS blocksize of 256MB for large file-systems. |
| dfs.namenode.handler.count | 100 | More NameNode server threads to handle RPCs from large number of DataNodes. |

* Configurations for DataNode:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| dfs.datanode.data.dir | Comma separated list of paths on the local filesystem of a DataNodewhere it 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. |

etc/hadoop/yarn-site.xml

Configurations for ResourceManager and NodeManager:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| yarn.acl.enable | true /false | Enable ACLs? Defaults to *false*. |
| yarn.admin.acl | Admin ACL | ACL to set admins on the cluster. ACLs are of for *comma-separated-usersspacecomma-separated-groups*. Defaults to special value of **\*** which means*anyone*. Special value of just *space* means no one has access. |
| yarn.log-aggregation-enable | *false* | Configuration to enable or disable log aggregation |

* Configurations for ResourceManager:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| yarn.resourcemanager.address | ResourceManager host:port for clients to submit jobs. | *host:port* If set, overrides the hostname set in yarn.resourcemanager.hostname. |
| yarn.resourcemanager.scheduler.address | ResourceManager host:port for ApplicationMasters to talk to Scheduler to obtain resources. | *host:port* If set, overrides the hostname set in yarn.resourcemanager.hostname. |
| yarn.resourcemanager.resource-tracker.address | ResourceManager host:port for NodeManagers. | *host:port* If set, overrides the hostname set in yarn.resourcemanager.hostname. |
| yarn.resourcemanager.admin.address | ResourceManager host:port for administrative commands. | *host:port* If set, overrides the hostname set in yarn.resourcemanager.hostname. |
| yarn.resourcemanager.webapp.address | ResourceManager web-ui host:port. | *host:port* If set, overrides the hostname set in yarn.resourcemanager.hostname. |
| yarn.resourcemanager.hostname | ResourceManager host. | *host* Single hostname that can be set in place of setting allyarn.resourcemanager\*address resources. Results in default ports for ResourceManager components. |
| yarn.resourcemanager.scheduler.class | ResourceManager Scheduler class. | CapacityScheduler (recommended), FairScheduler (also recommended), or FifoScheduler |
| yarn.scheduler.minimum-allocation-mb | Minimum limit of memory to allocate to each container request at the Resource Manager. | In MBs |
| yarn.scheduler.maximum-allocation-mb | Maximum limit of memory to allocate to each container request at the Resource Manager. | In MBs |
| yarn.resourcemanager.nodes.include-path /yarn.resourcemanager.nodes.exclude-path | List of permitted/excluded NodeManagers. | If necessary, use these files to control the list of allowable NodeManagers. |

* Configurations for NodeManager:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| yarn.nodemanager.resource.memory-mb | Resource i.e. available physical memory, in MB, for given NodeManager | Defines total available resources on the NodeManager to be made available to running containers |
| yarn.nodemanager.vmem-pmem-ratio | Maximum ratio by which virtual memory usage of tasks may exceed physical memory | The virtual memory usage of each task may exceed its physical memory limit by this ratio. The total amount of virtual memory used by tasks on the NodeManager may exceed its physical memory usage by this ratio. |
| yarn.nodemanager.local-dirs | Comma-separated list of paths on the local filesystem where intermediate data is written. | Multiple paths help spread disk i/o. |
| yarn.nodemanager.log-dirs | Comma-separated list of paths on the local filesystem where logs are written. | Multiple paths help spread disk i/o. |
| yarn.nodemanager.log.retain-seconds | *10800* | Default time (in seconds) to retain log files on the NodeManager Only applicable if log-aggregation is disabled. |
| yarn.nodemanager.remote-app-log-dir | */logs* | HDFS directory where the application logs are moved on application completion. Need to set appropriate permissions. Only applicable if log-aggregation is enabled. |
| yarn.nodemanager.remote-app-log-dir-suffix | *logs* | Suffix appended to the remote log dir. Logs will be aggregated to ${yarn.nodemanager.remote-app-log-dir}/${user}/${thisParam} Only applicable if log-aggregation is enabled. |
| yarn.nodemanager.aux-services | mapreduce\_shuffle | Shuffle service that needs to be set for Map Reduce applications. |

* Configurations for History Server (Needs to be moved elsewhere):

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| yarn.log-aggregation.retain-seconds | *-1* | How long to keep aggregation logs before deleting them. -1 disables. Be careful, set this too small and you will spam the name node. |
| yarn.log-aggregation.retain-check-interval-seconds | *-1* | Time between checks for aggregated log retention. If set to 0 or a negative value then the value is computed as one-tenth of the aggregated log retention time. Be careful, set this too small and you will spam the name node. |

etc/hadoop/mapred-site.xml

Configurations for MapReduce Applications:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| mapreduce.framework.name | yarn | Execution framework set to Hadoop YARN. |
| mapreduce.map.memory.mb | 1536 | Larger resource limit for maps. |
| mapreduce.map.java.opts | -Xmx1024M | Larger heap-size for child jvms of maps. |
| mapreduce.reduce.memory.mb | 3072 | Larger resource limit for reduces. |
| mapreduce.reduce.java.opts | -Xmx2560M | Larger heap-size for child jvms of reduces. |
| mapreduce.task.io.sort.mb | 512 | Higher memory-limit while sorting data for efficiency. |
| mapreduce.task.io.sort.factor | 100 | More streams merged at once while sorting files. |
| mapreduce.reduce.shuffle.parallelcopies | 50 | Higher number of parallel copies run by reduces to fetch outputs from very large number of maps. |

* Configurations for MapReduce JobHistory Server:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| mapreduce.jobhistory.address | MapReduce JobHistory Server *host:port* | Default port is 10020. |
| mapreduce.jobhistory.webapp.address | MapReduce JobHistory Server Web UI *host:port* | Default port is 19888. |
| mapreduce.jobhistory.intermediate-done-dir | /mr-history/tmp | Directory where history files are written by MapReduce jobs. |
| mapreduce.jobhistory.done-dir | /mr-history/done | Directory where history files are managed by the MR JobHistory Server. |

## Monitoring Health of NodeManagers

Hadoop provides a mechanism by which administrators can configure the NodeManager to run an administrator supplied script periodically to determine if a node is healthy or not.

Administrators can determine if the node is in a healthy state by performing any checks of their choice in the script. If the script detects the node to be in an unhealthy state, it must print a line to standard output beginning with the string ERROR. The NodeManager spawns the script periodically and checks its output. If the script’s output contains the string ERROR, as described above, the node’s status is reported as unhealthy and the node is black-listed by the ResourceManager. No further tasks will be assigned to this node. However, the NodeManager continues to run the script, so that if the node becomes healthy again, it will be removed from the blacklisted nodes on the ResourceManager automatically. The node’s health along with the output of the script, if it is unhealthy, is available to the administrator in the ResourceManager web interface. The time since the node was healthy is also displayed on the web interface.

The following parameters can be used to control the node health monitoring script in etc/hadoop/yarn-site.xml.

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| yarn.nodemanager.health-checker.script.path | Node health script | Script to check for node’s health status. |
| yarn.nodemanager.health-checker.script.opts | Node health script options | Options for script to check for node’s health status. |
| yarn.nodemanager.health-checker.script.interval-ms | Node health script interval | Time interval for running health script. |
| yarn.nodemanager.health-checker.script.timeout-ms | Node health script timeout interval | Timeout for health script execution. |

The health checker script is not supposed to give ERROR if only some of the local disks become bad. NodeManager has the ability to periodically check the health of the local disks (specifically checks nodemanager-local-dirs and nodemanager-log-dirs) and after reaching the threshold of number of bad directories based on the value set for the config property yarn.nodemanager.disk-health-checker.min-healthy-disks, the whole node is marked unhealthy and this info is sent to resource manager also. The boot disk is either raided or a failure in the boot disk is identified by the health checker script.

## Slaves File

List all slave hostnames or IP addresses in your etc/hadoop/slaves file, one per line. Helper scripts (described below) will use the etc/hadoop/slaves file to run commands on many hosts at once. It is not used for any of the Java-based Hadoop configuration. In order to use this functionality, ssh trusts (via either passphraseless ssh or some other means, such as Kerberos) must be established for the accounts used to run Hadoop.

## Hadoop Rack Awareness

Many Hadoop components are rack-aware and take advantage of the network topology for performance and safety. Hadoop daemons obtain the rack information of the slaves in the cluster by invoking an administrator configured module. See the [Rack Awareness](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/RackAwareness.html) documentation for more specific information.

It is highly recommended configuring rack awareness prior to starting HDFS.

## Logging

Hadoop uses the [Apache log4j](http://logging.apache.org/log4j/2.x/) via the Apache Commons Logging framework for logging. Edit the etc/hadoop/log4j.properties file to customize the Hadoop daemons’ logging configuration (log-formats and so on).

## Operating the Hadoop Cluster

Once all the necessary configuration is complete, distribute the files to the HADOOP\_CONF\_DIR directory on all the machines. This should be the same directory on all machines.

In general, it is recommended that HDFS and YARN run as separate users. In the majority of installations, HDFS processes execute as ‘hdfs’. YARN is typically using the ‘yarn’ account.

### **Hadoop Startup**

To start a Hadoop cluster you will need to start both the HDFS and YARN cluster.

The first time you bring up HDFS, it must be formatted. Format a new distributed filesystem as *hdfs*:

[hdfs]$ $HADOOP\_PREFIX/bin/hdfs namenode -format <cluster\_name>

Start the HDFS NameNode with the following command on the designated node as *hdfs*:

[hdfs]$ $HADOOP\_PREFIX/sbin/hadoop-daemon.sh --config $HADOOP\_CONF\_DIR --script hdfs start namenode

Start a HDFS DataNode with the following command on each designated node as *hdfs*:

[hdfs]$ $HADOOP\_PREFIX/sbin/hadoop-daemons.sh --config $HADOOP\_CONF\_DIR --script hdfs start datanode

If etc/hadoop/slaves and ssh trusted access is configured (see [Single Node Setup](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html)), all of the HDFS processes can be started with a utility script. As *hdfs*:

[hdfs]$ $HADOOP\_PREFIX/sbin/start-dfs.sh

Start the YARN with the following command, run on the designated ResourceManager as *yarn*:

[yarn]$ $HADOOP\_YARN\_HOME/sbin/yarn-daemon.sh --config $HADOOP\_CONF\_DIR start resourcemanager

Run a script to start a NodeManager on each designated host as *yarn*:

[yarn]$ $HADOOP\_YARN\_HOME/sbin/yarn-daemons.sh --config $HADOOP\_CONF\_DIR start nodemanager

Start a standalone WebAppProxy server. Run on the WebAppProxy server as *yarn*. If multiple servers are used with load balancing it should be run on each of them:

[yarn]$ $HADOOP\_YARN\_HOME/sbin/yarn-daemon.sh --config $HADOOP\_CONF\_DIR start proxyserver

If etc/hadoop/slaves and ssh trusted access is configured (see [Single Node Setup](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html)), all of the YARN processes can be started with a utility script. As *yarn*:

[yarn]$ $HADOOP\_PREFIX/sbin/start-yarn.sh

Start the MapReduce JobHistory Server with the following command, run on the designated server as *mapred*:

[mapred]$ $HADOOP\_PREFIX/sbin/mr-jobhistory-daemon.sh --config $HADOOP\_CONF\_DIR start historyserver

### **Hadoop Shutdown**

Stop the NameNode with the following command, run on the designated NameNode as *hdfs*:

[hdfs]$ $HADOOP\_PREFIX/sbin/hadoop-daemon.sh --config $HADOOP\_CONF\_DIR --script hdfs stop namenode

Run a script to stop a DataNode as *hdfs*:

[hdfs]$ $HADOOP\_PREFIX/sbin/hadoop-daemons.sh --config $HADOOP\_CONF\_DIR --script hdfs stop datanode

If etc/hadoop/slaves and ssh trusted access is configured (see [Single Node Setup](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html)), all of the HDFS processes may be stopped with a utility script. As *hdfs*:

[hdfs]$ $HADOOP\_PREFIX/sbin/stop-dfs.sh

Stop the ResourceManager with the following command, run on the designated ResourceManager as *yarn*:

[yarn]$ $HADOOP\_YARN\_HOME/sbin/yarn-daemon.sh --config $HADOOP\_CONF\_DIR stop resourcemanager

Run a script to stop a NodeManager on a slave as *yarn*:

[yarn]$ $HADOOP\_YARN\_HOME/sbin/yarn-daemons.sh --config $HADOOP\_CONF\_DIR stop nodemanager

If etc/hadoop/slaves and ssh trusted access is configured (see [Single Node Setup](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html)), all of the YARN processes can be stopped with a utility script. As *yarn*:

[yarn]$ $HADOOP\_PREFIX/sbin/stop-yarn.sh

Stop the WebAppProxy server. Run on the WebAppProxy server as *yarn*. If multiple servers are used with load balancing it should be run on each of them:

[yarn]$ $HADOOP\_YARN\_HOME/sbin/yarn-daemon.sh --config $HADOOP\_CONF\_DIR stop proxyserver

Stop the MapReduce JobHistory Server with the following command, run on the designated server as *mapred*:

[mapred]$ $HADOOP\_PREFIX/sbin/mr-jobhistory-daemon.sh --config $HADOOP\_CONF\_DIR stop historyserver

## Web Interfaces

Once the Hadoop cluster is up and running check the web-ui of the components as described below:

| **Daemon** | **Web Interface** | **Notes** |
| --- | --- | --- |
| NameNode | <http://nn_host:port/> | Default HTTP port is 50070. |
| ResourceManager | <http://rm_host:port/> | Default HTTP port is 8088. |
| MapReduce JobHistory Server | <http://jhs_host:port/> | Default HTTP port is 19888. |