# Contents

ABOUT THIS GUIDE ............................................................................................................................................ 1

INTRODUCTION TO HADOOP HIGH AVAILABILITY ............................................................................................... 1

OVERVIEW .................................................................................................................................................................... 1

BACKGROUND ............................................................................................................................................................... 1

ARCHITECTURE .............................................................................................................................................................. 1

HDFS HIGH AVAILABILITY HARDWARE CONFIGURATION ..................................................................................... 2

HDFS HIGH AVAILABILITY SOFTWARE CONFIGURATION ...................................................................................... 3

CONFIGURATION OVERVIEW ............................................................................................................................................. 3

CHANGES TO EXISTING CONFIGURATION PARAMETERS .................................................................................................. 4

NEW CONFIGURATION PARAMETERS .................................................................................................................................. 4

Configure dfs.nameservices . ........................................................................................................................................... 4

Configure dfs.ha.namenodes.[nameservice ID] . ........................................................................................................... 5

Configure dfs.namenode.rpc-address.[nameservice ID] . ............................................................................................. 5

Configure dfs.namenode.http-address.[nameservice ID] . ............................................................................................. 6

Configure dfs.namenode.shared.edits.dir . .................................................................................................................. 7

CLIENT FAILOVER CONFIGURATION .................................................................................................................................... 7

FENCING CONFIGURATION ............................................................................................................................................... 7

Configuring the sshfence fencing method .................................................................................................................. 8

Configuring the shell fencing method ....................................................................................................................... 9

AUTOMATIC FAILOVER CONFIGURATION ........................................................................................................................... 10

Component Overview .......................................................................................................................................... 10

Deploying ZooKeeper .......................................................................................................................................... 11

Configuring Automatic Failover .......................................................................................................................... 11

Initializing the HA state in ZooKeeper .................................................................................................................... 12

Securing access to ZooKeeper ........................................................................................................................................... 13

HDFS HIGH AVAILABILITY INITIAL DEPLOYMENT ................................................................................................ 14

STARTING THE NAMENODES ........................................................................................................................................... 14

DEPLOYING AUTOMATIC FAILOVER .................................................................................................................................. 15

VERIFYING AUTOMATIC FAILOVER ................................................................................................................................... 16
UPGRADING AN HA CONFIGURATION FROM A CDH4 BETA RELEASE TO THE LATEST RELEASE .................................17

UPGRADING FROM CDH4 BETA 2 ............................................................................................................................17

UPGRADING FROM CDH4 BETA 1 ............................................................................................................................17

Step 1: Shut Down the Cluster .............................................................................................................................17

Step 2: Unconfigure HA .......................................................................................................................................17

Step 3: Upgrade to the latest CDH4 Release and Upgrade HDFS .......................................................................17

Step 4: Redeploy HA ............................................................................................................................................17

HDFS HIGH AVAILABILITY ADMINISTRATION .....................................................................................................17

HA ADMINISTRATION USING THE HAADMIN COMMAND .................................................................................18

transitionToActive and transitionToStandby .....................................................................................................18

failover ................................................................................................................................................................18

getServiceState ...................................................................................................................................................19

checkHealth ........................................................................................................................................................19

USING THE DFSADMIN COMMAND WHEN HA IS ENABLED .........................................................................19

CONFIGURING OTHER CDH COMPONENTS TO USE HDFS HA ........................................................................19

CONFIGURING HBASE TO USE HDFS HA ............................................................................................................20

HBase-HDFS HA Troubleshooting ........................................................................................................................20
About this Guide

This CDH4 High Availability Guide is for Apache Hadoop system administrators who want to enable continuous availability by configuring a CDH4 cluster without single points of failure.

Introduction to Hadoop High Availability

1. Overview
2. Background
3. Architecture

Overview

This guide provides an overview of the HDFS High Availability (HA) feature and how to configure and manage an HA HDFS cluster.

This document assumes that the reader has a general understanding of components and node types in an HDFS cluster. For details, see the Apache HDFS Architecture Guide.

Background

Prior to CDH4, the NameNode was a single point of failure (SPOF) in an HDFS cluster. Each cluster had a single NameNode, and if that machine or process became unavailable, the cluster as a whole would be unavailable until the NameNode was either restarted or brought up on a separate machine.

This impacted the total availability of the HDFS cluster in two major ways:

1. In the case of an unplanned event such as a machine crash, the cluster would be unavailable until an operator restarted the NameNode.
2. Planned maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.

The HDFS HA feature addresses the above problems by providing the option of running two redundant NameNodes in the same cluster in an Active/Passive configuration with a hot standby. This allows a fast failover to a new NameNode in the case that a machine crashes, or a graceful administrator-initiated failover for the purpose of planned maintenance.

Architecture

In a typical HA cluster, two separate machines are configured as NameNodes. At any point in time, exactly one of the NameNodes is in an Active state, and the other is in a Standby state. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if necessary.
In order for the Standby node to keep its state synchronized with the Active node, the current implementation requires that the two nodes both have access to a directory on a shared storage device (for example, an NFS mount from a NAS). This restriction will likely be relaxed in future releases.

When any namespace modification is performed by the Active node, it durably logs a record of the modification to an edit log file stored in the shared directory. The Standby node constantly watches this directory for edits, and when edits occur, the Standby node applies them to its own namespace. In the event of a failover, the Standby will ensure that it has read all of the edits from the shared storage before promoting itself to the Active state. This ensures that the namespace state is fully synchronized before a failover occurs.

In order to provide a fast failover, it is also necessary that the Standby node has up-to-date information regarding the location of blocks in the cluster. In order to achieve this, the DataNodes are configured with the location of both NameNodes, and they send block location information and heartbeats to both.

It is vital for the correct operation of an HA cluster that only one of the NameNodes be Active at a time. Otherwise, the namespace state would quickly diverge between the two, risking data loss or other incorrect results. In order to ensure this and prevent the so-called "split-brain scenario," the administrator must configure at least one fencing method for the shared storage. During a failover, if it cannot be verified that the previous Active NameNode has relinquished its Active state, the fencing process is responsible for cutting off the previous Active NameNode's access to the shared edits storage. This prevents it from making any further edits to the namespace, allowing the new Active NameNode to safely proceed with failover.

### HDFS High Availability Hardware Configuration

In order to deploy an HA cluster, you should prepare the following:

- **NameNode machines** - the machines on which you run the Active and Standby NameNodes should have equivalent hardware to each other, and equivalent hardware to what would be used in a non-HA cluster.

- **Shared storage** - you will need to have a shared directory which both NameNode machines can have read/write access to. Typically, this is a remote filer which supports NFS and is mounted on each of the NameNode machines. In this release, only one shared edits directory is supported. The availability of the system is limited by the availability of this shared edits directory, and therefore in order to remove all single points of failure there must be redundancy for the shared edits directory. That is, there must be multiple network paths to the storage, and redundancy in the storage itself (disk, network, and power). Because of this, it is recommended that the shared storage server be a high-quality dedicated NAS appliance rather than a simple Linux server.
Note

In an HA cluster, the Standby NameNode also performs checkpoints of the namespace state, and thus it is not necessary to run a Secondary NameNode, CheckpointNode, or BackupNode in an HA cluster. In fact, to do so would be an error. If you are reconfiguring a non-HA-enabled HDFS cluster to be HA-enabled, you can reuse the hardware which you had previously dedicated to the Secondary NameNode.

HDFS High Availability Software Configuration

Here are the general steps to configure HDFS HA, each of which is described in more detail in the following sections:

1. Configuration Overview
2. Changes to Existing Configuration Parameters
3. New Configuration Parameters
4. Client Failover Configuration
5. Fencing Configuration
6. Automatic Failover Configuration

An important note on compatibility

The configuration keys for HA may change between the CDH4 Beta 1 and CDH4 Beta 2 releases. If they do, this will be documented in Incompatible Changes.

Configuration Overview

As with HDFS Federation configuration, HA configuration is backward compatible and allows existing single NameNode configurations to work without change. The new configuration is designed such that all the nodes in the cluster can have the same configuration without the need for deploying different configuration files to different machines based on the type of the node.

HA clusters reuse the NameService ID to identify a single HDFS instance that may consist of multiple HA NameNodes. In addition, a new abstraction called NameNode ID is introduced. Each distinct NameNode in the cluster has a different NameNode ID. To support a single configuration file for all of the NameNodes, the relevant configuration parameters are suffixed with the NameService ID as well as the NameNode ID.
Changes to Existing Configuration Parameters

The following configuration parameter has changed for YARN implementations:

fs.defaultFS - Formerly fs.default.name, the default path prefix used by the Hadoop FS client when none is given. (fs.default.name is deprecated for YARN implementations, but will still work.)

Optionally, you may now configure the default path for Hadoop clients to use the new HA-enabled logical URI. For example, if you use "mycluster" as the NameService ID as shown below, this will be the value of the authority portion of all of your HDFS paths. You can configure the default path in your core-site.xml file:

- For MRv1:

```xml
<property>
  <name>fs.default.name/name</name>
  <value>hdfs://mycluster</value>
</property>
```

- For YARN:

```xml
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://mycluster</value>
</property>
```

New Configuration Parameters

To configure HA NameNodes, you must add several configuration options to your hdfs-site.xml configuration file.

The order in which you set these configurations is unimportant, but the values you choose for dfs.nameservices and dfs.ha.namnodes.[NameService ID] will determine the keys of those that follow. Thus, you should decide on these values before setting the rest of the configuration options.

Configure dfs.nameservices.

dfs.nameservices - the logical name for this new nameservice

Choose a logical name for this nameservice, for example "mycluster", and use this logical name for the value of this config option. The name you choose is arbitrary. It will be used both for configuration and as the authority component of absolute HDFS paths in the cluster.
Note
If you are also using HDFS Federation, this configuration setting should also include the list of other nameservices, HA or otherwise, as a comma-separated list.

```xml
<property>
  <name>dfs.nameservices</name>
  <value>mycluster</value>
</property>
```

Configure `dfs.ha.namenodes.[nameservice ID]`.

dfs.ha.namenodes.[nameservice ID] - unique identifiers for each NameNode in the nameservice

Configure a list of comma-separated NameNode IDs. This will be used by DataNodes to determine all the NameNodes in the cluster. For example, if you used "mycluster" as the NameService ID previously, and you wanted to use "nn1" and "nn2" as the individual IDs of the NameNodes, you would configure this as follows:

```xml
<property>
  <name>dfs.ha.namenodes.mycluster</name>
  <value>nn1,nn2</value>
</property>
```

Note
In this release, only a maximum of two NameNodes may be configured per nameservice.

Configure `dfs.namenode.rpc-address.[nameservice ID]`.

dfs.namenode.rpc-address.[nameservice ID].[name node ID] - the fully-qualified RPC address for each NameNode to listen on

For both of the previously-configured NameNode IDs, set the full address and IPC port of the NameNode process. Note that this results in two separate configuration options.
HDFS High Availability Software Configuration

For example:

```xml
<property>
    <name>dfs.namenode.rpc-address.mycluster.nn1</name>
    <value>machine1.example.com:8020</value>
</property>
<property>
    <name>dfs.namenode.rpc-address.mycluster.nn2</name>
    <value>machine2.example.com:8020</value>
</property>
```

**Note**

If necessary, you can similarly configure the "servicerpc-address" setting.

Configure `dfs.namenode.http-address.[nameservice ID]`.

dfs.namenode.http-address.[nameservice ID].[name node ID] - the fully-qualified HTTP address for each NameNode to listen on

Similarly to rpc-address above, set the addresses for both NameNodes' HTTP servers to listen on. For example:

```xml
<property>
    <name>dfs.namenode.http-address.mycluster.nn1</name>
    <value>machine1.example.com:50070</value>
</property>
<property>
    <name>dfs.namenode.http-address.mycluster.nn2</name>
    <value>machine2.example.com:50070</value>
</property>
```

**Note**

If you have Hadoop's Kerberos security features enabled, you should also set the https-address similarly for each NameNode.
Configure `dfs.namenode.shared.edits.dir`.

`dfs.namenode.shared.edits.dir` - the location of the shared storage directory

Configure the path to the remote shared edits directory which the Standby NameNode uses to stay up-to-date with all the file system changes the Active NameNode makes. You should only configure one of these directories. This directory should be mounted read/write on both NameNode machines. The value of this setting should be the absolute path to this directory on the NameNode machines. For example:

```xml
<property>
  <name>dfs.namenode.shared.edits.dir</name>
  <value>file:///mnt/filer1/dfs/ha-name-dir-shared</value>
</property>
```

Client Failover Configuration

`dfs.client.failover.proxy.provider.[nameservice ID]` - the Java class that HDFS clients use to contact the Active NameNode

Configure the name of the Java class which the DFS Client will use to determine which NameNode is the current Active, and therefore which NameNode is currently serving client requests. The only implementation which currently ships with Hadoop is the `ConfiguredFailoverProxyProvider`, so use this unless you are using a custom one. For example:

```xml
<property>
  <name>dfs.client.failover.proxy.provider.mycluster</name>
  <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

Fencing Configuration

`dfs.ha.fencing.methods` - a list of scripts or Java classes which will be used to fence the Active NameNode during a failover

It is critical for correctness of the system that only one NameNode is in the Active state at any given time. Thus, during a failover, the `haadmin` command first ensures that the Active NameNode is either in the Standby state, or that the Active NameNode process has terminated, before `haadmin` transitions the other NameNode to the Active state. The method that `haadmin` uses to ensure this is called the fencing method.
You must configure at least one fencing method. To specify more than one method, put them in a carriage-return-separated list; the methods will be attempted in order until one of them succeeds.

**Important**
There is no default fencing method. If you do not configure a fencing method, HA will fail.

There are two fencing methods which ship with Hadoop:

- **sshfence**
- **shell**

**Important**
Both of these methods fail if the system hosting the originally active primary NameNode (the one you started in the first step when you first deployed your HA configuration) becomes unreachable. For an example of a method that does not fail if the NameNode is unreachable, see the sample script later in this section.

For information on implementing your own custom fencing method, see the org.apache.hadoop.ha.NodeFencer class.

**Configuring the sshfence fencing method**

`sshfence` - SSH to the Active NameNode and kill the process

The `sshfence` option uses SSH to connect to the target node and uses `fuser` to kill the process listening on the service's TCP port. In order for this fencing option to work, it must be able to SSH to the target node without providing a passphrase. Thus, you must also configure the `dfs.ha.fencing.ssh.private-key-files` option, which is a comma-separated list of SSH private key files.

**Important**
The files must be accessible to the user running the NameNode processes (typically the hdfs user on the NameNode hosts).
For example:

```xml
<property>
  <name>dfs.ha.fencing.methods</name>
  <value>sshfence</value>
</property>

<property>
  <name>dfs.ha.fencing.ssh.private-key-files</name>
  <value>/home/exampleuser/.ssh/id_rsa</value>
</property>
```

Optionally, you can configure a non-standard username or port to perform the SSH as shown below. You can also configure a timeout, in milliseconds, for the SSH, after which this fencing method will be considered to have failed:

```xml
<property>
  <name>dfs.ha.fencing.methods</name>
  <value>sshfence([username]:[port])</value>
</property>
<property>
  <name>dfs.ha.fencing.ssh.connect-timeout</name>
  <value></value>
</property>
```

**Configuring the shell fencing method**

**shell** - run an arbitrary shell command to fence the Active NameNode

The shell fencing method runs an arbitrary shell command, which you can configure as shown below:

```xml
<property>
  <name>dfs.ha.fencing.methods</name>
  <value>shell(/path/to/my/script.sh arg1 arg2 ...)</value>
</property>
```

The string between '(' and ')' is passed directly to a bash shell and cannot include any closing parentheses.

When executed, the first argument to the configured script will be the address of the NameNode to be fenced, followed by all arguments specified in the configuration.
The shell command will be run with an environment set up to contain all of the current Hadoop configuration variables, with the '_' character replacing any '.' characters in the configuration keys. If the shell command returns an exit code of 0, the fencing is determined to be successful. If it returns any other exit code, the fencing was not successful and the next fencing method in the list will be attempted.

**Note**

This fencing method does not implement any timeout. If timeouts are necessary, they should be implemented in the shell script itself (for example, by forking a subshell to kill its parent in some number of seconds).

**Example shell fencing script**

Here's an example of a fencing script. This script fences off an inactive NameNode from a NetApp NFS filer.

```bash
ssh -o PasswordAuthentication=no -x root@my.netapp.filer.net exportfs -b enable save "${target_address}" /vol/namenode
```

In order for this script to work, you must set up passwordless `ssh` from both the active and standby NameNodes to your NetApp filer. You should also make sure that this script has the Linux execute (`x`) permission bit set.

**Automatic Failover Configuration**

The above sections describe how to configure manual failover. In that mode, the system will not automatically trigger a failover from the active to the standby NameNode, even if the active node has failed. This section describes how to configure and deploy automatic failover.

**Component Overview**

Automatic failover adds two new components to an HDFS deployment: a ZooKeeper quorum, and the ZKFailoverController process (abbreviated as ZKFC).

Apache ZooKeeper is a highly available service for maintaining small amounts of coordination data, notifying clients of changes in that data, and monitoring clients for failures. The implementation of automatic HDFS failover relies on ZooKeeper for the following things:

- **Failure detection** - each of the NameNode machines in the cluster maintains a persistent session in ZooKeeper. If the machine crashes, the ZooKeeper session will expire, notifying the other NameNode that a failover should be triggered.
• **Active NameNode election** - ZooKeeper provides a simple mechanism to exclusively elect a node as active. If the current active NameNode crashes, another node can take a special exclusive lock in ZooKeeper indicating that it should become the next active NameNode.

The **ZKFailoverController (ZKFC)** is a new component - a ZooKeeper client which also monitors and manages the state of the NameNode. Each of the machines which runs a NameNode also runs a ZKFC, and that ZKFC is responsible for:

• **Health monitoring** - the ZKFC pings its local NameNode on a periodic basis with a health-check command. So long as the NameNode responds promptly with a healthy status, the ZKFC considers the node healthy. If the node has crashed, frozen, or otherwise entered an unhealthy state, the health monitor will mark it as unhealthy.

• **ZooKeeper session management** - when the local NameNode is healthy, the ZKFC holds a session open in ZooKeeper. If the local NameNode is active, it also holds a special lock \( znode \). This lock uses ZooKeeper's support for "ephemeral" nodes; if the session expires, the lock node will be automatically deleted.

• **ZooKeeper-based election** - if the local NameNode is healthy, and the ZKFC sees that no other node currently holds the lock \( znode \), it will itself try to acquire the lock. If it succeeds, then it has "won the election", and is responsible for running a failover to make its local NameNode active. The failover process is similar to the manual failover described above: first, the previous active is fenced if necessary, and then the local NameNode transitions to active state.

**Deploying ZooKeeper**

In a typical deployment, ZooKeeper daemons are configured to run on three or five nodes. Since ZooKeeper itself has light resource requirements, it is acceptable to collocate the ZooKeeper nodes on the same hardware as the HDFS NameNode and Standby Node. Operators using MapReduce v2 (MRv2) often choose to deploy the third ZooKeeper process on the same node as the YARN ResourceManager. It is advisable to configure the ZooKeeper nodes to store their data on separate disk drives from the HDFS metadata for best performance and isolation.

See the [ZooKeeper documentation](#) for instructions on how to set up a ZooKeeper ensemble. In the following sections we assume that you have set up a ZooKeeper cluster running on three or more nodes, and have verified its correct operation by connecting using the ZooKeeper command-line interface (CLI).

**Configuring Automatic Failover**

**Note**

Before you begin configuring automatic failover, you must shut down your cluster. It is not currently possible to transition from a manual failover setup to an automatic failover setup while the cluster is running.
Configuring automatic failover requires two additional configuration parameters. In your hdfs-site.xml file, add:

```xml
<property>
  <name>dfs.ha.automatic-failover.enabled</name>
  <value>true</value>
</property>
```

This specifies that the cluster should be set up for automatic failover. In your core-site.xml file, add:

```xml
<property>
  <name>ha.zookeeper.quorum</name>
  <value>zk1.example.com:2181,zk2.example.com:2181,zk3.example.com:2181</value>
</property>
```

This lists the host-port pairs running the ZooKeeper service.

As with the parameters described earlier in this document, these settings may be configured on a per-nameservice basis by suffixing the configuration key with the nameservice ID. For example, in a cluster with federation enabled, you can explicitly enable automatic failover for only one of the nameservices by setting `dfs.ha.automatic-failover.enabled.my-nameservice-id`.

There are several other configuration parameters which you can set to control the behavior of automatic failover, but they are not necessary for most installations. See the configuration section of the Hadoop documentation for details.

**Initializing the HA state in ZooKeeper**

After you have added the configuration keys, the next step is to initialize the required state in ZooKeeper. You can do so by running the following command from one of the NameNode hosts.

```
$ hdfs zkfc -formatZK
```

This will create a znode in ZooKeeper in which the automatic failover system stores its data.
Securing access to ZooKeeper

If you are running a secure cluster, you will probably want to ensure that the information stored in ZooKeeper is also secured. This prevents malicious clients from modifying the metadata in ZooKeeper or potentially triggering a false failover.

In order to secure the information in ZooKeeper, first add the following to your core-site.xml file:

```xml
<property>
  <name>ha.zookeeper.auth</name>
  <value>@/path/to/zk-auth.txt</value>
</property>

<property>
  <name>ha.zookeeper.acl</name>
  <value>@/path/to/zk-acl.txt</value>
</property>
```

Note the '@' character in these values – this specifies that the configurations are not inline, but rather point to a file on disk.

The first configured file specifies a list of ZooKeeper authentications, in the same format as used by the ZooKeeper CLI. For example, you may specify something like `digest:hdfs-zkfcs:mypassword` where `hdfs-zkfcs` is a unique username for ZooKeeper, and `mypassword` is some unique string used as a password.

Next, generate a ZooKeeper Access Control List (ACL) that corresponds to this authentication, using a command such as the following:

```bash
$ java -cp $ZK_HOME/lib/*:$ZK_HOME/zookeeper-3.4.2.jar org.apache.zookeeper.server.auth.DigestAuthenticationProvider hdfs-zkfcs:mypassword
output: hdfs-zkfcs:mypassword->hdfs-zkfcs:P/QVnYyU/nF/mGYvB/xurX8dYs=
```

Copy and paste the section of this output after the '->' string into the file `zk-acls.txt`, prefixed by the string "digest:". For example:

```
digest:hdfs-zkfcs:v1UvLnd8M1acsE80rDuu6ONESbM=:rwcd
```

To put these ACLs into effect, rerun the `zkfc -formatZK` command as described above.
After doing so, you can verify the ACLs from the ZooKeeper CLI as follows:

```
[zk: localhost:2181(CONNECTED) 1] getAcl /hadoop-ha
'digest,'hdfs-zkfcs:vlUvLnd8M1acsE80rDuu6ONESbM=
: cdrwa
```

## HDFS High Availability Initial Deployment

After you have set all of the necessary configuration options, you are ready to start the two HA NameNodes.

### Important

If you are setting up a new HDFS cluster, you should first format the NameNode you will use as your primary NameNode; see [Formatting the NameNode](#).

### Starting the NameNodes

1. Start the primary (formatted) NameNode:

   ```bash
   $ sudo service hadoop-hdfs-namenode start
   ```

2. Start the standby NameNode:

   ```bash
   $ sudo -u hdfs hadfs namenode -bootstrapStandby
   $ sudo service hadoop-hdfs-namenode start
   ```
Note

- If the primary NameNode is in standby mode, you are prompted to transition it to active, for example:

```
NameNode ha-nn-uri.nn2 at
c0405.hal.cloudera.com/172.29.81.122:17020
is not currently in ACTIVE state.
Do you want to automatically transition it to active now? (Y or N) y
12/04/17 22:10:28 INFO ha.BootstrapStandby: Transitioning the running
namenode to active...
```

- If the primary NameNode is in safe mode, the operation will fail and you will need to force the primary NameNode out of safe mode.

Starting the standby NameNode with the `-bootstrapStandby` option copies over the contents of the primary NameNode's metadata directories (including the namespace information and most recent checkpoint) to the standby NameNode. (The location of the directories containing the NameNode metadata is configured via the configuration options `dfs.namenode.name.dir` and/or `dfs.namenode.edits.dir`.)

You can visit each NameNode's web page by browsing to its configured HTTP address. Notice that next to the configured address is the HA state of the NameNode (either "Standby" or "Active"). Whenever an HA NameNode starts and automatic failover is not enabled, it is initially in the Standby state. If automatic failover is enabled the first NameNode that is started will become active.

**Deploying Automatic Failover**

If you have configured automatic failover using the ZooKeeper FailoverController (ZKFC), you must install and start the `zkfc` daemon on each of the machines that runs a NameNode. Proceed as follows.

**To install ZKFC on Red Hat-compatible systems:**

```
$ sudo yum install hadoop-hdfs-zkfc
```

**To install ZKFC on Ubuntu and Debian systems:**

```
$ sudo apt-get install hadoop-hdfs-zkfc
```
To install ZKFC on SUSE systems:

```
$ sudo zypper install hadoop-hdfs-zkfc
```

To start the zkfc daemon:

```
$ sudo service hadoop-hdfs-zkfc start
```

It is not important that you start the ZKFC and NameNode daemons in a particular order. On any given node you can start the ZKFC before or after its corresponding NameNode.

You should add monitoring on each host that runs a NameNode to ensure that the ZKFC remains running. In some types of ZooKeeper failures, for example, the ZKFC may unexpectedly exit, and should be restarted to ensure that the system is ready for automatic failover.

Additionally, you should monitor each of the servers in the ZooKeeper quorum. If ZooKeeper crashes, then automatic failover will not function. If the ZooKeeper cluster crashes, no automatic failovers will be triggered. However, HDFS will continue to run without any impact. When ZooKeeper is restarted, HDFS will reconnect with no issues.

**Verifying Automatic Failover**

After the initial deployment of a cluster with automatic failover enabled, you should test its operation. To do so, first locate the active NameNode. As mentioned above, you can tell which node is active by visiting the NameNode web interfaces.

Once you have located your active NameNode, you can cause a failure on that node. For example, you can use `kill -9 <pid of NN>` to simulate a JVM crash. Or you can power-cycle the machine or its network interface to simulate different kinds of outages. After you trigger the outage you want to test, the other NameNode should automatically become active within several seconds. The amount of time required to detect a failure and trigger a failover depends on the configuration of `ha.zookeeper.session-timeout.ms`, but defaults to 5 seconds.

If the test does not succeed, you may have a misconfiguration. Check the logs for the zkfc daemons as well as the NameNode daemons in order to further diagnose the issue.
Upgrading an HA Configuration from a CDH4 Beta Release to the Latest Release

Upgrading from CDH4 Beta 2

To upgrade your HA configuration from CDH4 Beta 2 to the latest release, proceed as follows:

1. Follow the directions for upgrading a cluster under Upgrading to the Latest Version of CDH4.
2. Start the primary NameNode (if you have not already done so) and standby NameNode, following the instructions under HDFS High Availability Initial Deployment.

Upgrading from CDH4 Beta 1

Upgrading from CDH4 Beta 1 requires upgrading HDFS, and unfortunately that can't be done while HA is configured. To upgrade HDFS in an HA configuration, you must unconfigure HA, upgrade HDFS, and then reconfigure HA. Proceed as follows.

Step 1: Shut Down the Cluster

Follow the instructions in Step 1: Prepare the cluster for the upgrade under Upgrading to the Latest Version of CDH4.

Step 2: Unconfigure HA

1. Disable the software configuration. You should comment out the HA properties rather than deleting them.
2. Move the NameNode metadata directories on the standby NameNode. The location of these directories is configured via the configuration options dfs.namenode.name.dir and/or dfs.namenode.edits.dir. Move them to a backup location; after the upgrade completes, you can remove them.

Step 3: Upgrade to the latest CDH4 Release and Upgrade HDFS

Follow the instructions in the upgrade procedure, starting with Step 2: Download the CDH4 package on each of the hosts in your cluster. (You have already done the first step).

Step 4: Redeploy HA

1. Uncomment the properties you commented out in Step 2: Unconfigure HA.
2. Start the primary NameNode and the standby NameNode, following the instructions under HDFS High Availability Initial Deployment.

HDFS High Availability Administration

HA Administration using the haadmin command
Using the dfsadmin command when HA is enabled
HA Administration using the haadmin command

Now that your HA NameNodes are configured and started, you will have access to some additional commands to administer your HA HDFS cluster. Specifically, you should familiarize yourself with all of the subcommands of the hdfs haadmin command. Running this command without any additional arguments will display the following usage information:

```
Usage: DFSHAAdmin [-ns <nameserviceId>]
   [-transitionToActive <serviceId>]
   [-transitionToStandby <serviceId>]
   [-failover [--forcefence] [--forceactive] <serviceId> <serviceId>]
   [-getServiceState <serviceId>]
   [-checkHealth <serviceId>]
   [-help <command>]
```

This page describes high-level uses of each of these subcommands. For specific usage information of each subcommand, you should run hdfs haadmin -help <command>.

**transitionToActive and transitionToStandby**

`transitionToActive` and `transitionToStandby` - transition the state of the given NameNode to Active or Standby

These subcommands cause a given NameNode to transition to the Active or Standby state, respectively. These commands do not attempt to perform any fencing, and thus should rarely be used. Instead, you should almost always use the hdfs haadmin -failover subcommand.

**failover**

`failover` - initiate a failover between two NameNodes

This subcommand causes a failover from the first provided NameNode to the second. If the first NameNode is in the Standby state, this command simply transitions the second to the Active state without error. If the first NameNode is in the Active state, an attempt will be made to gracefully transition it to the Standby state. If this fails, the fencing methods (as configured by dfs.ha.fencing.methods) will be attempted in order until one of the methods succeeds. Only after this process will the second NameNode be transitioned to the Active state. If no fencing method succeeds, the second NameNode will not be transitioned to the Active state, and an error will be returned.
Configuring other CDH components to use HDFS HA

getServiceState

getServiceState - determine whether the given NameNode is Active or Standby

Connect to the provided NameNode to determine its current state, printing either "standby" or "active" to STDOUT appropriately. This subcommand might be used by cron jobs or monitoring scripts which need to behave differently based on whether the NameNode is currently Active or Standby.

checkHealth

cHECKHealth - check the health of the given NameNode

Connect to the provided NameNode to check its health. The NameNode is capable of performing some diagnostics on itself, including checking if internal services are running as expected. This command will return 0 if the NameNode is healthy, non-zero otherwise. One might use this command for monitoring purposes.

Note

The checkHealth command is not yet implemented, and at present will always return success, unless the given NameNode is completely down.

Using the dfsadmin command when HA is enabled

When you use the dfsadmin command with HA enabled, you should use the -fs option to specify a particular NameNode using the RPC address, or service RPC address, of the NameNode. Not all operations are permitted on a standby NameNode. If the specific NameNode is left unspecified, only the operation to set quotas (-setQuota, -clrQuota, -setSpaceQuota, -clrSpaceQuota), report basic file system information (-report), and check upgrade progress (-upgradeProgress) will failover and perform the requested operation on the active NameNode. The "refresh" options (-refreshNodes, -refreshServiceAcl, -refreshUserToGroupsMappings, and -refreshSuperUserGroupsConfiguration) must be run on both the active and standby NameNodes.

Configuring other CDH components to use HDFS HA

You can use the HDFS High Availability NameNodes with other components of CDH such as HBase by changing the distributed file system URI to the name specified in the dfs.nameservices property. These clients must also have access to hdfs-site.xml's dfs.client.* settings to properly use HA.
Configuring HBase to use HDFS HA

For example, suppose you have the HDFS HA property `dfs.nameservices` set to `ha-nn`. To configure HBase to use the HA NameNodes, specify that same value as part of your `hdfs-site.xml`'s `hbase.rootdir` value.

```
<!-- Configure HBase to use the HA NameNode nameservice -->
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://ha-nn/hbase</value>
</property>
```

HBase-HDFS HA Troubleshooting

Problem: HMasters fail to start.

Solution: Check for this error in the `hmaster` logs:

```
java.lang.IllegalArgumentException: java.net.UnknownHostException: ha-nn
  at org.apache.hadoop.security.SecurityUtil.buildTokenService(SecurityUtil.java:431)
  at org.apache.hadoop.hdfs.NameNodeProxies.createNonHAProxy(NameNodeProxies.java:161)
  at org.apache.hadoop.hdfs.NameNodeProxies.createProxy(NameNodeProxies.java:126)
...
```

If so, verify that Hadoop's `hdfs-site.xml` and `core-site.xml` files are in your `hbase/conf` directory. This may be necessary if you put your configurations in non-standard places.