Cloudera Runtime 7.1.5

HDFS Overview

Date published: 2020-02-20
Date modified: 2020-11-30

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction 4
Overview of HDFS 4

NameNodes 4
Moving NameNode roles 5
Moving highly available NameNode, failover controller, and JournalNode roles using the Migrate Roles wizard 5
Moving a NameNode to a different host using Cloudera Manager 6
Sizing NameNode heap memory 7
Environment variables for sizing NameNode heap memory 7
Monitoring heap memory usage 7
Files and directories 7
Disk space versus namespace 8
Replication 8
Examples of estimating NameNode heap memory 9
Remove or add storage directories for NameNode data directories 10

DataNodes 10
How NameNode manages blocks on a failed DataNode 10
Replace a disk on a DataNode host 11
Remove a DataNode 11
Fixing block inconsistencies 12
Add storage directories using Cloudera Manager 12
Remove storage directories using Cloudera Manager 13
Configuring storage balancing for DataNodes 13
Configure storage balancing for DataNodes using Cloudera Manager 13
Perform a disk hot swap for DataNodes using Cloudera Manager 14

JournalNodes 15
Moving the JournalNode edits directory for a role group using Cloudera Manager 15
Moving the JournalNode edits directory for a role instance using Cloudera Manager 16
Synchronizing the contents of JournalNodes 17

Multiple NameNodes overview 17
Multiple Namenodes configurations 18
Known issue and its workaround 18
Adding multiple namenodes using the HDFS service 18

Introduction

Hadoop Distributed File System (HDFS) is a Java-based file system that provides scalable and reliable data storage. An HDFS cluster contains a NameNode to manage the cluster namespace and DataNodes to store data.

Overview of HDFS

Hadoop Distributed File System (HDFS) is a Java-based file system for storing large volumes of data. Designed to span large clusters of commodity servers, HDFS provides scalable and reliable data storage.

HDFS forms the data management layer of Apache Hadoop. YARN provides the resource management while HDFS provides the storage.

HDFS is a scalable, fault-tolerant, distributed storage system that works closely with a wide variety of concurrent data access applications. By distributing storage and computation across many servers, the combined storage resource grows linearly with demand.

Components of an HDFS cluster

An HDFS cluster contains the following main components: a NameNode and DataNodes.

The NameNode manages the cluster metadata that includes file and directory structures, permissions, modifications, and disk space quotas. The file content is split into multiple data blocks, with each block replicated at multiple DataNodes.

The NameNode actively monitors the number of replicas of a block. In addition, the NameNode maintains the namespace tree and the mapping of blocks to DataNodes, holding the entire namespace image in RAM.

High-Availability (HA) clusters contain a standby for the active NameNode to avoid a single point of failure. These clusters use JournalNodes to synchronize the active and standby NameNodes.

Benefits of HDFS

HDFS provides the following benefits as a result of which data is stored efficiently and is highly available in the cluster:

- Rack awareness: A node's physical location is considered when allocating storage and scheduling tasks.
- Minimal data motion: Hadoop moves compute processes to the data on HDFS. Processing tasks can occur on the physical node where the data resides. This significantly reduces network I/O and provides very high aggregate bandwidth.
- Utilities: Dynamically diagnose the health of the file system and rebalance the data on different nodes.
- Version rollback: Allows operators to perform a rollback to the previous version of HDFS after an upgrade, in case of human or systemic errors.
- Standby NameNode: Provides redundancy and supports high availability (HA).
- Operability: HDFS requires minimal operator intervention, allowing a single operator to maintain a cluster of thousands of nodes

Related Information

Apache Hadoop HDFS

NameNodes

NameNodes maintain the namespace tree for HDFS and a mapping of file blocks to DataNodes where the data is stored.

A simple HDFS cluster can have only one primary NameNode, supported by a secondary NameNode that periodically compresses the NameNode edits log file that contains a list of HDFS metadata modifications. This reduces the amount of disk space consumed by the log file on the NameNode, which also reduces the restart time for the primary NameNode. A high-availability cluster contains two NameNodes: active and standby.

Moving NameNode roles

Depending on your cluster configuration, you can follow either of the two approaches to migrate the NameNode role: using the Cloudera Manager wizard in a highly available cluster to automate the migration process, or manually moving the NameNode role to a new host.

Important: Both procedures require cluster downtime.

Moving highly available NameNode, failover controller, and JournalNode roles using the Migrate Roles wizard

The Migrate Roles wizard allows you to move roles of a highly available HDFS service from one host to another. You can use it to move NameNode, JournalNode, and Failover Controller roles.

Before you begin

- This procedure requires cluster downtime, not a shutdown. The services discussed in this list must be running for the migration to complete.
- The configuration of HDFS and services that depend on it must be valid.
- The destination host must be commissioned and healthy.
- The NameNode must be highly available using quorum-based storage.
- HDFS automatic failover must be enabled, and the cluster must have a running ZooKeeper service.
- If a Hue service is present in the cluster, its HDFS Web Interface Role property must refer to an HttpFS role, not to a NameNode role.
- A majority of configured JournalNode roles must be running.
- The Failover Controller role that is not located on the source host must be running.
- On hosts running active and standby NameNodes, back up the data directories.
- On hosts running JournalNodes, back up the JournalNode edits directory.
- If the source host is not functioning properly, or is not reliably reachable, decommission the host.
- If CDP and HDFS metadata was recently upgraded, and the metadata upgrade was not finalized, finalize the metadata upgrade.

About this task

Note: Nameservice federation (multiple namespaces) is not supported.

Procedure

1. If the host to which you want to move the NameNode is not in the cluster, add the required host.
2. Go to the HDFS service.
3. Click the Instances tab.
4. Click the Migrate Roles button.

5. Click the Source Host text field and specify the host running the roles to migrate.

   In the Search field optionally enter hostnames to filter the list of hosts and click Search.

   Select the checkboxes next to the desired host. The list of available roles to migrate displays. Clear any roles you do not want to migrate. When migrating a NameNode, the co-located Failover Controller must be migrated as well.
6. Click the Destination Host text field and specify the host to which the roles will be migrated.

   On destination hosts, indicate whether to delete data in the NameNode data directories and JournalNode edits directory. If you choose not to delete data and such role data exists, the Migrate Roles command will not complete successfully.
7. Acknowledge that the migration process incurs service unavailability by selecting the Yes, I am ready to restart the cluster now checkbox.
8. Click Continue. The Command Progress screen displays listing each step in the migration process.
9. When the migration completes, click Finish.

What to do next

After moving a NameNode, if you have a Hive or Impala service, perform the following steps:

1. Go to the Hive service.
2. Stop the Hive service.
3. Select Actions Update Hive Metastore NameNodes .
4. If you have an Impala service, restart the Impala service or run an INVALIDATE METADATA query.

Note: The Update Hive Metastore NameNodes step only needs to be run if you have non-HA namenodes, or if the nameservice identifier is changed.

Moving a NameNode to a different host using Cloudera Manager

You can use Cloudera Manager to manually move a NameNode from one host to another.

Before you begin

This procedure requires cluster downtime.

Procedure

1. If the host to which you want to move the NameNode is not in the cluster, add the required host.
2. Stop all the cluster services.
3. Make a backup of the dfs.name.dir directories on the existing NameNode host.

   Make sure you back up the fsimage and edits files. They should be the same across all of the directories specified by the dfs.name.dir property.
4. Copy the files you backed up from dfs.name.dir directories on the old NameNode host to the host where you want to run the NameNode.
5. Go to the HDFS service.
6. Click the Instances tab.
7. Select the checkbox next to the NameNode role instance and then click the Delete button. Click Delete again to confirm.
8. In the Review configuration changes page that appears, click Skip.
9. Click Add Role Instances to add a NameNode role instance.
10. Select the host where you want to run the NameNode and then click Continue.
11. Specify the location of the dfs.name.dir directories where you copied the data on the new host, and then click Accept Changes.

12. Start the cluster services.

    After the HDFS service has started, Cloudera Manager distributes the new configuration files to the DataNodes, which will be configured with the IP address of the new NameNode host.

Sizing NameNode heap memory

Each workload has a unique byte-distribution profile. Some workloads can use the default JVM settings for heap memory and garbage collection, but others require tuning. You can size your NameNode JVM if the dynamic heap settings cause a bottleneck.

All Hadoop processes run on a Java Virtual Machine (JVM). Each daemon runs on its own JVM across a cluster of hosts.

The legacy NameNode configuration is one active (and primary) NameNode for the entire namespace and one Secondary NameNode for checkpoints (but not failover). The recommended high-availability configuration replaces the Secondary NameNode with a standby NameNode that prevents a single point of failure. Each NameNode uses its own JVM.

Note: For optimal performance, a maximum of 300 million files on the NameNode is recommended.

Environment variables for sizing NameNode heap memory

You can configure the values of *HADOOP_HEAPSIZE* and *HADOOP_NAMENODE_OPTS* to size the NameNode heap memory.

HADOOP_HEAPSIZE sets the JVM heap size for all Hadoop project servers such as HDFS, YARN, and MapReduce. *HADOOP_HEAPSIZE* is an integer passed to the JVM as the maximum memory (Xmx) argument. For example:

```
HADOOP_HEAPSIZE=1024
```

HADOOP_NAMENODE_OPTS is specific to the NameNode and sets all JVM flags, which must be specified. *HADOOP_NAMENODE_OPTS* overrides the *HADOOP_HEAPSIZE* Xmx value for the NameNode. For example:

```
HADOOP_NAMENODE_OPTS=-Xms1024m -Xmx1024m -XX:+UseParNewGC -XX:+UseConcMarkSw
eepGC -XX:CMSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled -X
X:+PrintTenuringDistribution -XX:OnOutOfMemoryError={{AGENT_COMMON_DIR}}/kil
lparent.sh
```

Both *HADOOP_NAMENODE_OPTS* and *HADOOP_HEAPSIZE* are stored in /etc/hadoop/conf/hadoop-env.sh.

Monitoring heap memory usage

You can use several ways to monitor the heap memory usage: Cloudera Manager, NameNode web UI, or the command line.

- Cloudera Manager: Look at the NameNode chart for heap memory usage. If you need to build the chart from scratch, run:

  ```
  select jvm_max_memory_mb, jvm_heap_used_mb where roleType="NameNode"
  ```

- NameNode Web UI: Scroll down to the Summary and look for "Heap Memory used."
- Command line: Generate a heap dump.

Files and directories

Persistence of HDFS metadata is implemented using fsimage file and edits files.

Attention:

Do not attempt to modify metadata directories or files. Unexpected modifications can cause HDFS downtime, or even permanent data loss. This information is provided for educational purposes only.

Persistence of HDFS metadata broadly consist of two categories of files:

fsimage

Contains the complete state of the file system at a point in time. Every file system modification is assigned a unique, monotonically increasing transaction ID. An fsimage file represents the file system state after all modifications up to a specific transaction ID.

edits file

Contains a log that lists each file system change (file creation, deletion or modification) that was made after the most recent fsimage.

Checkpointing is the process of merging the content of the most recent fsimage, with all edits applied after that fsimage is merged, to create a new fsimage. Checkpointing is triggered automatically by configuration policies or manually by HDFS administration commands.

Disk space versus namespace

The HDFS default block size (*dfs.blocksize*) is set to 128 MB. Each namespace object on the NameNode consumes approximately 150 bytes.

On DataNodes, data files are measured by disk space consumed—the actual data length—and not necessarily the full block size. For example, a file that is 192 MB consumes 192 MB of disk space and not some integral multiple of the block size. Using the default block size of 128 MB, a file of 192 MB is split into two block files, one 128 MB file and one 64 MB file. On the NameNode, namespace objects are measured by the number of files and blocks. The same 192 MB file is represented by three namespace objects (1 file inode + 2 blocks) and consumes approximately 450 bytes of memory.

Large files split into fewer blocks generally consume less memory than small files that generate many blocks. One data file of 128 MB is represented by two namespace objects on the NameNode (1 file inode + 1 block) and consumes approximately 300 bytes of memory. By contrast, 128 files of 1 MB each are represented by 256 namespace objects (128 file inodes + 128 blocks) and consume approximately 38,400 bytes. The optimal split size, then, is some integral multiple of the block size, for memory management as well as data locality optimization.

By default, Cloudera Manager allocates a maximum heap space of 1 GB for every million blocks (but never less than 1 GB). How much memory you actually need depends on your workload, especially on the number of files, directories, and blocks generated in each namespace. If all of your files are split at the block size, you could allocate 1 GB for every million files. But given the historical average of 1.5 blocks per file (2 block objects), a more conservative estimate is 1 GB of memory for every million blocks.

Important: Cloudera recommends 1 GB of NameNode heap space per million blocks to account for the namespace objects, necessary bookkeeping data structures, and the remote procedure call (RPC) workload. In practice, your heap requirements will likely be less than this conservative estimate.

Replication

The default block replication factor (*dfs.replication*) is three. Replication affects disk space but not memory consumption.

Replication changes the amount of storage required for each block but not the number of blocks. If one block file on a DataNode, represented by one block on the NameNode, is replicated three times, the number of block files is tripled but not the number of blocks that represent them.

With replication off, one file of 192 MB consumes 192 MB of disk space and approximately 450 bytes of memory. If you have one million of these files, or 192 TB of data, you need 192 TB of disk space and, without considering the RPC workload, 450 MB of memory: (1 million inodes + 2 million blocks) * 150 bytes. With default replication on, you need 576 TB of disk space: (192 TB * 3) but the memory usage stay the same, 450 MB. When you account for

bookkeeping and RPCs, and follow the recommendation of 1 GB of heap memory for every million blocks, a much safer estimate for this scenario is 2 GB of memory (with or without replication).

Examples of estimating NameNode heap memory

You can estimate the heap memory by consider either the number of file inodes and blocks, or the cluster capacity.

Example 1: Estimating NameNode heap memory used

Alice, Bob, and Carl each have 1 GB (1024 MB) of data on disk, but sliced into differently sized files. Alice and Bob have files that are some integral of the block size and require the least memory. Carl does not and fills the heap with unnecessary namespace objects.

Alice: 1 x 1024 MB file

- 1 file inode
- 8 blocks (1024 MB / 128 MB)

Total = 9 objects * 150 bytes = 1,350 bytes of heap memory

Bob: 8 x 128 MB files

- 8 file inodes
- 8 blocks

Total = 16 objects * 150 bytes = 2,400 bytes of heap memory

Carl: 1,024 x 1 MB files

- 1,024 file inodes
- 1,024 blocks

Total = 2,048 objects * 150 bytes = 307,200 bytes of heap memory

Example 2: Estimating NameNode heap memory needed

In this example, memory is estimated by considering the capacity of a cluster. Values are rounded. Both clusters physically store 4800 TB, or approximately 36 million block files (at the default block size). Replication determines how many namespace blocks represent these block files.

Cluster A: 200 hosts of 24 TB each = 4800 TB.

- Blocksize=128 MB, Replication=1
- Cluster capacity in MB: 200 * 24,000,000 MB = 4,800,000,000 MB (4800 TB)
- Disk space needed per block: 128 MB per block * 1 = 128 MB storage per block
- Cluster capacity in blocks: 4,800,000,000 MB / 128 MB = 36,000,000 blocks

At capacity, with the recommended allocation of 1 GB of memory per million blocks, Cluster A needs 36 GB of maximum heap space.

Cluster B: 200 hosts of 24 TB each = 4800 TB.

- Blocksize=128 MB, Replication=3
- Cluster capacity in MB: 200 * 24,000,000 MB = 4,800,000,000 MB (4800 TB)
- Disk space needed per block: 128 MB per block * 3 = 384 MB storage per block
- Cluster capacity in blocks: 4,800,000,000 MB / 384 MB = 12,000,000 blocks

At capacity, with the recommended allocation of 1 GB of memory per million blocks, Cluster B needs 12 GB of maximum heap space.

Both Cluster A and Cluster B store the same number of block files. In Cluster A, however, each block file is unique and represented by one block on the NameNode; in Cluster B, only one-third are unique and two-thirds are replicas.

Remove or add storage directories for NameNode data directories

This is a cluster administration task where existing Namenode DataDir must be changed. Perform the following steps to change the Namenode directories with downtime.

Procedure

1. Take a backup of the existing DataDir paths on NNs. For instance, for each NN, if there are two DataDir paths, taking backup of one DataDir is sufficient.
2. Create new mounts where the Namenode data will reside and ensure the permissions and ownerships are all same as that of the old Namenode data directories.
3. Shutdown the Standby Namenode.
4. Log in to the host and copy the NN DataDir content from old mount to new mount.
5. Verify the permission and ownership of fsimage* edits* VERSION files.
6. Navigate to Cloudera Manager > HDFS > Instances > click on the SBNN role (stopped one) > Configuration > search for dfs.namenode.name.dir > change the mount path to new one > Save Changes
7. Start the SBNN NameNode.
8. Wait for the SBNN to start and is out of safe mode.
9. Navigate to Cloudera Manager > HDFS > Configuration > Federation and High Availability > Actions > Manual Failover > select SBNN option > click Manual Failover > wait until the failover is complete.
10. You must verify using the NN webUIs if the previous SBNN node is now the ANN and ANN node is now the SBNN.
11. Repeat steps 2-7 on the new SBNN which was previous ANN.

DataNodes

DataNodes store data in a Hadoop cluster and is the name of the daemon that manages the data. File data is replicated on multiple DataNodes for reliability and so that localized computation can be executed near the data.

Within a cluster, DataNodes should be uniform. If they are not uniform, issues can occur. For example, DataNodes with less memory fill up more quickly than DataNodes with more memory, which can result in job failures.

Important: The default replication factor for HDFS is three. That is, three copies of data are maintained at all times. Cloudera recommends that you do not configure a lower replication factor when you have at least three DataNodes. A lower replication factor may lead to data loss.

How NameNode manages blocks on a failed DataNode

A DataNode is considered dead after a set period without any heartbeats (10.5 minutes by default).

When this happens, the NameNode performs the following actions to maintain the configured replication factor (3x replication by default):

1. The NameNode determines which blocks were on the failed DataNode.
2. The NameNode locates other DataNodes with copies of these blocks.
3. The DataNodes with block copies are instructed to copy those blocks to other DataNodes to maintain the configured replication factor.

Keep the following in mind when working with dead DataNodes:

- If the DataNode failed due to a disk failure, follow the procedure in Replace a disk on a DataNode host on page 11 or Perform a disk hot swap for DataNodes using Cloudera Manager on page 14 to bring a repaired

DataNode back online. If a DataNode failed to heartbeat for other reasons, they need to be recommissioned to be added back to the cluster.

- If a DataNode fails to heartbeat for reasons other than disk failure, it needs to be recommissioned to be added back to the cluster.
- If a DataNode rejoins the cluster, there is a possibility for surplus replicas of blocks that were on that DataNode. The NameNode will randomly remove excess replicas adhering to Rack-Awareness policies.

Replace a disk on a DataNode host

In your CDP Private Cloud Base cluster, you can replace faulty disks on the DataNode host. You must stop all the managed services and decommission the DataNode role instance before replacing the faulty disk.

Before you begin

Procedure

1. Stop managed services.
2. Decommission the DataNode role instance.
3. Replace the failed disk.
4. Recommission the DataNode role instance.
5. Run the HDFS fsck utility to validate the health of HDFS.

   The utility normally reports over-replicated blocks immediately after a DataNode is reintroduced to the cluster, which is automatically corrected over time.
6. Start managed services.

Remove a DataNode

Before removing a DataNode, ensure that all the prerequisites for removal are satisfied.

Before you begin

- The number of DataNodes in your cluster must be greater than or equal to the replication factor you have configured for HDFS. (This value is typically 3.)

  In order to satisfy this requirement, add the DataNode role on other hosts as required and start the role instances before removing any DataNodes.
- Ensure the DataNode that is to be removed is running.

Procedure

1. Decommission the DataNode role.

   When asked to select the role instance to decommission, select the DataNode role instance.

   The decommissioning process moves the data blocks to the other available DataNodes.

   Important: There must be at least as many DataNodes running as the replication factor or the decommissioning process will not complete.
2. Stop the DataNode role.

   When asked to select the role instance to stop, select the DataNode role instance.

3. Verify the integrity of the HDFS service.
   a) Run the following command to identify any problems in the HDFS file system:

   ```
   hdfs fsck /
   ```

   b) Fix any errors reported by the fsck command.

   If required, create a Cloudera support case.
4. After all errors are resolved, perform the following steps.
   a) Remove the DataNode role.
   b) Manually remove the DataNode data directories.

   You can determine the location of these directories by examining the DataNode Data Directory property in the HDFS configuration. In Cloudera Manager, go to the HDFS service, select the Configuration tab and search for the property.

Fixing block inconsistencies

You can use the output of `hdfs fsck` or `hdfs dfsadmin -report` commands for information about inconsistencies with the HDFS data blocks such as missing, misreplicated, or underreplicated blocks. You can adopt different methods to address these inconsistencies.

Missing blocks: Ensure that the DataNodes in your cluster and the disks running on them are healthy. This should help in recovering those blocks that have recoverable replicas (indicated as missing blocks). If a file contains corrupt or missing blocks that cannot be recovered, then the file would be missing data, and all this data starting from the missing block becomes inaccessible through the CLI tools and FileSystem API. In most cases, the only solution is to delete the data file (by using the `hdfs fsck <path> -delete` command) and recover the data from another source.

Important: You must exercise caution when deleting files to ensure that you do not lose any important data. Cloudera recommends that you contact support for further assistance or any guidance on deleting files.

Underreplicated blocks: HDFS automatically attempts to fix this issue by replicating the underreplicated blocks to other DataNodes and match the replication factor. If the automatic replication does not work, you can run the HDFS Balancer to address the issue.

Misreplicated blocks: Run the `hdfs fsck -replicate` command to trigger the replication of misreplicated blocks. This ensures that the blocks are correctly replicated across racks in the cluster.

Add storage directories using Cloudera Manager

You can add a new storage directory and specify the storage type using Cloudera Manager.

Procedure

1. Go to the HDFS service.
2. Click the Configuration tab.
3. Select Scope DataNode .
4. Add the new storage directory to the DataNode Data Directory property.

   To specify the storage type for the HDFS heterogeneous storage, add the storage type, surrounded by brackets, at the front of the path. For example: [SSD]/data/example_dir/.
5. Enter a Reason for change, and then click Save Changes to commit the changes.
6. Restart the DataNode.

   Important: You must restart the DataNodes for heterogeneous storage configuration changes to take effect.

Remove storage directories using Cloudera Manager

You can use Cloudera Manager to remove existing storage directories and specify new directories.

Procedure

1. Stop the cluster.
2. Go to the HDFS service.
3. Click the Configuration tab.
4. Select Scope DataNode .
5. Remove the current directories and add new ones to the DataNode Data Directory property.
6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Copy the contents under the old directory to the new directory.
8. Start the cluster.

Configuring storage balancing for DataNodes

You can configure HDFS to distribute writes on each DataNode in a manner that balances out available storage among that DataNode's disk volumes.

By default a DataNode writes new block replicas to disk volumes solely on a round-robin basis. You can configure a volume-choosing policy that causes the DataNode to take into account how much space is available on each volume when deciding where to place a new replica.

You can configure

- how much DataNode volumes are allowed to differ in terms of bytes of free disk space before they are considered imbalanced, and
- what percentage of new block allocations will be sent to volumes with more available disk space than others.

Configure storage balancing for DataNodes using Cloudera Manager

You must configure certain dfs.datanode properties to enable storage balancing for DataNodes.

Procedure

1. Go to the HDFS service.
2. Click the Configuration tab.
3. Select ScopeDataNode.
4. Select CategoryAdvanced.

5. Configure the following properties (you can use the Search box to locate the properties).

Property	Value	Description
`dfs.datanode.fsdataset.volume.choosing.policy`	`org.apache.hadoop.hdfs.server.datanode.fsdataset.AvailableSpaceVolumeChoosingPolicy`	Enables storage balancing among the DataNode's volumes.
`dfs.datanode.available-space-volume-choosing-policy.balanced-space-threshold`	10737418240 (default)	The amount by which volumes are allowed to differ from each other in terms of bytes of free disk space before they are considered imbalanced. The default is 10737418240 (10 GB). If the free space on each volume is within this range of the other volumes, the volumes will be considered balanced and block assignments will be done on a pure round-robin basis.
`dfs.datanode.available-space-volume-choosing-policy.balanced-space-preference-fraction`	0.75 (default)	What proportion of new block allocations will be sent to volumes with more available disk space than others. The allowable range is 0.0-1.0, but set it in the range 0.5 - 1.0 (that is, 50-100%), since there should be no reason to prefer that volumes with less available disk space receive more block allocations.

6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Restart the role.

Perform a disk hot swap for DataNodes using Cloudera Manager

You can replace disks on your CDP Private Cloud Base cluster without shutting down a DataNode. This is referred to as hot swap.

About this task

Caution:

- Hot swap can only add disks with empty data directories.
- Removing a disk does not move the data off the disk, which could potentially result in data loss.
- Do not perform hot swap on multiple hosts at the same time.

Procedure

1. Configure data directories to remove the disk you are swapping out.
   a) Go to the HDFS service.
   b) Click the Instances tab.
   c) In the Role Type column, click on the affected DataNode.
   d) Click the Configuration tab.
   e) Select Scope DataNode .
   f) Select Category Main .
   g) Change the value of the DataNode Data Directory property to remove the directories that are mount points for the disk you are removing.

      Warning: Change the value of this property only for the specific DataNode instance where you are planning to hot swap the disk. Do not edit the role group value for this property. Doing so will cause data loss.
2. Enter a Reason for change, and then click Save Changes to commit the changes.
3. Refresh the affected DataNode. Select ActionsRefresh DataNode configuration.
4. Remove the old disk and add the replacement disk.
5. Change the value of the DataNode Data Directory property to add back the directories that are mount points for the disk you added.
6. Enter a Reason for change, and then click Save Changes to commit the changes.
7. Refresh the affected DataNode. Select Actions > Refresh DataNode configuration.
8. Run the hdfs fsck / command to validate the health of HDFS.

JournalNodes

High-availability clusters use JournalNodes to synchronize active and standby NameNodes. The active NameNode writes to each JournalNode with changes, or "edits," to HDFS namespace metadata. During failover, the standby NameNode applies all edits from the JournalNodes before promoting itself to the active state.

Moving the JournalNode edits directory for a role group using Cloudera Manager

Depending on your requirements, you can change the location of the edits directory for each JournalNode in the JournalNode Default Group.

Procedure

1. Stop all services on the cluster in Cloudera Manager.
   a) Go to the Cluster.
   b) Select Actions Stop .
2. Find the list of JournalNode hosts.
   a) Go to the HDFS service.
   b) Click JournalNode under Status Summary.

3. Move the location of each JournalNode (jn) directory at the command line.
   a) Connect to each host with a JournalNode.
   b) Per host, copy the JournalNode (jn) directory to its new location with the -a option to preserve permissions.

```
cp -a /<old_path_to_jn_dir>/jn /<new_path_to_jn_dir>/jn
```

   c) Per host, rename the old jn directory to avoid confusion.

```
mv /<old_path_to_jn_dir>/jn /<old_path_to_jn_dir>/jn_to_delete
```

4. Reconfigure the JournalNode Default Group.
   a) Go to the HDFS service.
   b) Click the Configuration tab.
   c) Click JournalNode under Scope.
   d) Set dfs.journalnode.edits.dir to the path of the new jn directory for all JournalNodes in the group.
   e) Click Save Changes.
5. Redeploy the client configuration for the cluster.
   a) Go to the Cluster.
   b) Select Actions Deploy Client Configuration .
6. Start all services on the cluster by selecting Actions Start .
7. Delete the old jn_to_delete directories from the command line.

Moving the JournalNode edits directory for a role instance using Cloudera Manager

Depending on your requirements, you can change the location of the edits directory for one JournalNode instance.

Procedure

1. Reconfigure the JournalNode Edits Directory.
   a) Go to the HDFS service in Cloudera Manager.
   b) Click JournalNode under Status Summary.
   c) Click the JournalNode link for the instance you are changing.
   d) Click the Configuration tab.
   e) Set dfs.journalnode.edits.dir to the path of the new jn directory.
2. Move the location of the JournalNode (jn) directory at the command line.
   a) Connect to host of the JournalNode.
   b) Copy the JournalNode (jn) directory to its new location with the -a option to preserve permissions.

```
cp -a /<old_path_to_jn_dir>/jn /<new_path_to_jn_dir>/jn
```

   c) Rename the old jn directory to avoid confusion.

```
mv /<old_path_to_jn_dir>/jn /<old_path_to_jn_dir>/jn_to_delete
```

3. Redeploy the HDFS client configuration.
   a) Go to the HDFS service.
   b) Select Actions Deploy Client Configuration .
4. Perform a rolling restart for HDFS by selecting Actions Rolling Restart .

   Use the default settings.
5. From the command line, delete the old jn_to_delete directory.

Synchronizing the contents of JournalNodes

You can synchronize the contents of the JournalNodes in your CDP Private Cloud Base cluster. Enabling this capability helps in maintaining consistency in the contents of all the JournalNodes across the cluster. For example, a JournalNode whose contents go out of synchronization can automatically copy the contents of the other JournalNodes in the cluster to ensure consistency.

Before you begin

The cluster on which you want to enable JournalNode synchronization must have High Availability enabled.

Procedure

1. Go to the HDFS service.
2. Click the Instances tab.
3. Click the link labeled NameNode (Active).

   The NameNode instance page displays.
4. Go to the Process tab.
5. Click the hdfs-site.xml link under hdfs/hdfs.sh ["namenode"] to open a read-only copy of the file in a separate browser window.
6. Note the URI value set for JournalNodes in the various dfs.namenode.shared.edits.dir parameters.

   The URI value should be in the format:

   ```
   qjournal://host1:port1;host2:port2;host3:port3/journalId
   ```
7. Return to the HDFS service page and click the Configuration tab.

   Note: Ensure that you click the Configuration tab on the HDFS service page, *not* on the NameNode role page.
8. Search for the HDFS Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml configuration option.
9. Add the key dfs.namenode.shared.edits.dir to the selected configuration option and provide the value noted in Step 6.
10. Click Save Changes.
11. Restart HDFS.

Multiple NameNodes overview

This feature provides CDP Private Cloud Base the ability to use multiple standby namenodes.

Prerequisites

- Cloudera Manager version must be 7.7.1 or higher
- CDP parcel version must be 7.1.8 or higher
- HDFS should have High Availability enabled
- The latest upgrade of HDFS must be finalized
- The cluster must be healthy according to Cloudera Manager and without any stale configurations

This feature is enabled only for the 7.1.8 clusters. This requires the High Availability feature enabled. After configuring an additional NameNode to the cluster, you cannot roll back either the Cloudera Manager upgrade or the CDP parcel upgrade. You must add an additional namenode after the upgrade to 718 is finalized.

Multiple Namenodes configurations

This feature is not enabled by default in Cloudera Manager. You must first enable this feature in Cloudera Manager configuration. You can then configure HDFS service with multiple standby NameNodes.

Enabling multiple namenode support feature in Cloudera Manager

Procedure

1. Navigate to the /etc/default/cloudera-scm-server file.
2. Add export CMF_FF_HDFS_MULTIPLE_SNN=true at the end of the cloudera-scm-server file.
3. Restart the Cloudera Manager Service.

Known issue and its workaround

Before you proceed further configuring additional namenodes, you must be aware of the following known issue.

Known Issue Description

In case the JournalNodes were restarted on a cluster recently, the Add new NameNode wizard for HDFS service might fail to bootstrap the new NameNode, if there was not a new fsImage created since the restart of the JournalNodes. However, while restarting the JournalNodes the edit logs were rolled in the system.

Known Issue Workaround

If the bootstrap fails during the Add new NameNode wizard, you must perform the following steps:

1. Delete the newly added NameNode and FailoverController
2. Move the active HDFS NameNode to safe mode
3. Perform the Save Namespace operation on the active HDFS NameNode
4. Leave safe mode on the active HDFS NameNode
5. Try to add the new NameNode again

Note: Entering safe mode disables write operation to HDFS and causes a service disruption. If you can not afford to enter safe mode, just delete the newly added NameNode and FailoverController in the HDFS service. You must wait until HDFS automatically creates a new fsImage and try adding the new NameNode again with the wizard.

Adding multiple namenodes using the HDFS service

You can add multiple namenodes using the HDFS service in Cloudera Manager.
You must ensure that Cloudera Manager is in a clean state without any health issues and no stale configurations on the cluster. If there are any health issues or stale configurations, then you must restart the cluster separately from the command and not within the command. For this, you must clear the Rolling Restart HDFS and all dependent services to activate the changes option in step 10b. Also, if there are any health issues or stale configurations that cannot be cleared from the cluster, you must restart the cluster separately.

Procedure

1. Log in to Cloudera Manager UI
2. Navigate to Clusters
3. Select the HDFS service
4. Go to Instances

5. Click Federation and High Availability
6. Click Actions
7. Click Add Additional NameNode(s). Add Additional NameNode wizard opens.
8. In the Assign Roles page, select a host which is not assigned to any namenode.
9. Click Continue.
10. In the Review Changes page, you can specify the namenode data directory.
    a) If you want to delete the data present in the namenode data directory you have provided, then you must select the Clear any existing data present in name directories of Standby NameNode option. Remember to take a backup of the namenode data directory.
    b) If you want to restart HDFS service and its dependencies, then you must select the Rolling Restart HDFS and all dependent services to activate the changes option. Remember that if you have not enabled High Availability feature, the rolling restart may cause a cluster downtime. Note: If you do not select the Rolling Restart option, then you can click Continue and restart the cluster later.
11. Click Continue. This runs the required commands and adds the additional namenodes.
12. Click Finish.