

Bright Cluster Manager 7.1

Hadoop Deployment Manual

Revision: e861b62

Date: Mon Sep 24 2018



©2015 Bright Computing, Inc. All Rights Reserved. This manual or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of Bright Computing, Inc.

Trademarks

Linux is a registered trademark of Linus Torvalds. PathScale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of Novell, Inc. PGI is a registered trademark of NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, Inc. ScaleMP is a registered trademark of ScaleMP, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. Bright Computing, Inc. shall not be liable for technical or editorial errors or omissions which may occur in this document. Bright Computing, Inc. shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to Bright Computing, Inc.

The Bright Cluster Manager product principally consists of free software that is licensed by the Linux authors free of charge. Bright Computing, Inc. shall have no liability nor will Bright Computing, Inc. provide any warranty for the Bright Cluster Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the Bright Cluster Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the Bright Cluster Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Table of Contents	i
0.1 About This Manual	v
0.2 About The Manuals In General	v
0.3 Getting Administrator-Level Support	v
1 Introduction	1
1.1 What Is Hadoop About?	1
1.2 Available Hadoop Implementations	2
1.3 Further Documentation	2
1.4 Version Support Matrix	2
1.4.1 Apache Hadoop 1.2.1	3
1.4.2 Hortonworks HDP 1.3.11	4
1.4.3 Apache Hadoop 2.7.1	4
1.4.4 Cloudera CDH 4.6.0	4
1.4.5 Cloudera CDH 4.7.1	5
1.4.6 Cloudera CDH 5.2.4	5
1.4.7 Cloudera CDH 5.3.8	6
1.4.8 Cloudera CDH 5.4.8	6
1.4.9 Hortonworks HDP 2.1.15	6
1.4.10 Hortonworks HDP 2.2.8	7
1.4.11 Hortonworks HDP 2.3.2	7
1.4.12 Pivotal HD 2.1.0	8
1.4.13 Pivotal HD 3.0.1	8
2 Installing Hadoop	9
2.1 Command-line Installation Of Hadoop Using <code>cm-hadoop-setup -c <filename></code>	9
2.1.1 Usage	9
2.1.2 An Install Run	9
2.2 Ncurses Installation Of Hadoop Using <code>cm-hadoop-setup</code>	12
2.3 Avoiding Misconfigurations During Hadoop Installation	13
2.3.1 NameNode Configuration Choices	13
2.4 Installing Hadoop With Lustre	14
2.4.1 Lustre Internal Server Installation	14
2.4.2 Lustre External Server Installation	14
2.4.3 Lustre Client Installation	14
2.4.4 Lustre Hadoop Configuration	15
2.5 Hadoop Installation In A Cloud	17

3	Hadoop Cluster Management	19
3.1	Managing A Hadoop Instance With <code>cmgui</code>	19
3.1.1	The HDFS Instance Overview Tab	20
3.1.2	The HDFS Instance Settings Tab	20
3.1.3	The HDFS Instance HDFS Tab	23
3.1.4	The HDFS Instance MapReduce Or YARN Tab	23
3.1.5	The HDFS Instance HBase Tab	24
3.1.6	The HDFS Instance Zookeeper Tab	25
3.1.7	The HDFS Instance Spark Tab	25
3.1.8	The HDFS Instance More Tab	25
3.1.9	The HDFS Instance Hadoop Configuration Groups Tab	25
3.1.10	The HDFS Instance Monitoring Tab	31
3.1.11	The HDFS Instance Notes Tab	31
3.2	Managing A Hadoop Instance With <code>cmsh</code>	31
3.2.1	<code>cmsh</code> And <code>hadoop</code> Mode	31
3.2.2	<code>cmsh</code> And <code>configurationoverlay</code> Mode	36
3.2.3	<code>cmsh</code> And The <code>roleoverview</code> Command In <code>device</code> Mode	38
3.3	Hadoop Maintenance Operations With <code>cm-hadoop-maint</code>	38
4	Running Hadoop Jobs	41
4.1	Shakedown Runs	41
4.2	Example End User Job Run	43
5	Spark support in Bright Cluster Manager	45
5.1	Spark Installation In Bright Cluster Manager	45
5.1.1	Prerequisites For Spark Installation, And What Spark Installation Does	45
5.1.2	Spark Installation With <code>cm-spark-setup</code>	46
5.2	Spark Removal With <code>cm-spark-setup</code>	48
5.3	Using Spark	48
5.3.1	Using Spark In YARN Mode	48
5.3.2	Using Spark In Standalone Mode	48
6	Hadoop-related Projects	51
6.1	Accumulo	51
6.1.1	Accumulo Installation With <code>cm-accumulo-setup</code>	51
6.1.2	Accumulo Removal With <code>cm-accumulo-setup</code>	53
6.1.3	Accumulo MapReduce Example	53
6.2	Hive	53
6.2.1	Hive Installation With <code>cm-hive-setup</code>	54
6.2.2	Hive Removal With <code>cm-hive-setup</code>	55
6.2.3	Beeline	55
6.3	Kafka	56
6.3.1	Kafka Installation With <code>cm-kafka-setup</code>	56
6.3.2	Kafka Removal With <code>cm-kafka-setup</code>	57
6.4	Pig	57
6.4.1	Pig Installation With <code>cm-pig-setup</code>	57
6.4.2	Pig Removal With <code>cm-pig-setup</code>	58

6.4.3	Using Pig	58
6.5	Sqoop	58
6.5.1	Sqoop Installation With <code>cm-sqoop-setup</code>	58
6.5.2	Sqoop Removal With <code>cm-sqoop-setup</code>	59
6.6	Storm	60
6.6.1	Storm Installation With <code>cm-storm-setup</code>	60
6.6.2	Storm Removal With <code>cm-storm-setup</code>	61
6.6.3	Using Storm	61
A	Details And Examples Of Hadoop Configuration	63
A.1	Hadoop Components Activation And Deactivation Using Roles	63
A.2	Only The Enabled Hadoop Components And Roles Are Available For Activation From <code>cmgui</code> And <code>cmsh</code>	63
A.3	Example Of Role Priority Overrides In Configuration Groups With <code>cmsh</code>	64
A.4	Cloning Hadoop Configuration Groups In <code>cmgui</code> And <code>cmsh</code>	66
A.4.1	Cloning Hadoop Configuration Groups In <code>cmgui</code>	66
A.4.2	Cloning Hadoop Configuration Groups In <code>cmsh</code>	70
A.5	Considerations And Best Practices When Creating Or Cloning Hadoop Configurations . .	71

Preface

Welcome to the *Hadoop Deployment Manual* for Bright Cluster Manager 7.1.

0.1 About This Manual

This manual is aimed at helping cluster administrators install, understand, configure, and manage the Hadoop capabilities of Bright Cluster Manager. The administrator is expected to be reasonably familiar with the Bright Cluster Manager *Administrator Manual*.

0.2 About The Manuals In General

Regularly updated versions of the Bright Cluster Manager 7.1 manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at `http://support.brightcomputing.com/manuals`.

- The *Installation Manual* describes installation procedures for a basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.
- The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.
- The *OpenStack Deployment Manual* describes how to deploy OpenStack with Bright Cluster Manager.
- The *Hadoop Deployment Manual* describes how to deploy Hadoop with Bright Cluster Manager.
- The *UCS Deployment Manual* describes how to deploy the Cisco UCS server with Bright Cluster Manager.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: `<Alt>-<Backarrow>` in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications. The manuals also regularly incorporate customer feedback. Administrator and user input is greatly valued at Bright Computing. So any comments, suggestions or corrections will be very gratefully accepted at `manuals@brightcomputing.com`.

0.3 Getting Administrator-Level Support

Unless the Bright Cluster Manager reseller offers support, support is provided by Bright Computing over e-mail via `support@brightcomputing.com`. Section 10.2 of the *Administrator Manual* has more details on working with support.

1

Introduction

1.1 What Is Hadoop About?

Hadoop is the core implementation of a distributed data processing technology used for the analysis of very large and often unstructured datasets. The dataset size typically ranges from several terabytes to petabytes. The size and lack of structure of the dataset means that it cannot be stored or handled efficiently in regular relational databases, which typically manage regularly structured data of the order of terabytes.

For very large unstructured data-sets, the term *big data* is often used. The analysis, or *data-mining* of big data is typically carried out more efficiently by Hadoop than by relational databases, for certain types of parallelizable problems. This is because of the following characteristics of Hadoop, in comparison with relational databases:

1. **Less structured input:** Key value pairs are used as records for the data sets instead of a database.
2. **Scale-out rather than scale-up design:** For large data sets, if the size of a parallelizable problem increases linearly, the corresponding cost of scaling up a single machine to solve it tends to grow exponentially, simply because the hardware requirements tend to get exponentially expensive. If, however, the system that solves it is a cluster, then the corresponding cost tends to grow linearly because it can be solved by scaling out the cluster with a linear increase in the number of processing nodes.

Scaling out can be done, with some effort, for database problems, using a parallel relational database implementation. However scale-out is inherent in Hadoop, and therefore often easier to implement with Hadoop. The Hadoop scale-out approach is based on the following design:

- **Clustered storage:** Instead of a single node with a special, large, storage device, a distributed filesystem (HDFS) using commodity hardware devices across many nodes stores the data.
 - **Clustered processing:** Instead of using a single node with many processors, the parallel processing needs of the problem are distributed out over many nodes. The procedure is called the *MapReduce* algorithm, and is based on the following approach:
 - The distribution process “maps” the initial state of the problem into processes out to the nodes, ready to be handled in parallel.
 - Processing tasks are carried out on the data at nodes themselves.
 - The results are “reduced” back to one result.
3. **Automated failure handling at application level for data:** Replication of the data takes place across the *DataNodes*, which are the nodes holding the data. If a *DataNode* has failed, then another node which has the replicated data on it is used instead automatically. Hadoop switches over quickly in comparison to replicated database clusters due to not having to check database table consistency.

1.2 Available Hadoop Implementations

Bright Cluster Manager 7.1 integrates with a number of Hadoop distributions provided by the following organizations:

1. Apache (<http://apache.org>): This is the upstream source for the Hadoop core and some related components which all the other implementations use.
2. Cloudera (<http://www.cloudera.com>): Cloudera provides some extra premium functionality and components on top of a Hadoop suite. One of the extra components that Cloudera provides is the Cloudera Management Suite, a major proprietary management layer, with some premium features.
3. Hortonworks (<http://hortonworks.com>): Hortonworks Data Platform (HDP) is a fully open-source Hadoop suite.
4. Pivotal HD (<http://pivotal.io/big-data/pivotal-hd>): Pivotal Hadoop Distribution is a completely Apache-compliant distribution with extensive analytic toolsets. Pivotal HD, versions 2.1.0 and 3.0.1, are based on Apache Hadoop 2.2.0 and 2.6.0 respectively.

The ISO image for Bright Cluster Manager, available at <http://www.brightcomputing.com/Download>, can include Hadoop for all 4 implementations. During installation from the ISO, the administrator can choose which implementation to install (section 3.3.14 of the *Installation Manual*).

The contents and versions of the Hadoop distributions supported by Bright Computing are listed in Section 1.4.

1.3 Further Documentation

Further documentation is provided in the installed tarballs of the Hadoop version, after the Bright Cluster Manager installation (Chapter 2) has been carried out. The default location for the tarballs is under `/cm/local/apps/hadoop`. The documentation is unpacked into a relative directory path, with a starting point indicated in the table below:

Hadoop version	Relative path
Apache 1.2.1	<code>hadoop-1.2.1/docs/index.html</code>
Apache 2.7.1	<code>hadoop-2.7.1/share/doc/hadoop/index.html</code>
Cloudera CDH 5.4.8	<code>hadoop-2.6.0-cdh5.4.8/share/doc/index.html</code>
Hortonworks HDP	<i>Online documentation is available at http://docs.hortonworks.com/</i>

1.4 Version Support Matrix

The Hadoop and Hadoop-related software versions that Bright Cluster Manager supports are listed in this section for the various Hadoop implementations in sections 1.4.1-1.4.13.

Each software is provided as a package, either from a Bright repository, or from the project site, or from the implementation provider. How it is obtained, and where it is obtained from, are indicated by superscripts as follows:

Superscript	Obtained as	Location
a	package in	cm-apache-hadoop
b	package in	cm-apache-hadoop-extras
c	package in	cm-cloudera-hadoop
d	package in	cm-hortonworks-hadoop
x	pick up from	Sqoop, Spark, Apache Storm
<i>none</i>	pick up from	Hortonworks, Cloudera, Pivotal

Thus, x as a superscript means the software must be picked up from the corresponding Apache project website. The website is either:

- <http://sqoop.apache.org> for Sqoop or
- <http://spark.apache.org> for Spark or
- <https://storm.apache.org> for Apache Storm

Similarly, no superscript means that the software is available from the corresponding implementation provider website, which is one of the following options:

- <http://hortonworks.com> for Hortonworks. Direct links for Hortonworks downloads are currently (September 2015) accessible at:
<http://s3.amazonaws.com/public-repo-1.hortonworks.com/index.html>
- <http://www.cloudera.com> for Cloudera. Direct links for Cloudera downloads are currently (September 2015) accessible at:
<http://archive.cloudera.com/cdh4/cdh/4/>
or
<http://archive.cloudera.com/cdh5/cdh/5/>
- <http://pivotal.io/big-data/pivotal-hd> for Pivotal

1.4.1 Apache Hadoop 1.2.1

- `hadoop-1.2.1.tar.gz`^a
- `zookeeper-3.4.6.tar.gz`^a
- `hbase-0.98.15-hadoop1-bin.tar.gz`^a
- `apache-hive-1.2.1-bin.tar.gz`^b
- `pig-0.15.0.tar.gz`^b
- `spark-1.5.1-bin-hadoop1.tgz`^b
- `accumulo-1.5.4-bin.tar.gz`^b
- `apache-storm-0.9.5.tar.gz`^b
- `sqoop-1.4.6-bin__hadoop-1.0.0.tar.gz`^b
- `kafka_2.11-0.8.2.2.tgz`^b

1.4.2 Hortonworks HDP 1.3.11

This software is available from the Hortonworks website except where specified.

- `hadoop-1.2.0.1.3.11.0-26.tar.gzd`
- `zookeeper-3.4.5.1.3.11.0-26.tar.gzd`
- `hbase-0.94.6.1.3.11.0-26-security.tar.gzd`
- `hive-0.11.0.1.3.11.0-26.tar.gz`
- `pig-0.11.1.1.3.11.0-26.tar.gz`
- `spark-1.5.1-bin-hadoop1.tgzb`
- `accumulo-1.5.4-bin.tar.gzb`
- `apache-storm-0.9.5.tar.gzb`
- `sqoop-1.4.3.1.3.11.0-26.bin__hadoop-1.2.0.1.3.11.0-26.tar.gz`
- `kafka_2.11-0.8.2.2.tgzb`

1.4.3 Apache Hadoop 2.7.1

- `hadoop-2.7.1.tar.gza`
- `zookeeper-3.4.6.tar.gza`
- `hbase-1.1.1-bin.tar.gza`
- `apache-hive-1.2.1-bin.tar.gzb`
- `pig-0.15.0.tar.gzb`
- `spark-1.5.1-bin-hadoop2.6.tgzb`
- `accumulo-1.7.0-bin.tar.gzb`
- `apache-storm-0.9.5.tar.gzb`
- `sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gzb`
- `sqoop-1.99.5-bin-hadoop200.tar.gzx`
- `kafka_2.11-0.8.2.2.tgzb`

1.4.4 Cloudera CDH 4.6.0

This software is available from the Cloudera website except where specified.

- `hadoop-2.0.0-cdh4.6.0.tar.gz`
- `zookeeper-3.4.5-cdh4.6.0.tar.gz`
- `hbase-0.94.15-cdh4.6.0.tar.gz`
- `hive-0.10.0-cdh4.6.0.tar.gz`
- `pig-0.11.0-cdh4.6.0.tar.gz`
- `spark-1.5.1-bin-cdh4.tgzx`
- `accumulo-1.6.2-bin.tar.gzx`

- `apache-storm-0.9.5.tar.gz`^b
- `sqoop-1.4.3-cdh4.6.0.tar.gz`
- `sqoop2-1.99.2-cdh4.6.0.tar.gz`
- `kafka_2.11-0.8.2.2.tgz`^b

1.4.5 Cloudera CDH 4.7.1

This software is available from the Cloudera website except where specified.

- `hadoop-2.0.0-cdh4.7.1.tar.gz`^c
- `zookeeper-3.4.5-cdh4.7.1.tar.gz`^c
- `hbase-0.94.15-cdh4.7.1.tar.gz`^c
- `hive-0.10.0-cdh4.7.1.tar.gz`
- `pig-0.11.0-cdh4.7.1.tar.gz`
- `spark-1.5.1-bin-cdh4.tgz`^x
- `accumulo-1.6.2-bin.tar.gz`^x
- `apache-storm-0.9.5.tar.gz`^b
- `sqoop-1.4.3-cdh4.7.1.tar.gz`
- `sqoop2-1.99.2-cdh4.7.1.tar.gz`
- `kafka_2.11-0.8.2.2.tgz`^b

1.4.6 Cloudera CDH 5.2.4

This software is available from the Cloudera website except where specified.

- `hadoop-2.5.0-cdh5.2.4.tar.gz`
- `zookeeper-3.4.5-cdh5.2.4.tar.gz`
- `hbase-0.98.6-cdh5.2.4.tar.gz`
- `hive-0.13.1-cdh5.2.4.tar.gz`
- `pig-0.12.0-cdh5.2.4.tar.gz`
- `spark-1.5.1-bin-hadoop2.4.tgz`^x
- `accumulo-1.6.2-bin.tar.gz`^x
- `apache-storm-0.9.4.tar.gz`^x
- `sqoop-1.4.5-cdh5.2.4.tar.gz`
- `sqoop2-1.99.3-cdh5.2.4.tar.gz`
- `kafka_2.11-0.8.2.2.tgz`^b

1.4.7 Cloudera CDH 5.3.8

This software is available from the Cloudera website except where specified.

- `hadoop-2.5.0-cdh5.3.8.tar.gz`
- `zookeeper-3.4.5-cdh5.3.8.tar.gz`
- `hbase-0.98.6-cdh5.3.8.tar.gz`
- `hive-0.13.1-cdh5.3.8.tar.gz`
- `pig-0.12.0-cdh5.3.8.tar.gz`
- `spark-1.5.1-bin-hadoop2.4.tgzx`
- `accumulo-1.7.0-bin.tar.gzb`
- `apache-storm-0.9.5.tar.gzb`
- `sqoop-1.4.5-cdh5.3.8.tar.gz`
- `sqoop2-1.99.4-cdh5.3.8.tar.gz`
- `kafka_2.11-0.8.2.2.tgzb`

1.4.8 Cloudera CDH 5.4.8

This software is available from the Cloudera website except where specified.

- `hadoop-2.6.0-cdh5.4.8.tar.gzc`
- `zookeeper-3.4.5-cdh5.4.8.tar.gzc`
- `hbase-1.0.0-cdh5.4.8.tar.gzc`
- `hive-1.1.0-cdh5.4.8.tar.gz`
- `pig-0.12.0-cdh5.4.8.tar.gz`
- `spark-1.5.1-bin-hadoop2.6.tgzb`
- `accumulo-1.7.0-bin.tar.gzb`
- `apache-storm-0.9.5.tar.gzb`
- `sqoop-1.4.5-cdh5.4.8.tar.gz`
- `sqoop2-1.99.5-cdh5.4.8.tar.gz`
- `kafka_2.11-0.8.2.2.tgzb`

1.4.9 Hortonworks HDP 2.1.15

This software is available from the Hortonworks website except where specified.

- `hadoop-2.4.0.2.1.15.0-946.tar.gz`
- `zookeeper-3.4.5.2.1.15.0-946.tar.gz`
- `hbase-0.98.0.2.1.15.0-946-hadoop2.tar.gz`
- `apache-hive-0.13.1.2.1.15.0-946-bin.tar.gz`
- `pig-0.12.1.2.1.15.0-946.tar.gz`

- spark-1.5.1-bin-hadoop2.4.tgz^x
- accumulo-1.5.1.2.1.15.0-946-bin.tar.gz
- apache-storm-0.9.1.2.1.15.0-946.tar.gz
- sqoop-1.4.4.2.1.15.0-946.bin__hadoop-2.4.0.2.1.15.0-946.tar.gz
- kafka_2.11-0.8.2.2.tgz^b

1.4.10 Hortonworks HDP 2.2.8

This software is available from the Hortonworks website except where specified.

- hadoop-2.6.0.2.2.8.0-3150.tar.gz
- zookeeper-3.4.6.2.2.8.0-3150.tar.gz
- hbase-0.98.4.2.2.8.0-3150-hadoop2.tar.gz
- apache-hive-0.14.0.2.2.8.0-3150-bin.tar.gz
- pig-0.14.0.2.2.8.0-3150.tar.gz
- spark-1.5.1-bin-hadoop2.6.tgz^b
- accumulo-1.6.1.2.2.8.0-3150-bin.tar.gz
- apache-storm-0.9.3.2.2.8.0-3150.tar.gz
- sqoop-1.4.5.2.2.6.0-2800.bin__hadoop-2.6.0.2.2.8.0-3150.tar.gz
- sqoop-1.99.6-bin-hadoop200.tar.gz^x
- kafka_2.11-0.8.2.2.tgz^b

1.4.11 Hortonworks HDP 2.3.2

This software is available from the Hortonworks website except where specified.

- hadoop-2.7.1.2.3.2.0-2950.tar.gz^d
- zookeeper-3.4.6.2.3.2.0-2950.tar.gz^d
- hbase-1.1.1-bin.tar.gz^d
- apache-hive-1.2.1.2.3.2.0-2950-bin.tar.gz
- pig-0.15.0.2.3.2.0-2950.tar.gz
- spark-1.5.1-bin-hadoop2.6.tgz^b
- accumulo-1.7.0.2.3.2.0-2950-bin.tar.gz
- apache-storm-0.10.0.2.3.2.0-2950.tar.gz
- sqoop-1.4.6.2.3.0.0-2557.bin__hadoop-2.7.1.2.3.2.0-2950.tar.gz
- sqoop-1.99.6-bin-hadoop200.tar.gz^x
- kafka_2.11-0.8.2.2.tgz^b

1.4.12 Pivotal HD 2.1.0

The software is available from the Pivotal website except where specified.

- PHD-2.1.0.0-175.tar.gz
- apache-hive-1.2.1-bin.tar.gz^b
- pig-0.15.0.tar.gz^b
- spark-1.2.1-bin-hadoop2.4.tgz^x
- accumulo-1.7.0-bin.tar.gz^b
- apache-storm-0.9.5.tar.gz^b
- sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz^b
- sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz^b
- kafka_2.11-0.8.2.2.tgz^b

1.4.13 Pivotal HD 3.0.1

The software is available from the Pivotal website except where specified.

- PHD-3.0.1.0-1-centos6.tar.gz
or
PHD-3.0.1.0-1-suse11sp3.tar.gz
- apache-hive-1.2.1-bin.tar.gz^b
- pig-0.15.0.tar.gz^b
- spark-1.5.1-bin-hadoop2.6.tgz^b
- accumulo-1.7.0-bin.tar.gz^b
- apache-storm-0.9.5.tar.gz^b
- sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz^b

2

Installing Hadoop

In Bright Cluster Manager, a Hadoop instance can be configured and run either via the command-line (section 2.1) or via an ncurses GUI (section 2.2). Both options can be carried out with the `cm-hadoop-setup` script, which is run from a head node. The script is a part of the `cluster-tools` package, and uses tarballs from the Apache Hadoop project. The Bright Cluster Manager With Hadoop installation ISO includes the `cm-apache-hadoop` package, which contains tarballs from the Apache Hadoop project suitable for `cm-hadoop-setup`.

2.1 Command-line Installation Of Hadoop Using `cm-hadoop-setup -c <filename>`

2.1.1 Usage

```
[root@bright71 ~]# cm-hadoop-setup -h
```

```
USAGE: /cm/local/apps/cluster-tools/bin/cm-hadoop-setup [-c <filename>
| -u <name> | -h]
```

OPTIONS:

```
-c <filename>  -- Hadoop config file to use
-u <name>      -- uninstall Hadoop instance
-h            -- show usage
```

EXAMPLES:

```
cm-hadoop-setup -c /tmp/config.xml
cm-hadoop-setup -u foo
cm-hadoop-setup    (no options, a gui will be started)
```

Some sample configuration files are provided in the directory
`/cm/local/apps/cluster-tools/hadoop/conf/`:

<code>hadoop1conf.xml</code>	(for Hadoop 1.x)
<code>hadoop2conf.xml</code>	(for Hadoop 2.x)
<code>hadoop2fedconf.xml</code>	(for Hadoop 2.x with NameNode federation)
<code>hadoop2haconf.xml</code>	(for Hadoop 2.x with High Availability)
<code>hadoop2lustreconf.xml</code>	(for Hadoop 2.x with Lustre support)

2.1.2 An Install Run

An XML template can be used based on the examples in the directory `/cm/local/apps/cluster-tools/hadoop/conf/`.

In the XML template, the path for a tarball component is enclosed by `<archive>` `</archive>` tag pairs. The tarball components can be as indicated:

- `<archive>hadoop tarball</archive>`
- `<archive>hbase tarball</archive>`
- `<archive>zookeeper tarball</archive>`

The tarball components can be picked up from URLs as listed in section 1.2. The paths of the tarball component files that are to be used should be set up as needed before running `cm-hadoop-setup`.

The downloaded tarball components should be placed in the `/tmp/` directory if the default definitions in the default XML files are used:

Example

```
[root@bright71 ~]# cd /cm/local/apps/cluster-tools/hadoop/conf
[root@bright71 conf]# grep 'archive>' hadoop1conf.xml | grep -o /.*.gz
/tmp/hadoop-1.2.1.tar.gz
/tmp/zookeeper-3.4.6.tar.gz
/tmp/hbase-0.98.12.1-hadoop1-bin.tar.gz
```

Files under `/tmp` are not intended to stay around permanently. The administrator may therefore wish to place the tarball components in a more permanent part of the filesystem instead, and change the XML definitions accordingly.

A Hadoop instance name, for example `Myhadoop`, can also be defined in the XML file, within the `<name></name>` tag pair.

Hadoop NameNodes and SecondaryNameNodes handle HDFS metadata, while DataNodes manage HDFS data. The data must be stored in the filesystem of the nodes. The default path for where the data is stored can be specified within the `<dataroot></dataroot>` tag pair. Multiple paths can also be set, using comma-separated paths. NameNodes, SecondaryNameNodes, and DataNodes each use the value, or values, set within the `<dataroot></dataroot>` tag pair for their root directories.

If needed, more specific tags can be used for each node type. This is useful in the case where hardware differs for the various node types. For example:

- a NameNode with 2 disk drives for Hadoop use
- a DataNode with 4 disk drives for Hadoop use

The XML file used by `cm-hadoop-setup` can in this case use the tag pairs:

- `<namenodedatadirs></namenodedatadirs>`
- `<datanodedatadirs></datanodedatadirs>`

If these are not specified, then the value within the `<dataroot></dataroot>` tag pair is used.

Example

- `<namenodedatadirs>/data1,/data2</namenodedatadirs>`
- `<datanodedatadirs>/data1,/data2,/data3,/data4</datanodedatadirs>`

Hadoop should then have the following `dfs.*.name.dir` properties added to it via the `hdfs-site.xml` configuration file. For the preceding tag pairs, the property values should be set as follows:

Example

- `dfs.namenode.name.dir` with values:
`/data1/hadoop/hdfs/namenode, /data2/hadoop/hdfs/namenode`

- `dfs.datanode.name.dir` with values:
`/data1/hadoop/hdfs/datanode, /data2/hadoop/hdfs/datanode, /data3/hadoop/hdfs/datanode, /data4/hadoop/hdfs/datanode`

An install run then displays output like the following:

Example

```
-rw-r--r-- 1 root root 63851630 Feb  4 15:13 hadoop-1.2.1.tar.gz
[root@bright71 ~]# cm-hadoop-setup -c /tmp/hadoop1conf.xml
Reading config from file '/tmp/hadoop1conf.xml'... done.
Hadoop flavor 'Apache', release '1.2.1'
Will now install Hadoop in /cm/shared/apps/hadoop/Apache/1.2.1 and conf\
figure instance 'Myhadoop'
Hadoop distro being installed... done.
Zookeeper being installed... done.
HBase being installed... done.
Creating module file... done.
Configuring Hadoop instance on local filesystem and images... done.
Updating images:
starting imageupdate for node 'node003'... started.
starting imageupdate for node 'node002'... started.
starting imageupdate for node 'node001'... started.
starting imageupdate for node 'node004'... started.
Waiting for imageupdate to finish... done.
Creating Hadoop instance in cmdaemon... done.
Formatting HDFS... done.
Waiting for datanodes to come up... done.
Setting up HDFS... done.
```

The Hadoop instance should now be running. The name defined for it in the XML file should show up within `cmgui`, in the Hadoop HDFS resource tree folder (figure 2.1).

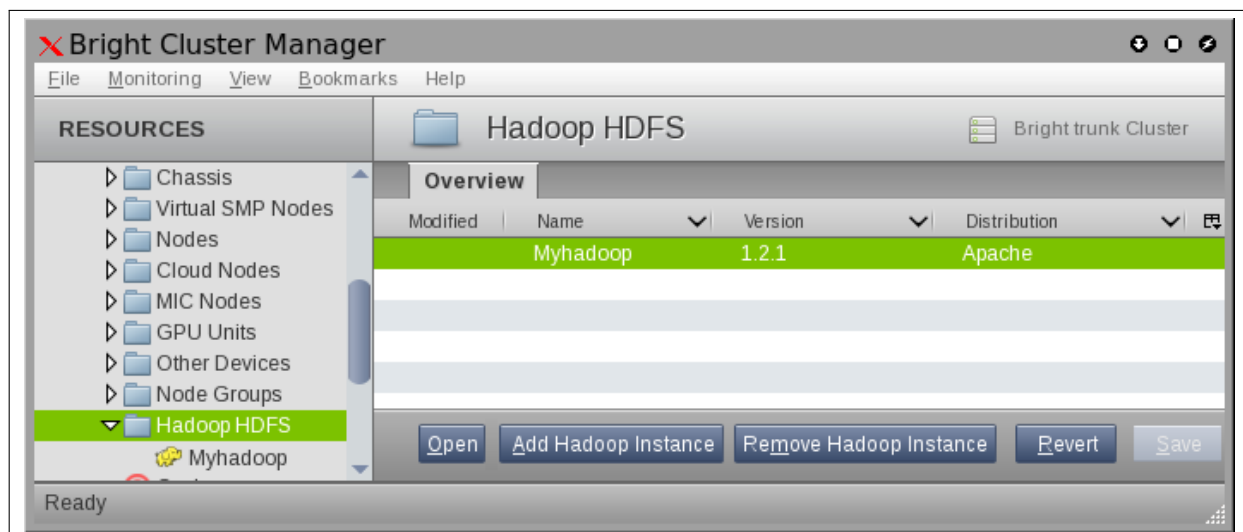


Figure 2.1: A Hadoop Instance In `cmgui`

The instance name is also displayed within `cmsh` when the `list` command is run in `hadoop` mode:

Example

```
[root@bright71 ~] cmsg
[bright71]% hadoop
[bright71->hadoop]% list
Name (key) Hadoop version Hadoop distribution Configuration directory
-----
Myhadoop 1.2.1 Apache /etc/hadoop/Myhadoop
```

The instance can be removed as follows:

Example

```
[root@bright71 ~]# cm-hadoop-setup -u Myhadoop
Requested uninstall of Hadoop instance 'Myhadoop'.
Uninstalling Hadoop instance 'Myhadoop'
```

Removing:

```
/etc/hadoop/Myhadoop
/var/lib/hadoop/Myhadoop
/var/log/hadoop/Myhadoop
/var/run/hadoop/Myhadoop
/tmp/hadoop/Myhadoop/
/etc/hadoop/Myhadoop/zookeeper
/var/lib/zookeeper/Myhadoop
/var/log/zookeeper/Myhadoop
/var/run/zookeeper/Myhadoop
/etc/hadoop/Myhadoop/hbase
/var/log/hbase/Myhadoop
/var/run/hbase/Myhadoop
/etc/init.d/hadoop-Myhadoop-*
```

Module file(s) deleted.

Uninstall successfully completed.

2.2 Ncurses Installation Of Hadoop Using `cm-hadoop-setup`

Running `cm-hadoop-setup` without any options starts up an ncurses GUI (figure 2.2).

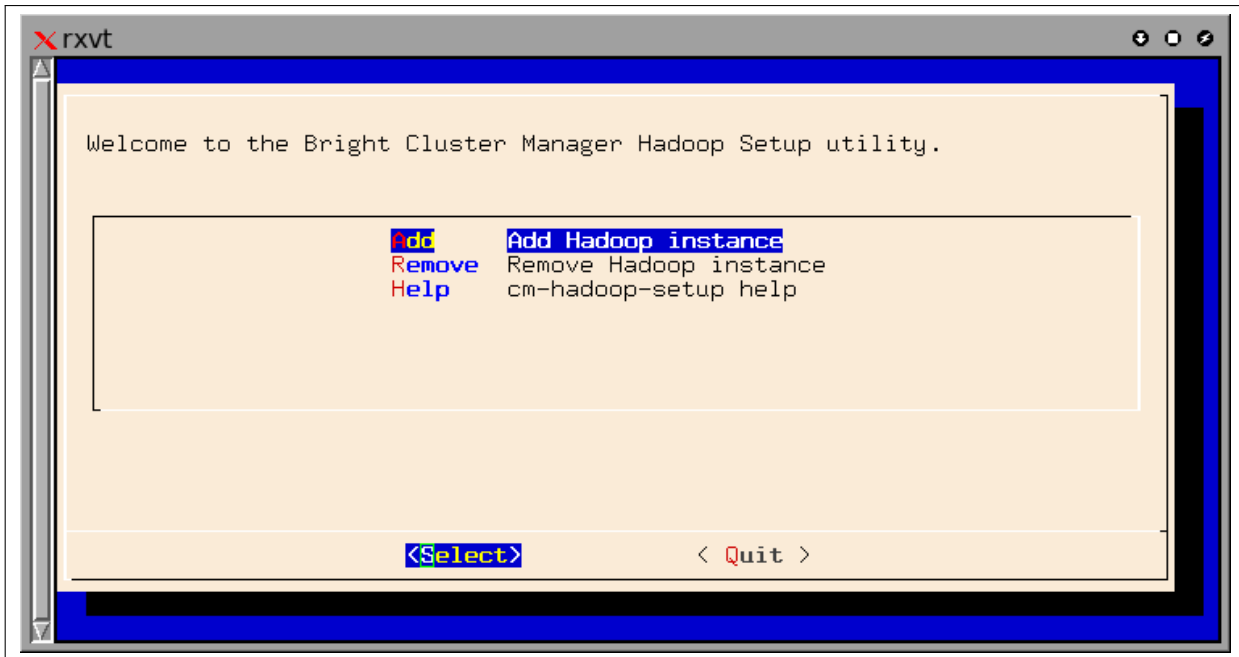


Figure 2.2: The cm-hadoop-setup Welcome Screen

This provides an interactive way to add and remove Hadoop instances, along with HBase and Zookeeper components. Some explanations of the items being configured are given along the way. In addition, some minor validation checks are done, and some options are restricted.

The suggested default values will work. Other values can be chosen instead of the defaults, but some care in selection usually a good idea. This is because Hadoop is a complex software, which means that values other than the defaults can sometimes lead to unworkable configurations (section 2.3).

The ncurses installation results in an XML configuration file. This file can be used with the `-c` option of `cm-hadoop-setup` to carry out the installation.

Installation Of Additional Tools

Sections 2.1 and 2.2 cover the the installation of Hadoop with a minimal configuration. Support for ZooKeeper, HBase, and additional tools such as Hive and Spark depends upon the Hadoop distribution and version. The version support matrix (section 1.4), and the appropriate sections in chapter 6 describe installation of the additional components.

2.3 Avoiding Misconfigurations During Hadoop Installation

A misconfiguration can be defined as a configuration that works badly or not at all.

For Hadoop to work well, the following common misconfigurations should normally be avoided. Some of these result in warnings from Bright Cluster Manager validation checks during configuration, but can be overridden. An override is useful for cases where the administrator would just like to, for example, test some issues not related to scale or performance.

2.3.1 NameNode Configuration Choices

One of the following NameNode configuration options must be chosen when Hadoop is installed. The choice should be made with care, because changing between the options after installation is not possible.

Hadoop 1.x NameNode Configuration Choices

- NameNode can optionally have a SecondaryNameNode.

SecondaryNameNode offloads metadata operations from NameNode, and also stores the meta-

data offline to some extent.

It is not by any means a high availability solution. While recovery from a failed head node is possible from SecondaryNameNode, it is not easy, and it is not recommended or supported by Bright Cluster Manager.

Hadoop 2.x NameNode Configuration Choices

- NameNode and SecondaryNameNode can run as in Hadoop 1.x.

However, the following configurations are also possible:

- **NameNode HA with manual failover:** In this configuration Hadoop has NameNode1 and NameNode2 up at the same time, with one active and one on standby. Which NameNode is active and which is on standby is set manually by the administrator. If one NameNode fails, then failover must be executed manually. Metadata changes are managed by ZooKeeper, which relies on a quorum of JournalNodes. The number of JournalNodes is therefore set to 3, 5, 7...
- **NameNode HA with automatic failover:** As for the manual case, except that in this case ZooKeeper manages failover too. Which NameNode is active and which is on standby is therefore decided automatically.
- **NameNode Federation:** In NameNode Federation, the storage of metadata is split among several NameNodes, each of which has a corresponding SecondaryNameNode. Each pair takes care of a part of HDFS.

In Bright Cluster Manager there are 4 NameNodes in a default NameNode federation:

- /user
- /tmp
- /staging
- /hbase

User applications do not have to know this mapping. This is because ViewFS on the client side maps the selected path to the corresponding NameNode. Thus, for example, `hdfs -ls /tmp/example` does not need to know that /tmp is managed by another NameNode.

Cloudera advise against using NameNode Federation for production purposes at present, due to its development status.

2.4 Installing Hadoop With Lustre

The Lustre filesystem has a client-server configuration. Its installation on Bright Cluster Manager is covered in section 7.7 of the *Installation Manual*.

2.4.1 Lustre Internal Server Installation

The procedure for installing a Lustre server varies. It is covered in section 7.7.3 of the *Installation Manual*.

2.4.2 Lustre External Server Installation

Lustre can also be configured so that the servers run external to Bright Cluster Manager. The Lustre Intel IEEL 2.x version can be configured in this manner.

2.4.3 Lustre Client Installation

It is preferred that the Lustre clients are installed on the head node as well as on all the nodes that are to be Hadoop nodes. The clients should be configured to provide a Lustre mount on the nodes. If the Lustre client cannot be installed on the head node, then Bright Cluster Manager has the following limitations during installation and maintenance:

- the head node cannot be used to run Hadoop services
- end users cannot perform Hadoop operations, such as job submission, on the head node. Operations such as those should instead be carried out while logged in to one of the Hadoop nodes

In the remainder of this section, a Lustre mount point of `/mnt/lustre` is assumed, but it can be set to any convenient directory mount point.

The user IDs and group IDs of the Lustre server and clients should be consistent. It is quite likely that they differ when first set up. The IDs should be checked at least for the following users and groups:

- **users:** `hdfs`, `mapred`, `yarn`, `hbase`, `zookeeper`, `hive`
- **groups:** `hadoop`, `zookeeper`, `hbase`, `hive`

If they do not match on the server and clients, then they must be made consistent manually, so that the UID and GID of the Lustre server users are changed to match the UID and GID of the Bright Cluster Manager users.

Once consistency has been checked, and read/write access is working to LustreFS, the Hadoop integration can be configured.

2.4.4 Lustre Hadoop Configuration

Lustre Hadoop XML Configuration File Setup

Hadoop integration can be configured by using the file `/cm/local/apps/cluster-tools/hadoop/conf/hadoop2lustreconf.xml` as a starting point for the configuration. It can be copied over to, for example, `/root/hadooplustreconfig.xml`.

The Intel Distribution for Hadoop (IDH) and Cloudera can both run with Lustre under Bright Cluster Manager. The configuration for these can be done as follows:

- IDH
 - A subdirectory of `/mnt/lustre` must be specified in the `hadoop2lustreconf.xml` file within the `<afs></afs>` tag pair:

Example

```
<afs>
  <fstype>lustre</fstype>
  <fsroot>/mnt/lustre/hadoop</fsroot>
</afs>
```

- Cloudera
 - A subdirectory of `/mnt/lustre` must be specified in the `hadoop2lustreconf.xml` file within the `<afs></afs>` tag pair:
 - In addition, an `<fsjar></fsjar>` tag pair must be specified manually for the jar that the Intel IEEL 2.x distribution provides:

Example

```
<afs>
  <fstype>lustre</fstype>
  <fsroot>/mnt/lustre/hadoop</fsroot>
  <fsjar>/root/lustre/hadoop-lustre-plugin-2.3.0.jar</fsjar>
</afs>
```

The installation of the Lustre plugin is automatic if this jar name is set to the right name, when the `cm-hadoop-setup` script is run.

Lustre Hadoop Installation With `cm-hadoop-setup`

The XML configuration file specifies how Lustre should be integrated in Hadoop. If the configuration file is at `</root/hadooplustreconfig.xml>`, then it can be run as:

Example

```
cm-hadoop-setup -c </root/hadooplustreconfig.xml>
```

As part of configuring Hadoop to use Lustre, the execution will:

- Set the ACLs on the directory specified within the `<fsroot><fsroot>` tag pair. This was set to `/mnt/lustre/hadoop` earlier on as an example.
- Copy the Lustre plugin from its jar path as specified in the XML file, to the correct place on the client nodes.

Specifically, the subdirectory `./share/hadoop/common/lib` is copied into a directory relative to the Hadoop installation directory. For example, the Cloudera version of Hadoop, version 2.30-cdh5.1.2, has the Hadoop installation directory `/cm/share/apps/hadoop/Cloudera/2.3.0-cdh5.1.2`. The copy is therefore carried out in this case from:

```
/root/lustre/hadoop-lustre-plugin-2.3.0.jar
```

to

```
/cm/shared/apps/hadoop/Cloudera/2.3.0-cdh5.1.2/share/hadoop/common/lib
```

Lustre Hadoop Integration In `cmsh` **and** `cmgui`

In `cmsh`, Lustre integration is indicated in `hadoop` mode:

Example

```
[hadoop2->hadoop]% show hdfs1 | grep -i lustre
Hadoop root for Lustre           /mnt/lustre/hadoop
Use Lustre                       yes
```

In `cmgui`, the Overview tab in the items for the Hadoop HDFS resource indicates if Lustre is running, along with its mount point (figure 2.3).

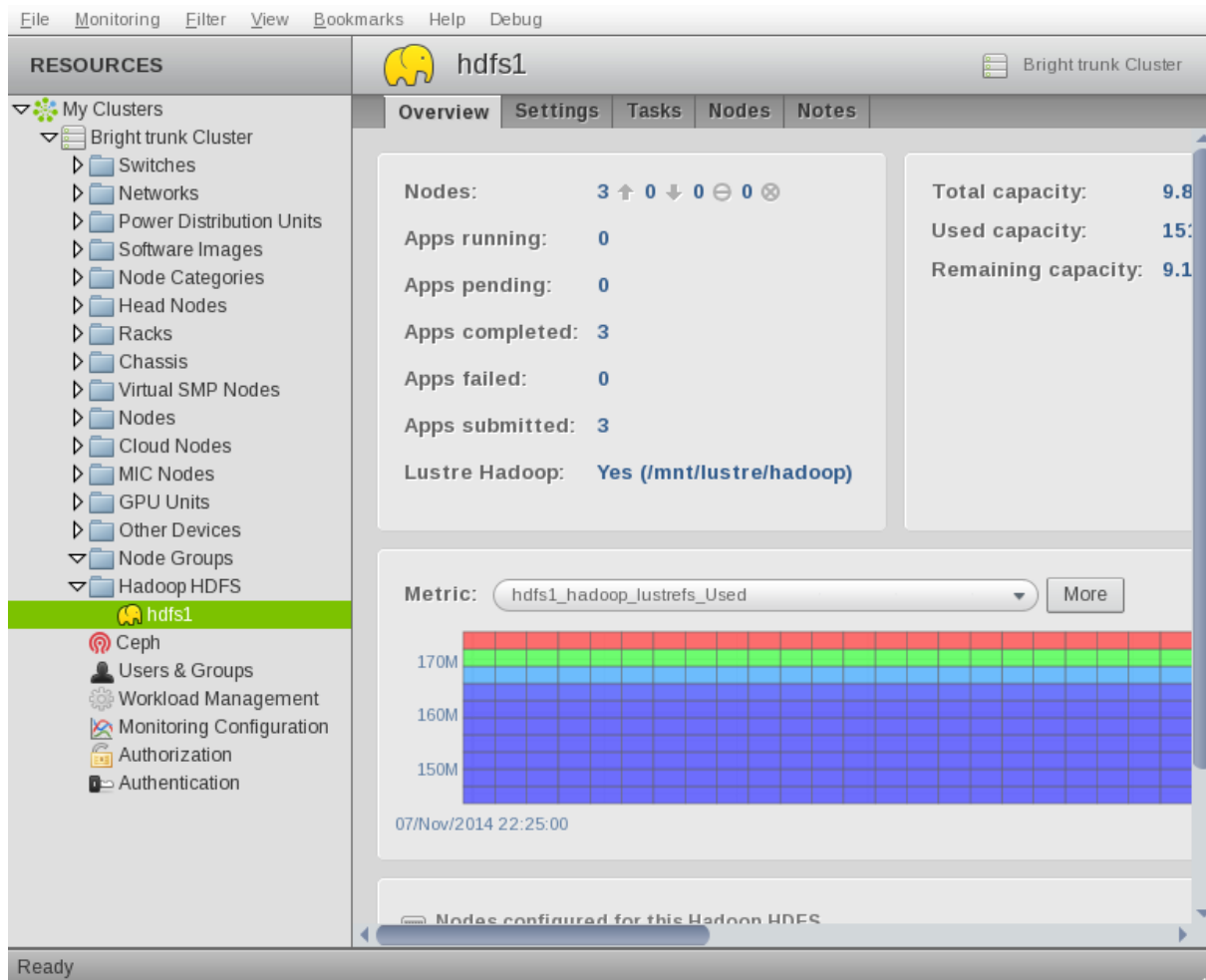


Figure 2.3: A Hadoop Instance With Lustre In cmgui

2.5 Hadoop Installation In A Cloud

Hadoop can make use of cloud services so that it runs as a Cluster-On-Demand configuration (Chapter 2 of the *Cloudbursting Manual*), or a Cluster Extension configuration (Chapter 3 of the *Cloudbursting Manual*). In both cases the cloud nodes used should be at least `m1.medium`.

- For Cluster-On-Demand the following considerations apply:
 - There are no specific issues. After a stop/start cycle Hadoop recognizes the new IP addresses, and refreshes the list of nodes accordingly (section 2.4.1 of the *Cloudbursting Manual*).
- For Cluster Extension the following considerations apply:
 - To install Hadoop on cloud nodes, the XML configuration: `/cm/local/apps/cluster-tools/hadoop/conf/hadoop2clusterextensionconf.xml` can be used as a guide.
 - In the `hadoop2clusterextensionconf.xml` file, the cloud director that is to be used with the Hadoop cloud nodes must be specified by the administrator with the `<edge></edge>` tag pair:

Example


```
<edge>
  <hosts>eu-west-1-director</hosts>
</edge>
```

Maintenance operations, such as a format, will automatically and transparently be carried out by `cmdaemon` running on the cloud director, and not on the head node.

There are some shortcomings as a result of relying upon the cloud director:

- Cloud nodes depend on the same cloud director
- While Hadoop installation (`cm-hadoop-setup`) is run on the head node, users must run Hadoop commands—job submissions, and so on—from the director, not from the head node.
- It is not possible to mix cloud and non-cloud nodes for the same Hadoop instance. That is, a local Hadoop instance cannot be extended by adding cloud nodes.

3

Hadoop Cluster Management

The management of a Hadoop cluster using `cmgui`, `cmsh`, and the command line, is described in this chapter.

3.1 Managing A Hadoop Instance With `cmgui`

In `cmgui`, the `Hadoop instances` folder in the resource tree opens up to display the Hadoop instances running on the cluster (figure 2.1). Clicking on a Hadoop instance makes the tabs associated with Hadoop data management accessible (figure 3.1).

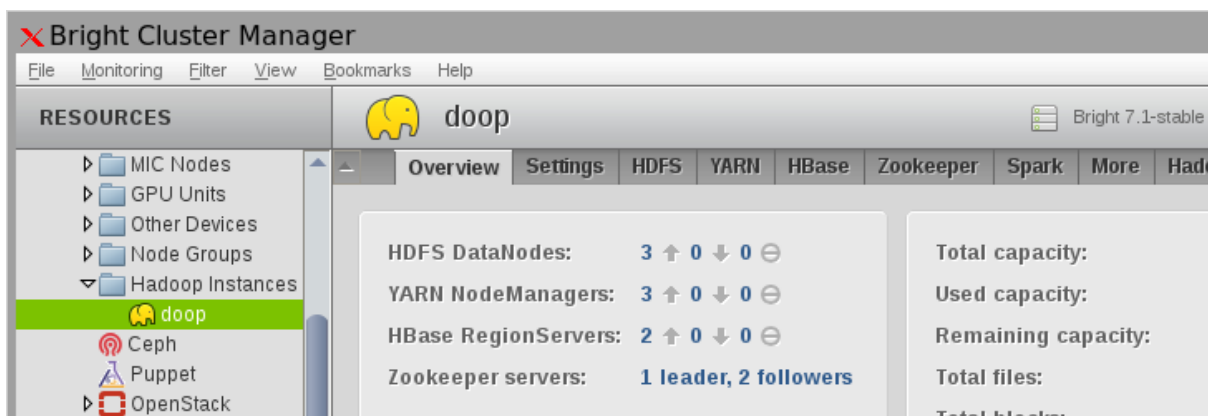


Figure 3.1: Tabs For A Hadoop Instance In `cmgui`

The following Hadoop tabs are described within this section:

1. Overview Tab (section 3.1.1)
2. Settings Tab (section 3.1.2)
3. HDFS Tab (section 3.1.3)
4. MapReduce or YARN Tab (section 3.1.4)
5. HBase Tab (section 3.1.5)
6. Zookeeper Tab (section 3.1.6)
7. Spark Tab (section 3.1.7)
8. More Tab (section 3.1.8)
9. Hadoop Configuration Groups Tab (section 3.1.9)

10. Monitoring Tab (section 3.1.10)

11. Notes Tab (section 3.1.11)

Not all of these tabs are necessarily displayed, depending on the software installed.

For example, if a user chooses to not install the HBase and Zookeeper components during the Hadoop installation procedure then the HBase and Zookeeper tabs are not displayed for this instance.

3.1.1 The HDFS Instance Overview Tab

The Overview tab pane (figure 3.2) aggregates the information about all Hadoop components and conveniently displays it in blocks.

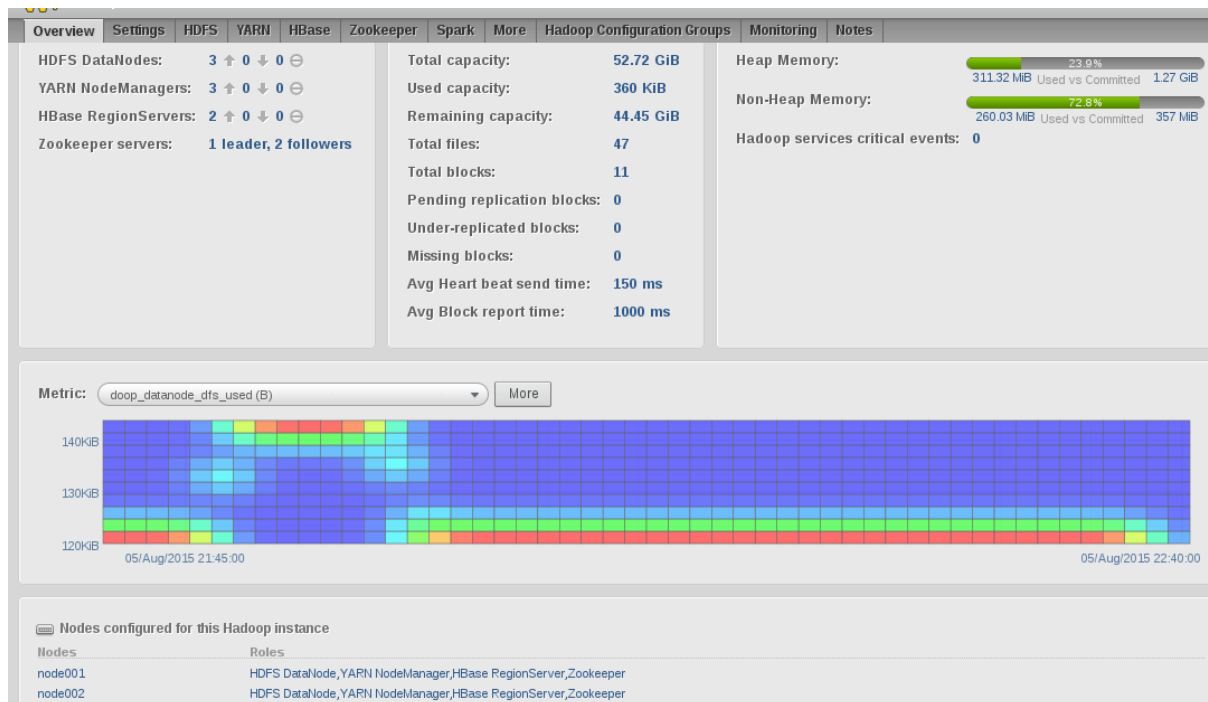


Figure 3.2: Overview Tab View For A Hadoop Instance In cmgui

The following items can be viewed:

- **Statistics:** In the top block there are statistics associated with the Hadoop instance. These include numbers of live/dead/decommissioned Hadoop services running on a cluster, as well as memory and capacity usage.
- **Metrics:** In the middle block, a metric can be selected for display as a heat map.
 - The `More` button allows other Hadoop-related metrics to be monitored in a somewhat similar way to the monitoring visualization system (section 9.3 of the *Administrator Manual*). The extra Hadoop-related metrics that can then be viewed are organized in subtabs, and further views of selected nodes can be added with the `Node details` button.
- **Roles:** The third block displays the Hadoop/Spark roles associated with each node used by this Hadoop instance.

3.1.2 The HDFS Instance Settings Tab

The Settings tab pane (Figure 3.3) presents the general details about Hadoop instance installation and configuration and allows a user to configure a number of HDFS-related parameters.

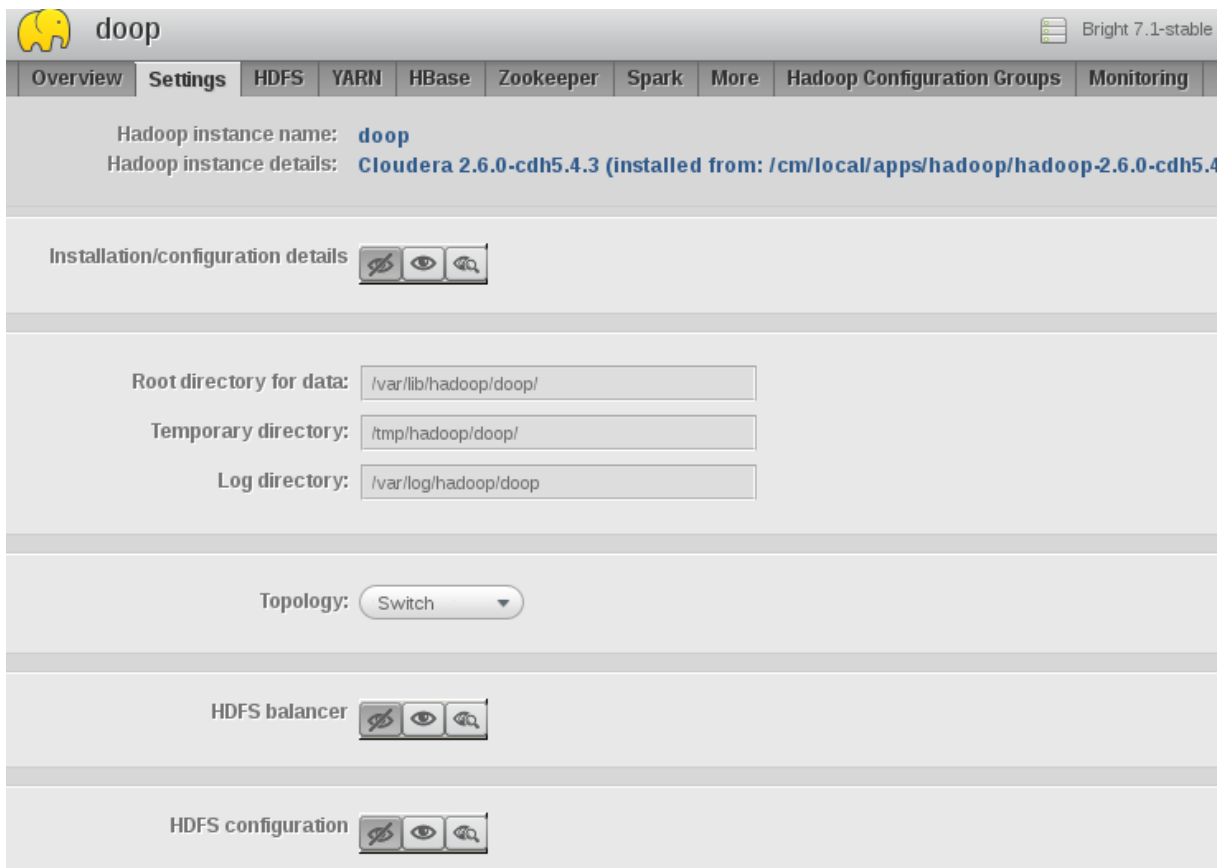


Figure 3.3: Settings Tab View For A Hadoop Instance In cmgui

Major details about Hadoop installation, such as the locations of Hadoop components or temporary files placement, can be viewed, but cannot be changed.

The remaining parameters (figure 3.4) can be viewed and changed. These are:

Topology: Switch

None
Switch
Rack

HDFS balancer

Balancer period: 48 hours Balancer threshold: 10 Balancer policy: datanode

HDFS configuration

HDFS default block size: 134217728 bytes I/O buffer size: 65536 bytes ☒ HDFS permissions enabled

HDFS default replication factor: 3 First block report delay: 10 s ☐ HTTPS for web UIs enabled

HDFS maximum replication factor: 50 HDFS Umask: 022 ☐ WebHDFS enabled

Compression codec: +

Serialization classes: org.apache.hadoop.io.serializer -
org.apache.hadoop.io.serializer -
org.apache.hadoop.io.serializer - +

Figure 3.4: Settings Tab View Details For A Hadoop Instance In cmgui

- **Topology:** Hadoop can be made aware of a cluster topology so that HDFS data replication is done more efficiently. Topology options are:
 - none: No topology-based optimization is set.
 - Switch: HDFS datanodes become switch-aware, which allows HDFS to minimize data access between switches.
 - Rack: HDFS datanodes become rack-aware to minimize data access between racks.
- **HDFS balancer:** Configuration values used for HDFS balancing (i.e., moving data blocks from over-utilized to under-utilized nodes). The parameters are:
 - Balancer period: Sets the period in hours between balancing operations.
 - Balancer threshold: Defines the maximum difference (in %) between the percentage of disk usage on any given DataNode and the average percentage of disk usage across all DataNodes.
 - Balancer policy: Sets a balancing policy.
 - * blockpool: Balancing is done at the block pool level.
 - * datanode: (default) Balances the storage at the DataNode level.
- **HDFS configuration:** Global settings for HDFS filesystem including the following parameters:
 - HDFS default block size
 - HDFS default replication factor
 - HDFS maximum replication factor
 - I/O buffer size
 - First block report delay
 - HDFS Umask
 - HDFS permissions enabled
 - HTTPS for web UIs enabled
 - WebHDFS enabled
 - ...

3.1.3 The HDFS Instance HDFS Tab

The HDFS tab as well as tabs displayed in sections 3.1.4-3.1.7 all follow a similar layout pattern. The pattern is: a block at the top overviews the resources of a corresponding Hadoop component and subtabs below, *Operations* and *Configuration*, allow a user to manage the resources of this component.

Specifically, the HDFS tab pane (figure 3.5), which focuses on the HDFS component, displays HDFS NameNode and DataNode activities at the top, and available HDFS-specific operations and configuration in the associated subtabs beneath. This is now elaborated upon next.



Figure 3.5: HDFS Tab For A Hadoop Instance In cmgui

For the HDFS tabbed pane, the top of the pane displays NameNode and DataNode JVM use, DataNodes' status information, DFS disk usage and some file/block-related total counts.

Below the main pane are the following two subtabs:

- An *Operations* subtab. This allows the following operations to be carried out with buttons:
 - HDFS: HDFS start, stop, and restart
 - (De-)commission: add and remove DataNodes from the overall DataNodes pool
 - HDFS Balancer: start or stop the HDFS balancer
 - Safemode: enter or leave safemode (i.e., a read-only mode for the HDFS).
 - Format: Format the HDFS filesystem
- A *Configuration* subtab. This provides a list of Hadoop configuration groups (section 3.1.9) that use the HDFS service and the roles associated with these configuration groups.

Hadoop configuration groups are discussed in the dedicated section 3.1.9.

Double-clicking on a configuration group, or clicking on the open button for a selected configuration group, opens up a configuration group editor window for that configuration group.

3.1.4 The HDFS Instance MapReduce Or YARN Tab

The next tab in the row of Hadoop tabs in figure 3.1, after the HDFS tab, is:

- either the MapReduce tab (figure 3.6), as used in older Hadoop distributions such as Apache Hadoop 1.2.1
- or the YARN tab (figure 3.7) for more recent distributions



Figure 3.6: MapReduce Tab For A Hadoop Instance In cmgui

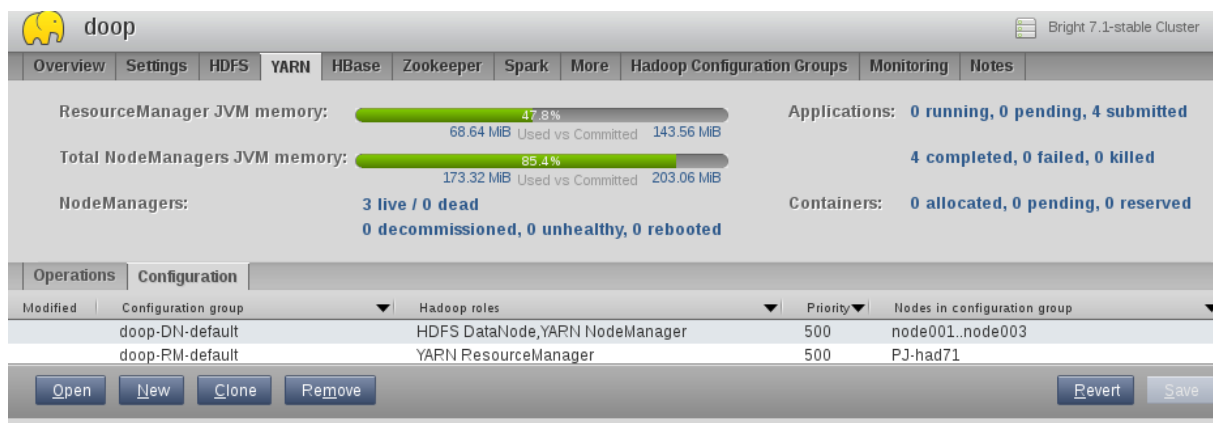


Figure 3.7: YARN Tab For A Hadoop Instance In cmgui

The MapReduce or YARN tab pane follow the pattern of sections 3.1.3-3.1.7. That is: the top block of the pane tracks job/application execution resources, while the two subtabs below the top block, Operations and Configuration, allow a user to perform operations on MapReduce (or YARN) and to configure MapReduce (or YARN) components via corresponding configuration groups.

The following operations can be performed in the Operations subtab:

- MapReduce or YARN: MapReduce (or YARN) start, stop, and restart
- (De-) commission: add and remove TaskTrackers (or NodeManagers) from the overall TaskTrackers (NodeManagers) pool

3.1.5 The HDFS Instance HBase Tab

In the HBase tab pane (figure 3.8), the patten of sections 3.1.3-3.1.7 is followed. Thus, the top block tracks HBase resources, while the subtabs below it allow a user to perform HBase operations, or allow configuration of nodes via the HBase-related configuration groups.

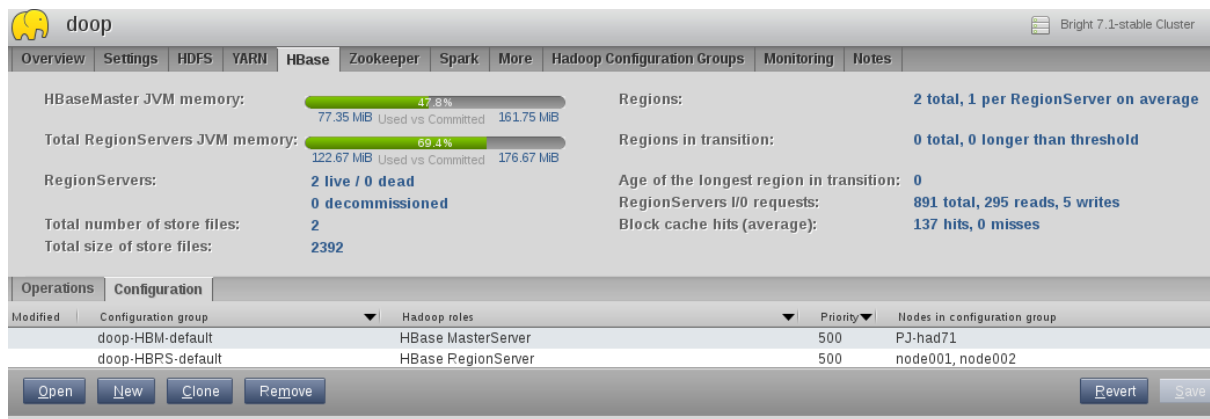


Figure 3.8: HBase Tab For A Hadoop Instance In cmgui

3.1.6 The HDFS Instance Zookeeper Tab

In the Zookeeper tab pane (figure 3.9), following the pattern of sections 3.1.3-3.1.7, the top block tracks Zookeeper resources, while the subtabs below it allow a user to perform Zookeeper operations, or allow configuration of nodes via the Zookeeper-related configuration groups.

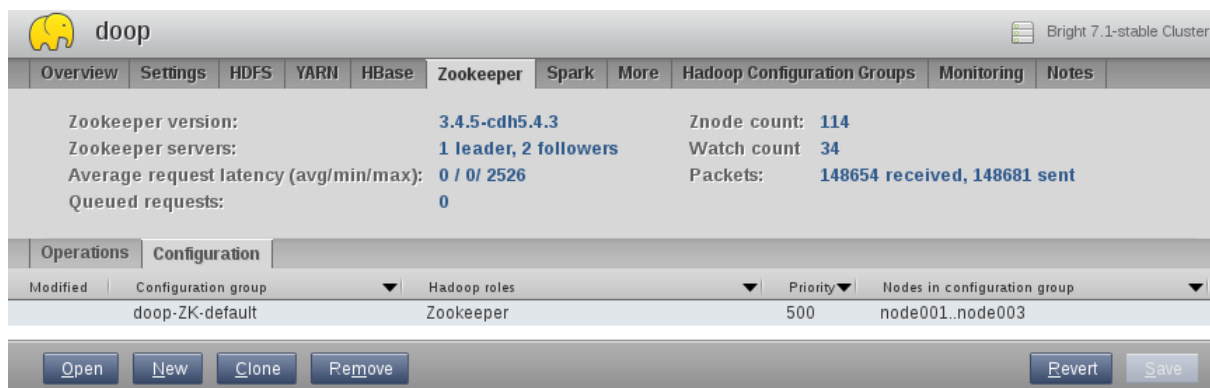


Figure 3.9: Zookeeper Tab For A Hadoop Instance In cmgui

3.1.7 The HDFS Instance Spark Tab

The Spark tab pane appears if Spark (Chapter 5) has been installed. The tab follows the common pattern of Sections 3.1.3-3.1.7.

3.1.8 The HDFS Instance More Tab

The More tab pane is reserved for future use for the Hive and Sqoop projects.

3.1.9 The HDFS Instance Hadoop Configuration Groups Tab

While the main Hadoop Configuration Groups tab shows all of the Hadoop configuration groups for the Hadoop instance, the Configuration sub-tabs described earlier in sections 3.1.3-3.1.7 show only those Hadoop configuration groups that have at least one role related to a corresponding component. Modifications done in main tab, or modifications done in one of the sub-tabs in sections 3.1.3-3.1.7, are automatically synchronized with each other.

Configuration Overlays And Hadoop Configuration Groups

In Bright Cluster Manager, a Hadoop configuration group is a special Hadoop case of the general Bright Cluster Manager concept of a configuration overlay.

- A *configuration overlay* assigns roles (section 2.1.5 of the *Administrator Manual*) for groups of nodes. The number of roles can be quite large, and priorities can be set for these.

Multiple configuration overlays can be set for a node. A priority can be set for each configuration overlay, so that a configuration overlay with a higher priority is applied to its associated node instead of a configuration overlay with a lower priority. The configuration overlay with the highest priority then determines the actual assigned role.

- A *Hadoop configuration group* is a configuration overlay that assigns a group of roles to a Hadoop instance. Thus, when the Hadoop configuration group overlays the Hadoop instance, then roles are assigned to nodes according to the configuration, along with a priority. Whether the Hadoop configuration group assignment is used, or whether the original role assignment is used, depends upon the configured priorities.

Configuration overlays can take on priorities in the range 0-1000, except for 250 and 750, which are forbidden. Setting a priority of -1 means that the configuration group is ignored.

The priorities of 250, 500, and 750 are also special, as indicated by the following table:

priority	assigned to node from
-1	configuration group not assigned
250	category
500	configuration overlay with default priority
750	node

Roles assigned at category level have a fixed priority of 250, while roles assigned at node level have a fixed priority of 750. The configuration overlay priority is variable, but is set to 500 by default. Thus, for example, roles assigned at the node level override roles assigned at the category level. Roles assigned at the node level also override roles assigned by the default configuration overlay.

Display And Management Of Hadoop Configuration Groups Within Hadoop Tab Of cmgui

The Hadoop Configuration Groups tab pane (figure 3.10) displays a list of all the configuration groups used by the Hadoop instance.

Modified	Configuration group	Hadoop roles	Priority	Nodes in configuration group
	doop-DN-default	HDFS DataNode,YARN NodeMa...	500	node001..node003
	doop-HBM-default	HBase MasterServer	500	PJ-had71
	doop-HBRS-default	HBase RegionServer	500	node001, node002
	doop-NN-default	HDFS NameNode	500	PJ-had71
	doop-RM-default	YARN ResourceManager	500	PJ-had71
	doop-SNN-default	HDFS SecondaryNameNode	500	PJ-had71
	doop-ZK-default	Zookeeper	500	node001..node003

Figure 3.10: Hadoop Configuration Groups Tab For A Hadoop Instance In cmgui

The names of the configuration groups take the following form by default:

<hadoop instance name>-<role abbreviation>-default

Example

doop-DN-default

Hadoop/Spark Roles: The role abbreviations used are indicated by the following table of roles available under Hadoop:

Table 3.1.9: Hadoop/Spark Roles And Abbreviations

role	abbreviation	cmsh role
DataNode	DN	Hadoop::DataNode
HBase MasterServer*	HBM	Hadoop::HBaseServer
HBase RegionServer	HBRS	Hadoop::HBaseClient
Hive	HV	Hadoop::Hive
JournalNode	JN	Hadoop::Journal
JobTracker ¹	JT	Hadoop::JobTracker
Key Management Server	KM	Hadoop::KMServer
HDFS NFS Gateway	NFS	Hadoop::NFSGateway
NameNode*	NN [†]	Hadoop::NameNode
YARN ResourceManager ²	RM	Hadoop::YARNServer
SecondaryNameNode	SNN	Hadoop::SecondaryNameNode
Spark YARN	SY	Hadoop::SparkYARN
Spark Master*	SM	Hadoop::SparkMaster
Spark Worker	SW	Hadoop::SparkWorker
Sqoop	SQ	Hadoop::Sqoop
ZooKeeper	ZK	Hadoop::ZooKeeper

* If these are in use, then modifying them should be done with great care due to the dependency of other roles on them

¹ for Hadoop v1

² for Hadoop v2

[†] Becomes NN1 and NN2 with high-availability

Each configuration group in figure 3.10 can be double-clicked in order to configure the group and their underlying role or roles. Double-clicking or using the `Open` button on a group, for example `doop-DN-default` in the figure, opens up an editor window within which the priority of the configuration group can be adjusted and other parameters of the underlying roles can also be adjusted, from within role subtabs (figures 3.11 and 3.12).

Edit Hadoop Configuration Group

Configuration Group:

Priority:

Nodes in Configuration Group: [Add/remove nodes](#)

Configure HDFS DataNode | **Configure YARN NodeManager**

Settings applied to:

Data directories: [-](#) [+](#)

Number of failed volumes tolerated:

Reserved space for Non DFS use: byte

Bandwidth for balancer: byte/sec

DataNode Java heap size: MB

DataNode ports: [✖](#) [👁](#) [👁](#)

IPC port:

HTTP port:

HTTPS port:

Transceiver port:

More DataNode parameters: [✖](#) [👁](#) [👁](#)

Handler count:

Max number of transfer threads:

Heartbeat interval: sec

[Add role](#) [Remove role](#) [Cancel](#) [Ok](#)

Figure 3.11: Hadoop Configuration Groups Tab, After Opening DataNode Configuration Group: DataNode Role Configuration

Edit Hadoop Configuration Group

Configuration Group:

Priority:

Nodes in Configuration Group: [Add/remove nodes](#)

Configure HDFS DataNode **Configure YARN NodeManager**

Settings applied to:

Localization directories: [+](#)

Log directories: [+](#)

Log aggregation enabled: ☐

Application log directory:

Application log directory suffix:

Log retain time: sec

More NodeManager parameters: [\[icon\]](#)

Docker container execution:

Shuffle service name:

Shuffle class:

ShuffleHandler port:

NodeManager heap size: MB

Container manager: [\[icon\]](#)

Container manager port:

Containers memory: MB

Physical memory limits enforced: ☒

Virtual memory limits enforced: ☒

Virtual/physical memory ratio:

Vcores capacity:

Monitoring interval: msec

Handler count:

NodeManager ports: [\[icon\]](#)

ApplicationMaster: [\[icon\]](#)

MapReduce jobs: [\[icon\]](#)

[Add role](#) [Remove role](#) [Cancel](#) [Ok](#)

Figure 3.12: Hadoop Configuration Groups Tab, After Opening DataNode Configuration Group: YARN NodeManager Role Configuration

There is a great deal of flexibility in dealing with configuration groups and roles. Configuration groups can be created, cloned, and removed using the buttons in figure 3.10, while roles that have been opened for editing can not only be modified, but also added or removed.

However, it should be noted that the asterisked roles in the preceding table are roles that other roles can depend upon. Modifying them should therefore only be done with extreme care. It is not difficult to misconfigure the Hadoop NameNode role so that it leads to the HDFS filesystem becoming unavailable, and hence to potential data loss.

Subsets of the configuration groups of figure 3.10 are displayed in the individual service resource tabs, such in the HDFS or Zookeeper resource tabs, under their individual (HDFS or Zookeeper) Configuration subtab. The subsets displayed are the ones associated with the resource.

For example: In the Hadoop Configuration Groups tab (figure 3.10) all the configuration groups are shown. On the other hand, in the HBase tab (figure 3.8) only the subset of HBase-related configuration groups are shown.

Double-clicking or using the Open button on a configuration group within a subset also opens up an editor window for the configuration group just as in figure 3.10.

Further roles can be assigned within the editor window by clicking on the Add Role button.

3.1.10 The HDFS Instance Monitoring Tab

The Monitoring tab pane (figure 3.13), displays metrics related to Hadoop monitoring.

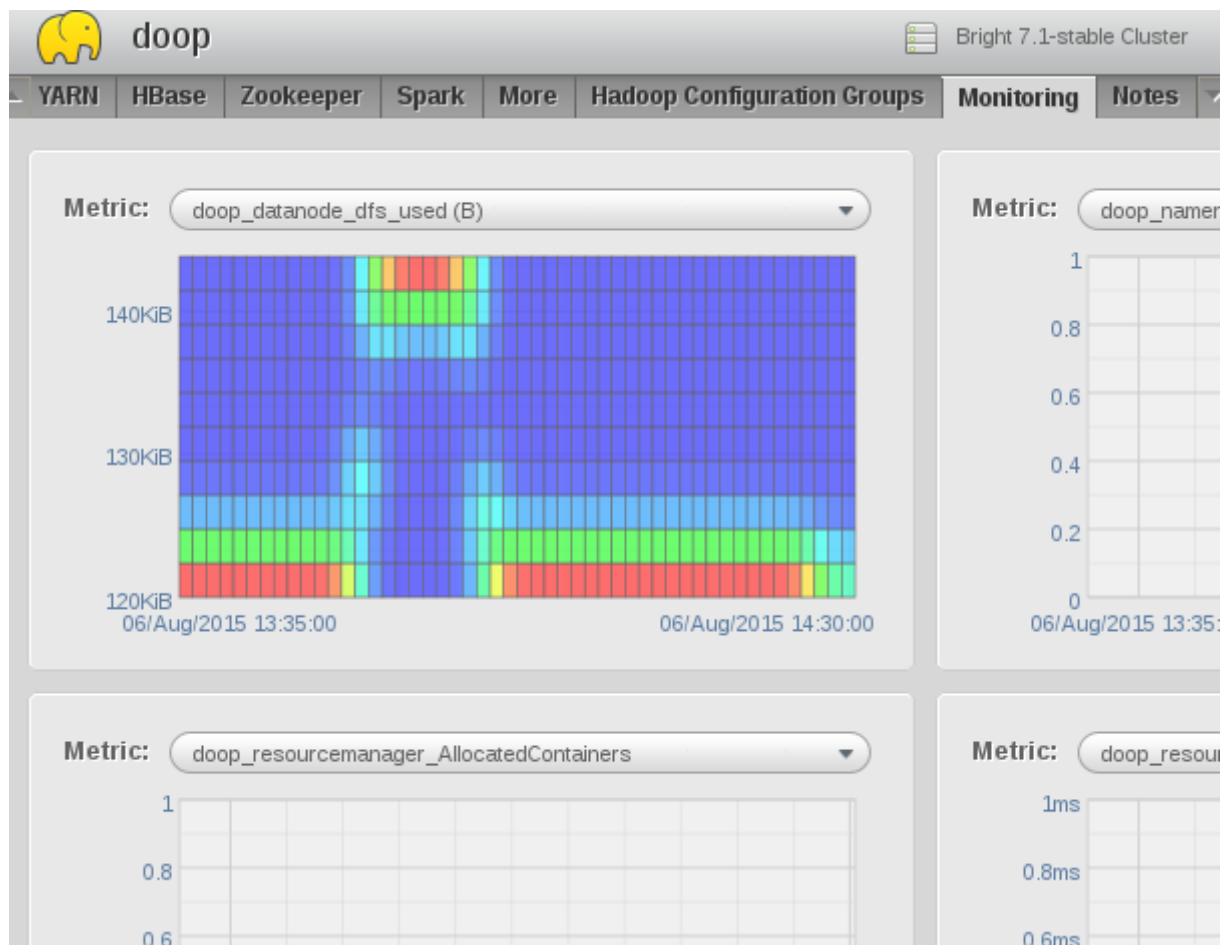


Figure 3.13: Monitoring Tab For A Hadoop Instance In `cmgui`

3.1.11 The HDFS Instance Notes Tab

This tab provides a simple notepad for the administrator for each Hadoop instance.

3.2 Managing A Hadoop Instance With `cmsh`

3.2.1 `cmsh` And `hadoop` Mode

The `cmsh` front end uses the `hadoop` mode to display information on Hadoop-related values and to carry out Hadoop-related tasks.

Example

```
[root@bright71 conf]# cmsh
[bright71]% hadoop
[bright71->hadoop]%
```

The `show` And `overview` Commands

The `overview` Command: Within `hadoop` mode, the `overview` command displays two sections of interest, that correspond somewhat to `cmgui`'s Overview tab in the Hadoop resource (section 3.1.1), providing Hadoop-related information on the system resources that are used. The first section gives an overview of the cluster state with regard to Hadoop usage. The second section shows the Hadoop role node assignments along with the configuration groups that the roles are associated with.

Example

```
[bright71->hadoop]% overview doop
```

Parameter	Value

Name	doop
Capacity total	52.72GB
Capacity used	282.3MB
Capacity remaining	43.97GB
Heap memory total	1.279GB
Heap memory used	305.5MB
Heap memory remaining	1004MB
Non-heap memory total	348MB
Non-heap memory used	252.3MB
Non-heap memory remaining	95.72MB
Nodes available	3
Nodes dead	0
Nodes decommissioned	0
Nodes decommission in progress	0
Total files	52
Total blocks	15
Missing blocks	0
Under-replicated blocks	0
Scheduled replication blocks	0
Pending replication blocks	0
Block report average Time	1000
Applications running	0
Applications pending	0
Applications submitted	0
Applications completed	0
Applications failed	0
Federation setup	no

Hadoop role	Nodes	Configuration group	Nodes up

Hadoop::DataNode	node001..node003	doop-DN-default	3 of 3
Hadoop::HBaseClient	node001,node002	doop-HBRS-default	2 of 2
Hadoop::HBaseServer	bright71	doop-HBM-default	1 of 1
Hadoop::NameNode	bright71	doop-NN-default	1 of 1
Hadoop::SecondaryNameNode	bright71	doop-SNN-default	1 of 1
Hadoop::SparkYARN	bright71	doop-SY-default	1 of 1
Hadoop::YARNClient	node001..node003	doop-DN-default	3 of 3
Hadoop::YARNServer	bright71	doop-RM-default	1 of 1
Hadoop::ZooKeeper	node001..node003	doop-ZK-default	3 of 3

The show Command: The show command displays parameters that correspond mostly to the Settings tab of cmgui in the Hadoop resource (section 3.1.2):

Example

```
[bright71->hadoop[doop]]% show
```

Parameter	Value

Automatic failover	Disabled
Buffer size	65536
Cluster ID	

```

Compression codecs
Configuration directory                /etc/hadoop/doop
Configuration directory for HBase      /etc/hadoop/doop/hbase
Configuration directory for Hive
Configuration directory for Spark      /etc/hadoop/doop/spark
Configuration directory for Sqoop
Configuration directory for ZooKeeper  /etc/hadoop/doop/zooke+
Connection maximum idle time          30000
Creation time                          Mon, 03 Aug 2015 16:56+
Delay for first block report           10
Enable HA for YARN                     no
Enable NFS gateway                     no
Enable WebHDFS                         no
HA enabled                            no
HA name service                        ha
HBase version                          1.0.0-cdh5.4.3
HDFS Permission                       yes
HDFS Umask                             022
HDFS audit enabled                     no
HDFS balancer period                   48
HDFS balancer policy                   dataNode
HDFS balancer threshold                10
HDFS default block size                134217728
HDFS default replication factor        3
HDFS maximum replication factor        50
Hadoop distribution                    Cloudera
Hadoop root for Lustre
Hadoop version                         2.6.0-cdh5.4.3
Hive version
Idle threshold number of connections  4000
Installation directory for HBase       /cm/shared/apps/hadoop+
Installation directory for Hadoop instance /cm/shared/apps/hadoop+
Installation directory for Hive
Installation directory for Spark       /cm/shared/apps/hadoop+
Installation directory for Sqoop
Installation directory for ZooKeeper   /cm/shared/apps/hadoop+
Log directory for Hadoop instance      /var/log/hadoop/doop
Maximum number of retries for IPC connections 30
Name                                   doop
Network
Readonly                              no
Revision
Root directory for data                /var/lib/hadoop/doop/
Serialization classes                  org.apache.hadoop.io.s+
Spark version
Sqoop version
Temporary directory for Hadoop instance /tmp/hadoop/doop/
Topology                              Switch
Use HTTPS                              no
Use Lustre                             no
Use federation                         no
Use only HTTPS                         no
Whether is a Spark instance            no
YARN automatic failover                Disabled
ZooKeeper version                      3.4.5-cdh5.4.3

```



```
description                               installed from: /cm/lo+
notes                                    notes here!
```

If setting or getting a value, then using the `set` or `get` command on its own within `hadoop` mode shows a help text that can help decide on what parameter should be set or gotten.

Example

```
[bright71->hadoop[doop]]% get
...
Parameters:
  Readonly ..... Mark data as readonly
  Revision ..... Entity revision
  automaticfailover ... Automatic failover can be controlled by\
                        either Hadoop itself of CMDaemon
  buffersize ..... Buffer size in I/O operations (in bytes\
                        ). Defined in io.file.buffer.size
  clusterid ..... Cluster ID for federation
  compressioncodecs ... Comma-separated list of compression cod\
                        ec classes. Defined in io.compression.codecs
  configurationdirectory Configuration directory
  configurationdirectoryforhbase Configuration directory for HBase
  configurationdirectoryforhive Configuration directory for Hive
  ...
```

The `*services` Commands For Hadoop Services

Hadoop services can be started, stopped, and restarted, with:

- `restartallservices`
- `startallservices`
- `stopallservices`

Example

```
[bright71->hadoop]% restartallservices apache121
Will now stop all Hadoop services for instance 'apache121'... done.
Will now start all Hadoop services for instance 'apache121'... done.
[bright71->hadoop]%
```

The `*balancer` Commands For Hadoop, And Related Parameters

For applications to have efficient access to HDFS, the file block level usage across nodes need to be reasonably balanced. The following balancer commands can be run from within `hadoop` mode:

- `startbalancer`: starts balancing
- `stopbalancer`: stops balancing
- `statusbalancer`: displays status of balancer

Example

```
[bright71->hadoop]% statusbalancer doop
Code: 1
Redirecting to /bin/systemctl status  hadoop-doop-balancer.service
hadoop-doop-balancer.service - Hadoop Balancer daemon for instance doop
  Loaded: loaded (/usr/lib/systemd/system/hadoop-doop-balancer.service
  static)
  Active: inactive (dead)
```

The following parameters:

- `hdfsbalancerperiod`
- `hdfsbalancerthreshold`
- `hdfsbalancerpolicy`

can be used to set or retrieve the period, threshold, and policy for the balancer running on the instance.

Example

```
[bright71->hadoop]% get doop hdfsbalancerperiod
2
[bright71->hadoop]% set doop balancerperiod 3
[bright71->hadoop*]% commit
[bright71->hadoop]% startbalancer doop
Code: 0
Starting Hadoop balancer daemon (hadoop-doop-balancer):starting ba\
lancer, logging to /var/log/hadoop/doop/hdfs/hadoop-hdfs-balancer-\
bright71.out
Time Stamp  Iteration#  Bytes Moved  Bytes To Move  Bytes Being Moved
The cluster is balanced. Exiting...

[bright71->hadoop]%
Thu Mar 20 15:27:02 2014 [notice] bright71: Started balancer for doop
For details type: events details 152727
```

The preceding Hadoop services and balancer commands run tasks and use parameters that correspond mostly to the HDFS tab (section 3.1.2) of `cmgui`.

The `formathdfs` Command

Usage:

```
formathdfs <HDFS>
```

The `formathdfs` command formats an instance so that it can be reused. Existing Hadoop services for the instance are stopped first before formatting HDFS, and started again after formatting is complete.

Example

```
[bright71->hadoop]% formathdfs doop
Will now format and set up HDFS for instance 'doop'.
Stopping datanodes... done.
Stopping namenodes... done.
Formatting HDFS... done.
Starting namenode (host 'bright71')... done.
Starting datanodes... done.
Waiting for datanodes to come up... done.
Setting up HDFS... done.
[bright71->hadoop]%
```

The `manualfailover` Command

Usage:

```
manualfailover [-f|--from <NameNode>] [-t|--to <other NameNode>] <HDFS>
```

The `manualfailover` command allows the active status of a NameNode to be moved to another NameNode in the Hadoop instance. This is only available for Hadoop instances within Hadoop distributions that support NameNode failover.

3.2.2 cmsh And configurationoverlay Mode

Hadoop configuration groups are introduced in section 3.1.9 as a special case of configuration overlays. Within cmgui Hadoop configuration groups can be accessed from within the tabs associated with a Hadoop instance (section 3.1.9).

Configuration Overlay Listing

In cmsh, the Hadoop configuration groups are listed and accessed via configurationoverlay mode:

Example

```
[root@bright71 ~]# cmsh
[bright71->configurationoverlay]% list -f name:17,nodes:16,roles:36
name (key)          nodes          roles
-----
doop-DN-default     node001..node003 Hadoop::DataNode, Hadoop::YARNClient
doop-HBM-default    bright71        Hadoop::HBaseServer
doop-HBRS-default   node001,node002 Hadoop::HBaseClient
doop-NN-default     bright71        Hadoop::NameNode
doop-RM-default     bright71        Hadoop::YARNServer
doop-SNN-default    bright71        Hadoop::SecondaryNameNode
doop-SY-default     bright71        Hadoop::SparkYARN
doop-ZK-default     node001..node003 Hadoop::ZooKeeper
```

Configuration Overlay Mode And Configuration Overlay Properties

A configuration overlay object can be used. That is, the shell can drop within a particular Hadoop configuration group with the use command. The properties of the object, that is the Hadoop configuration group, can then be shown:

Example

```
[bright71->configurationoverlay]% use doop-dn-default
[bright71->configurationoverlay[doop-DN-default]]% show
Parameter          Value
-----
Categories
Name                doop-DN-default
Nodes               node001..node003
Priority             500
Readonly             no
Revision
Roles                Hadoop::DataNode, Hadoop::YARNClient
```

Configuration Overlay Roles Submode, And Role Properties – All Instances

A roles submode can be entered within the configuration overlay object. That is, the Hadoop configuration group roles submode can be entered. The roles that the configuration overlay is associated with can be listed:

Example

```
[bright71->configurationoverlay[doop-DN-default]]% roles
[bright71->configurationoverlay[doop-DN-default]->roles]% list
Name (key)
-----
Hadoop::DataNode
Hadoop::YARNClient
```

A particular role can be used and its CMDaemon properties, relevant to all instances, viewed and modified:

```
...figurationoverlay[doop-DN-default]->roles]% use hadoop::datanode
...figurationoverlay[doop-DN-default]->roles[Hadoop::DataNode]]% show
Parameter                                     Value
-----
Configurations                               <1 in submode>
Name                                           Hadoop::DataNode
Provisioning associations                     <0 internally used>
Readonly                                       no
Revision
Type                                           HadoopDataNodeRole
```

Configuration Overlay Roles Submode, Role Properties – For A Selected Instance

Within a role, the `configurations` submode can be used to modify the properties of the role itself. The configuration list shows which instances are available.

Example

```
[...-DN-default]->roles[Hadoop::DataNode]]% configurations
[...-DN-default]->roles[Hadoop::DataNode]->configurations]% list
HDFS
-----
doop
doop2
```

Choosing an instance means that configuration settings will apply only to that instance. In the following example, the `doop` instance is chosen:

```
[...-DN-default]->roles[Hadoop::DataNode]->configurations]% use doop
[...-DN-default]->roles[Hadoop::DataNode]->configurations[doop]]% show
Parameter                                     Value
-----
Bandwidth for balancer                       1048576
Data directories                             /var/lib/hadoop/doop/hadoop/hdfs/
                                              datanode
DataNode Java heap size                     512
HDFS                                          doop
HTTP port                                    50075
HTTPS port                                   50475
Handler count                                10
Heap size                                    0
Heartbeat interval                           3
Maximum number of transfer threads           4096
Network
Number of failed volumes tolerated            0
Protocol port                                50020
Readonly                                      no
Reserved spaced for Non DFS use              1073741824
Revision
Transceiver port                             50010
Type                                           HadoopDataNodeHDFSConfiguration
```

The properties available here for the `Hadoop::DataNode` role correspond to the properties shown in the `Configure HDFS DataNode` subtab for figure 3.11.

3.2.3 `cmsh` And The `roleoverview` Command In `device` Mode

The `roleoverview` command can be run from `device` mode. It gives an overview of the roles associated with nodes, categories, and configuration overlays.

Example

```
[bright71->device]% roleoverview
```

Role	Nodes	Categories	Configuration Overlays
Hadoop::DataNode	node001..node003		doop-DN-default
Hadoop::HBaseClient	node001,node002		doop-HBRS-default
Hadoop::HBaseServer	bright71		doop-HBM-default
Hadoop::NameNode	bright71		doop-NN-default
Hadoop::SparkYARN	bright71		doop-SY-default
Hadoop::YARNClient	node001..node003		doop-DN-default
Hadoop::YARNServer	bright71		doop-RM-default
Hadoop::ZooKeeper	node001..node003		doop-ZK-default
boot	bright71		
login	bright71		
master	bright71		
monitoring	bright71		
provisioning	bright71		
slurmclient	node001..node003	default	
slurmserver	bright71		
storage	bright71		

3.3 Hadoop Maintenance Operations With `cm-hadoop-maint`

The Hadoop maintenance script, `cm-hadoop-maint`, is a Python script. It is called using the full path, and is run with no arguments displays a help page:

Example

```
[root@bright71 ~]# /cm/local/apps/cluster-tools/hadoop/cm-hadoop-maint
```

Hadoop instance name must be specified. Exiting.

```
USAGE: /cm/local/apps/cluster-tools/hadoop/cm-hadoop-maint -i <name>
[ -b | -f | --start | --stop | --restart | --startonly <set> |
  --stoponly <set> | --restartonly <set> | --enterSafeMode |
  --leaveSafeMode | --failover [ <from> <to> ] | --failoverstatus |
  --yarnfailover [ <from> <to> ] | --yarnfailoverstatus |
  --copyconfig <nodes> | --prepare <nodes> | -h ]
```

OPTIONS:

```
-i <name>          -- instance name
-b                -- cluster balancer utility
-f                -- format & init HDFS
--start           -- start all services
--stop            -- stop all services
--restart         -- restart all services
--startonly <set> -- start all services for <set>
--stoponly <set>  -- stop all services for <set>
--restartonly <set> -- restart all services for <set>
--enterSafeMode   -- enter safemode
--leaveSafeMode   -- leave safemode
```

```

--failover          -- executes a manual failover for HDFS
--failoverstatus    -- returns failover status for HDFS
--yarnfailover      -- executes a manual failover for YARN
--yarnfailoverstatus -- returns failover status for YARN
--copyconfig <nodes> -- copies Hadoop configuration files to nodes
                    (e.g. login nodes)
--prepare <nodes>   -- prepare nodes to be used for Hadoop
                    deployment (e.g. new nodes)
-h                  -- show usage

```

`<set>` can be one of the following values: `hdfs`, `mapred`, `yarn`, `zk`, `hbase`, `spark`, `sqoop`, `hive`

EXAMPLES:

```

cm-hadoop-maint -i hdfs1 -f
cm-hadoop-maint -i hdfs2 --stop
cm-hadoop-maint -i hdfs2 --stoponly hdfs
cm-hadoop-maint -i hdfs1 --failover nn1 nn2
    executes failover from nn1 to nn2
cm-hadoop-maint -i hdfs1 --failover
    executes failover from active to standby namenode
    if both namenodes are standby, automatically chooses one
cm-hadoop-maint -i hdfs1 --copyconfig node005..node007

```

If Hadoop is used with options, then the name of the Hadoop instance, specified with `-i`, is mandatory.

The other options are now explained in some more detail:

- `-b` starts the balancer daemon
- `-f` formats the Hadoop filesystem and reinitializes it with a standard set of directories, e.g. `/user`, `/tmp`.
- `--start`, `--stop`, `--restart` allow administrators to start, stop, or restart all services relevant to the Hadoop instance.

To operate on a one of the services only, the suffix `only` is appended to the options, and the service is specified as the parameter to the option. The specific service is chosen from `hdfs`, `mapred`, `yarn`, `zk`, `hbase`, `spark`, `sqoop`, or `hive`, so that the format for these options is:

- `--startonly <service>`
- `--stoponly <service>`
- `--restartonly <service>`

- `--enterSafeMode` and `--leaveSafeMode` act on the “safe mode” state of NameNode
- `--failover`, `--yarnfailover`: trigger a failover for HDFS, or for YARN
- `--failoverstatus`, `--yarnfailoverstatus`: get the status of High Availability for HDFS or for YARN
- `--copyconfig <nodes>`: Copies Hadoop configuration files to one or more nodes. For example, a Hadoop administrator may wish to add a login node to the Hadoop instance. The login node needs to have relevant Hadoop configuration files, under `/etc/hadoop`. The administrator assigns the login role to the node, and then copies configuration files with the `--copyconfig` option.

- `--prepare <nodes>`: Prepares a node that has a different image for use with the Hadoop instance. For example, a Hadoop administrator may wish to add a new node, such as a DataNode, to the Hadoop instance. If the new node has to use a software image that the other Hadoop nodes are already using, then the new node is automatically provisioned with the needed Hadoop configuration files and directories. However, if the new node is to use a different software image, then the new node is not automatically provisioned. It should instead be “prepared” with the option `--prepare`. Running the script with this option provisions the node. After the node has rebooted and is up and running again, the node should be added by the administrator to the Hadoop instance by using Hadoop configuration groups.

4

Running Hadoop Jobs

4.1 Shakedown Runs

The `cm-hadoop-tests.sh` script is provided as part of Bright Cluster Manager's `cluster-tools` package. The administrator can use the script to conveniently submit example jar files in the Hadoop installation to a job client of a Hadoop instance:

```
[root@bright71 ~]# cd /cm/local/apps/cluster-tools/hadoop/
[root@bright71 hadoop]# ./cm-hadoop-tests.sh <instance>
```

The script runs endlessly, and runs several Hadoop test scripts. If most lines in the run output are elided for brevity, then the structure of the truncated output looks something like this in overview:

Example

```
[root@bright71 hadoop]# ./cm-hadoop-tests.sh apache220
...
=====
Press [CTRL+C] to stop...
=====
...
=====
start cleaning directories...
=====
...
=====
clean directories done
=====

=====
start doing gen_test...
=====
...
14/03/24 15:05:37 INFO terasort.TeraSort: Generating 10000 using 2
14/03/24 15:05:38 INFO mapreduce.JobSubmitter: number of splits:2
...
    Job Counters
        ...
    Map-Reduce Framework
        ...
    org.apache.hadoop.examples.terasort.TeraGen$Counters
...
14/03/24 15:07:03 INFO terasort.TeraSort: starting
```



```

...
14/03/24 15:09:12 INFO terasort.TeraSort: done
...
=====
gen_test done
=====

=====
start doing PI test...
=====

Working Directory = /user/root/bbp
...

```

During the run, the Overview tab in cmgui (introduced in section 3.1.1) for the Hadoop instance should show activity as it refreshes its overview every three minutes (figure 4.1):

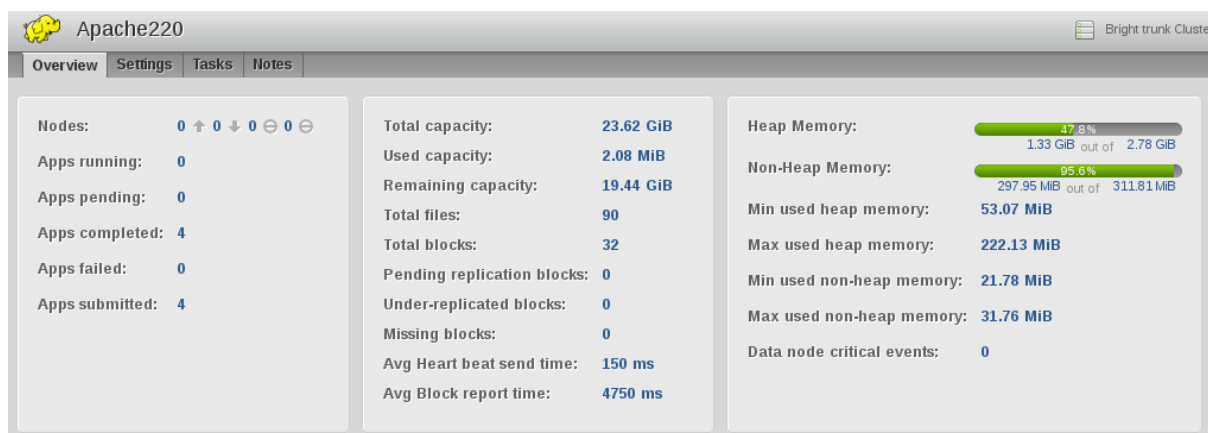


Figure 4.1: Overview Of Activity Seen For A Hadoop Instance In cmgui

In cmsh the overview command shows the most recent values that can be retrieved when the command is run:

```
[mk-hadoop-centos6->hadoop]% overview apache220
```

Parameter	Value
Name	Apache220
Capacity total	27.56GB
Capacity used	7.246MB
Capacity remaining	16.41GB
Heap memory total	280.7MB
Heap memory used	152.1MB
Heap memory remaining	128.7MB
Non-heap memory total	258.1MB
Non-heap memory used	251.9MB
Non-heap memory remaining	6.155MB
Nodes available	3
Nodes dead	0
Nodes decommissioned	0
Nodes decommission in progress	0
Total files	72
Total blocks	31
Missing blocks	0

```

Under-replicated blocks      2
Scheduled replication blocks 0
Pending replication blocks   0
Block report average Time    59666
Applications running         1
Applications pending         0
Applications submitted       7
Applications completed       6
Applications failed          0
High availability            Yes (automatic failover disabled)
Federation setup             no

```

Role	Node
DataNode, Journal, NameNode, YARNClient, YARNServer, ZooKeeper	node001
DataNode, Journal, NameNode, YARNClient, ZooKeeper	node002

4.2 Example End User Job Run

Running a job from a jar file individually can be done by an end user.

An end user `fred` can be created and issued a password by the administrator (Chapter 6 of the *Administrator Manual*). The user must then be granted HDFS access for the Hadoop instance by the administrator:

Example

```
[bright71->user[fred]]% set hadoopdfsaccess apache220; commit
```

The possible instance options are shown as tab-completion suggestions. The access can be unset by leaving a blank for the instance option.

The user `fred` can then submit a run from a pi value estimator, from the example jar file, as follows (some output elided):

Example

```

[fred@bright71 ~]$ module add hadoop/Apache220/Apache/2.2.0
[fred@bright71 ~]$ hadoop jar $HADOOP_PREFIX/share/hadoop/mapreduce/hado\
op-mapreduce-examples-2.2.0.jar pi 1 5
...
Job Finished in 19.732 seconds
Estimated value of Pi is 4.00000000000000000000

```

The `module add` line is not needed if the user has the module loaded by default (section 2.2.3 of the *Administrator Manual*).

The input takes the number of maps and number of samples as options—1 and 5 in the example. The result can be improved with greater values for both.

Spark support in Bright Cluster Manager

Apache Spark is an engine for processing Hadoop data. It can carry out general data processing, similar to MapReduce, but typically faster.

Spark can also carry out the following, with the associated high-level tools:

- stream feed processing with Spark Streaming
- SQL queries on structured distributed data with Spark SQL
- processing with machine learning algorithms, using MLlib
- graph computation, for arbitrarily-connected networks, with graphX

The Apache Spark tarball can be downloaded from <http://spark.apache.org/>. Different pre-built tarballs are available there, for Hadoop 1.x, for CDH 4, and for Hadoop 2.x.

Apache Spark can be installed on top of an existing Hadoop instance (section 5.1) or without Hadoop (section 5.1.2).

5.1 Spark Installation In Bright Cluster Manager

Bright Cluster Manager provides `cm-spark-setup` to carry out Spark installation.

5.1.1 Prerequisites For Spark Installation, And What Spark Installation Does

The following applies to using `cm-spark-setup`:

- A Hadoop instance is typically already installed. Spark can however be installed without HDFS (section 5.1.2).
- Spark can be installed in two different deployment modes: Standalone or YARN.
 - Standalone mode. This is the default for Apache Hadoop 1.x, Cloudera CDH 4.x, and Hortonworks HDP 1.3.x.
 - * It is possible to force the Standalone mode deployment by using the additional option:
`--standalone`
 - * When installing in standalone mode, the script installs Spark on the active head node and on the DataNodes of the chosen Hadoop instance.
The Spark Master service runs on the active head node by default, but can be specified to run on another node by using the option `--master`.
Spark Worker services run on all DataNodes if HDFS is running. If HDFS is not running (section 5.1.2), then Spark Worker services run on all nodes specified with the `--workernodes` option.

- YARN mode. This is the default for Apache Hadoop 2.x, Cloudera CDH 5.x, Hortonworks 2.x, and Pivotal 2.x. The default can be overridden by using the `--standalone` option.
 - * When installing in YARN mode, the script installs Spark only on the active head node.
- Depending on the installation mode, the script creates a one or more dedicated Hadoop Configuration Groups for Spark:
 - Standalone mode. Two Hadoop Configuration Groups will be created, one for Spark Master and one for Spark Worker roles.
 - YARN mode. Only one Hadoop Configuration Group will be created, for Spark YARN role.
- Spark is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Spark configuration files are copied by the script to under `/etc/hadoop/`
- When installing Spark on a Bright Cluster Manager which has Lustre running on it, and has a Hadoop instance installed on top of it, as described in section 2.4, then both installation modes are available:
 - Standalone mode: Only nodes that can access LustreFS should be selected as worker nodes. It is recommended to set `SPARK_WORKER_DIR` to use a subdirectory of LustreFS that uses the hostname as part of its path, in order to avoid having different workers using the same directory. The additional option `--workerdir` can be used. Care may be needed to escape characters:

Example

```
--workerdir "/mnt/hadoop/tmp/spark-\'hostname\'/'
```

- YARN mode: Configurations are written to the NodeManager. Subsequent operations with Spark should then be carried out on that node.

5.1.2 Spark Installation With `cm-spark-setup`

The `cm-spark-setup` utility has the following usage:

```
USAGE: /cm/local/apps/cluster-tools/bin/cm-spark-setup [\
  [-i <name> | --is <name>] -j <path> -t <file> [--standalone]\
  [--master <host>] [--workernodes <hosts>] | -u <name> | -h ]
```

OPTIONS:

```
-i <name>           -- instance name
-j <path>           -- Java home path
-t <file>           -- Spark tarball
--standalone        -- force install in Standalone mode
--master <host>     -- host to use as master
--workernodes <hosts> -- hosts to use as workernodes
--workerdir         -- directory for workers
-u <name>           -- uninstall Spark for instance <name>
-h                 -- show usage
```

`cm-spark-setup` With A Pre-Existing HDFS

Spark can be installed with a pre-existing Hadoop instance.

Spark Installed In YARN Mode: The following `cm-spark-setup` installation session shows a Spark tarball being installed in YARN mode with an existing Hadoop instance `hdfs1` with a Java 1.7.0 runtime environment:

Example

```
[root@bright71 ~]# cm-spark-setup -i hdfs1 \
-j /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/ \
-t /tmp/spark-1.3.0-bin-hadoop2.4.tgz
Spark release '1.3.0-bin-hadoop2.4'
Found Hadoop instance 'hdfs1', release: 2.6.0
Spark will be installed in YARN (client/cluster) mode.
Spark being installed... done.
Creating directories for Spark... done.
Creating module file for Spark... done.
Creating configuration files for Spark... done.
Waiting for NameNode to be ready... done.
Copying Spark assembly jar to HDFS... done.
Waiting for NameNode to be ready... done.
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

Spark Installed In Standalone Mode: The following `cm-spark-setup` installation session shows a Spark tarball being installed in Standalone mode with an existing Hadoop instance `hdfs1` with a Java 1.7.0 runtime environment, and with an alternative Spark Master service running on `node005`:

Example

```
[root@bright71 ~]# cm-spark-setup -i hdfs1 \
-j /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/ \
-t /tmp/spark-1.3.0-bin-hadoop2.4.tgz \
--standalone --master node005
Spark release '1.3.0-bin-hadoop2.4'
Found Hadoop instance 'hdfs1', release: 2.6.0
Spark will be installed to work in Standalone mode.
Spark Master service will be run on node: node005
Spark will use all DataNodes as WorkerNodes.
Spark being installed... done.
Creating directories for Spark... done.
Creating module file for Spark... done.
Creating configuration files for Spark... done.
Updating images... done.
Initializing Spark Master service... done.
Initializing Spark Worker service... done.
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

`cm-spark-setup` Without A Pre-Existing HDFS

Spark can also be installed in Standalone mode without requiring a pre-existing Hadoop instance. The Spark instance name can then be specified. When using `cm-spark-setup` for this case, the Spark Worker services will run on all nodes that are specified with option `--workernodes`.

Example

```
[root@bright71 ~]# cm-spark-setup --is Spark131 \
-j /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/ \
-t /root/spark-1.3.1-bin-hadoop2.6.tgz \
--master node001 --workernodes node002..node006
```

5.2 Spark Removal With `cm-spark-setup`

`cm-spark-setup` uses the `-u` option to uninstall the Spark instance.

Example

```
[root@bright71 ~]# cm-spark-setup -u hdfs1
Requested removal of Spark for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Spark directories... done.
Removal successfully completed.
Finished.
```

5.3 Using Spark

Spark can be run in YARN mode (section 5.3.1) or Standalone mode (section 5.3.2).

5.3.1 Using Spark In YARN Mode

Spark supports two deploy modes for launching Spark applications on YARN:

- `yarn-client`
- `yarn-cluster`

An example Spark application that comes with the Spark installation is SparkPi. SparkPi can be launched in the two deploy modes as follows:

1. In `yarn-client` mode, the Spark driver runs as a client process. The SparkPi application then runs as a child thread of Application Master.

```
[root@bright71 ~]# module load spark/hdfs1
[root@bright71 ~]# spark-submit --master yarn-client \
--class org.apache.spark.examples.SparkPi \
$SPARK_PREFIX/lib/spark-examples-*.jar
```

2. In `yarn-cluster` mode, the Spark driver runs inside an Application Master process. This is then managed by YARN on the cluster.

```
[root@bright71 ~]# module load spark/hdfs1
[root@bright71 ~]# spark-submit --master yarn-cluster \
--class org.apache.spark.examples.SparkPi \
$SPARK_PREFIX/lib/spark-examples-*.jar
```

5.3.2 Using Spark In Standalone Mode

The SparkPi application can be run in standalone mode as follows:

Example

```
[root@bright71 ~]# module load spark/hdfs1
[root@bright71 ~]# spark-submit --class org.apache.spark.examples.SparkPi \
/cm/shared/apps/hadoop/Adobe/spark-1.3.1-bin-hadoop2.6/\
lib/spark-examples-1.3.1-hadoop2.6.0.jar
15/06/15 15:33:52 INFO SparkContext: Running Spark version 1.3.1

...

15/06/15 15:34:05 INFO DAGScheduler: Job 0 finished:\
  reduce at SparkPi.scala:35, took 9.313538 s
Pi is roughly 3.14238
15/06/15 15:34:05 INFO ContextHandler: stopped\
  o.s.j.s.ServletContextHandler/metrics/json,null

...

15/06/15 15:34:06 INFO RemoteActorRefProvider$RemotingTerminator:\
  Remote daemon shut down; proceeding with flushing remote transports.
[root@bright71 ~]#
```


6

Hadoop-related Projects

Several projects use the Hadoop framework. These projects may be focused on data warehousing, data-flow programming, or other data-processing tasks which Hadoop can handle well. Bright Cluster Manager provides tools to help install the following projects:

- Accumulo (section 6.1)
- Hive (section 6.2)
- Kafka (section 6.3)
- Pig (section 6.4)
- Sqoop (section 6.5)
- Storm (section 6.6)

6.1 Accumulo

Apache Accumulo is a highly-scalable, structured, distributed, key-value store based on Google's BigTable. Accumulo works on top of Hadoop and ZooKeeper. Accumulo stores data in HDFS, and uses a richer model than regular key-value stores. Keys in Accumulo consist of several elements.

An Accumulo instance includes the following main components:

- Tablet Server, which manages subsets of all tables
- Garbage Collector, to delete files no longer needed
- Master, responsible of coordination
- Tracer, collection traces about Accumulo operations
- Monitor, web application showing information about the instance

Also a part of the instance is a client library linked to Accumulo applications.

The Apache Accumulo tarball can be downloaded from <http://accumulo.apache.org/>. For Hortonworks HDP 2.1.x, the Accumulo tarball can be downloaded from the Hortonworks website (section 1.2).

6.1.1 Accumulo Installation With `cm-accumulo-setup`

Bright Cluster Manager provides `cm-accumulo-setup` to carry out the installation of Accumulo.

Prerequisites For Accumulo Installation, And What Accumulo Installation Does

The following applies to using `cm-accumulo-setup`:

- A Hadoop instance, with ZooKeeper, must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- The `cm-accumulo-setup` script only installs Accumulo on the active head node and on the DataNodes of the chosen Hadoop instance.
- The script assigns no roles to nodes.
- Accumulo executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`.
- Accumulo configuration files are copied by the script to under `/etc/hadoop/`. This is done both on the active headnode, and on the necessary image(s).
- By default, Accumulo Tablet Servers are set to use 1GB of memory. A different value can be set via `cm-accumulo-setup`.
- The secret string for the instance is a random string created by `cm-accumulo-setup`.
- A password for the `root` user must be specified.
- The Tracer service will use Accumulo user `root` to connect to Accumulo.
- The services for Garbage Collector, Master, Tracer, and Monitor are, by default, installed and run on the headnode. They can be installed and run on another node instead, as shown in the next example, using the `--master` option.
- A Tablet Server will be started on each DataNode.
- `cm-accumulo-setup` tries to build the native map library.
- Validation tests are carried out by the script.
- When installing Accumulo on a Hadoop instance configured to run on Lustre within Bright Cluster Manager (section 2.4), the services for Garbage Collector, Master, Tracer, and Monitor will be run on the node which is the ResourceManager.

The options for `cm-accumulo-setup` are listed on running `cm-accumulo-setup -h`.

An Example Run With `cm-accumulo-setup`

The option

`-p <rootpass>`

is mandatory. The specified password will also be used by the Tracer service to connect to Accumulo. The password will be stored in `accumulo-site.xml`, with read and write permissions assigned to `root` only.

The option

`-s <heapsize>`

is not mandatory. If not set, a default value of 1GB is used.

The option

`--master <nodename>`

is not mandatory. It is used to set the node on which the Garbage Collector, Master, Tracer, and Monitor services run. If not set, then these services are run on the head node by default.

Example

```
[root@bright71 ~]# cm-accumulo-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-\
openjdk.x86_64/ -p <rootpass> -s <heapsize> -t /tmp/accumulo-1.6.2-bin.tar.gz \
--master node005
Accumulo release '1.6.2'
Accumulo GC, Master, Monitor, and Tracer services will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.6.0
Accumulo being installed... done.
Creating directories for Accumulo... done.
Creating module file for Accumulo... done.
Creating configuration files for Accumulo... done.
Updating images... done.
Setting up Accumulo directories in HDFS... done.
Executing 'accumulo init'... done.
Initializing services for Accumulo (on DataNodes)... done.
Initializing master services for Accumulo... done.
Waiting for NameNode to be ready... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

6.1.2 Accumulo Removal With `cm-accumulo-setup`

`cm-accumulo-setup` should also be used to remove the Accumulo instance. Data and metadata will not be removed.

Example

```
[root@bright71 ~]# cm-accumulo-setup -u hdfs1
Requested removal of Accumulo for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Accumulo directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.1.3 Accumulo MapReduce Example

Accumulo jobs must be run using `accumulo` system user.

Example

```
[root@bright71 ~]# su - accumulo
bash-4.1$ module load accumulo/hdfs1
bash-4.1$ cd $ACCUMULO_HOME
bash-4.1$ bin/tool.sh lib/accumulo-examples-simple.jar \
org.apache.accumulo.examples.simple.mapreduce.TeraSortIngest \
-i hdfs1 -z $ACCUMULO_ZOOKEEPERS -u root -p secret \
--count 10 --minKeySize 10 --maxKeySize 10 \
--minValueSize 78 --maxValueSize 78 --table sort --splits 10
```

6.2 Hive

Apache Hive is a data warehouse software. It stores its data using HDFS, and can query it via the SQL-like HiveQL language. Metadata values for its tables and partitions are kept in the Hive Metastore, which is an SQL database, typically MySQL or Postgres. Data can be exposed to clients using the following client-server mechanisms:

- Metastore, accessed with the `hive` client
- HiveServer2, accessed with the `beeline` client

The Apache Hive tarball should be downloaded from one of the locations specified in Section 1.2, depending on the chosen distribution.

6.2.1 Hive Installation With `cm-hive-setup`

Bright Cluster Manager provides `cm-hive-setup` to carry out Hive installation:

Prerequisites For Hive Installation, And What Hive Installation Does

The following applies to using `cm-hive-setup`:

- A Hadoop instance must already be installed.
- Before running the script, the version of the `mysql-connector-java` package should be checked. Hive works with releases 5.1.18 or earlier of this package. If `mysql-connector-java` provides a newer release, then the following must be done to ensure that Hive setup works:
 - a suitable 5.1.18 or earlier release of Connector/J is downloaded from `http://dev.mysql.com/downloads/connector/j/`
 - `cm-hive-setup` is run with the `--conn` option to specify the connector version to use.

Example

```
--conn /tmp/mysql-connector-java-5.1.18-bin.jar
```

- Before running the script, the following statements must be executed explicitly by the administrator, using a MySQL client:

```
GRANT ALL PRIVILEGES ON <metastoredb>.* TO 'hive'@'%' \
  IDENTIFIED BY '<hivepass>';
FLUSH PRIVILEGES;
DROP DATABASE IF EXISTS <metastoredb>;
```

In the preceding statements:

- `<metastoredb>` is the name of metastore database to be used by Hive. The same name is used later by `cm-hive-setup`.
 - `<hivepass>` is the password for `hive` user. The same password is used later by `cm-hive-setup`.
 - The DROP line is needed only if a database with that name already exists.
- The `cm-hive-setup` script installs Hive by default on the active head node. It can be installed on another node instead, as shown in the next example, with the use of the `--master` option. In that case, Connector/J should be installed in the software image of the node.
 - The script creates a dedicated Hadoop Configuration Group for Hive.
 - Hive executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`
 - Hive configuration files are copied by the script to under `/etc/hadoop/`
 - The instance of MySQL on the head node is initialized as the Metastore database for the Bright Cluster Manager by the script. A different MySQL server can be specified by using the options `--mysqlserver` and `--mysqlport`.

- The data warehouse is created by the script in HDFS, in `/user/hive/warehouse`
- The Metastore and HiveServer2 services are started up by the script
- Validation tests are carried out by the script using `hive` and `beeline`.
- When installing Hive on a Hadoop instance configured to run on Lustre within Bright Cluster Manager (section 2.4), Hive should be deployed on a node that has access to LustreFS (by using the `--master` option if needed). Subsequent operations with Hive should be carried out on that node.

An Example Run With `cm-hive-setup`

Example

```
[root@bright71 ~]# cm-hive-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-op\
enjdk.x86_64/ -p <hivepass> --metastoredb <metastoredb> -t /tmp/apache\
-hive-1.1.0-bin.tar.gz --master node005
Hive release '1.1.0-bin'
Using MySQL server on active headnode.
Hive service will be run on node: node005
Using MySQL Connector/J installed in /usr/share/java/
Hive being installed... done.
Creating directories for Hive... done.
Creating module file for Hive... done.
Creating configuration files for Hive... done.
Initializing database 'metastore_hdfs1' in MySQL... done.
Waiting for NameNode to be ready... done.
Creating HDFS directories for Hive... done.
Updating images... done.
Waiting for NameNode to be ready... done.
Hive setup validation...
-- testing 'hive' client...
-- testing 'beeline' client...
Hive setup validation... done.
Installation successfully completed.
Finished.
```

6.2.2 Hive Removal With `cm-hive-setup`

`cm-hive-setup` should also be used to remove the Hive instance. Data and metadata will not be removed.

Example

```
[root@bright71 ~]# cm-hive-setup -u hdfs1
Requested removal of Hive for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Hive directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.2.3 Beeline

The latest Hive releases include HiveServer2, which supports Beeline command shell. Beeline is a JDBC client based on the SQLLine CLI (<http://sqlline.sourceforge.net/>). In the following example, Beeline connects to HiveServer2:

Example

```
[root@bright71 ~]# beeline -u jdbc:hive2://node005.cm.cluster:10000 \
  -d org.apache.hive.jdbc.HiveDriver -e 'SHOW TABLES;'
Connecting to jdbc:hive2://node005.cm.cluster:10000
Connected to: Apache Hive (version 1.1.0)
Driver: Hive JDBC (version 1.1.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
+-----+
| tab_name |
+-----+
| test     |
| test2    |
+-----+
2 rows selected (0.243 seconds)
Beeline version 1.1.0 by Apache Hive
Closing: 0: jdbc:hive2://node005.cm.cluster:10000
```

6.3 Kafka

Apache Kafka is a distributed publish-subscribe messaging system. Among other usages, Kafka is used as a replacement for message broker, for website activity tracking, for log aggregation. The Apache Kafka tarball should be downloaded from <http://kafka.apache.org/>, where different pre-built tarballs are available, depending on the preferred Scala version.

6.3.1 Kafka Installation With `cm-kafka-setup`

Bright Cluster Manager provides `cm-kafka-setup` to carry out Kafka installation.

Prerequisites For Kafka Installation, And What Kafka Installation Does

The following applies to using `cm-kafka-setup`:

- A Hadoop instance, with ZooKeeper, must already be installed.
- `cm-kafka-setup` installs Kafka only on the ZooKeeper nodes.
- The script assigns no roles to nodes.
- Kafka is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Kafka configuration files are copied by the script to under `/etc/hadoop/`.

An Example Run With `cm-kafka-setup`

Example

```
[root@bright71 ~]# cm-kafka-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-open\
jdk.x86_64/ -t /tmp/kafka_2.11-0.8.2.1.tgz
Kafka release '0.8.2.1' for Scala '2.11'
Found Hadoop instance 'hdfs1', release: 1.2.1
Kafka being installed... done.
Creating directories for Kafka... done.
Creating module file for Kafka... done.
Creating configuration files for Kafka... done.
Updating images... done.
Initializing services for Kafka (on ZooKeeper nodes)... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

6.3.2 Kafka Removal With `cm-kafka-setup`

`cm-kafka-setup` should also be used to remove the Kafka instance.

Example

```
[root@bright71 ~]# cm-kafka-setup -u hdfs1
Requested removal of Kafka for Hadoop instance hdfs1.
Stopping/removing services... done.
Removing module file... done.
Removing additional Kafka directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.4 Pig

Apache Pig is a platform for analyzing large data sets. Pig consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. Pig programs are intended by language design to fit well with “embarrassingly parallel” problems that deal with large data sets. The Apache Pig tarball should be downloaded from one of the locations specified in Section 1.2, depending on the chosen distribution.

6.4.1 Pig Installation With `cm-pig-setup`

Bright Cluster Manager provides `cm-pig-setup` to carry out Pig installation.

Prerequisites For Pig Installation, And What Pig Installation Does

The following applies to using `cm-pig-setup`:

- A Hadoop instance must already be installed.
- `cm-pig-setup` installs Pig only on the active head node.
- The script assigns no roles to nodes.
- Pig is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Pig configuration files are copied by the script to under `/etc/hadoop/`.
- When installing Pig on a Hadoop instance configured to run on Lustre within Bright Cluster Manager (section 2.4), Pig configuration files will be automatically copied to a node that has access to LustreFS (NodeManager). Subsequent operations with Pig should be carried out on that node.

An Example Run With `cm-pig-setup`

Example

```
[root@bright71 ~]# cm-pig-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-open\
jdk.x86_64/ -t /tmp/pig-0.15.0.tar.gz
Pig release '0.15.0'
Pig being installed... done.
Creating directories for Pig... done.
Creating module file for Pig... done.
Creating configuration files for Pig... done.
Waiting for NameNode to be ready...
Waiting for NameNode to be ready... done.
Validating Pig setup...
Validating Pig setup... done.
Installation successfully completed.
Finished.
```


6.4.2 Pig Removal With `cm-pig-setup`

`cm-pig-setup` should also be used to remove the Pig instance.

Example

```
[root@bright71 ~]# cm-pig-setup -u hdfs1
Requested removal of Pig for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Pig directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.4.3 Using Pig

Pig consists of an executable, `pig`, that can be run after the user loads the corresponding module. Pig runs by default in “MapReduce Mode”, that is, it uses the corresponding HDFS installation to store and deal with the elaborate processing of data. More thorough documentation for Pig can be found at <http://pig.apache.org/docs/r0.15.0/start.html>.

Pig can be used in interactive mode, using the Grunt shell:

```
[root@bright71 ~]# module load hadoop/hdfs1
[root@bright71 ~]# module load pig/hdfs1
[root@bright71 ~]# pig
14/08/26 11:57:41 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
14/08/26 11:57:41 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
14/08/26 11:57:41 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the Ex\
ecType
...
...
grunt>
```

or in batch mode, using a Pig Latin script:

```
[root@bright71 ~]# module load hadoop/hdfs1
[root@bright71 ~]# module load pig/hdfs1
[root@bright71 ~]# pig -v -f /tmp/smoke.pig
```

In both cases, Pig runs in “MapReduce mode”, thus working on the corresponding HDFS instance.

6.5 Sqoop

Apache Sqoop is a tool designed to transfer bulk data between Hadoop and an RDBMS. Sqoop uses MapReduce to import and export data. Bright Cluster Manager supports transfers between Sqoop and MySQL.

RHEL7 and SLES12 use MariaDB, and are not yet supported by the available versions of Sqoop at the time of writing (April 2015). At present, the latest Sqoop stable release is 1.4.5, while the latest Sqoop2 version is 1.99.5. Sqoop2 is incompatible with Sqoop; it is not feature-complete; and it is not yet intended for production use. The Bright Computing utility `cm-sqoop-setup` does not as yet support Sqoop2.

6.5.1 Sqoop Installation With `cm-sqoop-setup`

Bright Cluster Manager provides `cm-sqoop-setup` to carry out Sqoop installation:

Prerequisites For Sqoop Installation, And What Sqoop Installation Does

The following requirements and conditions apply to running the `cm-sqoop-setup` script:

- A Hadoop instance must already be installed.
- Before running the script, the version of the `mysql-connector-java` package should be checked. Sqoop works with releases 5.1.34 or later of this package. If `mysql-connector-java` provides a newer release, then the following must be done to ensure that Sqoop setup works:
 - a suitable 5.1.34 or later release of Connector/J is downloaded from <http://dev.mysql.com/downloads/connector/j/>
 - `cm-sqoop-setup` is run with the `--conn` option in order to specify the connector version to be used.

Example

```
--conn /tmp/mysql-connector-java-5.1.34-bin.jar
```

- The `cm-sqoop-setup` script installs Sqoop only on the active head node. A different node can be specified by using the option `--master`.
- The script creates a dedicated Hadoop Configuration Group for Sqoop.
- Sqoop executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Sqoop configuration files are copied by the script and placed under `/etc/hadoop/`
- The Metastore service is started up by the script.
- When installing Sqoop on a Hadoop instance configured to run on Lustre within Bright Cluster Manager (section 2.4), Sqoop should be deployed on a node that has access to LustreFS (by using the `--master` option if needed). Subsequent operations with Sqoop should be carried out on that node.

An Example Run With `cm-sqoop-setup`

Example

```
[root@bright71 ~]# cm-sqoop-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-op\
enjdk.x86_64/ -t /tmp/sqoop-1.4.5.bin__hadoop-2.0.4-alpha.tar.gz\
--conn /tmp/mysql-connector-java-5.1.34-bin.jar --master node005
Using MySQL Connector/J from /tmp/mysql-connector-java-5.1.34-bin.jar
Sqoop release '1.4.5.bin__hadoop-2.0.4-alpha'
Sqoop service will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.2.0
Sqoop being installed... done.
Creating directories for Sqoop... done.
Creating module file for Sqoop... done.
Creating configuration files for Sqoop... done.
Updating images... done.
Installation successfully completed.
Finished.
```

6.5.2 Sqoop Removal With `cm-sqoop-setup`

`cm-sqoop-setup` should be used to remove the Sqoop instance.

Example

```
[root@bright71 ~]# cm-sqoop-setup -u hdfs1
Requested removal of Sqoop for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Sqoop directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.6 Storm

Apache Storm is a distributed realtime computation system. While Hadoop is focused on batch processing, Storm can process streams of data.

Other parallels between Hadoop and Storm:

- users run “jobs” in Hadoop and “topologies” in Storm
- the master node for Hadoop jobs runs the “JobTracker” or “ResourceManager” daemons to deal with resource management and scheduling, while the master node for Storm runs an analogous daemon called “Nimbus”
- each worker node for Hadoop runs daemons called “TaskTracker” or “NodeManager”, while the worker nodes for Storm runs an analogous daemon called “Supervisor”
- both Hadoop, in the case of NameNode HA, and Storm, leverage “ZooKeeper” for coordination

6.6.1 Storm Installation With `cm-storm-setup`

Bright Cluster Manager provides `cm-storm-setup` to carry out Storm installation.

Prerequisites For Storm Installation, And What Storm Installation Does

The following applies to using `cm-storm-setup`:

- A Hadoop instance, with ZooKeeper, must already be installed.
- The `cm-storm-setup` script only installs Storm on the active head node and on the DataNodes of the chosen Hadoop instance by default. A node other than master can be specified by using the option `--master`, or its alias for this setup script, `--nimbus`.
- The script assigns no roles to nodes.
- Storm executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`.
- Storm configuration files are copied by the script to under `/etc/hadoop/`. This is done both on the active headnode, and on the necessary image(s).
- Validation tests are carried out by the script.

An Example Run With `cm-storm-setup`

Example

```
[root@bright71 ~]# cm-storm-setup -i hdfs1 \
-j /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/ \
-t apache-storm-0.9.4.tar.gz
--nimbus node005
Storm release '0.9.4'
Storm Nimbus and UI services will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.2.0
Storm being installed... done.
```

```

Creating directories for Storm... done.
Creating module file for Storm... done.
Creating configuration files for Storm... done.
Updating images... done.
Initializing worker services for Storm (on DataNodes)... done.
Initializing Nimbus services for Storm... done.
Executing validation test... done.
Installation successfully completed.
Finished.

```

The `cm-storm-setup` installation script submits a validation topology (topology in the Storm sense) called `WordCount`. After a successful installation, users can connect to the Storm UI on the host `<nimbus>`, the Nimbus server, at `http://<nimbus>:10080/`. There they can check the status of `WordCount`, and can kill it.

6.6.2 Storm Removal With `cm-storm-setup`

The `cm-storm-setup` script should also be used to remove the Storm instance.

Example

```

[root@bright71 ~]# cm-storm-setup -u hdfs1
Requested removal of Storm for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Storm directories... done.
Updating images... done.
Removal successfully completed.
Finished.

```

6.6.3 Using Storm

The following example shows how to submit a topology, and then verify that it has been submitted successfully (some lines elided):

```

[root@bright71 ~]# module load storm/hdfs1
[root@bright71 ~]# storm jar /cm/shared/apps/hadoop/Apache/\
  apache-storm-0.9.3/examples/storm-starter/\
  storm-starter-topologies-*.jar \
  storm.starter.WordCountTopology WordCount2

...

470 [main] INFO backtype.storm.StormSubmitter - Jar not uploaded to m\
  aster yet. Submitting jar...
476 [main] INFO backtype.storm.StormSubmitter - Uploading topology ja\
  r /cm/shared/apps/hadoop/Apache/apache-storm-0.9.3/examples/storm-start\
  er/storm-starter-topologies-0.9.3.jar to assigned location: /tmp/storm-\
  hdfs1-local/nimbus/inbox/stormjar-bflabdd0-f31a-41ff-b808-4daad1dfdaa3.\
  jar
Start uploading file '/cm/shared/apps/hadoop/Apache/apache-storm-0.9.3/\
  examples/storm-starter/storm-starter-topologies-0.9.3.jar' to '/tmp/sto\
  rm-hdfs1-local/nimbus/inbox/stormjar-bflabdd0-f31a-41ff-b808-4daad1dfda\
  a3.jar' (3248859 bytes)
[=====] 3248859 / 3248859
File '/cm/shared/apps/hadoop/Apache/apache-storm-0.9.3/examples/storm-s\
  tarter/storm-starter-topologies-0.9.3.jar' uploaded to '/tmp/storm-hdfs\
  1-local/nimbus/inbox/stormjar-bflabdd0-f31a-41ff-b808-4daad1dfdaa3.jar'

```

```
(3248859 bytes)
508 [main] INFO backtype.storm.StormSubmitter - Successfully uploaded\
  topology jar to assigned location: /tmp/storm-hdfs1-local/nimbus/inbox\
/stormjar-bflabdd0-f31a-41ff-b808-4daad1dfdaa3.jar
508 [main] INFO backtype.storm.StormSubmitter - Submitting topology W\
ordCount2 in distributed mode with conf "topology.workers":3,"topology\
.debug":true
687 [main] INFO backtype.storm.StormSubmitter - Finished submitting t\
opology: WordCount2
[root@hadoopdev ~]# storm list
```

...

Topology_name	Status	Num_tasks	Num_workers	Uptime_secs
WordCount2	ACTIVE	28	3	15



Details And Examples Of Hadoop Configuration

This appendix supplements section 3.1.9's introduction to Hadoop/Sqoop configuration under Bright Cluster Manager.

A.1 Hadoop Components Activation And Deactivation Using Roles

Hadoop components such as HDFS or YARN are activated and deactivated using roles. Bright Cluster Manager 7.1 includes 18 possible roles representing possible Hadoop- or Spark-related service, at the time of writing (August 2015).

For example, assigning the HadoopNameNode role to a node makes the node store HDFS meta-data, and be in control of HDFS datanodes that store the actual data in HDFS. Similarly, assigning the DataNode role to a node makes it serve as an HDFS datanode.

A.2 Only The Enabled Hadoop Components And Roles Are Available For Activation From `cmgui` And `cmsh`

Bright Cluster Manager version 7.1 introduced *configuration overlays* (section 3.1.9) to deal with the challenges in configuring Hadoop/Spark components, such as large number of configuration parameters, flexible assignment of services to groups of nodes, and so on. Configuration overlays are the main way of configuring Hadoop- or Spark-related components.

For a given Hadoop cluster instance only a subset of the Hadoop/Spark roles shown in table 3.1.9 is available to the cluster administrator. The actual set of enabled and disabled roles depends on a chosen Hadoop distribution, on the configuration mode (for example HDFS HA *versus* HDFS non-HA) (section 2.3.1) and on the Hadoop-components that are actually selected during the installation procedure.

Example

Hadoop 1.x installation, with HDFS High Availability with manual failover (section 2.3.1), and with the HBase datastore component, enables and disables the roles indicated by the following table:

Enabled	Disabled
Hadoop::NameNode	Hadoop::SecondaryNameNode
Hadoop::DataNode	
Hadoop::Journal	
Hadoop::JobTracker	Hadoop::YARNServer
Hadoop::TaskTracker	Hadoop::YARNClient
Hadoop::HBaseServer	
Hadoop::HBaseClient	
Hadoop::Zookeeper	

Among the disabled roles are two YARN roles. This is because YARN resource manager is a part of Hadoop 2.x distributions.

A.3 Example Of Role Priority Overrides In Configuration Groups With `cmsh`

Configuration groups and role priorities are introduced in section 3.1.9. A summary of some of the important points from there is:

- A role can be directly assigned to a node. The fixed priority for the assignment is then 750.
- A role can be assigned to a node via a category to which the node belongs to. The fixed priority for the assignment is then 250.
- A role can be assigned to a node via a Hadoop configuration group. The default priority for a configuration group is 500, but can be set to any integer from -1 to 1000, except for the values 250 and 750. The values 250 and 750 are reserved for category assignment and for direct role assignment respectively. A priority of -1 disables a Hadoop configuration group.

Thus, due to priority considerations, the configuration of a role assigned via a Hadoop configuration group by default overrides configuration of a role assigned via a category. In turn, a role assigned directly to via node a node assignment overrides the category role and default Hadoop configuration group role.

To illustrate role priorities further, an example Hadoop configuration group, `examplehcg`, is created for an existing Hadoop instance `doop`. For the instance, from within `cmsh`, four Hadoop roles are set for five nodes, and their configuration overlay priority is set to 400 as follows (some text omitted):

Example

```
[bright71]% configurationoverlay
[bright71->configurationoverlay]% add examplehcg
...verlay*[examplehcg*]]% set nodes node001..node005
...verlay*[examplehcg*]]% set priority 400
...verlay*[examplehcg*]]% roles
...verlay*[examplehcg*]->roles]% assign hadoop::datanode
...amplehcg*]->roles*[Hadoop::DataNode*]]% assign hadoop::yarnclient
...amplehcg*]->roles*[Hadoop::YARNClient*]]% assign hadoop::hbaseclient
...amplehcg*]->roles*[Hadoop::HBaseClient*]]% assign hadoop::zookeeper
...amplehcg*]->roles*[Hadoop::ZooKeeper*]]% commit
...
[hadoopdev->configurationoverlay]% list
Name (key) Pri Nodes      Cat Roles
-----
examplehcg 400 node001.. node005      Hadoop::DataNode, Hadoop::YARNClient,
                                     Hadoop::HBaseClient, Hadoop::ZooKeeper
```

Next, the following role assignments:

- Hadoop::HBaseClient to the default category default
- Hadoop::DataNode directly to node002 and node003
- Hadoop::HBaseClient directly to node005

can be carried out in cmsh as follows:

Example

```
[bright71->category]% !#check if nodes in default category first
[bright71->category]% listnodes default
Type                Hostname ...
-----
PhysicalNode        node001 ...
PhysicalNode        node002 ...
PhysicalNode        node003 ...
PhysicalNode        node004 ...
PhysicalNode        node005 ...
PhysicalNode        node006 ...
PhysicalNode        node007 ...
[bright71->category]% use default
[bright71->category[default]]% roles; assign hadoop::hbaseclient; commit
...
[bright71]% device; use node002
[bright71->device[node002]]% roles; assign hadoop::datanode; commit
[bright71]% device; use node003
[bright71->device[node003]]% roles; assign hadoop::datanode; commit
[bright71]% device; use node005
[bright71->device[node005]]% roles; assign hadoop::hbaseclient; commit
```

An overview of the configuration with the `overview` command with the `-verbose` option then shows the sources of the roles, in order of priority (some text omitted and reformatted for clarity):

```
[bright71->hadoop]% overview -v doop
Parameter          Value
-----
Name                doop
...
Hadoop role         Node      Source
-----
Hadoop::DataNode    node001  overlay:examplehcg
Hadoop::DataNode    node002  node002 [750], overlay:examplehcg [400]
Hadoop::DataNode    node003  node003 [750], overlay:examplehcg [400]
Hadoop::DataNode    node004  overlay:examplehcg
Hadoop::DataNode    node005  overlay:examplehcg

Hadoop::HBaseClient node001  overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node002  overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node003  overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node004  overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node005  node005 [750], overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node006  category:default
Hadoop::HBaseClient node007  category:default
```



```

Hadoop::YARNClient      node001  overlay:examplehcg
Hadoop::YARNClient      node002  overlay:examplehcg
Hadoop::YARNClient      node003  overlay:examplehcg
Hadoop::YARNClient      node004  overlay:examplehcg
Hadoop::YARNClient      node005  overlay:examplehcg

Hadoop::ZooKeeper       node001  overlay:examplehcg
Hadoop::ZooKeeper       node002  overlay:examplehcg
Hadoop::ZooKeeper       node003  overlay:examplehcg
Hadoop::ZooKeeper       node004  overlay:examplehcg
Hadoop::ZooKeeper       node005  overlay:examplehcg
...

```

The logic behind the results of the preceding setup is as follows:

- The `Hadoop::HBaseClient`, `Hadoop::YARNClient`, and `Hadoop::ZooKeeper` roles are first assigned at configuration overlay level to `node001..node005`. These roles initially take the altered preset priority of 400 instead of the default of 500, and are active for these nodes, unless overridden by changes further on.
- The `Hadoop::HBaseClient` role is assigned from category level to `node001..node007`. The role on the nodes takes on a priority of 250, and because of that cannot override the configuration overlay role for `node001..node005`. The role is active at this point for `node006` and `node007`.
- Next, the `Hadoop::DataNode` role is assigned directly from node level to `node002` and `node003`. The role on the nodes take on a priority of 750. The value of 400 from the `examplehcg` configuration group assignment is overridden. However, the `Hadoop::DataNode` configuration of `examplehcg` still remains valid for `node001`, `node004`, `node005` so far.
- Then, the `Hadoop::HBaseClient` role is assigned directly from node level to `node005`. The role on the node takes on a priority of 750. The value of 400 for the role from the `examplehcg` configuration is overridden for this node too.

A.4 Cloning Hadoop Configuration Groups In `cmgui` And `cmsh`

Hadoop contains many components, which results in many corresponding Bright Cluster Manager roles. The huge number of configurable parameters for these components results in an unfeasibly large number of settings—more than 220—for configuring Hadoop/Spark.

For ease of use, it is expected that most Hadoop management and configuration operations are carried out with the `cmgui` front end (section 3.1), rather than with the `cmsh` front end (section 3.2). This is because `cmgui` displays Hadoop-related configurations in a more user-friendly manner than `cmsh`.

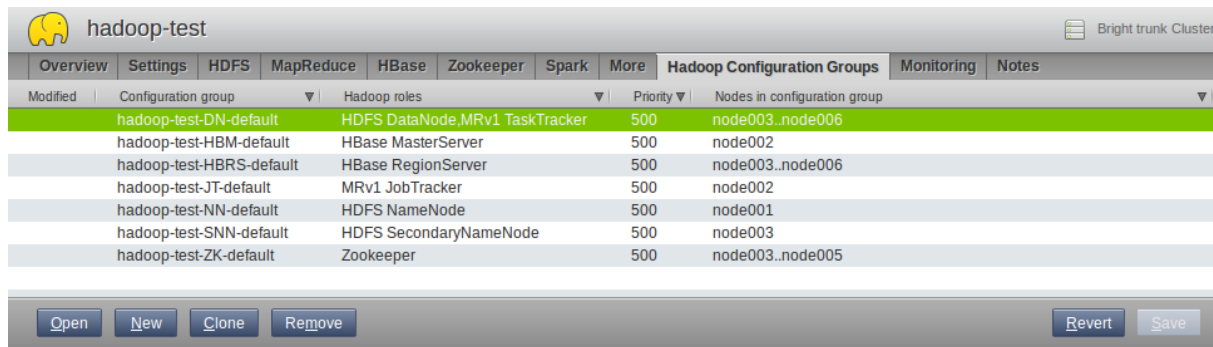
The `cmsh` front end, however, provides full access to the management capabilities of Bright Cluster Manager. In terms of the number of roles and types of roles to be assigned, `cmsh` is more flexible than `cmgui` because:

- it allows a Hadoop configuration group (configuration overlay) to be created with zero roles
- it allows any available role in Bright Cluster Manager to be assigned. These roles can be outside of Hadoop- or Spark-related roles.

The cloning operations of Hadoop using `cmgui` are covered first in this section A.4.1. The same operations using `cmsh` are described afterwards, in section A.4.2.

A.4.1 Cloning Hadoop Configuration Groups In `cmgui`

In the following example, the `cmgui` front end is used to manage the Hadoop cluster instance shown in figure A.1.



Modified	Configuration group	Hadoop roles	Priority	Nodes in configuration group
	hadoop-test-DN-default	HDFS DataNode, MRv1 TaskTracker	500	node003..node006
	hadoop-test-HBM-default	HBase MasterServer	500	node002
	hadoop-test-HBRS-default	HBase RegionServer	500	node003..node006
	hadoop-test-JT-default	MRv1 JobTracker	500	node002
	hadoop-test-NN-default	HDFS NameNode	500	node001
	hadoop-test-SNN-default	HDFS SecondaryNameNode	500	node003
	hadoop-test-ZK-default	Zookeeper	500	node003..node005

Figure A.1: Hadoop Configuration Group tab in cmgui

For this cluster, a situation is imagined where the nodes `node005` and `node006` suddenly experience an extra, non-Hadoop-related, memory-intensive workload, while the remaining nodes `node003` and `node004` are fully dedicated to Hadoop usage. In that case it makes sense to reduce the memory that Hadoop requires for `node005` and `node006`. The MapReduce TaskTracker services on `node005` and `node006` could have their memory parameters reduced, such as the Java heap size, max map tasks number, and so on. At the same time, the configurations of HDFS DataNodes on these two nodes should be left alone. These requirements can be achieved as follows:

- The `hadoop-test-DN-default` configuration group can be cloned with the `Clone` button in the Hadoop Configurations Groups tab. An editing window 'Clone Hadoop Configuration Group' pops up with a new, cloned-from-`hadoop-test-DN-default` group. It gets a default suffix of `'-cloned'`.
- The nodes in the cloned configuration group are set to `node005` and `node006`.
- The HDFS DataNode role is removed from the configuration group. In this particular example, the DataNode role might also be left as is.
- The priority of the group should be checked to see that it is set to higher than that of `hadoop-test-DN-default`. By default, a cloned group is set to the priority of the parent group, plus 10.
- Lower values are set for relevant TaskTracker configuration parameters. In this case, the Java heap size value within TaskTracker can be reduced. Figures A.2 and A.3 show the original state of the configuration group before clicking on the `Clone` button, and the cloned state after reducing the memory-related parameters.

Edit Hadoop Configuration Group










Configuration Group:

Priority: ⓘ

Nodes in Configuration Group: [Add/remove nodes](#)

Configure HDFS DataNode | **Configure MRv1 TaskTracker**

Settings applied to:

TaskTracker Web UI port: <input type="text" value="50060"/>	TaskTracker Java heap size: <input type="text" value="2048"/> MB
HTTP threads count: <input type="text" value="80"/>	JVM settings:   
Maximum map tasks: <input type="text" value="8"/>	Number of tasks per JVM: <input type="text" value="10"/>
Maximum reduce tasks: <input type="text" value="2"/>	Map JVM options: <input type="text" value="-Xmx200M"/>
Map speculative execution: <input checked="" type="checkbox"/>	Reduce JVM options: <input type="text" value="-Xmx64M"/>
Reduce speculative execution: <input checked="" type="checkbox"/>	MapReduce advanced:   
	Map output compression:   

[Add role](#) [Remove role](#) [Cancel](#) [Ok](#)

Figure A.2: Hadoop Configuration Group Prior To Cloning

Figure A.3: Example Of Cloned Hadoop Configuration Group

- The cloned Hadoop configuration group and all the changes to it should be saved, by clicking on the OK button of the edit window, then on the Save button of the parent Hadoop Configuration Groups window.

As a result of these changes, Bright Cluster Manager restarts MapReduce TaskTracker service with the configuration settings that are defined in `hadoop-test-DN-default-cloned`. MapReduce in figure A.4 compared with before now displays one more Hadoop configuration group—the cloned group.

Modified	Configuration group	Hadoop roles	Priority	Nodes in configuration group
	hadoop-test-DN-default	HDFS DataNode, MRv1 TaskTracker	500	node003..node006
	hadoop-test-DN-default-cloned	MRv1 TaskTracker	510	node005, node006
	hadoop-test-JT-default	MRv1 JobTracker	500	node002

Figure A.4: Hadoop Configuration Groups for MapReduce after example configuration

There is no imposed limit on the number of Hadoop configuration groups that can be used for a given Hadoop cluster instance. For large numbers, it can be difficult to see which configurations from which groups are actually applied to nodes or sets of nodes.

To help with that, the Hadoop Configuration Groups display window (figure A.1) displays updated information on the roles and configuration groups that are applied to the nodes. For example, the MapReduce TaskTracker defined in `hadoop-test-DN-default-cloned` has the Settings applied to field in figure A.3, where `node005` and `node006` are listed. These nodes are displayed in the Hadoop Configuration Groups display window right away.

Also at the same time, the nodes in `hadoop-test-DN-default` have changed. The role settings for its TaskTracker nodes are now applied only to `node003` and `node004`. These changes are also displayed in the Hadoop Configuration Groups display window right away.

A.4.2 Cloning Hadoop Configuration Groups In `cmsh`

The following session discusses the cloning operation that is described in section A.4.1 once more. Only this time, it is done using `cmsh` rather than `cmgui` (some text omitted for clarity):

Example

```
[hadoopdev]% configurationoverlay
[hadoopdev->configurationoverlay]% list
Name (key)                Pri Nodes                Roles
-----
hadoop-test-DN-default    500 node003..node006    Hadoop::DataNode, Hadoop::
hadoop-test-HBM-default   500 node002             Hadoop::HBaseServer
hadoop-test-HBRS-default  500 node003..node006    Hadoop::HBaseClient
hadoop-test-JT-default    500 node002             Hadoop::JobTracker
hadoop-test-NN-default    500 node001             Hadoop::NameNode
hadoop-test-SNN-default   500 node003             Hadoop::SecondaryNameNode
hadoop-test-ZK-default    500 node003..node005    Hadoop::ZooKeeper
[...overlay]% clone hadoop-test-dn-default hadoop-test-dn-default-cloned
[...overlay*[hadoop-test-dn-default-cloned*]]% set priority 510
[...hadoop-test-dn-default-cloned*]]% roles; unassign hadoop::datanode
[...overlay*[hadoop-test-dn-default-cloned*]]% commit
[...overlay[hadoop-test-dn-default-cloned]->roles]% list
Name (key)
-----
Hadoop::TaskTracker
[...fault-cloned]->roles]% use hadoop::tasktracker; configurations; list
HDFS
-----
hadoop-test
[->roles[Hadoop::TaskTracker]->configurations]% use hadoop-test; show
Parameter                Value
-----
File merging number       32
HDFS                      hadoop-test
HTTP port                 50060
...
Map speculative execution  yes
Maximum map tasks         8
...
TaskTracker heap size     2048
Type                      HadoopTaskTrackerHDFSConfiguration
[...ker]->configurations[hadoop-test]]% set tasktrackerheapsizesize 1024
[...ker*]->configurations*[hadoop-test*]]% set maximummaptasks 4; commit
```

The result of this is the Hadoop configuration group `hadoop-test-DN-default-cloned`, which is seen in the `cmgui` equivalent in figure A.3.

A.5 Considerations And Best Practices When Creating Or Cloning Hadoop Configurations

The `cmgui` front end is the recommended way to carry out Hadoop configuration operations, and for installing, configuring and managing the Hadoop cluster instances. The following are considerations and best practices:

- Naming conventions: It is recommended to start a name for a new or cloned Hadoop configuration group with the name of the Hadoop cluster instance. This is automatically done for the default Hadoop configuration groups created during Hadoop installation.
- A Hadoop configuration group can include zero nodes, but it has to have at least one role assigned. An exception to this is that the `cmsh` front end allows a user to create a Hadoop configuration group with no roles assigned, but such a group cannot be connected to any Hadoop instance, and such groups are therefore not displayed in `cmgui`.
- If a Hadoop configuration group has no roles assigned to it, then it can be seen only via the `configurationoverlay` mode of `cmsh`.
- Hadoop configuration groups that are not in use should be disabled using `-1` as a priority value. If the configuration group is disabled, then the configurations in all roles, for all nodes in this group, will no longer be used. Instead the next highest priority configuration will be used.
- A history of configuration changes can be tracked using the cloning functionality. For example, the parent group can be the configuration group that always has the current configuration. A list of groups with earlier configurations can then be kept, where each is derived from a parent by cloning it, and setting its priority to `-1`, and also including the timestamp (for example, `YYYYMMDD`, for easy sorting) in its name:

Example

```
hadoop-config[500]
hadoop-config-cloned-20150514[-1]
hadoop-config-cloned-20141104[-1]
hadoop-config-cloned-20131008[-1]
...
```

- Hadoop/Spark roles that correspond to key Hadoop services (the asterisked services in table 3.1.9) are deliberately not provided by `cmgui` or `cmsh` as options for addition or removal when editing or creating a Hadoop configuration group. This is done because of the risk of data loss if the key services are misconfigured.

A workaround for this restriction is that a configuration group with a key Hadoop role can be cloned. The cloned group, which includes the service, can then be built upon further.

- A Hadoop configuration group is associated with a Hadoop instance if it has at least one role with a configuration linked to that Hadoop instance. For example, the following commands investigate the `hadoop-test-dn-default` group. The Hadoop cluster instances for which the MapReduce TaskTracker role configurations are defined are shown:

```
[hadoopdev]% configurationoverlay; use hadoop-test-dn-default; roles
[hadoopdev->configurationoverlay[hadoop-test-DN-default]->roles]%
```

```
[...-DN-default]->roles}% use hadoop::tasktracker; configurations; list  
HDFS  
-----  
hadoop-test
```

- Assignment of Hadoop or Spark-related roles directly to nodes or to node categories should be avoided. Hadoop configuration groups (configuration overlays) should be used instead.

If the setup can benefit from the direct assignment of roles to nodes or to categories, then the administrator should be aware of priorities and their outcome for role assignments that overlay each other (example in section A.3).