Bright Cluster Manager 8.1

Big Data Deployment Manual

Revision: 8511be6

Date: Tue May 20 2025

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. PBS Professional, PBS Pro, and Green Provisioning are trademarks of Altair Engineering, 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

Preface

Welcome to the *Big Data Deployment Manual* for Bright Cluster Manager 8.1.

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 8.1 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 *Big Data Deployment Manual*—this manual—describes how to deploy Big Data with Bright Cluster Manager.
- The *Machine Learning Manual* describes how to install and configure machine learning capabilities 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`.

There is also a feedback form available via Bright View, via the Account icon,  , following the clickpath:

`Account`→`Help`→`Feedback`

0.3 Getting Administrator-Level Support

If the reseller from whom Bright Cluster Manager was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website `https://support.brightcomputing.com`. This allows the administrator to submit a support request via a web form, and opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue. The followup communication for this ticket typically goes via standard e-mail. Section 13.2 of the *Administrator Manual* has more details on working with support.

0.4 Getting Professional Services

Bright Computing normally differentiates between professional services (customer asks Bright Computing to do something or asks Bright Computing to provide some service) and support (customer has a question or problem that requires an answer or resolution). Professional services can be provided after consulting with the reseller, or the Bright account manager.

1

Introduction

1.1 What Are Hadoop And Big Data About?

Hadoop is a popular 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 8.1 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.

The ISO image for Bright Cluster Manager, available at `http://www.brightcomputing.com/Download`, can include Hadoop for all 3 implementations. During installation from the ISO, the administrator can choose which implementation to install (section 3.3.15 of the *Installation Manual*).

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

SLES 12 Big Data JDK/JVMs: Only Oracle JDK/JVM Supported

At the time of writing (December 2016), support on SLES 12 Big Data deployments is provided by Bright Cluster Manager only for the JVM/JDK that Oracle provides.

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	`hadoop-1.2.1/docs/index.html`
Apache 2.7.4	`hadoop-2.7.4/share/doc/hadoop/index.html`
Cloudera CDH 5.13.0	`hadoop-2.6.0-cdh5.13.0/share/doc/index.html`
Hortonworks HDP	*Online documentation is available at* `http://docs.hortonworks.com/`

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 1.4.1-1.4.12.

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	Alluxio, Drill, Flink, Ignite, Sqoop, Spark, Storm
none	pick up from	Hortonworks, Cloudera

Thus, x as a superscript means the software must be picked up from the corresponding Apache project website. The website URL associated with the project is given in the following table:

Project	URL
Alluxio	http://alluxio.org
Drill	http://drill.apache.org
Flink	http://flink.apache.org
Ignite	http://ignite.apache.org
Sqoop	http://sqoop.apache.org
Spark	http://spark.apache.org
Apache Storm	http://storm.apache.org

Similarly, no superscript means that the software is available from the corresponding implementation provider website, as follows:

Project	URL
Hortonworks	http://hortonworks.com
Cloudera	http://www.cloudera.com or directly from these URLs, depending on version: http://archive.cloudera.com/cdh4/cdh/4/ http://archive.cloudera.com/cdh5/cdh/5/

In addition, all the tar.gz, tgz, and zip file source packages listed in the version matrix can be picked up from the Bright Computing website at http://support.brightcomputing.com/bigdata/

Some other Big Data tools, like Apache Giraph, should be built from source, depending on the chosen Hadoop version and distribution. More details are given in the sections for these tools in Chapter 7.

1.4.1 Apache Hadoop 1.2.1

- http://support.brightcomputing.com/bigdata/hadoop-1.2.1.tar.gz[a]
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.11.tar.gz[a]
- http://support.brightcomputing.com/bigdata/hbase-0.98.24-hadoop1-bin.tar.gz[a]
- http://support.brightcomputing.com/bigdata/apache-hive-1.2.1-bin.tar.gz[x]
- http://support.brightcomputing.com/bigdata/pig-0.16.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/spark-1.6.2-bin-hadoop1-scala2.11.tgz[x]

- http://support.brightcomputing.com/bigdata/accumulo-1.5.4-bin.tar.gz[x]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6.bin__hadoop-1.0.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]
- Drill not supported
- Flink not supported
- Ignite not supported
- Alluxio not supported

1.4.2 Hortonworks HDP 1.3.11

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

- http://support.brightcomputing.com/bigdata/hadoop-1.2.0.1.3.11.0-26.tar.gz[d]
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.5.1.3.11.0-26.tar.gz[d]
- http://support.brightcomputing.com/bigdata/hbase-0.94.6.1.3.11.0-26-security.tar.gz[d]
- http://support.brightcomputing.com/bigdata/hive-0.11.0.1.3.11.0-26.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.11.1.1.3.11.0-26.tar.gz
- http://support.brightcomputing.com/bigdata/spark-1.5.1-bin-hadoop1.tgz[x]
- http://support.brightcomputing.com/bigdata/accumulo-1.5.4-bin.tar.gz[x]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.3.1.3.11.0-26.bin__hadoop-1.2.0.1.3.11.0-26.tar.gz
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]
- Drill not supported
- Flink not supported
- Ignite not supported
- Alluxio not supported

1.4.3 Apache Hadoop 2.7.4

- http://support.brightcomputing.com/bigdata/hadoop-2.7.4.tar.gz[a]
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.11.tar.gz[a]
- http://support.brightcomputing.com/bigdata/hbase-1.3.1-bin.tar.gz[a]
- http://support.brightcomputing.com/bigdata/apache-hive-2.3.2-bin.tar.gz[b]
- http://support.brightcomputing.com/bigdata/pig-0.17.0.tar.gz[b]
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.7.tgz[b]
- http://support.brightcomputing.com/bigdata/accumulo-1.8.1-bin.tar.gz[b]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.99.7-bin-hadoop200.tar.gz[x]
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]
- http://support.brightcomputing.com/bigdata/apache-drill-1.11.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop27-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.7-bin.tar.gz[x]

1.4.4 Apache Hadoop 2.9.0

- http://support.brightcomputing.com/bigdata/hadoop-2.9.0.tar.gz
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.11.tar.gz[a]
- http://support.brightcomputing.com/bigdata/hbase-1.3.1-bin.tar.gz[a]
- http://support.brightcomputing.com/bigdata/apache-hive-2.3.2-bin.tar.gz[b]
- http://support.brightcomputing.com/bigdata/pig-0.17.0.tar.gz[b]
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.7.tgz[b]
- http://support.brightcomputing.com/bigdata/accumulo-1.8.1-bin.tar.gz[b]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.99.7-bin-hadoop200.tar.gz[x]
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]

- Drill not supported
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop27-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.8-bin.tar.gz[x]

1.4.5 Cloudera CDH 4.7.1

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

- http://support.brightcomputing.com/bigdata/hadoop-2.0.0-cdh4.7.1.tar.gz[c]
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.5-cdh4.7.1.tar.gz[c]
- http://support.brightcomputing.com/bigdata/hbase-0.94.15-cdh4.7.1.tar.gz[c]
- http://support.brightcomputing.com/bigdata/hive-0.10.0-cdh4.7.1.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.11.0-cdh4.7.1.tar.gz
- http://support.brightcomputing.com/bigdata/spark-1.6.2-bin-cdh4.tgz[x]
- http://support.brightcomputing.com/bigdata/accumulo-1.6.2-bin.tar.gz[x]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.3-cdh4.7.1.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop2-1.99.2-cdh4.7.1.tar.gz
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]
- Drill not supported
- Flink not supported
- Ignite not supported
- Alluxio not supported

1.4.6 Cloudera CDH 5.10.2

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

- http://support.brightcomputing.com/bigdata/hadoop-2.6.0-cdh5.10.2.tar.gz
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.5-cdh5.10.2.tar.gz
- http://support.brightcomputing.com/bigdata/hbase-1.2.0-cdh5.10.2.tar.gz
- http://support.brightcomputing.com/bigdata/hive-1.1.0-cdh5.10.2.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.12.0-cdh5.10.2.tar.gz
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.6.tgz[x]

- http://support.brightcomputing.com/bigdata/accumulo-1.8.1-bin.tar.gz[b]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6-cdh5.10.2.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop2-1.99.5-cdh5.10.2.tar.gz
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]
- http://support.brightcomputing.com/bigdata/apache-drill-1.11.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop26-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.6-bin.tar.gz[x]

1.4.7 Cloudera CDH 5.11.2

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

- http://support.brightcomputing.com/bigdata/hadoop-2.6.0-cdh5.11.2.tar.gz
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.5-cdh5.11.2.tar.gz
- http://support.brightcomputing.com/bigdata/hbase-1.2.0-cdh5.11.2.tar.gz
- http://support.brightcomputing.com/bigdata/hive-1.1.0-cdh5.11.2.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.12.0-cdh5.11.2.tar.gz
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.6.tgz[x]
- http://support.brightcomputing.com/bigdata/accumulo-1.8.1-bin.tar.gz[b]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6-cdh5.11.2.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop2-1.99.5-cdh5.11.2.tar.gz
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]
- http://support.brightcomputing.com/bigdata/apache-drill-1.11.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop26-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.6-bin.tar.gz[x]

1.4.8 Cloudera CDH 5.12.1

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

- http://support.brightcomputing.com/bigdata/hadoop-2.6.0-cdh5.12.1.tar.gz
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.5-cdh5.12.1.tar.gz
- http://support.brightcomputing.com/bigdata/hbase-1.2.0-cdh5.12.1.tar.gz
- http://support.brightcomputing.com/bigdata/hive-1.1.0-cdh5.12.1.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.12.0-cdh5.12.1.tar.gz
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.6.tgz[x]
- http://support.brightcomputing.com/bigdata/accumulo-1.8.1-bin.tar.gz[b]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6-cdh5.12.1.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop2-1.99.5-cdh5.12.1.tar.gz
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]
- http://support.brightcomputing.com/bigdata/apache-drill-1.11.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop26-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.6-bin.tar.gz[x]

1.4.9 Cloudera CDH 5.13.1

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

- http://support.brightcomputing.com/bigdata/hadoop-2.6.0-cdh5.13.1.tar.gz[c]
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.5-cdh5.13.1.tar.gz[c]
- http://support.brightcomputing.com/bigdata/hbase-1.2.0-cdh5.13.1.tar.gz[c]
- http://support.brightcomputing.com/bigdata/hive-1.1.0-cdh5.13.1.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.12.0-cdh5.13.1.tar.gz
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.6.tgz[x]
- http://support.brightcomputing.com/bigdata/accumulo-1.8.1-bin.tar.gz[b]
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.1.tar.gz[b]
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6-cdh5.13.1.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop2-1.99.5-cdh5.13.1.tar.gz
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]

- http://support.brightcomputing.com/bigdata/apache-drill-1.11.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop26-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.6-bin.tar.gz[x]

1.4.10 Hortonworks HDP 2.4.3

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

- http://support.brightcomputing.com/bigdata/hadoop-2.7.1.2.4.3.0-227.tar.gz
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.6.2.4.3.0-227.tar.gz
- http://support.brightcomputing.com/bigdata/hbase-1.1.2.2.4.3.0-227.tar.gz
- http://support.brightcomputing.com/bigdata/apache-hive-1.2.1000.2.4.3.0-227-bin.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.15.0.2.4.3.0-227.tar.gz
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.7.tgz[b]
- http://support.brightcomputing.com/bigdata/accumulo-1.7.0.2.4.3.0-227-bin.tar.gz
- http://support.brightcomputing.com/bigdata/apache-storm-0.10.0.2.4.3.0-227.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6.2.4.3.0-227.bin__hadoop-2.7.1.2.4.3.0-227.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop-1.99.7-bin-hadoop200.tar.gz[x]
- http://support.brightcomputing.com/bigdata/kafka_2.11-1.0.0.tgz[b]
- http://support.brightcomputing.com/bigdata/apache-drill-1.11.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop27-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.7-bin.tar.gz[x]

1.4.11 Hortonworks HDP 2.5.6

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

- http://support.brightcomputing.com/bigdata/hadoop-2.7.3.2.5.6.0-40.tar.gz
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.6.2.5.6.0-40.tar.gz
- http://support.brightcomputing.com/bigdata/hbase-1.1.2.2.5.6.0-40.tar.gz
- http://support.brightcomputing.com/bigdata/apache-hive-2.1.0.2.5.6.0-40-bin.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.16.0.2.5.6.0-40.tar.gz
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.7.tgz[b]
- http://support.brightcomputing.com/bigdata/accumulo-1.7.0.2.5.6.0-40-bin.tar.gz
- http://support.brightcomputing.com/bigdata/apache-storm-1.0.1.2.5.6.0-40.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6.2.5.6.0-40.bin__hadoop-2.7.3.2.5.6.0-40.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop-1.99.7-bin-hadoop200.tar.gz[x]
- http://support.brightcomputing.com/bigdata/kafka_2.10-0.10.0.2.5.6.0-40.tgz
- http://support.brightcomputing.com/bigdata/apache-drill-1.11.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop27-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.7-bin.tar.gz[x]

1.4.12 Hortonworks HDP 2.6.3

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

- http://support.brightcomputing.com/bigdata/hadoop-2.7.3.2.6.3.0-235.tar.gz[d]
- http://support.brightcomputing.com/bigdata/zookeeper-3.4.6.2.6.3.0-235.tar.gz[d]
- http://support.brightcomputing.com/bigdata/hbase-1.1.2.2.6.3.0-235.tar.gz[d]
- http://support.brightcomputing.com/bigdata/apache-hive-2.1.0.2.6.3.0-235-bin.tar.gz
- http://support.brightcomputing.com/bigdata/pig-0.16.0.2.6.3.0-235.tar.gz
- http://support.brightcomputing.com/bigdata/spark-2.2.1-bin-hadoop2.7.tgz[b]

- http://support.brightcomputing.com/bigdata/accumulo-1.7.0.2.6.3.0-235-bin.tar.gz
- http://support.brightcomputing.com/bigdata/apache-storm-1.1.0.2.6.3.0-235.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop-1.4.6.2.6.3.0-235.bin__hadoop-2.7.3.2.6.3.0-235.tar.gz
- http://support.brightcomputing.com/bigdata/sqoop-1.99.7-bin-hadoop200.tar.gz[x]
- http://support.brightcomputing.com/bigdata/kafka_2.10-0.10.1.2.6.3.0-235.tgz
- http://support.brightcomputing.com/bigdata/apache-drill-1.11.0.tar.gz[x]
- http://support.brightcomputing.com/bigdata/flink-1.3.2-bin-hadoop27-scala_2.11.tgz[x]
- http://support.brightcomputing.com/bigdata/apache-ignite-hadoop-2.3.0-bin.zip[x]
- http://support.brightcomputing.com/bigdata/alluxio-1.6.1-hadoop-2.7-bin.tar.gz[x]

2

Installing Hadoop

If a big data distribution has been selected, then the user can install Bright Cluster Manager with it. There are currently 12 supported Hadoop/Spark cluster versions, along with a chosen mode of operation in Bright Cluster Manager 8.1. Besides this, a number of Hadoop components can be installed/removed depending on the user's needs.

A Hadoop instance can be installed via

- the command-line (section 2.1)
- an Ncurses GUI (section 2.2)
- or via a Hadoop installation wizard in Bright View (section 2.3)

The first two options are carried out with the `cm-hadoop-setup` script, which is run from a head node. The script is part of the `cluster-tools` package, and uses tarballs from the Apache Hadoop project. The third option runs as its own application.

Finally, if the Bright Cluster Manager installation ISO provided by Bright Computing is the Bright Cluster Manager With Hadoop installation ISO, then this 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@bright81 ~]# cm-hadoop-setup -h

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

  OPTIONS:
  -c <filename>       --  Hadoop config file to use
  -u <name>           --  uninstall Hadoop instance <name>
  --upgrade <name>    --  upgrade Hadoop instance <name>
  -t <file>           --  Hadoop tarball
  -h                  --  show usage

  EXAMPLES:
  cm-hadoop-setup -c /tmp/config.xml
  cm-hadoop-setup -u foo
  cm-hadoop-setup --upgrade foo -t /cm/local/apps/hadoop/hadoop-2.7.2.tar.gz
  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/:

  hadoop1conf.xml         (for Hadoop 1.x)
  hadoop2conf.xml         (for Hadoop 2.x)
  hadoop2fedconf.xml      (for Hadoop 2.x with NameNode federation)
  hadoop2haconf.xml       (for Hadoop 2.x with High Availability)
  hadoop2lustreconf.xml   (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 take the form 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 1.4, or from the Bright Computing website at `http://support.brightcomputing.com/bigdata/`. Care must be taken in ensuring version compatibility. The version compatibility is as outlined in section 1.4.

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 `/cm/local/apps/hadoop/` directory if the default definitions in the default XML files are used:

Example

```
[root@bright81 ~]# cd /cm/local/apps/cluster-tools/hadoop/conf
[root@bright81 conf]# grep 'archive>' hadoop1conf.xml | grep -o /.*.gz
/cm/local/apps/hadoop/hadoop-1.2.1.tar.gz
/cm/local/apps/hadoop/zookeeper-3.4.10.tar.gz
/cm/local/apps/hadoop/hbase-0.98.24-hadoop1-bin.tar.gz
```

The administrator may wish to place the tarball components in another 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 data storage can be specified within the tag pair:

- `<dataroot></dataroot>`

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>

A configuration file, hdfs-site.xml, is generated during deployment from the preceding specification. Hadoop then has the following dfs.*.name.dir properties added to it via that generated file:

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

The example configuration file at /cm/local/apps/cluster-tools/hadoop/conf/hadoop2conf.xml can be used as a template. It can be copied to a file with an arbitrary name, such as hadoop274conf.xml, which might be used for a Hadoop2 installation using Hadoop 2.7.4. The new XML file can then be modified to suit requirements and software that is on the system. The administrator should check that the entries for the tags that indicate file paths are correct. After suitable modification, an install run then displays output like the following:

Example

```
[root@bright81 ~]# cm-hadoop-setup -c hadoop274conf.xml
Reading config from file '/cm/local/apps/cluster-tools/hadoop/conf/hadoop274conf.xml'... done.
Executing pre-deployment checks...
Executing pre-deployment checks... done.
Hadoop flavor 'Apache', release '2.7.4'
Will now install Hadoop in /cm/shared/apps/hadoop/Apache/2.7.4 and configure instance 'Myhadoop'
Hadoop distro being installed...
Hadoop distro being installed... done.
Key Management Service supported.
ZooKeeper being installed... done.
HBase being installed...
Creating module file... done.
Configuring Hadoop instance on local filesystem and images... done.
Updating images...
Updating images... done.
Creating Hadoop instance in CMDaemon...
Creating Hadoop instance in CMDaemon... done.
Setting up ZooKeeper hosts...
Setting up ZooKeeper hosts... done.
Formatting HDFS...
Formatting HDFS... done.
Setting up NameNode...
Setting up NameNode... done.
```

```
Setting up SecondaryNameNode...
Setting up SecondaryNameNode... done.
Setting up HDFS...
Setting up HDFS... done.
Setting up DataNodes...
Setting up DataNodes... done.
Setting up HBase hosts...
Setting up HBase hosts... done.
Setting up YARN hosts...
Setting up YARN hosts... done.
Waiting for Hadoop to be ready for validation test...
Waiting for Hadoop to be ready for validation test... done.
Validating Hadoop setup...
Validating Hadoop setup... done.
Validating ZooKeeper installation...
Validating ZooKeeper installation... done.
Validating HBase installation...
Validating HBase installation... done.
Installation successfully completed.
Finished.
```

The Hadoop instance should now be running. The name defined for it in the XML file (in this case Myhadoop) shows up within Bright View via the clickpath `Big Data→Big Data Instances` (figure 2.1):

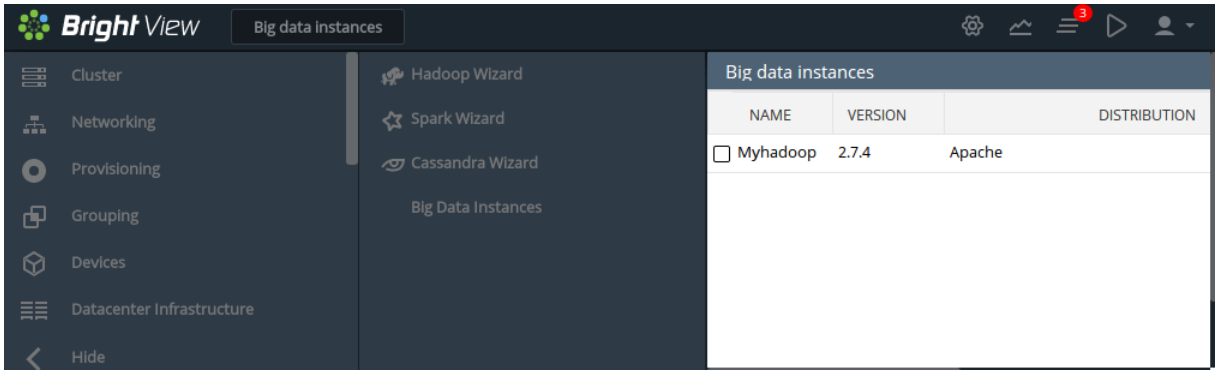

Figure 2.1: A Hadoop Instance In Bright View

Double-clicking on the instance, or clicking on the associated `Edit` button opens up a panel which displays configuration settings for the instance.

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

Example

```
[root@bright81 ~] cmsh
[bright81]% hadoop
[bright81->hadoop]% list
Name (key) Hadoop version Hadoop distribution Configuration directory
---------- -------------- ------------------- ----------------------
Myhadoop   2.7.4          Apache              /etc/hadoop/Myhadoop
```

The instance can be uninstalled as follows:

Example

```
[root@bright81 ~]# cm-hadoop-setup -u Myhadoop
Requested removal of Hadoop instance 'Myhadoop'.
Stopping all Hadoop services for instance 'Myhadoop'...
Stopping all Hadoop services for instance 'Myhadoop'... done.
Removing instance 'Myhadoop' from CMDaemon...
Removing instance 'Myhadoop' from CMDaemon... done.
Waiting a few seconds before removing files from local fs...
Waiting a few seconds before removing files from local fs... done.
Removing Hadoop instance 'Myhadoop'

Removing:
                /etc/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
                /var/lib/hadoop/Apache274/
                /usr/lib/systemd/system/hadoop-Myhadoop-*
Updating images...
Updating images... done.
Instance 'Myhadoop' removed from images.
Removing log files...
Removing log/data files... done.
Module file(s) deleted.
Removal successfully completed.
```

2.1.3 Deploy Hadoop On A Specific Network

By default Hadoop is configured to use the management network `internalnet`, as defined in the base partition. This means that CMDaemon writes out Hadoop configuration using the IP addresses or fully qualified domain names of that network.

This feature is not available for Hadoop 1 and for Hadoop 2 releases older than 2.6.*x*.

When multiple internal networks are available, users would want to specify a different internal network for Hadoop traffic. It is possible to do that by using the tag `<network>`:

Example

```
<hadoopConfig>
  ...
  <instance>
    <name>Apache274</name>
    <network>hadoopnet</network>
    <dataroot>/var/lib/hadoop/Apache274/</dataroot>
    ...
</hadoopConfig>
```

The Hadoop instance is configured by this to use the selected network. Hadoop has several services in general for RPC and for HTTP, and each service offers different endpoints. For web interfaces the value `0.0.0.0` is used, so that the web UI is available on all networks. Master services, such as the NameNode and ResourceManager, also bind their RPC endpoint to all interfaces. The main purpose of

selecting a specific network is to make sure that DataNodes and NodeManagers route their traffic on the correct interface. CMDaemon writes out the correct IP addresses for the following properties in the corresponding Hadoop configuration files:

- `dfs.datanode.address`
- `dfs.datanode.ipc.address`
- `yarn.nodemanager.address`
- `yarn.nodemanager.localizer.address`

Once the Hadoop instance has been installed, it is not possible to change the selected network, as indicated in `cmsh`:

```
[bright81->bigdata]% show hadoop1 | grep Network
Network                                          hadoopnet
```

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).

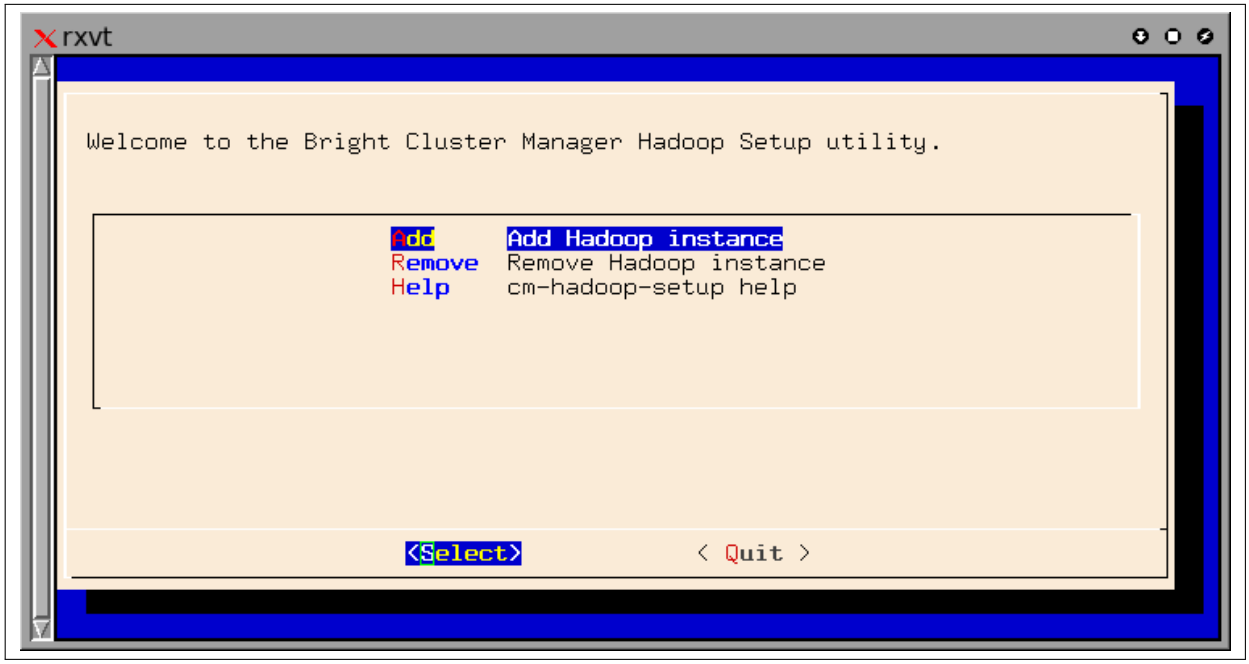

Figure 2.2: The `cm-hadoop-setup` Welcome Screen

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.

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.

2.3 Installation And Removal Of Hadoop In Bright View

A Hadoop instance can be installed or removed using the Bright View wizard. The wizard is launched from via the clickpath `Big Data→Hadoop Wizard`.

Packages that provide the Hadoop or Spark instance should be installed separately, before the wizard is used. How to pick up the correct packages is covered in section 1.4.

Typically, the administrator uses YUM to install a package from the Bright Computing repository, with the default paths already preconfigured. The packages are:

- `cm-apache-hadoop.noarch`
- `cm-apache-hadoop-extras.noarch`
- `cm-hortonworks-hadoop.noarch`
- `cm-cloudera-hadoop.noarch`

An alternative is that the administrator can download a big data distribution of Bright Cluster Manager, and when the wizard is run, the paths can be set.

The wizard (figure 2.3) starts off with a page that allows the instance name and root points to be set. The path to the Java home can be modified if needed by disabling the `Use default Java home` switch. Notes are included to help make choices.

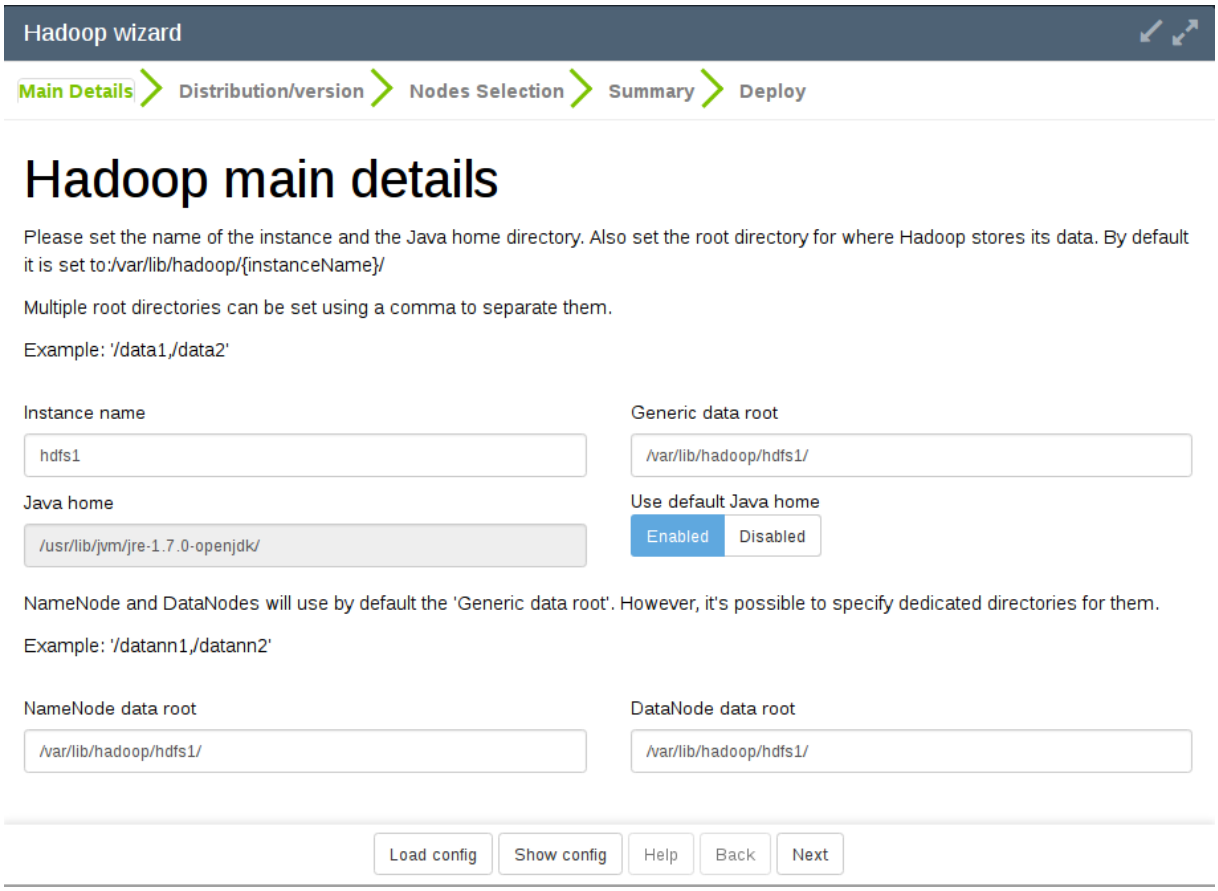

Figure 2.3: Installing Hadoop With Bright View: Main Details Page

The next screen (figure 2.4) allows a Hadoop version to be chosen.

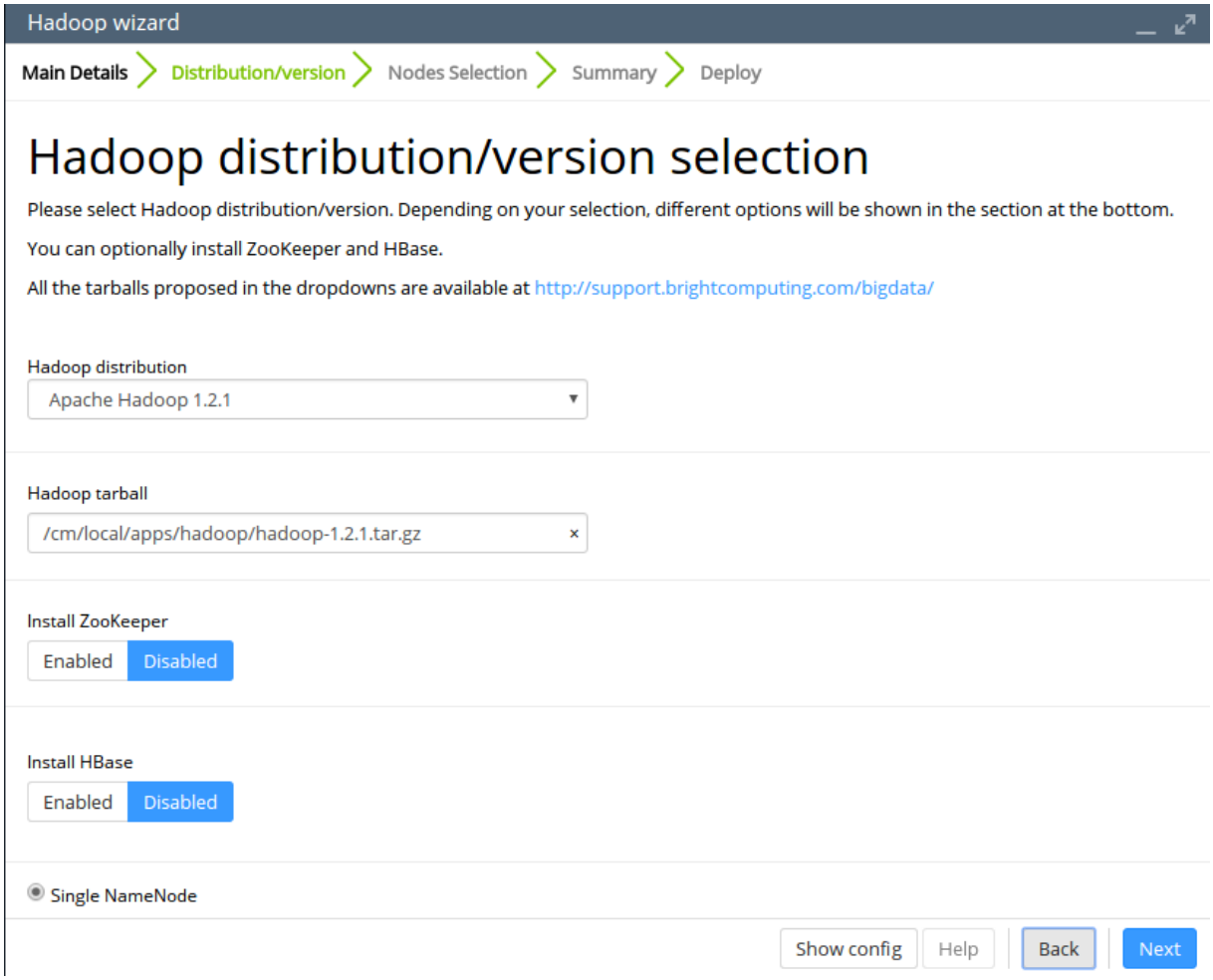

Figure 2.4: Installing Hadoop With Bright View: Version, Tarball, Related Extras Page

If the administrator would like to install ZooKeeper or HBase, then their installation can be enabled in this screen and their path can be set.

If HBase is used, then ZooKeeper must also be running. Options for ZooKeeper and HBase become available from Bright View (section 3.1.5 and 3.1.6) when it is run later.

Depending on the selected Hadoop distribution, and whether it is based on Hadoop 1.*x* (2 possibilities) or Hadoop 2.*x* (10 possibilities), the administrator sees either:

- only one option, `Single NameNode` (figure 2.4)
- several installation options for Hadoop 2.*x*-based distros (figure 2.5)

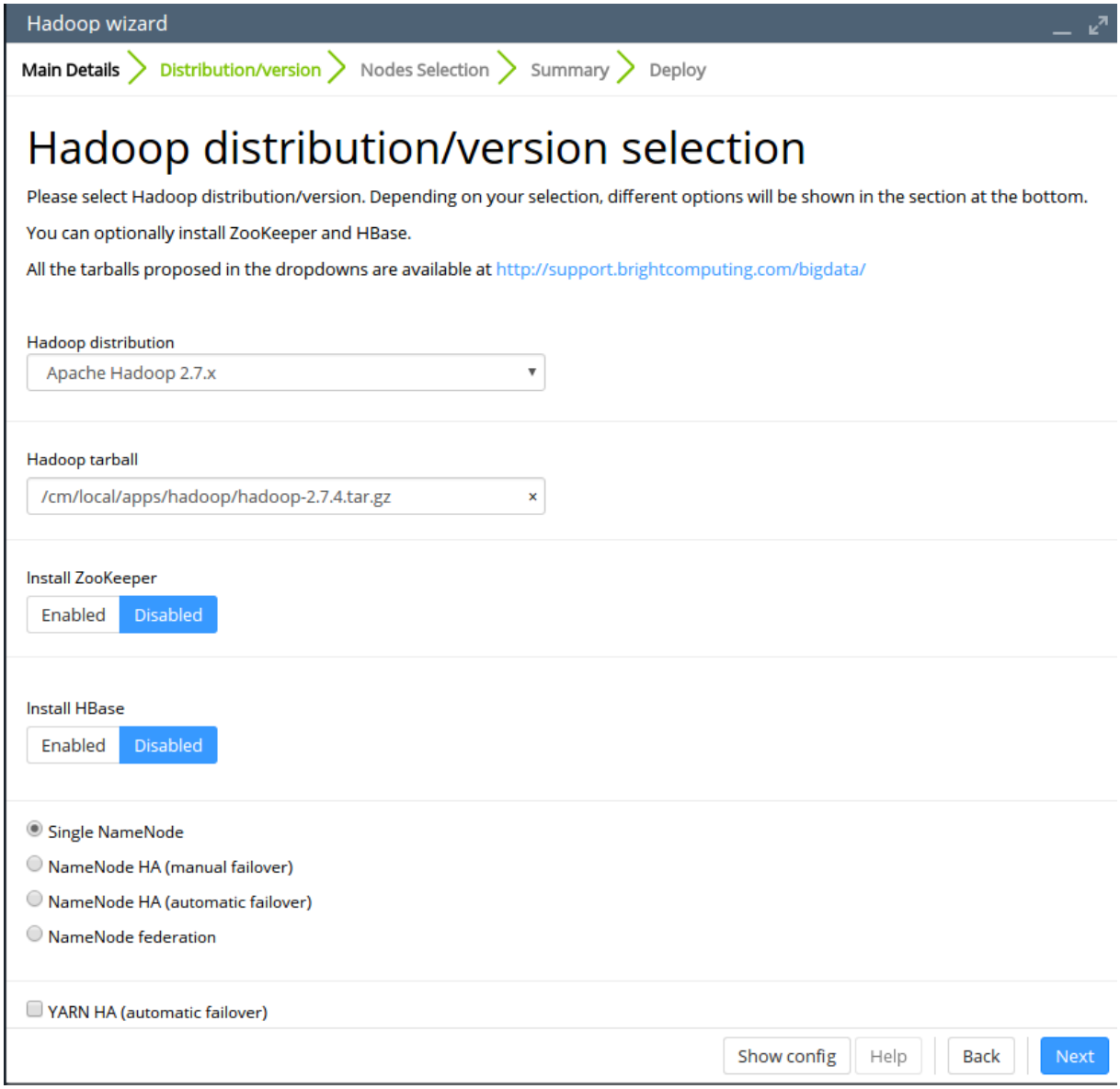

Figure 2.5: Installing Hadoop With Bright View: Distribution/Version Selection Page

The options are as follows:

- `Single NameNode`: This corresponds to regular HDFS, with no High Availability. This is the option for Hadoop 1.*x*-based distributions as well an option for Hadoop 2.*x*-based distributions that are configured to run like Hadoop 1.*x* distributions.

  NameNode can optionally have a SecondaryNameNode, which is configurable in the next screen of the wizard.

  A SecondaryNameNode offloads metadata operations from NameNode, and also stores the metadata 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

The items listed next are only for Hadoop 2.*x* versions:

- `NameNode High Availability (manual failover)`: This provides HDFS High Availability 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 High Availability (automatic failover)`: This provides HDFS High Availability with automatic failover[1]. It is 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.

  Federation is described further at `http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/Federation.html`.
- `YARN HA (automatic failover`: This option is available only for more recent Hadoop 2.*x* distributions, for example

  - Apache Hadoop 2.7.x
  - Cloudera CDH 5.3.x and higher
  - Hortonworks HDP 2.1 and higher

  This option also requires ZooKeeper to be installed.

Having chosen the Hadoop version and required associated software, the next screens are service allocation and selection screens for nodes. These screens allow the administrator to define which nodes (head nodes or regular nodes) are allocated to Hadoop and associated services. Suggestions are given and some restrictions are enforced. A node selection screen for an installation based on Hadoop 1.*x* is shown in figure 2.6.

[1] `http://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html#Automatic_Failover` explains the difference between manual and automatic failovers. From a Bright Cluster Manager perspective, manual failover requires JournalNode services. Hence, if selected, JournalNode roles will be assigned to some nodes. Automatic failover also relies on JournalNodes and ZooKeeper. Hence if automatic failover is chosen then ZooKeeper is marked for installation.

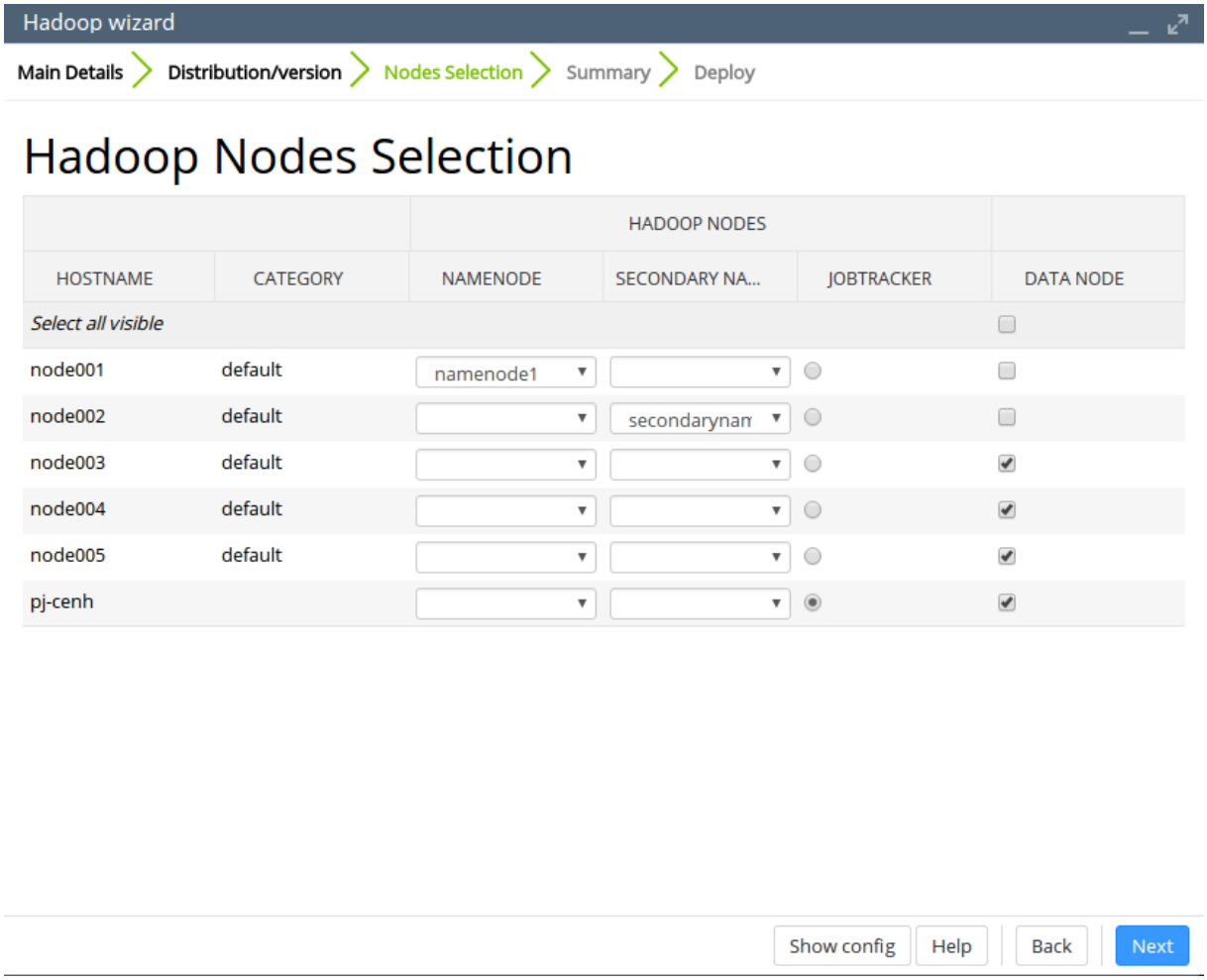

Figure 2.6: Installing Hadoop With Bright View: Example Node Selection Page For Hadoop 1.*x*

Other allocation screens may be shown depending on the Hadoop distribution, and other components (ZooKeeper and HBase) being configured.

The allocations can be made to the following host types:

- JournalNode
- JobTracker NameNode
- YARN server
- Key Management server
- DataNode
- ZooKeeper
- HBase

The roles associated with these services are then assigned to these nodes.

Some notes about the allocation choices that can be made:

- If YARN is chosen, then the administrator has to define two nodes for YARN ResourceManagers.

- DataNodes are configured explicitly by the administrator too. DataNodes are automatically coupled with MapReduce TaskTrackers for Hadoop 1.*x*-based distributions, and with their successor, YARN NodeManagers for Hadoop 2.*x*-based distributions.
- The wizard groups TaskTrackers/NodeManagers with the DataNode service, so that there is no dedicated selection window for TaskTrackers/NodeManagers. After installation, TaskTrackers/NodeNanagers can be de-coupled from DataNodes if needed.
- If the chosen distribution is Cloudera CDH 5.3 or higher, or HDP 2.2 or higher, then an administrator can choose to install Key Management Server.

After nodes have been allocated to the various Hadoop components, the screen that follows is the summary page (figure 2.7).

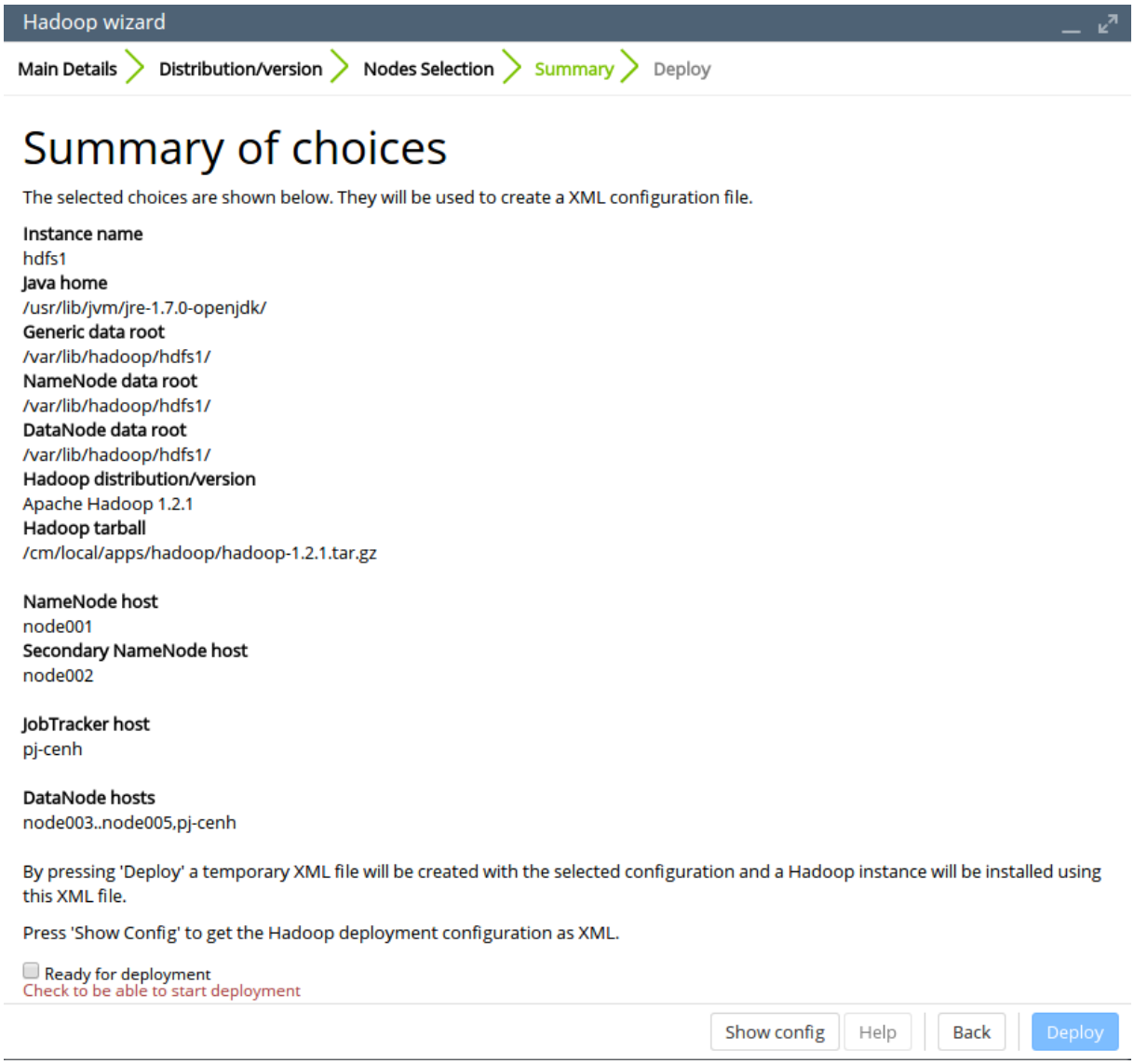

Figure 2.7: Installing Hadoop With Bright View: Wizard Summary Page

The summary shows the configuration settings, and is the final page in the wizard before installation. It does some validation to check all the required tarballs are in place, and that the JRE is available on all necessary nodes.

The summary page also allows the configuration to be viewed, and then saved locally, as an XML file. The saved XML file can be further customized if needed, and a locally saved XML configuration can be reloaded in the first page of the wizard using the `Load config` button (figure 2.3).

After the configuration has been checked by the administrator, the `Ready for deployment` checkbox can be ticked, and the `Deploy` button can then be clicked to carry out the installation of the instance. Installation progress is displayed, and the `Finish` button becomes active when installation is complete (figure 2.8):

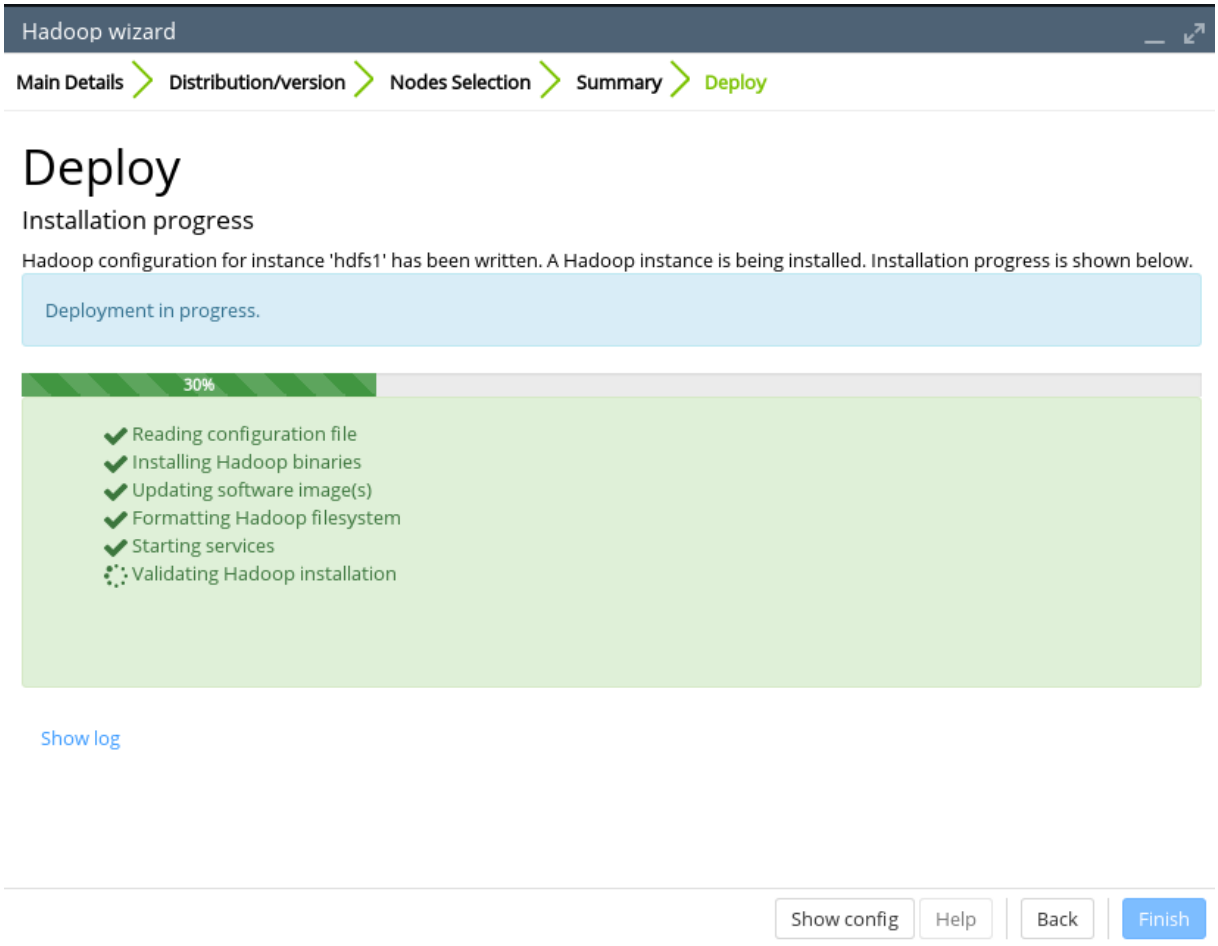

Figure 2.8: Installing Hadoop With Bright View: Wizard Installation Progress Page

Clicking on the `Finish` button ends the wizard.

2.4 Installing Hadoop With Lustre

The Lustre filesystem has a client-server configuration.

2.4.1 Lustre External Server Installation

Lustre can be configured so that the servers run external to Bright Cluster Manager. The Lustre Intel IEEL 3.x version can be configured in this manner.

2.4.2 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.

2.4.3 Lustre Hadoop Configuration With HAL

Intel's HAL (Hadoop Adapter for Lustre) plugin allows Hadoop to use Lustre as a replacement for HDFS.

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/hadoop2lustreconf.xml`.

The vanilla Apache and Cloudera CDH can both run with Lustre under Bright Cluster Manager. The configuration for these can be done as follows:

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

  Example

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

  As an alternative to specifying it by hand, the jar file can be built from the sources at `https://github.com/intel-hpdd/lustre-connector-for-hadoop`. The jar file must be built against a specific Hadoop version. Bright support should be contacted to get help with that.

  The installation of the Lustre plugin is automatic if the 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/hadoop2lustreconf.xml>`, then it can be run as:

Example

```
cm-hadoop-setup -c </root/hadoop2lustreconf.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 Bright View

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

Example

```
[hadoop2->bigdata]%  show apache274 | grep -i lustre
Hadoop root for Lustre                  /mnt/lustre/hadoop
Use Lustre                              yes
```

In Bright View, the click path `Big Data`→`Big Data Instances`[*instance name*]→`Edit`→`HDFS` shows whether Lustre is running.

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 7 describe installation of the additional components.

2.4.4 Lustre Hadoop Configuration With HAL And HAM

In addition to the HAL plugin, the HAM (Hadoop Adapter for MapReduce) can also be deployed. It allows Slurm to be used as a replacement for YARN. Hadoop jobs will be run on all the nodes with a Slurm Client role. In addition to the instructions in section 2.4.3, the following configurations must be taken care of.

Lustre Hadoop XML Configuration File Setup

Hadoop integration can be configured by using the file `/cm/local/apps/cluster-tools/hadoop/conf/hadoop2lustreslurmconf.xml` as a starting point for the configuration. It can be copied over, for example, to `/root/hadooplustreslurmconfig.xml`.

The configuration for these can be done as follows:

- The binary path for the Slurm installation must be specified in the `hadoop2lustreslurmconf.xml` file within the `<wlm></wlm>` tag pair:

- In addition, an <adapterjar></adapterjar> tag pair must be specified manually for the jar that the Intel IEEL 3.x distribution provides:

  Example

```
<wlm>
  <name>slurm</name>
  <binpath>/cm/shared/apps/slurm/14.11.6/bin</binpath>
  <adapterjar>/root/lustre/hadoop-hpc-scheduler-3.1.0-ieel-2.2.jar</adapterjar>
</wlm>
```

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

- The <datanodes></datanodes> tag pair should include (in <hosts></hosts>) the list of Slurm Clients
- The <yarnserver></yarnserver> tag pair should include (in <host></host>) one of the Slurm Clients, which will be used as "edge node" for the Hadoop installation.

2.4.5 Lustre Hadoop Configuration With The Seagate LustreFS plugin

The Seagate LustreFS plugin is an open source alternative to Intel's HAL plugin. The Seagate plugin allows Hadoop to use Lustre as a replacement for HDFS. It is available at https://github.com/Seagate/lustrefs.

Lustre Hadoop XML Configuration File Setup

Hadoop integration can be configured by using the file /cm/local/apps/cluster-tools/hadoop/conf/hadoop2lustreseagateconf.xml as a starting point for the configuration. It can be copied over to, for example, /root/hadoop2lustreseagateconfig.xml.

Standard vanilla Apache can run with Lustre under Bright Cluster Manager. The configuration for this can be carried out by modifying the hadoop2lustreseagateconf.xml as follows:

- A subdirectory of /mnt/lustre is specified within the <afs></afs> tag pair
- The value lustreseagate is specified within the <fstype></fstype> tag pair
- A <fsjar></fsjar> tag pair must be specified manually for the jar that was built from Seagate's GitHub repository:

  Example

```
<afs>
  <fstype>lustreseagate</fstype>
  <fsroot>/mnt/lustre/hadoop</fsroot>
  <fsjar>/root/lustre/lustrefs-hadoop-0.9.1.jar</fsjar>
</afs>
```

  If this jar name is set correctly, then Lustre plugin installation is automatic when cm-hadoop-setup is run.

Lustre Hadoop Installation With `cm-hadoop-setup`

The XML configuration file specifies how Lustre should be integrated with Hadoop. If the configuration file is at `</root/hadoop2lustreseagateconf.xml>`, then it can be used as follows:

Example

```
cm-hadoop-setup -c </root/hadoop2lustreseagateconf.xml>
```

As part of the integration configuration of Hadoop with Lustre, the execution:

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

  In particular, the subdirectory `./share/hadoop/common/lib` is copied into a directory relative to the Hadoop installation directory. For example, the Apache version of Hadoop, version 2.7.2, has the Hadoop installation directory `/cm/share/apps/hadoop/Apache/2.7.2`. The copy is then carried out in this case from:
  `/root/lustre/lustrefs-hadoop-0.9.1.jar`
  to
  `/cm/shared/apps/hadoop/Apache/2.7.2/share/hadoop/common/lib`

2.5 Installing Hadoop With GPFS

IBM Spectrum Scale, previously known as IBM General Parallel File System (GPFS), is a software-defined storage. A guide to installing it on Bright Cluster Manager is in the Bright Knowledge Base at `http://kb.brightcomputing.com/faq/index.php?action=artikel&cat=18&id=327`.

GPFS in its current incarnation as IBM Spectrum Scale HDFS Transparency can be integrated with Hadoop so that the NameNode and DataNodes are replaced by alternative NameNode and Datanodes. The original and alternative NameNode and Datanodes are just Hadoop services, and the architecture of the alternative services provide an HDFS-compliant API. This means that HDFS clients can transparently access the underlying GPFS via the standard interfaces offered by the alternative Hadoop services. The alternative NameNode and DataNodes should be able to access GPFS, so it is best to run them on nodes with direct access to GPFS. The other services (e.g. YARN ResourceManager, HBase Master) can run on other nodes, provided they can access NameNode and DataNodes.

2.5.1 GPFS Hadoop Configuration With Transparency Package

Prerequisites for the installation of Hadoop with GPFS:

- Java 1.8.0 JRE must be installed on all the nodes to be used in the Hadoop instance.
- GPFS must be installed and available as a mount point on at least one node. The node can be a head node or a compute node. The mount point should have the same path across all nodes.
- IBM Spectrum Scale HDFS Transparency package must be installed on all the nodes to be used as NameNode and DataNodes. At the time of writing (September 2017) several packages are available at:

  `https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General%20Parallel%20File%20System%20(GPFS)/page/2nd%20generation%20HDFS%20Transparency%20Protocol`

  In the preceding URL, the "`...General%20`" bit is actually meant to be joined to the "`Parallel...`" bit, so that it is all one long URL.

  The following packages have been tested and are known to work with the latest Apache Hadoop 2.7.x, Cloudera Hadoop CDH 5.12.x, and Hortonworks HDP 2.6.x:

- gpfs.hdfs-protocol-2.7.0-5.x86_64.rpm
- gpfs.hdfs-protocol-2.7.2-3.x86_64.rpm
- gpfs.hdfs-protocol-2.7.3-0.x86_64.rpm

GPFS Hadoop XML Configuration File Setup

Hadoop integration can be configured by using the file `/cm/local/apps/cluster-tools/hadoop/conf/hadoop2gpfs.xml` as a starting point for the configuration. It can be copied over to, for example, `/root/hadoop2gpfsconf.xml`.

The vanilla Apache, Cloudera CDH, and Hortonworks HDP Hadoop can all run with GPFS under Bright Cluster Manager. The configuration for these can be done within the `<afs></afs>` (alternative file system) tag pair, as follows:

- The `<fstype></fstype>` tag pair must be set to the value `gpfs`
- The `<fsroot></fsroot>` tag pair must be set to the GPFS mount point. For example `/gpfs1`.
- The `<fsdataroot></fsdataroot>` tag pair must be set to the preferred subdirectory of the GPFS mount point to be used as root for the Hadoop data. For example, the value could be set to `projectx`, in which case the NameNode automatically creates a subdirectory `projectx`. This directory then becomes accessible as `/gpfs1/projectx` via POSIX.
- The `<adapterpath></adapterpath>` tag pair must be set within the XML hierarchy for the path of the HDFS Transparency package. The default path is as shown in following example.

Example

```
<afs>
  <fstype>gpfs</fstype>
  <fsroot>/gpfs1/</fsroot>
  <fsdataroot>projectx</fsdataroot>
  <adapterpath>/usr/lpp/mmfs/hadoop</adapterpath>
</afs>
```

The SecondaryNameNode cannot be specified, since it is not used.

The HDFS Transparency package configuration files are placed in `/usr/lpp/mmfs/hadoop/etc/hadoop`.

Also part of the HDFS Transparency package are some scripts that are used to synchronize scripts among servers and to manage NameNode and DataNode services. These are placed within `/usr/lpp/mmfs/hadoop/sbin`. They should be left alone, since Bright Cluster Manager takes care of the relevant configuration files and services.

2.6 Installing Hadoop With BeeGFS

BeeGFS is a high-performance parallel file system, developed by the Fraunhofer Competence Center for High Performance Computing. It is optimized for intensive I/O. Bright Cluster Manager provides a deployment tool for BeeGFS (Chapter 10 of the *Administrator Manual*).

Hadoop can be configured to use BeeGFS as an alternative to HDFS.

If so, then:

- NameNode and DataNodes, which are normally used by Hadoop, are no longer needed since they are replaced by BeeGFS services.
- Hadoop clients and YARN services, namely ResourceManager and NodeManagers, access data from the underlying BeeGFS

  either

– via a POSIX interface, called the *file* scheme. This provides a direct access, and is suggested for performance reasons.

or

– via Java connector, called the *beegfs* scheme. This uses Java classes and shared libraries.

In both cases, clients can access Hadoop data via the usual `hdfs dfs` commands.

2.6.1 BeeGFS Hadoop Configuration With Transparency Package

Prerequisites and possible configurations for the installation of Hadoop with BeeGFS:

- All the nodes involved in the Hadoop deployment should be already configured as metadata, storage, and client nodes via `cm-beegfs-setup`.
- The management node for BeeGFS can be freely chosen.
- ACLs should be enabled on BeeGFS.

  They can be enabled by setting the following properties to `true` for the `Metadata` role:

  – `storeUseExtendedAttribs`
  – `storeClientXAttrs`
  – `storeClientACLs`

  and the following properties to `true` for the Client role:

  – `sysXAttrsEnabled`
  – `sysACLsEnabled`

  Further information on these properties can be found in section 10.3.2 of the *Administrator Manual*, which discusses BeeGFS objects.

  When ACLs are enabled, Bright Cluster Manager ensures that the home directories for Hadoop (`/user/<username>/`) can be properly managed by their respective users. During deployment `cm-hadoop-setup` informs the administrator if ACLs are not enabled.
- In order to use the Hadoop connector, Java 1.8.0 JRE must be installed on all the nodes to be used in the Hadoop instance.
- The mount point for BeeGFS should be the same on all selected nodes. Hadoop will use a subdirectory of the mount point as root (i.e. `/`) for its data. This lets multiple Hadoop instances co-exist using the same BeeGFS filesystem, and with each instance getting its own, necessary, subdirectory.

Compatibility With Additional Tools

Apache ZooKeeper, HBase, Spark, Hive, Pig have been tested with Hadoop on BeeGFS and are known to work. The appropriate sections in chapter 7 describe the installation of the additional components.

BeeGFS Hadoop XML Configuration File Setup (POSIX Interface)

Hadoop integration can be configured by using the file `/cm/local/apps/cluster-tools/hadoop/conf/hadoop2beegfsposixconf.xml` as a starting point for the configuration. It can be copied over to, for example, `/root/hadoop2beegfsconf.xml`.

The vanilla Apache 2.7.x can run with BeeGFS under Bright Cluster Manager. The configuration for these can be done within the `<afs></afs>` (alternative file system) tag pair, as follows:

- The `<fstype></fstype>` tag pair must be set to the value `beegfs-posix` or `beegfs`
- The `<fsroot></fsroot>` tag pair must be set to the BeeGFS mount point. For example `/mnt/beegfs`.

- The <fsdataroot></fsdataroot> tag pair must be set to the preferred subdirectory of the BeeGFS mount point to be used as root for the Hadoop data. For example projectx. The script cm-hadoop-setup automatically creates a subdirectory projectx, which will be accessible as /mnt/beegfs/projectx via POSIX.
- The <fsjar></fsjar> tag pair can be left empty.

Example

```
<afs>
  <fstype>beegfs-posix</fstype>
  <fsroot>/mnt/beegfs/</fsroot>
  <fsdataroot>projectx</fsdataroot>
</afs>
```

NameNode and SecondaryNameNode cannot be specified, since they are not used.

BeeGFS Hadoop XML Configuration File Setup (Hadoop Connector)

Hadoop integration can be configured by using the file /cm/local/apps/cluster-tools/hadoop/conf/hadoop2beegfsjavaconf.xml as a starting point for the configuration. It can be copied over to, for example, /root/hadoop2beegfsconf.xml.

The vanilla Apache 2.7.x can run with BeeGFS under Bright Cluster Manager. The configuration for these can be done within the alternative filesystem <afs></afs> tag pair, as follows:

- The <fstype></fstype> tag pair must be set to the value beegfs
- The <fsroot></fsroot> tag pair must be set to the BeeGFS mount point. For example /mnt/beegfs.
- The <fsdataroot></fsdataroot> tag pair must be set to the preferred subdirectory of the BeeGFS mount point to be used as root for the Hadoop data. For example, the value could be set toprojectx, in which case the cm-hadoop-setup script automatically creates a subdirectory projectx. This subdirectory then becomes accessible as /mnt/beegfs/projectx via POSIX.
- In addition, an <fsjar></fsjar> tag pair must be specified manually for the jar that the BeeGFS provides at https://www.beegfs.io/wiki/HadoopConnector. At the time of writing (August 2017), the direct link for the connector is http://www.beegfs.com/downloads/hadoop/beegfs-hadoop-connector-v1.0.tar.gz.

Example

```
<afs>
  <fstype>beegfs-java</fstype>
  <fsroot>/mnt/beegfs/</fsroot>
  <fsdataroot>projectx</fsdataroot>
  <fsjar>/root/beegfs.jar</fsjar>
</afs>
```

NameNode and SecondaryNameNode cannot be specified, since they are not used.

BeeGFS Hadoop Installation With cm-hadoop-setup

The XML configuration file specifies how BeeGFS should be integrated in Hadoop. If the configuration file is at </root/hadoop2beegfsconf.xml>, then it can be run as:

Example

```
cm-hadoop-setup -c </root/hadoop2beegfsconf.xml>
```

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

- Create the subdirectory specified within the <fsdataroot><fsdataroot> tag pair. In an example earlier it was set to projectx, which means the directory /mnt/beegfs/projectx would then be created. Within this directory, the script creates four subdirectories /tmp, /user, /staging, /hbase.
- If the Hadoop connector has been specified, then the script makes the shared library /opt/beegfs/lib/libjbeegfs.so, and one additional jar file /opt/beegfs/lib/libjbeegfs.so, available to Hadoop services and clients.
- Create configuration overlays for YARN services, which will be run on the selected nodes.

2.7 Installation Of Other Hadoop Components

Bright Cluster Manager supports a number of popular systems relying on the main Hadoop framework. Section 7 covers some of these systems and describes their installation/de-installation procedures.

2.8 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 2.6.1 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

Big Data Cluster Management

The basic management of a big data cluster using Bright View, `cmsh`, and the command line, is described in this chapter.

3.1 Managing A Hadoop Instance With Bright View

In Bright View, the `Big Data Instances` menu option in the resource tree opens up to display a list of the big data instances running on the cluster (figure 3.1).

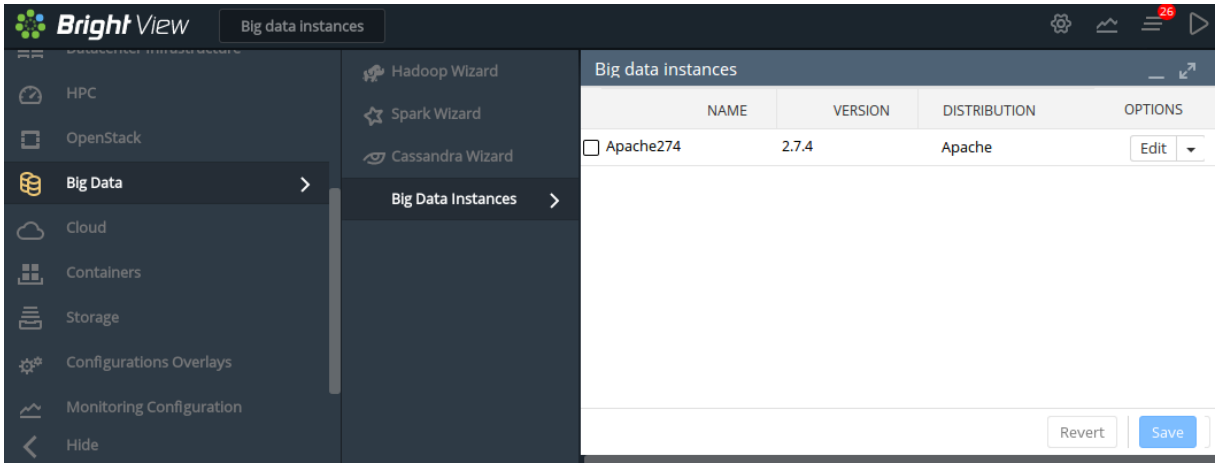

Figure 3.1: Big Data Instances List In Bright View

Clicking on the `Edit` button for a big data instance opens up the menu options associated with the instance (figure 3.2):

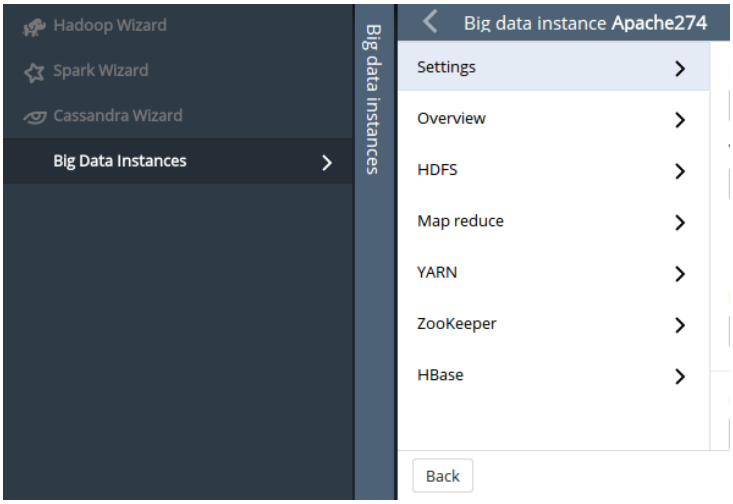

Figure 3.2: Big Data Instance Menu Options In Bright View

The following big data instance menu options are described further within this section:

1. `Settings` (section 3.1.1)
2. `Overview` (section 3.1.2)
3. `HDFS` (section 3.1.3)
4. `MapReduce` or `YARN` (section 3.1.4)
5. `Zookeeper` (section 3.1.5)
6. `HBase` (section 3.1.6)

Not all of these tabs are necessarily displayed. What is displayed depends 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 Big Data Instance `Settings` Option

The `Settings` option opens up a pane that displays settings options for the big data instance (figure 3.3). It allows the cluster administrator to configure a number of these options, in the pane or in further subpanes.

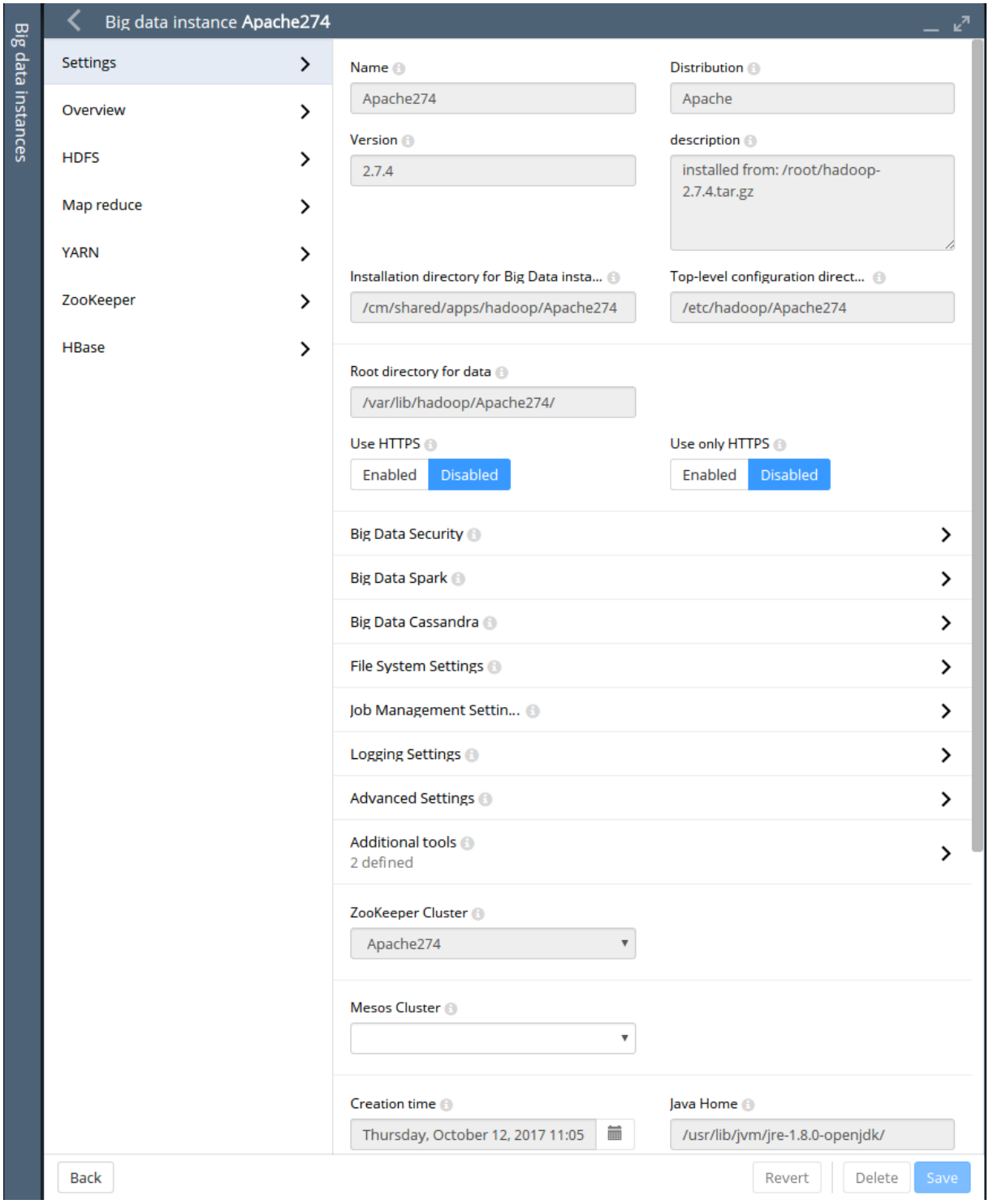

Figure 3.3: Settings Options For A Big Data Instance In Bright View

The locations of Hadoop components or temporary files placement, can be viewed from within this option.

Among the other parameters that can be viewed can be viewed and changed are:

- **Topology:** The `Topology` option is available via the clickpath `Big Data→Big Data Instances→Edit→Settings→File System Settings→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:** HDFS balancing values configure how data blocks are moved from overused to underused nodes. They are settable via the clickpath `Big Data→Big Data Instances→Edit→Settings→File System Settings`. The values that can be set are:

  - `HDFS balancer period`: The period in hours between balancing operations.
  - `HDFS 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.
  - `HDFS 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`
  - `...`

3.1.2 The Big Data Instance `Overview` Option

The `Overview` option opens up a pane (figure 3.4) that aggregates the information about all big data components and conveniently displays it in blocks.

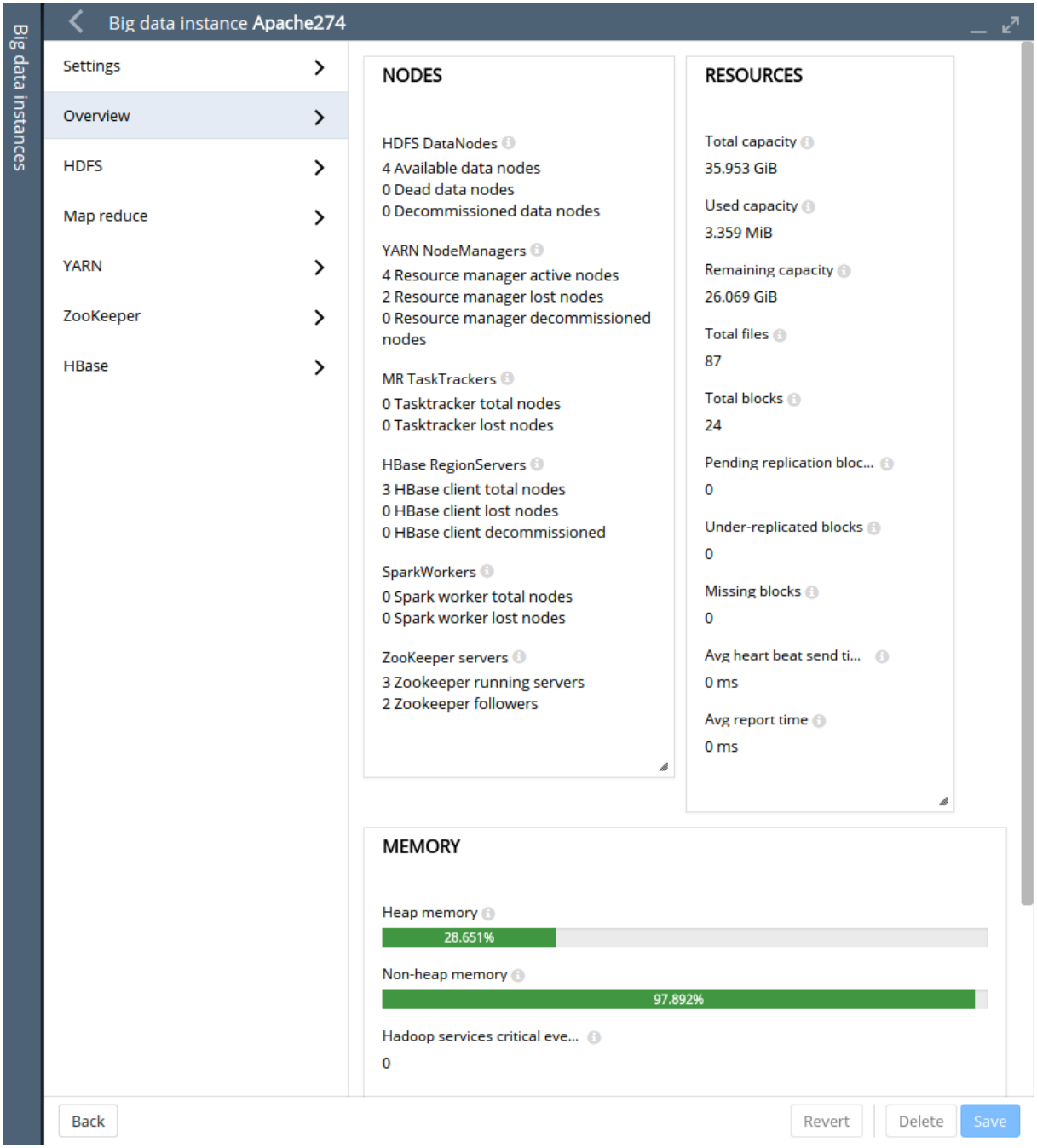

Figure 3.4: Overview Pane For A Big Data Instance In Bright View

The overview pane quantifies the state of the cluster, by listing for big data, the following information in blocks:

- `NODES`: the number of live, dead, and decommissioned components, such as Hadoop, YARN, HBase, Spark, ZooKeeper.
- `RESOURCES`: the storage resources consumed
- `MEMORY`: the heap, non-heap, and critical event memory use

3.1.3 The Big Data Instance HDFS Option

The HDFS option opens a pane that presents the HDFS state (figure 3.5). The pane quantifies the state of HDFS for the cluster, by listing, in blocks:

- MEMORY: The JVM memory used by the NameNode, secondary NameNode, and DataNode
- DISK: the capacity of the HDFS
- NODES: the number of live, dead, and decommissioned HDFS nodes.
- FILES: the files and blocks used. Corrupt and missing blocks are also listed, as well as the replication status.

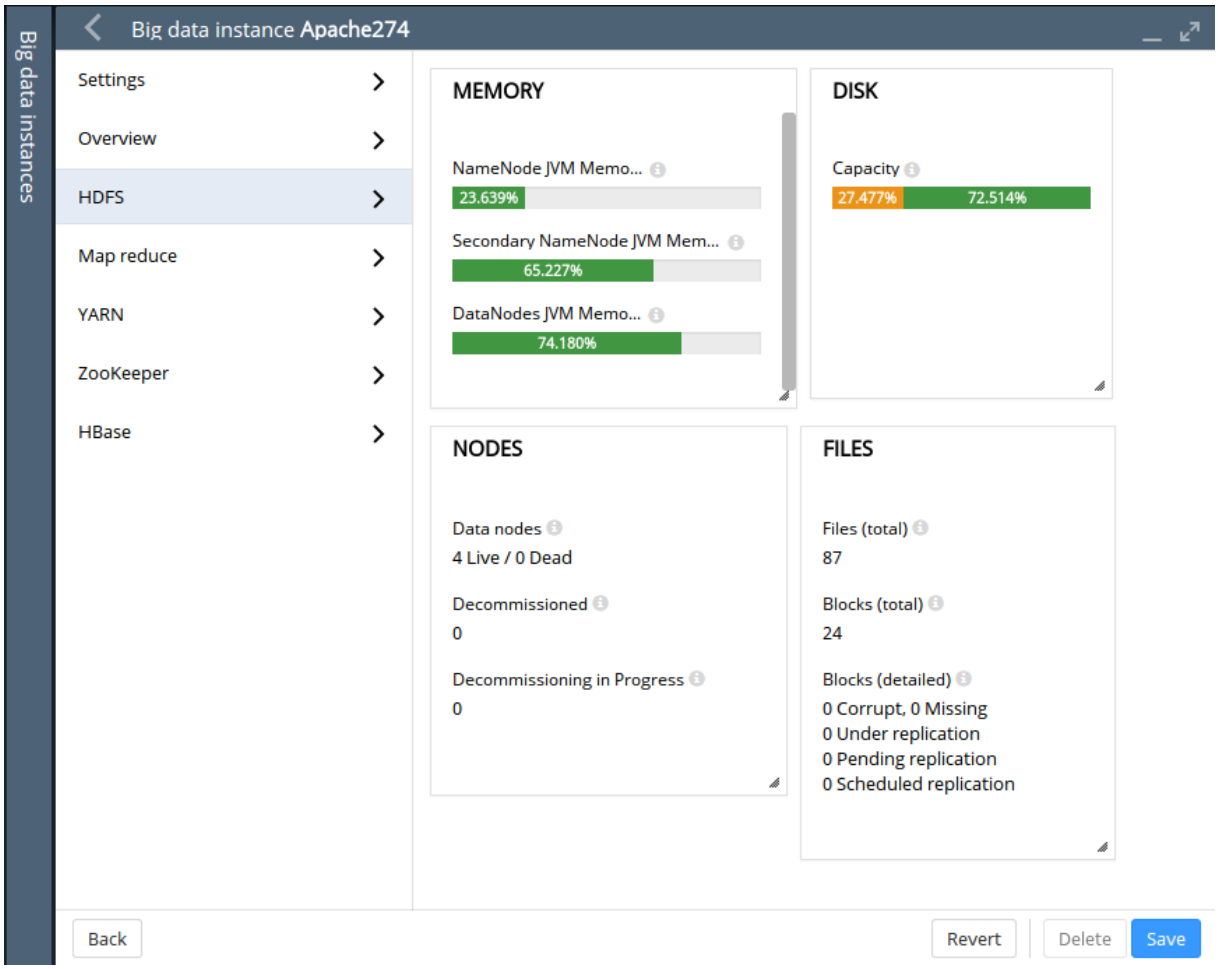

Figure 3.5: HDFS Pane For A Big Data Instance In Bright View

3.1.4 The Big Data Instance MapReduce Or YARN Options

MapReduce is used in older Hadoop distributions, such as Apache Hadoop 1.2.1. If the MapReduce option is selected, then a pane with MapReduce properties is displayed (figure 3.6).

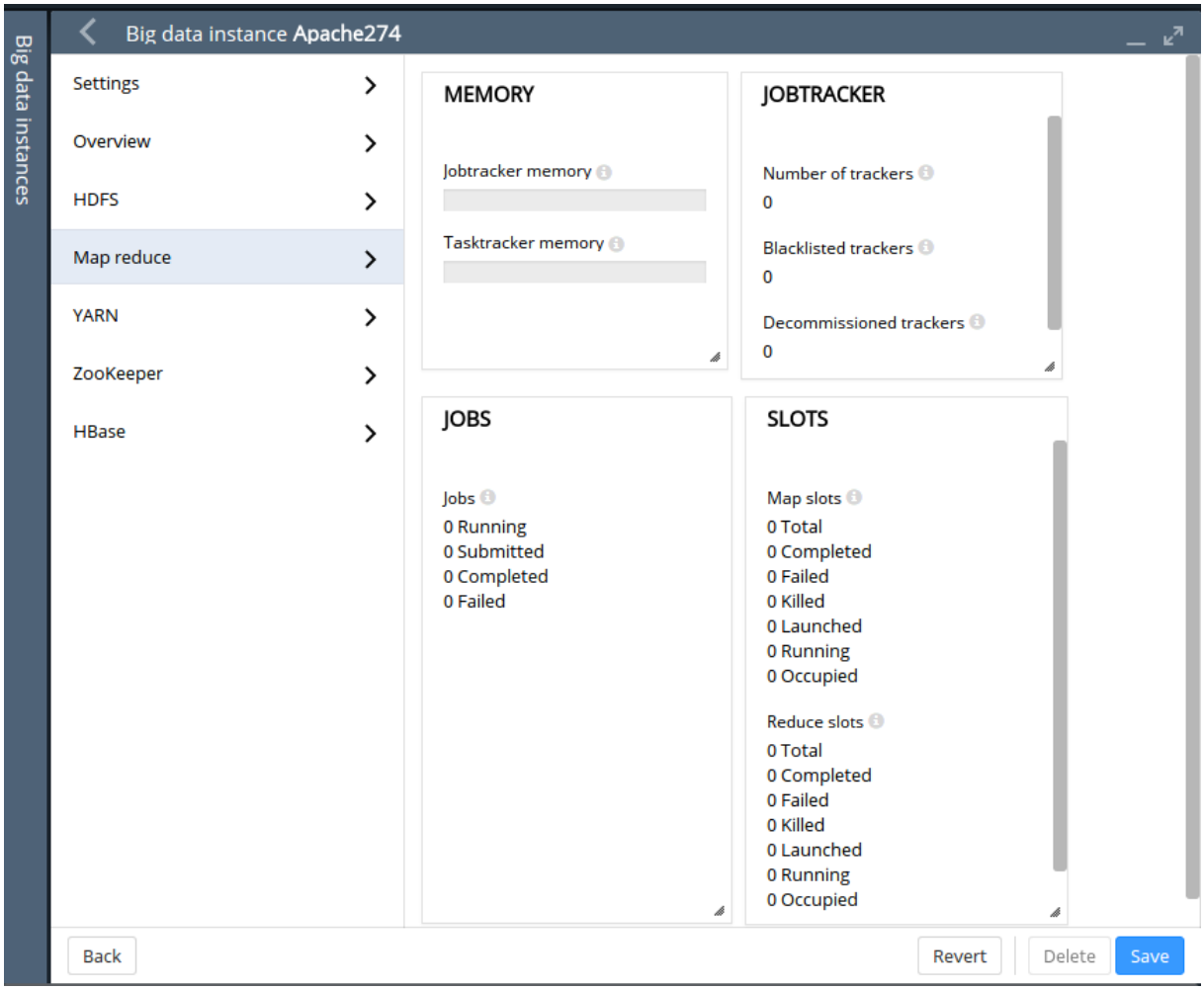

Figure 3.6: MapReduce Option For A Big Data Instance In Bright View

The MapReduce pane shows:

- `MEMORY`: Memory used by the jobtrackers and tasktrackers
- `JOBTRACKER`: Numbers of jobtrackers according to status
- `JOBS`: Numbers of jobs according to status
- `SLOTS`: Map and reduce slots

More recent Hadoop distributions use YARN instead. If the YARN option is selected, then a pane with YARN properties is displayed (figure 3.7).

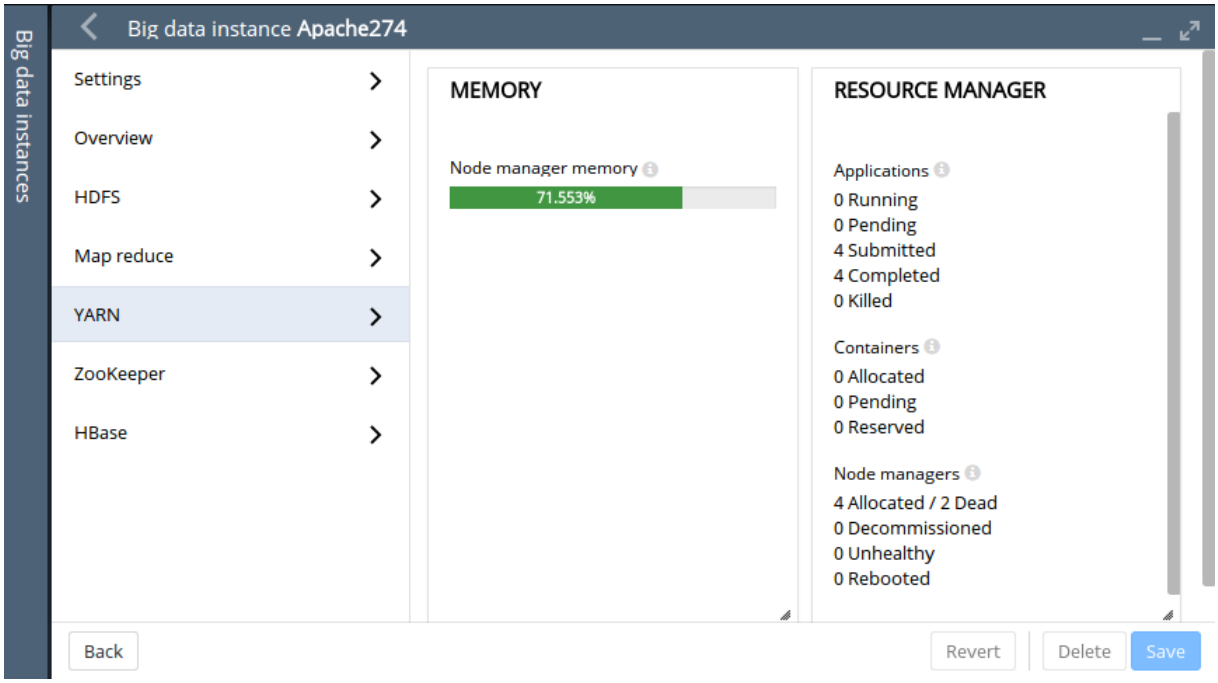

Figure 3.7: YARN Option For A Big Data Instance In Bright View

The YARN pane shows:

- `MEMORY`: The node manager memory use
- `RESOURCES`: Resources allocated to applications, containers, and node managers

3.1.5 The Big Data Instance `Zookeeper` Option

The `Zookeeper` option opens up a pane that shows the ZooKeeper status, resource consumption, statistics, and performance (figure 3.8):

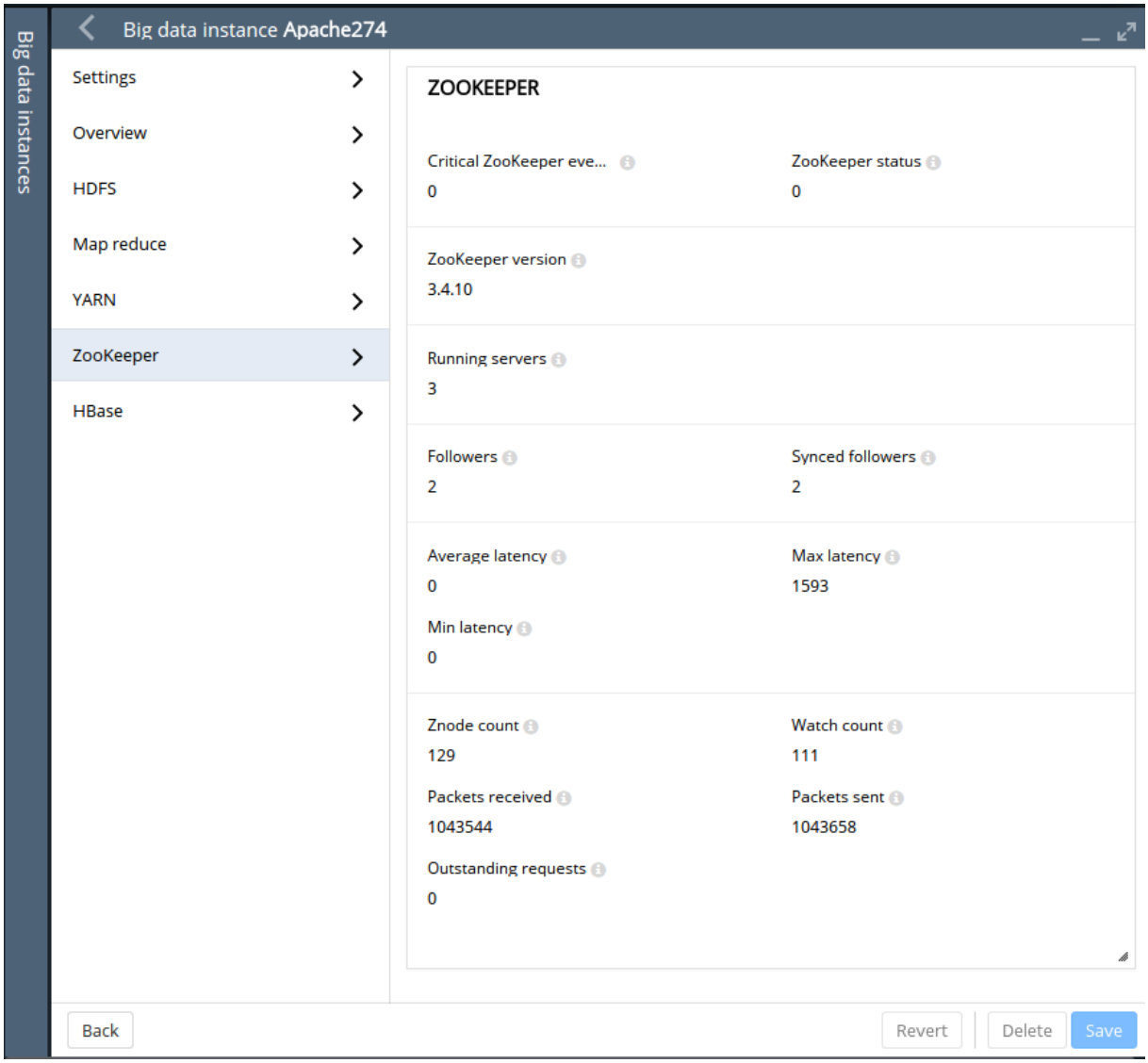

Figure 3.8: Zookeeper Pane For A Big Data Instance In Bright View

3.1.6 The Big Data Instance `HBase` Option

The `HBase` option opens up a pane that shows the HBase resource consumption (figure 3.9).

The HBase pane shows:

- `MEMORY`: Memory used by the HBase master and HBase regions servers
- `REGIONS`: The status of the RegionServers
- `FILES`: The number and total size of the store files
- `PERFORMANCE`: Performance of RegionServers I/O and average block cache hits

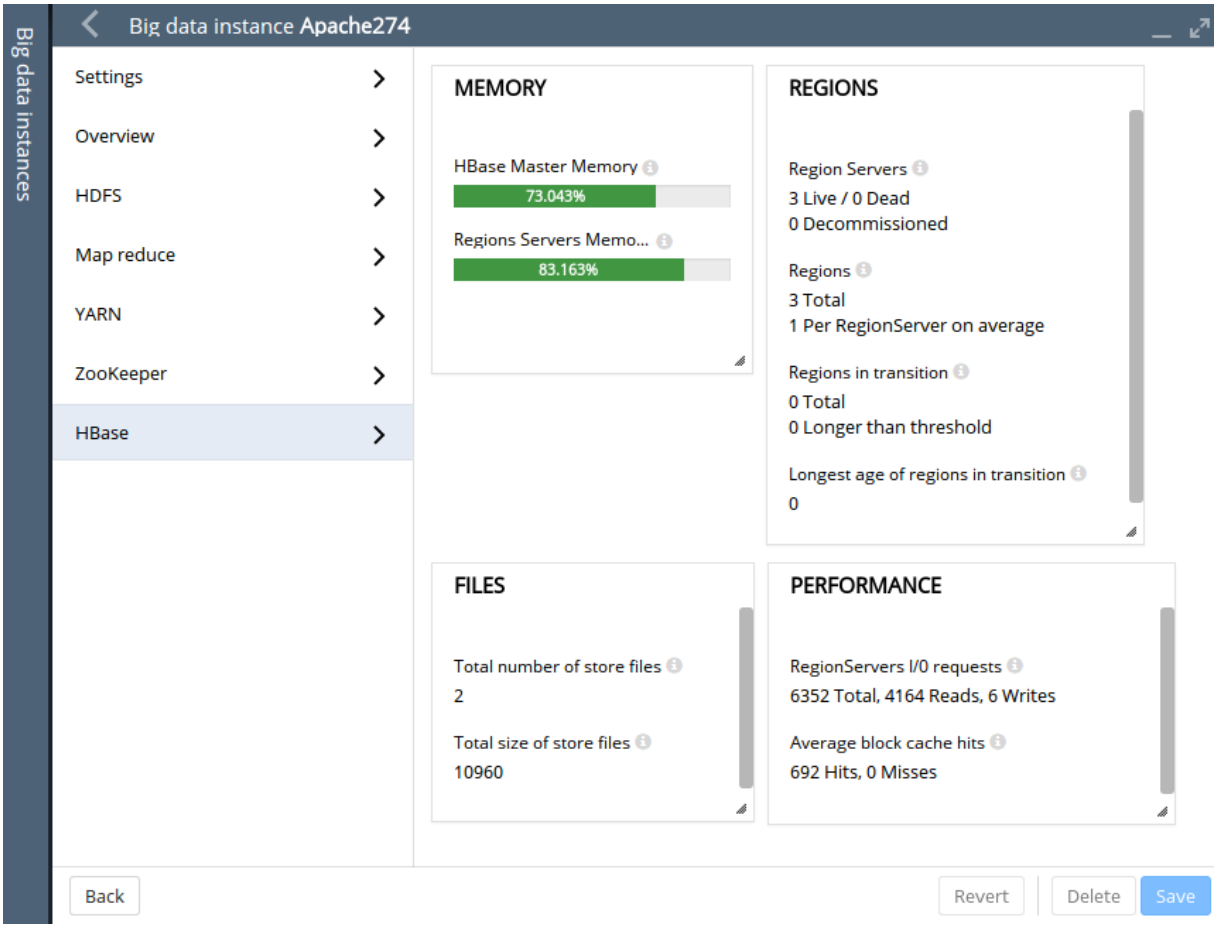

Figure 3.9: HBase Pane For A Big Data Instance In Bright View

3.1.7 Big Data Configuration Overlays

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 big data configuration overlay assigns a group of roles to a big data instance. Thus, when a configuration overlay is set for a big data instance, then roles are assigned to nodes according to the configuration, along with a priority. Whether the configuration overlay 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 overlay 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 overlay 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 Big Data Configuration Overlays In Bright View

The `Configuration Overlays` resource opens a pane that lists configuration overlays and their main properties.

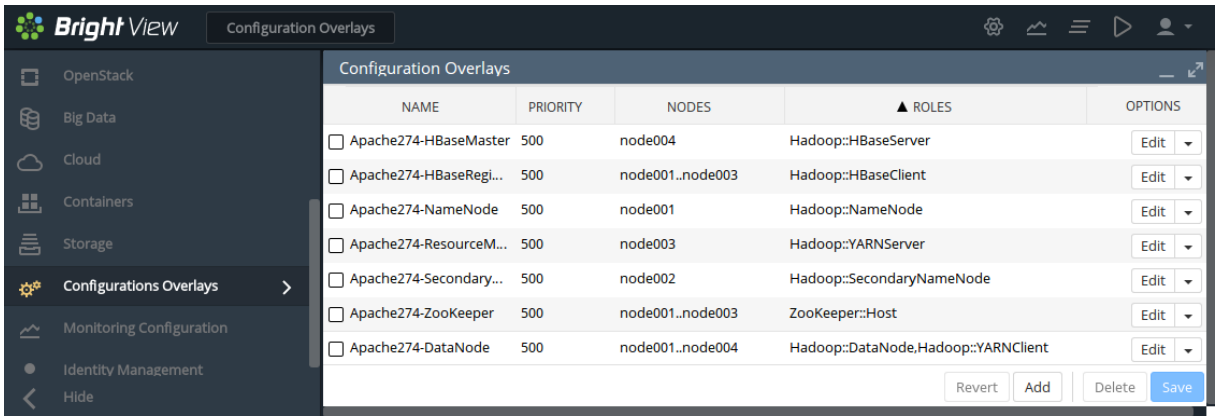

Figure 3.10: Configuration Overlays For A Big Data Instance In Bright View

It is possible to edit the properties of a particular overlay. The properties that can be modified include nodes, categories, roles, and priorities, but there are many more at many configuration levels. The semi-collapsed view in figure 3.11 illustrates one of the deeper subpanes and the path to get there.

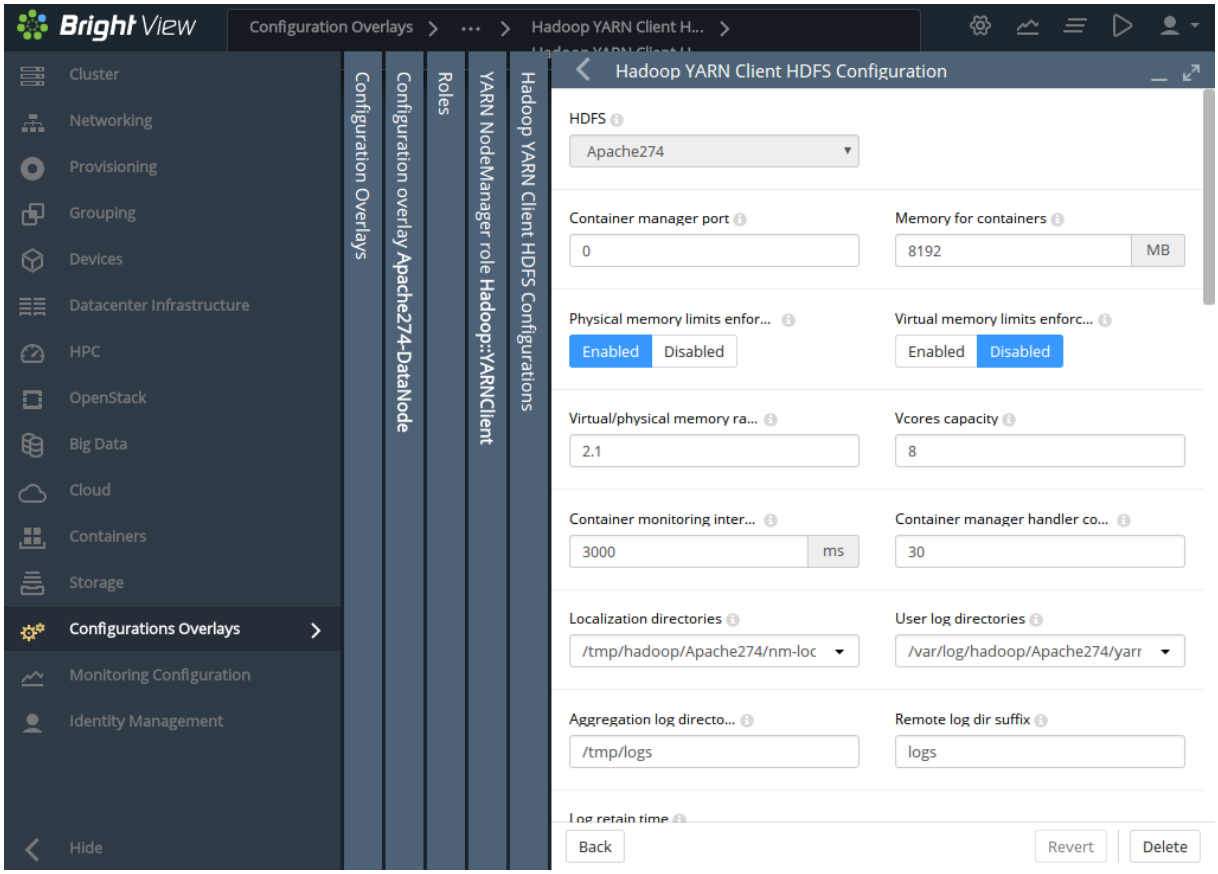

Figure 3.11: Deeper Level Configuration For The Configuration Overlay Of A Big Data Instance In Bright View

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

<hadoop instance name>-<big data service role>

Example

`doop-DataNode` for an instance called `doop` and with a role of `Hadoop::DataNode`

There is a great deal of flexibility in dealing with configuration overlays and their roles. Roles can be created, added, or removed, while configuration overlays can in addition also be cloned.

However, it should be noted that the HBase MasterServer, NameNode, and Spark Master roles are often depended upon by other roles. Modifying these roles should therefore only be done with great 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.

3.1.8 Big Data Instance Monitoring Visualization

The monitoring icon of Bright View—the icon at the top right of Bright View that is a stylized graph plot—brings up the monitoring visualization display. The icon, and monitoring visualization capabilities of Bright Cluster Manager, are described further in section 12.3 of the *Administrator Manual*. Just as for other measurables, measurables from a big data instance can be displayed in the plot panels using drag-and-drop (figure 3.12).

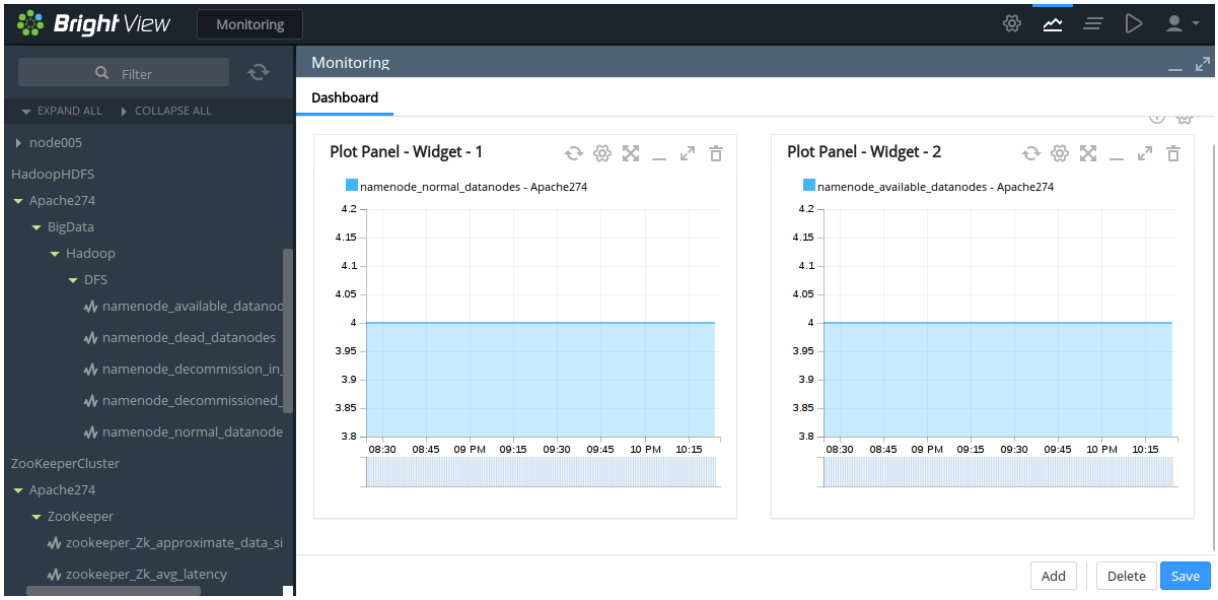

Figure 3.12: Monitoring Visualisation For A Big Instance In Bright View

3.2 Managing A Big Data Instance With cmsh

3.2.1 cmsh And bigdata Mode

The cmsh front end uses the bigdata mode to display information on big-data-related values and to carry out big-data-related tasks.

Example

```
[root@bright81 conf]# cmsh
[bright81]% bigdata
[bright81->bigdata]%
```

The show And overview Commands

The overview Command: Within bigdata mode, the overview command displays three sections of interest.

1. A parameter value section that corresponds somewhat to excerpts from the big data instance menu options (figure 3.2)
2. A configuration overlay section, that corresponds somewhat to the big-data-related Configuration Overlays pane (figure 3.10)
3. A big data service section, that corresponds to the big data services used by the various nodes

Example

```
[bright81->bigdata]% overview Apache274
Parameter                      Value
------------------------------ --------------------------------
Name                           Apache274
Capacity total                 35.95GB
Capacity used                  3.355MB
Capacity remaining             26.04GB
Heap memory total              1.728GB
Heap memory used               425.4MB
```

```
Heap memory remaining           1.313GB
Non-heap memory total           925.2MB
Non-heap memory used            903.4MB
Non-heap memory remaining       21.84MB
Nodes available                 4
Nodes dead                      0
Nodes decommissioned            0
Nodes decommission in progress  0
Total files                     87
Total blocks                    24
Missing blocks                  0
Under-replicated blocks         0
Scheduled replication blocks    0
Pending replication blocks      0
Block report average Time       0
Applications running            0
Applications pending            0
Applications submitted          4
Applications completed          4
Applications failed             0
Federation setup                no

Big Data role              Nodes             Configuration group          Nodes up
-------------------------- ----------------- ---------------------------- ---------
Hadoop::DataNode           node001..node004  Apache274-DataNode             4 of 4
Hadoop::HBaseClient        node001..node003  Apache274-HBaseRegionServer    3 of 3
Hadoop::HBaseServer        node004           Apache274-HBaseMaster          1 of 1
Hadoop::NameNode           node001           Apache274-NameNode             1 of 1
Hadoop::SecondaryNameNode  node002           Apache274-SecondaryNN          1 of 1
Hadoop::YARNClient         node001..node004  Apache274-DataNode             4 of 4
Hadoop::YARNServer         node003           Apache274-ResourceManager      1 of 1

Big Data service                     Nodes             Service status
------------------------------------ ----------------- ---------------
hadoop-Apache274-datanode            node001..node004  [ 4 / 4 UP ]
hadoop-Apache274-hbase-master        node004           [ 1 / 1 UP ]
hadoop-Apache274-hbase-regionserver  node001..node003  [ 3 / 3 UP ]
hadoop-Apache274-jobhistory          node003           [ 1 / 1 UP ]
hadoop-Apache274-namenode            node001           [ 1 / 1 UP ]
hadoop-Apache274-nodemanager         node001..node004  [ 4 / 4 UP ]
hadoop-Apache274-resourcemanager     node003           [ 1 / 1 UP ]
hadoop-Apache274-secondarynamenode   node002           [ 1 / 1 UP ]
hadoop-Apache274-timelineserver      node003           [ 1 / 1 UP ]
hadoop-Apache274-zookeeper           node001..node003  [ 3 / 3 UP ]
```

The `show` Command: The `show` command displays parameters that correspond mostly to the `Settings` option of the big data instance in Bright View (section 3.1.1):

Example

```
[bright81->bigdata]% show apache274
Parameter                                      Value
---------------------------------------------- ------------------------------------------------
Additional tools                               <2 in submode>
Advanced Settings                              <submode>
Big Data Cassandra                             <submode>
```

```
Big Data Security                                <submode>
Big Data Spark                                   <submode>
Creation time                                    Thu, 12 Oct 2017 11:05:59 CEST
Distribution                                     Apache
File System Settings                             <submode>
Hadoop Edge Nodes
Installation directory for Big Data instance     /cm/shared/apps/hadoop/Apache274
Installed components
Java Home                                        /usr/lib/jvm/jre-1.8.0-openjdk/
Job Management Settings                          <submode>
Logging Settings                                 <submode>
Mesos Cluster
Name                                             Apache274
Native libraries path                            /cm/shared/apps/hadoop/native_libraries
Revision
Root directory for data                          /var/lib/hadoop/Apache274/
Top-level configuration directory                /etc/hadoop/Apache274
Use HTTPS                                        no
Use only HTTPS                                   yes
Version                                          2.7.4
ZooKeeper Cluster                                Apache274
description                                      installed from: /root/hadoop-2.7.4.tar.gz
```

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

```
[bright81->bigdata]% get
Name:
          get - Get specific hadoophdfs property

Usage:
          get [hadoophdfs] <parameter>

Arguments:
          hadoophdfs
               name of the hadoophdfs, omit if current is set

Parameters:
          Revision ............ Entity revision
          additionaltools ..... Submode for additional tools
          advancedsettings .... Submode containing Advanced settings
          bigdatacassandra .... Submode containing Cassandra settings
          bigdatasecurity ..... Submode containing Big Data security settings
          bigdataspark ........ Submode containing Spark settings
          creationtime ........ Time when Hadoop instance was created
          description ......... Description
          distribution ........ Distribution
          filesystemsettings .. Submode containing File System settings
          hadoopedgenodes ..... Edge Nodes used to access the instance
          installationdirectoryforbigdatainstance  Installation directory
          installedcomponents . Components installed using cmhadoop-*-setup, since 7.2
          javahome ............ JAVA_HOME path used by this Big Data instance
          jobmanagementsettings  Submode containing Job Management settings
```

```
loggingsettings ..... Submode containing Logging settings
mesoscluster ........ The Mesos cluster instance (pointer)
name ................ Big data instance name
nativelibrariespath . The path to native libraries and tools used in securing cluster
notes ............... Notes
rootdirectoryfordata  Root directory for other data directories
top-levelconfigurationdirectory  Configuration directory
usehttps ............ Use HTTPS for web UIs
useonlyhttps ........ Use only HTTPS for web UIs
version ............. Version
zookeepercluster .... The ZooKeeper cluster instance (pointer)
```

The `*services` Commands For Big Data Services

Big data services can be started, stopped, and restarted, with:

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

Example

```
[bright81->bigdata[Apache274]]% restartallservices
Will now stop all Hadoop services for instance 'Apache274'... done.
Will now start all Hadoop services for instance 'Apache274'... done.
[bright81->bigdata[Apache274]]%
```

The `*balancer` Commands For HDFS, 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

```
[bright81->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 `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

```
[bright81->bigdata]% formathdfs apache274
Will now format and set up HDFS for instance 'Apache274'.
Stopping HBase regionservers... done.
Stopping HBase master... done.
Stopping datanodes... done.
Stopping namenodes... done.
Formatting HDFS... done.
Starting namenode (host 'node001')... done.
Starting datanodes... done.
Waiting for datanodes to come up... done.
Setting up HDFS... done.
Starting HBase master... done.
Starting HBase regionservers... done.
[bright81->bigdata[Apache274]]%
```

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 overlays are introduced in section 3.1.7. Within Bright View, Hadoop configuration overlays can be accessed from within the Configuration Overlays resource.

Configuration Overlay Listing

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

Example

```
[root@bright81 ~]# cmsh
[bright81->configurationoverlay]% list -f name:28,priority:8,nodes:16,roles:36
name (key)                   priority nodes            roles
---------------------------- -------- ---------------- ----------------------------------
Apache274-DataNode           500      node001..node004 Hadoop::DataNode, Hadoop::YARNClient
Apache274-HBaseMaster        500      node004          Hadoop::HBaseServer
Apache274-HBaseRegionServer  500      node001..node003 Hadoop::HBaseClient
Apache274-NameNode           500      node001          Hadoop::NameNode
Apache274-ResourceManager    500      node003          Hadoop::YARNServer
Apache274-SecondaryNN        500      node002          Hadoop::SecondaryNameNode
Apache274-ZooKeeper          500      node001..node003 ZooKeeper::Host
[bright81->configurationoverlay]%
```

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

```
[bright81->configurationoverlay]% use apache274-datanode
[bright81->configurationoverlay[Apache274-DataNode]]% show
Parameter                      Value
------------------------------ ----------------------------------------
```

```
Categories
Customizations                  <0 in submode>
Name                            Apache274-DataNode
Nodes                           node001..node004
Priority                        500
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 are assigned to the configuration overlay can be listed:

Example

```
[bright81->configurationoverlay[Apache274-DataNode]]% roles
[bright81->configurationoverlay[Apache274-DataNode]->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[Apache274-DataNode]->roles]% use hadoop::datanode
...figurationoverlay[Apache274-DataNode]->roles[Hadoop::DataNode]]% show
Parameter                       Value
------------------------------- --------------------------------
Add services                    yes
Configurations                  <1 in submode>
Name                            Hadoop::DataNode
Provisioning associations       <0 internally used>
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

```
[...che274-DataNode]->roles[Hadoop::DataNode]]% configurations
[...che274-DataNode]->roles[Hadoop::DataNode]->configurations]% list
HDFS
------------
Apache274
```

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

```
[...che274-DataNode]->roles[Hadoop::DataNode]->configurations]% use Apache274
[...che274-DataNode]->roles[Hadoop::DataNode]->configurations[Apache274]]% show
Parameter                         Value
--------------------------------- --------------------------------
Bandwidth for balancer            1048576
Data directories                  /var/lib/hadoop/Apache274/datanode
```

```
DataNode Java heap size              512
HDFS                                 Apache274
HTTP port                            50075
HTTPS port                           50475
Handler count                        10
Heartbeat interval                   3
Java Home                            /usr/lib/jvm/jre-1.8.0-openjdk/
Maximum number of transfer threads   4096
Number of failed volumes tolerated   0
Protocol port                        50020
Reserved spaced for non-HDFS use     1073741824
Revision
Transceiver port                     50010
Type                                 HadoopDataNodeHDFSConfiguration
```

The properties available here for the Hadoop::DataNode role correspond to the properties reached from figure 3.10, if the clickpath:

Configuration Overlays→Apache274-DataNode→Edit→Roles→Hadoop::DataNode →Edit→Configurations→Apache274→Edit

is followed.

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

```
[bright81->device]% roleoverview
Role                        Nodes              Configuration Overlays       Nodes up
--------------------------- ------------------ ---------------------------- ---------
Hadoop::DataNode            node001..node004   Apache274-DataNode             4 of 4
Hadoop::HBaseClient         node001..node003   Apache274-HBaseRegionServer    3 of 3
Hadoop::HBaseServer         node004            Apache274-HBaseMaster          1 of 1
Hadoop::NameNode            node001            Apache274-NameNode             1 of 1
Hadoop::SecondaryNameNode   node002            Apache274-SecondaryNN          1 of 1
Hadoop::YARNClient          node001..node004   Apache274-DataNode             4 of 4
Hadoop::YARNServer          node003            Apache274-ResourceManager      1 of 1
ZooKeeper::Host             node001..node003   Apache274-ZooKeeper            3 of 3
boot                        bright81                                          1 of 1
login                       bright81                                          1 of 1
master                      bright81                                          1 of 1
monitoring                  bright81                                          1 of 1
provisioning                bright81                                          1 of 1
storage                     bright81                                          1 of 1
```

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. If it is run with no arguments, then it displays a help page:

Example

```
[root@bright81 ~]# /cm/local/apps/cluster-tools/hadoop/cm-hadoop-maint

Hadoop instance name must be specified. Exiting.
```

```
USAGE: /cm/local/apps/cluster-tools/bin/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 |
  --setJavaHome <path> <tool> | --copyconfig <nodes> | --prepare <nodes> |
  --cleanupYarn | -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                   --  execute a manual failover for HDFS
--failoverstatus             --  return failover status for HDFS
--yarnfailover               --  execute a manual failover for YARN
--yarnfailoverstatus         --  return failover status for YARN
--setJavaHome <path> <tool>  --  update JAVA_HOME for selected tool
--cleanupYarn                --  cleanup YARN localization directories
--copyconfig <nodes>         --  copy 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, zookeeper,
hbase, spark, sqoop, hive, accumulo, drill, flink, kafka, storm
<tool> can be one of the following values: hadoop, zookeeper, hbase,
spark, sqoop, hive, accumulo, drill, flink, kafka, storm

EXAMPLES:
cm-hadoop-maint -i hdfs1 -f
cm-hadoop-maint -i hdfs2 --stop
cm-hadoop-maint -i hdfs2 --stoponly hdfs
cm-hadoop-maint -i hdfsha --failover nn1 nn2
  executes failover from nn1 to nn2
cm-hadoop-maint -i hdfsha --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`: start the balancer daemon
- `-f`: format the Hadoop filesystem and reinitialize it with a standard set of directories, for example: `/user`, `/tmp`.
- `--start`, `--stop`, `--restart`: allow administrators to start, stop, or restart all services relevant to the Hadoop instance.

To operate on only one of the services, 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`: enter or leave NameNode `safe mode`
- `--failover`, `--yarnfailover`: trigger a failover for HDFS, or for YARN
- `--failoverstatus`, `--yarnfailoverstatus`: get the status of High Availability for HDFS or for YARN
- `--setJavaHome` <*path*> <*tool*>: allows administrators to change the value of `JAVA_HOME` after the instance has been deployed. The value should be set on a per-tool basis. The script will update the corresponding environment files and restart the relevant services.
- `--cleanupYarn`: cleans up YARN localization directories
- `--copyconfig` <*nodes*>: Copy 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*>: Prepare 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 `--prepare` option. 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 overlays.

3.4 Hadoop Measurables

Bright Cluster Manager runs health checks to check that the main Hadoop services are running correctly, and collects metrics to help users evaluate the performance of the instance. These measurables can help diagnose possible issues.

3.4.1 Hadoop Metrics

Almost all Hadoop metrics are gathered per node. The metrics collected depend on the Hadoop roles defined for a node—normally defined via Configuration Overlays—and also depend on the Hadoop instance that the node is associated with.

Bright Cluster Manager enables a *metric sink*[1] in all Hadoop deployments.

The metric sink extension makes the Hadoop process itself flush all of its metrics to a spool. The primary metric collection script `sample_hadoop` then collects all the values as specified in the configuration file `sample_hadoop.conf` and as specified by its roles.

In Bright Cluster Manager all the metrics gathered have the original names as documented in the Hadoop documentation at: `https://hadoop.apache.org/docs/stable/`

[1] A *metric sink* is Hadoop terminology. Metrics in Hadoop are described as "statistical information exposed by Hadoop daemons, used for monitoring, performance tuning and debug." In other words, they behave just like Bright Cluster Manager metrics, except that they are Hadoop-focussed.

hadoop-project-dist/hadoop-common/Metrics.html. Within Bright Cluster Manager the metrics are prefixed with the instance name and the process they relate to, for example: Apache274_hbasemaster_SentBytes.

Further details regarding this or the metrics themselves can be found in the Hadoop metrics documentation at the preceding URL.

3.4.2 Collection Scripts

The following metric collections scripts are available for Hadoop:

Collection script	Description
sample_hadoop	Main collection script for all metrics available inside Hadoop services. As described in more detail in the documentation URL.
sample-hdfs-usage	Samples HDFS configured capacity, remaining and used HDFS disk space, unused non-HDFS space.
sample-hdfsadmin-report	Samples the number of available, dead, decommissioned, decommissioning, and normal datanodes.

Example

```
[bright81]% device use node001
[bright81->device[node001]]% latestmonitoringdata
Measurable                          Type                Value          Age
----------------------------------- ------------------- -------------- ----------
...
hdfs1_namenode_MemHeapCommitted     HADOOP_JVM          1022 MiB       2m 25s
hdfs1_namenode_MemHeapUsed          HADOOP_JVM          168 MiB        2m 25s
hdfs1_namenode_MemMax               HADOOP_JVM          1022 MiB       2m 25s
hdfs1_namenode_MemNonHeapCommitted  HADOOP_JVM          132 MiB        2m 25s
hdfs1_namenode_MemNonHeapUsed       HADOOP_JVM          42.2 MiB       2m 25s
...
```

The script sample-hdfsadmin-report is the only one that gathers metrics for the Hadoop instance itself, as these metrics are not related to specific nodes. The metrics for the instance can be viewed from within the bigdata mode of cmsh, and using the Hadoop instance. Running the latestmonitoringdata command for the instance then displays the instance metrics.

Example

```
[mycluster1]% hadoop  use hdfs1
[mycluster1->bigdata[hdfs1]]% latestmonitoringdata
Measurable                                        Type          Value     Age
------------------------------------------------- ------------ --------- ----------
hdfs1_namenode_available_datanodes                HADOOP_DFS    6         3m 4s
hdfs1_namenode_dead_datanodes                     HADOOP_DFS    0         3m 4s
hdfs1_namenode_decommission_in_progress_datanod+ HADOOP_DFS    0         3m 4s
hdfs1_namenode_decommissioned_datanodes           HADOOP_DFS    0         3m 4s
hdfs1_namenode_normal_datanodes                   HADOOP_DFS    6         3m 4s
...
```

3.4.3 Hadoop Healthchecks

The following health checks are defined for each Hadoop instance:

Health Check*	Description
hdfs_ls	Checks if the content of the '/' directory can be listed in Hadoop, by running hdfs as user**: # hdfs dfs -ls / The health check is enabled only on the NameNodes. Returns PASS only if this listing succeeds without any error.
hdfs_namenode	Checks the HDFS filesystem on '/', by running hdfs as user: # hdfs fsck / Returns PASS if HDFS is in a healthy state according to the hdfs fsck command. FAIL implies an unhealthy state caused by, for example: • Corrupt blocks • Under-replicated blocks • Missing replicas.
hdfs_nodecapacity	Checks HDFS disk capacity, by running hdfs as user: # hdfs fsck / -report Checks if the reported capacity is above a certain threshold. By default, this threshold is defined both as a percentage (minimum 10% disk space free) and as an amount in MiB (1000 MiB). The parameters for this health check can be set via cmsh. Returns PASS only if the available disk space remains above the configured threshold.
yarn_nodemanager	Checks YARN Node Managers, by running yarn as user: # yarn node -list Returns PASS only if YARN Node Manager is running, decommissioned, or not used because of Slurm.

* Bright Cluster Manager prefixes the health check with the instance name. For example: hdfs1_hdfs_ls for the instance hdfs1 and health check hdfs_ls

** For Hadoop 1.*x*, the command hadoop fs is used instead of hdfs dfs.

3.5 Big Data Metric And Healthcheck Configuration

3.5.1 Collection Scripts Definition In cmsh

Bright Cluster Manager allows manipulation of the collection scripts in the Monitoring Setup submode. For Hadoop collection scripts intervals can be configured, as well as custom scripts or health check scripts added.

```
[mycluster1]% monitoring
```

```
[mycluster1->monitoring]% setup
[mycluster1->monitoring->setup]% list | head -2 ; list | grep -i Hadoop
Type           Name (key)              Arguments  Measurables  Node execution filters
-------------- ----------------------- ---------- ------------ ----------------------
Collection     HadoopAdminReport                  0 / 229      <1 in submode>
Collection     HadoopHDFS                         0 / 229      <1 in submode>
Collection     HadoopListFileSystem               0 / 229      <1 in submode>
Collection     HadoopNameNode                     0 / 229      <1 in submode>
Collection     HadoopNodeCapacity                 0 / 229      <1 in submode>
Collection     HadoopServices                     0 / 229      <1 in submode>
Collection     HadoopYarnNodeManager              0 / 229      <1 in submode>
[mycluster1->monitoring->setup]% show  hadoopadminreport
Parameter                      Value
------------------------------ ------------------------------------------------------------------
Arguments
Automatic reinitialize         yes
Consolidator
Description                    Samples metrics from HDFS admin report command
Disabled                       no
Execution multiplexer          <0 in submode>
Fuzzy offset                   0
Gap                            0
Interval                       2m
Maximal age                    0s
Maximal samples                4096
Measurables                    0 / 229
Name                           HadoopAdminReport
Node execution filters         <1 in submode>
Notes                          <0 bytes>
Offset                         0s
Only when idle                 no
Revision
Script                         /cm/local/apps/cmd/scripts/metrics/hadoop/sample-hdfsadmin-report
Timeout                        20
Type                           Collection
When                           Timed
```

These collection scripts use node execution filters that restrict the scripts to run on appropriate nodes.

Example

`hadoop-hdfs-namenode` is configured to run on nodes with the `Hadoop::NameNode` role.

Multiple Hadoop instances can spread accross overlapping nodes and the same collection script can process multiple instances in one run.

ZooKeeper, Spark, Cassandra and also the Big Data Tools themselves (Alluxio, HBase, and so on) are sampled using separate scripts that only run on nodes that have the corresponding roles assigned to them.

3.6 Centralized Logging For Hadoop

Troubleshooting Hadoop services is more convenient with all the Hadoop logs centralized.

Centralized logging for Hadoop uses the ELK stack. The acronym ELK comprises:

- ElasticSearch: for storage

- LogStash: for gathering logs
- Kibana: as a front-end.

Installing and enabling the ELK stack for a given Hadoop instance can be carried out as follows:

Installing the ELK stack: Bright Cluster Manager provides an Ncurses setup script, `cm-elk-setup`, that can be run on the headnode.

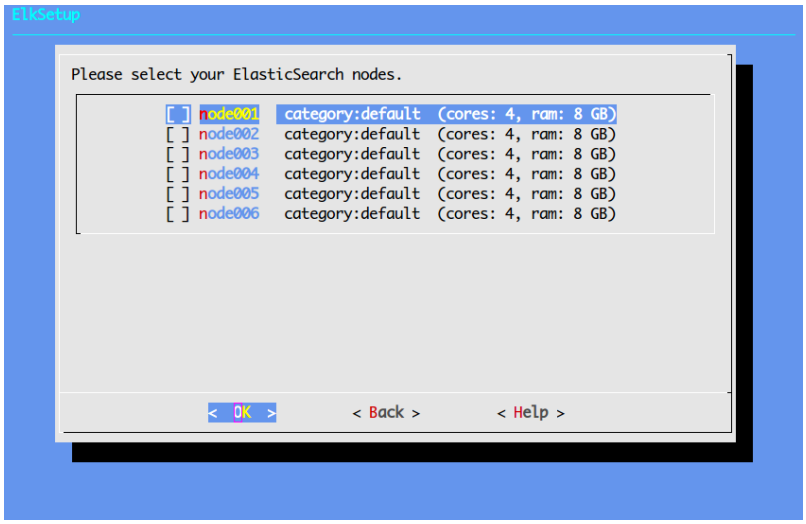

Figure 3.13: The `cm-elk-setup` Ncurses wizard.

This script prompts for ElasticSearch nodes, username and password for Kibana, and so on. A setup configuration file, `cm-elk-setup.conf`, is then written out. It can be used for installation as follows:

```
cm-elk-setup -c /path/to/cm-elk-setup.conf
```

Installation can take about 10 minutes.

Enabling the ELK stack: After ELK stack installation is done, the administrator can enter the `bigdata` mode of `cmsh`, and then enter the `loggingsettings` submode, to run the `enablecentralizedlogging` command. Amongst other things, this sets the `Centralized Logging` parameter to `yes`:

Example

```
[mycluster1]% bigdata use hdfs1
[mycluster1->bigdata[hdfs1]]% loggingsettings
[mycluster1->bigdata[hdfs1]->loggingsettings]% enablecentralizedlogging
Commit configurationoverlay 'ELKLogstashKibana' ... ok.
Commit configurationoverlay 'hdfs1-ZooKeeper' ... ok.
Commit hadoophdfs 'hdfs1' ... ok.
[mycluster1->bigdata[hdfs1]->loggingsettings]% show
Parameter                        Value
-------------------------------- ------------------------------------------------
Base type                        BigDataLoggingSettings
Lookup
Modified                         no
Removed                          no
UID                              281474976723898
```

```
Centralized Logging                yes
...
```

The disablecentralizedlogging command can be run to disable centralized logging.

3.6.1 Enabling Centralized Logging—More detail

The enablecentralizedlogging command actually makes two changes besides just toggling the Centralized logging parameter in the loggingsettings submode. The two changes are indicated by the configurationoverlay output in the preceding example, to inform the administrator that the command also:

- updates the Logstash role configuration inside the ELKLogstashKibana configuration overlay.
- enables centralized logging for ZooKeeper. This is separate from Hadoop.

Logstash changes In Logstash, an *input* generates events, while a *filter* modifies them. The Logstash role changes that take place create the following:

- An input that specifically binds UDP port 4560 on the Logstash server node. The Centralized Logging parameter, which is now set to yes, configures Hadoop processes to have their environment set to submit log messages from log4j as serialized classes to this port 4560 input. The log4j.properties configuration file is also updated.
- A filter is added to do minimum processing on the raw log4j messages. This includes, for example, creating extra meta fields that did not work out of the box like hadoop_priority, hadoop_application and hadoop_service.

The Logstash filter and input can be found in cmsh within the LogStash::Server role. The listeners mode allows inputs to be set, and the filters mode allows filters to be configured:

Example

```
[mycluster1]% configurationoverlay
[mycluster1->configurationoverlay]% use elklogstashkibana
[mycluster1->configurationoverlay[ELKLogStashKibana]]% roles
[mycluster1->configurationoverlay[ELKLogStashKibana]->roles]% list
Name (key)
----------------------------
LogStash::Server
kibana
[mycluster1->configurationoverlay[ELKLogStashKibana]->roles]% use logstash::server
[...[ELKLogStashKibana]->roles[LogStash::Server]]% listeners
[...[ELKLogStashKibana]->roles[LogStash::Server]->listeners]% list
Name (key)
----------------------
hadoop-log4j-listener
rsyslog
[...[ELKLogStashKibana]->roles[LogStash::Server]->listeners]% show hadoop-log4j-listener
Parameter                      Value
------------------------------ ------------------------------------------------
UID                            281474976716770
Custom config                  <90 bytes>
Revision
Type                           LogstashServerCustomListener
name                           hadoop-log4j-listener
[...[ELKLogStashKibana]->roles[LogStash::Server]->listeners]% ..
```

```
[...[ELKLogStashKibana]->roles[LogStash::Server]]% filters
[...[ELKLogStashKibana]->roles[LogStash::Server]->filters]% show hadoop-log4j-filter
Parameter                      Value
------------------------------ ------------------------------------------------
UID                            281474976716771
Custom config                  <393 bytes>
Revision
Type                           LogstashServerCustomFilter
name                           hadoop-log4j-filter
```

The `Custom config` properties contain the actual configuration file contents. These are written by Bright Cluster Manager to the Logstash configuration directory (`/etc/logstash/conf.d`).

Example

```
# ls -al /etc/logstash/conf.d
total 20
drwxrwxr-x 2 root root 265 Mar 13 16:31 .
drwxr-xr-x 3 root root  20 Mar  8 17:35 ..
-rw-r--r-- 1 root root 131 Mar  8 17:37 20-cmdaemon-rsyslog-input-rsyslog.conf
-rw-r--r-- 1 root root 160 Mar 13 16:31 22-cmdaemon-custom-listener-hadoop-log4j-listener.conf
-rw-r--r-- 1 root root 522 Mar  8 17:37 50-cmdaemon-syslog-filter-rsyslog.conf
-rw-r--r-- 1 root root 463 Mar 13 16:31 52-cmdaemon-custom-filter-hadoop-log4j-filter.conf
-rw-r--r-- 1 root root 163 Mar  8 17:37 90-cmdaemon-elastic-output-elastic.conf
```

Restart of services: Bright Cluster Manager writes out all changes to Hadoop environment files and to the `log4j.properties` files. This triggers a restart of the affected services automatically. Enabling or disabling centralized logging therefore results in a restart of all Hadoop services.

3.6.2 Accessing The Centralized Logs

Kibana interface for drill-down: Opening the Kibana web interface at the private network address `http://10.141.255.254:5601/app/kibana` prompts once to create an index. Once the index is created, all Hadoop logs can be accessed via the interface.

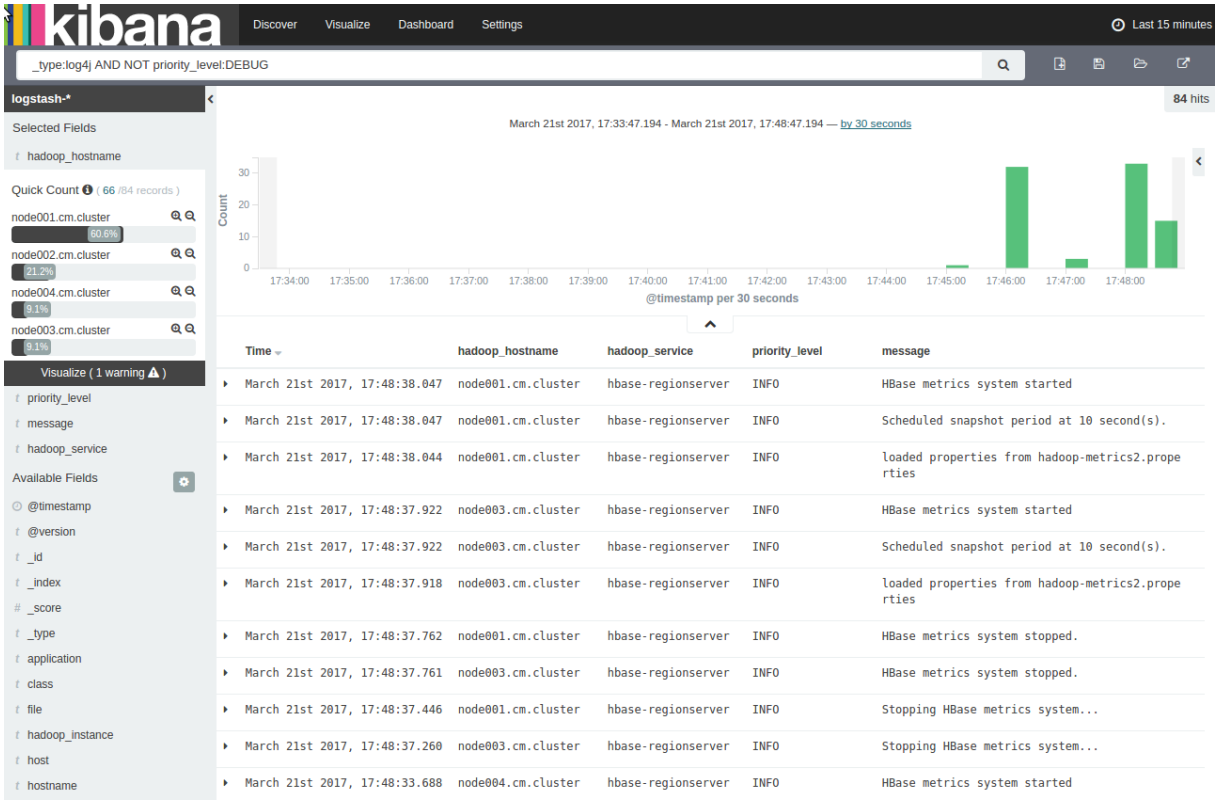

Figure 3.14: The Kibana web interface with Hadoop Centralized Logging enabled.

Tail all Hadoop logs at once: It is also possible to use other tools on the log data that is stored in ElasticSearch. For example, the `elktail` command line utility, available from `https://github.com/knes1/elktail`:

```
trigen@zenbook:~> elktail | nowrap | head -n 10
2017-03-22T13:36:31.461Z - hdfs1 - node001.cm.cluster - INFO - namenode - Triggering log roll on remote NameNode node002.cm.cluster/10
2017-03-22T13:36:31.466Z - hdfs1 - node002.cm.cluster - INFO - namenode - Ending log segment 2868
2017-03-22T13:36:31.466Z - hdfs1 - node002.cm.cluster - INFO - namenode - Roll Edit Log from 10.141.0.1
2017-03-22T13:36:31.466Z - hdfs1 - node002.cm.cluster - INFO - namenode - Rolling edit logs
2017-03-22T13:36:31.467Z - hdfs1 - node002.cm.cluster - INFO - namenode - Number of transactions: 2 Total time for transactions(ms): 0
2017-03-22T13:36:31.486Z - hdfs1 - node002.cm.cluster - INFO - namenode - Number of transactions: 2 Total time for transactions(ms): 0
2017-03-22T13:36:31.493Z - hdfs1 - node002.cm.cluster - INFO - namenode - Starting log segment at 2870
2017-03-22T13:36:31.493Z - hdfs1 - node002.cm.cluster - INFO - namenode - Finalizing edits file /var/lib/hadoop/hdfs1/namenode/current
2017-03-22T13:36:31.572Z - hdfs1 - node001.cm.cluster - INFO - namenode - Reading org.apache.hadoop.hdfs.server.namenode.RedundantEdit
2017-03-22T13:36:31.572Z - hdfs1 - node001.cm.cluster - INFO - namenode - Start loading edits file http://node001.cm.cluster:8480/getJ
```

Figure 3.15: The `elktail` console output.

The preceding example used the following configuration file for `elktail`.

```
"SearchTarget":
  "Url": "http://node001.cm.cluster:9200/",
  "IndexPattern": "logstash-[0-9].*"
,
"QueryDefinition":
  "Terms": [
    "_type:log4j",
    "NOT priority_level:DEBUG"
  ],
  "Format": "%@timestamp - %hadoop_instance - %hadoop_hostname - ...
             ...  %priority_level - %hadoop_service - %message",
```

```
    "TimestampField": "@timestamp"
,
"InitialEntries": 50,
"User": "",
"SSHTunnelParams": ""
```

3.6.3 Centralized Logging Troubleshooting

- It is not trivial to configure Hadoop to also include YARN application logs or YARN job logs in the centralized logging output. These logs therefore still need to be accessed via the usual Hadoop interfaces.
- The index of ElasticSearch can sometimes become incorrect. This can happen, for example, if an instance is uninstalled, then reinstalled with the same name. In that case, the following `curl` command can be run:

  `curl -XDELETE http://<elasticsearch_node_ip>:9200/.kibana`

  and the `logstash` service restarted. The web interface then prompts to re-create the index.

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@bright81 ~]# cd /cm/local/apps/cluster-tools/hadoop/
[root@bright81 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@bright81 hadoop]# ./cm-hadoop-tests.sh Apache274
...
================================================================
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` pane in Bright View (introduced in section 3.1.2) for the Hadoop instance should show activity as it refreshes.

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

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

```
[bright81->user[fred]]% set hadoophdfsaccess Apache274; 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@bright81 ~]$ module add hadoop/Apache274/Apache/2.7.4
[fred@bright81 ~]$ hadoop jar $HADOOP_PREFIX/share/hadoop/hado\
op-mapreduce-examples-2.7.4.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.

5

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*. The tarball is also included in the Bright Cluster Manager repository package `cm-apache-hadoop-extras`.

Apache Spark can be installed in YARN mode, that is on top of an existing Hadoop instance (section 5.2) or in Standalone Mode, that is without Hadoop (section 5.3).

Bright Cluster Manager also provides scripts to install Apache Zeppelin (section 5.4) and Alluxio (section 5.5).

5.1 Spark Installation In Bright Cluster Manager–Overview

5.1.1 Prerequisites For Spark Installation, And What Spark Installation Does

The following applies to installing Spark (section 5.1.2) on Bright Cluster Manager:

- Spark can be installed in two different deployment modes: Standalone or YARN.
- When installing in YARN mode, the script installs Spark only on the active head node.
- YARN mode is not supported for Apache Hadoop 1.*x*, Cloudera CDH 4.*x*, and Hortonworks HDP 1.3.*x*.
- Depending on the installation mode, the script creates one or more dedicated Hadoop Configuration Groups for Spark:
  - Standalone mode: Two configuration overlays are created, one for Spark Master and one for Spark Worker roles.
  - YARN mode: Only one configuration overlay is 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/`
- If installing Spark on a Bright Cluster Manager which has Lustre running on it, and which 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 Utility: `cm-spark-setup`

Bright Cluster Manager provides `cm-spark-setup` to carry out Spark installation. The `cm-spark-setup` utility has the following usage:

```
cm-spark-setup

Hadoop instance name must be specified. Exiting.

  USAGE: /cm/local/apps/cluster-tools/bin/cm-spark-setup [ [-i <name> -t <file>\
 [-j <path>] [-n <node>]] | -c <filename> | -u <name> | --update <name> -t <file> | -h ]

  OPTIONS:
  -i <name>                       --  instance name
  -t <file>                       --  Spark tarball
  -j <path>                       --  Java home path
  -c <filename>                   --  add Spark instance using config file <filename>
  --update <name>                 --  update Spark for instance <name>
  -u <name>                       --  uninstall Spark for instance <name>
  -n <node>                       --  node for Spark YARN role
  -h                              --  show usage


  EXAMPLES:

  Spark installation in YARN mode
  cm-spark-setup -i hdfs1 -t /tmp/spark-2.2.1-bin-hadoop2.7.tgz
  cm-spark-setup -i hdfs1 -t /tmp/spark-2.2.1-bin-hadoop2.7.tgz \
  -j /usr/lib/jvm/jre-1.8.0-openjdk.x86_64/

  Spark installation in Standalone mode:
  cm-spark-setup -c /tmp/spark-conf.xml

  Spark update:
  cm-spark-setup --update hdfs1 -t /tmp/spark-2.2.1-bin-hadoop2.7.tgz

  Spark removal:
  cm-spark-setup -u hdfs1
```

5.2 Spark Installation In YARN Mode

Hadoop 2.*x* is required for Spark in YARN node. Cloudera CDH 4.*x* is not supported for this deployment mode.

Both the options `-i` <*name*> and `-t` <*file*> are required. These are to specify the Hadoop instance name and to specify the Spark tarball file.

The option `-j` <*path*> is not mandatory. It is used to set the Java Home in Spark environment files. If it is not specified, then the script uses the value retrieved from the Hadoop instance.

`cm-spark-setup` installs Spark by default on the active head node. A different node can be specified by using the option `-n`.

Example

```
[root@bright81 ~]# cm-spark-setup -i Apache274 \
 -t /cm/local/apps/hadoop/spark-2.2.0-bin-hadoop2.7.tgz
Java home not specified, using: /usr/lib/jvm/jre-1.8.0-openjdk/
Spark release '2.2.0-bin-hadoop2.7'
Found Hadoop instance 'Apache274', release: 2.7.4
Spark will be installed in YARN (client/cluster) mode.
Spark being installed... done.
Creating module file for Spark... done.
Creating configuration files for Spark... done.
Updating images... done.
Waiting for NameNode to be ready... done.
Initializing Spark YARN role... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Validating Spark setup...
-- testing '--master yarn --deploy-mode client' mode...
-- testing '--master yarn --deploy-mode cluster' mode...
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

5.2.1 Using Spark In YARN Mode (Spark 2.x)

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

- `client`
- `cluster`

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

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

   Example

   ```
   [root@bright81 ~]# module load spark/Apache274/Apache/2.2.0-bin-hadoop2.7

   [root@bright81 ~]# spark-submit --master yarn --deploy-mode client \
    --class org.apache.spark.examples.SparkPi \
    $SPARK_PREFIX/examples/jars/spark-examples_*.jar
   ```

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

Example

```
[root@bright81 ~]# module load spark/hdfs1
[root@bright81 ~]# spark-submit --master yarn --deploy-mode cluster \
 --class org.apache.spark.examples.SparkPi \
 $SPARK_PREFIX/examples/jars/spark-examples_*.jar
```

Spark has an interactive Scala shell and an interactive Python shell. They can be started up as follows:

Scala shell example:

Example

```
[root@bright81 ~]# module load spark/Apache274/Apache/2.2.0-bin-hadoop2.7
[root@bright81 ~]# spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/10/20 22:03:27 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falli\
ng back to uploading libraries under SPARK_HOME.
17/10/20 22:03:53 WARN ObjectStore: Version information not found in metastore. hive.metast\
ore.schema.verification is not enabled so recording the schema version 1.2.0
17/10/20 22:03:53 WARN ObjectStore: Failed to get database default, returning NoSuchObjectE\
xception
17/10/20 22:03:54 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObj\
ectException
Spark context Web UI available at http://10.141.255.254:4040
Spark context available as 'sc' (master = yarn, app id = application_1508528548591_0005).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_144)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Python shell example:

Example

```
[root@bright81 ~]# module load spark/hdfs1
[root@bright81 ~]# pyspark
Python 2.7.5 (default, Nov 20 2015, 02:00:19)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/10/20 22:11:09 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falli\
ng back to uploading libraries under SPARK_HOME.
17/10/20 22:11:32 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObj\
ectException
```

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Python version 2.7.5 (default, Nov 20 2015 02:00:19)
SparkSession available as 'spark'.
>>>
```

5.2.2 Using Spark In YARN Mode (Spark 1.x)

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@bright81 ~]# module load spark/Apache274/Apache/2.2.0-bin-hadoop2.7
[root@bright81 ~]# 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@bright81 ~]# module load spark/Apache274/Apache/2.2.0-bin-hadoop2.7
[root@bright81 ~]# spark-submit --master yarn-cluster \
 --class org.apache.spark.examples.SparkPi \
 $SPARK_PREFIX/examples/jars/spark-examples_2.11-2.2.0.jar
```

Spark has an interactive Scala shell and an interactive Python shell. They can be started up as follows:

Scala shell example

Example

```
[root@bright81 ~]# module load spark/Apache274/Apache/2.2.0-bin-hadoop2.7
[root@bright81 ~]# spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/10/20 22:24:53 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falli\
ng back to uploading libraries under SPARK_HOME.
17/10/20 22:25:11 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObj\
ectException
Spark context Web UI available at http://10.141.255.254:4040
Spark context available as 'sc' (master = yarn, app id = application_1508528548591_0008).
Spark session available as 'spark'.
Welcome to
      ____              __
```

```
      / __/__  ___ _____/ /__
     _\ \/ _ \/ _ `/ __/  '_/
    /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
       /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_144)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Python shell example:

Example

```
[root@bright81 ~]# module load spark/Apache274/Apache/2.2.0-bin-hadoop2.7
[root@bright81 ~]# pyspark --master yarn-client
Python 2.7.5 (default, Nov 20 2015, 02:00:19)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Warning: Master yarn-client is deprecated since 2.0. Please use master "yarn" with specifie\
d deploy mode instead.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/10/20 22:30:21 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falli\
ng back to uploading libraries under SPARK_HOME.
17/10/20 22:30:47 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObj\
ectException
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Python version 2.7.5 (default, Nov 20 2015 02:00:19)
SparkSession available as 'spark'.
>>>
```

5.2.3 Spark Removal With `cm-spark-setup`

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

Example

```
[root@bright81 ~]# cm-spark-setup -u hdfs1
Undoing Jupyter, JupyterHub and Toree integration... done.
Requested removal of Spark for Hadoop instance 'Apache274'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Cleaning ZooKeeper... done.
Removing additional Spark directories... done.
Removing Spark-related metrics... done.
Removal successfully completed.
Finished.
```

5.3 Spark Installation In Standalone Mode

Spark installation in Standalone mode is also carried out by `cm-spark-setup`, which takes an XML configuration file with the `-c` option. If the Bright Cluster Manager package is used, then the XML file is placed at `/cm/local/apps/cluster-tools/hadoop/conf/sparkconf.xml`. The configuration file has three main sections:

- `<masters>`: for Spark Master configuration
- `<workers>`: for Spark Workers configuration
- `<zookeeper>`: for ZooKeeper configuration, if Zookeeper is used for Spark Master HA

More information on Spark installation configuration can be found at `http://spark.apache.org/docs/latest/`.

An example XML file follows:

Example

```
<sparkConfig>
  <archive>/cm/local/apps/hadoop/spark-2.0.0-bin-hadoop2.7.tgz</archive>
  <javahome>/usr/lib/jvm/jre-1.7.0-openjdk.x86_64/</javahome>
  <instance>
    <name>spark1</name>
    <masters recovery="ZOOKEEPER">
      <hosts>node001</hosts>
      <port>7070</port>
      <webport>9080</webport>
      <historywebport>19080</historywebport>
      <localdir>/tmp/spark/spark1/</localdir>
    </masters>
    <workers>
      <hosts>node00[1..5]</hosts>
      <port>7071</port>
      <webport>9081</webport>
      <numcores>0</numcores> <!-- automatic -->
      <numinstances>1</numinstances>
      <workerdir>/tmp/spark/spark1/</workerdir>
      <cleanupenabled>true</cleanupenabled>
      <cleanupinterval>1800</cleanupinterval>
      <cleanupappdatattl>604800</cleanupappdatattl>
    </workers>
    <zookeeper>
      <archive>/cm/local/apps/hadoop/zookeeper-3.4.9.tar.gz</archive>
      <hosts>node00[1..3]</hosts>
      <clientport>12181</clientport>
      <serverport>12888</serverport>
      <electionport>13888</electionport>
    </zookeeper>
  </instance>
</sparkConfig>
```

Regarding the tags in the configuration file:

- The attribute `recovery` can assume three different values: `NONE`, `FILESYSTEM` or `ZOOKEEPER`. The value is set for the Spark property `spark.deploy.recoveryMode`. If `ZOOKEEPER` is chosen, then the section with tag `<zookeeper>` is mandatory.

- The value for the tag <localdir> sets the environment variable SPARK_LOCAL_DIRS
- The value for the tag <numcores> sets the environment variable SPARK_WORKER_CORES. The default value 0 means automatic
- The value for the tag <numinstances> sets the environment variable SPARK_WORKER_INSTANCES
- The value for tag <workerdir> sets the environment variable SPARK_WORKER_DIR
- The cleanup* tags have values that work as follows:
  - <cleanupenabled> enables the periodic cleanup of worker/application directories. This corresponds to the property spark.worker.cleanup.enabled.
  - <cleanupinterval> sets the interval, in seconds, for cleaning up the worker directory. This corresponds to the property spark.worker.cleanup.interval
  - <cleanupappdatattl> sets the number of seconds to retain application worker directories. This corresponds to spark.worker.cleanup.appDataTtl.

An installation with a properly set up XML configuration file should result in a session similar to the following:

Example

```
[root@bright81 ~]# cm-spark-setup -c /root/sparkconf.xml
Reading config from file '/root/sparkconf.xml'... done.
Spark release '2.0.0-bin-hadoop2.7'
Creating Spark instance 'spark1'... done.
Spark will be installed in Standalone mode with ZooKeeper.
Spark Master service will be run on: node001
Spark Worker service will be run on: node001,node002,node003,node004,node005
ZooKeeper will be run on: node001,node002,node003
Spark being installed... done.
ZooKeeper being installed... done.
Creating directories for Spark... done.
Creating module file for Spark... done.
Creating module file for ZooKeeper... done.
Creating configuration files for Spark... done.
Creating configuration files for ZooKeeper... done.
Updating images... done.
Initializing ZooKeeper service... done.
Initializing Spark Master service... done.
Initializing Spark Worker services... done.
Updating configuration in CMDaemon... done.
Validating ZooKeeper setup... done.
Validating Spark setup...
-- testing Python application...
-- testing R application...
-- testing Java application...
-- testing Scala application...
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

5.3.1 Deploy Spark On A Specific Network

By default Spark is configured to use the management network `internalnet`, as defined in the base partition. As a consequence, CMDaemon writes out the Spark configuration using the IP addresses of that network.

When multiple internal networks are available, users would want to specify a different internal network for Spark traffic. It is possible to do that by using the tag `<network>`, as shown in the following example:

Example

```
<sparkConfig>
  ...
  <instance>
    <name>spark1</name>
    <network>sparknet</network>
    <masters recovery="ZOOKEEPER">
    ...
</sparkConfig>
```

The Spark instance is configured by this to use the selected network, and Spark services will bind to the correct IP as defined by `SPARK_LOCAL_IP` in `spark-env.sh`.

Once the Spark instance has been installed, the selected network cannot be changed, as indicated in `cmsh`:

```
[bright81->bigdata]% show spark1 | grep Network
Network                                          sparknet
```

5.3.2 Using Spark In Standalone Mode

Spark can run applications written in different languages, such as Java, Python, R, and Scala:

Example

Calculating Pi in Java

```
[root@bright81 ~]# module load spark/spark1
[root@bright81 ~]# run-example JavaSparkPi
16/09/15 11:00:19 INFO SparkContext: Running Spark version 2.0.0

...

16/09/15 11:00:27 INFO DAGScheduler: Job 0 finished: reduce at JavaSparkPi.java:52,\
 took 3.503070 s
Pi is roughly 3.14536
16/09/15 11:00:27 INFO SparkUI: Stopped Spark web UI at http://10.141.255.254:4040

...

16/09/15 11:00:27 INFO SparkContext: Successfully stopped SparkContext
16/09/15 11:00:27 INFO ShutdownHookManager: Shutdown hook called
16/09/15 11:00:27 INFO ShutdownHookManager: Deleting directory /tmp/spark-3362c57d-\
b560-4d4d-91d1-c0b608a60e4e
```

Example

Calculating Pi in Python

```
[root@bright81 ~]# module load spark/spark1
[root@bright81 ~]# spark-submit $SPARK_PREFIX/examples/src/main/python/pi.py
16/09/15 11:01:14 INFO SparkContext: Running Spark version 2.0.0

...

16/09/15 11:01:21 INFO DAGScheduler: Job 0 finished: reduce at /cm/shared/apps/hado\
op/Apache/spark-2.0.0-bin-hadoop2.7/examples/src/main/python/pi.py:43, took 3.04015\
3 s
Pi is roughly 3.143780
16/09/15 11:01:21 INFO SparkUI: Stopped Spark web UI at http://10.141.255.254:4040

...

16/09/15 11:01:22 INFO ShutdownHookManager: Shutdown hook called
16/09/15 11:01:22 INFO ShutdownHookManager: Deleting directory /tmp/spark-dc8b2f83-\
7b45-40a0-8c2c-475a8a21d13a
16/09/15 11:01:22 INFO ShutdownHookManager: Deleting directory /tmp/spark-dc8b2f83-\
7b45-40a0-8c2c-475a8a21d13a/pyspark-a7f16fac-e1ca-4f59-a9c5-24b37d57468b
```

Example

Doing some operations on DataFrames in R

```
[root@bright81 ~]# module load spark/spark1
[root@bright81 ~]# spark-submit $SPARK_PREFIX/examples/src/main/r/dataframe.R
Loading required package: methods

Attaching package: 'SparkR'

The following objects are masked from 'package:stats':

    cov, filter, lag, na.omit, predict, sd, var, window

The following objects are masked from 'package:base':

    as.data.frame, colnames, colnames<-, drop, endsWith, intersect,
    rank, rbind, sample, startsWith, subset, summary, transform, union

16/09/15 11:02:12 INFO SparkContext: Running Spark version 2.0.0

...

16/09/15 11:02:26 INFO HiveClientImpl: Warehouse location for Hive client (version \
1.2.1) is file:/root/spark-warehouse
16/09/15 11:02:27 INFO HiveMetaStore: 0: create_database: Database(name:default, de\
scription:default database, locationUri:file:/root/spark-warehouse, parameters:)
16/09/15 11:02:27 INFO audit: ugi=root ip=unknown-ip-addr cmd=create_database: Data\
base(name:default, description:default database, locationUri:file:/root/spark-wareh\
ouse, parameters:)
root
 |-- name: string (nullable = true)
 |-- age: double (nullable = true)
16/09/15 11:02:27 INFO MemoryStore: Block broadcast_1 stored as values in memory (e\
stimated size 202.1 KB, free 366.1 MB)
16/09/15 11:02:27 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in mem\
```

```
ory (estimated size 23.8 KB, free 366.1 MB)

...

16/09/15 11:02:31 INFO DAGScheduler: ResultStage 1 (json at NativeMethodAccessorImp\
l.java:-2) finished in 3.629 s
16/09/15 11:02:31 INFO DAGScheduler: Job 1 finished: json at NativeMethodAccessorIm\
pl.java:-2, took 3.648738 s
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
16/09/15 11:02:31 INFO SparkSqlParser: Parsing command: people
16/09/15 11:02:31 INFO SparkSqlParser: Parsing command: SELECT name FROM people WHE\
RE age >= 13 AND age <= 19

...

16/09/15 11:02:36 INFO DAGScheduler: ResultStage 2 (dfToCols at NativeMethodAccesso\
rImpl.java:-2) finished in 3.780 s
16/09/15 11:02:36 INFO DAGScheduler: Job 2 finished: dfToCols at NativeMethodAccess\
orImpl.java:-2, took 3.798385 s
16/09/15 11:02:36 INFO CodeGenerator: Code generated in 19.25368 ms
    name
1 Justin
16/09/15 11:02:36 INFO SparkUI: Stopped Spark web UI at http://10.141.255.254:4040

...

16/09/15 11:02:36 INFO SparkContext: Successfully stopped SparkContext
16/09/15 11:02:41 INFO ShutdownHookManager: Shutdown hook called
16/09/15 11:02:41 INFO ShutdownHookManager: Deleting directory /tmp/spark-0959626e-\
eba1-4967-bb81-d80472a64347
```

Example

Calculating Pi in Scala

```
[root@bright81 ~]# module load spark/spark1
[root@bright81 ~]# run-example SparkPi
16/09/15 11:03:21 INFO SparkContext: Running Spark version 2.0.0

...

16/09/15 11:03:27 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, to\
ok 2.704767 s
Pi is roughly 3.1429757148785744
16/09/15 11:03:27 INFO SparkUI: Stopped Spark web UI at http://10.141.255.254:4040

...

16/09/15 11:03:27 INFO SparkContext: Successfully stopped SparkContext
16/09/15 11:03:27 INFO ShutdownHookManager: Shutdown hook called
16/09/15 11:03:27 INFO ShutdownHookManager: Deleting directory /tmp/spark-43047c9a-\
b9ba-437d-b0c9-89f44550fc06
```

Example

Doing a wordcount on a file stored in HDFS

```
[root@bright81 ~]# su - foobar
[foobar@bright81 ~]$ module load hadoop
[foobar@bright81 ~]$ hdfs dfs -copyFromLocal /tmp/hamlet.txt /user/foobar/hamlet.txt
[foobar@bright81 ~]$ module load spark/spark1
[foobar@bright81 ~]$ spark-submit $SPARK_PREFIX/examples/src/main/python/wordcount.py \
 hdfs://node001:8020/user/foobar/hamlet.txt
16/09/16 15:39:09 INFO SparkContext: Running Spark version 2.0.0

...

16/09/16 15:39:28 INFO DAGScheduler: ResultStage 1 (collect at /cm/shared/apps/hado\
op/Apache/spark-2.0.0-bin-hadoop2.7/examples/src/main/python/wordcount.py:40) finis\
hed in 0.113 s
16/09/16 15:39:28 INFO DAGScheduler: Job 0 finished: collect at /cm/shared/apps/had\
oop/Apache/spark-2.0.0-bin-hadoop2.7/examples/src/main/python/wordcount.py:40, took\
 4.703843 s
: 1285
pardon: 1
cheefe: 1
better.: 1

...

goodly: 1
Sits: 1
indirectly: 1
16/09/16 15:39:28 INFO SparkUI: Stopped Spark web UI at http://10.141.255.254:4040
16/09/16 15:39:28 INFO StandaloneSchedulerBackend: Shutting down all executors
16/09/16 15:39:28 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each ex\
ecutor to shut down
16/09/16 15:39:28 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoi\
nt stopped!
16/09/16 15:39:28 INFO MemoryStore: MemoryStore cleared
16/09/16 15:39:28 INFO BlockManager: BlockManager stopped
16/09/16 15:39:28 INFO BlockManagerMaster: BlockManagerMaster stopped
16/09/16 15:39:28 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: Out\
putCommitCoordinator stopped!
16/09/16 15:39:28 INFO SparkContext: Successfully stopped SparkContext
16/09/16 15:39:29 INFO ShutdownHookManager: Shutdown hook called
16/09/16 15:39:29 INFO ShutdownHookManager: Deleting directory /tmp/spark/spark1/sp\
ark-4d8c1021-3d7f-4f96-8490-c1592dc2d252
16/09/16 15:39:29 INFO ShutdownHookManager: Deleting directory /tmp/spark/spark1/sp\
ark-4d8c1021-3d7f-4f96-8490-c1592dc2d252/pyspark-df311525-870e-404d-ae7d-b262b69534\
b4
```

5.4 Zeppelin Installation

Apache Zeppelin is a web-based notebook that enables interactive data analytics. Zeppelin can be installed when Spark has already been deployed in YARN mode (section 5.2) or in Standalone mode (section 5.3).

The Apache Zeppelin tarball should be downloaded from `http://zeppelin.apache.org/`. The following table shows the compatibility matrix between Spark and Zeppelin versions.

Table 5.4: Spark And Zeppelin Compatibility Matrix

Spark	Zeppelin 0.6.0	Zeppelin 0.6.1	Zeppelin 0.6.2	Zeppelin 0.7.x
Spark 1.6.*x*	✓	✗	✓	✓
Spark 2.0.*x*	✗	✓	✓	✓
Spark 2.1.*x*	✗	✗	✗	✓
Spark 2.2.*x*	✗	✗	✗	✓

5.4.1 Zeppelin Installation With `cmhadoop-zeppelin-setup`

Bright Cluster Manager provides `cmhadoop-zeppelin-setup` to install Zeppelin.

Prerequisites For Zeppelin Installation, And What Zeppelin Installation Does

The following applies to using `cmhadoop-zeppelin-setup`:

- Spark should already be installed
- The `cmhadoop-zeppelin-setup` script installs a Zeppelin service by default on the active headnode. A different host can be specified by using the option `--host`
- The script creates a dedicated configuration overlay for Zeppelin
- Zeppelin is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Zeppelin configuration files are copied by the script to the directory `/etc/hadoop/`
- The Zeppelin web application runs by default on port 18090 of its host.

An Example Run With `cmhadoop-zeppelin-setup`

The option

`-j` *<path>*

is mandatory. It is used to set the Java Home in Zeppelin environment files. The path should point to a Java Development Kit.

Example

```
[root@bright81 ~]# cmhadoop-zeppelin-setup -i spark1 -j /usr/lib/jvm/java-1.7.0-\
openjdk/ -t /root/zeppelin-0.6.2-bin-all.tgz
Zeppelin release '0.6.2-bin-all'
Zeppelin service will be run on the head node.
Found Hadoop instance 'spark1', release: 2.0.0-bin-hadoop2.7
Spark Master found.
Zeppelin being installed... done.
Creating directories for Zeppelin... done.
Creating module file for Zeppelin... done.
Creating configuration files for Zeppelin... done.
Updating configuration in CMDaemon... done.
Installation successfully completed.
Finished.
```

5.4.2 Zeppelin Removal With `cmhadoop-zeppelin-setup`

`cmhadoop-zeppelin-setup` uses the `-u` option to uninstall the Zeppelin instance.

Example

```
[root@bright81 ~]# cmhadoop-zeppelin-setup -u spark1
Requested removal of Zeppelin for Hadoop instance 'spark1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Zeppelin directories... done.
Removal successfully completed.
Finished.
```

5.5 Alluxio Installation

Alluxio (formerly known as Tachyon) is a memory-centric distributed storage system. It lies between computation frameworks and various storage systems, in order to enable reliable data sharing across cluster jobs.

An Alluxio instance includes the following main components:

- Master, which is primarily responsible for managing the global metadata of the system
- Workers, to manage local resources
- Client, to interact with the Alluxio servers

The Alluxio tarball for Spark should be downloaded from `http://alluxio.org/`. The Alluxio version to be downloaded depends on the Spark version; a compatibility table is available at `http://alluxio.org/docs/master/en/Running-Spark-on-Alluxio.html`.

5.5.1 Alluxio Installation With `cmhadoop-alluxio-setup`

Bright Cluster Manager provides `cmhadoop-alluxio-setup` to carry out Alluxio installation:

Prerequisites For Alluxio Installation, And What Alluxio Installation Does

The following applies to using `cmhadoop-alluxio-setup` to install Alluxio on Spark:

- A Spark instance must already be installed
- `cmhadoop-alluxio-setup` installs Alluxio only on the active head node and on the Spark Worker nodes of the chosen Spark instance
- The script sets the storage layer address to a subdirectory of the Alluxio installation directory
- The script creates two dedicated configuration overlays for Alluxio: one for Alluxio Master and one for Alluxio Workers
- Alluxio is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Alluxio configuration files are copied by the script to under `/etc/hadoop/`

The options for `cmhadoop-alluxio-setup` are listed on running `cmhadoop-alluxio-setup -h`.

An Example Run With `cmhadoop-alluxio-setup` for Alluxio versions older than 1.5.0

The option

`-j` *<path>*

is not mandatory. It is used to set the Java Home in Alluxio environment files. If the option is not specified, then the script will use the value retrieved from the Spark instance.

The option

`-sparkJar` *<path>*

is mandatory. It is used to specify the client jar file for Spark.

Example

```
[root@bright81 ~]# cmhadoop-alluxio-setup -i spark1 -t /tmp/alluxio-1.4.0\
-hadoop2.7-bin.tar.gz --sparkJar /tmp/alluxio-1.4.0-spark-client-jar-with-\
dependencies.jar
Java home not specified, using: /usr/lib/jvm/jre-1.8.0-openjdk.x86_64/
Alluxio release '1.4.0-hadoop2.7-bin'
Found Spark instance 'spark1', release: 2.1.1-bin-hadoop2.7
Alluxio Master will be run on the head node.
Alluxio being installed... done.
Creating directories for Alluxio... done.
Creating module file for Alluxio... done.
Creating configuration files for Alluxio... done.
Updating images... done.
Formatting Alluxio FS... done.
Initializing Alluxio Master service... done.
Initializing Alluxio Worker services... done.
Updating configuration in CMDaemon... done.
Validating Alluxio setup... done.
Installation successfully completed.
Finished.
```

An Example Run With `cmhadoop-alluxio-setup` For Alluxio Versions Newer Than 1.4.0

The option

`-j` *<path>*

is not mandatory. It is used to set the Java Home in Alluxio environment files. If the option is not specified, then the script will use the value retrieved from the Spark instance.

Example

```
[root@bright81 ~]# cmhadoop-alluxio-setup -i spark1 -t /tmp/alluxio-1.5.0\
-hadoop-2.8-bin.tar.gz
Java home not specified, using: /usr/lib/jvm/jre-1.8.0-openjdk.x86_64/
Alluxio release '1.5.0-hadoop-2.8-bin'
Found Spark instance 'spark1', release: 2.1.1-bin-hadoop2.7
Alluxio Master will be run on the head node.
Alluxio being installed... done.
Creating directories for Alluxio... done.
Creating module file for Alluxio... done.
Creating configuration files for Alluxio... done.
Updating images... done.
Formatting Alluxio FS... done.
Initializing Alluxio Master service... done.
Initializing Alluxio Worker services... done.
Updating configuration in CMDaemon... done.
Validating Alluxio setup... done.
Installation successfully completed.
Finished.
```

5.5.2 Alluxio Removal With `cmhadoop-alluxio-setup`

`cmhadoop-alluxio-setup` uses the `-u` option to uninstall the Alluxio instance.

Example

```
[root@bright81 ~]# cmhadoop-alluxio-setup -u spark1
Requested removal of Alluxio for Hadoop instance 'spark1'.
```

```
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Alluxio directories... done.
Removing Alluxio-related metrics... done.
Removal successfully completed.
Finished.
```

5.5.3 Using Alluxio

Alluxio consists of several components: one Master, multiple Workers, and an executable client, `alluxio`, that can be run after the user loads the corresponding module. The following example shows how to execute a Spark word count example on data copied to the Alluxio storage layer.

Example

```
[root@bright81 ~]# module load alluxio/spark1/Apache/1.5.0-hadoop-2.8-bin
[root@bright81 ~]# su -c "alluxio fs chmod 777 /wordcount" alluxio
Changed permission of /wordcount to 777
[root@bright81 ~]# alluxio fs copyFromLocal /cm/local/examples/hamlet.txt /wordcount/input.txt
Copied file:///cm/local/examples/hamlet.txt to /wordcount/input.txt
[root@bright81 ~]# module load spark/spark1/Apache/2.1.1-bin-hadoop2.7
[root@bright81 ~]# spark-submit --jars /cm/shared/apps/hadoop/Apache/alluxio-1.5.0\
-hadoop-2.8-bin//core/client/runtime/target/alluxio-core-client-runtime-1.5.0-jar-\
with-dependencies.jar /cm/local/apps/cluster-tools/hadoop/conf/alluxio/test.py\
 alluxio://ml-bigdatadev.cm.cluster:19998/wordcount/input.txt\
 alluxio://ml-bigdatadev.cm.cluster:19998/wordcount/output
17/06/23 15:08:18 INFO SparkContext: Running Spark version 2.1.1

...

17/06/23 15:08:22 INFO DAGScheduler: Submitting ShuffleMapStage 0 (PairwiseRDD[3] \
at reduceByKey at /cm/local/apps/cluster-tools/hadoop/conf/alluxio/test.py:23), wh\
ich has no missing parents
17/06/23 15:08:22 INFO MemoryStore: Block broadcast_1 stored as values in memory (\
estimated size 9.6 KB, free 366.0 MB)
17/06/23 15:08:22 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in me\
mory (estimated size 5.9 KB, free 366.0 MB)

...

17/06/23 15:08:28 INFO SparkContext: Successfully stopped SparkContext
17/06/23 15:08:29 INFO ShutdownHookManager: Shutdown hook called
17/06/23 15:08:29 INFO ShutdownHookManager: Deleting directory /tmp/spark/spark1/s\
park-84cd0e22-54fc-4fcc-939e-13bff2590630
17/06/23 15:08:29 INFO ShutdownHookManager: Deleting directory /tmp/spark/spark1/s\
park-84cd0e22-54fc-4fcc-939e-13bff2590630/pyspark-a3d1ecc7-573d-4e30-9df1-d755b643\
9e3e
[root@bright81 ~]# alluxio fs cat /wordcount/output/part*
(u'', 167)
(u'fardels', 1)
(u'flesh', 1)

...

(u'dread', 1)
```

```
(u'the', 14)
(u'resolution', 1)
```

5.6 Spark On Mesos

If Spark is deployed in Standalone Mode, then it can make use of a pre-existing Mesos deployment to run Spark jobs. In order to use Mesos from Spark, users can connect Spark with the `cm-spark-maint` utility (section 5.7). Using `cm-spark-maint`, users can enable Dynamic Resource Allocation on Mesos. The Marathon framework needs to be installed alongside Mesos, since the script creates a dedicated Marathon application for the Mesos External Shuffle Service.

After Spark is connected to Mesos, the Spark module file contains a few more environment variables. The most important ones are:

- `MESOS_MASTER`: This allows users to submit their Spark application to the Spark master, or to Mesos, in "client" mode
- `MESOS_DISPATCHER`: This allows users to submit their Spark application to the Spark master, or to Mesos, in "cluster" mode

Example

Submitting a Spark application to Spark master

```
[root@bright81 ~]# module load spark
[root@bright81 ~]# spark-submit run-example --master $MASTER SparkPi
```

Example

Submitting a Spark application to Mesos (client mode)

```
[root@bright81 ~]# module load spark
[root@bright81 ~]# spark-submit run-example --master $MESOS_MASTER SparkPi
```

For client mode, the SparkPi application is available in the Mesos UI, listed among the "Active Frameworks".

Example

Submitting a Spark application to Mesos (cluster mode)

```
[root@bright81 ~]# module load spark
[root@bright81 ~]# spark-submit --master $MESOS_DISPATCHER --deploy-mode cluster \
 --class org.apache.spark.examples.SparkPi $SPARK_PREFIX/examples/jars/spark-examples_*.jar
```

For cluster mode, the SparkPi application is available in the Mesos UI, by first navigating to the framework with name "Spark Cluster" and then checking the corresponding task.

5.7 Spark Maintenance Operations With `cm-spark-maint`

The Spark maintenance script, `cm-spark-maint`, is a Python script. It is called using the full path. If it is run with no arguments, then it displays a help page:

Example

```
[root@bright81 ~]# /cm/local/apps/cluster-tools/hadoop/cm-spark-maint

Spark instance name must be specified. Exiting.
```

```
  USAGE: /cm/local/apps/cluster-tools/hadoop/cm-spark-maint -i <name> \
[ --enable-dynalloc | --disable-dynalloc | --connect-mesos <mesos> | --disconnect-mesos \
  --cleanup-history | -h ]

  OPTIONS:
  -i <name>                     -- instance name
  --enable-dynalloc             -- enables Dynamic Resource Allocation
  --disable-dynalloc            -- disables Dynamic Resource Allocation
  --connect-mesos <mesos>       -- connects Spark Standalone to Mesos
  --disconnect-mesos            -- disconnects Spark Standalone from Mesos
  --cleanup-history             -- removes Spark History files
  -h                            -- show usage

  EXAMPLES:
  cm-spark-maint -i hdfs1 --enable-dynalloc
  cm-spark-maint -i spark1 --disable-dynalloc
  cm-spark-maint -i spark1 --connect-mesos default
```

The name of the instance, specified with `-i`, is mandatory. For Spark on YARN, it takes the name of the Hadoop instance, while for Spark Standalone it is the name of the Spark instance itself. The options perform idempotent (harmlessly repeatable) operations, which are explained next in more detail:

- `--enable-dynalloc`: enables Dynamic Resource Allocation. Depending on the Spark configuration, there are 2 scenarios:

  1. Spark on YARN: the script adds the Spark YARN shuffle jar file to $CLASSPATH for the NodeManagers, and adds the `spark_shuffle` service to the list of auxiliary services. It also sets the Spark properties `spark.shuffle.service.enabled` and `spark.dynamicAllocation.enabled` to `true`.
  2. Spark Standalone: the script sets the Spark properties `spark.shuffle.service.enabled` and `spark.dynamicAllocation.enabled` to `true`. If Spark is connected to Mesos, and the Marathon framework is also installed alongside Mesos, then the script deploys an application via Marathon on each Mesos Slave node, to run the Mesos External Shuffle Service. If the Spark instance name is, for example, `spark1`, then the application name becomes: `spark-spark1-mesos-external-shuffle`.

  If Dynamic Resource Allocation has already been enabled, then the script does not perform any changes (the "idempotent" aspect).

- `--disable-dynalloc`: disables Dynamic Resource Allocation. The script sets the Spark properties `spark.shuffle.service.enabled` and `spark.dynamicAllocation.enabled` to `false`. If Spark is connected to Mesos, then the script deletes the Marathon application dedicated to run the Mesos External Shuffle Service.

  If Dynamic Resource Allocation is already disabled, then the script does not perform any changes.

- `--connect-mesos`: connects the Spark Standalone instance to the Mesos instance specified for `<mesos>`. CMDaemon sets the environment variables `MESOS_MASTER` and `MESOS_DISPATCHER`. The environment variables are made available by loading the Spark module, with for example `module load spark2`. CMDaemon also starts the Spark Mesos Dispatcher service on the nodes where the Spark Master is already running. To enable the Dynamic Resource Allocation, the users must re-run the script with the `--enable-dynalloc` option.

- `--disconnect-mesos`: disconnects the Spark Standalone instance from the Mesos instance that it is connected to. This makes `MESOS_MASTER` and `MESOS_DISPATCHER` unavailable. CMDaemon also stops the Spark Mesos Dispatcher service.

If Dynamic Resource Allocation has already been enabled, then the script also deletes the Marathon application dedicated to run the Mesos External Shuffle Service.

- `--cleanup-history`: removes the Spark History Server event files, for when Spark was deployed in Standalone Mode. Example: for Spark version 2.2.0, for an instance with the name `spark1`, the contents of the directory `/cm/shared/apps/hadoop/Apache/spark-2.2.0-bin-hadoop2.7/spark-events/spark1/` are cleaned up.

6

Cassandra Support In Bright Cluster Manager

Apache Cassandra is a database that features linear scalability and proven fault-tolerance on commodity hardware. Cassandra has no single point of failure, since every node in the cluster is identical—it has no master-slave architecture. Fault-tolerance is realized by automatic replication of data to multiple nodes. Depending on the application, Cassandra can outperform popular NoSQL alternatives.

Apache Cassandra can be installed alongside Hadoop and Spark instances. Indeed, Apache Spark can use Hadoop and Cassandra as source and destination for analytics.

6.1 Cassandra Installation In Bright Cluster Manager–Overview

6.1.1 Prerequisites For Cassandra Installation, And What Cassandra Installation Does

The following applies to installing Cassandra 3.7-bin (section 6.1.2) on Bright Cluster Manager:

- Cassandra depends on Java 1.8.0. For Centos 6 and 7 this version of Java can be installed via `yum`, while for SLES, Java should be downloaded from the Oracle website at `http://java.com/en/download/manual.jsp`
- Cassandra requires the Jolokia JVM Agent, which is available as a package `cm-apache-hadoop-extras` in the Bright Computing repository.
- The Cassandra Query Language Shell, `cqlsh`, runs with Python 2.7 or higher. CENTOS 6 ships with Python 2.6. In order to install a separate Python 2.7, the instructions in `http://kb.brightcomputing.com/faq/index.php?action=artikel&cat=18&id=198` can be used as a guide.
- The script installs Cassandra on selected nodes, and creates a dedicated Hadoop Configuration Group for Cassandra
- Cassandra is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Cassandra configuration files are copied by the script and placed under `/etc/hadoop/`

`cm-cassandra-setup` takes an XML configuration file with the `-c` option. The configuration file has four main sections:

- `<topology>`: for Cassandra nodes and seeds
- `<ports>`: for communication among Cassandra nodes, and also for communication with clients
- `<directories>`: for storing data
- `<params>`: for specifying configuration details

More information on the Cassandra installation configuration can be found at `http://wiki.apache.org/cassandra/GettingStarted/`.

An example XML file is:

Example

```
<cassandraConfig>
  <archive>/cm/local/apps/hadoop/apache-cassandra-3.7-bin.tar.gz</archive>
  <javahome>/usr/lib/jvm/jre-1.8.0-openjdk/</javahome>
  <jolokiajar>/cm/local/apps/hadoop/jolokia-jvm-1.3.3-agent.jar</jolokiajar>
  <instance>
    <name>cass1</name>
    <topology>
      <nodes>node001..node005</nodes>
      <seeds>node001..node002</seeds>
    </topology>
    <ports>
      <storageport>7000</storageport>
      <sslstorageport>7001</sslstorageport>
      <nativetransportport>9042</nativetransportport>
      <nativetransportportssl>9142</nativetransportportssl>
      <jmxport>7199</jmxport>
      <jolokiaport>8778</jolokiaport>
    </ports>
    <directories>
      <commitlog>/var/lib/cassandra/cass1/commitlog</commitlog>
      <datafile>/var/lib/cassandra/cass1/data</datafile>
      <savedcaches>/var/lib/cassandra/cass1/saved_caches</savedcaches>
      <hints>/var/lib/cassandra/cass1/hints</hints>
    </directories>
    <params>
      <numtokens>256</numtokens>
      <endpointsnitch>org.apache.cassandra.locator.SimpleSnitch</endpointsnitch>
      <maxheapsize></maxheapsize>
      <heapnewsize></heapnewsize>
    </params>
  </instance>
</cassandraConfig>
```

Regarding the XML tags in the configuration file:

- The following XML tags correspond to the YAML property names indicated in `cassandra.yaml`:

XML	YAML
`<name>`	`cluster_name`
`<storageport>`	`storage_port`
`<nativetransportport>`	`native_transport_port`
`<numtokens>`	`num_tokens`
`<endpointsnitch>`	`endpoint_snitch`

- Multiple Cassandra instances can be installed, with each one having a different `<name>`.
- Values for the tags `<sslstorageport>` and `<nativetransportportssl>` can be set but are currently unsupported

- Cassandra ring uses vnodes to assign tokens, and each node has the amount of tokens specified in `numtokens`.
- `<endpointsnitch>` is the class which contains the logic to group nodes in "datacenters" and "racks." Possible values include:
  - `org.apache.cassandra.locator.SimpleSnitch`. This is the default.
  - `org.apache.cassandra.locator.GossipingPropertyFileSnitch`. This is the suggested value for production use. Bright Cluster Manager writes a specific `cassandra-rackdc.properties` on each node, and topology information is exchanged via the gossip protocol.
  - `org.apache.cassandra.locator.PropertyFileSnitch`. This value causes Bright Cluster Manager to write the same `cassandra-topology.properties` on each node.
  - `org.apache.cassandra.locator.RackInferringSnitch`. This value tells Cassandra to assume that datacenter and rack correspond to the 2nd and 3rd octet respectively, of the IP address of each node.

  For the values of `GossipingPropertyFileSnitch` and `PropertyFileSnitch`, Bright Cluster Manager uses any values already set by the administrator for the rack properties (section 3.12 of the *Administrator Manual*). Thus:
  - `DC1` is set as the default datacenter value for Cassandra. The `AdvancedConfig` directive `CassandraDefaultDataCenter` (Appendix C, page 637 of the *Administrator Manual*) can be used to override the default value `DC1`.
  - If there is a rack name associated with the node in Bright Cluster Manager, then it is set as the rack name for Cassandra. If there is no rack name associated with the node in Bright Cluster Manager, then a rack name of `RAC1` is set by default. The `AdvancedConfig` directive `CassandraDefaultRack` (Appendix C, page 638 of the *Administrator Manual*) can be used to override the default value `RAC1`.
- The tag `<topology>` comprises `<nodes>` and `<seeds>`:
  - `<nodes>` specifies the list of nodes on which Cassandra is deployed
  - `<seeds>` specifies the subset of nodes to be used as "seeds". Two nodes is a common choice.
- The tag `<jmxport>` is used to set the port that Cassandra makes available for JMX connections. It is by default accessible only from localhost.
- The tag `<jolokiaport>` is used to set the port that the Jolokia JVM Agent uses to expose metrics.
- The tag `<directories>` specifies the set of directories that Cassandra uses. Multiple comma-separated entries can be specified for `<datafile>`
- The tags `<maxheapsize>` and `<heapnewsize>` can be left empty or set in pairs, e.g. `4G` and `800M`. If empty, then proper values are automatically calculated by `cassandra-env.sh`

6.1.2 Cassandra Installation With `cm-cassandra-setup`

An installation with `cm-cassandra-setup` using an XML configuration file looks like the following:

Example

```
[root@bright81 ~]# cm-cassandra-setup -c cass.xml
Reading config from file '/root/cass.xml'... done.
Cassandra release '3.7-bin'
Cassandra being installed... done.
```

```
Creating Cassandra instance 'cass1'... done.
Creating directories for Cassandra... done.
Creating module file for Cassandra... done.
Creating configuration files for Cassandra... done.
Updating images... done.
Updating configuration in CMDaemon... done.
Adding seeds to Cassandra instance... done.
Adding remaining nodes...
- adding node003...
- adding node004...
- adding node005...
All nodes added.
Installation successfully completed.
Finished.
```

6.1.3 Cassandra Removal With `cm-cassandra-setup`

To uninstall a Cassandra instance, the `-u` option can be used:

Example

```
[root@bright81 ~]# cm-cassandra-setup -u cass1
Requested removal of Cassandra for Hadoop instance 'cass1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Cassandra directories... done.
Removal successfully completed.
Finished.
```

6.2 Cassandra Endpoint Snitches In Bright Cluster Manager

Bright Cluster Manager currently does not support switching snitches. Once an endpoint snitch is defined and the Cassandra instance has been deployed, then changing the endpoint snitch requires manual operations.

6.2.1 `SimpleSnitch`

If the default endpoint snitch `SimpleSnitch` is chosen, then Cassandra does not use the values already set by the administrator for the rack properties in Bright Cluster Manager. Datacenter and rack are then always `datacenter1` and `rack1`, as shown in the example:

Example

```
[root@bright81 ~]# module load cassandra
[root@bright81 ~]# nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens Owns         Host ID                               Rack
                                  (effective)
UN  10.141.0.5  74.37 KiB  256    42.1%        3f7dfed3-c7ed-44ea-aa0a-fad24ccc3b0d  rack1
UN  10.141.0.4  102.59 KiB 256    37.6%        28419a23-5b67-4b90-9177-c443b3df7c99  rack1
UN  10.141.0.1  105.7 KiB  256    40.3%        ce29fb2d-e0b0-493a-820d-b927c2b514b7  rack1
UN  10.141.0.3  126.7 KiB  256    38.4%        15bc50f4-35ce-4111-8077-cb7c85236b2f  rack1
UN  10.141.0.2  83.57 KiB  256    41.7%        d65ca173-c729-4bbc-b519-491e6e2fd49b  rack1
```

6.2.2 GossipingPropertyFileSnitch and PropertyFileSnitch

When choosing GossipingPropertyFileSnitch or PropertyFileSnitch, Bright Cluster Manager by default uses DC1 for the datacenter and RAC1 for the rack. The AdvancedConfig directives CassandraDefaultDataCenter and CassandraDefaultRack can be used to override these defaults.

Example

```
[root@bright81 ~]# module load cassandra
[root@bright81 ~]# nodetool status
Datacenter: DC1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns          Host ID                               Rack
                                   (effective)
UN  10.141.0.5  74.34 KiB  256     37.2%         5717a49a-b874-40ea-9d9a-e64b8cc3d493  RAC1
UN  10.141.0.4  102.55 KiB 256     40.4%         6d47a1a1-da02-487a-82c5-901bc3d693c8  RAC1
UN  10.141.0.1  86.74 KiB  256     42.7%         e5c19c88-3e43-466c-a725-d4b4c31c5dba  RAC1
UN  10.141.0.3  126.76 KiB 256     40.4%         984d17ae-6a46-4a6d-88ed-59c5cba7902a  RAC1
UN  10.141.0.2  83.58 KiB  256     39.4%         efd3b226-ee3e-4491-a7f3-1aaae6131016  RAC1
```

If the administrator has already set values for the rack properties in Bright Cluster Manager, those will be used in Cassandra.

Example

```
[root@bright81 ~]# module load cassandra
[root@bright81 ~]# nodetool status
Datacenter: DC1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns          Host ID                               Rack
                                   (effective)
UN  10.141.0.5  285.63 KiB  256    36.6%         d276a67a-1fa9-4d26-acf5-53298359c8ed  rack2
UN  10.141.0.4  121.53 KiB  256    37.7%         9157a546-f455-4bb9-b819-49220838b469  rack2
UN  10.141.0.1  346.84 KiB  256    41.3%         c1475481-8d77-49bd-a25b-d892a0cfe40a  rack1
UN  10.141.0.3  211.27 KiB  256    42.5%         f85fa3a0-c40c-4f8d-b1c5-3ec2bb0ba679  rack1
UN  10.141.0.2  245.63 KiB  256    41.9%         9b175841-ac2a-4160-bbc2-e670a4511c7b  rack1
```

6.2.3 RackInferringSnitch

Example

```
[root@bright81 ~]# module load cassandra
[root@bright81 ~]# nodetool status
Datacenter: 141
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns          Host ID                               Rack
                                   (effective)
UN  10.141.0.5  88.28 KiB  256     36.7%         823e1bb6-d941-4dd8-9d5b-869277dd42af  0
UN  10.141.0.4  15.37 KiB  256     39.4%         04df0e46-61eb-4feb-86f3-142624ba53ef  0
UN  10.141.0.1  74.33 KiB  256     40.5%         7f442510-79c4-4401-90b9-f6c96ab3d794  0
UN  10.141.0.3  121.54 KiB 256     41.7%         2d633433-51a1-4af5-ac1d-b052892a9e14  0
UN  10.141.0.2  88.76 KiB  256     41.7%         aad92c3a-3d75-49aa-8e01-688706fea294  0
```

6.2.4 Deploy Cassandra On A Specific Network

By default Cassandra is configured to use the management network `internalnet`, as defined in the base partition. This means that CMDaemon writes out the Cassandra configuration using the IP addresses of that network.

When multiple internal networks are available, users would want to specify a different internal network for Cassandra traffic. It is possible to do that by using the tag `<network>`, as shown in the following example:

Example

```
<cassandraConfig>
  ...
  <instance>
    <name>cass1</name>
    <network>cassnet</network>
    <topology>
    ...
</cassandraConfig>
```

The Cassandra instance is configured by this to use the selected network, and Cassandra services then bind to the correct IP, as defined by `listen_address` in `cassandra.yaml`.

Once the Cassandra instance has been installed, it is not possible to change the selected network, as indicated by `cmsh`:

```
[bright81->bigdata]% show cass1 | grep Network
Network                                          cassnet
```

6.3 Cassandra Maintenance Operations With `cm-cassandra-maint`

The Cassandra maintenance script, `cm-cassandra-maint`, is a Python script. It is called using the full path. If it is run with no arguments, then it displays a help page:

Example

```
[root@bright81 ~]# /cm/local/apps/cluster-tools/hadoop/cm-cassandra-maint

Cassandra instance name must be specified. Exiting.

  USAGE: /cm/local/apps/cluster-tools/hadoop/cm-cassandra-maint -i <name>
    [--addnode <host> | --removenode <host> | --replacenode <host> --withnode <host>] | -h ]

  OPTIONS:
  -i <name>             --  instance name
  --addnode <host>      --  add node to Cassandra instance
  --removenode <host>   --  remove node from Cassandra instance
  --replacenode <host>  --  dead node to be replaced
  --withnode <host>     --  replacement node
  -h                    --  show usage

  EXAMPLES:
  cm-cassandra-maint -i cass1 --addnode node005
  cm-cassandra-maint -i cass1 --removenode node005
  cm-cassandra-maint -i cass1 --replacenode node004 --withnode node005
```

The name of the Cassandra instance, specified with `-i`, is mandatory. The other options are explained next in more detail:

- --addnode: adds a node to the Cassandra instance. The script configures the new node for Cassandra and bootstraps the corresponding service. After the new node is up (UN), the script executes a nodetool cleanup on all the nodes associated with the Cassandra instance
- --removenode: removes a node from the Cassandra instance. The script only goes ahead with node removal if the node is not a "seed" node. If the node is up (UN), then it is first decommissioned, otherwise it is removed. If removal fails, then the node is assassinated via nodetool
- --replacenode <host> --withnode <host>: replaces a dead Cassandra node with a new node. This option should be used only if the node to be replaced is dead (DN). The new node is bootstrapped with the option cassandra.replace_address in the cassandra.yaml file. After the new node is up (UN), the script executes a nodetool cleanup on all the nodes associated with the Cassandra instance.

7

Big Data Software Tools From Hadoop-related Projects

Besides Spark (Chapter 5) and Cassandra (Chapter 6), there are several other projects that 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 utilities to help install the following software from these projects:

- Accumulo (section 7.1)
- Alluxio (section 7.2)
- Drill (section 7.3)
- Flink (section 7.4)
- Giraph (section 7.5)
- Hive (section 7.6)
- Ignite (section 7.7)
- Kafka (section 7.8)
- Pig (section 7.9)
- Sqoop (section 7.10)
- Sqoop2 (section 7.11)
- Storm (section 7.12)

7.1 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).

7.1.1 Accumulo Installation With `cmhadoop-accumulo-setup`

Bright Cluster Manager provides `cmhadoop-accumulo-setup` to carry out the installation of Accumulo as part of the `cm-apache-hadoop-extras` package.

Prerequisites For Accumulo Installation, And What Accumulo Installation Does

The following applies to using `cmhadoop-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 `cmhadoop-accumulo-setup` script only installs Accumulo on the active head node and on the DataNodes of the chosen Hadoop instance.
- The script creates two dedicated configuration overlays for Accumulo: one for Accumulo Master and one for Accumulo Tablet Servers.
- 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 `cmhadoop-accumulo-setup`.
- The secret string for the instance is a random string created by `cmhadoop-accumulo-setup`.
- A password for the `root` user must be specified.
- The Tracer service uses 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.
- `cmhadoop-accumulo-setup` tries to build the native map library. If no Java Development Kit is available, then the script displays a warning message.
- 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 `cmhadoop-accumulo-setup` are listed on running `cmhadoop-accumulo-setup -h`.

An Example Run With `cmhadoop-accumulo-setup`

The option

`-j` <*path*>

is not mandatory. It is used to set the Java home path in Accumulo environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

The option

`-p` <*rootpass*>

is mandatory. The specified password is also used by the Tracer service to connect to Accumulo. The password is 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@bright81 ~]# cmhadoop-accumulo-setup -i hdfs1 -p 12345 -s 900MB \
 -t /tmp/accumulo-1.7.0-bin.tar.gz --master node005
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Accumulo release '1.7.0'
Accumulo GC, Master, Monitor, and Tracer services will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.7.1
Accumulo being installed... done (with native library).
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... done.
Initializing master services for Accumulo... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

7.1.2 Accumulo Removal With `cmhadoop-accumulo-setup`

`cmhadoop-accumulo-setup` uses the `-u` option to uninstall the Accumulo instance. Data and metadata are not removed.

Example

```
[root@bright81 ~]# cmhadoop-accumulo-setup -u hdfs1
Requested removal of Accumulo for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
```

```
Cleaning ZooKeeper... done.
Removing additional Accumulo directories... done.
Removal successfully completed.
Finished.
```

7.1.3 Accumulo MapReduce Example

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

Example

```
[root@bright81 ~]# 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
```

7.2 Alluxio

Alluxio (formerly known as Tachyon) is a memory-centric distributed storage system. It lies between computation frameworks and various storage systems, in order to enable reliable data sharing across cluster jobs.

An Alluxio instance includes the following main components:

- Master, which is primarily responsible for managing the global metadata of the system
- Workers, to manage local resources
- Client, to interact with the Alluxio servers

The Alluxio tarball can be downloaded from `http://alluxio.org/`. The appropriate version to use is listed in section 1.4.

7.2.1 Alluxio Installation With `cmhadoop-alluxio-setup`

Bright Cluster Manager provides `cmhadoop-alluxio-setup` to carry out Alluxio installation:

Prerequisites For Alluxio Installation, And What Alluxio Installation Does

The following applies to using `cmhadoop-alluxio-setup` with Hadoop:

- A Hadoop instance must already be installed.
- `cmhadoop-alluxio-setup` installs Alluxio only on the active head node and on the DataNodes of the chosen Hadoop instance.
- The script sets the under storage address to the HDFS namenode address
- The script creates two dedicated configuration overlays for Alluxio: one for Alluxio Master and one for Alluxio Workers
- Alluxio is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Alluxio configuration files are copied by the script to under `/etc/hadoop/`

The options for `cmhadoop-alluxio-setup` are listed on running `cmhadoop-alluxio-setup -h`.

An Example Run With `cmhadoop-alluxio-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Alluxio environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-alluxio-setup -i hdfs1 -t /tmp/alluxio-1.5.0\
-hadoop-2.7-bin.tar.gz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Alluxio release '1.5.0-hadoop-2.7-bin'
Alluxio Master will be run on the head node.
Alluxio being installed... done.
Creating directories for Alluxio... done.
Creating module file for Alluxio... done.
Creating configuration files for Alluxio... done.
Updating images... done.
Initializing Alluxio directory in HDFS... done.
Updating Hadoop configuration... done.
Formatting Alluxio FS... done.
Waiting for NameNode to be ready... done.
Initializing Alluxio Master service... done.
Initializing Alluxio Worker services... done.
Updating configuration in CMDaemon... done.
Validating Alluxio setup... done.
Installation successfully completed.
Finished.
```

7.2.2 Alluxio Removal With `cmhadoop-alluxio-setup`

`cmhadoop-alluxio-setup` uses the `-u` option to remove the Alluxio instance.

Example

```
[root@bright81 ~]# cmhadoop-alluxio-setup -u hdfs1
Requested removal of Alluxio for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Alluxio directories... done.
Removing Alluxio-related metrics... done.
Removal successfully completed.
Finished.
```

7.2.3 Using Alluxio

Alluxio consists of several components: one Master, multiple Workers, and an executable client, `alluxio`, that can be run after the user loads the corresponding module. The following example shows how to execute a MapReduce job, wordcount, on data copied to the Alluxio storage layer.

Example

```
[root@bright81 ~]# module load alluxio/hdfs1/Apache/1.5.0-hadoop-2.7-bin
[root@bright81 ~]# alluxio fs copyFromLocal /cm/local/examples/hamlet.txt\
 /wordcount/input.txt
Copied file:///cm/local/examples/hamlet.txt to /wordcount/input.txt
[root@bright81 ~]# module load hadoop/hdfs1/Apache/2.7.3
[root@bright81 ~]# hadoop jar /cm/shared/apps/hadoop/hdfs1/share/hadoop/mapreduce/\
```

```
hadoop-mapreduce-examples-*.jar wordcount -libjars /cm/shared/apps/hadoop/Apache/a\
lluxio-1.5.0-hadoop-2.7-bin/client/hadoop/alluxio-1.5.0-hadoop-client.jar\
 alluxio://ml-bigdatadev.cm.cluster:19998/wordcount/input.txt\
 alluxio://ml-bigdatadev.cm.cluster:19998/wordcount/output
17/06/23 12:36:18 INFO metrics.MetricsSystem: Starting sinks with config: .
17/06/23 12:36:18 INFO hadoop.HadoopConfigurationUtils: Loading Alluxio properties\
 from Hadoop configuration:
17/06/23 12:36:18 INFO alluxio.AbstractClient: Alluxio client (version 1.5.0) is t\
rying to connect with FileSystemMasterClient @ ml-bigdatadev.cm.cluster/10.141.255\
.254:19998
17/06/23 12:36:18 INFO alluxio.AbstractClient: Client registered with FileSystemMa\
sterClient @ ml-bigdatadev.cm.cluster/10.141.255.254:19998

...

17/06/23 12:36:21 INFO mapreduce.Job: The url to track the job: http://node003.cm.\
cluster:8088/proxy/application_1498213212420_0005/
17/06/23 12:36:21 INFO mapreduce.Job: Running job: job_1498213212420_0005
17/06/23 12:36:31 INFO mapreduce.Job: Job job_1498213212420_0005 running in uber m\
ode : false
17/06/23 12:36:31 INFO mapreduce.Job:  map 0% reduce 0%
17/06/23 12:36:39 INFO mapreduce.Job:  map 100% reduce 0%
17/06/23 12:36:48 INFO mapreduce.Job:  map 100% reduce 100%
17/06/23 12:36:49 INFO mapreduce.Job: Job job_1498213212420_0005 completed success\
fully

...

[root@bright81 ~]# alluxio fs cat /wordcount/output/*
'Tis 1
'tis 1
And 5

...


would 2
wrong,1
you 1
```

7.3 Drill

Apache Drill is an SQL query engine for Big Data exploration. Drill supports a variety of NoSQL databases and file systems. A single query can join data from multiple datastores. In addition, Drill supports data locality, so putting Drill and the datastore on the same nodes is a good idea.

The Apache Drill tarball can be downloaded from `http://drill.apache.org/`. Hadoop version and distribution compatibilities are listed in section 1.4.

By default a storage named `dfs` is defined. This points to the local filesystem, and not the HDFS filesystem by default.

During setup an additional storage named `hdfs` is created, which is initialized as a connection to HDFS.

Interfacing with HDFS requires a Java JDK, and not just the JRE. In case Drill is already deployed, `cm-hadoop-maint` can be used to change the `JAVA_HOME` enviroment variable to have it point to a JDK for Drill, and/or Hadoop itself.

7.3.1 Drill Installation With cmhadoop-drill-setup

Bright Cluster Manager provides cmhadoop-drill-setup to carry out Drill installation:

Prerequisites For Drill Installation, And What Drill Installation Does

The following applies to using cmhadoop-drill-setup:

- A Hadoop instance must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- The cmhadoop-drill-setup script installs Drill services by default on the DataNodes of the chosen Hadoop instance.
- The script creates a dedicated configuration overlay for Drill.
- Drill executables are copied by the script to a subdirectory under /cm/shared/hadoop/
- Drill configuration files are copied by the script to under /etc/hadoop/
- The Drillbit services are started up by the script
- Validation tests are carried out by the script using sqlline.

The options for cmhadoop-drill-setup are listed on running cmhadoop-drill-setup -h.

An Example Run With cmhadoop-drill-setup

The option -j <*path*> is not mandatory. It is used to set the Java home path in Drill environment files. If the option is not specified, then the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-drill-setup -i hdfs1 -t /tmp/apache-drill-1.4.0.tar.gz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Drill release '1.4.0'
Found Hadoop instance 'hdfs1', release: 2.7.1
Drill being installed... done.
Creating directories for Drill... done.
Creating module file for Drill... done.
Creating configuration files for Drill... done.
Updating images... done.
Initializing services for Drill... done.
Updating configuration in CMDaemon... done.
Validating Drill setup... done.
Installation successfully completed.
Finished.
```

7.3.2 Drill Removal With cmhadoop-drill-setup

cmhadoop-drill-setup uses the -u option to uninstall Drill.

Example

```
[root@bright81 ~]# cmhadoop-drill-setup -u hdfs1
Requested removal of Drill for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
```

```
Cleaning ZooKeeper... done.
Removing additional Drill directories... done.
Removal successfully completed.
Finished.
```

7.4 Flink

Apache Flink is an open source platform for distributed stream and batch data processing. Flink's pipelined runtime system enables the execution of bulk batch and stream processing programs. Flink does not provide its own data storage system—input data must be stored in a distributed storage system such as HDFS or HBase.

The Apache Flink tarball can be downloaded from `http://flink.apache.org/`. Hadoop versions and distributions compatibilities are listed in section 1.4.

7.4.1 Flink Installation With `cmhadoop-flink-setup`

Bright Cluster Manager provides `cmhadoop-flink-setup` to carry out Flink installation:

Prerequisites For Flink Installation, And What Flink Installation Does

The following applies to using `cmhadoop-flink-setup`:

- A Hadoop instance must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- The `cmhadoop-flink-setup` script by default only installs Flink Job Manager on the active head node and Flink Task Managers on the DataNodes of the chosen Hadoop instance. A node other than master can be specified by using the option `--master`, or its alias for this setup script, `--jobmanager`.
- The script creates two dedicated configuration overlays for Flink: one for the Job Manager and one for Task Managers.
- Flink executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Flink configuration files are copied by the script to under `/etc/hadoop/`
- The Flink Job Manager and Task Manager services are started up by the script
- Validation tests are carried out by the script using `flink`.

The options for `cmhadoop-flink-setup` are listed on running `cmhadoop-flink-setup -h`.

An Example Run With `cmhadoop-flink-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Flink environment files. If the option is not specified, then the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-flink-setup -i hdfs1 -t /tmp/flink-0.10.1-bin-hadoop27.tgz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Flink release '0.10.1-bin-hadoop27'
Found Hadoop instance 'hdfs1', release: 2.7.1
ZooKeeper found. Configuring Flink JobManager HA.
Flink Job Manager will be run on the head node.
Flink being installed... done.
Creating directories for Flink... done.
```

```
Creating module file for Flink... done.
Creating configuration files for Flink... done.
Updating images... done.
Initializing Task Managers for Flink... done.
Initializing Job Manager for Flink... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Validating Flink setup... done.
Installation successfully completed.
Finished.
```

7.4.2 Flink Removal With `cmhadoop-flink-setup`

`cmhadoop-flink-setup` uses the `-u` option to uninstall Flink.

Example

```
[root@bright81 ~]# cmhadoop-flink-setup -u hdfs1
Requested removal of Flink for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Flink directories... done.
Removal successfully completed.
Finished.
```

7.5 Giraph

Apache Giraph is an iterative graph processing system built for high scalability. Giraph is inspired by the Bulk Synchronous Parallel model of distributed computation introduced by Leslie Valiant. Giraph is built on top of Apache Hadoop and it uses the MapReduce framework to run Giraph jobs. The input to a Giraph computation is a graph composed of vertices and directed edges. As an example, Giraph can compute the length of the shortest paths from a source node to all other nodes.

The Apache Giraph tarball should be built from sources, since it depends on the Hadoop distribution and version. A specific patch is needed to build Giraph against Hadoop 2.7.x, as shown in the following examples:

Building Giraph For Hadoop 2.7.3

A patch is needed for Hadoop 2.7.3.

```
[foobar@bright81 ~]$ curl -O http://www-us.apache.org/dist/giraph/giraph-1.2.0/gi\
raph-dist-1.2.0-hadoop2-src.tar.gz
[foobar@bright81 ~]$ tar xvzf giraph-dist-1.2.0-hadoop2-src.tar.gz
[foobar@bright81 ~]$ cd giraph-1.2.0-hadoop2/
[foobar@bright81 ~]$ curl -O https://issues.apache.org/jira/secure/attachment/128\
43736/GIRAPH-1110.02.patch
[foobar@bright81 ~]$ patch -p1 < GIRAPH-1110.02.patch
[foobar@bright81 ~]$ mvn -Dhadoop.version=2.7.3 -Phadoop_2 -fae -DskipTests clean package
```

For installation, the tarball can be found in `./giraph-dist/target/`, and the JAR file can be found in `./giraph-examples/target/`:

```
[foobar@bright81 ~]$ ls giraph-dist/target/giraph-1.2.0-hadoop2-for-hadoop-2.7.3-bin.tar.gz
[foobar@bright81 ~]$ ls giraph-examples/target/giraph-examples-1.2.0-hadoop2-for-\
hadoop-2.7.3-jar-with-dependencies.jar
```

Building Giraph For Hadoop 2.7.4

No patch needed for Hadoop 2.7.4.

```
[foobar@bright81 ~]$ curl -O http://www-us.apache.org/dist/giraph/giraph-1.2.0/gi\
raph-dist-1.2.0-hadoop2-src.tar.gz
[foobar@bright81 ~]$ tar xvzf giraph-dist-1.2.0-hadoop2-src.tar.gz
[foobar@bright81 ~]$ cd giraph-1.2.0-hadoop2/
[foobar@bright81 ~]$ mvn -Dhadoop.version=2.7.4 -Phadoop_2 -fae -DskipTests clean package
```

For installation, the tarball can be found in `./giraph-dist/target/`, and the JAR file can be found in `./giraph-examples/target/`:

```
[foobar@bright81 ~]$ ls giraph-dist/target/giraph-1.2.0-hadoop2-for-hadoop-2.7.4-bin.tar.gz
[foobar@bright81 ~]$ ls giraph-examples/target/giraph-examples-1.2.0-hadoop2-for-\
hadoop-2.7.4-jar-with-dependencies.jar
```

7.5.1 Giraph Installation With `cmhadoop-giraph-setup`

Bright Cluster Manager provides `cmhadoop-giraph-setup` to carry out Giraph installation:

Prerequisites For Giraph Installation, And What Giraph Installation Does

The following applies to using `cmhadoop-giraph-setup`:

- A Hadoop instance must already be installed.
- `cmhadoop-giraph-setup` installs Giraph only on the active head node.
- The script creates no roles for for Giraph.
- Giraph is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Giraph configuration files are copied by the script to under `/etc/hadoop/`.

The options for `cmhadoop-giraph-setup` are listed on running `cmhadoop-giraph-setup -h`.

An Example Run With `cmhadoop-giraph-setup`

The option `-j` *<path>* is not mandatory. It is used to set the Java home path in Giraph environment files. If the option is not specified, then the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-giraph-setup -i hdfs1 -t /tmp/giraph-1.2.0-hadoop2-\
for-hadoop-2.7.4-bin.tar.gz --examplejar /giraph-examples-1.2.0-hadoop2-for-had\
oop-2.7.4-jar-with-dependencies.jar
Java home not specified, using: /usr/lib/jvm/jre-1.8.0-openjdk.x86_64/
Giraph release '1.2.0-hadoop2-for-hadoop-2.7.4-bin'
Found Hadoop instance 'hdfs1', release: 2.7.4
ZooKeeper found for instance hdfs1.
Giraph being installed... done.
Creating directories for Giraph... done.
Creating module file for Giraph... done.
Creating configuration files for Giraph... done.
Waiting for NameNode to be ready... done.
Validating Giraph setup... done.
Installation successfully completed.
Finished.
```

7.5.2 Giraph Removal With cmhadoop-giraph-setup

cmhadoop-giraph-setup uses the -u option to uninstall the Giraph instance. Data and metadata will not be removed.

Example

```
[root@bright81 ~]# cmhadoop-giraph-setup -u hdfs1
Requested removal of Giraph for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Giraph directories... done.
Removal successfully completed.
Finished.
```

7.6 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 PostgreSQL. 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.

7.6.1 Hive Installation With cmhadoop-hive-setup

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

Prerequisites For Hive Installation, And What Hive Installation Does

The following applies to using cmhadoop-hive-setup:

- A Hadoop instance must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- For the MySQL backend: 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/
  - cmhadoop-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
    ```
- For the PostgreSQL backend: the package postgresql-jdbc for RHEL or SLES, or libpostgresql-jdbc-java in case of Ubuntu, should be installed on the node selected for Hive services.

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

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

- For the PostgreSQL backend: before running the script, the following statements must be executed explicitly by the administrator, using a PostgreSQL client:

```
DROP DATABASE IF EXISTS <metastoredb>;
CREATE DATABASE <metastoredb>;
CREATE USER hive WITH PASSWORD '<hivepass>';
GRANT ALL PRIVILEGES ON DATABASE <metastoredb> TO hive;
```

- For any backend, in the preceding statements:
  - *<metastoredb>* is the name of metastore database to be used by Hive. The same name is used later by `cmhadoop-hive-setup`.
  - *<hivepass>* is the password for `hive` user. The same password is used later by `cmhadoop-hive-setup`.
  - The DROP line is needed only if a database with that name already exists.

- The `cmhadoop-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 configuration overlay 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/`

- By default, 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 `--dbserver` and `--dbport`.

- In order to use PostgreSQL, the options `--dbserver` and `--dbport` must be specified, along with `--usepostgresql`

- 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.

The options for `cmhadoop-hive-setup` are listed on running `cmhadoop-hive-setup -h`.

An Example Run With `cmhadoop-hive-setup` and MySQL

The option `-j` <*path*> is not mandatory. It is used to set the Java home path in Hive environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-hive-setup -i hdfs1 -p <hivepass> --metastoredb <metastoredb> \
  -t /tmp/apache-hive-2.1.1-bin.tar.gz --master node005
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Hive release '2.1.1-bin'
Using MySQL server on active headnode.
Succesfully connected to Hive database with provided credentials.
Hive service will be run on node: node005
Hive being installed... done.
Using MySQL connector installed in /usr/share/java/
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.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Hive setup validation...
Hive setup validation... done.
Installation successfully completed.
Finished.
```

An Example Run With `cmhadoop-hive-setup` and PostgreSQL

The option `-j` <*path*> is not mandatory. It is used to set the Java home path in Hive environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance. The options `--dbserver` <*node*>, `--dbport` <*port*>, and `--usepostgresql` are mandatory.

Example

```
[root@bright81 ~]# cmhadoop-hive-setup -i hdfs1 -p <hivepass> --metastoredb <metastoredb> \
  -t /tmp/apache-hive-2.1.1-bin.tar.gz --master node005 \
  --dbserver ml-bigdatadev --dbport 5432 --usepostgresql
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Hive release '2.1.1-bin'
Using PostgreSQL server: ml-bigdatadev.cm.cluster
Succesfully connected to Hive database with provided credentials.
Hive service will be run on node: node005
Hive being installed... done.
Using PostgreSQL connector installed in /usr/share/java/
Creating directories for Hive... done.
Creating module file for Hive... done.
Creating configuration files for Hive... done.
Initializing database 'metastore_hdfs1' in PostgreSQL... done.
Waiting for NameNode to be ready... done.
Creating HDFS directories for Hive... done.
Updating images... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Hive setup validation...
```

```
Hive setup validation... done.
Installation successfully completed.
Finished.
```

7.6.2 Hive Removal With cmhadoop-hive-setup

cmhadoop-hive-setup uses the -u option to uninstall the Hive instance. Data and metadata will not be removed.

Example

```
[root@bright81 ~]# cmhadoop-hive-setup -u hdfs1
Requested removal of Hive for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Cleaning ZooKeeper... done.
Removing additional Hive directories... done.
Removal successfully completed.
Finished.
```

7.6.3 Beeline

The latest Hive releases include HiveServer2, which supports the 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@bright81 ~]# module load hive
[root@bright81 ~]# 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 2.1.1)
Driver: Hive JDBC (version 2.1.1)
Transaction isolation: TRANSACTION_REPEATABLE_READ
+------------------+--+
|     tab_name     |
+------------------+--+
| validation_data  |
+------------------+--+
1 row selected (0.23 seconds)
Beeline version 2.1.1 by Apache Hive
Closing: 0: jdbc:hive2://node005.cm.cluster:10000
```

7.7 Ignite

Apache Ignite is a high-performance, integrated and distributed in-memory platform for computing. Specifically Apache Ignite In-Memory MapReduce eliminates the overhead associated with NameNode, since data locality is assured via a hashing function. It also uses push-based resource allocation for better performance.

7.7.1 Ignite Installation With cmhadoop-ignite-setup

Bright Cluster Manager provides cmhadoop-ignite-setup to carry out Ignite installation.

Prerequisites For Ignite Installation, And What Ignite Installation Does

The following applies to using `cmhadoop-ignite-setup`:

- A Hadoop instance must already be installed. Ignite installation is not supported for Hadoop 1.x and Cloudera CDH 4.x
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- `cmhadoop-ignite-setup` installs Ignite only on the ResourceManager nodes
- The script creates no roles for for Ignite
- Ignite is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Ignite configuration files are copied by the script to under `/etc/hadoop/`
- `cmhadoop-ignite-setup` creates a configuration overlay (section 3.1.7), for example `hdfs1-ignite`, comprising the nodes for the Hadoop instance. The configuration overlay contains a customization properties section that can be used to set the property `mapreduce.jobtracker.address` to be one of the Ignite servers in `mapred-site.xml`. For example: `mapreduce.jobtracker.address` is set with value `node001.cm.cluster:11211`

The options for `cmhadoop-ignite-setup` are listed on running `cmhadoop-ignite-setup -h`.

An Example Run With `cmhadoop-ignite-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path for Ignite environment files. If the option is not specified, then the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-ignite-setup -i hdfs1 -t /tmp/apache-ignite-hadoop-1.4.0-bin.zip
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Ignite release '1.4.0-bin'
Found Hadoop instance 'hdfs1', release: 2.7.1
Ignite being installed... done.
Creating directories for Ignite... done.
Creating module file for Ignite... done.
Creating configuration files for Ignite... done.
Updating images... done.
Initializing services for Ignite... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Validating Ignite setup... done.
Installation successfully completed.
Finished.
```

7.7.2 Ignite Removal With `cmhadoop-ignite-setup`

`cmhadoop-ignite-setup` uses the `-u` option to uninstall the Ignite instance.

Example

```
[root@bright81 ~]# cmhadoop-ignite-setup -u hdfs1
Requested removal of Ignite for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
```

```
Updating images... done.
Removing additional Ignite directories... done.
Removal successfully completed.
Finished.
```

7.7.3 Using Ignite

Ignite can be used to run Hadoop jobs, using the Ignite job tracker, by using "In-Memory MapReduce". Further details on this can be found at `http://ignite.apache.org/use-cases/hadoop/mapreduce`. The examples that follow show how to calculate Pi using the Ignite framework or YARN.

Example

Using Ignite

```
[root@bright81 ~]# module load hadoop/hdfs1
[root@bright81 ~]# hadoop jar $HADOOP_PREFIX/share/hadoop/mapreduce/hadoop-mapred\
uce-examples-2.7.1.jar pi 10 1000
Number of Maps  = 10
Samples per Map = 1000

...

Starting Job
Jan 04, 2016 11:30:22 AM org.apache.ignite.internal.client.impl.connection.GridCl\
ientNioTcpConnection <init>

...

16/01/04 11:30:29 INFO mapreduce.Job:  map 100% reduce 100%
16/01/04 11:30:29 INFO mapreduce.Job: Job job_1a759599-56fc-43da-baf0-c68548a7c5d\
b_0002 completed successfully
16/01/04 11:30:29 INFO mapreduce.Job: Counters: 0
Job Finished in 8.015 seconds
Estimated value of Pi is 3.14080000000000000000
```

The MapReduce framework can be switched via cmsh:

```
[ml-hadooptest->hadoop]% use hdfs1
[ml-hadooptest->hadoop[hdfs1]]% get frameworkformapreduce
Ignite
[ml-hadooptest->hadoop[hdfs1]]% set frameworkformapreduce yarn
[ml-hadooptest->hadoop*[hdfs1*]]% commit
```

The same example can be now run using YARN framework. For the same run it results in a more than 7 times slower job execution.

Example

Using YARN

```
[root@bright81 ~]# module load hadoop/hdfs1
[root@bright81 ~]# hadoop jar $HADOOP_PREFIX/share/hadoop/mapreduce/hadoop-mapred\
uce-examples-2.7.1.jar pi 10 1000
Number of Maps  = 10
Samples per Map = 1000

...
```

```
Starting Job
16/01/04 11:33:37 INFO impl.TimelineClientImpl: Timeline service address: http://\
node003.cm.cluster:8188/ws/v1/timeline/

...

16/01/04 11:34:22 INFO mapreduce.Job:  map 100% reduce 100%
16/01/04 11:34:36 INFO mapreduce.Job: Job job_1451903604981_0001 completed succes\
sfully

...

Job Finished in 60.144 seconds
Estimated value of Pi is 3.14080000000000000000
```

7.8 Kafka

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

7.8.1 Kafka Installation With `cmhadoop-kafka-setup`

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

Prerequisites For Kafka Installation, And What Kafka Installation Does

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

- A Hadoop instance, with ZooKeeper, must already be installed.
- `cmhadoop-kafka-setup` installs Kafka only on the ZooKeeper nodes.
- The script creates a dedicated configuration overlay for Kafka.
- 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/`.

The options for `cmhadoop-kafka-setup` are listed on running `cmhadoop-kafka-setup -h`.

An Example Run With `cmhadoop-kafka-setup`

The option `-j` *<path>* is not mandatory. It is used to set the Java home path in Kafka environment files. If the option is not specified, the script will use the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-kafka-setup -i hdfs1 -t /tmp/kafka_2.11-0.11.0.1.tgz
Java home not specified, using: /usr/lib/jvm/jre-1.8.0-openjdk.x86_64/
Kafka release '0.11.0.1' for Scala '2.11'
Found Hadoop instance 'hdfs1', release: 2.7.4
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.
```

```
Updating configuration in CMDaemon... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

7.8.2 Kafka Removal With `cmhadoop-kafka-setup`

`cmhadoop-kafka-setup` uses the `-u` option to uninstall the Kafka instance.

Example

```
[root@bright81 ~]# cmhadoop-kafka-setup -u hdfs1
Requested removal of Kafka for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Cleaning ZooKeeper... done.
Removing additional Kafka directories... done.
Removal successfully completed.
Finished.
```

7.8.3 Using Kafka

The Kafka cluster stores streams of records in categories called topics. The following example shows how to list, create, and delete topics. Once the Kafka module has been loaded, then the `KAFKA_ZOOKEEPER` environment variable can be conveniently used for Kafka commands.

```
[root@bright81 ~]# module show kafka/hdfs1
-------------------------------------------------------------------
/cm/shared/modulefiles/kafka/hdfs1/Apache/0.11.0.1:

module-whatis  adds Kafka to your environment variables
append-path  PATH /cm/shared/apps/hadoop/Apache/kafka_2.11-0.11.0.1//bin
setenv  JAVA_HOME /usr/lib/jvm/jre-1.8.0-openjdk.x86_64/
setenv  KAFKA_BASE_DIR /cm/shared/apps/hadoop/Apache/kafka_2.11-0.11.0.1/
setenv  KAFKA_CONF_DIR /etc/hadoop/hdfs1/kafka
setenv  KAFKA_OPTS
setenv  KAFKA_ZOOKEEPER node001.cm.cluster:2181,node002.cm.cluster:2181,\
node003.cm.cluster:2181/kafka-hdfs1
-------------------------------------------------------------------

[root@bright81 ~]# module load kafka/hdfs1
[root@bright81 ~]# kafka-topics.sh --zookeeper $KAFKA_ZOOKEEPER --list
[root@bright81 ~]# kafka-topics.sh --zookeeper $KAFKA_ZOOKEEPER --create \
 --replication-factor 3 --partitions 1 --topic my-replicated-topic
Created topic "my-replicated-topic".
[root@bright81 ~]# kafka-topics.sh --zookeeper $KAFKA_ZOOKEEPER --list
my-replicated-topic
[root@bright81 ~]# kafka-topics.sh --zookeeper $KAFKA_ZOOKEEPER --delete \
 --topic my-replicated-topic
Topic my-replicated-topic is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
[root@bright81 ~]# kafka-topics.sh --zookeeper $KAFKA_ZOOKEEPER --list
[root@bright81 ~]#
```

7.9 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.

7.9.1 Pig Installation With `cmhadoop-pig-setup`

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

Prerequisites For Pig Installation, And What Pig Installation Does

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

- A Hadoop instance must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- `cmhadoop-pig-setup` installs Pig by default on the active head node. A different node can be specified by using the option `--node`.
- The script creates a dedicated configuration overlay for Pig.
- 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 are automatically copied to a node that has access to LustreFS (NodeManager). Subsequent operations with Pig should be carried out on that node.

The options for `cmhadoop-pig-setup` are listed on running `cmhadoop-pig-setup -h`.

An Example Run With `cmhadoop-pig-setup`

The option `-j` <*path*> is not mandatory. It is used to set the Java home path in Pig environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-pig-setup -i hdfs1 -t /tmp/pig-0.16.0.tar.gz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Pig release '0.16.0'
Found Hadoop instance 'hdfs1', release: 2.7.1
Pig configuration files will be written on the head node.
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.
```

7.9.2 Pig Removal With `cmhadoop-pig-setup`

`cmhadoop-pig-setup` uses the `-u` option to uninstall the Pig instance.

Example

```
[root@bright81 ~]# cmhadoop-pig-setup -u hdfs1
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Removing additional Pig directories... done.
Removal successfully completed.
Finished.
```

7.9.3 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. Further ocumentation for Pig can be found at `http://pig.apache.org/docs/r0.16.0/start.html`.

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

```
[root@bright81 ~]# module load hadoop/hdfs1
[root@bright81 ~]# module load pig/hdfs1
[root@bright81 ~]# 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@bright81 ~]# module load hadoop/hdfs1
[root@bright81 ~]# module load pig/hdfs1
[root@bright81 ~]# pig -v -f /tmp/smoke.pig
```

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

7.10 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.

At present, the latest Sqoop stable release is 1.4.6, while the latest Sqoop2 version is 1.99.6. Sqoop2 is incompatible with Sqoop; it is not feature-complete; and it is not yet intended for production use. Sqoop2 is described in section 7.11.

7.10.1 Sqoop Installation With `cmhadoop-sqoop-setup`

Bright Cluster Manager provides `cmhadoop-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 `cmhadoop-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/`
  - `cmhadoop-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 `cmhadoop-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 configuration overlay 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.

The options for `cmhadoop-sqoop-setup` are listed on running `cmhadoop-sqoop-setup -h`.

An Example Run With `cmhadoop-sqoop-setup`

The option `-j` <*path*> is mandatory. It is used to set the Java Home in Sqoop environment files. The path should point to a Java Development Kit.

Example

```
[root@bright81 ~]# cmhadoop-sqoop-setup -i hdfs1 -j /usr/lib/jvm/java-1.7.0-op\
enjdk.x86_64/ -t /tmp/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz --master node005
Using MySQL Connector/J installed in /usr/share/java/
Sqoop release '1.4.6.bin__hadoop-2.0.4-alpha'
Sqoop service will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.7.1
Sqoop being installed... done.
Creating directories for Sqoop... done.
Creating module file for Sqoop... done.
Creating configuration files for Sqoop... done.
Updating images... done.
Updating configuration in CMDaemon... done.
Installation successfully completed.
Finished.
```

Extra Jar Files Needed When Using Sqoop With MySQL For Some Hadoop Versions

When using one the following versions of Hadoop:

- Cloudera CDH 5.7.x and later
- Hortonworks HDP 2.5.x and later

additional jar files are needed to import data with Sqoop, namely:

- `json-20160212.jar`, available at: `https://repo1.maven.org/maven2/org/json/json/20160212/json-20160212.jar`
- `commons-lang3-3.4.jar`, available at: `https://repo1.maven.org/maven2/org/apache/commons/commons-lang3/3.4/commons-lang3-3.4.jar`

Both files should be copied to the Sqoop `/lib` subdirectory of the Sqoop installation.

Example

```
[root@bright81 ~]# cmsh
[bright81]% hadoop; use hdfs1; get installationdirectoryforsqoop
/cm/shared/apps/hadoop/Cloudera/sqoop-1.4.6-cdh5.7.5/
[bright81->hadoop[hdfs1]]% quit
[root@bright81 ~]# cp json-20160212.jar /cm/shared/apps/hadoop/Cloudera/sqoop\
-1.4.6-cdh5.7.5/lib
[root@bright81 ~]# cp commons-lang3-3.4.jar /cm/shared/apps/hadoop/Cloudera/sqoop\
-1.4.6-cdh5.7.5/lib
```

7.10.2 Sqoop Removal With `cmhadoop-sqoop-setup`

`cmhadoop-sqoop-setup` uses the `-u` option to remove the Sqoop instance.

Example

```
[root@bright81 ~]# cmhadoop-sqoop-setup -u hdfs1
Requested removal of Sqoop for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Sqoop directories... done.
Removal successfully completed.
Finished.
```

7.10.3 Using Sqoop To Import A Table From MySQL

The following example shows how to import data from the MySQL instance installed on the head node. First, the following statements must be executed explicitly by the administrator, using a MySQL client:

```
GRANT SELECT ON cmdaemon.* TO 'sqoop'@'%%' IDENTIFIED BY 'sqoop';
FLUSH PRIVILEGES;
```

Now that user `sqoop` has been granted access to MySQL, it's possible to proceed with the import (some lines elided):

```
[root@bright81 ~]# module load sqoop/hdfs1
[root@bright81 ~]# sqoop import --connect jdbc:mysql://hadoop.cm.cluster/cmdae\
mon --username sqoop --password sqoop --table Devices --direct --verbose --fet\
ch-size 0

...

15/12/16 14:17:33 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
15/12/16 14:17:33 DEBUG tool.BaseSqoopTool: Enabled debug logging.

...
```

```
15/12/16 14:17:34 INFO manager.SqlManager: Executing SQL statement: SELECT t.*\
 FROM `Devices` AS t LIMIT 1

...

15/12/16 14:17:34 DEBUG orm.ClassWriter: Writing source file: /tmp/sqoop-root/\
compile/732ef860a1294c9c074a7d963519fe5c/Devices.java
15/12/16 14:17:34 DEBUG orm.ClassWriter: Table name: Devices
15/12/16 14:17:34 DEBUG orm.ClassWriter: Columns: uniqueKey:-5, revision:12, r\
eadonly:-7, tag:12, hostname:12, mac:12, creationTime:-5, partition:-5, ethern\
etSwitch:-5, rack:-5, rackPosition:-5, rackHeight:-5, indexInsideContainer:-5,\
 powerControl:12, customPowerScript:12, customPowerScriptArgument:12, customPi\
ngScript:12, customPingScriptArgument:12, notes:-1, userdefined1:12, userdefin\
ed2:12, derivedMonConfId:-5,
15/12/16 14:17:34 DEBUG orm.ClassWriter: sourceFilename is Devices.java
15/12/16 14:17:34 DEBUG orm.CompilationManager: Found existing /tmp/sqoop-root\
/compile/732ef860a1294c9c074a7d963519fe5c/
15/12/16 14:17:34 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /cm/share\
d/apps/hadoop/hdfs1

...

15/12/16 14:18:04 INFO mapreduce.ImportJobBase: Transferred 875 bytes in 26.16\
34 seconds (33.4437 bytes/sec)
15/12/16 14:18:04 INFO mapreduce.ImportJobBase: Retrieved 7 records.
15/12/16 14:18:04 DEBUG util.ClassLoaderStack: Restoring classloader: sun.misc\
.Launcher$AppClassLoader@21533b2c
```

Sqoop creates a directory inside HDFS, and it saves the result of the import operation, which can be shown with:

```
[root@bright81 ~]# module load hadoop/hdfs1
[root@bright81 ~]# hdfs dfs -cat /user/root/Devices/*
38654705665,,0,00000000a000,switch01,00:00:00:00:00:00,1444115644,21474836481,\
281474976710713,0,0,1,0,apc,,,,,,,,81604382722
38654705666,,0,00000000a000,hadoop,FA:16:3E:23:45:3C,1448905041,21474836481,28\
1474976710714,0,0,1,0,apc,,,,,,,,81604382723
38654705667,,0,00000000a000,node001,FA:16:3E:D7:3F:95,1448905042,21474836481,2\
81474976710742,0,0,1,0,apc,,,,,,,,81604382724
38654705668,,0,00000000a000,node002,FA:16:3E:FE:8E:EA,1448905041,21474836481,2\
81474976710745,0,0,1,0,apc,,,,,,,,81604382725
38654705669,,0,00000000a000,node003,FA:16:3E:60:38:36,1448905041,21474836481,2\
81474976710748,0,0,1,0,apc,,,,,,,,81604382726
38654705670,,0,00000000a000,node004,FA:16:3E:6A:03:D6,1448905042,21474836481,2\
81474976710751,0,0,1,0,apc,,,,,,,,81604382727
38654705671,,0,00000000a000,node005,FA:16:3E:07:28:C1,1448905042,21474836481,2\
81474976710754,0,0,1,0,apc,,,,,,,,81604382728
```

7.11 Sqoop2

Sqoop2 is the forthcoming version of Sqoop, designed to support data transfer across any two data sources. Bright Cluster Manager provides `cmhadoop-sqoop-setup` for Sqoop2 installation, as well as for Sqoop installation. The `cmhadoop-sqoop-setup` behavior follows the same pattern as described in the Sqoop section (section 7.10).

An Example Run With `cmhadoop-sqoop-setup`

The option `-j` <*path*> is mandatory. It is used to set the Java home path in Sqoop2 environment files. The path should point to a Java Development Kit.

Example

```
[root@bright81 ~]# cmhadoop-sqoop-setup -i hdfs1 -j /usr/lib/jvm/java-1.7.0-op\
 enjdk.x86_64/ -t /tmp/sqoop-1.99.6-bin-hadoop200.tar.gz --master node005
Using MySQL Connector/J installed in /usr/share/java/
Sqoop release '1.99.6-bin-hadoop200'
Sqoop service will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.7.1
Sqoop being installed... done.
Creating directories for Sqoop... done.
Creating module file for Sqoop... done.
Creating configuration files for Sqoop... done.
Updating images... done.
Updating configuration in CMDaemon... done.
Validating Sqoop setup... done.
Installation successfully completed.
Finished.
```

7.11.1 Sqoop2 Removal With `cmhadoop-sqoop-setup`

`cmhadoop-sqoop-setup` uses the `-u` option to remove the Sqoop2 instance.

Example

```
[root@bright81 ~]# cmhadoop-sqoop-setup -u hdfs1
Requested removal of Sqoop for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Sqoop directories... done.
Removal successfully completed.
Finished.
```

7.12 Storm

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

Other comparisons 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

7.12.1 Storm Installation With `cmhadoop-storm-setup`

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

Prerequisites For Storm Installation, And What Storm Installation Does

The following applies to using `cmhadoop-storm-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 `cmhadoop-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 creates two dedicated configuration overlays for Storm: one for the Storm Nimbus and one for Storm Supervisors.
- 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.

The options for `cmhadoop-storm-setup` are listed on running `cmhadoop-storm-setup -h`.

An Example Run With `cmhadoop-storm-setup`

The option `-j <path>` is not mandatory. It is used to set the Java Home in Storm environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright81 ~]# cmhadoop-storm-setup -i hdfs1 -t /tmp/apache-storm-0.10.0.tar.gz  \
 --nimbus node005
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Storm release '0.10.0'
Storm Nimbus and UI services will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.7.1
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... done.
Initializing Nimbus services for Storm... done.
Updating configuration in CMDaemon... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

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

7.12.2 Storm Removal With `cmhadoop-storm-setup`

`cmhadoop-storm-setup` uses the `-u` option to remove the Storm instance.

Example

```
[root@bright81 ~]# cmhadoop-storm-setup -u hdfs1
Requested removal of Storm for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Cleaning ZooKeeper... done.
Removing additional Storm directories... done.
Removal successfully completed.
Finished.
```

7.12.3 Using Storm

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

```
[root@bright81 ~]# module load storm/hdfs1
[root@bright81 ~]# storm jar $STORM_BASE_DIR/examples/storm-starter/\
 storm-starter-topologies-*.jar\
 storm.starter.WordCountTopology WordCount2

...

638  [main] INFO  b.s.u.Utils - Using defaults.yaml from resources
745  [main] INFO  b.s.u.Utils - Using storm.yaml from resources
819  [main] INFO  b.s.u.Utils - Using defaults.yaml from resources
847  [main] INFO  b.s.u.Utils - Using storm.yaml from resources
851  [main] INFO  b.s.StormSubmitter - Generated ZooKeeper secret payload for MD5-\
digest: -6720275370401821887:-5556780662562789347
853  [main] INFO  b.s.s.a.AuthUtils - Got AutoCreds []
871  [main] INFO  b.s.u.StormBoundedExponentialBackoffRetry - The baseSleepTimeMs \
[2000] the maxSleepTimeMs [60000] the maxRetries [5]
884  [main] INFO  b.s.u.StormBoundedExponentialBackoffRetry - The baseSleepTimeMs \
[2000] the maxSleepTimeMs [60000] the maxRetries [5]
914  [main] INFO  b.s.u.StormBoundedExponentialBackoffRetry - The baseSleepTimeMs \
[2000] the maxSleepTimeMs [60000] the maxRetries [5]
921  [main] INFO  b.s.StormSubmitter - Uploading topology jar /cm/shared/apps/hado\
op/Apache/apache-storm-0.10.0//examples/storm-starter/storm-starter-topologies-0.1\
0.0.jar to assigned location: /tmp/storm-hdfs1-local/nimbus/inbox/stormjar-6b72206\
e-237a-4aca-8d1d-dcf1e93ec2f1.jar
Start uploading file '/cm/shared/apps/hadoop/Apache/apache-storm-0.10.0//examples/\
storm-starter/storm-starter-topologies-0.10.0.jar' to '/tmp/storm-hdfs1-local/nimb\
us/inbox/stormjar-6b72206e-237a-4aca-8d1d-dcf1e93ec2f1.jar' (3305718 bytes)
[==================================================] 3305718 / 3305718
File '/cm/shared/apps/hadoop/Apache/apache-storm-0.10.0//examples/storm-starter/st\
orm-starter-topologies-0.10.0.jar' uploaded to '/tmp/storm-hdfs1-local/nimbus/inbo\
x/stormjar-6b72206e-237a-4aca-8d1d-dcf1e93ec2f1.jar' (3305718 bytes)
1002 [main] INFO  b.s.StormSubmitter - Successfully uploaded topology jar to assig\
ned location: /tmp/storm-hdfs1-local/nimbus/inbox/stormjar-6b72206e-237a-4aca-8d1d\
-dcf1e93ec2f1.jar
1002 [main] INFO  b.s.StormSubmitter - Submitting topology WordCount2 in distribut\
ed mode with conf "storm.zookeeper.topology.auth.scheme":"digest","storm.zookeeper\
.topology.auth.payload":"-6720275370401821887:-5556780662562789347","topology.work\
ers":3,"topology.debug":true
1168 [main] INFO  b.s.StormSubmitter - Finished submitting topology: WordCount2

[root@bright81 ~]# storm list
```

```
...

Topology_name        Status     Num_tasks  Num_workers  Uptime_secs
-------------------------------------------------------------------
WordCount2           ACTIVE     28         3            15
```

8

Securing Hadoop

By default Hadoop assumes that the entire network of machines and users is trusted, and has few security features enabled. This is because security was considered mostly after Hadoop became popular.

Without security features enabled, possible risks include identity spoofing, local (data at rest) access, and transmitted data (data in motion) snooping.

The security layer provided by Bright Cluster Manager focuses on security that is be configured within Hadoop and related tools. It includes Kerberos, because Hadoop relies heavily on Kerberos, and also includes native libraries used by Hadoop. It excludes firewall rules beyond those directly needed for the included security changes, and also excludes configuring full disk encryption.

There are three security features that can be enabled as an option or a reasonably straightforward change:

- Kerberos: can be enabled/disabled independently
- SSL: can be enabled/disabled independently
- Wire-encryption: dependent on Kerberos and SSL.

Other features require manual changes. Some of these others are covered in the manual:

- Data at rest encryption (section 8.6.4)
- Authorization ACLs (not covered)
- Auditing (not covered)

8.1 Security Setup Support Matrix

The following table indicates the support within Linux distributions for Hadoop flavors and the associated Hadoop components.

Within the table the ✓ indicates that compatibility has been tested and components have been observed to work together without major issues, while the ✗ indicates some issue. Issues are elaborated upon in annotations later on, after the table.

Table 8.1: Compatibility Matrix For Hadoop Flavors And The Associated Hadoop Components

Hadoop setup	CentOS6			CentOS7			SLES12		
	Apache	Hortonworks HDP	Cloudera CDH	Apache	Hortonworks HDP	Cloudera CDH	Apache	Hortonworks HDP	Cloudera CDH
Hadoop	✓	✓	✓	✓	✓	✓	✓	✓	✓
NameNode HA	✓	✓	✓	✓	✓	✓	✓	✓	✓
Yarn HA	✓	✓	✓	✓	✓	✓	✓	✓	✓
NN & Yarn HA	✓	✓	✓	✓	✓	✓	✓	✓	✓
Federation	✓	✓	✓	✓	✓	✓	✓	✓	✓
KMS	✓	✓	✓	✓	✓	✓	✓	✓	✓
ZooKeeper	✓	✓	✓	✓	✓	✓	✓	✓	✓
HBase[a]	✓	✓	✓	✓	✓	✓	✓	✓	✓
Spark	✓	✓	✓	✓	✓	✓	✓	✓	✓
Hive[b]	✓	✓	✓	✓	✓	✓	✓	✓	✓
Beeline[c]	✓	✓	✓	✗[1]	✗[1]	✗[1]	✗[1]	✗[1]	✗[1]
Sqoop	✓	✓	✓	✓	✓	✓	✓	✓	✓
Pig	✓	✓	✓	✓	✓	✓	✓	✓	✓
Kafka	✓	✓	✓	✓	✓	✓	✓	✓	✓
Accumulo	✓	✓	✓	✓	✓	✓	✓	✓	✓
Drill >= 1.10.x	✓	✓	✓	✓	✓	✓	✓	✓	✓

Annotations To Table 8.1

Known error issues:

- 1: The `beeline` commandline utility on CentOS7 and SLES12 may give errors renewing tickets when using it with Hive.

Other annotations:

- a: HBase Master installed on the head node with `cm-hadoop-setup`.
- b: Hive tested with `hive` commandline utility only.
- c: Hive tested with `hive` and `beeline` commandline utilities.

The support tests are only executed for the supported latest tarball versions of Apache Hadoop, Cloudera CDH, and Hortonworks HDP. The version support matrix (section 1.4) lists the latest supported tarball versions of these flavors.

Each column of the table refers to such a tarball flavor. For each column, the support for the tools associated with the flavor, such as HBase, ZooKeeper, Spark, and so on, is indicated. The appropriate version of the associated tool must be the one listed in the version support matrix for the latest supported tarball version of the flavor.

Thus, for example, if the latest flavor version of Apache Hadoop is Apache Hadoop *x.y* in the support matrix, then the Apache HBase version must be the version listed in that support matrix for Apache Hadoop *x.y*.

8.2 Supported Configurations

Enabling security in Hadoop is currently supported for a subset of Linux and Hadoop distributions.

Linux flavor	Status
CentOS 6, Red Hat Enterprise Linux Server 6, Scientific Linux 6	Supported with both OpenJDK and Oracle Java 7 Virtual Machines (HotSpot).
CentOS 7, Red Hat Enterprise Linux Server 7, Scientific Linux 7	Supported with both OpenJDK and Oracle Java 7 Virtual Machines (HotSpot).
SUSE Linux Enterprise Server 12	Supported *only* with the Oracle Java Virtual Machine.

The Oracle JVM is recommended because in the experience of Bright Computing the Hadoop components work better out of the box compared with OpenJDK or IBM Java Runtime Environment. For SLES12, Oracle JVM currently (December 2016) turns out to be the only JVM that works without needing patch scripts to get security features working, or to get some other advanced features working such as Yarn HA combined with NameNode HA.

Security requires the Java Cryptography Extension (JCE) for Java 7 to provide better encryption capabilities for the JVM. This is included in OpenJDK by default. If the administrator plans on using HotSpot with Oracle JVM, then the JCE must be downloaded from Oracle.

Installing Oracle HotSpot JVM

- The Oracle JVM can be installed using the instructions from the Knowledgebase article at: `http://kb.brightcomputing.com/faq/index.php?action=artikel&cat=18&id=196`. The Oracle JVM is installed, for example, at `/cm/shared/apps/java/current`.
- The file `UnlimitedJCEPolicyJDK7.zip` can be downloaded from: `http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html`.

  It should be unzipped, and the `README.txt` should be followed. At the time of writing it suggested copying a jar file, for example:

  ```
  cp *.jar /cm/shared/apps/java/current/lib/security/
  ```

When installing Hadoop via Bright View or `cm-hadoop-setup` it is possible to change the suggested `JAVA_HOME` to the location where the Oracle JVM is installed (`/cm/shared/apps/java/current`).

Hadoop flavor	Status
Apache Hadoop 1	Will not be supported.
Apache Hadoop 2	Supported for 2.7.x and above
Cloudera	Supported for 5.7.x and above
Hortonworks	Supported for HDP 2.5.x and above

Hadoop setup	Status
Single NameNode, Single Yarn	Supported
NameNode HA, Single Yarn	Supported
Single NameNode, Yarn HA	Supported
NameNode HA, Yarn HA	Supported
Federation	Supported

Hadoop component	Status
HDFS	Supported (section 8.4.1)
YARN	Supported (section 8.4.2)
KMS	Supported (section 8.4.3)
Job History	Supported.
Timeline	Supported, but is disabled with HBase, section 8.4.4 and disabled in case of Cloudera and Yarn HA.
ZooKeeper	Supported (section 8.4.5)
HBase	Supported (Without Timeline, Without SSL, section 8.4.4)
HBase	Not supported on SLES12. *Apache HBase distributions come packaged with relatively old versions of, for example,* `hadoop-common.jar`*. This may cause Kerberos ticket renewals to fail. It can be fixed with some manual effort, as explained in a Bright Computing knowledgebase article for HBase.* *The HBase distributions packaged by Cloudera and Hortonworks are not affected.*
Hive, Pig	Supported.
Kafka	Supported (section 8.4.6)
Flink	Supported version 1.1.4. (section 8.4.7)
Drill	Supported version 1.10.0. (section 8.4.8)
Spark on Yarn	Supported.
Spark standalone	Not supported.

Hadoop component	Status
Accumulo, Alluxio, Giraph, Ignite, Impala, Sqoop, Storm, Tez	Support is planned.
Giraph, Zeppelin	Not supported by the project.

8.3 Enabling/Disabling Security Features

Configuring Hadoop security features (enabling and disabling them) is carried out with the Bright Cluster Manager `cmhadoop-security-setup` script.

Prerequisites before installation:
Hadoop has the following requirements that need to be considered before starting the installation of security features:

- Time synchronization should be working properly. Kerberos relies on the time being synced across the nodes. The script only does a very basic precondition check using `ntpstat` or `ntptime` (depending on the OS), and the administrator should ensure there are no issues.
- There is no Kerberos server installed on the head node. The setup script assumes it can install the packages and own the configuration files.
- It should be possible to install the Kerberos client on the compute nodes
- Security changes require a restart of all services related to the Big Data instance. Restarts will be executed by the script after enabling/disabling.

Tips For Enabling/Disabling Security Features:

- Securing multiple Hadoop instances within one cluster is currently supported only within one Kerberos realm, and only works if the regular nodes do not overlap.
- Carrying out an installation gradually, and checking at each stage is sensible. For example: after Kerberos is enabled, the cluster should be checked to see if it is still able to run jobs. After that SSL can be enabled, and things checked once again, and so on. If there is an error, then it is most likely due to the last enabled feature.
- During configuration the setup script stops services when needed, but using the cluster can make this step more error prone. It is therefore recommended to stop all jobs in the Hadoop instance before enabling or disabling security components.

  For example: Securing HBase requires some tedious ACL changes in ZooKeeper. Changing the security enabled/disabled states using the script carries out these ACL changes, but the cluster has no access to its data as these changes are implemented. If the administrator does end up with the cluster stuck in a state where the data cannot be accessed, then the cluster needs to be re-secured with the `--recover` flag (section 8.3.2).

8.3.1 Using `cmhadoop-security-setup`

This `cmhadoop-security-setup` script is executed on the head node as root. It has options as indicated by the following table:

Script parameters	Description
`-i hdfs1 --kerberos`	Install Kerberos if configuration not already present
`-i hdfs1 --kerberos --force`	Install Kerberos even if configuration already present
`-i hdfs1 --regenerate-keytabs`	Regenerate Kerberos keytabs only
`-u hdfs1 --kerberos`	Uninstall only Kerberos if present
`-i hdfs1 --ssl`	Install SSL certificates and enable `HTTPS_ONLY`
`-u hdfs1 --ssl`	Uninstall SSL certificates and enable `HTTP_AND_HTTPS`
`-i hdfs1 --ssl --wire-encryption`	Enable wire-encryption to enable encrypted shuffle and secure RPC calls
`-u hdfs1 --ssl --wire-encryption`	Enable wire-encryption to enable encrypted shuffle and secure RPC calls
`--recover`	Add this flag to preceding commands only in certain cases (section 8.3.2)
`-s hdfs1`	Show currently enabled or disabled components
`--test hdfs1`	Perform some tests related to security that are expected to succeed

Options can be combined, for example:
`-i --kerberos --ssl`
The `-u` option only disables (uninstalls) the specified security feature. Thus:
`-u --ssl`
does not uninstall Kerberos.
Some further examples follow:

1. Viewing the security configuration of Hadoop:

Example

```
[root@bright81 ~]# cmhadoop-security-setup -s hdfs1
Found Hadoop instance 'hdfs1', release: 2.7.2
Security features:
----------------------------------------
[x] kerberos (enabled)
[ ] ssl
[x] wire_encryption (enabled)
----------------------------------------
```

2. Installation run of Kerberos to Hadoop:

Example

```
[root@bright81 ~]# cmhadoop-security-setup -i hdfs1 --kerberos
Found Hadoop instance 'hdfs1', release: 2.7.2
Please provide password for Kerberos master: ********
Please repeat this password a second time: ********
```

```
Installing Kerberos server+client packages on the headnode...
Installing Kerberos client packages in involved software images.
Installing Kerberos client config files in software images.
Updating images...
Removing Kerberos related files
Initializing Kerberos Key Distribution Center principal database.
In case random device entrophy is low, this can take a while.
Adding admin user in Kerberos Key Distribution Center.
Generating intermediate keytabs...
Generating intermediate keytabs... done.
Generating definitive keytabs for distribution...
Generating definitive keytabs for distribution... done.
Distributing keytabs to nodes...
Distributing keytabs to nodes... done.
User 'sectest' already exists... removing..
Creating user 'sectest'..
Waiting for user to become available within HDFS..
HDFS access is granted to user sectest with authentication to Kerberos... ok.
Waiting 30 seconds to give CMDaemon some time to write out configuration...
Waiting 30 seconds to give CMDaemon some time to write out configuration... done
Stopping all services for HDFS...
Stopping all services for HDFS... done.
Comitting security settings to CMDaemon...
Comitting security settings to CMDaemon... done.
Finished.
```

3. Uninstalling Kerberos from Hadoop:

Example

```
[root@bright81 ~]# cmhadoop-security-setup -u hdfs1 --kerberos
2.7.1.2.4.2.0-258
Found Hadoop instance 'hdfs1', release: 2.7.1.2.4.2.0-258
Removing Kerberos security from configuration...
Stopping HBase services...
Stopping HBase services... done.
Wait for HBase services to shutdown..
Wait for HBase services to shutdown.. done.
Authenticating as HBase to Kerberos...
Authenticating as HBase to Kerberos... done.
Making ACL less strict on HBase paths in ZooKeeper...
Making ACL less strict on HBase paths in ZooKeeper... done.
HBase zkCli disabled strict ACL on paths in Zookeeper
Stopping all services for HDFS...
Stopping all services for HDFS... done.
Removing Kerberos related files
Finished.
```

By default the script timeout for stopping HBase is set to 60 seconds. Sometimes HBase can take longer than that. If the preceding output shows that HBase fails to stop, then simply trying again a minute or so later is likely to succeed.

4. Uninstalling SSL encryption from Hadoop:

Example

```
[root@bright81 ~]# cmhadoop-security-setup -u hdfs1 --wire-encryption
Found Hadoop instance 'hdfs1', release: 2.7.2
Removing wire encryption from configuration...
Restarting all services for HDFS...
Restarting all services for HDFS... done.
Finished.
```

8.3.2 Using cmhadoop-security-setup --recover

Adding security features to Hadoop with the cmhadoop-security-setup involves many steps that have to be executed in a specific order. It is possible that for some cluster configurations a step may fail before the script finishes, leaving the cluster in an invalid state. A failure or hung state can occur if some services are in an invalid state, and are not responding properly because of that state.

All the cmhadoop-security-setup commands are idempotent if they are successful. That is, it is safe to run all cmhadoop-security-setup commands more than once. If however the script keeps failing or hangs, then adding the --recover flag might help.

Sometimes an additional --force flag is required, and suggested by the script.

A Re-install

A re-install can be carried out, even when the instance is in an invalid state, by using the exact same command as in normal usage, but with the added option: --recover:

Example

```
[root@bright81 ~]# cmhadoop-security-setup -i hdfs1 --kerberos --recover
...
```

An Uninstall

An uninstall can be carried out, even when the instance is in an invalid state, by using the exact same command as normal usage, but with the added option: --recover.

Example

```
[root@bright81 ~]# cmhadoop-security-setup -u hdfs1 --kerberos --recover
...
```

8.3.3 Using The cmsh Security Submode

Within the hadoop mode of cmsh, there is a security submode for each instance:

Example

```
[root@bright81 ~]# cmsh
[bright81]% hadoop
[bright81->hadoop]% use hdfs1
[bright81->hadoop[hdfs1]]% security
[bright81->hadoop[hdfs1]->security]%
```

The show command in security mode displays the current Hadoop security configuration values. These are read-only values.

Example

```
[bright81->hadoop[hdfs1]->security]% show
Parameter                      Value
------------------------------ ----------------------------------
Revision
Kerberos enabled               yes
SSL enabled                    no
Wire encryption enabled        no
```

Commands that are particular to the `security` submode are also available:

Command	Description
`regeneratekeytabs`	Regenerates keytabs in case Kerberos is installed.
`uninstallkerberos`	Uninstalls Kerberos in case Kerberos is installed.
`enablessl`	Enables SSL if not already installed.
`disablessl`	Disables SSL if it is not already installed.
`enablewireencryption`	Enables wire encryption if not already installed.
`disablewireencryption`	Disables wire encryption if not already installed.

A user must have the correct permissions to execute these commands:

It should be noted that there is no command in `security` mode to install Kerberos. This is because initializing the KDC (Kerberos Key Distribution Center) requires some input, such as the KDC password. To install Kerberos, the `cmhadoop-security-setup` script (section 8.3) must be used.

8.4 How Individual Components Are Secured

8.4.1 HDFS

HDFS startup scripts use the `jsvc` utility—and not `java`—to launch the DataNode (secure) service.

Also DataNodes use privileged ports—ports below number 1024, which only `root` can open. That is why `jsvc` is no longer executed as a `hdfs`-owned process, but with `root`-ownership. However `jsvc` drops its privileges after opening the privileged ports.

8.4.2 YARN

If Hadoop security features are set up, then YARN uses the `container-executor` utility to launch containers/jobs. The utility uses some lines of configuration from `container-executor.cfg`.

The `container-executor` utility enforces strict permissions so that YARN jobs have appropriately restricted access to files, folders, and even caches. The restrictions are according to their user/group privileges.

By default, the users `hdfs`, `yarn`, and `mapred` are not allowed to run jobs. Users with a UID below `1000` are also forbidden from doing so. This is why users like `hbase`, with a UID below 1000, are not allowed to run YARN jobs with security features set up.

8.4.3 KMS

The Key Management Server (KMS) is supported with and without HTTPS.

The ports used (16000 and 16001 by default) are not different. The only difference is that Tomcat needs a different `server.xml`, one that enables SSL and configures the keystore.

The `cmhadoop-security-setup` script replaces `server.xml` with a symlink pointing to the proper configuration, and makes sure the contents of this file and file permissions are properly configured.

8.4.4 HBase

HBase stores much of its information within ZooKeeper, and by default this is basically world readable. However, on enabling the Kerberos security feature, ACLs to access this data become more restrictive.

The `hbase` user is by default an administrator that is allowed to manage these permissions. Authenticating as the `hbase` user is made possible by the `hbase.keytab` configuration within the Hadoop configuration.

Timeline is not supported by default and is therefore be disabled when securing HBase. When disabling Hadoop security it will be re-enabled again.

HBase uses SASL to communicate with ZooKeeper. The principal and keytab to use are defined in the JAAS (Java Authentication and Authorization Service) files.

Command	Description
`hbase shell`	Launch HBase administration shell. (uses `hbase/hbase.jaas`)
`hbase zkcli`	Launch ZooKeeper client via HBase command. (uses `hbase/hbase.jaas`)

8.4.5 ZooKeeper

ZooKeeper uses SASL. The principal and keytab to use are defined in the JAAS files.

Command	Description
`zkCli.sh -server <host>:2181`	Launch ZooKeeper administration shell.

8.4.6 Kafka

Currently the *Kerberos* and *SSL* security features works with Kafka (*Wire-encryption* is already enabled with *SSL*). Enabling Kerberos means it is necessary for applications to be authenticated against Kerberos, Kafka services among themselves authenticate with Kerberos and finally also the communication between Kafka and ZooKeeper.

For long running applications it is generally recommended to create keytabs per application (and configure it to automatically renew tickets). For simpler applications it is also possible to simply use the TGT Cache from the environment, this is configured by default by Bright.

Kafka uses a plaintext protocol by default, using Kerberos makes it use SASL (but still in plaintext). Enabling SSL will change this to SASL over SSL. As with ZooKeeper and HBase SASL with JAAS configuration files are used to configure which keytabs and principals should be used in which case. For convenience a `kafka-client.jaas` is created which provides all configuration needed for running the bundled scripts below.

Command	Description
`kafka-topics.sh`	Kafka admin tool for topics uses the `KafkaServer` and `Client` directives from `kafka/kafka-client.jaas`.
`kafka-console-producer.sh`	Kafka console producer uses the `KafkaClient` directive from `kafka/kafka-client.jaas`.
`kafka-console-consumer.sh`	Kafka console consumer uses the `KafkaClient` directive from `kafka/kafka-client.jaas`.

When security is enabled, loading the module file for Kafka will set the environment variable `$KAFKA_OPTS`, such that the client jaas file is provided to above scripts.

Example

```
[root@bright81 ~]# module load kafka/hdfs1/Apache/0.10.1.0
[root@bright81 ~]# echo $KAFKA_OPTS
-Djava.security.auth.login.config=/etc/hadoop/hdfs1/kafka/kafka-client.jaas
```

All Kafka services use `KafkaServer` and `Client` from `kafka/kafka.jaas`. This is a separate file because it is also possible to approach JAAS files differently for applications. The official documentation for Kafka is pretty useful explaining the use of JAAS files: `http://kafka.apache.org/documentation/#security_overview`.

A full example running a Kafka console consumer and producer from the Headnode, with security enabled, assuming a Kafka server is also running on the Headnode.

Example

```
# Create a topic
[root@bright81 ~]# module load kafka/hdfs1/Apache/0.10.1.0
[root@bright81 ~]# kafka-topics.sh --zookeeper $(hostname -f):2181/kafka-hdfs1 --create
# Start a producer
[root@bright81 ~]# kafka-console-producer.sh --broker-list $(hostname -f):9092

# In another terminal, start a consumer
[root@bright81 ~]# kafka-console-consumer.sh --bootstrap-server $(hostname -f):9092

# Mark topic for deletion
[root@bright81 ~]# kafka-topics.sh --zookeeper $(hostname -f):2181/kafka-hdfs1 --delete
```

With the above example the user can type some text in the producer (or pipe some data via stdin), and the consumer should receive each line as a message.

The user needs to be authenticated against Kerberos for this to work. For this the default generated user `sectest` on the Headnode can be tried.

8.4.7 Flink

Flink can be secured when deployed. It can also be deployed on a secured cluster.

Support for further securing Flink services themselves will be added in the near future. For now using Flink on a Kerberized cluster is the same as for a non-Kerberized cluster, except that authentication against Kerberos is needed.

At the time of writing (October 2017), Flink 1.2 was just released, adding more robust support for Kerberos. Support for this new Flink feature is on the roadmap for Bright Cluster Manager.

For now, Flink 1.2 works, but runs "insecurely" on a Kerberized cluster.

For secure use, version 1.1.4 is recommended at the time of writing.

8.4.8 Drill

Drill is supported from version 1.10.0 onwards.

In Bright Cluster Manager Drill is configured with Kerberos according to the instructions in the Drill manual `https://drill.apache.org/docs/configuring-kerberos-authentication/`.

The installation of Drill should be carried out before securing the cluster.

This is because Bright Cluster Manager does not configure PAM for Drill by default. Since the Drill web interface in 1.10.0 does not support “KERBEROS” authentication, and only supports “PLAIN” (with PAM), it means that Drill therefore disables the web interface at startup. This then means that the security setup script cannot properly create the HDFS storage connection using the REST API.

With some manual effort it may be possible to deploy Drill directly on top of an already-secured cluster. However, the easiest way to do it is to uninstall security temporarily, install Drill, and then re-install security.

Example

First, what principal to use for Drill with `kinit` from the `drill-admin.keytab` must be determined. Then, as the `drill` user, authentication to Kerberos should be carried out. The query can then be executed using `sqlline`:

```
[root@bright81 tmp]# klist -k -t /etc/hadoop/hdfs1/drill-admin.keytab
Keytab name: FILE:/etc/hadoop/hdfs1/drill-admin.keytab
KVNO Timestamp           Principal
---- ------------------- ------------------------------------------------------
2 06/09/2017 12:09:48 drill/node006.cm.cluster@CM.CLUSTER
2 06/09/2017 12:09:48 drill/node006.cm.cluster@CM.CLUSTER
2 06/09/2017 12:09:48 drill/node006.cm.cluster@CM.CLUSTER
2 06/09/2017 12:09:48 drill/node006.cm.cluster@CM.CLUSTER
[root@bright81 tmp]# su drill
[drill@bright81 tmp]# kinit -k -t /etc/hadoop/hdfs1/drill-admin.keytab \
drill/node006.cm.cluster@CM.CLUSTER
[drill@bright81 tmp]# module load drill/hdfs1/Apache/1.10.0
[drill@bright81 tmp]# sqlline -u "jdbc:drill:zk=node001.cm.cluster:2181;auth=kerberos"
0: jdbc:drill:zk=node001.cm.cluster:2181> SELECT * FROM dfs.`/cm/shared/apps/hadoop/\
Apache/apache-drill-1.10.0//sample-data/region.parquet`;
+--------------+--------------+-----------------------+
| R_REGIONKEY  |    R_NAME    |       R_COMMENT       |
+--------------+--------------+-----------------------+
| 0            | AFRICA       | lar deposits. blithe  |
| 1            | AMERICA      | hs use ironic, even   |
| 2            | ASIA         | ges. thinly even pin  |
| 3            | EUROPE       | ly final courts cajo  |
| 4            | MIDDLE EAST  | uickly special accou  |
+--------------+--------------+-----------------------+
```

The same query, but selected from the HDFS filesystem. `sqlline` must be run with the correct Linux user `drill`.

Example

```
0: jdbc:drill:zk=node001.cm.cluster:2181> SELECT  * FROM hdfs.`/user/drill/nation.parquet`;
+--------------+--------------+-----------------------+
| R_REGIONKEY  |    R_NAME    |       R_COMMENT       |
+--------------+--------------+-----------------------+
| 0            | AFRICA       | lar deposits. blithe  |
| 1            | AMERICA      | hs use ironic, even   |
| 2            | ASIA         | ges. thinly even pin  |
| 3            | EUROPE       | ly final courts cajo  |
| 4            | MIDDLE EAST  | uickly special accou  |
+--------------+--------------+-----------------------+
```

Troubleshooting If the following error is seen, then it is possible that Kerberos authentication was carried out properly, but that `sqlline` is not running as the `drill` Linux user.

```
0: jdbc:drill:zk=node001.cm.cluster:2181> SELECT * FROM hdfs.`/user/drill/nation.parquet`;
Error: RESOURCE ERROR: Failed to create schema tree.
```

8.4.9 Spark On Yarn

Spark on Yarn can be deployed before or after securing the cluster.

To secure Spark, enabling wire-encryption is recommended. Other than this recommendation, there are no additional Spark-specific security features enabled.

Setting Up Users For Testing Of YARN Jobs And Data Storage

Spark provides a `spark-defaults.conf` file for `spark-submit`. This is a default values configuration file that holds some defaults that are managed by Bright Cluster Manager in `/etc/hadoop/<instance_name>/spark/`.

Bright Cluster Manager uses `spark-submit` to do some very simple validation. The default user specified in the defaults file is `sparktest@CM.CLUSTER`. The setup script makes sure this user and associated keytab exist, and tests if they are able to run YARN jobs.

If different principals are used that need to store data on `HDFS` first and then run Spark on Yarn jobs on this data, then it is likely that the values in the defaults file need to be overwritten.

Alternatively, if other principal/keytab values need to be used instead of the ones from the default file, then the command line options of `spark-submit` can be used. That is, the default file values can be overridden, instead of overwritten, by running `spark-submit` as follows:

Example

```
module load spark
spark-submit --master yarn --deploy-mode client \
             --principal override@CM.CLUSTER \
             --keytab /etc/hadoop/hdfs1/override.keytab \
             --class org.apache.spark.examples.SparkPi spark-examples_*.jar
```

In the preceding example, the replacement user is called `override`. The user `spark-submit` could then be removed from the CMDaemon database of Bright Cluster Manager via the `user` submode.

8.5 Background Information

This section gives a background description of the meaning of securing each individual component.

8.5.1 Kerberos

Kerberos is a system that enforces authentication.

Securely-encrypted communication channels can be established between principals. A principal is typically either a user or a service, and is designated by the form `<principal>/<host>@<realm>`.

Conceptually each principal has its own key. The key is known only to the principal itself and to the KDC (Kerberos Key Distribution Center—a central database for all the Kerberos keys).

Using a smart protocol, principals can prove to each other that they are who they claim to be, without ever exposing their key. During this process they establish a unique session key for secure communication that is only known between the two endpoints. It is designed so that even someone with access to the network cannot carry out a man-in-the-middle attack.

Service on head node	Description
`kadmin`	Kerberos 5 Password-changing and Administration
`krb5kdc`	Kerberos 5 KDC

Keytabs

A principal in Kerberos has a password that is used to unlock the key. It is possible to use the `kadmin` Kerberos client to authenticate as a given principal using this password.

It is also possible to construct a file called a keytab within the KDC. A keytab stores the key for a principal. The permissions on these keytabs should therefore be carefully set.

The setup script generates all keytabs, and it distributes them to the correct nodes, depending on the role that they fulfill within the Hadoop instance.

The principals that a keytab stores in the keytab directory can be viewed with a command such as: `klist -k -t` *<full path to keytab directory>*.

The keytabs that are generated depend on how the cluster is configured, but can include:

Keytab	Roles	Principals
kms.keytab	Key Management Server (KMS)	HTTP
hdfs.keytab	Namenode, Secondary Namenode, Datanode	hdfs, HTTP
yarn.keytab	Node Manager, Resource Manager	yarn, HTTP
mapred.keytab	Resource Manager	mapred, HTTP
hbase.keytab	HBase Server, HBase Client	hbase
zookeeper.keytab	ZooKeeper	zookeeper
cmdaemon.keytab	Bright Cluster Manager	hdfs
hbase.keytab	-	hbase
sectest.keytab	Bright Cluster Manager	sectest

SPNEGO

SPNEGO is an API used on top of Kerberos to protect certain HTTP services. It is a relatively old extension to the HTTP protocol supported by most modern browsers, as well as some text web browsers. It assumes that the <*principal*> part of the principal name (<*principal*>/<*host*>@<*realm*>) is always HTTP. The browser can construct the principal name before issuing an HTTP request, because it knows the hostname that it is requesting from, the principal (HTTP), and optionally the realm as well. The browser may ask for credentials to request a ticket or read from the environment.

8.5.2 SSL

With this security feature, HTTPS is always enforced. The Hadoop notion of "HTTP_AND_HTTPS" is not used in the configuration, and "HTTPS_ONLY" is used instead.

The setup script handles the generation of SSL certificates and the keystore.

An exception to HTTPS enforcement is the KMS Server, which still runs over plain HTTP.

8.5.3 Wire-encryption

The wire-encryption feature is straightforward to configure. It only needs the dependency on the Java Cryptography Extension (JCE), and it then provides “Unlimited Strength” encryption. Bright provides the JCE by default. However, if a third party or custom Java Virtual Machine is used for the cluster, then the extension may need to be downloaded.

Using the Oracle Hotspot JRE (or JDK) the appropriate zip file can be downloaded from Oracle. For Java 7 at the time of writing (October 2017) this is `UnlimitedJCEPolicyJDK7.zip`.

The zip file contains instructions, but there are two policy `jar` files that need to be copied to `$JAVA_HOME/lib/security`.

If Hadoop or components are being installed on a JDK, then `$JAVA_HOME/jre/lib/security` must be used. The extra `jre` subdirectory) in the path should not be missed out.

A final caveat is that the files can already be there, but they need to be overwritten in order to make it Unlimited Encryption Strength actually work.

Bright Cluster Manager provides a small utility, `TestUCE`, that can be run on a head node to verify if JCE is enabled properly:

Example

```
[root@bright81 tmp]# /path/to/bin/java -cp /cm/local/apps/cluster-tools/hadoop/java/ TestUCE
Unlimited cryptography enabled: true
```

On compute nodes the following equivalent command can be used:

Example

```
[root@bright81 tmp]# /path/to/bin/java -cp /cm/local/apps/cluster-tools/bin/ TestUCE
Unlimited cryptography enabled: true
```

Symptoms of Wire encryption malfunctioning are not always obvious, but if they occur, then they typically occur when running a job:

- java.security.InvalidKeyException: Illegal key size in application logs.
- security.UserGroupInformation: PriviledgedActionException as:sectest@CM.CLUSTER (auth:KERBEROS) cause:java.io.IOException: org.apache.hadoop.ipc.RemoteException(java.lang.NullPointerException): java.lang.NullPointerException
- java.io.IOException: Version Mismatch (Expected: 28, Received: -6646) in DataNode logs.

The encryption used is as follows:

- For Hadoop <= 2.6, 3DES encryption is enabled.
- For Hadoop >= 2.6, AES/CTR/NoPadding with a 256-bit key is enabled.
  AES offers the greatest cryptographic strength and the best performance.

8.6 Common Tasks Within A Secured Hadoop Instance

8.6.1 Kerberos Commands

Command	Description
`kadmin.local`	Interface to Kerberos administration (that is, for adding principals and so on). Typically used by `root` on the head node.
`kadmin`	Same as `kadmin.local` only via Kerberos and therefore requires credentials. Typically used on compute nodes or non-`root` users.
`klist`	List cached Kerberos tickets.
`klist -k -e -t <keytab>`	List principals inside keytab (`-k`) with extra information such as timestamps (`-t` and encryption types (`-e`) .
`kinit [ <principal> ]`	Authenticate to Kerberos to obtain and cache Ticket Granting Ticket. Default principal used is current user@<realm>.
`kinit -k -t <keytab> [ <principal> ]`	Authenticate to Kerberos to obtain and cache Ticket Granting Ticket. Default principal used is current user@<realm>.
`kdestroy`	Destroy Kerberos tickets.

8.6.2 Example: Requesting Kerberos Tickets

This example demonstrates how a Hadoop service might use Kerberos over the command line, using `curl` and the Kerberos command line utilities. The authentication is carried out as user `hdfs`. The request is carried out from the NameNode, to the KMS server, via HTTP.

A sensible first step is to destroy and verify any tickets that may still exist in cache:

Example

```
[root@bright81 tmp]# kdestroy
[root@bright81 tmp]# klist
klist: Credentials cache file '/tmp/krb5cc_0' not found
```

Next, authentication is carried out as the `hdfs` user. The `cmdaemon.keytab` is used, and `klist` is used to verify that a ticket has been obtained from the KDC Ticket Granting Service. This is just for demonstration purposes—Hadoop services each have their own keytabs with principals for the services they provide:

Example

```
[root@bright81 tmp]# kinit -k -t /etc/hadoop/hdfs1/cmdaemon.keytab hdfs
[root@bright81 tmp]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hdfs@CM.CLUSTER

Valid starting       Expires              Service principal
07/05/2016 14:07:25  07/06/2016 00:07:25  krbtgt/CM.CLUSTER@CM.CLUSTER
    renew until 07/06/2016 14:07:25
```

The `curl` request requires "`--negotiate -u :`" in order to enable it to use SPNEGO authentication. The output might look like the following.

Example

```
[root@bright81 tmp]# curl --negotiate -u : -v -XOPTIONS \
  http://node001.cm.cluster:16000/kms/v1/keys
> OPTIONS /kms/v1/keys HTTP/1.1
> User-Agent: curl/7.29.0
> Host: node001.cm.cluster:16000
> Accept: */*
>
< HTTP/1.1 401 Unauthorized
< Server: Apache-Coyote/1.1
< WWW-Authenticate: Negotiate
< Set-Cookie: hadoop.auth=; Expires=Thu, 01-Jan-1970 00:00:00 GMT; HttpOnly
< Content-Type: text/html;charset=utf-8
< Content-Length: 997
< Date: Tue, 05 Jul 2016 12:19:17 GMT
<
> OPTIONS /kms/v1/keys HTTP/1.1
> Authorization: **long string here**
> User-Agent: curl/7.29.0
> Host: node001.cm.cluster:16000
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< WWW-Authenticate: Negotiate **long string here**
< Set-Cookie: hadoop.auth="u=hdfs&p=hdfs@CM.CLUSTER&t=kerberos-dt&e=1467...
< Allow: OPTIONS,POST
< Content-Type: application/vnd.sun.wadl+xml
< Content-Length: 511
< Date: Tue, 05 Jul 2016 12:19:17 GMT
```

Running into the initial `401 Unauthorized` response multiple times is avoided by using cookies.

If Kerberos authentication had not been done, then `curl` would not have been able to successfully respond to the `401` reponse with a negotiate request using SPNEGO.

The ticket cache now has an additional service ticket. This ticket can be used until its expiry for successive requests:

Example

```
[root@bright81 tmp]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hdfs@CM.CLUSTER

Valid starting       Expires              Service principal
07/05/2016 14:07:25  07/06/2016 00:07:25  krbtgt/CM.CLUSTER@CM.CLUSTER
    renew until 07/06/2016 14:07:25
07/05/2016 14:07:30  07/06/2016 00:07:25  HTTP/node001.cm.cluster@CM.CLUSTER
    renew until 07/06/2016 14:07:25
```

8.6.3 Example: Adding A New User, Able To Run Hadoop Jobs

Currently adding a user has to be carried out in two steps.

In CMDaemon a regular user has to be created with permissions to the Hadoop HDFS instance.

Example

```
[bright81->user]% add testuser
ok.
[bright81->user*[testuser*]]% set hadoophdfsaccess hdfs1
Commit user 'testuser' ... ok.
[bright81->user[testuser]]%
```

Using the `hdfs` user, it is possible to verify if `testuser` has the correct permissions within HDFS:

Example

```
[root@bright81 tmp]# kinit -k -t /etc/hadoop/hdfs1/cmdaemon.keytab hdfs
[root@bright81 tmp]# module load hadoop/hdfs1/Apache/2.7.2
[root@bright81 tmp]# hdfs dfs -ls /user | grep testuser
drwx------   - testuser   hadoop          0 2016-07-05 14:54 /user/testuser
```

A principal can be created for the user in Kerberos:

Example

```
[root@bright81 tmp]# kadmin.local
Authenticating as principal hdfs/admin@CM.CLUSTER with password.
kadmin.local:  add_principal testuser
WARNING: no policy specified for testuser@CM.CLUSTER; defaulting to no policy
Enter password for principal "testuser@CM.CLUSTER": **********************
Re-enter password for principal "testuser@CM.CLUSTER": **********************
Principal "testuser@CM.CLUSTER" created.
kadmin.local:  quit
```

A shell can be created as the new `testuser`. Writing something to HDFS is then possible:

Example

```
[testuser@bright81 ~]$ kinit
Password for testuser@CM.CLUSTER: **********************
[testuser@bright81 ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_1003
Default principal: testuser@CM.CLUSTER

Valid starting       Expires              Service principal
07/05/2016 14:57:26  07/06/2016 00:57:26  krbtgt/CM.CLUSTER@CM.CLUSTER
    renew until 07/06/2016 14:57:24
[testuser@bright81 ~]$ module load hadoop/hdfs1
[testuser@bright81 ~]$ hdfs dfs -touchz /user/testuser/dummy.txt
[testuser@bright81 ~]$ hdfs dfs -ls /user/testuser/
Found 1 items
-rw-r--r--   3 testuser hadoop          0 2016-07-05 14:57 /user/testuser/dummy.txt
```

The user `testuser` should be ready to run YARN jobs at this point.

8.6.4 Example: Enabling Data At Rest Encryption

By default the `hdfs` user is used as the key manager user.

Example

```
[bright81 ~]# kinit -k -t /etc/hadoop/hdfs1/cmdaemon.keytab hdfs
[bright81 ~]# module load hadoop/hdfs1/Apache/2.7.2
[bright81 ~]# hadoop key create myzonekey
myzonekey has been successfully created with options Optionscipher='AES/CTR/NoPadding',\
 bitLength=128, description='null', attributes=null.
KMSClientProvider[http://node001.cm.cluster:16000/kms/v1/] has been updated.
```

A directory that is to be encrypted is created in HDFS. The encrypted zone is then set under this directory using the key.

```
[bright81 ~]# hdfs dfs -mkdir /myzone
[bright81 ~]# hdfs crypto -createZone -keyName myzonekey -path /myzone
Added encryption zone /myzone
```

The directory can be verified as being encrypted:

Example

```
[bright81 ~]# hdfs crypto -listZones
/myzone  myzonekey
```

8.6.5 Example: Granting A User Privileges Within HBase

By default the `hbase` user has administrator privileges within HBase. Bright Cluster Manager provides a `hbase.keytab` by default.

If a new user `testuser` has been created with access to HDFS, then authentication can be carried out on the head node as `hbase`. An `hbase` shell can then be opened:

Example

```
kinit -k -t /etc/hadoop/hdfs1/hbase.keytab hbase
module load hbase
hbase shell
```

Some examples of privileges that can be granted to `testuser`:

Example

```
grant 'testuser', 'RX'    # read, execute
grant 'testuser', 'CRWXA' # create, read, write, execute, admin
```

For more granular control (specific tables, column families, column qualifiers) the HBase manual should be referred to.

9

Updating Big Data Software

For Bright Cluster Manager version 7.3 onwards, it is possible to carry out version updates of some of the Big Data software from the Hadoop-related projects, without having to uninstall and reinstall the packages. For Hadoop, a full upgrade is available. For Spark, ZooKeeper and HBase, an update procedure is available.

Update Or Upgrade?

In general, when it concerns operating systems and software, an update is a minor change, and an upgrade is a major change, but the exact definition depends on the context.

The definitions of the terms *update* and *upgrade* in this chapter are as follows:

- **update** means that a software is just replaced with another version, typically newer. However, the data stays as it is. It may also not be compatible with the updated software. Compared to an upgrade, an update is a lightweight operation, and more flexible, because the replacement software can be older, the same, or newer.
- **upgrade** means that the software is replaced with another, newer version, and also that a full procedure of carrying data over from the older version to the newer version is provided. The upgrade process is a non-reversible task, unless an explicit *downgrade* procedure is also provided.

Backups Are Strongly Advised Before Updating Or Upgrading

Before carrying out an update or upgrade, a backup of the data should always be made. While updating is a less critical operation, and is very likely safe, it is still possible, given the complicated nature of Big Data software, for problems to come up.

For updating to a newer version, in case of failure the operation can, in principle, be reversed. That is, on failure, the software version is reversed by "updating" to the earlier version. The Bright Cluster Manager configuration and data should still be unchanged, so that restoring the configuration and data from backup should not be necessary.

For upgrading to a newer version, in case of failure the operation cannot, in principle, be reversed. That means that on failure, the software (project) should be uninstalled, then reconfigured, and the data should be restored from backup. For most projects, however, modification of the data is not needed, and a full restore from backup should not be needed. If in doubt, then support can always be consulted.

9.1 Upgrading Hadoop

9.1.1 General Upgrade Notes

Upgrading Hadoop must be done very carefully. Most organizations plan the procedure months in advance, depending on how critical the big data applications are. Bright Cluster Manager offers a simple upgrade, but it should stil be carried out carefully.

The following notes and cautions apply to the Hadoop upgrade procedure for Bright Cluster Manager:

- Upgrading Hadoop from versions below 2.4 is not supported.
- A rolling upgrade not supported. Some downtime is required.
- A downgrade is not supported.
- Data loss is possible if things go wrong in corner cases. If in doubt, Bright Cluster Manager support should be consulted before going ahead.

The `cm-hadoop-setup` tool is used to carry out an upgrade, using the option `--upgrade` <*instance name*>. The target version tarball must be downloaded to the filesystem, and the option `-t` <*path*> is used to provide the full path to the tarball.

Example

```
[root@bright81 ~]# cm-hadoop-setup --upgrade hdfs1 -t /cm/local/apps/hadoop/hadoop-2.7.2.tar.gz
```

9.1.2 Detailed Upgrade Notes

From version 2.4.0 of Hadoop onwards, a Hadoop cluster component upgrade proceeds according to the following general method:

- Disable component
- Upgrade component
- Enable component

The component can be a Hadoop aspect such as Name Nodes, or Data Nodes. The idea behind the method is to support rolling upgrades. That is, to carry out an upgrade without a downtime.

In Bright Cluster Manager, the upgrade procedure follows this method, and should in theory result in no downtime for the cluster. However, in practice, Bright Cluster Manager manages other software too, which means that some configurations may still require downtime, which is why a rolling upgrade is not supported.

A typical full output from an upgrade procedure looks like this:

```
[root@bright81 ~]# cm-hadoop-setup --upgrade "hdfs1" -t "hadoop-2.7.2.tar.gz"
Hadoop instance 'hdfs' has version 2.7.1 (distribution: Apache)
Requested upgrade to version 2.7.2 (distribution: Apache)
Querying rollingUpgrade status... done.
Entering safe mode... done.
Preparing rollingUpgrade... done.
Stopping YARN / HDFS services... done.
Moving symlink '/cm/shared/apps/hadoop/hadoop271_nohb_nozk' from '/cm/shared/apps/h\
adoop/Apache/2.7.1' to '/cm/shared/apps/hadoop/Apache/2.7.2'... done.
Starting NameNode with rollingUpgrade... done.
Starting Secondary NameNode... done.
Restarting DataNodes... done.
Restarting YARN services... done.
Testing rollingUpgrade... done.
Finalizing rollingUpgrade... done.
Restarting NameNode... done.
Restarting Secondary NameNode... done.
Waiting for YARN to be ready... done.
Committing changes do CMDaemon... done.
Creating new module file... done.
Hadoop instance successfully updated.
Finished.
```

The procedure carries out the following steps:

- **Environment check** - this checks if the instance is eligible for the proposed upgrade. It checks the parameters, versions, and whether another upgrade is already running.
- **Preparation** - this part of the procedure takes some precautions to stop the instance becoming unrecoverable.
- **Disable feature: Name Node and Secondary Name Node** - name node and secondary name node are stopped in this step.
- **Software updates** - Update the Hadoop distribution files.
- **Enable feature: Name Node** - temporarily restart Name Node as part of the rolling upgrade.
- **Enable feature: Secondary Name Node** - restart the Secondary Name Node, as usual.
- **Disable and reenable feature: Data Nodes** - restart of Data Nodes. When the Data Nodes are stopped, the new files are already in the proper location, but the old ones are still running. On restart, the new ones take their place.
- **Test** - at this stage, some tests are done to check if the upgrade was successful. A rollback can still be done on failure.
- **Finalize** - in this phase, the upgrade is definitive. Services are restarted or checked to see if they are working, and the configurations on are updated in the CMDaemon database.

Due to the asynchronous architecture of the steps involved, Bright Cluster Manager may still be provisioning images or restarting services in the background when the procedure itself has finished. The upgraded instances may therefore take some time to become available for new jobs. For example, the instance may have gone into safe mode during the upgrade. The health status of the instance in Bright View or `cmsh` should be checked for warnings.

The older `hadoop-examples-2.7.1.jar` may no longer work with the new software. The new jar files should be used to submit jobs to Hadoop after the upgrade. For example:

Example

```
[root@bright81 ~]# module load hadoop/hdfs1/Apache/2.7.2
[root@bright81 ~]# hadoop jar /cm/shared/apps/hadoop/hdfs1/hadoop-examples-2.7.2.jar\
 wordcount /wordcount_input /output
[root@bright81 ~]# module rm hadoop/hdfs1/Apache/2.7.2
```

9.2 Updating Spark

- Upgrading Spark from versions below 1.5.1 is not supported.
- Spark does not document an official upgrade method. This means that an update to a newer version is likely to be the same as an upgrade, because there is no data that needs to be converted into a new format.
- An update of both standalone or Yarn mode is supported, and they work in a similar way.
- A rolling update is not supported. Some downtime is required.

Spark Yarn

If a Hadoop instance called `hdfs1` exists, then the update can be carried out as follows:

Example

```
[root@bright81 ~]# cm-spark-setup --update hdfs1 -t /cm/local/apps/hadoop/spark-1.6.1-\
bin-hadoop2.6.tgz
Stopping services... done.
Spark is already installed in /cm/shared/apps/hadoop/Apache/spark-1.6.1-bin-hadoop2.6/
Updating service files... done.
Renaming module file... done.
Copying Spark assembly jar to HDFS... done.
Updating configuration in CMDaemon... done.
Restarting services... done.
Update successfully completed.
Finished.
```

A job to the updated Spark can then be submitted as follows (some output ellipsized):

Example

```
[root@bright81 ~]# module load spark/hdfs1/Apache/1.6.1-bin-hadoop2.6
[root@bright81 ~]# spark-submit /cm/shared/apps/hadoop/Apache/hdfs1/examples/src/main/p\
ython/pi.py
...
Pi is roughly 3.149340
...
[root@bright81 ~]# module rm spark/hdfs1/Apache/1.6.1-bin-hadoop2.6
```

Spark Standalone Mode

If a Spark instance called `spark1` is available, then an update looks like:

Example

```
[root@bright81 ~]# cm-spark-setup --update spark1 -t /cm/local/apps/hadoop/spark-1.6.1-\
bin-hadoop2.6.tgz
Stopping services... done.
Spark is already installed in /cm/shared/apps/hadoop/Apache/spark-1.6.1-bin-hadoop2.6/
Updating service files... done.
Renaming module file... done.
Copying Spark assembly jar to HDFS... done.
Updating configuration in CMDaemon... done.
Restarting services... done.
Update successfully completed.
Finished.
```

A job to the updated Spark can then be submitted as follows:

Example

```
[root@bright81 ~]# module load spark/spark1/Apache/1.6.1-bin-hadoop2.6
[root@bright81 ~]# spark-submit /cm/shared/apps/hadoop/Apache/spark1/examples/src/main/p\
ython/pi.py
...
Pi is roughly 3.136280
...
[root@bright81 ~]# module rm spark/spark1/Apache/1.6.1-bin-hadoop2.6
```

9.3 Updating ZooKeeper

- Upgrading ZooKeeper from versions below 3.4.6 is not supported.
- ZooKeeper does not document an official way of upgrading. This means that an update to a newer version is likely to be the same as an upgrade, because the conversion of data into a new format is not needed.
- An upgrade of both Spark standalone or Hadoop are supported, and they work in a similar way.
- A rolling update is not supported. Some downtime is required.

Updating ZooKeeper On A Hadoop Instance

If a Hadoop instance called `hdfs1` is available, then an update looks like:

Example

```
[root@bright81 ~]# cmhadoop-zookeeper-setup --update hdfs1 -t /cm/local/apps/hadoop/zookeep\
er-3.4.8.tar.gz
Connection to cluster established.
Stopping services... done.
ZooKeeper is already installed in /cm/shared/apps/hadoop/Apache/zookeeper-3.4.8/
Updating service files... done.
Renaming module file... done.
Updating configuration in CMDaemon... done.
Restarting services... done.
Update successfully completed.
Finished.
```

The updated ZooKeeper shell can then be invoked as follows to check the updated version is in place:

Example

```
[root@bright81 ~]# module load zookeeper/hdfs1/Apache/3.4.8
[root@bright81 ~]# echo quit | zkCli.sh | grep zookeeper.version
2017-01-24 16:17:10,629 [myid:] - INFO  [main:Environment@100] - Client environment:zookeep\
er.version=3.4.8--1, built on 02/06/2016 03:18 GM
[root@bright81 ~]# module rm zookeeper/hdfs1/Apache/3.4.8
```

Updating ZooKeeper On A Spark Instance

If there is a Spark instance called `spark1` available, then an update might look like:

Example

```
[root@bright81 ~]# cmhadoop-zookeeper-setup --update spark1 -t /cm/local/apps/hadoop/zookee\
per-3.4.8.tar.gz
Connection to cluster established.
Stopping services... done.
ZooKeeper is already installed in /cm/shared/apps/hadoop/Apache/zookeeper-3.4.8/
Updating service files... done.
Renaming module file... done.
Updating configuration in CMDaemon... done.
Restarting services... done.
Update successfully completed.
Finished.
```

The updated ZooKeeper shell can then be invoked as follows to check the updated version is in place:

Example

```
[root@bright81 ~]# module load zookeeper/spark1/Apache/3.4.8
[root@bright81 ~]# echo quit | zkCli.sh | grep zookeeper.version
2017-01-24 16:17:10,629 [myid:] - INFO  [main:Environment@100] - Client environment:zookeep\
er.version=3.4.8--1, built on 02/06/2016 03:18 GM
[root@bright81 ~]# module rm zookeeper/spark1/Apache/3.4.8
```

9.4 Updating HBase

- Upgrading HBase from versions below 1.1.1 is not supported.
- An HBase upgrade may be complex, and it may require file format conversion. The documentation at `https://github.com/apache/hbase/blob/master/src/main/asciidoc/_chapters/upgrading.adoc` should be checked.
- HBase upgrade may not support version skipping. It may be necessary to upgrade through all the versions in between. The documentation at `https://github.com/apache/hbase/blob/master/src/main/asciidoc/_chapters/upgrading.adoc` should be checked.
- A rolling update not supported. Some downtime is required.

If there is a Hadoop instance called `hdfs1` available then an update looks like the following:

Example

```
[root@bright81 ~]# cmhadoop-hbase-setup --update hdfs1 -t /cm/local/apps/hadoop/hbase-1.2.1\
-bin.tar.gz
# Stopping services... done.
# HBase is already installed in /cm/shared/apps/hadoop/Apache/hbase-1.2.1-bin/
# Updating service files... done.
# Renaming module file... done.
# Updating configuration in CMDaemon... done.
# Restarting services... done.
# Update successfully completed.
# Finished.
```

A shell of the updated HBase can then be invoked as follows:

Example

```
[root@bright81 ~]# module load hbase/hdfs1/Apache/1.2.1-bin
hbase-shell
...
# Version 1.2.1, r25b281972df2f5b15c426c8963cbf77dd853a5ad, Thu Feb 18 23:01:49 CST 2016
...
[root@bright81 ~]# module rm hbase/hdfs1/Apache/1.2.1-bin
```

9.5 Updating Hive

- Upgrading Hive from versions below 1.2.1 is not supported.
- Hive does not document an official upgrade method. However, with upgrades, the MetaStore schema may change, which means that carrying out an upgrade may break Hive. An upgrade should therefore only be carried out after first considering the consequences. The documentation at `https://cwiki.apache.org/confluence/display/Hive/Hive+Schema+Tool` should be checked to see how to update the MetaStore schema.
- A rolling update is not supported. Some downtime is required.

Updating Hive On A Hadoop Instance

If a Hadoop instance called `hdfs1` is available, then an update looks like:

Example

```
[root@bright81 ~]# cmhadoop-hive-setup --update hdfs1 -t /cm/local/apps/hadoop/apache-hive-\
2.1.1-bin.tar.gz
Stopping services... done.
Hive is already installed in /cm/shared/apps/hadoop/Apache/hive-2.1.1-bin/
Updating service files... done.
Renaming module file... done.
Updating configuration in CMDaemon... done.
Restarting services... done.
Update successfully completed.
Finished.
```

A shell of the updated Hive can then be invoked as follows:

Example

```
[root@bright81 ~]# module load hive/hdfs1/Apache/2.1.1-bin
beeline
Beeline version 2.1.1 by Apache Hive
beeline>
[root@bright81 ~]# module rm hive/hdfs1/Apache/2.1.1-bin
```

9.6 Updating Pig

- Upgrading Pig from versions below 0.14 is not supported.
- The Pig project does not document an official upgrade. However, since data conversion into a new format is not done, it means that an update to a newer version is likely to be the same as an upgrade.
- A rolling update is not supported. Some downtime is required.

Updating Pig On A Hadoop Instance

If a Hadoop instance called `hdfs1` is available, then an update looks like:

Example

```
[root@bright81 ~]# cmhadoop-pig-setup --update hdfs1 -t /cm/local/apps/hadoop/pig-0.16.0.ta\
r.gz
Stopping services... done.
Pig is already installed in /cm/shared/apps/hadoop/Apache/pig-0.16.0/
Updating service files... done.
Renaming module file... done.
Updating configuration in CMDaemon... done.
Restarting services... done.
Update successfully completed.
Finished.
```

Grunt, the shell of the updated Pig, can then be invoked as follows to check on the version details:

Example

```
[root@bright81 ~]# module load pig/hdfs1/Apache/0.16.0
[root@bright81 ~]# pig
...
17/01/24 16:22:27 INFO pig.Main: Apache Pig version 0.16.0 (r1746530) compiled Jun 01 2016,\
 23:10:49
...
[root@bright81 ~]# module rm pig/hdfs1/Apache/0.16.0
```

A

Details And Examples Of Hadoop Configuration

This appendix supplements section 3.1.7'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 8.1 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 metadata, 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 3.1.7) 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 3.1.7 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) (page 21) 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), 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 3.1.7. 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

```
[bright81]% configurationoverlay
[bright81->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..      Hadoop::DataNode, Hadoop::YARNClient,
               node005        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

```
[bright81->category]% !#check if nodes in default category first
[bright81->category]% listnodes default
Type                 Hostname ...
-------------------- -------- ...
PhysicalNode         node001  ...
PhysicalNode         node002  ...
PhysicalNode         node003  ...
PhysicalNode         node004  ...
PhysicalNode         node005  ...
PhysicalNode         node006  ...
PhysicalNode         node007  ...
[bright81->category]% use default
[bright81->category[default]]% roles; assign hadoop::hbaseclient; commit
...
[bright81]% device; use node002
[bright81->device[node002]]% roles; assign hadoop::datanode; commit
[bright81]% device; use node003
[bright81->device[node003]]% roles; assign hadoop::datanode; commit
[bright81]% device; use node005
[bright81->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):

```
[bright81->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 overriden 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.

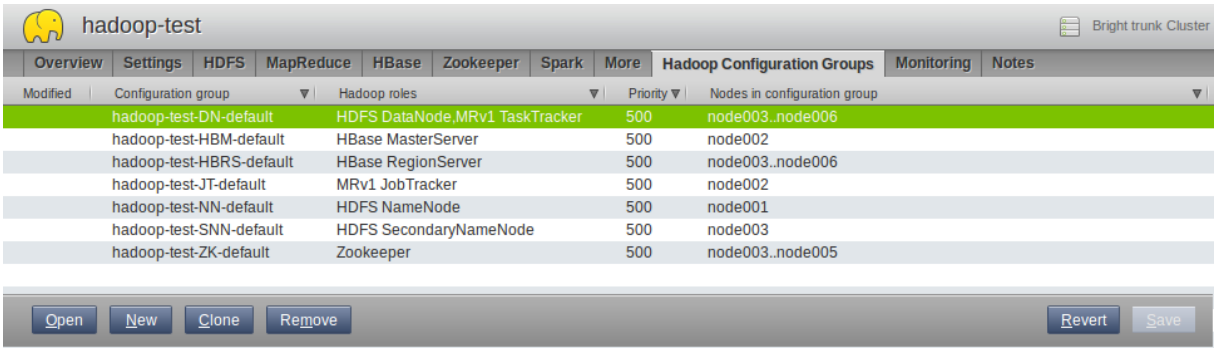

Figure A.1: Hadoop Configuration Group tab in cmgui

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.

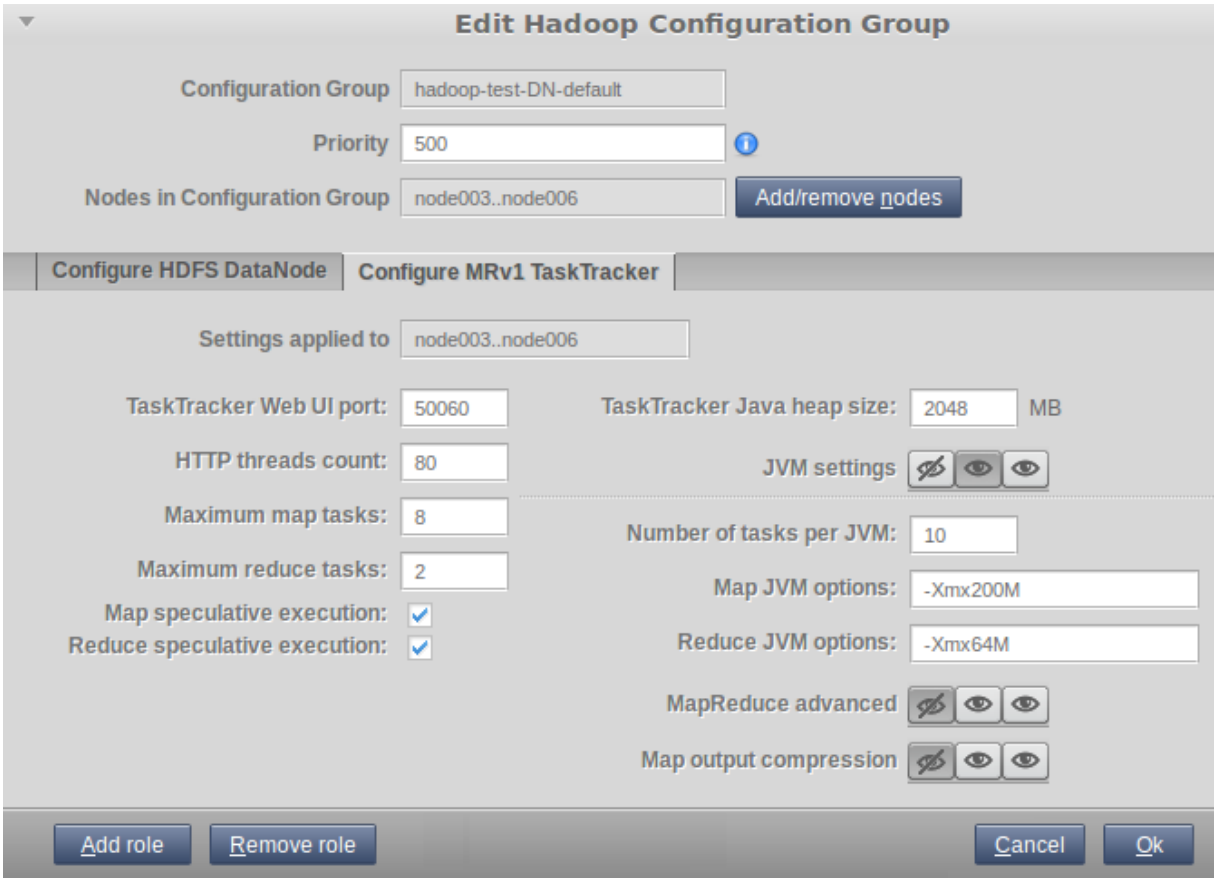

Figure A.2: Hadoop Configuration Group Prior To Cloning

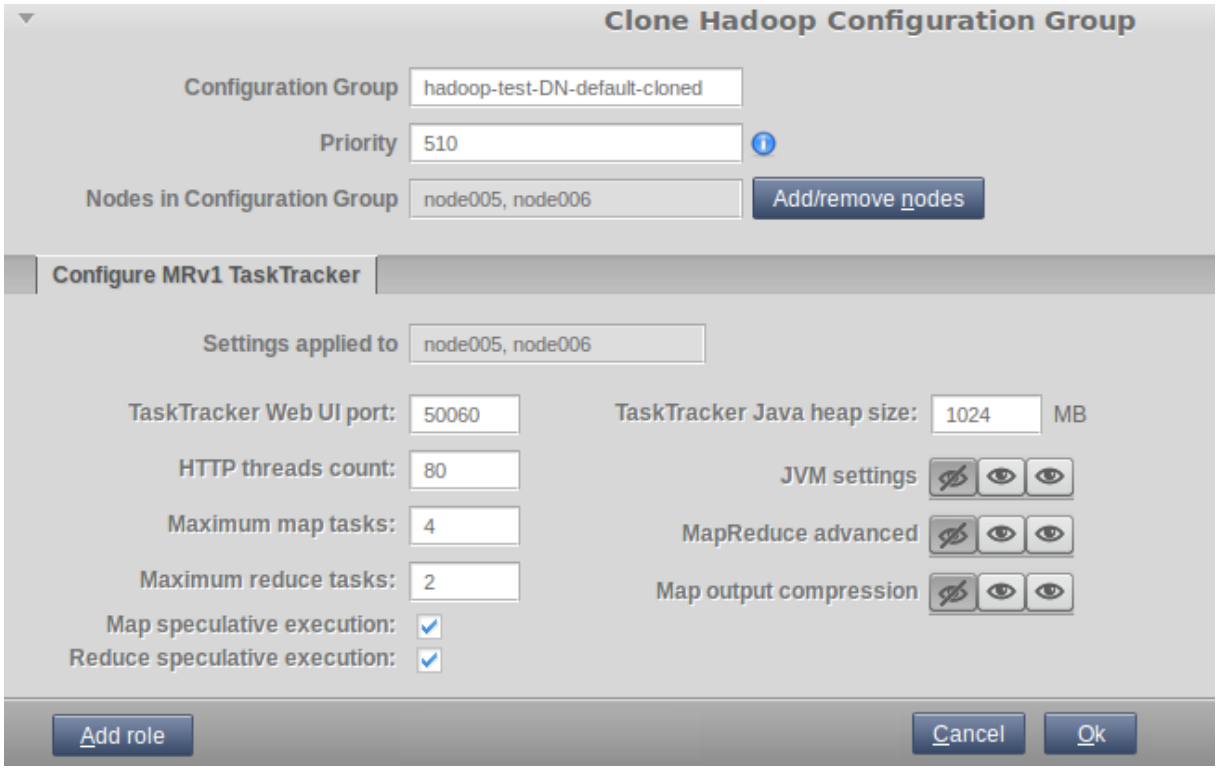

Figure A.3: Example Of Cloned Hadoop Configuration Group

- 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.

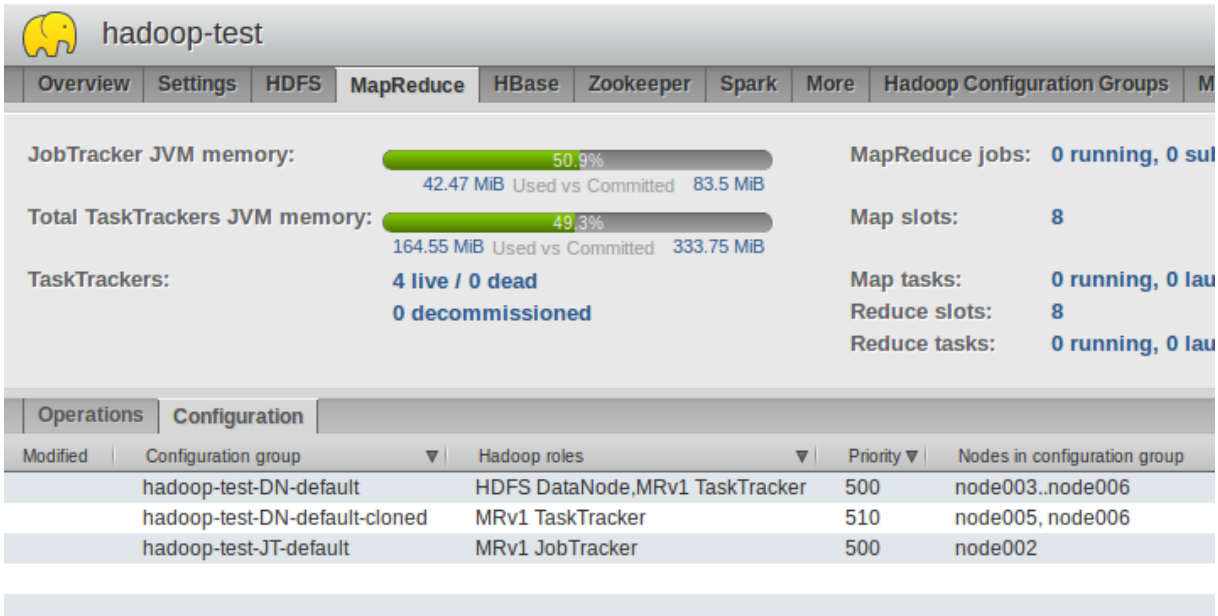

Figure A.4: Hadoop Configuration Groups for MapReduce after example configuration

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 tasktrackerheapsize 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 3.1.7) 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).

A.6 Customizations For Configuration Overlays

Configuration overlays (section 3.1.7) have a feature called *customizations* which have been introduced in Bright Cluster Manager version 7.2.

A customization in the configuration overlay sense is a set of modifications applicable to a file. A configuration overlay may contain multiple customizations so that several files are modified by one overlay.

Customizations are useful for Big Data and for OpenStack, where new plugin installations often require this kind of flexibility.

If a given configuration can however be managed using roles and configuration overlays, then using those is recommended instead of using customizations.

A.6.1 Customization Example Overview

Original Configuration File

For Hadoop, a typical configuration file for an instance `hdfs1` might be at `/etc/hadoop/hdfs1/hdfs-site.xml`, containing key-value properties as follows:

```
<property>
  <name>mapreduce.job.map.output.collector.class</name>
  <value>org.apache.hadoop.mapred.MapTask$MapOutputBuffer</value>
</property>

<property>
  <name>mapreduce.job.reduce.shuffle.consumer.plugin.class</name>
  <value>org.apache.hadoop.mapreduce.task.reduce.Shuffle</value>
</property>

<property>
  <!-- automatically managed by cmdaemon -->
  <name>dfs.datanode.data.dir</name>
  <value>/var/lib/hadoop/hdfs1/datanode</value>
  <final>true</final>
</property>
```

The last property is already managed by Bright Cluster Manager. Using customizations to change this property section is therefore not recommended, and the appropriate DataNode role (section 3.1.7) should be used instead. Nonetheless, customizing properties via customizations overrules values that Bright Cluster Manager may have written.

Original Customization State

Bright Cluster Manager reads a customization file specification, then modifies the configuration file, then writes the modified configuration files to disk, and then restarts the dependent services.

Bright Cluster Manager recognizes whether the file is managed by OpenStack or by Hadoop. The manager for the customization is indicated by the entry under the `Manager` heading of the `customizationoverview` command in `cmsh`:

```
[cluster1->configurationoverlay[hdfs1-DataNode]]% customizationsoverview
Node     File [Type]         Customization             Conf. Overlay [Prio] Manager
-------- ------------------- ------------------------- -------------------- -------
node001  hdfs-site.xml [XML] mapreduce.job.map.output+ hdfs1-DataNode [500] hadoop
node002  hdfs-site.xml [XML] mapreduce.job.map.output+ hdfs1-DataNode [500] hadoop
node003  hdfs-site.xml [XML] mapreduce.job.map.output+ hdfs1-DataNode [500] hadoop
node001  hdfs-site.xml [XML] mapreduce.job.reduce.shu+ hdfs1-DataNode [500] hadoop
node002  hdfs-site.xml [XML] mapreduce.job.reduce.shu+ hdfs1-DataNode [500] hadoop
node003  hdfs-site.xml [XML] mapreduce.job.reduce.shu+ hdfs1-DataNode [500] hadoop
```

The customization strings in the output to `customizationoverview` displayed in the preceding transcript are abbreviations of the following `key=value` pairs:

```
mapreduce.job.map.output.collector.class = org.apache.hadoop.mapred.MyFirstPlugin$MapOutputBuffer
```

and

```
mapreduce.job.reduce.shuffle.consumer.plugin.class = org.apache.hadoop.mapred.MyFirstPlugin$Shuffle
```

Modification Applied

The modifications can be carried out via `cmsh` (section A.6.2) or `cmgui` (section A.6.3).

The MapReduce properties, which currently contain the two default `mapreduce.job*` values, can be modified by applying a reference to a custom class.

When the planned customizations are applied, the configuration file, `hdfs-site.xml` is modified. The modifications can be applied to nodes `node00[1-3]` according to the following configuration overlay specification:

```
<property>
  <!-- automatically customized by cmdaemon -->
  <name>mapreduce.job.map.output.collector.class</name>
  <value>org.apache.hadoop.mapred.MyFirstPlugin$MapOutputBuffer</value>
</property>

<property>
  <!-- automatically customized by cmdaemon -->
  <name>mapreduce.job.reduce.shuffle.consumer.plugin.class</name>
  <value>org.apache.hadoop.mapred.MyFirstPlugin$Shuffle</value>
</property>
```

The dependent services on these nodes are correctly restarted by Bright Cluster Manager.

A.6.2 Adding Customizations To A Configuration Overlay In `cmsh`

Inside the `configuration overlays` mode there is a submode for `customizations`. One or more customization files can be added in the `customizations` submode.

A file type, typically XML, is guessed by Bright Cluster Manager using its knowledge of certain paths on the filesystem and the file extension. The file type can be changed manually.

Inside a customization file there is another submode for entries. Customization entries can be added to the file. These entries are typically key-value pairs with an action, where the default action is to add the key-value pair to the configuration. How these entries are handled depends on the customization file type.

```
[cluster1->configurationoverlay[hdfs1-DataNode]]% customizations
[cluster1...-DataNode]->customizations*]% add /etc/hadoop/hdfs1/hdfs-site.xml
ok.
```

```
[...customizations*[/etc/hadoop/hdfs1/hdfs-site.xml*]]% list
File (key)                                 Type            Label           Enabled
------------------------------------------ --------------- --------------- -------
/etc/hadoop/hdfs1/hdfs-site.xml            XML File                        yes
[.../hdfs-site.xml*]]% entries
[.../hdfs-site.xml*]->entries] add "mapreduce.job.map.output.collector.class"\
  "org.apache.hadoop.mapred.MyFirstPlugin$MapOutputBuffer"
ok.
[...entries[mapreduce.job.map.output.collector.class]]% list
Key                             Value                           Action    Enabled
------------------------------- ------------------------------- --------- -------
mapreduce.job.map.output.collec+ org.apache.hadoop.mapred.MyFirs+ Smart Add yes

[...entries[mapreduce.job.map.output.collector.class]]% commit
Commit configurationoverlay 'hdfs1-DataNode' ... ok.
```

A label can be assigned to related customizations. This is useful for customizations that are spread out over multiple configuration overlays. The `cmsh` commands `customizationsenable` and `customizationsdisable` allow a label to be set.

A.6.3 Managing Customizations From `cmgui`

In `cmgui`, within the `Configuration Groups` tab for the big data instance, there is a further subtab `Configuration Groups`. Opening a configuration group opens up the a configuration group dialog (figure A.5) within which there is a `Manage customizations` button. The `Manage customizations` button opens up a customizations configuration dialog. This allows one or more customization files to be added or removed.

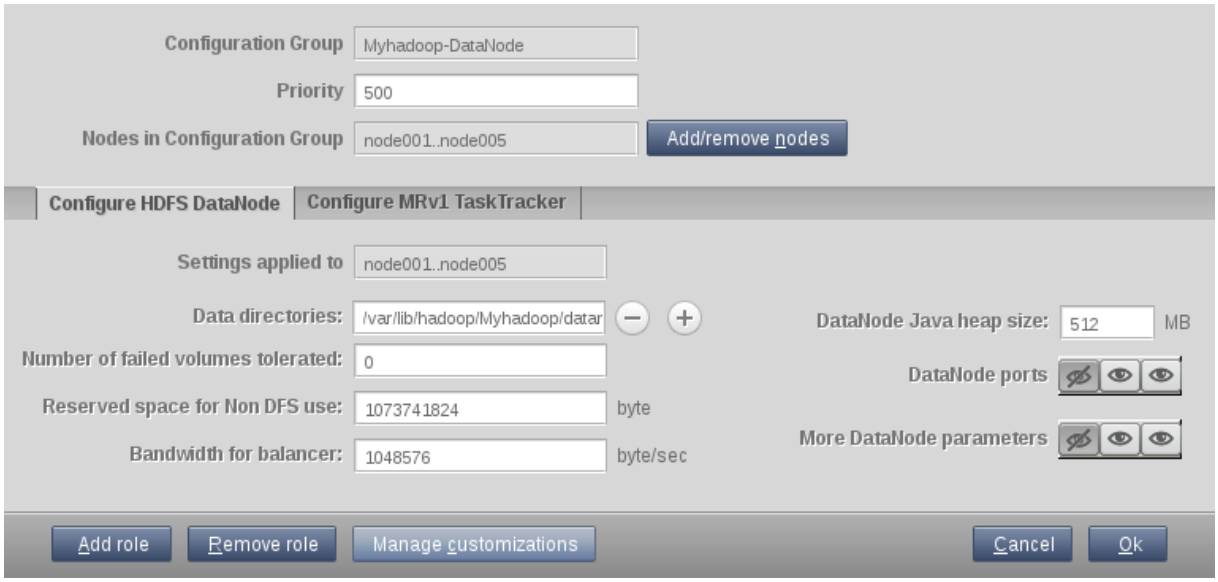

Figure A.5: Hadoop configuration group showing the customization button

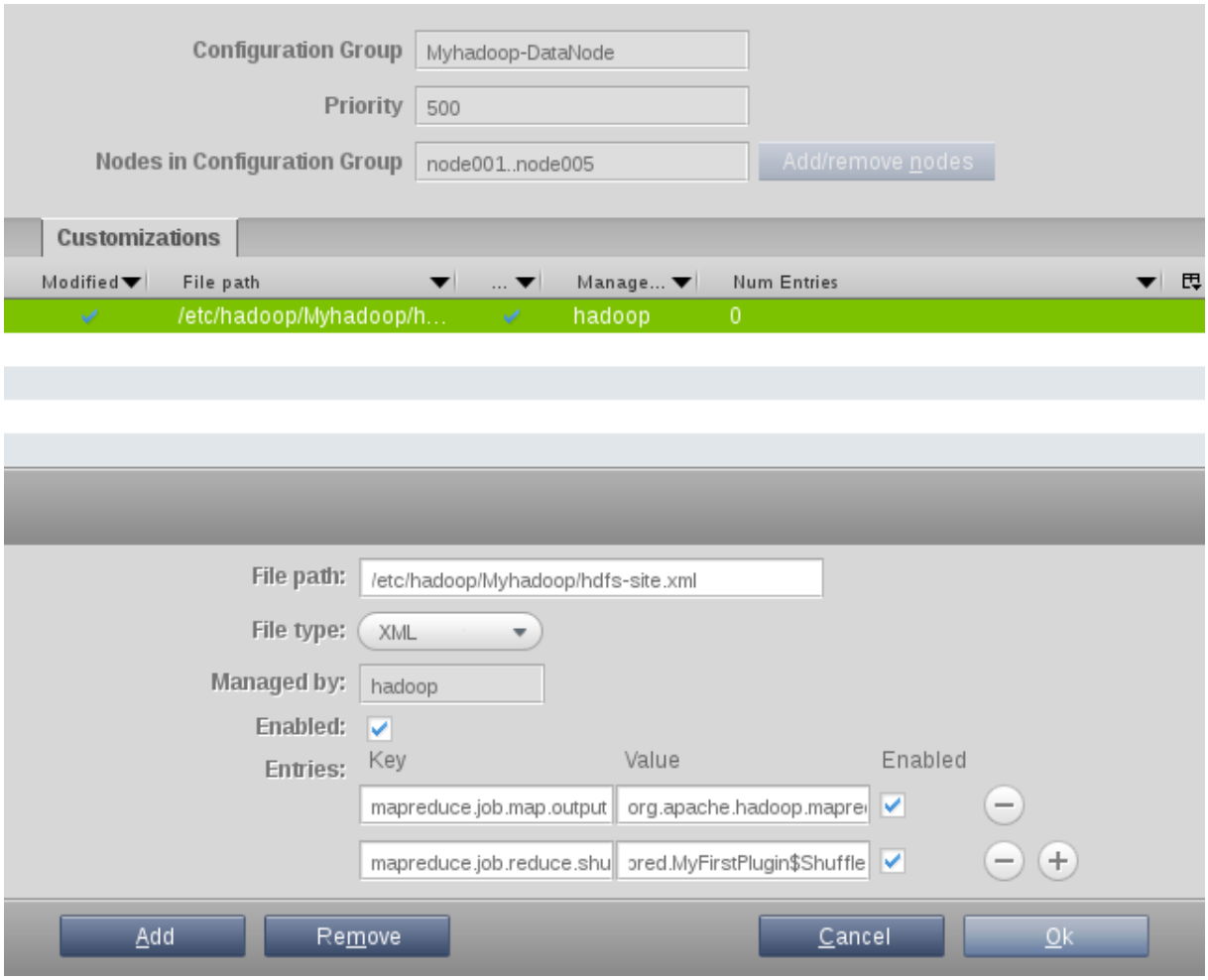

Figure A.6: Managing customizations within a Hadoop configuration group

A.6.4 Magic Strings Available For Customizations

Linux flavor	Status
`${INSTANCE}`	Will be replaced by the Big Data instance name, *i.e.*; `hdfs1`.
`${HADOOP_CONF_DIR}`	Will be replaced by the Big Data instance configuration directory, *i.e.*; `/etc/hadoop/hdfs1`.
`${HOSTNAME_FQDN}`	Will be replaced by the fully qualified domain name of the compute node, *e.g.*; `node001.cm.cluster`.
`${HOSTNAME_FQDN_LOWER}`	Same as above, but guarenteed to be lowercase (*i.e.*; Kerberos principals should be lowercase).
`${HOSTNAME_SHORT}`	Will be replaced by the computenode's short domain name, *e.g.*; `node001`.
`${HOSTNAME_SHORT_LOWER}`	Same as above, but guaranteed to be lowercase (*i.e.*; Kerberos principals should be lowercase).

This works for all the Bright Cluster Manager configuration values for managed Hadoop XML property files and for environment shell script files, and not just for customizations. In practice, customizations are the most likely place where these would be used.

Example

```
[mycluster]% configurationoverlay
[mycluster->configurationoverlay]% use hdfs1-kafka
[mycluster->configurationoverlay[hdfs1-Kafka]]% customizations
[mycluster->...customizations]% add /etc/hadoop/hdfs1/kafka/server.properties
[mycluster->...customizations*[...kafka/server.properties*]]% set type environment file
[mycluster->...customizations*[...kafka/server.properties*]]% entries
[mycluster->......entries]% add listeners SASL_PLAINTEXT://$HOSTNAME_FQDN:9092
[mycluster->......entries*[listeners*]]% commit
Commit configurationoverlay 'hdfs1-Kafka' ... ok.
```

The preceding customization writes out the value, with the magic string replaced with the appropriate hostname. On `node001` the result would be as follows:

```
advertised.listeners=SASL_PLAINTEXT://node001.cm.cluster:9092
```

Another example (which Bright Cluster Manager actually already manages) would be to construct the principal for tools *kms-site.xml*.

```
hadoop.kms.authentication.kerberos.principal=HTTP/${HOSTNAME_FQDN_LOWER}@CM.CLUSTER
```