# Oracle® Big Data Appliance
# Software User's Guide

Release 5.2

F25130-02

March 2021

ORACLE®

Oracle Big Data Appliance Software User's Guide, Release 5.2

F25130-02

# Contents

## 2    Security for Oracle Big Data Appliance

## 3    Administering Oracle Big Data Appliance

# 4    Supporting User Access to Oracle Big Data Appliance

# 5   Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance

# Part II   Oracle DataSource for Apache Hadoop

# 6   Oracle DataSource for Apache Hadoop (OD4H)

## Glossary

## Index

# Changes in Oracle Big Data Appliance Release 5.2

See Changes In Oracle Big Data Appliance 5.2 in the *Oracle Big Data Appliance Owner's Guide* for a summary of new features as well as a list of important software updates since the previous release.

# Part I

# Administration

This part describes Oracle Big Data Appliance and provides instructions for routine administrative tasks. It contains the following chapters:

- Introducing Oracle Big Data Appliance
- Security for Oracle Big Data Appliance
- Administering Oracle Big Data Appliance
- Supporting User Access to Oracle Big Data Appliance
- Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance

**ORACLE**®

# 1
# Introducing Oracle Big Data Appliance

This chapter presents an overview of Oracle Big Data Appliance and describes the software installed on the system. This chapter contains the following sections:

- What Is Big Data?
- The Oracle Big Data Solution
- Software for Big Data Appliance
- Acquiring Data for Analysis
- Organizing Big Data
- Analyzing and Visualizing Big Data

## 1.1 What Is Big Data?

Using transactional data as the source of business intelligence has been commonplace for many years. As digital technology and the World Wide Web spread into every aspect of modern life, other sources of data can make important contributions to business decision making. Many businesses are looking to these new data sources. They are finding opportunities in analyzing vast amounts of data that until recently was discarded.

Big data is characterized by:

- A variety of data sources
- A complexity of data types
- A high volume of data flow
- A high velocity of data transactions

These characteristics pinpoint the challenges in deriving value from big data, and the differences between big data and traditional data sources that primarily provide highly structured, transactional data.

### 1.1.1 High Variety

Big data is derived from a variety of sources, such as:

- Equipment sensors: Medical, manufacturing, transportation, and other machine sensor transmissions
- Machines: Call detail records, web logs, smart meter readings, Global Positioning System (GPS) transmissions, and trading systems records
- Social media: Data streams from social media sites such as Facebook and blogging sites such as Twitter

Analysts can mine this data repeatedly as they devise new ways of extracting meaningful insights. What seems irrelevant today might prove to be highly pertinent to your business tomorrow.

*Challenge*: Delivering flexible systems to handle this high variety

## 1.1.2 High Complexity

As the variety of data types increases, the complexity of the system increases. The complexity of data types also increases in big data because of its low structure.

*Challenge*: Finding solutions that apply across a broad range of data types.

## 1.1.3 High Volume

Social media can generate terabytes of daily data. Equipment sensors and other machines can generate that much data in less than an hour.

Even traditional data sources for data warehouses, such as customer profiles from customer relationship management (CRM) systems, transactional enterprise resource planning (ERP) data, store transactions, and general ledger data, have increased tenfold in volume over the past decade.

*Challenge*: Providing scalability and ease in growing the system

## 1.1.4 High Velocity

Huge numbers of sensors, web logs, and other machine sources generate data continuously and at a much higher speed than traditional sources, such as individuals entering orders into a transactional database.

*Challenge*: Handling the data at high speed without stressing the structured systems

# 1.2 The Oracle Big Data Solution

Oracle Big Data Appliance is an engineered system comprising both hardware and software components. The hardware is optimized to run the enhanced big data software components.

Oracle Big Data Appliance delivers:

- A complete and optimized solution for big data

- Single-vendor support for both hardware and software

- An easy-to-deploy solution

- Tight integration with Oracle Database and Oracle Exadata Database Machine

Oracle provides a big data platform that captures, organizes, and supports deep analytics on extremely large, complex data streams flowing into your enterprise from many data sources. You can choose the best storage and processing location for your data depending on its structure, workload characteristics, and end-user requirements.

Oracle Database enables all data to be accessed and analyzed by a large user community using identical methods. By adding Oracle Big Data Appliance in front of Oracle Database, you can bring new sources of information to an existing data warehouse. Oracle Big Data Appliance is the platform for acquiring and organizing big data so that the relevant portions with true business value can be analyzed in Oracle Database.

For maximum speed and efficiency, Oracle Big Data Appliance can be connected to Oracle Exadata Database Machine running Oracle Database. Oracle Exadata Database Machine provides outstanding performance in hosting data warehouses and transaction processing databases. Moreover, Oracle Exadata Database Machine can be connected to Oracle Exalytics In-Memory Machine for the best performance of business intelligence and planning applications. The InfiniBand connections between these engineered systems provide high parallelism, which enables high-speed data transfer for batch or query workloads.

The following figure shows the relationships among these engineered systems.

**Figure 1-1    Oracle Engineered Systems for Big Data**
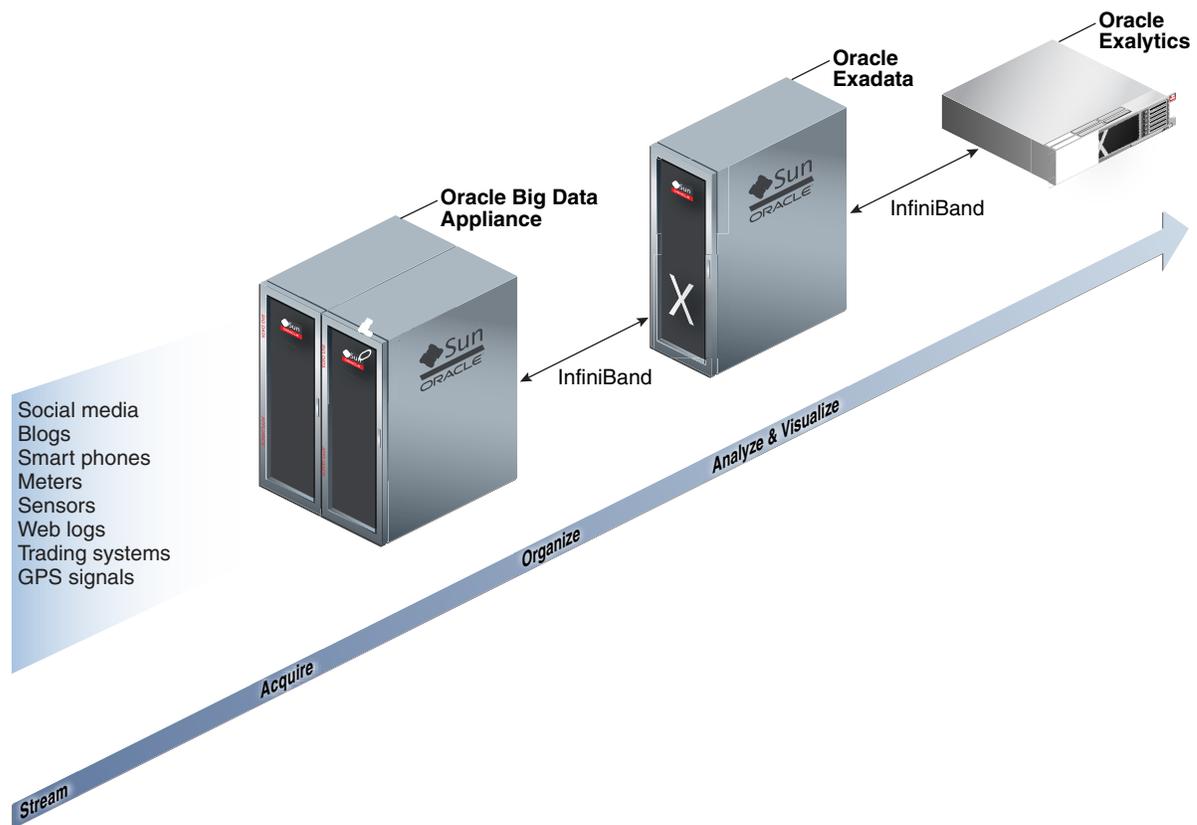


# 1.3 Software for Big Data Appliance

The **Oracle Linux** operating system and Cloudera's Distribution including Apache Hadoop (CDH) underlie all other software components installed on Oracle Big Data Appliance. **CDH** is an integrated stack of components that have been tested and packaged to work together.

CDH has a batch processing infrastructure that can store files and distribute work across a set of computers. Data is processed on the same computer where it is stored. In a single Oracle Big Data Appliance rack, CDH distributes the files and workload across 18 servers, which compose a **cluster**. Each server is a node in the cluster.

The software framework consists of these primary components:

- **File system**: The **Hadoop Distributed File System (HDFS)** is a highly scalable file system that stores large files across multiple servers. It achieves reliability by replicating data across multiple servers without RAID technology. It runs on top of the Linux file system on Oracle Big Data Appliance.

- **MapReduce engine**: The MapReduce engine provides a platform for the massively parallel execution of algorithms written in Java. Oracle Big Data Appliance 3.0 runs YARN by default.

- **Administrative framework**: Cloudera Manager is a comprehensive administrative tool for CDH. In addition, you can use Oracle Enterprise Manager to monitor both the hardware and software on Oracle Big Data Appliance.

- **Apache projects**: CDH includes Apache projects for MapReduce and HDFS, such as Hive, Pig, Oozie, ZooKeeper, HBase, Sqoop, and Spark.

- **Cloudera applications**: Oracle Big Data Appliance installs all products included in Cloudera Enterprise Data Hub Edition, including Impala, Search, and Navigator.

## 1.3.1 Software Component Overview

The major software components perform three basic tasks:

- Acquire

- Organize

- Analyze and visualize

The best tool for each task depends on the density of the information and the degree of structure. The following figure shows the relationships among the tools and identifies the tasks that they perform.

**Figure 1-2    Oracle Big Data Appliance Software Overview**

> **Note:**
>
> In Oracle Big Data Appliance 5.0, Oracle Big Data SQL is temporarily de-supported.
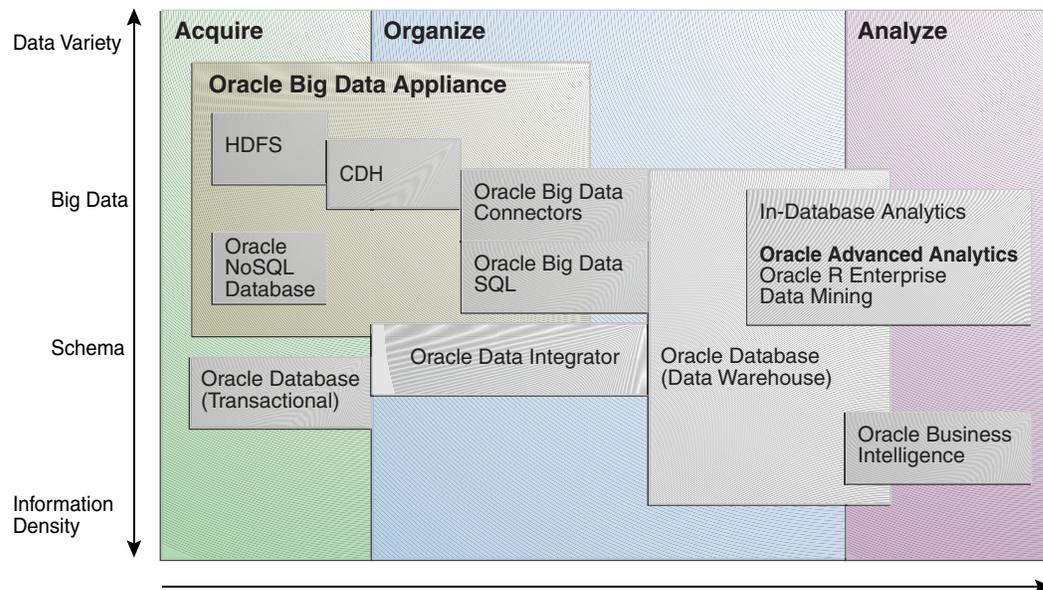
# 1.4 Acquiring Data for Analysis

Databases used for online transaction processing (OLTP) are the traditional data sources for data warehouses. The Oracle solution enables you to analyze traditional data stores with big data in the same Oracle data warehouse. Relational data continues to be an important source of business intelligence, although it runs on separate hardware from Oracle Big Data Appliance.

Oracle Big Data Appliance provides these facilities for capturing and storing big data:

- Hadoop Distributed File System
- Apache Hive
- Oracle NoSQL Database

## 1.4.1 Hadoop Distributed File System

Cloudera's Distribution including Apache Hadoop (CDH) on Oracle Big Data Appliance uses the Hadoop Distributed File System (HDFS). HDFS stores extremely large files containing record-oriented data. On Oracle Big Data Appliance, HDFS splits large data files into chunks of 256 megabytes (MB), and replicates each chunk across three different nodes in the cluster. The size of the chunks and the number of replications are configurable.

Chunking enables HDFS to store files that are larger than the physical storage of one server. It also allows the data to be processed in parallel across multiple computers with multiple processors, all working on data that is stored locally. Replication ensures the high availability of the data: if a server fails, the other servers automatically take over its work load.

HDFS is typically used to store all types of big data.

> ✎ **See Also:**
>
> - For conceptual information about Hadoop technologies, refer to this third-party publication:
>
>   *Hadoop: The Definitive Guide, Third Edition* by Tom White (O'Reilly Media Inc., 2012, ISBN: 978-1449311520).
>
> - For documentation about Cloudera's Distribution including Apache Hadoop, see the Cloudera documentation at
>
>   http://www.cloudera.com/

## 1.4.2 Apache Hive

Hive is an open-source data warehouse that supports data summarization, ad hoc querying, and data analysis of data stored in HDFS. It uses a SQL-like language called **HiveQL**. An interpreter generates MapReduce code from the HiveQL queries. By storing data in Hive, you can avoid writing MapReduce programs in Java.

Hive is a component of CDH and is always installed on Oracle Big Data Appliance. Oracle Big Data Connectors can access Hive tables.

## 1.4.3 Oracle NoSQL Database

Oracle NoSQL Database is a distributed key-value database built on the proven storage technology of Berkeley DB Java Edition. Whereas HDFS stores unstructured data in very large files, Oracle NoSQL Database indexes the data and supports transactions. But unlike Oracle Database, which stores highly structured data, Oracle NoSQL Database has relaxed consistency rules, no schema structure, and only modest support for joins, particularly across storage nodes.

NoSQL databases, or "Not Only SQL" databases, have developed over the past decade specifically for storing big data. However, they vary widely in implementation. Oracle NoSQL Database has these characteristics:

- Uses a system-defined, consistent hash index for data distribution
- Supports high availability through replication
- Provides single-record, single-operation transactions with relaxed consistency guarantees
- Provides a Java API

Oracle NoSQL Database is designed to provide highly reliable, scalable, predictable, and available data storage. The key-value pairs are stored in shards or partitions (that is, subsets of data) based on a primary key. Data on each shard is replicated across multiple storage nodes to ensure high availability. Oracle NoSQL Database supports fast querying of the data, typically by key lookup.

An intelligent driver links the NoSQL database with client applications and provides access to the requested key-value on the storage node with the lowest latency.

Oracle NoSQL Database includes hashing and balancing algorithms to ensure proper data distribution and optimal load balancing, replication management components to

handle storage node failure and recovery, and an easy-to-use administrative interface to monitor the state of the database.

Oracle NoSQL Database is typically used to store customer profiles and similar data for identifying and analyzing big data. For example, you might log in to a website and see advertisements based on your stored customer profile (a record in Oracle NoSQL Database) and your recent activity on the site (web logs currently streaming into HDFS).

Oracle NoSQL Database is an optional component of Oracle Big Data Appliance and runs on a separate cluster from CDH.

> **See Also:**
>
> • Oracle NoSQL Database documentation
> • *Oracle Big Data Appliance Licensing Information*

## 1.5 Allocating Resources Among Services

You can allocate resources to each service—HDFS, YARN, Hive, and so forth—as a percentage of the total resource pool. Cloudera Manager automatically calculates the recommended resource management settings based on these percentages. The static service pools isolate services on the cluster, so that a high load on one service as a limited impact on the other services.

**To allocate resources among services:**

1. Log in as `admin` to Cloudera Manager.
2. Open the Clusters menu at the top of the page, then select **Static Service Pools** under Resource Management.
3. Select **Configuration**.
4. Follow the steps of the wizard, or click **Change Settings Directly** to edit the current settings.

## 1.6 Organizing Big Data

Oracle Big Data Appliance provides several ways of organizing, transforming, and reducing big data for analysis:

• MapReduce
• Oracle Big Data Connectors
• Oracle R Support for Big Data

### 1.6.1 MapReduce

The MapReduce engine provides a platform for the massively parallel execution of algorithms written in Java. MapReduce uses a parallel programming model for processing data on a distributed system. It can process vast amounts of data quickly

and can scale linearly. It is particularly effective as a mechanism for batch processing of unstructured and semistructured data. MapReduce abstracts lower-level operations into computations over a set of keys and values.

Although big data is often described as unstructured, incoming data always has some structure. However, it does not have a fixed, predefined structure when written to HDFS. Instead, MapReduce creates the desired structure as it reads the data for a particular job. The same data can have many different structures imposed by different MapReduce jobs.

A simplified description of a MapReduce job is the successive alternation of two phases: the Map phase and the Reduce phase. Each Map phase applies a transform function over each record in the input data to produce a set of records expressed as key-value pairs. The output from the Map phase is input to the Reduce phase. In the Reduce phase, the Map output records are sorted into key-value sets, so that all records in a set have the same key value. A reducer function is applied to all the records in a set, and a set of output records is produced as key-value pairs. The Map phase is logically run in parallel over each record, whereas the Reduce phase is run in parallel over all key values.

> **Note:**
>
> Oracle Big Data Appliance uses the Yet Another Resource Negotiator (YARN) implementation of MapReduce.

## 1.6.2 Oracle Big Data Connectors

Oracle Big Data Connectors facilitate data access between data stored in CDH and Oracle Database. The connectors are licensed separately from Oracle Big Data Appliance and include:

*   Oracle SQL Connector for Hadoop Distributed File System

*   Oracle Loader for Hadoop

*   Oracle XQuery for Hadoop

*   Oracle R Advanced Analytics for Hadoop

*   Oracle Shell for Hadoop Loaders

*   Oracle Data Integrator Enterprise Edition

> **Note:**
>
> Oracle Data Integrator (ODI) is not supported in this release of Oracle Big Data Appliance.

> **See Also:**
>
> Oracle Big Data Connectors

## 1.6.2.1 Oracle SQL Connector for Hadoop Distributed File System

Oracle SQL Connector for Hadoop Distributed File System (Oracle SQL Connector for HDFS) provides read access to HDFS from an Oracle database using **external tables**.

An external table is an Oracle Database object that identifies the location of data outside of the database. Oracle Database accesses the data by using the metadata provided when the external table was created. By querying the external tables, users can access data stored in HDFS as if that data were stored in tables in the database. External tables are often used to stage data to be transformed during a database load.

You can use Oracle SQL Connector for HDFS to:

- Access data stored in HDFS files
- Access Hive tables.
- Access Data Pump files generated by Oracle Loader for Hadoop
- Load data extracted and transformed by Oracle Data Integrator

## 1.6.2.2 Oracle Loader for Hadoop

Oracle Loader for Hadoop is an efficient and high-performance loader for fast movement of data from a Hadoop cluster into table in an Oracle database. It can read and load data from a wide variety of formats. Oracle Loader for Hadoop partitions the data and transforms it into a database-ready format in Hadoop. It optionally sorts records by a sorting key (such as a primary key) before loading the data or creating output files. The load runs as a MapReduce job on the Hadoop cluster.

## 1.6.2.3 Oracle XQuery for Hadoop

Oracle XQuery for Hadoop runs transformations expressed in the XQuery language by translating them into a series of MapReduce jobs, which are executed in parallel on the Hadoop cluster. The input data can be located in HDFS or Oracle NoSQL Database. Oracle XQuery for Hadoop can write the transformation results to HDFS, Oracle NoSQL Database, or Oracle Database.

## 1.6.2.4 Oracle R Advanced Analytics for Hadoop

Oracle R Advanced Analytics for Hadoop is a collection of R packages that provides:

- Interfaces to work with Hive tables, Apache Hadoop compute infrastructure, local R environment and database tables
- Predictive analytic techniques written in R or Java as Hadoop MapReduce jobs that can be applied to data in HDFS files

Using simple R functions, you can copy data between R memory, the local file system, HDFS, and Hive. You can write mappers and reducers in R, schedule these R programs to execute as Hadoop MapReduce jobs, and return the results to any of those locations.

## 1.6.2.5 Oracle Data Integrator Enterprise Edition

Oracle Data Integrator (ODI) Enterprise Edition extracts, transforms, and loads data into Oracle Database from a wide range of sources.

In ODI, a knowledge module (KM) is a code template dedicated to a specific task in the data integration process. You use Oracle Data Integrator Studio to load, select, and configure the KMs for your particular application. More than 150 KMs are available to help you acquire data from a wide range of third-party databases and other data repositories. You only need to load a few KMs for any particular job.

Oracle Data Integrator Enterprise Edition contains the KMs specifically for use with big data.

The ODI agent mounted on Oracle Big Data Appliance is the Standalone Agent (rather than the Colocated Agent or Java EE Agent ).

You can establish master-child relationships between ODI agents on Oracle Big Data Appliance. You can also configure an external HA master Java EE Agent to distribute jobs to multiple Standalone Agents on Oracle Big Data Appliance, which is useful if your enterprise uses ODI to extract data from other sources in addition to the appliance.

## 1.6.2.6 Oracle Shell for Hadoop Loaders

Oracle Shell for Hadoop Loaders is a helper shell that provides a simple to use command line interface to Oracle Loader for Hadoop, Oracle SQL Connector for HDFS, and the Copy to Hadoop feature of Big Data SQL.

## 1.6.3 Oracle R Support for Big Data

R is an open-source language and environment for statistical analysis and graphing It provides linear and nonlinear modeling, standard statistical methods, time-series analysis, classification, clustering, and graphical data displays. Thousands of open-source packages are available in the Comprehensive R Archive Network (CRAN) for a spectrum of applications, such as bioinformatics, spatial statistics, and financial and marketing analysis. The popularity of R has increased as its functionality matured to rival that of costly proprietary statistical packages.

Analysts typically use R on a PC, which limits the amount of data and the processing power available for analysis. Oracle eliminates this restriction by extending the R platform to directly leverage Oracle Big Data Appliance. Oracle R Distribution is installed on all nodes of Oracle Big Data Appliance.

Oracle R Advanced Analytics for Hadoop provides R users with high-performance, native access to HDFS and the MapReduce programming framework, which enables R programs to run as MapReduce jobs on vast amounts of data. Oracle R Advanced Analytics for Hadoop is included in the Oracle Big Data Connectors. See "Oracle R Advanced Analytics for Hadoop".

**Oracle R Enterprise** is a component of the Oracle Advanced Analytics option to Oracle Database. It provides:

- Transparent access to database data for data preparation and statistical analysis from R

- Execution of R scripts at the database server, accessible from both R and SQL

- A wide range of predictive and data mining in-database algorithms

Oracle R Enterprise enables you to store the results of your analysis of big data in an Oracle database, or accessed for display in dashboards and applications.

Both Oracle R Advanced Analytics for Hadoop and Oracle R Enterprise make Oracle Database and the Hadoop computational infrastructure available to statistical users without requiring them to learn the native programming languages of either one.

> **See Also:**
>
> - For information about R, go to
>
>   `http://www.r-project.org/`
> - For information about Oracle R Enterprise, go to
>
>   `http://docs.oracle.com/cd/E67822_01/index.htm`

# 1.7 Analyzing and Visualizing Big Data

After big data is transformed and loaded in Oracle Database, you can use the full spectrum of Oracle business intelligence solutions and decision support products to further analyze and visualize all your data.

# 1.8 Best Practices

The Data Warehouse Insider blog site provides expert advice on best practices for administering and using Oracle Big Data Appliance.

Hadoop Best Practices

# 2
# Security for Oracle Big Data Appliance

Oracle Big Data Appliance development focuses on delivering an engineered system that is highly secure. This spans all aspects of the product: strong authentication (Kerberos), authorization, network encryption, encryption for data at rest, auditing and lineage/impact analysis.

## 2.1 Overview

You can take the precautions described in this section to thwart unauthorized use of the software and data on Oracle Big Data Appliance:

- About Predefined Users and Groups
- About User Authentication
- About Fine-Grained Authorization
- About HDFS Transparent Encryption
- About HTTPS / Network Encryption
- Port Numbers Used on Oracle Big Data Appliance
- Additional Guidance for Securing Clusters

> ✎ **See Also:**
>
> Oracle Big Data Appliance development abides by Oracle's comprehensive OSSA (Oracle Software Security Assurance) standards.
>
> https://www.oracle.com/corporate/security-practices/assurance/

## 2.2 About Predefined Users and Groups

Every open-source package installed on Oracle Big Data Appliance creates one or more users and groups. Most of these users do not have login privileges, shells, or home directories. They are used by daemons and are not intended as an interface for individual users. For example, Hadoop operates as the `hdfs` user, MapReduce operates as `mapred`, and Hive operates as `hive`.

You can use the `oracle` identity to run Hadoop and Hive jobs immediately after the Oracle Big Data Appliance software is installed. This user account has login privileges, a shell, and a home directory.

Oracle NoSQL Database and Oracle Data Integrator run as the `oracle` user. Its primary group is `oinstall`.

> **✎ Note:**
>
> Do not delete, re-create, or modify the users that are created during installation, because they are required for the software to operate.

The following table identifies the operating system users and groups that are created automatically during installation of Oracle Big Data Appliance software for use by CDH components and other software packages.

**Table 2-1    Operating System Users and Groups**

| User Name | Group | Used By | Login Rights |
|---|---|---|---|
| flume | flume | **Apache Flume** parent and nodes | No |
| hbase | hbase | **Apache HBase** processes | No |
| hdfs | hadoop | **NameNode**, **DataNode** | No |
| hive | hive | Hive metastore and server processes | No |
| hue | hue | **Hue** processes | No |
| mapred | hadoop | ResourceManager, NodeManager, **Hive Thrift** daemon | Yes |
| mysql | mysql | MySQL server | Yes |
| oozie | oozie | Oozie server | No |
| oracle | dba, oinstall | Oracle NoSQL Database, Oracle Loader for Hadoop, Oracle Data Integrator, and the Oracle DBA | Yes |
| sqoop | sqoop | **Apache Sqoop** metastore | No |
| svctag | | Auto Service Request | No |
| zookeeper | zookeeper | **ZooKeeper** processes | No |

# 2.3 About User Authentication

Oracle Big Data Appliance supports Kerberos security as a software installation option. See Supporting User Access to Oracle Big Data Appliance for details about setting up clients and users to access a Kerberos-protected cluster.

# 2.4 About Fine-Grained Authorization

The typical authorization model on Hadoop is at the HDFS file level, such that users either have access to all of the data in the file or none. In contrast, Apache Sentry integrates with the Hive and Impala SQL-query engines to provide fine-grained authorization to data and metadata stored in Hadoop.

Oracle Big Data Appliance automatically configures Sentry during software installation, beginning with Mammoth utility version 2.5.

> ✎ **See Also:**
>
> - Cloudera Manager Help
>
> - How to Add or Remove Sentry on Oracle Big Data Appliance v4.2 or Higher with bdacli (Doc ID 2052733.1) on My Oracle Support.

# 2.5 About HDFS Transparent Encryption

HDFS Transparent Encryption protects Hadoop data that is at rest on disk. After HDFS Transparent Encryption is enabled for a cluster on Oracle Big Data Appliance, data writes and reads to encrypted *zones* (HDFS directories) on the disk are automatically encrypted and decrypted. This process is "transparent" because it is invisible to the application working with the data.

HDFS Transparent Encryption does not affect user access to Hadoop data, although it can have a minor impact on performance.

HDFS Transparent Encryption is an option that you can select during the initial installation of the software by the Mammoth utility. You can also enable or disable HDFS Transparent Encryption at any time by using the bdacli utility. Note that HDFS Transparent Encryption can be installed only on a Kerberos-secured cluster.

Oracle recommends that you set up the Navigator Key Trustee (the service that manages keys and certificates) on a separate server, external to the Oracle Big Data Appliance.

See the following MOS documents at My Oracle Support for instructions on installing and enabling HDFS Transparent Encryption.

| Title | MOS Doc ID |
|---|---|
| *How to Setup Highly Available Active and Passive Key Trustee Servers on BDA V4.4 Using 5.5 Parcels* | 2112644.1 Installing using parcels as described in this MOS document is recommended over package-based installation. See Cloudera's comments on Parcels. |
| *How to Enable/Disable HDFS Transparent Encryption on Oracle Big Data Appliance V4.4 with bdacli* | 2111343.1 |
| *How to Create Encryption Zones on HDFS on Oracle Big Data Appliance V4.4* | 2111829.1 |

> ✎ **Note:**
>
> If either HDFS Transparent Encryption or Kerberos is disabled, data stored in the HDFS Transparent Encryption zones in the cluster will remain encrypted and therefore inaccessible. To restore access to the data, re-enable HDFS Transparent Encryption using the same key provider.

> **✎ See Also:**
>
> Cloudera documentation about HDFS at-rest encryption at http://
> www.cloudera.com for more information about managing files in encrypted
> zones.

# 2.6 About HTTPS/Network Encryption

HTTPS Network/Encryption on the Big Data Appliance has two components :

- **Web Interface Encryption**

  Configures HTTPS for the following web interfaces: Cloudera Manager, Oozie, and
  HUE. This encryption is now enabled automatically in new Mammoth installations.
  For current installations it can be enabled via the bdacli utility. This feature does
  not require that Kerberos is enabled.

- **Encryption for Data in Transit and Services**

  There are two subcomponents to this feature. Both are options that can be
  enabled in the Configuration Utility at installation time or enabled/disabled using
  the bdacli utility at any time. Both require that Kerberos is enabled.

  – **Encrypt Hadoop Services**

    This includes SSL encryption for HDFS, MapReduce, and YARN web
    interfaces, as well as encrypted shuffle for MapReduce and YARN. It also
    enable authentication for access to the web consoles for the MapReduce, and
    YARN roles.

  – **Encrypt HDFS Data Transport**

    This option will enable encryption of data transferred between DataNodes and
    clients, and among DataNodes.

HTTPS/Network Encryption is enabled and disabled on a per cluster basis. The
Configuration Utility described in the *Oracle Big Data Appliance Owner's Guide*,
includes settings for enabling encryption for Hadoop Services and HDFS Data
Transport when a cluster is created. The bdacli utility reference pages (also in the
*Oracle Big Data Appliance Owner's Guide* ) provide HTTPS/Network Encryption
command line options.

> **✎ See Also:**
>
> Supporting User Access to Oracle Big Data Appliance for an overview of how
> Kerberos is used to secure CDH clusters.
>
> About HDFS Transparent Encryption for information about Oracle Big Data
> Appliance security for Hadoop data at-rest.
>
> Cloudera documentation at http://www.cloudera.com for more information
> about HTTPS communication in Cloudera Manager and network-level
> encryption in CDH.

## 2.6.1 Configuring Web Browsers to use Kerberos Authentication

If web interface encryption is enabled, each web browser accessing an HDFS, MapReduce, or YARN-encrypted web interface must be configured to authenticate with Kerberos. Note that this is not necessary for the Cloudera Manager, Oozie, and Hue web interfaces, which do not require Kerberos.

The following are the steps to configure Mozilla Firefox[1], Microsoft Internet Explorer[2], and Google Chrome[3] for Kerberos authentication.

**To configure Mozilla Firefox:**

1. Enter `about:config` in the Location Bar.

2. In the **Search** box on the about:config page, enter: `network.negotiate-auth.trusted-uris`

3. Under Preference Name, double-click the `network.negotiate-auth.trusted-uris`.

4. In the **Enter string value** dialog, enter the hostname or the domain name of the web server that is protected by Kerberos. Separate multiple domains and hostnames with a comma.

**To configure Microsoft Internet Explorer:**

1. Configure the Local Intranet Domain:

   a. Open Microsoft Internet Explorer and click the Settings "gear" icon in the top-right corner. Select `Internet options`.

   b. Select the **Security** tab.

   c. Select the **Local intranet** zone and click **Sites**.

   d. Make sure that the first two options, `Include all local (intranet) sites not listed in other zones` and `Include all sites that bypass the proxy server` are checked.

   e. Click **Advanced** on the `Local intranet` dialog box and, one at a time, add the names of the Kerberos-protected domains to the list of websites.

   f. Click **Close**.

   g. Click **OK** to save your configuration changes, then click **OK** again to exit the Internet Options panel.

2. Configure Intranet Authentication for Microsoft Internet Explorer:

   a. Click the **Settings** "gear" icon in the top-right corner. Select `Internet Options`.

   b. Select the **Security** tab.

   c. Select the Local Intranet zone and click the Custom level... button to open the Security Settings - Local Intranet Zone dialog box.

---

1. Mozilla Firefox is a registered trademark of the Mozilla Foundation.
2. Microsoft Internet Explorer is a registered trademark of Microsoft Corporation.
3. Google Chrome is a registered trademark of Google Inc

**ORACLE**

d. Scroll down to the **User Authentication** options and select `Automatic logon only in Intranet zone .`

e. Click **OK** to save your changes.

**To configure Google Chrome:**

If you are using Microsoft Windows, use the Control Panel to navigate to the Internet Options dialogue box. Configuration changes required are the same as those described above for Microsoft Internet Explorer.

On[4] or on Linux, add the `--auth-server-whitelist` parameter to the `google-chrome` command. For example, to run Chrome from a Linux prompt, run the google-chrome command as follows

```
google-chrome --auth-server-whitelist = "hostname/domain"
```

> **Note:**
>
> On Microsoft Windows, the Windows user must be an user in the Kerberos realm and must possess a valid ticket. If these requirements are not met, an HTTP 403 is returned to the browser upon attempt to access a Kerberos-secured web interface.

# 2.7 Port Numbers Used on Oracle Big Data Appliance

The following table identifies the port numbers that might be used in addition to those used by CDH.

**To view the ports used on a particular server:**

1. In Cloudera Manager, click the **Hosts** tab at the top of the page to display the Hosts page.

2. In the Name column, click a server link to see its detail page.

3. Scroll down to the Ports section.

> **See Also:**
>
> For the full list of CDH component port numbers, go to the Cloudera website at
>
> https://www.cloudera.com/documentation/enterprise/6/6.1/topics/cdh_ports.html#cdh_ports

---

[4] Mac OS is a registered trademark of Apple, Inc.

**Table 2-2    Oracle Big Data Appliance Port Numbers**

| Service | Port |
|---|---|
| Automated Service Monitor (ASM) | 30920 |
| HiveServer2 (for High Availability only) | 10015 |
| HTTP Oozie (for High Availability only) | 10021 |
| HTTPS Oozie (for High Availability only) | 10025 |
| MySQL Database | 3306 |
| Oracle Data Integrator Agent | 20910 |
| Oracle NoSQL Database administration | 5001 |
| Oracle NoSQL Database processes | 5010 to 5020 |
| Oracle NoSQL Database registration | 5000 |
| Port map | 111 |
| rpc.statd | 668 |
| ssh | 22 |
| xinetd (service tag) | 6481 |
| Key Management Server (when hosted on the appliance) | 16000 |

# 2.8 Additional Guidance for Securing Clusters

Use the following resources to learn how to further strengthen cluster security.

**Oracle Blogs**

- Secure Your Hadoop Cluster
- Securing Kafka Clusters

**Tutorials**

Securing the Oracle Big Data Appliance

# 3
# Administering Oracle Big Data Appliance

This chapter provides information about the software and services installed on Oracle Big Data Appliance. It contains these sections:

- Monitoring Multiple Clusters Using Oracle Enterprise Manager
- Managing Operations Using Cloudera Manager
- Using Hadoop Monitoring Utilities
- Using Cloudera Hue to Interact With Hadoop
- About the Oracle Big Data Appliance Software
- About CDH Clusters
- Effects of Hardware on Software Availability
- Managing a Hardware Failure
- Stopping and Starting Oracle Big Data Appliance
- Auditing Oracle Big Data Appliance
- Collecting Diagnostic Information for Oracle Customer Support

## 3.1 Monitoring Multiple Clusters Using Oracle Enterprise Manager

An Oracle Enterprise Manager plug-in enables you to use the same system monitoring tool for Oracle Big Data Appliance as you use for Oracle Exadata Database Machine or any other Oracle Database installation. With the plug-in, you can view the status of the installed software components in tabular or graphic presentations, and start and stop these software services. You can also monitor the health of the network and the rack components.

Oracle Enterprise Manager enables you to monitor all Oracle Big Data Appliance racks on the same InfiniBand fabric. It provides summary views of both the rack hardware and the software layout of the logical clusters.

> **Note:**
>
> Before you start, contact Oracle Support for up-to-date information about Enterprise Manager plug-in functionality.

### 3.1.1 Using the Enterprise Manager Web Interface

After opening Oracle Enterprise Manager web interface, logging in, and selecting a target cluster, you can drill down into these primary areas:

- **InfiniBand network**: Network topology and status for InfiniBand switches and ports. See Figure 3-1.

- **Hadoop cluster**: Software services for HDFS, MapReduce, and ZooKeeper.

- **Oracle Big Data Appliance rack**: Hardware status including server hosts, Oracle Integrated Lights Out Manager (Oracle ILOM) servers, power distribution units (PDUs), and the Ethernet switch.

The following figure shows a small section of the cluster home page.

**Figure 3-1    YARN Page in Oracle Enterprise Manager**
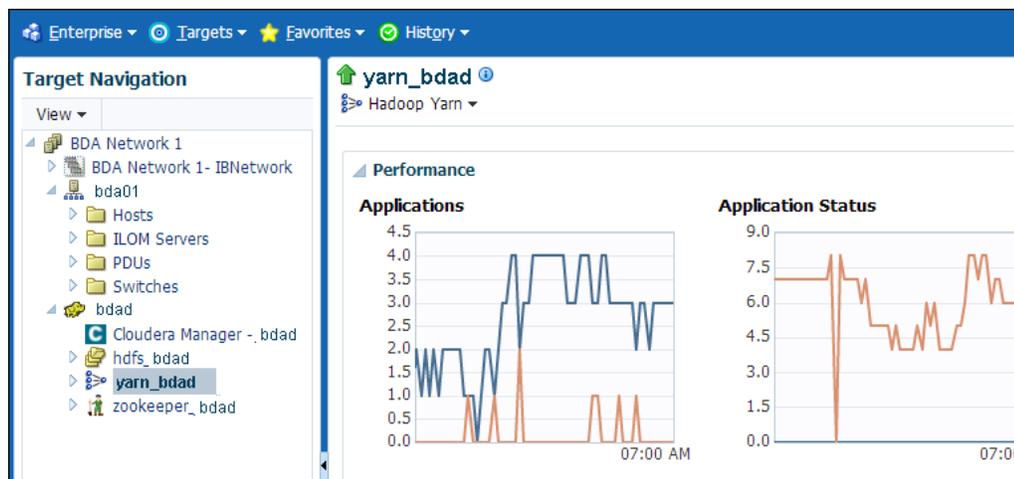


**To monitor Oracle Big Data Appliance using Oracle Enterprise Manager:**

1. Download and install the plug-in. See Oracle Enterprise Manager System Monitoring Plug-in Installation Guide for Oracle Big Data Appliance .

2. Log in to Oracle Enterprise Manager as a privileged user.

3. From the Targets menu, choose **Big Data Appliance** to view the Big Data page. You can see the overall status of the targets already discovered by Oracle Enterprise Manager.

4. Select a target cluster to view its detail pages.

5. Expand the target navigation tree to display the components. Information is available at all levels.

6. Select a component in the tree to display its home page.

7. To change the display, choose an item from the drop-down menu at the top left of the main display area.

## 3.1.2 Using the Enterprise Manager Command-Line Interface

The Enterprise Manager command-line interface (`emcli`) is installed on Oracle Big Data Appliance along with all the other software. It provides the same functionality as the web interface. You must provide credentials to connect to Oracle Management Server.

To get help, enter `emcli help`.

> **See Also:**
>
> *Oracle Enterprise Manager Command Line Interface Guide*

# 3.2 Managing Operations Using Cloudera Manager

Cloudera Manager is installed on Oracle Big Data Appliance to help you with Cloudera's Distribution including Apache Hadoop (CDH) operations. Cloudera Manager provides a single administrative interface to all Oracle Big Data Appliance servers configured as part of the Hadoop cluster.

Cloudera Manager simplifies the performance of these administrative tasks:

- Monitor jobs and services
- Start and stop services
- Manage security and Kerberos credentials
- Monitor user activity
- Monitor the health of the system
- Monitor performance metrics
- Track hardware use (disk, CPU, and RAM)

Cloudera Manager runs on the ResourceManager node (node03) and is available on port 7180.

**To use Cloudera Manager:**

1. Open a browser and enter a URL like the following:

   In this example, `bda1` is the name of the appliance, `node03` is the name of the server, `example.com` is the domain, and `7180` is the default port number for Cloudera Manager.

2. Log in with a user name and password for Cloudera Manager. Only a user with administrative privileges can change the settings. Other Cloudera Manager users can view the status of Oracle Big Data Appliance.

> **See Also:**
>
> https://www.cloudera.com/documentation/enterprise/latest/topics/cm_dg_about.html provides information on Cloudera monitoring and diagnostics.

## 3.2.1 Monitoring the Status of Oracle Big Data Appliance

In Cloudera Manager, you can choose any of the following pages from the menu bar across the top of the display:

- **Home**: Provides a graphic overview of activities and links to all services controlled by Cloudera Manager. See the following figure.

- **Clusters**: Accesses the services on multiple clusters.

- **Hosts**: Monitors the health, disk usage, load, physical memory, swap space, and other statistics for all servers in the cluster.

- **Diagnostics**: Accesses events and logs. Cloudera Manager collects historical information about the systems and services. You can search for a particular phrase for a selected server, service, and time period. You can also select the minimum severity level of the logged messages included in the search: TRACE, DEBUG, INFO, WARN, ERROR, or FATAL.

- **Audits**: Displays the audit history log for a selected time range. You can filter the results by user name, service, or other criteria, and download the log as a CSV file.

- **Charts**: Enables you to view metrics from the Cloudera Manager time-series data store in a variety of chart types, such as line and bar.

- **Backup**: Accesses snapshot policies and scheduled replications.

- **Administration**: Provides a variety of administrative options, including Settings, Alerts, Users, and Kerberos.

Rhe following figure shows the Cloudera Manager home page.

**Figure 3-2    Cloudera Manager Home Page**
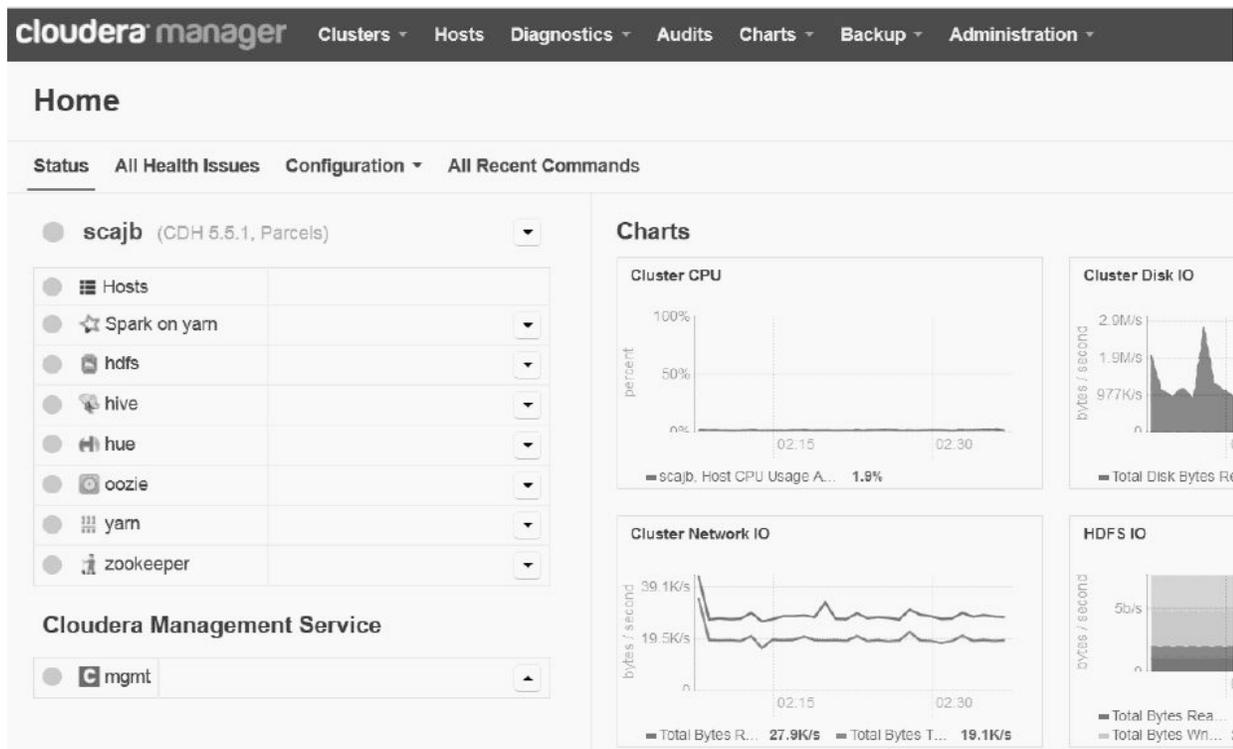
## 3.2.2 Performing Administrative Tasks

As a Cloudera Manager administrator, you can change various properties for monitoring the health and use of Oracle Big Data Appliance, add users, and set up Kerberos security.

**To access Cloudera Manager Administration:**

1. Log in to Cloudera Manager with administrative privileges.

2. Click **Administration**, and select a task from the menu.

## 3.2.3 Managing CDH Services With Cloudera Manager

Cloudera Manager provides the interface for managing these services:

- HDFS

- Hive

- Hue

- Oozie

- YARN

- ZooKeeper

You can use Cloudera Manager to change the configuration of these services, stop, and restart them. Additional services are also available, which require configuration before you can use them. See "Unconfigured Software."

> **Note:**
>
> Manual edits to Linux service scripts or Hadoop configuration files do not affect these services. You must manage and configure them using Cloudera Manager.

# 3.3 Using Hadoop Monitoring Utilities

You also have the option of using the native Hadoop utilities. These utilities are read-only and do not require authentication.

Cloudera Manager provides an easy way to obtain the correct URLs for these utilities. On the YARN service page, expand the Web UI submenu.

## 3.3.1 Monitoring MapReduce Jobs

You can monitor MapReduce jobs using the resource manager interface.

**To monitor MapReduce jobs:**

- Open a browser and enter a URL like the following:

  ```
  http://bda1node03.example.com:8088
  ```

In this example, `bda1` is the name of the rack, `node03` is the name of the server where the YARN resource manager runs, and `8088` is the default port number for the user interface.

The following figure shows the resource manager interface.

**Figure 3-3    YARN Resource Manager Interface**
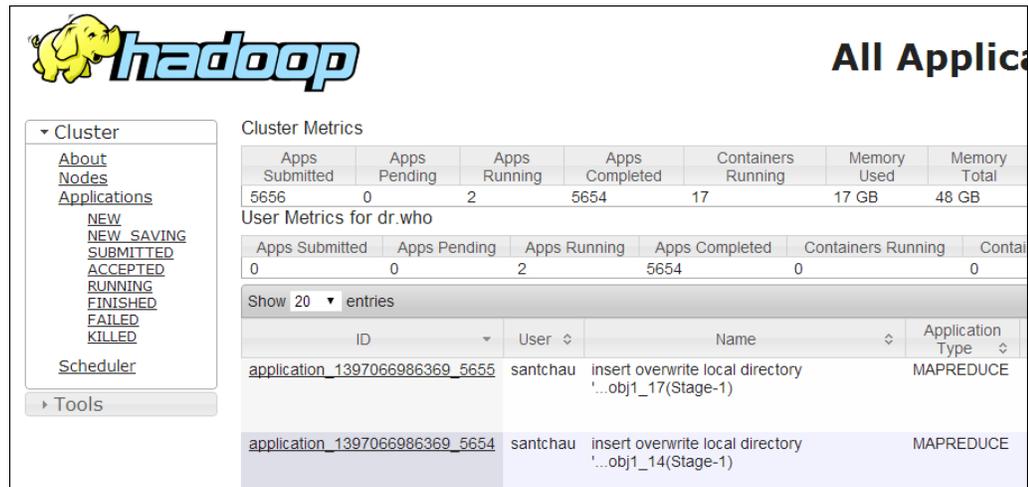


## 3.3.2 Monitoring the Health of HDFS

You can monitor the health of the Hadoop file system by using the DFS health utility on the first two nodes of a cluster.

**To monitor HDFS:**

• Open a browser and enter a URL like the following:

```
http://bda1node01.example.com:50070
```

In this example, `bda1` is the name of the rack, `node01` is the name of the server where the dfshealth utility runs, and `50070` is the default port number for the user interface.

Figure 3-3 shows the DFS health utility interface.

**Figure 3-4    DFS Health Utility**



## 3.4 Using Cloudera Hue to Interact With Hadoop

Hue runs in a browser and provides an easy-to-use interface to several applications to support interaction with Hadoop and HDFS. You can use Hue to perform any of the following tasks:

- Query Hive data stores

- Create, load, and delete Hive tables

- Work with HDFS files and directories

- Create, submit, and monitor MapReduce jobs

- Monitor MapReduce jobs

- Create, edit, and submit workflows using the Oozie dashboard

- Manage users and groups

Hue is automatically installed and configured on Oracle Big Data Appliance. It runs on port 8888 of the ResourceManager node. See the tables in About CDH Clusters for Hue's location within different cluster configurations.

**To use Hue:**

1. Log in to Cloudera Manager and click the **hue** service on the Home page.

2. On the hue page under Quick Links, click `Hue Web UI`.

3. Bookmark the Hue URL, so that you can open Hue directly in your browser. The following URL is an example:

   `http://bda1node03.example.com:8888`

4. Log in with your Hue credentials.

   If Hue accounts have not been created yet, log into the default Hue administrator account by using the following credentials:

   • Username: `admin`

   • Password: `cm-admin-password`

   where `cm-admin-password` is the password specified when the cluster for the Cloudera Manager admin user was activated. You can then create other user and administrator accounts.

The following figure shows the Hive Query Editor.

**Figure 3-5    Hive Query Editor**
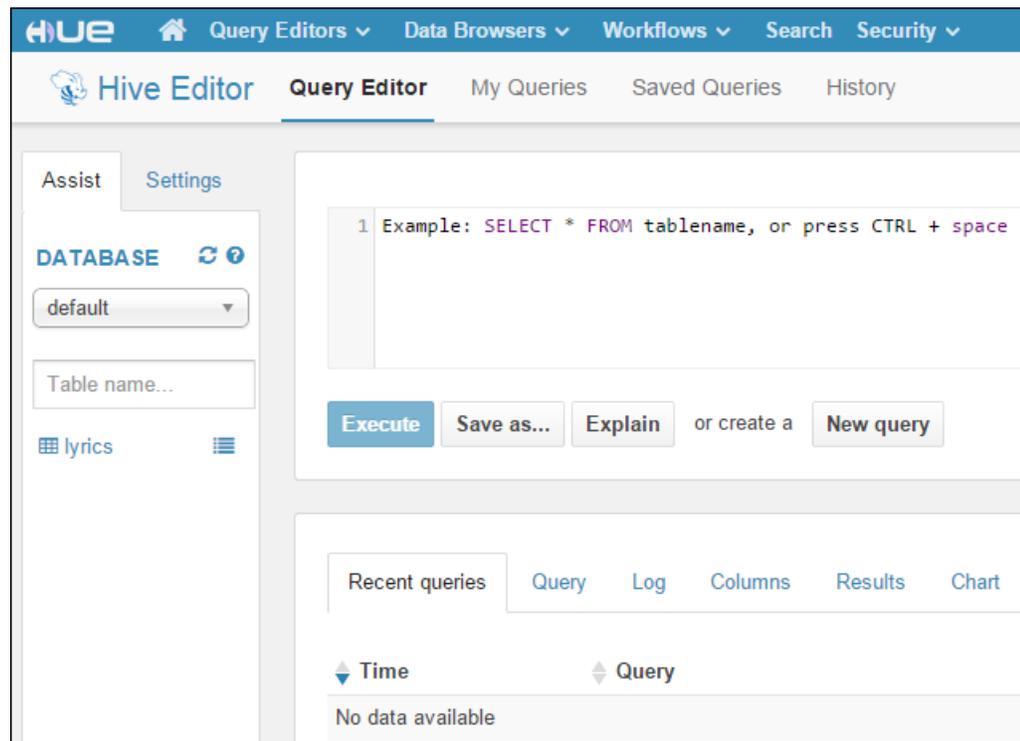
> ✎ **See Also:**
>
> *Hue User Guide* at
>
> https://www.cloudera.com/documentation/enterprise/6/6.0/topics/hue.html

# 3.5 About the Oracle Big Data Appliance Software

The following sections identify the software installed on Oracle Big Data Appliance.

This section contains the following topics:

- Unconfigured Software
- Allocating Resources Among Services

## 3.5.1 Unconfigured Software

Your Oracle Big Data Appliance license includes all components in Cloudera Enterprise Data Hub Edition. All CDH components are installed automatically by the Mammoth utility. Do not download them from the Cloudera website.

However, you must use Cloudera Manager to add some services before you can use them, such as the following:

- Apache Flume
- Apache HBase
- Apache Spark
- Apache Sqoop
- Cloudera Impala
- Cloudera Search

**To add a service:**

1. Log in to Cloudera Manager as the `admin` user.

2. On the Home page, expand the cluster menu in the left panel and choose **Add a Service** to open the Add Service wizard. The first page lists the services you can add.

3. Follow the steps of the wizard.

> ✎ **See Also:**
>
> - For a list of key CDH components:
>
>   `http://www.cloudera.com/content/www/en-us/products/apache-hadoop/key-cdh-components.html`

## 3.5.2 Allocating Resources Among Services

You can allocate resources to each service—HDFS, YARN, Hive, and so forth—as a percentage of the total resource pool. Cloudera Manager automatically calculates the recommended resource management settings based on these percentages. The static service pools isolate services on the cluster, so that a high load on one service as a limited impact on the other services.

**To allocate resources among services:**

1. Log in as `admin` to Cloudera Manager.

2. Open the Clusters menu at the top of the page, then select **Static Service Pools** under Resource Management.

3. Select **Configuration**.

4. Follow the steps of the wizard, or click **Change Settings Directly** to edit the current settings.

# 3.6 About CDH Clusters

There are slight variations in the location of the services within a cluster, depending on the configuration of the cluster.

Note that in general decommissioning or removing roles that were deployed by the Mammoth installer is not supported. In some cases this may be acceptable for slave roles. However, the role must be completely removed from Cloudera Manager. Also, removal of a role may result in lower performance or reduced storage.

This section contains the following topics:

- Roles on a Three-Node Development Cluster

- Role Locations on Rack 1 of a CDH Cluster with Four or More Nodes

- Role Locations on Additional Racks of a Cluster

- About MapReduce

- Automatic Failover of the NameNode

- Automatic Failover of the ResourceManager

## 3.6.1 For New Clusters: High Availability HiveServer2 and Oozie

High Availability HiveServer2 and HA Oozie is automatically included in new clusters.

As of Big Data Appliance 5.2, new clusters automatically include HA HiveServer2 and HA Oozie. This applies to new clusters only.

The Oozie and HiveServer2 roles are duplicated on separate nodes within each new cluster. With each pair of roles, both roles are continuously active. Traffic is routed through an HAProxy load balancer. In order to prevent the load balancer from becoming a single point of failure, there is an option in the Big Data Appliance Configuration Generation Utility to configure High Availability for the load balancer as well. If HA is configured for the load balancers, then the primary balancer is normally active. The secondary load balancer is activated only to provide failover if the first fails.

The Hiveserver2 and Oozie HA enhancements are not added to existing clusters upgraded to this release. At this time there are no manual steps to enable HiveServer2 and Oozie in upgraded clusters.

**Changes in Keystore and Trust Store Locations for HiveServer2 and Oozie**

It is important to note that in conjunction with the introduction of High Availability for HiveServer 2 and Oozie, the Keystore and Truststore locations for these roles has changed as of Big Data Appliance 5.2. You can see this in Configuration Manager, as shown at the end of this topic.

- **HiveServer2 TLS/SSL Server JKS Keystore File Location**: `/opt/cloudera/security/pki_load_balancer/node.jks`

- **HiveServer2 TLS/SSL Client Truststore File**: `/opt/cloudera/security/pki_load_balancer/kpoklkd.truststore`

> **✎ Note:**
>
> These changes apply to new clusters only. Upgraded clusters retain the original pre-5.2 Keystore and Truststore paths.

The following example from Configuration Manager shows the new locations for the Keystore and Truststore. It it also shows the address of the load balancer that handles HiveServer2 and Oozie traffic.

## 3.6.1.1 Configuring HA for the HiveServer2 and Oozie Load Balancer

With High Availability for HiveServer2 and Oozie, the load balancer remains a potential single point of failure. However you have the option to provide High Availability for the load balancer as well.

> **Note:**
>
> There are two prerequisites for configuring HA for the load balancer:
>
> • Provide a public virtual IP address for the load balancer.
>
> • In environments where there are strict firewall rules, these rules must allow the multicast address 224.0.0.18 and the execution of the VRRP protocol.

**Virtual IP Addresses to Support HA for the HiveServer2 and Oozie Load Balancer**

You can choose to set up High Availability for the load balancer when you pre-configure the Mammoth installation on the appliance. To support HA for the load balancer you must provide a public virtual IP address. This is used as a floating IP address that is automatically assigned to the active load balancer. In a failover event, the virtual IP address is re-assigned from the load balancer where the failure occurred to a load balancer on another node and this second load balancer is activated to take over HiverServer2 and Oozie traffic. Under normal conditions, the active load balancer is on Node02 of the cluster and the inactive load balancer is on Node01. The load balancer is actually installed on all nodes but is active on one node only and stopped on all others.

The Oracle Big Data Appliance Configuration Generation Utility now includes a field where you can add the virtual IP address for the load balancer. Also provide a valid hostname for the IP address.

> **✏ Important:**
>
> If no virtual IP address is provided, then there is no failover for the load balancer that handles HiverServer2 and Oozie traffic. Then if the load balancer or the node itself fails, you must manually change any connections to these services from the failed load balancer to a direct connection to one of the HiveServer2 or Oozie instances. Also any `mammoth -c` tests will fail in this circumstance because their entry point is through the load balancer.

See Using Oracle Big Data Appliance Configuration Generation Utility in the *Big Data Appliance Owner's Guide.*

## 3.6.2 Roles on a Three-Node Development Cluster

Oracle Big Data Appliance enables the use of three-node clusters for development purposes.

> **⚠ Caution:**
>
> Three-node clusters are generally not suitable for production environments because all of the nodes are master nodes. This puts constraints on high availability. The minimum recommended cluster size for a production environment is five nodes.

> **✎ Note:**
>
> In the following table, the new roles for HA HiveServer2 and Oozie support that are present on new clusters only are highlighted in italic font and prefixed with an asterisk. These highlighted roles are not present on upgraded clusters.

**Table 3-1    Role Locations for a Three-Node Development Cluster**

| Node1 | Node2 | Node3 |
|---|---|---|
| NameNode | NameNode/Failover | - |
| Failover Controller | Failover Controller | - |
| DataNode | DataNode | DataNode |
| NodeManager | NodeManager | NodeManager |
| JournalNode | JournalNode | JournalNode |
| - | HttpFS | Cloudera Manager and CM roles |
| - | MySQL Backup | MySQL Primary |
| ResourceManager | - | ResourceManager |
| - | - | JobHistory |
| - | ODI | Spark History |
| Hue Server | Hue Server | - |
| Hue Load Balancer | Hue Load Balancer | - |
| ZooKeeper | ZooKeeper | ZooKeeper |
| Active Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled) | Passive Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled) | - |
| Kerberos Master KDC (Only if MIT Kerberos is enabled and on-BDA KDCs are being used.) | Kerberos Slave KDC (Only if MIT Kerberos is enabled and on-BDA KDCs are being used.) | - |
| Sentry Server (if enabled) | Sentry Server (if enabled) | - |
| Hive Metastore | Hive Metastore | - |
| - | WebHCat | - |
| *Oozie | Oozie | - |
| *HiveServer2 | HiveServer2 | - |
| *HAProxy Load Balancer (stopped ) | *HAProxy Load Balancer (started) | - |

## 3.6.3 Role Locations on Rack 1 of a CDH Cluster with Four or More Nodes

All four master nodes and the roles they host are now located in the first rack of a cluster. In earlier releases, some critical roles are hosted on the second rack of a multirack cluster.

Clusters across multiple racks which are upgraded to Release 5.2 from older versions will retain their current multiple-rack layout, in which some critical roles are hosted on the second rack.

The table below identifies the roles on the first rack of CDH cluster. Node1 is the first server in the cluster and Node*nn* is the last server in the cluster. This service layout is the same for a single-rack cluster and the first rack of a multirack cluster.

> **✎ Note:**
>
> As of Big Data Appliance 5.2, new clusters include High Availability Oozie and for HiveServer2, and an HAProxy load balancer to manage traffic for these services. The new HA roles for these services are Node1. The load balancer is on Node2. This applies to new clusters only. Upgraded clusters do NOT include these roles. HA Oozie and HA Hiveserver2 are not present on upgraded clusters and the HAProxy load balancer is also not present. In the following table, these roles that are present on new clusters only are in italic font and marked with an asterisk.

**Table 3-2    Role Locations in the First Rack of a Cluster**

| Node1 | Node2 | Node3 | Node4 | Node5 to *nn* |
|---|---|---|---|---|
| Balancer | - | Cloudera Manager Server | - | - |
| Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent |
| DataNode | DataNode | DataNode | DataNode | DataNode |
| Failover Controller | Failover Controller | Big Data Manager (including BDM-proxy and BDM-notebook) | - | - |
| JournalNode | JournalNode | JournalNode | - | - |
| - | MySQL Backup | MySQL Primary | - | - |
| NameNode | NameNode | Navigator Audit Server and Navigator Metadata Server | - | - |
| NodeManager (in clusters of eight nodes or less) | NodeManager (in clusters of eight nodes or less) | NodeManager | NodeManager | NodeManager |
| Sentry Server (if enabled) | Sentry Server (if enabled) | SparkHistoryServer | Oracle Data Integrator Agent | - |
| Hive Metastore | HttpFS | - | Hive Metastore | - |
| ZooKeeper | ZooKeeper | ZooKeeper | Hive WEbHCat Server | - |
| Hue Server | - | JobHistory | Hue Server | - |
| Hue Load Balancer | - | Hue Load Balancer | - | - |
| Active Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled) | Passive Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled) | ResourceManager | ResourceManager | - |

**Table 3-2 (Cont.) Role Locations in the First Rack of a Cluster**

| Node1 | Node2 | Node3 | Node4 | Node5 to *nn* |
|---|---|---|---|---|
| Kerberos KDC (if MIT Kerberos is enabled and on-BDA KDCs are being used) | Kerberos KDC (if MIT Kerberos is enabled and on-BDA KDCs are being used) | - | - | - |
| If Kerberos is enabled: Hue Kerberos Ticket Renewer, Key Trustee KMS Key Management Server Proxy, Key Trustee Server Active Database | If Kerberos is enabled: Key Trustee KMS Key Management Server Proxy, Key Trustee Server Passive Database | - | If Kerberos is enabled: Hue Kerberos Ticket Renewer | - |
| *Oozie** | - | - | Oozie | - |
| *HiveServer2** | - | - | HiveServer2 | - |
| *HAProxy (stopped)** | *HAProxy ** | - | - | - |

In the Big Data Appliance Configuration Utility where you provide configuration details for the Mammoth installation, you can implement High Availability for the load balancer itself. if you provide a virtual IP address for a second HAProxy instance in the cluster. If a virtual IP address is provided, then when the cluster is started the HAProxy instance on Node2 is in the `start` state and the instance on Node1 is in the `stop` state. If the instance on Node2 transitions to `stop` for any reason, then the instance on Node1 is started.

> **Note:**
> If Oozie High Availability is enabled, then Oozie servers are hosted on Node4 and another node (preferably a ResourceNode) selected by the customer.

## 3.6.4 Role Locations on Additional Racks of a Cluster

> **Note:**
> This layout has changed from previous releases.

All critical roles run on the first rack of the cluster. The roles running on all nodes of rack 2 and additional racks are the same as those running on Node5 and above on rack 1:

- Cloudera Manager Agent
- DataNode
- NodeManager (if cluster includes eight nodes or less)

## 3.6.5 About MapReduce

Yet Another Resource Negotiator (YARN) is the version of MapReduce that runs on Oracle Big Data Appliance. MapReduce applications developed using MapReduce 1 (MRv1) may require recompilation to run under YARN.

The ResourceManager performs all resource management tasks. An MRAppMaster performs the job management tasks. Each job has its own MRAppMaster. The NodeManager has containers that can run a map task, a reduce task, or an MRAppMaster. The NodeManager can dynamically allocate containers using the available memory. This architecture results in improved scalability and better use of the cluster than MRv1.

YARN also manages resources for Spark and Impala.

> **See Also:**
>
> "Running Existing Applications on Hadoop 2 YARN" at
>
> http://hortonworks.com/blog/running-existing-applications-on-hadoop-2-yarn/

## 3.6.6 Automatic Failover of the NameNode

The NameNode is the most critical process because it keeps track of the location of all data. Without a healthy NameNode, the entire cluster fails. Apache Hadoop v0.20.2 and earlier are vulnerable to failure because they have a single name node.

The current version of Cloudera's Distribution including Apache Hadoop in Oracle Big Data Appliance reduces this vulnerability by maintaining redundant NameNodes. The data is replicated during normal operation as follows:

- CDH maintains redundant NameNodes on the first two nodes of a cluster. One of the NameNodes is in active mode, and the other NameNode is in hot standby mode. If the active NameNode fails, then the role of active NameNode automatically fails over to the standby NameNode.

- The NameNode data is written to a mirrored partition so that the loss of a single disk can be tolerated. This mirroring is done at the factory as part of the operating system installation.

- The active NameNode records all changes to the file system metadata in at least two JournalNode processes, which the standby NameNode reads. There are three JournalNodes, which run on the first three nodes of each cluster.

- The changes recorded in the journals are periodically consolidated into a single fsimage file in a process called **checkpointing**.

On Oracle Big Data Appliance, the default log level of the NameNode is DEBUG, to support the Oracle Audit Vault and Database Firewall plugin. If this option is not configured, then you can reset the log level to INFO.

> **✎ Note:**
>
> Oracle Big Data Appliance 2.0 and later releases do not support the use of an external NFS filer for backups and do not use NameNode federation.

The following figure shows the relationships among the processes that support automatic failover of the NameNode.

**Figure 3-6    Automatic Failover of the NameNode on Oracle Big Data Appliance**
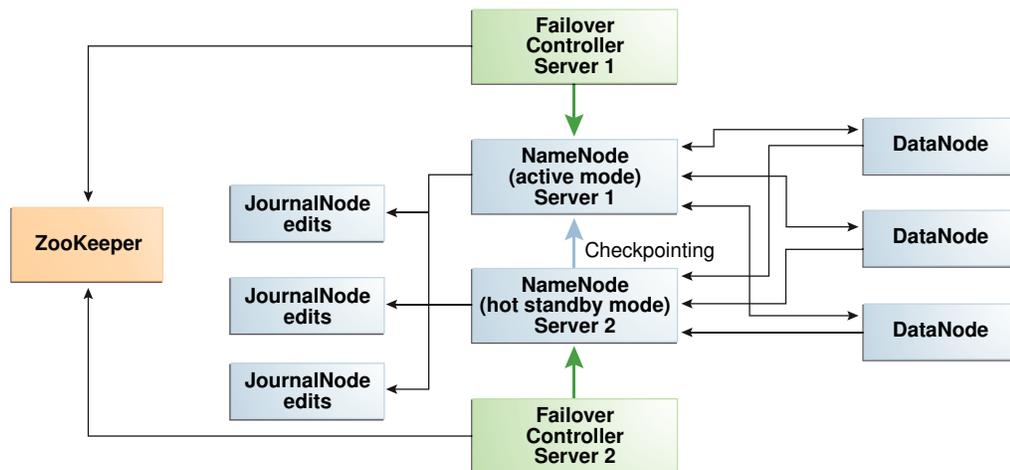


## 3.6.7 Automatic Failover of the ResourceManager

The ResourceManager allocates resources for application tasks and application masters across the cluster. Like the NameNode, the ResourceManager is a critical point of failure for the cluster. If all ResourceManagers fail, then all jobs stop running. Oracle Big Data Appliance supports ResourceManager High Availability in Cloudera 5 to reduce this vulnerability.

CDH maintains redundant ResourceManager services on node03 and node04. One of the services is in active mode, and the other service is in hot standby mode. If the active service fails, then the role of active ResourceManager automatically fails over to the standby service. No failover controllers are required.

The following figure shows the relationships among the processes that support automatic failover of the ResourceManager.

**Figure 3-7    Automatic Failover of the ResourceManager on Oracle Big Data Appliance**
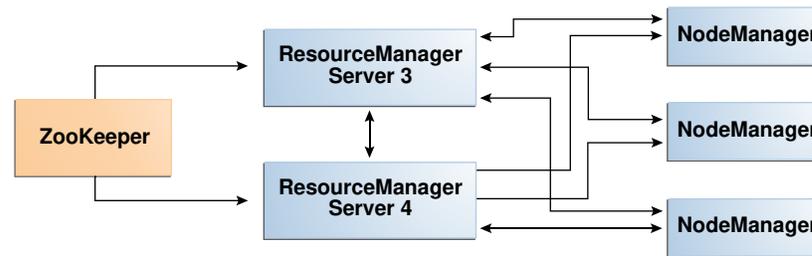


## 3.6.8 Map and Reduce Resource Allocation

Oracle Big Data Appliance dynamically allocates memory to YARN. The allocation depends upon the total memory on the node and whether the node is one of the four critical nodes.

If you add memory, update the NodeManager container memory by increasing it by 80% of the memory added. Leave the remaining 20% for overhead.

# 3.7 About Oracle NoSQL Database Clusters

Oracle NoSQL Database clusters do not have critical nodes and because the storage nodes are replicated by a factor of three, the risk of critical failure is minimal. Administrative services are distributed among the nodes in number equal to the replication factor. You can use the Administration CLI and Admin console to administer the cluster from any node that hosts the administrative processes.

If the node hosting Mammoth fails (the first node of the cluster), then follow the procedure for reinstalling it in "Prerequisites for Managing a Failing Node"

To repair or replace any failing Oracle NoSQL node, follow the procedure in "Managing a Failing Noncritical Node".

# 3.8 About Kafka Clusters

Critical services for Kafka clusters are as follows:

- Node 1: Zookeper, Kafka broker.
- Node 2: MySQL backup database, Zookeper, Kafka broker.
- Node 3: Zookeeper, Mysql primary database, Cloudera Manager server, Kafka broker.

## 3.8.1 Where Do the Services Run on a Kafka Cluster?

As with CDH clusters, there are some minor differences in the way services are distributed in multirack Kafka clusters, depending upon whether the cluster started as a multirack or was extended from a single-rack cluster.

Oracle Big Data Appliance enables the use of three-node clusters for development purposes.

> ⚠️ **Caution:**
>
> Three-node Kafka clusters are generally not suitable for production environments. The minimum recommended cluster size for a Kaka cluster in a production environment is four nodes, which enables recovery of the cluster if a node fails.

**Table 3-3    Service Locations in a Single-Rack Kafka Cluster**

| Node1 | Node2 | Node3 | All Additional Nodes |
|---|---|---|---|
| Kafka Broker | Kafka Broker | Kafka Broker | Kafka Broker |
| Zookeeper | Zookeeper | Zookeeper | Cloudera Manager Agent |
| Kerberos Master KDC (if Kerberos MIT enabled) | Kerberos Slave KDC (if Kerberos MIT enabled) | Cloudera Manager (and CM roles) | — |
| — | MySQL Backup | MySQL Primary | — |

**Table 3-4    Service Locations in First Rack of a Multirack Kafka Cluster (When Extended From a Single-Rack Cluster)**

| Node1 | Node2 | Node3 | All Additional Nodes |
|---|---|---|---|
| Kafka Broker | Kafka Broker | Kafka Broker | Kafka Broker |
| Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent |
| Zookeeper | — | Zookeeper | — |
| — | — | MySQL Primary | — |

**Table 3-5    Service Locations in Additional Racks of a Multirack Kafka Cluster (When Extended From a Single-Rack Cluster)**

| Node1 | Node2 | Node3 | All Additional Nodes |
|---|---|---|---|
| Kafka Broker | Kafka Broker | Kafka Broker | Kafka Broker |
| Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent |
| Zookeeper | — | — | — |
| MySQL Backup (on second rack only) | — | — | — |

**Table 3-6    Service Locations in First Rack of a Multirack Kafka Cluster (When Started as Multirack)**

| Node1 | Node2 | Node3 | All Additional Nodes |
|---|---|---|---|
| Kafka Broker | Kafka Broker | Kafka Broker | Kafka Broker |

**Table 3-6    (Cont.) Service Locations in First Rack of a Multirack Kafka Cluster (When Started as Multirack)**

| Node1 | Node2 | Node3 | All Additional Nodes |
| --- | --- | --- | --- |
| Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent |
| Zookeeper | Zookeeper | — | — |
| — | MySQL Primary | — | — |

**Table 3-7    Service Locations in Additional Racks of a Multirack Kafka Cluster (When Started as Multirack)**

| Node1 | Node2 | Node3 | All Additional Nodes |
| --- | --- | --- | --- |
| Kafka Broker | Kafka Broker | Kafka Broker | Kafka Broker |
| Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent | Cloudera Manager Agent |
| Zookeeper | — | — | — |
| MySQL Backup (on second rack only) | — | — | — |

# 3.9 Effects of Hardware on Software Availability

The effects of a server failure vary depending on the server's function within the CDH cluster. Oracle Big Data Appliance servers are more robust than commodity hardware, so you should experience fewer hardware failures. This section highlights the most important services that run on the various servers of the primary rack. For a full list, see "Role Locations on Rack 1 of a CDH Cluster with Four or More Nodes."

> **Note:**
>
> In a multirack cluster, some critical services run on the first server of the second rack. See "Role Locations on Additional Racks of a Cluster."

## 3.9.1 Logical Disk Layout

The layout of logical disk partitions for X7, X6, X5, and X4 server models is shown below.

On all drive configurations, the operating system is installed on disks 1 and 2. These two disks are mirrored. They include the Linux operating system, all installed software, NameNode data, and MySQL Database data. The NameNode and MySQL Database data are replicated on the two servers for a total of four copies.

Important changes that have occurred in Big Data Appliance disk partitioning over time are:

- In X7 – switchover to the use of EFI instead of BIOS and the replacement of USB with two SSDs

- In Oracle Linux 7 - swap partitioning is dropped.

In the table below, note that Linux disk/partition device names (such as `/dev/sdc` or `/dev/sdc1`) are not stable. They are picked by the kernel at boot time, so they may change as disks are removed and re-added.

**Table 3-8    Oracle Big Data Appliance Server Disk Partioning**

| Disks 1 and 2 (OS) | Disks 3 – 12 (Data) |
|---|---|
| **10 TB Drives (Big Data Appliance X7)** <br> Disk `/dev/sdc`: | **10 TB Drives (Big Data Appliance X7)** <br> Disks `/dev/sde` to `/dev/sdn`: |

<pre>
Number  Start   End                    Number  Start   End     Size
Size    File system                    File system  Name  Flags
Name               Flags                1     1049kB  9796GB  9796GB
 1      1049kB   201MB    200MB        ext4         ext4
fat16      EFI System Partition
boot
 2      201MB    701MB    500MB
ext4       raid
 3      701MB    1049GB 1049GB
ext4       raid
 4      1049GB   9796GB   8747GB
ext4       ext4
</pre>

Disk /dev/sdd:

<pre>
Number  Start   End     Size
File system  Name   Flags
 1      1049kB  201MB    200MB
fat16      fat32
 2      201MB    701MB
500MB                ext4     raid
 3      701MB   1049GB
1049GB               ext4     raid
 4      1049GB  9796GB  8747GB
ext4         ext4
</pre>

| **8 TB Drives (Big Data Appliance X6 and X5)** | **8 TB Drives (Big Data Appliance X6 and X5)** |
|---|---|

<pre>
Number  Start   End      Size        Number  Start   End      Size
File system     Name     Flags       File system  Name     Flags
 1       1049kB  500MB    499MB        1       1049kB  7864GB  7864GB
ext4            primary   boot        ext4         primary
 2       500MB   501GB
500GB                     primary
raid
 3       501GB   550MB    50.0GB
linux-swap(v1)  primary
 4       550GB   7864GB
7314GB   ext4             primary
</pre>

**Table 3-8    (Cont.) Oracle Big Data Appliance Server Disk Partioning**

| Disks 1 and 2 (OS) | Disks 3 – 12 (Data) |
|---|---|
| **4 TB Drives (Big Data Appliance X4)** | **4 TB Drives (Big Data Appliance X4)** |

```
Number  Start    End      Size        Number   Start     End      Size
File system       Name     Flags       File system       Name      Flags
 1      1049kB   500MB    499MB         1       1049kB    4000GB   4000GB
ext4             primary  boot         ext4               primary
 2      500MB    501GB
500GB                     primary
raid
 3      501GB    560GB    59.5GB
linux-swap(v1)   primary
 4      560GB    4000GB   3440GB
ext4             primary
```

## 3.9.2 Critical and Noncritical CDH Nodes

Critical nodes are required for the cluster to operate normally and provide all services to users. In contrast, the cluster continues to operate with no loss of service when a noncritical node fails.

Critical services are installed initially on the first four nodes of the cluster (within the first rack in the case of multirack clusters). The remaining nodes (node05 up to node18) only run noncritical services. If a hardware failure occurs on one of the critical nodes, then the services can be moved to another, noncritical server. For example, if node02 fails, then you might move its critical services node05. Table 3-2 identifies the location of services on the first rack.

### 3.9.2.1 High Availability or Single Points of Failure?

Some services have high availability and automatic failover. Other services have a single point of failure. The following list summarizes the critical services:

• **NameNodes**: High availability with automatic failover

• **ResourceManagers**: High availability with automatic failover

• **MySQL Database**: Primary and backup databases are configured with replication of the primary database to the backup database. There is no automatic failover. If the primary database fails, the functionality of the cluster is diminished, but no data is lost.

• **Cloudera Manager**: The Cloudera Manager server runs on one node. If it fails, then Cloudera Manager functionality is unavailable.

• **Hue server, Hue load balancer, Sentry, Hive metastore** (and also **HiveServer2** and **Oozie** in new clusters only) : High availability

• **Oozie server, Oracle Data Integrator agent**: These services have no redundancy. If the node fails, then the services are unavailable.

## 3.9.2.2 Where Do the Critical Services Run?

Critical services are hosted as shown below. See Role Locations on Rack 1 of a CDH Cluster with Four or More Nodes for the complete list of services on each node.

**Table 3-9    Critical Service Locations on a Single Rack**

| Node Name | Critical Functions |
|---|---|
| First NameNode | Balancer, Failover Controller, JournalNode, NameNode, ZooKeeper, Hue Server. |
| Second NameNode | Failover Controller, JournalNode, MySQL Backup Database, NameNode, ZooKeeper |
| First ResourceManager Node | Cloudera Manager Server, JobHistory, JournalNode, MySQL Primary Database, ResourceManager, ZooKeeper. |
| Second ResourceManager Node | Hive, Hue, Oozie, Solr, Oracle Data Integrator Agent, ResourceManager |

# 3.9.3 First NameNode Node

If the first NameNode fails or goes offline (such as a restart), then the second NameNode automatically takes over to maintain the normal activities of the cluster.

Alternatively, if the second NameNode is already active, it continues without a backup. With only one NameNode, the cluster is vulnerable to failure. The cluster has lost the redundancy needed for automatic failover.

# 3.9.4 Second NameNode Node

If the second NameNode fails, then the function of the NameNode either fails over to the first NameNode (node01) or continues there without a backup. However, the cluster has lost the redundancy needed for automatic failover if the first NameNode also fails.

The MySQL backup database also runs on this node. MySQL Database continues to run, although there is no backup of the master database.

# 3.9.5 First ResourceManager Node

If the first ResourceManager node fails or goes offline (such as in a restart of the server where the node is running), then the second ResourceManager automatically takes over the distribution of MapReduce tasks to specific nodes across the cluster.

If the second ResourceManager is already active when the first ResourceManager becomes inaccessible, then it continues as ResourceManager, but without a backup. With only one ResourceManager, the cluster is vulnerable because it has lost the redundancy needed for automatic failover.

If the first ResourceManager node fails or goes offline (such as a restart), then the second ResourceManager automatically takes over to distribute MapReduce tasks to specific nodes across the cluster.

Alternatively, if the second ResourceManager is already active, it continues without a backup. With only one ResourceManager, the cluster is vulnerable to failure. The cluster has lost the redundancy needed for automatic failover.

These services are also disrupted:

- **Cloudera Manager**: This tool provides central management for the entire CDH cluster. Without this tool, you can still monitor activities using the utilities described in "Using Hadoop Monitoring Utilities".

- **MySQL Database**: Cloudera Manager, Oracle Data Integrator, Hive, and Oozie use MySQL Database. The data is replicated automatically, but you cannot access it when the master database server is down.

## 3.9.6 Second ResourceManager Node

If the second ResourceManager node fails, then the function of the ResourceManager either fails over to the first ResourceManager or continues there without a backup. However, the cluster has lost the redundancy needed for automatic failover if the first ResourceManager also fails.

These services are also disrupted:

- **Oracle Data Integrator Agent** This service supports Oracle Data Integrator, which is one of the Oracle Big Data Connectors. You cannot use Oracle Data Integrator when the ResourceManager node is down.

- **Hive**: Hive provides a SQL-like interface to data that is stored in HDFS. Most of the Oracle Big Data Connectors can access Hive tables, which are not available if this node fails.

- **Hue**: This administrative tool is not available when the ResourceManager node is down.

- **Oozie**: This workflow and coordination service runs on the ResourceManager node, and is unavailable when the node is down.

## 3.9.7 Noncritical CDH Nodes

The noncritical nodes are optional in that Oracle Big Data Appliance continues to operate with no loss of service if a failure occurs. The NameNode automatically replicates the lost data to always maintain three copies. MapReduce jobs execute on copies of the data stored elsewhere in the cluster. The only loss is in computational power, because there are fewer servers on which to distribute the work.

# 3.10 Managing a Hardware Failure

If a server starts failing, you must take steps to maintain the services of the cluster with as little interruption as possible. You can manage a failing server easily using the `bdacli` utility, as described in the following procedures. One of the management steps is called decommissioning. *Decommissioning* stops all roles for all services, thereby preventing data loss. Cloudera Manager requires that you decommission a CDH node before retiring it.

When a noncritical node fails, there is no loss of service. However, when a critical node fails in a CDH cluster, services with a single point of failure are unavailable, as

described in "Effects of Hardware on Software Availability". You must decide between these alternatives:

- Wait for repairs to be made, and endure the loss of service until they are complete.

- Move the critical services to another node. This choice may require that some clients are reconfigured with the address of the new node. For example, if the second ResourceManager node (typically node03) fails, then users must redirect their browsers to the new node to access Cloudera Manager.

You must weigh the loss of services against the inconvenience of reconfiguring the clients.

## 3.10.1 Prerequisites for Managing a Failing Node

Ensure that you do the following before managing a failing or failed server, whether it is configured as a CDH node or an Oracle NoSQL Database node:

- Try restarting the services or rebooting the server.

- Determine whether the failing node is critical or noncritical.

- If the failing node is where Mammoth is installed:

  1. For a CDH node, select a noncritical node in the same cluster as the failing node.

     For a NoSQL node, repair or replace the failed server first, and use it for these steps.

  2. Upload the Mammoth bundle to that node and unzip it.

  3. Extract all files from `BDAMammoth-version.run`, using a command like the following:

     ```
     # ./BDAMammoth-ol6-4.0.0.run
     ```

     Afterward, you must run all Mammoth operations from this node.

     See *Oracle Big Data Appliance Owner's Guide* for information about the Mammoth utility.

  4. Follow the appropriate procedure in this section for managing a failing node.

  Mammoth is installed on the first node of the cluster, unless its services were migrated previously.

## 3.10.2 Managing a Failing CDH Critical Node

The procedure for dealing with a failed critical node is to migrate the critical services to another node.

After migrating the critical services from the failing node to another node, you have several options for reintegrating the failed node into the cluster:

- Reprovision the node
  This procedure reinstalls all of the software required for the node to operate as a DataNode. Reprovisioning is the only option for a node that is not repairable and has been replaced.

- Recommission the node

If you can repair a failed critical node to the extent that the DataNode role is working and are sure that any problems that may interfere with the DataNode role are resolved, you may be able to save time by recommissioning the node instead of reprovisioning it. Recommissioning is a much faster process. It reintegrates the node as a DataNode, but does not perform a full reinstallation and there is no need to reimage. It reverses the decommissioning of the node. A decommission puts the node in quarantine, a recommission takes the node out of quarantine.

> **Note:**
>
> A special procedure is required before you can recommission a failed Node1. See *Preliminary Steps for Recommissioning Node1* at the end of this section.

**To manage a failing critical node:**

1.  Log in as `root` to the "Mammoth node." (The node where Mammoth is installed. This is usually Node1.)

2.  Migrate the services to a noncritical node. (Replace *node_name* below with the name of the failing node.)

    ```
    bdacli admin_cluster migrate node_name
    ```

    When the command finishes, the failng node is decommissioned and its services are now running on a previously noncritical node.

3.  You may want to communicate the change to your user community so that they can redirect their clients to the new critical node as required.

4.  Repair or replace the failed server.

5.  As `root` on the Mammoth node, either reprovision or recommission the repaired or replaced server as a noncritical node. Use the same name as the migrated node for node_name, such as "bda1node02".

    *   To reprovision the node:

        > **Note:**
        >
        > If you intend to reprovision the node, it is recommended (though not required) that you reimage it first to ensure that there are no other problems with the software.

        ```
        # bdacli admin_cluster reprovision <node_name>
        ```

    *   To recommission the node:

        ```
        # bdacli admin_cluster reprovision <node_name>
        ```

6.  From the Mammoth node as `root`, reprovision the repaired or replaced server as a noncritical node. Use the same name as the migrated node for *node_name*, such as bda1node02:

```
bdacli admin_cluster reprovision node_name
```

7. If the failed node supported services like HBase or Impala, which Mammoth installs but does not configure, then use Cloudera Manager to reconfigure them on the new node.

**Preliminary Steps for Recommissioning Node1 Only**

Before recommissioning a failed (and repaired) Node1, do the following:

1. Determine where to relocate the Mammoth role. Mammoth ordinarily runs on Node1 and so when this node fails, the Mammoth role must be transferred to another node. It is best to avoid using other critical nodes and instead choose the first available DataNode.

2. On each node of the cluster:

    a. Update `/opt/oracle/bda/install/state/config.json`. Change `MAMMOTH_NODE` to point to the node where you plan to host the Mammoth role.

    b. Update `/opt/oracle/bda/cluster-hosts-infiniband` to add Node1.

3. On the node where you plan to host the Mammoth role:

    ```
    # setup-root-ssh -C
    ```

4. Use SSH to log on to each node as `root` and edit `/opt/oracle/bda/install/state/config.json`. Remove Node1 from the QUARANTINED arrays – `QUARANTINED_POSNS` and `QUARANTINED_HOSTS`.

5. On the node where you plan to host the Mammoth role:

    a. Run `mammoth -z`.
    This node is now the new Mammoth node.

    b. Log on to each node in the cluster and in `/opt/oracle/bda/install/state/config.json`, re-enter Node1 into the QUARANTINED arrays.

6. You can now recommission Node1. On the Mammoth node, run:

    ```
    # bdacli admin_cluster recommission node0
    ```

## 3.10.3 Managing a Failing Noncritical Node

Use the following procedure to replace a failing node in either a CDH or a NoSQL cluster.

**To manage a failing noncritical node:**

1. Log in as `root` to the node where Mammoth is installed (typically Node1).

2. Decommission the failing node. Replace *node_name* with the name of the failing node.

    ```
    bdacli admin_cluster decommission node_name
    ```

    In Configuraton Manager, verify that the node is decommissioned.

3. After decommissioning the failed node, the next steps depend upon which of these two conditions is true:

- The node can be repaired without reimaging.

- The node must be replaced, or, the node can be repaired, but requires
  reimaging.
  If a failed node cannot be accessed or if the OS is corrupted, it must be
  reimaged.

4. a. If the node is replacement or must be reimaged, then follow instructions in
   Document 1485745.1 in My Oracle Support. This document provides the links
   to imaging instructions for all Big Data Appliance releaeses.
   After reimaging, then reprovision the node:

   ```
   # bdacli admin_cluster reprovision node_name
   ```

   After this, the node will be ready for recommissiong.

   b. If the existing node can be repaired without reimaging, then recommission it.
   There is no need for reprovisioning.

   To recommission the node in either case, log to the Mammoth node as
   `root` on and run the following bdacli command. Use the same name as the
   decommissioned node for *node_name*:

   ```
   bdacli admin_cluster recommission node_name
   ```

5. If the node is part of a CDH cluster, log into Cloudera Manager, and locate the
   recommissioned node. Check that HDFS DataNode, YARN NodeManager, and
   any other roles that should be running are showing a green status light. If they are
   not, then manually restart them.

> ✎ **See Also:**
>
> *Oracle Big Data Appliance Owner's Guide* for the complete `bdacli` syntax

# 3.11 Stopping and Starting Oracle Big Data Appliance

This section describes how to shut down Oracle Big Data Appliance gracefully and
restart it.

- Prerequisites
- Stopping Oracle Big Data Appliance
- Starting Oracle Big Data Appliance

## 3.11.1 Prerequisites

You must have `root` access. Passwordless SSH must be set up on the cluster, so that
you can use the `dcli` utility.

**To ensure that passwordless-ssh is set up:**

1. Log in to the first node of the cluster as `root`.

2. Use a `dcli` command to verify it is working. This command should return the IP address and host name of every node in the cluster:

```
# dcli -C hostname
192.0.2.1: bda1node01.example.com
192.0.2.2: bda1node02.example.com
       .
       .
       .
```

3. If you do not get these results, then set up `dcli` on the cluster:

```
# setup-root-ssh -C
```

> **See Also:**
>
> *Oracle Big Data Appliance Owner's Guide* for details about these commands.

## 3.11.2 Stopping Oracle Big Data Appliance

Follow these procedures to shut down all Oracle Big Data Appliance software and hardware components.

> **Note:**
>
> The following services stop automatically when the system shuts down. You do not need to take any action:
>
> • Oracle Enterprise Manager agent
> • Auto Service Request agents

### 3.11.2.1 Stopping All Managed Services

Use Cloudera Manager to stop the services it manages, including `flume`, `hbase`, `hdfs`, `hive`, `hue`, `mapreduce`, `oozie`, and `zookeeper`.

1. Log in to Cloudera Manager as the `admin` user.

   See "Managing Operations Using Cloudera Manager".

2. In the Status pane of the opening page, expand the menu for the cluster and click **Stop**, and then click **Stop** again when prompted to confirm. See Figure 3-8.

   To navigate to this page, click the **Home** tab, and then the **Status** subtab.

3. On the Command Details page, click **Close** when all processes are stopped.

4. In the same pane under Cloudera Management Services, expand the menu for the `mgmt` service and click **Stop**.

5. Log out of Cloudera Manager.

**Figure 3-8    Stopping HDFS Services**
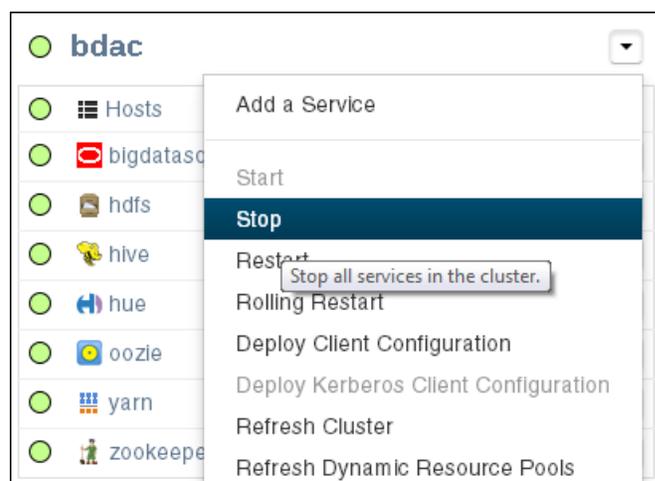


## 3.11.2.2 Stopping Cloudera Manager Server

Follow this procedure to stop Cloudera Manager Server.

1.  Log in as root to the node where Cloudera Manager runs (initially node03).

    > **Note:**
    >
    > The remaining tasks presume that you are logged in to a server as `root`. You can enter the commands from any server by using the `dcli` command. This example runs the `pwd` command on node03 from any node in the cluster:
    >
    > ```
    > # dcli -c node03 pwd
    > ```

2.  Stop the Cloudera Manager server:

    ```
    # service cloudera-scm-server stop
    Stopping cloudera-scm-server:                              [  OK  ]
    ```

3.  Verify that the server is stopped:

    ```
    # service cloudera-scm-server status
    cloudera-scm-server is stopped
    ```

After stopping Cloudera Manager, you cannot access it using the web console.

## 3.11.2.3 Stopping Oracle Data Integrator Agent

If Oracle Data Integrator is used on the cluster:

1.  Check the status of the Oracle Data Integrator agent:

    ```
    # dcli -C service odi-agent status
    ```

2.  Stop the Oracle Data Integrator agent, if it is running:

    ```
    # dcli -C service odi-agent stop
    ```

3. Ensure that the Oracle Data Integrator agent stopped running:

```
# dcli -C service odi-agent status
```

## 3.11.2.4 Dismounting NFS Directories

All nodes share an NFS directory on node03, and additional directories may also exist. If a server with the NFS directory (`/opt/exportdir`) is unavailable, then the other servers hang when attempting to shut down. Thus, you must dismount the NFS directories first.

1. Locate any mounted NFS directories:

```
# dcli -C mount | grep shareddir
192.0.2.1: bda1node03.example.com:/opt/exportdir on /opt/shareddir type nfs
(rw,tcp,soft,intr,timeo=10,retrans=10,addr=192.0.2.3)
192.0.2.2: bda1node03.example.com:/opt/exportdir on /opt/shareddir type nfs
(rw,tcp,soft,intr,timeo=10,retrans=10,addr=192.0.2.3)
192.0.2.3: /opt/exportdir on /opt/shareddir type none (rw,bind)
     .
     .
     .
```

The sample output shows a shared directory on node03 (192.0.2.3).

2. Dismount the shared directory:

```
# dcli -C umount /opt/shareddir
```

3. Dismount any custom NFS directories.

## 3.11.2.5 Stopping the Servers

The Linux `shutdown -h` command powers down individual servers. You can use the `dcli -g` command to stop multiple servers.

1. Create a file that lists the names or IP addresses of the other servers in the cluster, that is, not including the one you are logged in to.

2. Stop the other servers:

```
# dcli -g filename shutdown -h now
```

For *filename*, enter the name of the file that you created in step 1.

3. Stop the server you are logged in to:

```
# shutdown -h now
```

## 3.11.2.6 Stopping the InfiniBand and Cisco Switches

To stop the network switches, turn off a PDU or a breaker in the data center. The switches only turn off when power is removed.

The network switches do not have power buttons. They shut down only when power is removed

To stop the switches, turn off all breakers in the two PDUs.

## 3.11.3 Starting Oracle Big Data Appliance

Follow these procedures to power up the hardware and start all services on Oracle Big Data Appliance.

### 3.11.3.1 Powering Up Oracle Big Data Appliance

1. Switch on all 12 breakers on both PDUs.

2. Allow 4 to 5 minutes for Oracle ILOM and the Linux operating system to start on the servers.

If the servers do not start automatically, then you can start them locally by pressing the power button on the front of the servers, or remotely by using Oracle ILOM. Oracle ILOM has several interfaces, including a command-line interface (CLI) and a web console. Use whichever interface you prefer.

For example, you can log in to the web interface as `root` and start the server from the Remote Power Control page. The URL for Oracle ILOM is the same as for the host, except that it typically has a `-c` or `-ilom` extension. This URL connects to Oracle ILOM for bda1node4:

```
http://bda1node04-ilom.example.com
```

### 3.11.3.2 Starting the HDFS Software Services

Use Cloudera Manager to start all the HDFS services that it controls.

1. Log in as `root` to the node where Cloudera Manager runs (initially node03).

> **✎ Note:**
>
> The remaining tasks presume that you are logged in to a server as `root`. You can enter the commands from any server by using the `dcli` command. This example runs the `pwd` command on node03 from any node in the cluster:
>
> ```
> # dcli -c node03 pwd
> ```

2. Verify that the Cloudera Manager started automatically on node03:

```
# service cloudera-scm-server status
cloudera-scm-server (pid  11399) is running...
```

3. If it is not running, then start it:

```
# service cloudera-scm-server start
```

4. Log in to Cloudera Manager as the `admin` user.

See "Managing Operations Using Cloudera Manager".

5. In the Status pane of the opening page, expand the menu for the cluster and click **Start**, and then click **Start** again when prompted to confirm. See Figure 3-8.

To navigate to this page, click the **Home** tab, and then the **Status** subtab.

6. On the Command Details page, click **Close** when all processes are started.

7. In the same pane under Cloudera Management Services, expand the menu for the `mgmt` service and click **Start**.

8. Log out of Cloudera Manager (optional).

### 3.11.3.3 Starting Oracle Data Integrator Agent

If Oracle Data Integrator is used on this cluster:

1. Check the status of the agent:

```
# /opt/oracle/odiagent/agent_standalone/oracledi/agent/bin/startcmd.sh
OdiPingAgent [-AGENT_NAME=agent_name]
```

2. Start the agent:

```
# /opt/oracle/odiagent/agent_standalone/oracledi/agent/bin/agent.sh [-
NAME=agent_name] [-PORT=port_number]
```

## 3.12 Auditing Oracle Big Data Appliance

> **Notice:**
>
> Audit Vault and Database Firewall is no longer supported for use with Oracle Big Data Appliance. It is recommended that customers use Cloudera Navigator for monitoring.

## 3.13 Collecting Diagnostic Information for Oracle Customer Support

If you need help from Oracle Support to troubleshoot CDH issues, then you should first collect diagnostic information using the `bdadiag` utility with the `cm` option.

**To collect diagnostic information:**

1. Log in to an Oracle Big Data Appliance server as `root`.

2. Run `bdadiag` with at least the `cm` option. You can include additional options on the command line as appropriate. See the *Oracle Big Data Appliance Owner's Guide* for a complete description of the `bdadiag` syntax.

```
# bdadiag cm
```

The command output identifies the name and the location of the diagnostic file.

3. Go to My Oracle Support at http://support.oracle.com.

4. Open a Service Request (SR) if you have not already done so.

5. Upload the `bz2` file into the SR. If the file is too large, then upload it to `sftp.oracle.com`, as described in the next procedure.

**To upload the diagnostics to ftp.oracle.com:**

1. Open an SFTP client and connect to `sftp.oracle.com`. Specify port 2021 and remote directory `/support/incoming/`*`target`*, where *`target`* is the folder name given to you by Oracle Support.

2. Log in with your Oracle Single Sign-on account and password.

3. Upload the diagnostic file to the new directory.

4. Update the SR with the full path and the file name.

> ✎ **See Also:**
>
> My Oracle Support Note 549180.1 at
>
> http://support.oracle.com

# 4

# Supporting User Access to Oracle Big Data Appliance

This chapter describes how you can support users who run MapReduce jobs on Oracle Big Data Appliance or use Oracle Big Data Connectors. It contains these sections:

## 4.1 About Accessing a Kerberos-Secured Cluster

Apache Hadoop is not an inherently secure system. It is protected only by network security. After a connection is established, a client has full access to the system.

To counterbalance this open environment, Oracle Big Data Appliance supports Kerberos security as a software installation option. Kerberos is a network authentication protocol that helps prevent malicious impersonation. Oracle Big Data Appliance support two forms of Kerberos Hadoop security: MIT Kerberos and Microsoft  Active Directory Kerberos.

CDH provides these securities when configured to use Kerberos:

- The CDH master nodes, NameNodes, and JournalNodes resolve the group name so that users cannot manipulate their group memberships.

- Map tasks run under the identity of the user who submitted the job.

- Authorization mechanisms in HDFS and MapReduce help control user access to data.

Oracle Big Data Appliance provides the ability to configure Kerberos security directly using a Microsoft Active Directory (AD) server for Kerberos support (as supported by Cloudera Manager).

You have the option of enabling either form of Kerberos as part of the Mammoth configuration. You can also enable or disable Kerberos later through the `bdacli` utility.

If the Oracle Big Data Appliance cluster is secured with Kerberos, then you must take additional steps to authenticate a CDH client and individual users, as described in this chapter. Users must know their Kerberos user name, password, and realm.

The following table describes some frequently used Kerberos commands. For more information, see the MIT Kerberos documentation.

**Table 4-1    Kerberos User Commands**

| Command | Description |
|---|---|
| kinit *userid@realm* | Obtains a Kerberos ticket. |
| klist | Lists a Kerberos ticket if you have one already. |
| kdestroy | Invalidates a ticket before it expires. |
| kpasswd *userid@realm* | Changes your password. |

> ✎ **See Also:**
>
> - MIT Kerberos Documentation at `http://web.mit.edu/kerberos/krb5-latest/doc/`
>
> - CDH 5 Security Guide at https://www.cloudera.com/documentation/cdh/5-0-x/CDH5-Security-Guide/CDH5-Security-Guide.html.
>
> - If you choose to enable Active Directory Kerberos, either with Mammoth or with the bdacli utility, first read MOS (My Oracle Support) documents 2029378.1 and 2013585.1. These documents explain required preliminary steps and provide important information on known issues.

# 4.2 Providing Remote Client Access to CDH

Oracle Big Data Appliance supports full local access to all commands and utilities in Cloudera's Distribution including Apache Hadoop (CDH).

You can use a browser on any computer that has access to the client network of Oracle Big Data Appliance to access Cloudera Manager, Hadoop Map/Reduce Administration, the Hadoop Task Tracker interface, and other browser-based Hadoop tools.

To issue Hadoop commands remotely, however, you must connect from a system configured as a CDH client with access to the Oracle Big Data Appliance client network. This section explains how to set up a computer so that you can access HDFS and submit MapReduce jobs on Oracle Big Data Appliance.

## 4.2.1 Prerequisites

Ensure that you have met the following prerequisites:

- You must have these access privileges:

  – Sudo access to the client system

  – Login access to Cloudera Manager

  If you do not have these privileges, then contact your system administrator for help.

- The client system must run an operating system that Cloudera supports for CDH 6.

  ```
  https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/
  installation.html
  ```

- The client system must run a compatible version of Oracle JDK 1.8. See the Cloudera documentation Cloudera documentation for a list of versions that are compatible with CDH 6.x releases.

  To verify the version, use this command:

  ```
  $ java -version
  ```

- In the client configuration, ensure that the HDFS property `dfs.client.use.datanode.hostname` is set to "`true`".

  ```
  <property>
    <name>dfs.client.use.datanode.hostname</name>
    <value>true</value>
    <description>Whether clients should use datanode hostnames when
   connecting to datanodes.
    </description>
  </property>
  ```

  This is property is already set to "`true`" if you download the configuration from Cloudera Manager on Oracle Big Data Appliance. It may not be set to "`true`" if you acquire the configuration from other sources, including Cloudera.

## 4.2.2 Installing a CDH Client on Any Supported Operating System

To install a CDH client on any operating system identified as supported by Cloudera, follow these instructions.

1. Log in to the client system.
2. If an earlier version of Hadoop is already installed, then remove it.

   See the Cloudera documentation for removing an earlier CDH version at Uninstalling Cloudera Software and Managed Software
3. Follow the instructions provided in Document 1943912.1 at My Oracle Support.

## 4.2.3 Configuring a CDH Client for an Unsecured Cluster

After installing CDH, you must configure it for use with Oracle Big Data Appliance.

The commands in this procedure that reference `HADOOP_HOME` are used to support older Hadoop clients that require this environment variable. The cluster uses YARN (MRv2) and does not use `HADOOP_HOME`. If no older clients access the cluster, then you can omit these commands.

**To configure the Hadoop client:**

1. Log in to the client system and download the MapReduce client configuration from Cloudera Manager. In this example, Cloudera Manager listens on port 7180 (the default) of `bda01node03.example.com`, and the configuration is stored in a file named `yarn-conf.zip`.

```
$ wget -O yarn-conf.zip http://bda01node03.example.com:7180/cmf/services/3/
client-config
```

2. Unzip `mapreduce-config.zip` into a permanent location on the client system.

```
$ unzip yarn-config.zip
Archive:  yarn-config.zip
  inflating: yarn-conf/hadoop-env.sh
  inflating: yarn-conf/hdfs-site.xml
  inflating: yarn-conf/core-site.xml
  inflating: yarn-conf/mapred-site.xml
  inflating: yarn-conf/log4j.properties
  inflating: yarn-conf/yarn-site.xml
```

All files are stored in a subdirectory named `yarn-config`.

3. Make a backup copy of the Hadoop configuration files:

```
# cp /full_path/yarn-conf /full_path/yarn-conf-bak
```

4. Overwrite the existing configuration files with the downloaded configuration files.

```
# cd /full_path/yarn-conf
# cp * /usr/lib/hadoop/conf
```

# 4.2.4 Configuring a CDH Client for a Kerberos-Secured Cluster

Follow these steps to enable the CDH client to work with a secure CDH cluster.

**To configure a CDH client for Kerberos:**

1. Log in to the system where you created the CDH client.

2. Install the Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files:

   a. Download the files for your Java version:

   Java 6: http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html

   Java 7: http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

   Java 8: http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

   b. Decompress the downloaded file. This example unzips JCE-8:

   ```
   $ unzip UnlimitedJCEPolicyJDK8.zip
   Archive:  UnlimitedJCEPolicyJDK8.zip
      creating: UnlimitedJCEPolicy/
     inflating: UnlimitedJCEPolicy/US_export_policy.jar
     inflating: UnlimitedJCEPolicy/local_policy.jar
     inflating: UnlimitedJCEPolicy/README.txt
   ```

   > **Note:**
   >
   > The JCE-6 files unzip into a directory named jce instead of UnlimitedJCEPolicy.

    **c.** Copy the unzipped files into the Java security directory. For example:

```
$ cp UnlimitedJCEPolicy/* /usr/java/latest/jre/lib/security/
```

**3.** Follow the steps for configuring an unsecured client.

See "Configuring a CDH Client for an Unsecured Cluster."

**4.** Ensure that you have a user ID on the CDH cluster that had been added to the Kerberos realm.

See "Creating Hadoop Cluster Users."

**5.** On the CDH client system, create a file named `krb5.conf` in the `$HADOOP_CONF_DIR` directory. Enter configuration settings like the following, using values appropriate for your installation for the server names, domain, and realm:

```
[libdefaults]
     default_realm = EXAMPLE.COM
     dns_lookup_realm = false
     dns_lookup_kdc = false
     clockskew = 3600
     ticket_lifetime = 24h
     renew_lifetime = 7d
     forwardable = true
[realms]
     EXAMPLE.COM = {
          kdc = bda01node01.example:88
          admin_server = bda01node07:749
          default_domain = example.com
 }
[domain_realm]
        .com = EXAMPLE.COM
```

**6.** Activate the new configuration file:

```
export KRB5_CONFIG=$HADOOP_CONF_DIR/krb5.conf
export HADOOP_OPTS="-Djava.security.krb5.conf=$HADOOP_CONF_DIR/krb5.conf"
export KRB5CCNAME=$HADOOP_CONF_DIR/krb5cc_$USER
```

**7.** Verify that you have access to the Oracle Big Data Appliance cluster.

See "Verifying Access to a Cluster from the CDH Client."

## 4.2.5 Verifying Access to a Cluster from the CDH Client

Follow this procedure to ensure that you have access to the Oracle Big Data Appliance cluster.

**To verify cluster access:**

**1.** To access a Kerberos-protected CDH cluster, first obtain a ticket granting ticket (TGT):

```
$ kinit userid@realm
```

**2.** Verify that you can access HDFS on Oracle Big Data Appliance from the client, by entering a simple Hadoop file system command like the following:

```
$ hadoop fs -ls /user
Found 6 items
drwxr-xr-x   - jdoe      hadoop             0 2014-04-03 00:08 /user/jdoe
drwxrwxrwx   - mapred    hadoop             0 2014-04-02 23:25 /user/history
```

```
drwxr-xr-x   - hive      supergroup        0 2014-04-02 23:27 /user/hive
drwxrwxr-x   - impala    impala            0 2014-04-03 10:45 /user/impala
drwxr-xr-x   - oozie     hadoop            0 2014-04-02 23:27 /user/oozie
drwxr-xr-x   - oracle    hadoop            0 2014-04-03 11:49 /user/oracle
```

Check the output for HDFS users defined on Oracle Big Data Appliance, and not on the client system. You should see the same results as you would after entering the command directly on Oracle Big Data Appliance.

3. Submit a MapReduce job. You must be logged in to the client system under the same user name as your HDFS user name on Oracle Big Data Appliance.

The following example calculates the value of *pi*:

```
$ hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-mapreduce-
examples-*jar pi 10 1000000
Number of Maps  = 10
Samples per Map = 1000000
Wrote input for Map #0
Wrote input for Map #1
    .
    .
    .
Job Finished in 12.403 seconds
Estimated value of Pi is 3.14158440000000000000
```

4. Use Cloudera Manager to verify that the job ran on Oracle Big Data Appliance instead of the local system. Select **mapreduce Jobs** from the Activities menu for a list of jobs.

The following figure shows the job created by the previous example.

**Figure 4-1    Monitoring a YARN Job in Cloudera Manager**
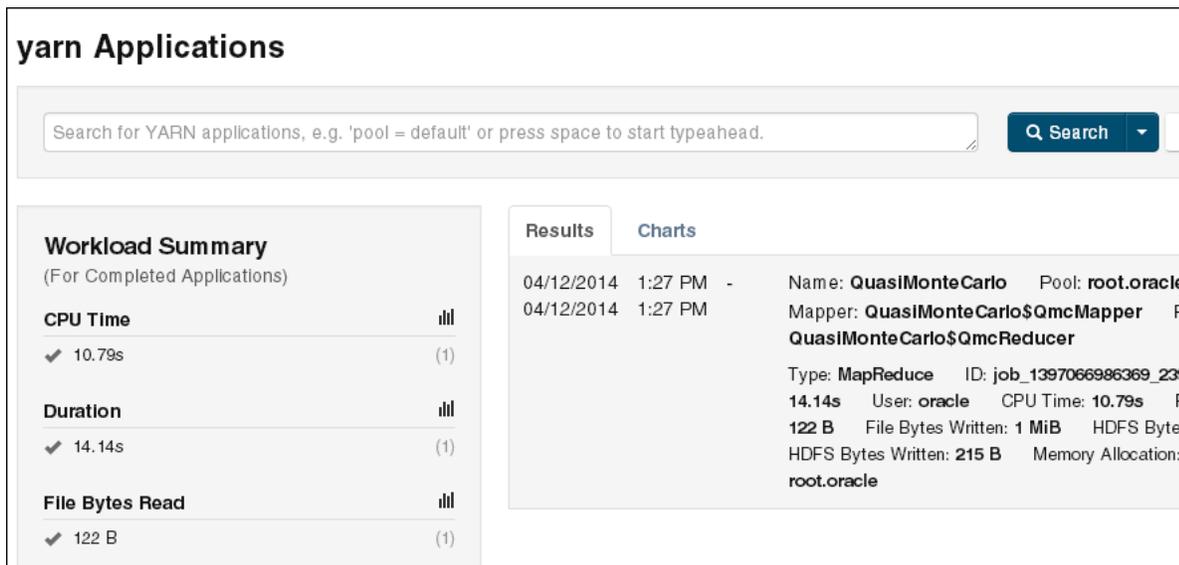


## 4.3 Providing Remote Client Access to Hive

Follow this procedure to provide remote client access to Hive.

**To set up a Hive client:**

1.  Set up a CDH client. See "Providing Remote Client Access to CDH."

2.  Log in to the client system and download the Hive client configuration from Cloudera Manager. In this example, Cloudera Manager listens on port 7180 (the default) of `bda01node03.example.com`, and the configuration is stored in a file named `hive-conf.zip`.

    ```
    $ wget -O hive-conf.zip http://bda01node03.example.com:7180/cmf/services/5/
    client-config
    Length: 1283 (1.3K) [application/zip]
    Saving to: 'hive-conf.zip'
    100%[=======================================>] 1,283        --.-K/s   in 0.001s
    2016-05-15 08:19:06 (2.17 MB/s) - `hive-conf.zip' saved [1283/1283]
    ```

3.  Unzip the file into a permanent installation directory, which will be the Hive configuration directory:

    ```
    $ unzip hive-conf.zip
    Archive:  hive-conf.zip
      inflating: hive-conf/hive-env.sh
      inflating: hive-conf/hive-site.xml
    ```

4.  Download the Hive software from the Cloudera website:

    ```
    $ wget http://archive.cloudera.com/cdh5/cdh/5/hive-<version>-
    cdh5.<version>.tar.gz
    Length: 49637596 (47M) [application/x-gzip]
    Saving to: 'hive-<version>-cdh5.<version>.tar.gz'
    100%[=======================================>] 49,637,596   839K/s   in 47s
    2016-05-15 08:22:18 (1.02 MB/s) - `hive-<version>-cdh5.<version>.tar.gz'
    saved [49637596/49637596]
    ```

5.  Decompress the file into a permanent installation directory, which will be the Hive home directory. The following command unzips the files into the current directory in a subdirectory named `hive-0.12.0-cdh5.0.0`:

    ```
    $ tar -xvzf hive-<version>-cdh5.<version>.tar.gz
    hive-<version>-cdh5.<version>/
    hive-<version>-cdh5.<version>/examples/
          .
          .
          .
    ```

6.  Set the following variables, replacing *hive-home-dir* and *hive-conf-dir* with the directories you created in steps 3 and 5.

    ```
    export HIVE_HOME=hive-home-dir
    export HIVE_CONF_DIR=hive-conf-dir
    alias hive=$HIVE_HOME/bin/hive
    ```

The following steps test whether you successfully set up a Hive client.

**To verify Hive access:**

1.  To access a Kerberos-protected CDH cluster, first obtain a ticket granting ticket (TGT):

    ```
    $ kinit userid@realm
    ```

2.  Open the Hive console:

```
$ hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-
common-<version>-cdh5.<version>.jar!/hive-log4j.properties
Hive history file=/tmp/oracle/hive_job_log_e10527ee-9637-4c08-9559-
a2e5cea6cef1_831268640.txt
hive>
```

3. List all tables:

```
hive> show tables;
OK
src
```

# 4.4 Managing User Accounts

This section describes how to create users who can access HDFS, MapReduce, and Hive. It contains the following topics:

- Creating Hadoop Cluster Users
- Providing User Login Privileges (Optional)

## 4.4.1 Creating Hadoop Cluster Users

When creating user accounts, define them as follows:

- To run MapReduce jobs, users must either be in the `hadoop` group or be granted the equivalent permissions.

- To create and modify tables in Hive, users must either be in the `hive` group or be granted the equivalent permissions.

- To create Hue users, open Hue in a browser and click the User Admin icon. See "Using Cloudera Hue to Interact With Hadoop."

### 4.4.1.1 Creating Users on an Unsecured Cluster

To create a user on an unsecured Hadoop cluster:

1. Open an ssh connection as the `root` user to a noncritical node (node04 to node18).

2. Create the user's home directory:

   ```
   # sudo -u hdfs hadoop fs -mkdir /user/user_name
   ```

   You use `sudo` because the HDFS super user is `hdfs` (not `root`).

3. Change the ownership of the directory:

   ```
   # sudo -u hdfs hadoop fs -chown user_name:hadoop /user/user_name
   ```

4. Verify that the directory is set up correctly:

   ```
   # hadoop fs -ls /user
   ```

5. Create the operating system user across all nodes in the cluster:

   ```
   # dcli useradd -G hadoop,hive[,group_name...] -m user_name
   ```

   In this syntax, replace *group_name* with an existing group and *user_name* with the new name.

6. Verify that the operating system user belongs to the correct groups:

```
# dcli id user_name
```

7. Verify that the user's home directory was created on all nodes:

```
# dcli ls /home | grep user_name
```

**Example 4-1    Creating a Hadoop User**

```
# sudo -u hdfs hadoop fs -mkdir /user/jdoe
# sudo -u hdfs hadoop fs -chown jdoe:hadoop /user/jdoe
# hadoop fs -ls /user
Found 5 items
drwx------    - hdfs     supergroup        0 2013-01-16 13:50 /user/hdfs
drwxr-xr-x   - hive     supergroup        0 2013-01-16 12:58 /user/hive
drwxr-xr-x   - jdoe     jdoe              0 2013-01-18 14:04 /user/jdoe
drwxr-xr-x   - oozie    hadoop            0 2013-01-16 13:01 /user/oozie
drwxr-xr-x   - oracle   hadoop            0 2013-01-16 13:01 /user/oracle
# dcli useradd -G hadoop,hive -m jdoe
# dcli id jdoe
bda1node01: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),127(hive),123(hadoop)
bda1node02: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),123(hadoop),127(hive)
bda1node03: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),123(hadoop),127(hive)
     .
     .
     .
# dcli ls /home | grep jdoe
bda1node01: jdoe
bda1node02: jdoe
bda1node03: jdoe
```

Example 4-1 creates a user named `jdoe` with a primary group of `hadoop` and an addition group of `hive`.

## 4.4.1.2 Creating Users on a Secured Cluster

To create a user on a Kerberos-secured cluster:

1. Connect to Kerberos as the HDFS principal and execute the following commands, replacing `jdoe` with the actual user name:

```
hdfs dfs -mkdir /user/jdoe
hdfs dfs -chown jdoe /user/jdoe
dcli -C useradd -G hadoop,hive -m jdoe
hash=$(echo "hadoop" | openssl passwd -1 -stdin)
dcli -C "usermod --pass='$hash' jdoe"
```

2. Log in to the key distribution center (KDC) and add a principal for the user. In the following example, replace jdoe, bda01node01, and example.com with the correct user name, server name, domain, and realm.

```
ssh -l root bda01node01.example.com kadmin.local
add_principal user_name@EXAMPLE.COM
```

## 4.4.2 Providing User Login Privileges (Optional)

Users do not need login privileges on Oracle Big Data Appliance to run MapReduce jobs from a remote client. However, for those who want to log in to Oracle Big Data Appliance, you must set a password. You can set or reset a password the same way.

**To set a user password across all Oracle Big Data Appliance servers:**

1. Create a Hadoop cluster user as described in "Creating Hadoop Cluster Users.".

2. Confirm that the user does not have a password:

```
# dcli passwd -S user_name
bda1node01.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
bda1node02.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
bda1node03.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
```

   If the output shows either "Empty password" or "Password locked," then you must set a password.

3. Set the password:

```
 hash=$(echo 'password' | openssl passwd -1 -stdin); dcli "usermod --
pass='$hash' user_name"
```

4. Confirm that the password is set across all servers:

```
# dcli passwd -S user_name
bda1node01.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
bda1node02.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
bda1node03.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
```

> ✎ **See Also:**
>
> - *Oracle Big Data Appliance Owner's Guide* for information about `dcli`.
> - The Linux `man` page for the full syntax of the `useradd` command.

# 4.5 Recovering Deleted Files

CDH provides an optional trash facility, so that a deleted file or directory is moved to a trash directory for a set period, instead of being deleted immediately from the system. By default, the trash facility is enabled for HDFS and all HDFS clients.

## 4.5.1 Restoring Files from the Trash

When the trash facility is enabled, you can easily restore files that were previously deleted.

**To restore a file from the trash directory:**

1. Check that the deleted file is in the trash. The following example checks for files deleted by the `oracle` user:

```
$ hadoop fs -ls .Trash/Current/user/oracle
Found 1 items
-rw-r--r--  3 oracle hadoop  242510990 2012-08-31 11:20 /user/oracle/.Trash/
Current/user/oracle/ontime_s.dat
```

2. Move or copy the file to its previous location. The following example moves `ontime_s.dat` from the trash to the HDFS `/user/oracle` directory.

```
$ hadoop fs -mv .Trash/Current/user/oracle/ontime_s.dat /user/oracle/
ontime_s.dat
```

## 4.5.2 Changing the Trash Interval

The **trash interval** is the minimum number of minutes that a file remains in the trash directory before being deleted permanently from the system. The default value is 1 day (24 hours).

**To change the trash interval:**

1. Open Cloudera Manager. See "Managing Operations Using Cloudera Manager".

2. On the Home page under Status, click **hdfs**.

3. On the hdfs page, click the Configuration subtab, and then select **View and Edit**.

4. Search for or scroll down to the Filesystem Trash Interval property under NameNode Default Group. See Figure 4-2.

5. Click the current value, and enter a new value in the pop-up form.

6. Click **Save Changes**.

7. Expand the Actions menu at the top of the page and choose **Restart**.

8. Open a connection as `root` to a node in the cluster.

9. Deploy the new configuration:

```
dcli -C bdagetclientconfig
```

The following figure shows the Filesystem Trash Interval property in Cloudera Manager.

**Figure 4-2    HDFS Property Settings in Cloudera Manager**

## 4.5.3 Disabling the Trash Facility

The trash facility on Oracle Big Data Appliance is enabled by default. You can change this configuration for a cluster. When the trash facility is disabled, deleted files and directories are not moved to the trash. They are not recoverable.

## 4.5.3.1 Completely Disabling the Trash Facility

The following procedure disables the trash facility for HDFS. When the trash facility is completely disabled, the client configuration is irrelevant.

**To completely disable the trash facility:**

1. Open Cloudera Manager. See "Managing Operations Using Cloudera Manager".

2. On the Home page under Status, click **hdfs**.

3. On the hdfs page, click the Configuration subtab, and then select **View and Edit**.

4. Search for or scroll down to the Filesystem Trash Interval property under NameNode Default Group. See Figure 4-2.

5. Click the current value, and enter a value of 0 (zero) in the pop-up form.

6. Click **Save Changes**.

7. Expand the Actions menu at the top of the page and choose **Restart**.

## 4.5.3.2 Disabling the Trash Facility for Local HDFS Clients

All HDFS clients that are installed on Oracle Big Data Appliance are configured to use the trash facility. An **HDFS client** is any software that connects to HDFS to perform operations such as listing HDFS files, copying files to and from HDFS, and creating directories.

You can use Cloudera Manager to change the local client configuration setting, although the trash facility is still enabled.

> ✎ **Note:**
>
> If you do not want any clients to use the trash, then you can completely disable the trash facility. See "Completely Disabling the Trash Facility."

**To disable the trash facility for local HDFS clients:**

1. Open Cloudera Manager. See "Managing Operations Using Cloudera Manager".

2. On the Home page under Status, click **hdfs**.

3. On the hdfs page, click the **Configuration** subtab, and then select **View and Edit**.

4. Search for or scroll down to the Filesystem Trash Interval property under Gateway Default Group. See Figure 4-2.

5. Search for or scroll down to the Use Trash property under Client Settings. See Figure 4-2.

Chapter 4
Recovering Deleted Files

6. Deselect the Use Trash check box.

7. Click **Save Changes**. This setting is used to configure all new HDFS clients downloaded to Oracle Big Data Appliance.

8. Open a connection as `root` to a node in the cluster.

9. Deploy the new configuration:

```
dcli -C bdagetclientconfig
```

## 4.5.3.3 Disabling the Trash Facility for a Remote HDFS Client

Remote HDFS clients are typically configured by downloading and installing a CDH client, as described in "Providing Remote Client Access to CDH." Oracle SQL Connector for HDFS and Oracle R Advanced Analytics for Hadoop are examples of remote clients.

**To disable the trash facility for a remote HDFS client:**

1. Open a connection to the system where the CDH client is installed.

2. Open `/etc/hadoop/conf/hdfs-site.xml` in a text editor.

3. Set the trash interval to zero:

```
<property>
    <name>fs.trash.interval</name>
    <value>0</value>
</property>
```

4. Save the file.

# 5

# Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance

This chapter provides information about optimizing communications between Oracle Exadata Database Machine and Oracle Big Data Appliance. It describes how you can configure Oracle Exadata Database Machine to use InfiniBand alone, or SDP over InfiniBand, to communicate with Oracle Big Data Appliance.

This chapter contains the following sections:

- About Optimizing Communications
- Prerequisites for Optimizing Communications
- Specifying the InfiniBand Connections to Oracle Big Data Appliance
- Specifying the InfiniBand Connections to Oracle Exadata Database Machine
- Enabling SDP on Exadata Database Nodes
- Creating an SDP Listener on the InfiniBand Network

## 5.1 About Optimizing Communications

Oracle Exadata Database Machine and Oracle Big Data Appliance use Ethernet by default, although typically they are also connected by an InfiniBand network. Ethernet communications are much slower than InfiniBand. After you configure Oracle Exadata Database Machine to communicate using InfiniBand, it can obtain data from Oracle Big Data Appliance many times faster than before.

Moreover, client applications that run on Oracle Big Data Appliance and push the data to Oracle Database can use Sockets Direct Protocol (SDP) for an additional performance boost. SDP is a standard communication protocol for clustered server environments, providing an interface between the network interface card and the application. By using SDP, applications place most of the messaging burden upon the network interface card, which frees the CPU for other tasks. As a result, SDP decreases network latency and CPU utilization, and thereby improves performance.

### 5.1.1 About Applications that Pull Data Into Oracle Exadata Database Machine

Oracle SQL Connector for Hadoop Distributed File System (HDFS) is an example of an application that pulls data into Oracle Exadata Database Machine. The connector enables an Oracle external table to access data stored in either HDFS files or a Hive table.

The external table provide access to the HDFS data. You can use the external table for querying HDFS data or for loading it into an Oracle database table.

---

Oracle SQL Connector for HDFS functions as a Hadoop client running on the database servers in Oracle Exadata Database Machine.

If you use Oracle SQL Connector for HDFS or another tool that pulls the data into Oracle Exadata Database Machine, then for the best performance, you should configure the system to use InfiniBand. See "Specifying the InfiniBand Connections to Oracle Big Data Appliance."

> **See Also :**
>
> Oracle Big Data Connectors
> for information about Oracle SQL Connector for HDFS

## 5.1.2 About Applications that Push Data Into Oracle Exadata Database Machine

Oracle Loader for Hadoop is an example of an application that pushes data into Oracle Exadata Database Machine. The connector is an efficient and high-performance loader for fast movement of data from a Hadoop cluster into a table in an Oracle database. You can use it to load data from Oracle Big Data Appliance to Oracle Exadata Database Machine.

Oracle Loader for Hadoop functions as a database client running on the Oracle Big Data Appliance. It must make database connections from Oracle Big Data Appliance to Oracle Exadata Database Machine over the InfiniBand network. Use of Sockets Direct Protocol (SDP) for these database connections further improves performance.

If you use Oracle Loader for Hadoop or another tool that pushes the data into Oracle Exadata Database Machine, then for the best performance, you should configure the system to use SDP over InfiniBand as described in this chapter.

# 5.2 Prerequisites for Optimizing Communications

Oracle Big Data Appliance and Oracle Exadata Database Machine racks must be cabled together using InfiniBand cables. The IP addresses must be unique across all racks and use the same subnet for the InfiniBand network.

> **See Also:**
>
> - *Oracle Big Data Appliance Owner's Guide* about multirack cabling
> - *Oracle Big Data Appliance Owner's Guide* about IP addresses and subnets

# 5.3 Specifying the InfiniBand Connections to Oracle Big Data Appliance

You can configure Oracle Exadata Database Machine to use the InfiniBand IP addresses of the Oracle Big Data Appliance servers. Otherwise, the default network is Ethernet. Use of the InfiniBand network improves the performance of all data transfers between Oracle Big Data Appliance and Oracle Exadata Database Machine.

**To identify the Oracle Big Data Appliance InfiniBand IP addresses:**

1. If you have not done so already, install a CDH client on Oracle Exadata Database Machine. See "Providing Remote Client Access to CDH."

2. Obtain a list of private host names and InfiniBand IP addresses for all Oracle Big Data Appliance servers.

   An Oracle Big Data Appliance rack can have 6, 12, or 18 servers.

3. Log in to Oracle Exadata Database Machine with `root` privileges.

4. Edit `/etc/hosts` on Oracle Exadata Database Machine and add the Oracle Big Data Appliance host names and InfiniBand IP addresses. The following example shows the sequential IP numbering:

   ```
   192.168.8.1        bda1node01.example.com     bda1node01
   192.168.8.2        bda1node02.example.com     bda1node02
   192.168.8.3        bda1node03.example.com     bda1node03
   192.168.8.4        bda1node04.example.com     bda1node04
   192.168.8.5        bda1node05.example.com     bda1node05
   192.168.8.6        bda1node06.example.com     bda1node06
   ```

5. Check `/etc/nsswitch.conf` for a line like the following:

   ```
   hosts:      files dns
   ```

   Ensure that the line does not reverse the order (`dns files`); if it does, your additions to `/etc/hosts` will not be used. Edit the file if necessary.

6. Ping all Oracle Big Data Appliance servers. Ensure that `ping` completes and shows the InfiniBand IP addresses.

   ```
   # ping bda1node01.example.com
   PING bda1node01.example.com (192.168.8.1) 56(84) bytes of data.
   64 bytes from bda1node01.example.com (192.168.8.1): icmp_seq=1 ttl=50
   time=20.2 ms
         .
         .
         .
   ```

7. Run CDH locally on Oracle Exadata Database Machine and test HDFS functionality by uploading a large file to an Oracle Big Data Appliance server. Check that your network monitoring tools (such as `sar`) show I/O activity on the InfiniBand devices.

   To upload a file, use syntax like the following, which copies localfile.dat to the HDFS testdir directory on node05 of Oracle Big Data Appliance:

   ```
   hadoop fs -put localfile.dat hdfs://bda1node05.example.com/testdir/
   ```

# 5.4 Specifying the InfiniBand Connections to Oracle Exadata Database Machine

You can configure Oracle Big Data Appliance to use the InfiniBand IP addresses of the Oracle Exadata Database Machine servers. This configuration supports applications on Oracle Big Data Appliance that must connect to Oracle Exadata Database Machine.

**To identify the Oracle Exadata Database Machine InfiniBand IP addresses:**

1. Obtain a list of private host names and InfiniBand IP addresses for all Oracle Exadata Database Machine servers.

2. Log in to Oracle Big Data Appliance with `root` privileges.

3. Edit `/etc/hosts` on Oracle Big Data Appliance and add the Oracle Exadata Database Machine host names and InfiniBand IP addresses.

4. Check `/etc/nsswitch.conf` for a line like the following:

   ```
   hosts:      files dns
   ```

   Ensure that the line does not reverse the order (`dns files`); if it does, your additions to /etc/hosts will not be used. Edit the file if necessary.

5. Restart the `dnsmasq` service:

   ```
   # service dnsmasq restart
   ```

6. Ping all Oracle Exadata Database Machine servers. Ensure that `ping` completes and shows the InfiniBand IP addresses.

7. Test the connection by downloading a large file to an Oracle Exadata Database Machine server. Check that your network monitoring tools (such as `sar`) show I/O activity on the InfiniBand devices.

   To download a file, use syntax like the following, which copies a file named `mydata.json` to the `dm01ce108` storage server:

   ```
   $ scp mydata.json oracle@dm01cel08-priv.example.com:mybigdata.json
   oracle@dm01cel08-priv.example.com's password: password
   ```

# 5.5 Enabling SDP on Exadata Database Nodes

SDP improves the performance of client applications that run on Oracle Big Data Appliance and push large data loads to Oracle Database on Oracle Exadata Database Machine.

The following procedure describes how to enable SDP on the database nodes in an Oracle Exadata Database Machine running Oracle Linux. You must also configure your application on a job-by-job basis to use SDP.

**To enable SDP on Oracle Exadata Database Machine:**

1. Open `/etc/infiniband/openib.conf` file in a text editor, and add the following line:

   ```
   set: SDP_LOAD=yes
   ```

2. Save these changes and close the file.

3. To enable both SDP and TCP, open `/etc/ofed/libsdp.conf` in a text editor, and add the `use both` rule:

```
use both server * :
use both client * :
```

4. Save these changes and close the file.

5. Open `/etc/modprobe.conf` file in a text editor, and add this setting:

```
options ib_sdp sdp_zcopy_thresh=0 recv_poll=0
```

6. Save these changes and close the file.

7. Replicate these changes across all database nodes in the Oracle Exadata Database Machine rack.

8. Restart all database nodes for the changes to take effect.

9. If you have multiple Oracle Exadata Database Machine racks, then repeat these steps on all of them.

**To specify SDP protocol for a load job:**

1. Add JVM options to the `HADOOP_OPTS` environment variable to enable JDBC SDP export:

```
HADOOP_OPTS="-Doracle.net.SDP=true -Djava.net.preferIPv4Stack=true"
```

2. In either the Hadoop command or the configuration file for the job, set the `mapred.child.java.opts` configuration property to enable the child task JVMs for SDP.

   For example, use these options in the command line for a MapReduce job:

```
-D mapred.child.java.opts="-Doracle.net.SDP=true -
Djava.net.preferIPv4Stack=true"
```

3. Configure standard Ethernet communications for the job.

   For example, Oracle Loader for Hadoop reads the value of the `oracle.hadoop.loader.connection.url` property from a job configuration file. The value has this syntax:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP)(HOST=hostName)(PORT=portNumber)))
    (CONNECT_DATA=(SERVICE_NAME=serviceName)))
```

   Replace *hostName*, *portNumber*, and *serviceName* with the appropriate values to identify the SDP listener on your Oracle Exadata Database Machine.

4. Configure the Oracle listener on Exadata to support the SDP protocol and bind it to a specific port address (such as 1522).

   For example, Oracle Loader for Hadoop reads the value of the `oracle.hadoop.loader.connection.oci_url` property from a job configuration file. The value has this syntax:

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=SDP)
    (HOST=hostName) (PORT=portNumber))
    (CONNECT_DATA=(SERVICE_NAME=serviceName)))
```

# 5.6 Creating an SDP Listener on the InfiniBand Network

To add a listener for the Oracle Big Data Appliance connections coming in on the InfiniBand network, first add a network resource for the InfiniBand network with virtual IP addresses.

> **Note:**
>
> These instructions apply to Exadata V2, X2-2 , and X3-2 nodes running Oracle Linux 5. Document 1580584.1 in My Oracle Support provides instructions for these same systems as well as for X4-2, X5-2, and X6-2 nodes running Oracle Linux 6 .

This example below lists two nodes for an Oracle Exadata Database Machine quarter rack. If you have an Oracle Exadata Database Machine half or full rack, you must repeat node-specific lines for each node in the cluster.

1. Edit `/etc/hosts` on each node in the Exadata rack to add the virtual IP addresses for the InfiniBand network. Make sure that these IP addresses are not in use. For example:

   ```
   # Added for Listener over IB
   192.168.10.21 dm01db01-ibvip.example.com dm01db01-ibvip
   192.168.10.22 dm01db02-ibvip.example.com dm01db02-ibvip
   ```

2. As the `root` user, create a network resource on one database node for the InfiniBand network. For example:

   ```
   # /u01/app/grid/product/12.1.0.1/bin/srvctl add network -k 2 -S
   192.168.10.0/255.255.255.0/bondib0
   ```

3. Verify that the network was added correctly with a command like the following examples:

   ```
   # /u01/app/grid/product/12.1.0.1/bin/crsctl stat res -t | grep net
   ora.net1.network
   ora.net2.network -- Output indicating new Network resource
   ```

   *or*

   ```
   # /u01/app/grid/product/12.1.0.1/bin/srvctl config network -k 2
   Network exists: 2/192.168.10.0/255.255.255.0/bondib0, type static -- Output
   indicating Network resource on the 192.168.10.0 subnet
   ```

4. Add the virtual IP addresses on the network created in Step 2, for each node in the cluster. For example:

   ```
   # srvctl add vip -n dm01db01 -A dm01db01-ibvip/255.255.255.0/bondib0 -k 2
   #
   # srvctl add vip -n dm01db02 -A dm01db02-ibvip/255.255.255.0/bondib0 -k 2
   ```

5. As the `oracle` user who owns Grid Infrastructure Home, add a listener for the virtual IP addresses created in Step 4.

   ```
   # srvctl add listener -l LISTENER_IB -k 2 -p TCP:1522,/SDP:1522
   ```

6. For each database that will accept connections from the middle tier, modify the `listener_networks init` parameter to allow load balancing and failover across multiple networks (Ethernet and InfiniBand). You can either enter the full `TNSNAMES` syntax in the initialization parameter or create entries in `tnsnames.ora` in the `$ORACLE_HOME/network/admin` directory. The `TNSNAMES.ORA` entries must exist in `GRID_HOME`. The following example first updates `tnsnames.ora`.

Complete this step on each node in the cluster with the correct IP addresses for that node. `LISTENER_IBREMOTE` should list all other nodes that are in the cluster. `DBM_IB` should list all nodes in the cluster.

> **Note:**
>
> The database instance reads the `TNSNAMES` only on startup. Thus, if you modify an entry that is referred to by any `init.ora` parameter (`LISTENER_NETWORKS`), then you must either restart the instance or issue an `ALTER SYSTEM SET LISTENER_NETWORKS` command for the modifications to take affect by the instance.

```
DBM =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01-scan)(PORT = 1521))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = dbm)
))
DBM_IB =
(DESCRIPTION =
(LOAD_BALANCE=on)
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01db01-ibvip)(PORT = 1522))
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01db02-ibvip)(PORT = 1522))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = dbm)
))
LISTENER_IBREMOTE =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01db02-ibvip.mycompany.com)(PORT =
1522))
))
LISTENER_IBLOCAL =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01db01-ibvip.mycompany.com)(PORT =
1522))
(ADDRESS = (PROTOCOL = SDP)(HOST = dm01db01-ibvip.mycompany.com)(PORT =
1523))
))
LISTENER_IPLOCAL =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm0101-vip.mycompany.com)(PORT = 1521))
))
LISTENER_IPREMOTE =
(DESCRIPTION =
(ADDRESS_LIST =
```

```
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01-scan.mycompany.com)(PORT = 1521))
))
```

7. Connect to the database instance as `sysdba`.

8. Modify the `listener_networks init` parameter by using the SQL `ALTER SYSTEM` command:

```
SQL> alter system set listener_networks=
     '((NAME=network2) (LOCAL_LISTENER=LISTENER_IBLOCAL)
        (REMOTE_LISTENER=LISTENER_IBREMOTE))',
     '((NAME=network1)(LOCAL_LISTENER=LISTENER_IPLOCAL)
        (REMOTE_LISTENER=LISTENER_IPREMOTE))' scope=both;
```

9. On the Linux command line, use the `srvctl` command to restart `LISTENER_IB` to implement the modification in Step 7:

```
# srvctl stop listener -l LISTENER_IB
# srvctl start listener -l LISTENER_IB
```

# Part II

# Oracle DataSource for Apache Hadoop

This part describes Oracle DataSource for Apache Hadoop (OD4H) storage handler for Oracle Database. It contains the following chapters:

- Oracle DataSource for Apache Hadoop (OD4H)

# 6

# Oracle DataSource for Apache Hadoop (OD4H)

Oracle DataSource for Apache Hadoop (formerly known as Oracle Table Access for Apache Hadoop) allows direct, fast, parallel, secure, and consistent access to master data in Oracle Database using Spark SQL through Hive metastore. This chapter discusses Oracle DataSource for Apache Hadoop (OD4H) in the following sections:

- Operational Data, Big Data and Requirements
- Overview of Oracle DataSource for Apache Hadoop (OD4H)
- How Does OD4H Work?
- Features of OD4H
- Using Hive SQL with OD4H
- Using Spark SQL with OD4H
- Writing Back To Oracle Database

## 6.1 Operational Data, Big Data and Requirements

The typical data architecture in most companies comprises of the following components:

- Oracle Database for operational, transactional, and master data, which is shared business object
- Big Data

Hadoop applications (such as Master Data Management (MDM) or Events processing) need access to data in both Hadoop storages (such as HDFS and NoSQL Database as a landing point for Web logs) and Oracle Database (as the reliable and auditable source of truth). You can adopt one of the following approaches to process such data that reside in both Hadoop storage and Oracle Database:

- Perform an Extract, Transform, and Load (ETL) Copy using tools such as Copy to Hadoop tool of Oracle
- Access the data directly using Oracle Big Data SQL and Oracle DataSource for Apache Hadoop (OD4H)

This chapter discusses Oracle DataSource for Apache Hadoop (OD4H).

## 6.2 Overview of Oracle DataSource for Apache Hadoop (OD4H)

Oracle DataSource for Apache Hadoop (OD4H) is the storage handler for Oracle Database that uses HCatalog and InputFormat.

This section discusses the following concepts:

- Opportunity with Hadoop 2.x
- Oracle Tables as Hadoop Data Source
- External Tables

## 6.2.1 Opportunity with Hadoop 2.x

Hadoop 2.x architecture decouples compute engines from cluster resources management and storages. It enables:

- A variety of SQL query engines. For instance, Hive SQL, Spark SQL, Big Data SQL, and so on.
- A variety of programmatic compute engines. For instance, MapReduce, Pig, Storm, Solr, Cascading, and so on.
- Elastic allocation of compute resources (CPU, memory) through YARN.
- A variety of data stores such as HDFS, NoSQL, as well as remote storages through HCatalog, InputFormat, OutputFormat and StorageHandler interfaces.

Oracle DataSource for Apache Hadoop (OD4H) is the storage handler for Oracle Database that uses HCatalog and InputFormat.

Following is an illustration of Hadoop 2.0 Architecture:

**Figure 6-1    Hadoop 2.0 Architecture**
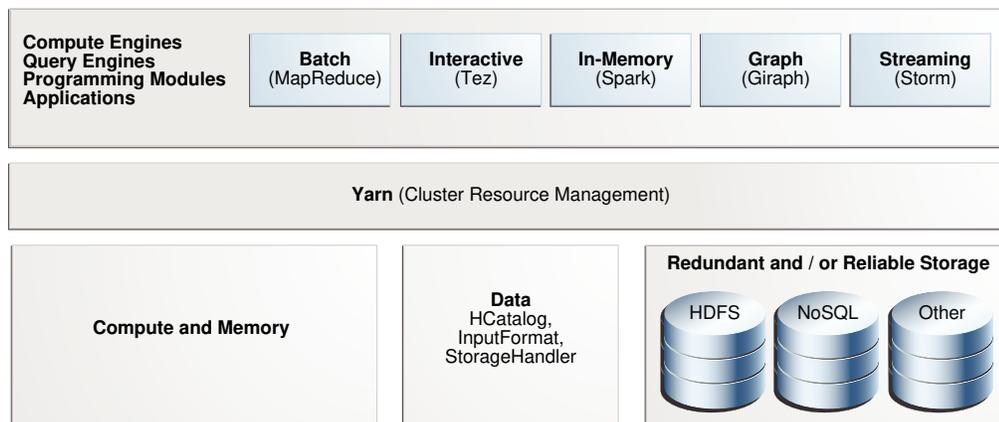


## 6.2.2 Oracle Tables as Hadoop Data Source

OD4H enables current and ad-hoc querying. This makes querying data faster and more secure. You can query data directly and retrieve only the data that you need, when you need it.

OD4H also provides Oracle's end-to-end security. This includes Identity Management, Column Masking, and Label and Row Security.

OD4H also furnishes direct access for Hadoop and Spark APIs such as Pig, MapReduce and others.

## 6.2.3 External Tables

External Tables turn Oracle tables into Hadoop and/or Spark datasources. The DDL for declaring External Tables is as follows:

```
CREATE[TEMPORARY] EXTERNAL TABLE [IF NOT EXISTS]  [db_name.]table_name
[(col_name data_type [COMMENTcol_comment],...)]
[COMMENT table_comment]
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
[WITHSERDEPROPERTIES(...)]
[TBLPROPERTIES (property_name=property_value,...)]

data_type
|SMALLINT
|INT
|BIGINT
|BOOLEAN
|FLOAT
|DOUBLE
|STRING
|BINARY
|TIMESTAMP
|DECIMAL
|DECIMAL(precision,scale)
|VARCHAR
|CHAR
```

> **See Also:**
>
> Refer the following link for Hive
> External Table syntax https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-CreateTable

> **Note:**
>
> Oracle supports only primitive types.

The following table shows the mappings between Oracle and Hive types.

| Oracle Data Type | Hive Data Type |
|---|---|
| NUMBER | INT when the scale is 0 and the precision is less than 10. |
| | BIGNIT when the scale is 0 and precision is less than 19. |
| | DECIMAL when the scale is greater than 0 or the precision is greater than 19. |

| | |
|---|---|
| CLOB<br><br>NCLOB | STRING |
| BINARY_DOUBLE | DOUBLE |
| BINARY_FLOAT | FLOAT |
| BLOB | BINARY |
| CHAR<br><br>NCHAR | CHAR |
| VARCHAR2<br><br>NVARCHAR2 | VARCHAR |
| ROWID<br><br>UROWID | BINARY |
| DATE | TIMESTAMP |
| TIMESTAMP | TIMESTAMP |
| TIMESTAMPTZ<br><br>TIMESTAMPLTZ | Unsupported |
| RAW | BINARY |

The properties of external tables can be described as follows:

## 6.2.3.1 TBLPROPERTIES

| Property | Use |
|---|---|
| oracle.hcat.osh.columns.mapping | Comma separated list to specify mapping between Hive columns and Oracle table columns. All external tables using OracleStorageHandler must define this. |
| mapreduce.jdbc.url | Connection URL to connect to the database |
| mapreduce.jdbc.username | Connection user name to connect to the database |
| mapreduce.jdbc.password | Connection password to connect to the database |
| mapreduce.jdbc.input.table.name | Oracle table name |
| mapreduce.jdbc.input conditions | To be used for querying the database. Must be used for query pushdown. |
| mapreduce.jdbc.input.query | To be used for querying the database. Query should be used only when a subset of the columns is selected. |
| mapreduce.jdbc.input.orderby | ORDER BY clause to be specified for pushing ordering to the database. |

| Property | Use |
| --- | --- |
| oracle.hcat.osh.splitterKind | To be used to specify how OracleStorageHandler must create splits, so that they are a good match for the physical structure of the target table in Oracle Database. The splitter kind applicable could be `SINGLE_SPLITTER`, `PARTITION_SPLITTER`, `ROW_SPLITTER`, `BLOCK_SPLITTER`. |
| oracle.hcat.osh.rowsPerSplit | Used only when `ROW_SPLITTER` splitterKind is applied on the table. Represents Number of rows per split for `LIMIT_RANGE` splitter. Default is 1000 |
| oracle.hcat.osh.authentication | Authentication method used to connect to Oracle Database. Can be `SIMPLE` (default), `ORACLE_WALLET`, `KERBEROS` |
| sun.security.krb5.principal | Kerberos principal. Used only when `KERBEROS` authentication is applied. |
| oracle.hcat.osh.kerb.callback | Callback for Kerberos authentication. Used only when Kerberos authentication is applied. |
| oracle.hcat.osh.maxSplits | Maximum number of splits for any splitter kind |
| oracle.hcat.osh.useChunkSplitter | Use chunk based ROW_SPLITTER and BLOCK_SPLITTER that use DBMS_PARALLEL_EXECUTE package to divide table into chunks that will map to hadoop splits.The default value is set to 'true'. |
| oracle.hcat.osh.chunkSQL | Used by `CUSTOM_SPLITTER`to create splits. The SQL string should be a SELECT statement that returns range of each chunk must have two columns: `start_id` and `end_id` The columns must be of `ROWID` type. |
| oracle.hcat.osh.useOracleParallelism | When configured, parallel queries will be executed while fetching rows from Oracle. Default value: `'false'` |
| oracle.hcat.osh.fetchSize | JDBC fetchsize for generated select queries used to fetch rows. Default value: 10 (set by Oracle JDBC Driver) |

> **Note:**
>
> In addition to the above, any JDBC connection properties (`oracle.jdbc.*` and `oracle.net.*`) can be specified as `TBLPROPERTIES`. They will be used while establishing connection to Oracle Database using JDBC driver.

> **Note:**
>
> Oracle DataSource for Apache Hadoop (OD4H) works with Oracle View and Oracle Tables.

## 6.2.3.2 SERDE PROPERTIES

| Property | Use |
|---|---|
| oracle.hcat.osh.columns.mapping | All external tables using OracleStorageHandler must define this. Its a comma separated list to specify mapping between hive columns (specified in create table) and oracle table columns. `WITHSERDEPROPERTIES` also enables the external table definition to refer only to select columns in the actual Oracle table. In other words, not all columns from the Oracle table need to be part of the Hive external table. The ordering of oracle columns in the mapping is the same as ordering of hive columns specified in create table. |

# 6.2.4 List of jars in the OD4H package

Oracle DataSource for Apache Hadoop (OD4H) contains the following list of jars.

OD4H consists of the following list of jars.

**Table 6-1   List of jars in OD4H**

| Name of JAR | Use |
|---|---|
| osh.jar | Contains OracleStorageHandler Implementation |
| ojdbc8.jar | An OD4H specific JDBC driver (which is optimized with internal calls), used by Spark or Hadoop tasks to connect to the database. |
| ucp.jar | For creating connection pools in OracleStorageHandler |
| oraclepki103.jar, osdt_core.jar, osdt_cert.jar, osdt_jce.jar | For Oracle Wallet authentication |
| orai18n.jar | Oracle Globalization Support |
| xdb.jar | Oracle XDB jar |

# 6.3 How does OD4H work?

Oracle DataSource for Apache Hadoop (OD4H) does not require creating a new table. You can start working with OD4H using the following steps:

1. Create a new Oracle table, or, reuse an existing table.

2. Create the Hive DDL for creating the external table referencing the Oracle Table.

3. Issue HiveSQL, SparkSQL, or other Spark/Hadoop queries and API calls.

The following sections show how to create a new Oracle Database Table, and a Hive DDL:

- Create a New Oracle Database Table

- Hive DDL

- [Creating External Table in Hive](#)

# 6.3.1 Create a new Oracle Database Table or Reuse an Existing Table

Here is an illustration of a partitioned Oracle table that we will use to demo how partition pruning works:

1. 
```
CREATE TABLE EmployeeData ( Emp_ID NUMBER,
     First_Name VARCHAR2(20),
     Last_Name VARCHAR2(20),
     Job_Title VARCHAR2(40),
     Salary NUMBER)
PARTITION BY RANGE (Salary)
 ( PARTITION salary_1 VALUES LESS THAN (60000)
     TABLESPACE tsa
 , PARTITION salary_2 VALUES LESS THAN (70000)
     TABLESPACE tsb
 , PARTITION salary_3 VALUES LESS THAN (80000)
     TABLESPACE tsc
 , PARTITION salary_4 VALUES LESS THAN (90000)
     TABLESPACE tsd
 , PARTITION salary_5 VALUES LESS THAN (100000)
     TABLESPACE tse
 );
```

> **Note:**
>
> You can use this syntax for table creation, in the following examples listed in this Book.

2. Issue queries from Hive, Spark, or any other Hadoop models (including joins with local Hive Tables.)

# 6.3.2 Hive DDL

In this example, we will associate two Hive external tables to the same Oracle table, using two different split patterns:

- `SIMPLE_SPLITTER`

- `PARTITION_SPLITTER`

> **Note:**
>
> It is possible that the external table has fewer columns than the base Oracle table.
> Since columns can have different names, use `TBLPROPERTY` for mapping with the base table.

In the following examples, we are using the following variables:

```
connection_string = jdbc:oracle:thin:@localhost:1521/<servicename>
```

```
oracle_user=od4h
```

```
oracle_pwd=od4h
```

The following command creates a Hive external table with the default split pattern, that is `SIMPLE_SPLITTER`.

```
CREATE EXTERNAL TABLE EmployeeDataSimple (
 Emp_ID int,
 First_Name string,
 Last_Name string,
 Job_Title string,
 Salary int
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
      'oracle.hcat.osh.columns.mapping' =
'Emp_ID,First_Name,Last_Name,Job_Title,Salary')
TBLPROPERTIES (
 'mapreduce.jdbc.url' = '${hiveconf:jdbc:oracle:thin:@localhost:1521/
<servicename>}',
 'mapreduce.jdbc.username' = '${hiveconf:od4h}',
 'mapreduce.jdbc.password' = '${hiveconf:od4h}',
 'mapreduce.jdbc.input.table.name' = 'EmployeeData'
);
```

The following example creates a Hive external table using `PARTITION_SPLITTER`.

```
DROP TABLE EmployeeDataPartitioned;
CREATE EXTERNAL TABLE EmployeeDataPartitioned (
 Emp_ID int,
 First_Name string,
 Last_Name string,
 Job_Title string,
 Salary int
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
      'oracle.hcat.osh.columns.mapping' =
'Emp_ID,First_Name,Last_Name,Job_Title,Salary')
TBLPROPERTIES (
 'mapreduce.jdbc.url' = '${hiveconf:jdbc:oracle:thin:@localhost:1521/
<servicename>}',
 'mapreduce.jdbc.username' = '${hiveconf:od4h}',
 'mapreduce.jdbc.password' = '${hiveconf:od4h}',
 'mapreduce.jdbc.input.table.name' = 'EmployeeData',
 'oracle.hcat.osh.splitterKind' = 'PARTITIONED_TABLE'
);
```

## 6.3.3 Create External Tables in Hive

You can create an external table in Hive in the following way:

```
DROP TABLE employees;

CREATE EXTERNAL TABLE employees (
  EMPLOYEE_ID INT,
  FIRST_NAME   STRING,
  LAST_NAME    STRING,
  SALARY       DOUBLE,
```

```
 HIRE_DATE    TIMESTAMP,
 JOB_ID       STRING
 )

 STORED BY 'oracle.hcat.osh.OracleStorageHandler'

WITH SERDEPROPERTIES (
 'oracle.hcat.osh.columns.mapping' =
'employee_id,first_name,last_name,salary,hire_date,job_id')

 TBLPROPERTIES (
    'mapreduce.jdbc.url' = 'jdbc:oracle:thin:@localhost:1521:orcl',
    'mapreduce.jdbc.username' = 'hr',
    'mapreduce.jdbc.password' = 'hr',
    'mapreduce.jdbc.input.table.name' = 'EMPLOYEES'
);
```

> **Note:**
>
> For using OD4H, ensure that the `ucp.jar`, the `ojdbc8.jar` file, and the `osh.jar` file are present in the Hive `CLASSPATH` environment variable. This is pre-configured on Oracle Big Data Appliance.
> To learn more about the `CLASSPATH` environment variable and other Hive configuration properties, refer to the following sources:
>
> https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Cli
>
> https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties
>
> For Cloudera distribution, refer to the following page that explains the steps for Cloudera Manager:
>
> https://www.cloudera.com/documentation/enterprise/5-14-x/topics/cm_mc_hive_udf.html
>
> For other distributions, refer to the respective documentation on adding additional JAR files to Hive or HiveServer2 environment.

# 6.4 Features of OD4H

The following topics discuss features of OD4H.

- Performance and Scalability Features
- Security Features
- Using Hive SQL with OD4H
- Using Spark SQL with OD4H

## 6.4.1 Performance And Scalability Features

Following sections discuss the performance and scalability features of OD4H:

- Splitters
- Predicate Pushdown

- Projection Pushdown

- Partition Pruning

- Smart Connection Management

HCatalog stores table metadata from Hive DDL. HiveSQl, Spark SQL and others, then use this metadata while submitting queries.

The Oracle table is divided into granules determined by the `splitterKind` property. These granules are then read into a split by `OracleStorageHandler`, by submitting generated queries.

`OracleStorageHandler` will not have to test all possible query types if the query plan determines which splits need to be scanned.

**Figure 6-2    OD4H in a Nutshell**



## 6.4.1.1 Splitters

While executing a query on a Hive external table through OD4H, the underlying Oracle table is dynamically divided into granules, which correspond to splits on the Hadoop side. Each split is processed by a single map task. With the help of the `ORACLE_SPLITTER_KIND` property, you can specify how the splits are created. This ensures that the splits are a good match for the physical structure of the target table in Oracle Database.

The different kinds of splitters available are:

**SINGLE_SPLITTER**

Creates one split for the table. Use `SINGLE_SPLITTER` where a single task is sufficient to process the query against the entire table.

**ROW_SPLITTER**

Limits the number of rows per Split. The default number of rows is 1000. You can specify number of rows by setting the `oracle.hcat.osh.rowsPerSplit` property. The default value of `oracle.hcat.osh.maxSplits` is 1 when `ROW_SPLITTER` is used. You can increase this value to enable parallel reads.

Based on the values provided in the `rowsPerSplit` property, OD4H will divide tables into splits. If the number of splits obtained is higher than the `maxSplits`, then `maxSplits` property will be used. The rows per split will be divided accordingly.

> **Note:**
>
> `oracle.hcat.osh.rowsPerSplit`is used only by `ROW_SPLITTER` and not any other splitter kind.

**BLOCK_SPLITTER**

Creates splits based on underlying storage of data blocks. With Block Splitter, you can specify the maximum number of splits to be generated. The default value of `oracle.hcat.osh.maxSplits` is 1, when `BLOCK_SPLITTER` is used. You can increase this value to enable parallel reads. `BLOCK_SPLITTER` requires `SELECT` privilege on the `SYS.DBA.EXTENTS` table, granted to the schema containing the Oracle target table. In the event that this permission does not exist, OD4H will use `SINGLE_SPLITTER`.

> **Note:**
>
> The actual number of splits under `BLOCK_SPLITTER` may be lesser than the value specified in the `oracle.hcat.osh.maxSplits` property.
> Do not use BLOCK_SPLITTER on partitioned tables or Index Organized tables.

> **Note:**
>
> For `ROW_SPLITTER` and `BLOCK_SPLITTER` types, use `oracle.hcat.osh.useChunkSplitter` to specify splitting mechanism. The default property value is `true`. This enables creating chunks corresponding to splits using the `DBMS_PARALLEL_EXECUTE` package. When the property value is `false`, custom SQL is generated for splitting. Since `DBMS_PARALLEL_EXECUTE` can only be used for tables and not views, if `mapreduce.jdbc.input.table.name` points to a view and not a table, then `oracle.hcat.osh.useChunkSplitter` should be set to `false`.

**PARTITION_SPLITTER**

Creates one split per partition. `PARTITION_SPLITTER` is used by default when the table is partitioned. You can override this setting by specifying

ROW_SPLITTER in table properties. With PARTITION_SPLITTER, the default value of oracle.hcat.osh.maxSplits table property is 64.

Following is an illustration of ROW_SPLITTER:

```
DROP TABLE employees;

CREATE EXTERNAL TABLE employees (
  EMPLOYEE_ID INT,
  FIRST_NAME   STRING,
  LAST_NAME    STRING,
  SALARY       DOUBLE,
  HIRE_DATE    TIMESTAMP,
  JOB_ID       STRING
 )
 STORED BY 'oracle.hcat.osh.OracleStorageHandler'

WITH SERDEPROPERTIES (
 'oracle.hcat.osh.columns.mapping' =
'employee_id,first_name,last_name,salary,hire_date,job_id')

TBLPROPERTIES (
'mapreduce.jdbc.url' = 'jdbc:oracle:thin:@localhost:1521:orcl',
'mapreduce.jdbc.username' = 'hr',
'mapreduce.jdbc.password' = 'hr',
'mapreduce.jdbc.input.table.name' = 'EMPLOYEES',
'oracle.hcat.osh.splitterKind' = 'ROW_SPLITTER',
'oracle.hcat.osh.rowsPerSplit' = '1500'
);
```

**CUSTOM_SPLITTER**

Use CUSTOM_SPLITTER If you want to provide a custom split generation mechanism. You can do this using CUSTOM_SPLITTER through oracle.hcat.osh.splitterKind property and a SELECT statement that emits ROWIDs corresponding to start and end of each split in oracle.hcat.osh.chunkSQL.

## 6.4.1.2 Choosing a Splitter

SINGLE_SPLITTER is used by default if no splitter is specified in the table properties for Hive external table, and the target Oracle table is not partitioned.

For an unpartitioned table, the default value of oracle.hcat.osh.maxSplits will be 1. For partitioned table, the default value of the same will be 64, and the default splitter will be PARTITION_SPLITTER. The default for maxSplits is set to limit the number of connections to the Oracle server. To increase this limit, you must increase the value of oracle.hcat.osh.maxSplits explicitly in hive table properties.

Use the following guidelines while choosing a splitter kind for a hive external table:

| Splitter Kind | Use |
| --- | --- |
| SINGLE_SPLITTER | When no parallelism is required. |
| PARTITION_SPLITTER | Used by default when target table is partitioned |

| Splitter Kind | Use |
|---|---|
| BLOCK_SPLITTER | When Oracle user has SELECT privilege on SYS.DBA_EXTENTS, and target table is not partitioned. |
| ROW_SPLITTER | When Oracle user does not have SELECT privilege on SYS.DBA_EXTENTS. |
| CUSTOM_SPLITTER | For fine grain control over generated splits. |

## 6.4.1.3 Predicate Pushdown

Predicate Pushdown is an optimization technique, in which you push predicates (WHERE condition) down to be evaluated by Oracle Database at the time of querying. This minimizes the amount of data fetched from Oracle Database to Hive, while performing a query.

Set the configuration property hive.optimize.ppd to either true or false for enabling Predicate Pushdown. The default value on hive-1.1.0 is set to true. Hence, Predicate Pushdown is always performed, unless you want to disable it.

> **Note:**
>
> OD4H does not push down all possible predicates. It considers only the part of the execution plan pertaining to Oracle table declared as external table. OD4H also rewrites sub-queries for the Oracle SQL engine and each split task. At present conditions involving operators >,=,< and != in a single condition over a column (e.g. key > 10) or a combination of multiple conditions separated by AND (e.g. key > 10 AND key < 20 AND key !=17) are pushed down.

Another option to reduce the amount of data fetched from the Oracle Database is to specify a condition at the time of table creation, using TBLPROPERTY mapreduce.jdbc.input.conditions. For instance:

```
mapreduce.jdbc.input.conditions = 'key > 10 OR key = 0'.
```

This will restrict the rows fetched from Oracle Database whenever any query is performed based on the condition specified. The external table that gets created, is analogous to a view on Oracle Database. This approach is only useful when you want to push down complex predicates that cannot be analyzed and automatically pushed down by OD4H.

> **✎ Note:**
>
> Due to incompatibilities between date and timestamp representation in Hive and Oracle, these columns are not pushed down by default in a query. You can enable this with certain limitations by setting the tableproperty `oracle.hcat.datetime.pushdown` to `true`. When set to `true`, the date representation in the query should be in the form `YYYY-MM-DD` and timestamp should be in the form "`YYYY-MM-DD HH:MM:SS`" with no decimal places. No other date or timestamp representation is supported when `oracle.hcat.datetime.pushdown` is set to `true`.

## 6.4.1.4 Projection Pushdown

Projection Pushdown is an optimization technique that fetches only the required columns from Oracle Database when a query is performed. If you want to fetch all columns during a query (not recommended), you can disable it by setting the `hive.io.file.read.all.columns` connection property to `true`. On Hive–1.1.0, this property is `false` by default.

## 6.4.1.5 Partition Pruning

If you refer to Employee Data Partition table, the partitions irrelevant to the query are removed from the partition list. This is done by executing an explain plan on the query to obtain the list of partitions and sub-partitions that are relevant to the query.

Table level partition pruning uses table level predicate pushdown, on the other hand partition pruning at the query level uses query level predicate pushdown.

Partition pruning is active when a `SELECT` query is run, in which the `WHERE` clause uses the partitioning key. Following is an example of partition pruning:

To query the partition, where salary is in the above range and prune other partitions, perform the following:

Hive External Table:

```
CREATE EXTERNAL TABLE EmployeeDataPartitioned (
 Emp_ID int,
 First_Name string,
 Last_Name string,
 Job_Title string,
 Salary int
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
     'oracle.hcat.osh.columns.mapping' =
'Emp_ID,First_Name,Last_Name,Job_Title,Salary')
TBLPROPERTIES (
 'mapreduce.jdbc.url' = '${hiveconf:connection_string}',
 'mapreduce.jdbc.username' = '${hiveconf:oracle_user}',
 'mapreduce.jdbc.password' = '${hiveconf:oracle_pwd}',
 'mapreduce.jdbc.input.table.name' = 'EmployeeData',
 'oracle.hcat.osh.oosKind' = 'PARTITIONED_TABLE'
);
```

The following `SELECT` statement shows how to query the partition, where salary is between 72000 to 78000, and prunes other partitions:

```
select * from EmployeeDataPartitioned where salary > 72000 and salary < 78000;
```

## 6.4.2 Smart Connection Management

**Connection Caching**

Each map task runs in its own JVM. Each JVM in turn caches a single connection to the Oracle database that you can reuse within the same query. The Mapper checks the cache before establishing a new connection and caching is not done once the query has completed executing.

**Oracle RAC Awareness**

JDBC and UCP are aware of various Oracle RAC instances. This can be used to split queries submitted to JDBC. The StorageHandler will depend on listener for load balancing.

**Handling Logon Storms**

Hadoop allows you to limit the number of mappers attempting to connect to the Database. Hadoop allows you to limit the number of mappers attempting to connect to the Database using `oracle.hcat.osh.maxSplits`. This parameter controls the degree of concurrency. However, subsequent tasks of the same query are guaranteed to query their table granule as per the System Commit Number (SCN) of the query. This ensures consistency of the result sets.

**Database Resident Connection Pooling (DRCP)**

It is recommended to configure DRCP for OD4H, and limit the maximum number of concurrent connections to the Oracle Database from OD4H.

Configuring Database Resident Connection Pooling
To configure DRCP, use the following steps:

1. Login as `SYSDBA`.

2. Start the default pool, `SYS_DEFAULT_CONNECTION_POOL` using `DBMS_CONNECTION_POOL.START_POOL` with the default settings.

   You can use `DBMS_CONNECTION_POOL.MINSIZE` and `DBMS_CONNECTION_POOL.MAXSIZE` with the default settings.

> ✏️ **See Also:**
>
> For more information on configuring DRCP see the Oracle Database Administration Guide

## 6.4.3 Security Features

Following are the security features of OD4H:

## 6.4.3.1 Improved Authentication

OD4H uses Oracle JDBC driver for connecting to Oracle Database. It provides all authentication methods supported by Oracle JDBC. OD4H supports authentication through use of basic authentication (user name and password), Oracle Wallet, and Kerberos. You can specify the authentication to be used for a table created in Hive, through the `oracle.hcat.osh.authentication` table property. This is useful only for strong authentication.

- Kerberos
- Oracle Wallet
- Basic Authentication

> **Note:**
>
> Oracle recommends using strong authentication such as Kerberos.

The various authentication processes are described with examples as follows:

1. Kerberos
   Uses Kerberos credentials of the Hadoop engine process. This `principal` should have access to the table.

> **See Also:**
>
> *Oracle Database JDBC Developer's Guide* for information on configuring database for Kerberos and details of client parameters

You can enable Kerberos configuration on Hive, by adding to `hive-env.sh` the following:

```
export HADOOP_OPTS="$HADOOP_OPTS -Djava.security.krb5.conf=<path to kerberos configuration>
```

To enable child JVMs to use Kerberos configuration, edit the `mapred-site.xml` to include the following property on all nodes of the cluster:

```
<property><name>mapred.child.java.opts</name>  <value>-Djava.security.krb5.conf=<path to kerberos configuration>></value></property>
```

Enable these configurations on Oracle Big Data Appliance using Cloudera manager..

Following is an illustration of Kerberos authentication:

```
CREATE EXTERNAL TABLE kerb_example (
id DECIMAL,
```

```
name STRING,
salary DECIMAL
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
          'oracle.hcat.osh.columns.mapping' = 'id,name,salary')
TBLPROPERTIES (
'mapreduce.jdbc.url' =
'jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=adc*******.xxxxxx.com)(PORT=5521))(CONNECT_DATA=
(SERVICE_NAME=project_name.xxx.rdbms.xxxx.com)))',
'mapreduce.jdbc.input.table.name' = 'kerb_example',
'mapreduce.jdbc.username' = 'CLIENT@xxxxxx.COM',
'oracle.hcat.osh.authentication' = 'KERBEROS',
'oracle.net.kerberos5_cc_name' = '/tmp/krb5cc_xxxxx',
'java.security.krb5.conf' = '/home/user/kerberos/krb5.conf',
'oracle.hcat.osh.kerb.callback' = 'KrbCallbackHandler',
'sun.security.krb5.principal' = 'CLIENT@xxxxx.COM'
);
```

The path specified in `oracle.security.krb5.conf` should be accessible to all nodes of the cluster. These paths should also match with the path of the corresponding properties in Oracle Database `sqlnet.ora`. The `keytab` path provided in `sqlnet.ora` should also be accessible from all nodes of the cluster.

If `sun.security.krb5.principal` is not specified, OD4H attempts to authenticate using default principal in Credential Cache specified by the `oracle.net.kerberos5_cc_name` property.

> **Note:**
>
> The `callback` is called only if the `principal` cannot be authenticated using a ticket obtained from the credential cache specified in `oracle.net.kerberos5_cc_nameproperty`.

A simple callback handler class is described as follows (The callback class must be available to the Hive classpath):

```
class KrbCallbackHandler
        implements CallbackHandler{

@Override
public void handle(Callback[] callbacks) throws IOException,
        UnsupportedCallbackException{
for (int i = 0; i < callbacks.length; i++){
    if (callbacks[i] instanceof PasswordCallback){
    PasswordCallback pc = (PasswordCallback)callbacks[i];
    System.out.println("set password to 'welcome'");
    pc.setPassword((new String("welcome")).toCharArray());
} else if (callbacks[i] instanceof NameCallback) {
    ((NameCallback)callbacks[i]).setName("client@xxxxx.COM");
}else{
    throw new UnsupportedCallbackException(callbacks[i],
            "Unrecognized Callback");
        }
```

```
        }
    }
```

2. Oracle Wallet

   The wallet should be available in the OS environment of each engine process. Following is an illustration of how to add Wallet authentication:

```
CREATE EXTERNAL TABLE wallet_example (
    id DECIMAL,
    salary DECIMAL
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
    name STRING,
        'oracle.hcat.osh.columns.mapping' = 'id,name,salary')
TBLPROPERTIES (
'mapreduce.jdbc.url' = 'jdbc:oracle:thin:/@inst1',
'mapreduce.jdbc.input.table.name' = 'wallet_example',
'oracle.hcat.osh.authentication' = 'ORACLE_WALLET',
'oracle.net.tns_admin' = '/scratch/user/view_storage/user_project6/
work',
'oracle.net.wallet_location' = '/scratch/user/view_storage/
user_project6/work'
);
```

   > **Note:**
   >
   > The paths specified in `oracle.net.tns_admin` and `oracle.net.wallet_location` should be accessible from all nodes of the cluster.

   > **See Also:**
   >
   > Managing the Secure External Password Store for Password Credentials

3. Basic Authentication (for demonstration purposes only)

   This is stored in HCatalog `TBLPROPERTIES` or supplied on HiveQL `SELECT` statement.

   When Basic Authentication is used, then the user name and password for Oracle Schema are specified in Hive external Table properties.

   > **Note:**
   >
   > Oracle does not recommend this authentication process in the production environment because the password is stored unmasked in HCatalog.

## 6.5 Use HiveQL with OD4H

HiveQL is a SQL like language provided by Hive. It can be used to query hive external tables created using OD4H.

To track the status of a running query on Oracle Big Data Appliance, you can run the Resource Manager web interface in your browser in the following way:

```
http://<domain>:8088/<cluster_name>
```

You can also see the logs of a query in Cloudera Manager, which also indicates the actual query sent to Oracle Database corresponding to your query on HiveQL. Hive and OD4H use the Simple Logging Facade for Java (SLF4J) framework for logging. You can use the logging configuration techniques of Hive for controlling the logging level of classes related to OD4H.

## 6.6 Use Spark SQL with OD4H

Spark SQL enables you to execute relational queries expressed in SQL and HiveSQL using Spark. Spark SQL mixex SQL queries with programmatic data manipulations supported by RDDs (Resilient Distributed Datasets) in Java, Python and Scala, with a single application. You can also use it to query external tables created using OD4H.

Before running the queries, perform the following steps to configure Spark SQL on Oracle Big Data Appliance:

1. Add `ojdbc7.jar` and `osh.jar` to CLASSPATH in `/usr/lib/spark/bin/compute-classpath.sh`

```
CLASSPATH="$CLASSPATH:/opt/oracle/od4h/lib/osh.jar"
CLASSPATH="$CLASSPATH:/opt/oracle/od4h/lib/ojdbc7.jar"
```

2. Edit `SPARK_HOME` in `/usr/lib/spark/conf/spark-env.sh`

```
export SPARK_HOME=/usr/lib/spark:/etc/hive/conf
```

3. You will need to specify additional environment variables in `/usr/lib/spark/conf/spark-env.sh`.

   The Hive related variables that you need to substitute for running the code example, are marked in bold. The file already contains Hadoop related environment variables.

```
export DEFAULT_HADOOP=/usr/lib/hadoop
export DEFAULT_HIVE=/usr/lib/hive
export DEFAULT_HADOOP_CONF=/etc/hadoop/conf
export DEFAULT_HIVE_CONF=/etc/hive/conf
export HADOOP_HOME=${HADOOP_HOME:-$DEFAULT_HADOOP}
export HADOOP_HDFS_HOME=${HADOOP_HDFS_HOME:-${HADOOP_HOME}/../hadoop-hdfs}
export HADOOP_MAPRED_HOME=${HADOOP_MAPRED_HOME:-${HADOOP_HOME}/../hadoop-mapreduce}
export HADOOP_YARN_HOME=${HADOOP_YARN_HOME:-${HADOOP_HOME}/../hadoop-yarn}
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-$DEFAULT_HADOOP_CONF}
export HIVE_CONF_DIR=${HIVE_CONF_DIR:-$DEFAULT_HIVE_CONF}

CLASSPATH="$CLASSPATH:$HIVE_CONF_DIR"
CLASSPATH="$CLASSPATH:$HADOOP_CONF_DIR"
```

```
if [ "x" != "x$YARN_CONF_DIR" ]; then
  CLASSPATH="$CLASSPATH:$YARN_CONF_DIR"
fi

# Let's make sure that all needed hadoop libs are added properly
CLASSPATH="$CLASSPATH:$HADOOP_HOME/client/*"
CLASSPATH="$CLASSPATH:$HIVE_HOME/lib/*"
CLASSPATH="$CLASSPATH:$($HADOOP_HOME/bin/hadoop classpath)"
```

Once configured, you can run some sample queries on spark SQL using scripts included in the `/shell/*QuerySpark.sh` demo file. By default, Spark prints queries on the console. To modify this behavior you can edit the spark logging configuration file `/usr/lib/spark/conf/log4j.properties`.

The log printed by OracleRecordReader shows the actual query sent to Oracle Database, as follows:

15/03/18 10:36:08 `INFO` OracleRecordReader: Reading records from Oracle Table using Query: `SELECT FIRST_NAME, LAST_NAME, EMP_ID FROM` EmployeeData

# 6.7 Writing Back to Oracle Database

In the typical use case for OD4H, you store the result sets of Hive or Spark SQL queries back to Oracle Database. OD4H implements OutputFormat to enable you to write back to an Oracle Database table from Hadoop.

After the data is inserted into an Oracle Database table, you can then use your favorite business intelligence tools for further data mining

The following query is from the OD4H demo code samples. It demonstrates writing back to an external table called EmployeeBonusReport.

**Example 6-1    Writing Hive or Spark Result Sets Back to Oracle Database**

```
INSERT INTO EmployeeBonusReport
           SELECT EmployeeDataSimple.First_Name,
EmployeeDataSimple.Last_Name,
                  EmployeeBonus.bonus
           FROM EmployeeDataSimple JOIN EmployeeBonus ON
                   (EmployeeDataSimple.Emp_ID=EmployeeBonus.Emp_ID)
           WHERE salary > 70000 and bonus > 7000"
```

# Glossary

### Apache Flume

A distributed service for collecting and aggregating data from almost any source into a data store such as HDFS or HBase.

See also Apache HBase; HDFS.

### Apache HBase

An open-source, column-oriented database that provides random, read/write access to large amounts of sparse data stored in a CDH cluster. It provides fast lookup of values by key and can perform thousands of insert, update, and delete operations per second.

### Apache Hive

An open-source data warehouse in CDH that supports data summarization, ad hoc querying, and data analysis of data stored in HDFS. It uses a SQL-like language called HiveQL. An interpreter generates MapReduce code from the HiveQL queries.

By using Hive, you can avoid writing MapReduce programs in Java.

See also Hive Thrift; MapReduce.

### Apache Sentry

Integrates with the Hive and Impala SQL-query engines to provide fine-grained authorization to data and metadata stored in Hadoop.

### Apache Solr

Provides an enterprise search platform that includes full-text search, faceted search, geospatial search, and hit highlighting.

### Apache Spark

A fast engine for processing large-scale data. It supports Java, Scala, and Python applications. Because it provides primitives for in-memory cluster computing, it is

particularly suited to machine-learning algorithms. It promises performance up to 100 times faster than MapReduce.

### Apache Sqoop

A command-line tool that imports and exports data between HDFS or Hive and structured databases. The name Sqoop comes from "SQL to Hadoop." Oracle R Advanced Analytics for Hadoop uses the Sqoop executable to move data between HDFS and Oracle Database.

### Apache YARN

An updated version of MapReduce, also called MapReduce 2. The acronym stands for Yet Another Resource Negotiator.

### ASR

Oracle Auto Service Request, a software tool that monitors the health of the hardware and automatically generates a service request if it detects a problem.

See also OASM.

### Balancer

A service that ensures that all nodes in the cluster store about the same amount of data, within a set range. Data is balanced over the nodes in the cluster, not over the disks in a node.

### CDH

Cloudera's Distribution including Apache Hadoop, the version of Apache Hadoop and related components installed on Oracle Big Data Appliance.

### Cloudera Hue

Hadoop User Experience, a web user interface in CDH that includes several applications, including a file browser for HDFS, a job browser, an account management tool, a MapReduce job designer, and Hive wizards. Cloudera Manager runs on Hue.

See also HDFS; Apache Hive.

### Cloudera Impala

A massively parallel processing query engine that delivers better performance for SQL queries against data in HDFS and HBase, without moving or transforming the data.

**Cloudera Manager**

Cloudera Manager enables you to monitor, diagnose, and manage CDH services in a cluster.

The Cloudera Manager agents on Oracle Big Data Appliance also provide information to Oracle Enterprise Manager, which you can use to monitor both software and hardware.

**Cloudera Navigator**

Verifies access privileges and audits access to data stored in Hadoop, including Hive metadata and HDFS data accessed through HDFS, Hive, or HBase.

**Cloudera Search**

Provides search and navigation tools for data stored in Hadoop. Based on Apache Solr.

**Cloudera's Distribution including Apache Hadoop (CDH)**

See CDH.

**cluster**

A group of servers on a network that are configured to work together. A server is either a master node or a worker node.

All servers in an Oracle Big Data Appliance rack form a cluster. Servers 1, 2, and 3 are master nodes. Servers 4 to 18 are worker nodes.

See Hadoop.

**DataNode**

A server in a CDH cluster that stores data in HDFS. A DataNode performs file system operations assigned by the NameNode.

See also HDFS; NameNode.

**Flume**

See Apache Flume.

**Hadoop**

A batch processing infrastructure that stores files and distributes work across a group of servers. Oracle Big Data Appliance uses Cloudera's Distribution including Apache Hadoop (CDH).

**Hadoop Distributed File System (HDFS)**

See HDFS.

**Hadoop User Experience (Hue)**

See Cloudera Hue.

**HBase**

See Apache HBase.

**HDFS**

Hadoop Distributed File System, an open-source file system designed to store extremely large data files (megabytes to petabytes) with streaming data access patterns. HDFS splits these files into data blocks and distributes the blocks across a CDH cluster.

When a data set is larger than the storage capacity of a single computer, then it must be partitioned across several computers. A distributed file system can manage the storage of a data set across a network of computers.

See also cluster.

**Hive**

See Apache Hive.

**Hive Thrift**

A remote procedure call (RPC) interface for remote access to CDH for Hive queries.

See also CDH; Apache Hive.

**HiveQL**

A SQL-like query language used by Hive.

See also Apache Hive.

**HotSpot**

A Java Virtual Machine (JVM) that is maintained and distributed by Oracle. It automatically optimizes code that executes frequently, leading to high performance. HotSpot is the standard JVM for the other components of the Oracle Big Data Appliance stack.

**Hue**

See Cloudera Hue.

**Impala**

See Cloudera Impala.

**Java HotSpot Virtual Machine**

See HotSpot.

**JobTracker**

A service that assigns tasks to specific nodes in the CDH cluster, preferably those nodes storing the data. MRv1 only.

See also Hadoop; MapReduce.

**Kerberos**

A network authentication protocol that helps prevent malicious impersonation. It was developed at the Massachusetts Institute of Technology (MIT).

**Mahout**

Apache Mahout is a machine learning library that includes core algorithms for clustering, classification, and batch-based collaborative filtering.

**MapReduce**

A parallel programming model for processing data on a distributed system. Two versions of MapReduce are available, MapReduce 1 and YARN (MapReduce 2). The default version on Oracle Big Data Appliance 3.0 and later is YARN.

A MapReduce program contains these functions:

- Mappers: Process the records of the data set.

- Reducers: Merge the output from several mappers.

- Combiners: Optimizes the result sets from the mappers before sending them to the reducers (optional and not supported by all applications).

See also Apache YARN.

**MySQL Database**

A SQL-based relational database management system. Cloudera Manager, Oracle Data Integrator, Hive, and Oozie use MySQL Database as a metadata repository on Oracle Big Data Appliance.

**NameNode**

A service that maintains a directory of all files in HDFS and tracks where data is stored in the CDH cluster.

See also HDFS.

**Navigator**

See Cloudera Navigator.

**node**

A server in a CDH cluster.

See also cluster.

**NodeManager**

A service that runs on each node and executes the tasks assigned to it by the ResourceManager. YARN only.

See also ResourceManager; YARN.

**NoSQL Database**

See Oracle NoSQL Database.

**OASM**

Oracle Automated Service Manager, a service for monitoring the health of Oracle Sun hardware systems. Formerly named Sun Automatic Service Manager (SASM).

**Oozie**

An open-source workflow and coordination service for managing data processing jobs in CDH.

**Oracle Database Instant Client**

A small-footprint client that enables Oracle applications to run without a standard Oracle Database client.

**Oracle Linux**

Oracle Linux is Oracle's commercial version of the Linux operating system. Oracle Linux is free to download, use, and redistribute without a support contract.

**Oracle NoSQL Database**

A distributed key-value database that supports fast querying of the data, typically by key lookup.

**Oracle R Distribution**

An Oracle-supported distribution of the R open-source language and environment for statistical analysis and graphing.

**Oracle R Enterprise**

A component of the Oracle Advanced Analytics Option. It enables R users to run R commands and scripts for statistical and graphical analyses on data stored in an Oracle database.

**Pig**

An open-source platform for analyzing large data sets that consists of the following:

- Pig Latin scripting language
- Pig interpreter that converts Pig Latin scripts into MapReduce jobs

Pig runs as a client application.

See also MapReduce.

**ResourceManager**

A service that assigns tasks to specific nodes in the CDH cluster, preferably those nodes storing the data. YARN only.

See also Hadoop; YARN.

**Search**

See Cloudera Search.

**Sentry**

See Apache Sentry.

**Solr**

See Apache Solr.

**Spark**

See Apache Spark.

**Sqoop**

See Apache Sqoop.

**table**

In Hive, all files in a directory stored in HDFS.

See also HDFS.

**TaskTracker**

A service that runs on each node and executes the tasks assigned to it by the JobTracker service. MRv1 only.

See also JobTracker.

**Whirr**

Apache Whirr is a set of libraries for running cloud services.

**YARN**

See Apache YARN.

**ZooKeeper**

A MapReduce 1 centralized coordination service for CDH distributed processes that maintains configuration information and naming, and provides distributed synchronization and group services.

# Index

## A

Apache Sentry, *2-2*
application adapters, *1-10*
applications
    data pull, *5-1*
    data push, *5-2*
authentication, *4-1*
authorization, *2-2*
Automated Service Manager, *2-7*

## B

bdadiag utility, *3-34*
Berkeley DB, *1-6*
big data description, *1-1*
business intelligence, *1-3*, *1-5*, *1-11*

## C

CDH
    about, *1-3*
    diagnostics, *3-34*
    file system, *1-5*
    remote client access, *4-2*
    security, *4-1*
chunking files, *1-5*
client access
    HDFS cluster, *4-3*
    HDFS secured cluster, *4-4*
    Hive, *4-6*
client configuration, *4-2*
Cloudera Manager
    about, *3-3*
    accessing administrative tools, *3-5*
    connecting to, *3-3*
    effect of hardware failure on, *3-25*
    software dependencies, *3-25*
    starting, *3-3*
    UI overview, *3-3*
Cloudera's Distribution including Apache
    Hadoop, *1-5*
clusters, definition, *1-3*

## D

data replication, *1-5*
DataNode, *3-23*
diagnostics, collecting, *3-34*
dnsmasq service, *5-4*
duplicating data, *1-5*

## E

emcli utility, *3-2*
encryption, *2-3*, *2-4*
engineered systems, *1-3*
Exadata Database Machine, *1-3*
Exadata InfiniBand connections, *5-3*
Exalytics In-Memory Machine, *1-3*
external tables, *1-9*

## F

failover
    JobTracker, *3-18*
    NameNode, *3-17*
files, recovering HDFS, *4-10*
first NameNode, *3-24*
Flume, *3-9*
ftp.oracle.com, *3-34*

## G

groups, *2-1*, *4-8*

## H

Hadoop Distributed File System, *1-4*
hadoop group, *4-8*
Hadoop version, *1-3*
HBase, *3-9*
HDFS
    about, *1-4*, *1-5*
HDFS Transparent Encryption, *2-3*
help from Oracle Support, *3-34*
Hive, *2-2*
    about, *1-6*