

Cloudera Introduction



Important Notice

© 2010-2016 Cloudera, Inc. All rights reserved.

Cloudera, the Cloudera logo, Cloudera Impala, and any other product or service names or slogans contained in this document are trademarks of Cloudera and its suppliers or licensors, and may not be copied, imitated or used, in whole or in part, without the prior written permission of Cloudera or the applicable trademark holder.

Hadoop and the Hadoop elephant logo are trademarks of the Apache Software Foundation. All other trademarks, registered trademarks, product names and company names or logos mentioned in this document are the property of their respective owners. Reference to any products, services, processes or other information, by trade name, trademark, manufacturer, supplier or otherwise does not constitute or imply endorsement, sponsorship or recommendation thereof by us.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Cloudera.

Cloudera may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Cloudera, the furnishing of this document does not give you any license to these patents, trademarks copyrights, or other intellectual property. For information about patents covering Cloudera products, see <http://tiny.cloudera.com/patents>.

The information in this document is subject to change without notice. Cloudera shall not be liable for any damages resulting from technical errors or omissions which may be present in this document, or from use of this document.

Cloudera, Inc.

1001 Page Mill Road, Bldg 3

Palo Alto, CA 94304

info@cloudera.com

US: 1-888-789-1488

Intl: 1-650-362-0488

www.cloudera.com

Release Information

Version: Cloudera Enterprise 5.9.x

Date: December 1, 2016

Table of Contents

About Cloudera Introduction.....6

Documentation Overview.....	6
-----------------------------	---

CDH Overview.....9

Apache Impala (incubating) Overview.....	9
<i>Impala Benefits</i>	10
<i>How Impala Works with CDH</i>	10
<i>Primary Impala Features</i>	11
Cloudera Search Overview.....	11
<i>How Cloudera Search Works</i>	12
<i>Understanding Cloudera Search</i>	13
<i>Cloudera Search and Other Cloudera Components</i>	13
<i>Cloudera Search Architecture</i>	15
<i>Cloudera Search Tasks and Processes</i>	18
Apache Sentry Overview.....	20
Apache Spark Overview.....	20
File Formats and Compression.....	21
<i>Using Apache Parquet Data Files with CDH</i>	21
<i>Using Apache Avro Data Files with CDH</i>	29
<i>Data Compression</i>	32
External Documentation.....	34

Cloudera Manager 5 Overview.....36

Terminology.....	36
Architecture.....	39
State Management.....	40
Configuration Management.....	41
Process Management.....	44
Software Distribution Management.....	44
Host Management.....	45
Resource Management.....	45
User Management.....	47
Security Management.....	47
Cloudera Management Service.....	48
Overview of EMC DSSD D5 Integration.....	49

Cloudera Manager Admin Console.....	50
<i>Starting and Logging into the Admin Console.....</i>	51
<i>Cloudera Manager Admin Console Home Page.....</i>	52
<i>Displaying Cloudera Manager Documentation.....</i>	54
<i>Displaying the Cloudera Manager Server Version and Server Time.....</i>	55
<i>Automatic Logout.....</i>	55
EMC DSSD D5 Storage Appliance Integration for Hadoop DataNodes.....	56
<i>Overview of EMC DSSD D5 Integration.....</i>	56
<i>Installing CDH with DSSD DataNodes.....</i>	56
Cloudera Manager API.....	56
<i>Backing Up and Restoring the Cloudera Manager Configuration</i>	58
<i>Using the Cloudera Manager Java API for Cluster Automation.....</i>	59
Extending Cloudera Manager.....	61

Cloudera Navigator 2 Overview.....62

Cloudera Navigator Data Management Overview.....	63
<i>Cloudera Navigator Data Management UI.....</i>	63
<i>Cloudera Navigator Data Management API.....</i>	64
<i>Displaying Cloudera Navigator Data Management Documentation.....</i>	65
<i>Displaying the Cloudera Navigator Data Management Component Version.....</i>	65
Cloudera Navigator Data Encryption Overview.....	65
<i>Cloudera Navigator Encryption Architecture.....</i>	67
<i>Cloudera Navigator Encryption Integration with an EDH.....</i>	68
<i>Cloudera Navigator Key Trustee Server Overview.....</i>	68
<i>Cloudera Navigator Key HSM Overview.....</i>	69
<i>Cloudera Navigator Encrypt Overview.....</i>	70

Cloudera Navigator Optimizer.....74

Frequently Asked Questions About Cloudera Software.....75

Cloudera Express and Cloudera Enterprise Features.....	75
Cloudera Manager 5 Frequently Asked Questions.....	77
<i>General Questions.....</i>	77
Cloudera Navigator 2 Frequently Asked Questions.....	79
Impala Frequently Asked Questions.....	80
<i>Trying Impala.....</i>	80
<i>Impala System Requirements.....</i>	82
<i>Supported and Unsupported Functionality In Impala.....</i>	83
<i>How do I?.....</i>	84
<i>Impala Performance.....</i>	84
<i>Impala Use Cases.....</i>	87
<i>Questions about Impala And Hive.....</i>	87

<i>Impala Availability</i>	88
<i>Impala Internals</i>	89
<i>SQL</i>	91
<i>Partitioned Tables</i>	92
<i>HBase</i>	93
Cloudera Search Frequently Asked Questions.....	93
<i>General</i>	93
<i>Performance and Fail Over</i>	95
<i>Schema Management</i>	95
<i>Supportability</i>	96

Getting Support.....97

Cloudera Support.....	97
<i>Information Required for Logging a Support Case</i>	97
Community Support.....	97
Get Announcements about New Releases.....	98
Report Issues.....	98

About Cloudera Introduction

Cloudera provides a scalable, flexible, integrated platform that makes it easy to manage rapidly increasing volumes and varieties of data in your enterprise. Cloudera products and solutions enable you to deploy and manage Apache Hadoop and related projects, manipulate and analyze your data, and keep that data secure and protected.

Cloudera provides the following products and tools:

- [CDH](#)—The Cloudera distribution of Apache Hadoop and other related open-source projects, including Apache Impala (incubating) and Cloudera Search. CDH also provides security and integration with numerous hardware and software solutions.
- [Apache Impala \(incubating\)](#)—A massively parallel processing SQL engine for interactive analytics and business intelligence. Its highly optimized architecture makes it ideally suited for traditional BI-style queries with joins, aggregations, and subqueries. It can query Hadoop data files from a variety of sources, including those produced by MapReduce jobs or loaded into Hive tables. The YARN resource management component lets Impala coexist on clusters running batch workloads concurrently with Impala SQL queries. You can manage Impala alongside other Hadoop components through the Cloudera Manager user interface, and secure its data through the Sentry authorization framework.
- [Cloudera Search](#)—Provides near real-time access to data stored in or ingested into Hadoop and HBase. Search provides near real-time indexing, batch indexing, full-text exploration and navigated drill-down, as well as a simple, full-text interface that requires no SQL or programming skills. Fully integrated in the data-processing platform, Search uses the flexible, scalable, and robust storage system included with CDH. This eliminates the need to move large data sets across infrastructures to perform business tasks.
- [Cloudera Manager](#)—A sophisticated application used to deploy, manage, monitor, and diagnose issues with your CDH deployments. Cloudera Manager provides the Admin Console, a web-based user interface that makes administration of your enterprise data simple and straightforward. It also includes the Cloudera Manager API, which you can use to obtain cluster health information and metrics, as well as configure Cloudera Manager.
- [Cloudera Navigator](#)—An end-to-end data management and security tool for the CDH platform. Cloudera Navigator enables administrators, data managers, and analysts to explore the large amounts of data in Hadoop, and simplifies the storage and management of encryption keys. The robust auditing, data management, lineage management, lifecycle management, and encryption key management in Cloudera Navigator allow enterprises to adhere to stringent compliance and regulatory requirements.

This introductory guide provides a general overview of CDH, Cloudera Manager, and Cloudera Navigator. This guide also includes frequently asked questions about Cloudera products and describes how to get support, report issues, and receive information about updates and new releases.

Documentation Overview

The following guides are included in the Cloudera documentation set:

Guide	Description
Overview of Cloudera and the Cloudera Documentation Set	Cloudera provides a scalable, flexible, integrated platform that makes it easy to manage rapidly increasing volumes and varieties of data in your enterprise. Cloudera products and solutions enable you to deploy and manage Apache Hadoop and related projects, manipulate and analyze your data, and keep that data secure and protected.
Cloudera Release Notes	This guide contains release and download information for installers and administrators. It includes release notes as well as information about versions and downloads. The guide also provides a release matrix that shows which major and minor release version of a product is supported with which release version of Cloudera Manager, CDH and, if applicable, Cloudera Impala.

Guide	Description
Cloudera QuickStart	This set of guides describes ways to rapidly begin experimenting with Cloudera software. The first section describes how to download and use QuickStart virtual machines, which provide everything you need to try CDH, Cloudera Manager, Impala, and Cloudera Search. Subsequent sections show you how to create a new installation of Cloudera Manager 5, CDH5, and managed services on a cluster of four hosts and an unmanaged CDH pseudo cluster. Quick start installations are for demonstration and POC applications only and are not recommended for production use.
Cloudera Installation	This guide provides instructions for installing Cloudera software.
Cloudera Upgrade	This topic provides an overview of upgrade procedures for Cloudera Manager and CDH. The procedures described here use Cloudera Manager to perform some or all of the upgrade steps. You can also upgrade unmanaged CDH clusters (clusters that are not managed by Cloudera Manager). See Upgrading Unmanaged CDH Using the Command Line .
Cloudera Administration	This guide describes how to configure and administer a Cloudera deployment. Administrators manage resources, availability, and backup and recovery configurations. In addition, this guide shows how to implement high availability, and discusses integration.
Cloudera Data Management	This guide describes how to perform data management using Cloudera Navigator. Data management activities include auditing access to data residing in HDFS and Hive metastores, reviewing and updating metadata, and discovering the lineage of data objects.
Cloudera Operation	This guide shows how to monitor the health of a Cloudera deployment and diagnose issues. You can obtain metrics and usage information and view processing activities. This guide also describes how to examine logs and reports to troubleshoot issues with cluster configuration and operation as well as monitor compliance.
Cloudera Security	This guide is intended for system administrators who want to secure a cluster using data encryption, user authentication, and authorization techniques. It provides conceptual overviews and how-to information about setting up various Hadoop components for optimal security, including how to setup a gateway to restrict access.
Apache Impala (incubating) - Interactive SQL	This guide describes Impala, its features and benefits, and how it works with CDH. This topic introduces Impala concepts, describes how to plan your Impala deployment, and provides tutorials for first-time users as well as more advanced tutorials that describe scenarios and specialized features. You will also find a language reference, performance tuning, instructions for using the Impala shell, troubleshooting information, and frequently asked questions.
Cloudera Search Guide	This guide explains how to configure and use Cloudera Search. This includes topics such as extracting, transforming, and loading data, establishing high availability, and troubleshooting.
Spark Guide	This guide describes Apache Spark, a general framework for distributed computing that offers high performance for both batch and interactive processing. The guide provides tutorial Spark applications, how to develop and run Spark applications, and how to use Spark with other Hadoop components.

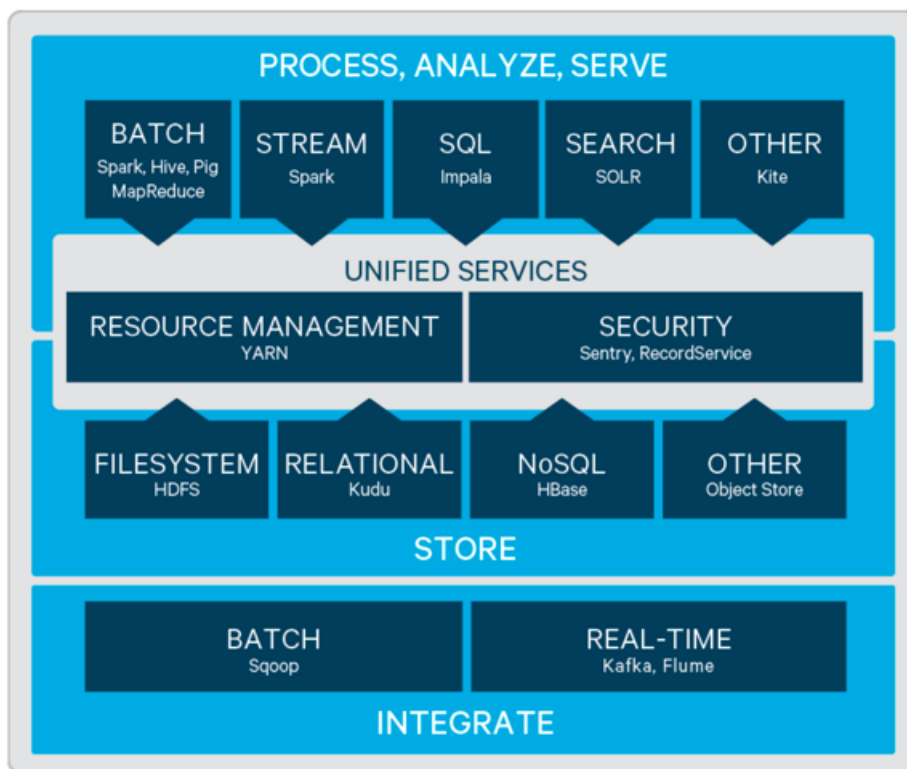
Guide	Description
Cloudera Glossary	This guide contains a glossary of terms for Cloudera components.

CDH Overview

CDH is the most complete, tested, and popular distribution of Apache Hadoop and related projects. CDH delivers the core elements of Hadoop – scalable storage and distributed computing – along with a Web-based user interface and vital enterprise capabilities. CDH is Apache-licensed open source and is the only Hadoop solution to offer unified batch processing, interactive SQL and interactive search, and role-based access controls.

CDH provides:

- **Flexibility**—Store any type of data and manipulate it with a variety of different computation frameworks including batch processing, interactive SQL, free text search, machine learning and statistical computation.
- **Integration**—Get up and running quickly on a complete Hadoop platform that works with a broad range of hardware and software solutions.
- **Security**—Process and control sensitive data.
- **Scalability**—Enable a broad range of applications and scale and extend them to suit your requirements.
- **High availability**—Perform mission-critical business tasks with confidence.
- **Compatibility**—Leverage your existing IT infrastructure and investment.



For information about CDH components, which is out of scope for Cloudera documentation, see the links in [External Documentation](#) on page 34.

Apache Impala (incubating) Overview

Impala provides fast, interactive SQL queries directly on your Apache Hadoop data stored in HDFS, HBase, or the Amazon Simple Storage Service (S3). In addition to using the same unified storage platform, Impala also uses the same metadata, SQL syntax (Hive SQL), ODBC driver, and user interface (Impala query UI in Hue) as Apache Hive. This provides a familiar and unified platform for real-time or batch-oriented queries.

Impala is an addition to tools available for querying big data. Impala does not replace the batch processing frameworks built on MapReduce such as Hive. Hive and other frameworks built on MapReduce are best suited for long running batch jobs, such as those involving batch processing of Extract, Transform, and Load (ETL) type jobs.



Note: Impala was accepted into the Apache incubator on December 2, 2015. In places where the documentation formerly referred to “Cloudera Impala”, now the official name is “Apache Impala (incubating)”.

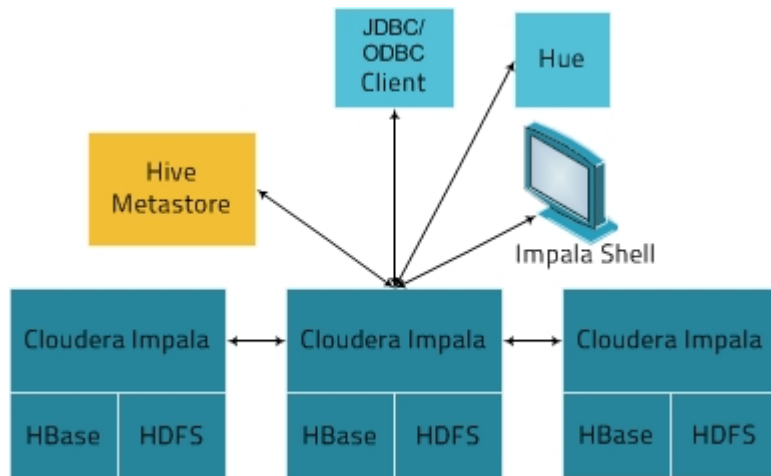
Impala Benefits

Impala provides:

- Familiar SQL interface that data scientists and analysts already know.
- Ability to query high volumes of data (“big data”) in Apache Hadoop.
- Distributed queries in a cluster environment, for convenient scaling and to make use of cost-effective commodity hardware.
- Ability to share data files between different components with no copy or export/import step; for example, to write with Pig, transform with Hive and query with Impala. Impala can read from and write to Hive tables, enabling simple data interchange using Impala for analytics on Hive-produced data.
- Single system for big data processing and analytics, so customers can avoid costly modeling and ETL just for analytics.

How Impala Works with CDH

The following graphic illustrates how Impala is positioned in the broader Cloudera environment:



The Impala solution is composed of the following components:

- Clients - Entities including Hue, ODBC clients, JDBC clients, and the Impala Shell can all interact with Impala. These interfaces are typically used to issue queries or complete administrative tasks such as connecting to Impala.
- Hive Metastore - Stores information about the data available to Impala. For example, the metastore lets Impala know what databases are available and what the structure of those databases is. As you create, drop, and alter schema objects, load data into tables, and so on through Impala SQL statements, the relevant metadata changes are automatically broadcast to all Impala nodes by the dedicated catalog service introduced in Impala 1.2.
- Impala - This process, which runs on DataNodes, coordinates and executes queries. Each instance of Impala can receive, plan, and coordinate queries from Impala clients. Queries are distributed among Impala nodes, and these nodes then act as workers, executing parallel query fragments.
- HBase and HDFS - Storage for data to be queried.

Queries executed using Impala are handled as follows:

1. User applications send SQL queries to Impala through ODBC or JDBC, which provide standardized querying interfaces. The user application may connect to any `impalad` in the cluster. This `impalad` becomes the coordinator for the query.
2. Impala parses the query and analyzes it to determine what tasks need to be performed by `impalad` instances across the cluster. Execution is planned for optimal efficiency.
3. Services such as HDFS and HBase are accessed by local `impalad` instances to provide data.
4. Each `impalad` returns data to the coordinating `impalad`, which sends these results to the client.

Primary Impala Features

Impala provides support for:

- Most common SQL-92 features of Hive Query Language (HiveQL) including [SELECT](#), [joins](#), and [aggregate functions](#).
- HDFS, HBase, and Amazon Simple Storage System (S3) storage, including:
 - [HDFS file formats](#): delimited text files, Parquet, Avro, SequenceFile, and RCFile.
 - Compression codecs: Snappy, GZIP, Deflate, BZIP.
- Common data access interfaces including:
 - [JDBC driver](#).
 - [ODBC driver](#).
 - Hue Beeswax and the Impala Query UI.
- [impala-shell command-line interface](#).
- [Kerberos authentication](#).

Cloudera Search Overview

Cloudera Search provides near real-time (NRT) access to data stored in or ingested into Hadoop and HBase. Search provides near real-time indexing, batch indexing, full-text exploration and navigated drill-down, as well as a simple, full-text interface that requires no SQL or programming skills.

Search is fully integrated in the data-processing platform and uses the flexible, scalable, and robust storage system included with CDH. This eliminates the need to move large data sets across infrastructures to perform business tasks.

Cloudera Search incorporates [Apache Solr](#), which includes Apache Lucene, SolrCloud, Apache Tika, and Solr Cell. Cloudera Search is included with CDH 5.

Using Search with the CDH infrastructure provides:

- Simplified infrastructure
- Better production visibility
- Quicker insights across various data types
- Quicker problem resolution
- Simplified interaction and platform access for more users and use cases
- Scalability, flexibility, and reliability of search services on the same platform used to run other types of workloads on the same data

The following table describes Cloudera Search features.

Table 1: Cloudera Search Features

Feature	Description
Unified management and monitoring with Cloudera Manager	Cloudera Manager provides unified and centralized management and monitoring for CDH and Cloudera Search. Cloudera Manager simplifies deployment, configuration, and monitoring of your search services. Many existing search solutions lack management and monitoring capabilities and

Feature	Description
	fail to provide deep insight into utilization, system health, trending, and other supportability aspects.
Index storage in HDFS	<p>Cloudera Search is integrated with HDFS for index storage. Indexes created by Solr/Lucene can be directly written in HDFS with the data, instead of to local disk, thereby providing fault tolerance and redundancy.</p> <p>Cloudera Search is optimized for fast read and write of indexes in HDFS while indexes are served and queried through standard Solr mechanisms. Because data and indexes are co-located, data processing does not require transport or separately managed storage.</p>
Batch index creation through MapReduce	To facilitate index creation for large data sets, Cloudera Search has built-in MapReduce jobs for indexing data stored in HDFS. As a result, the linear scalability of MapReduce is applied to the indexing pipeline.
Real-time and scalable indexing at data ingest	<p>Cloudera Search provides integration with Flume to support near real-time indexing. As new events pass through a Flume hierarchy and are written to HDFS, those events can be written directly to Cloudera Search indexers.</p> <p>In addition, Flume supports routing events, filtering, and annotation of data passed to CDH. These features work with Cloudera Search for improved index sharding, index separation, and document-level access control.</p>
Easy interaction and data exploration through Hue	A Cloudera Search GUI is provided as a Hue plug-in, enabling users to interactively query data, view result files, and do faceted exploration. Hue can also schedule standing queries and explore index files. This GUI uses the Cloudera Search API, which is based on the standard Solr API.
Simplified data processing for Search workloads	Cloudera Search relies on Apache Tika for parsing and preparation of many of the standard file formats for indexing. Additionally, Cloudera Search supports Avro, Hadoop Sequence, and Snappy file format mappings, as well as Log file formats, JSON, XML, and HTML. Cloudera Search also provides data preprocessing using Morphlines, which simplifies index configuration for these formats. Users can use the configuration for other applications, such as MapReduce jobs.
HBase search	Cloudera Search integrates with HBase, enabling full-text search of stored data without affecting HBase performance. A listener monitors the replication event stream and captures each write or update-replicated event, enabling extraction and mapping. The event is then sent directly to Solr indexers and written to indexes in HDFS, using the same process as for other indexing workloads of Cloudera Search. The indexes can be served immediately, enabling near real-time search of HBase data.

How Cloudera Search Works

In a near real-time indexing use case, Cloudera Search indexes events that are streamed through Apache Flume to be stored in CDH. Fields and events are mapped to standard Solr indexable schemas. Lucene indexes events, and integration with Cloudera Search allows the index to be directly written and stored in standard Lucene index files in HDFS. Flume event routing and storage of data in partitions in HDFS can also be applied. Events can be routed and streamed through multiple Flume agents and written to separate Lucene indexers that can write into separate index shards, for better scale when indexing and quicker responses when searching.

The indexes are loaded from HDFS to Solr cores, exactly like Solr would have read from local disk. The difference in the design of Cloudera Search is the robust, distributed, and scalable storage layer of HDFS, which helps eliminate costly downtime and allows for flexibility across workloads without having to move data. Search queries can then be

submitted to Solr through either the standard Solr API, or through a simple search GUI application, included in Cloudera Search, which can be deployed in Hue.

Cloudera Search batch-oriented indexing capabilities can address needs for searching across batch uploaded files or large data sets that are less frequently updated and less in need of near-real-time indexing. For such cases, Cloudera Search includes a highly scalable indexing workflow based on MapReduce. A MapReduce workflow is launched onto specified files or folders in HDFS, and the field extraction and Solr schema mapping is run during the mapping phase. Reducers use Solr to write the data as a single index or as index shards, depending on your configuration and preferences. Once the indexes are stored in HDFS, they can be queried using standard Solr mechanisms, as previously described above for the near-real-time indexing use case.

The Lily HBase Indexer Service is a flexible, scalable, fault tolerant, transactional, near real-time oriented system for processing a continuous stream of HBase cell updates into live search indexes. Typically, the time between data ingestion using the Flume sink to that content potentially appearing in search results is measured in seconds, although this duration is tunable. The Lily HBase Indexer uses Solr to index data stored in HBase. As HBase applies inserts, updates, and deletes to HBase table cells, the indexer keeps Solr consistent with the HBase table contents, using standard HBase replication features. The indexer supports flexible custom application-specific rules to extract, transform, and load HBase data into Solr. Solr search results can contain `columnFamily:qualifier` links back to the data stored in HBase. This way applications can use the Search result set to directly access matching raw HBase cells. Indexing and searching do not affect operational stability or write throughput of HBase because the indexing and searching processes are separate and asynchronous to HBase.

Understanding Cloudera Search

Cloudera Search fits into the broader set of solutions available for analyzing information in large data sets. With especially large data sets, it is impossible to store all information reliably on a single machine and then query that data. CDH provides both the means and the tools to store the data and run queries. You can explore data through:

- MapReduce jobs
- Cloudera Impala queries
- Cloudera Search queries

CDH provides storage of and access to large data sets using MapReduce jobs, but creating these jobs requires technical knowledge, and each job can take minutes or more to run. The longer run times associated with MapReduce jobs can interrupt the process of exploring data.

To provide more immediate queries and responses and to eliminate the need to write MapReduce applications, Cloudera offers Impala. Impala returns results in seconds instead of minutes.

Although Impala is a fast, powerful application, it uses SQL-based querying syntax. Using Impala can be challenging for users who are not familiar with SQL. If you do not know SQL, you can use Cloudera Search. In addition, Impala, Hive, and Pig all require a structure that is applied at query time, whereas Search supports free-text search on any data or fields you have indexed.

How Search Uses Existing Infrastructure

Any data already in a CDH deployment can be indexed and made available for query by Cloudera Search. For data that is not stored in CDH, Cloudera Search provides tools for loading data into the existing infrastructure, and for indexing data as it is moved to HDFS or written to HBase.

By leveraging existing infrastructure, Cloudera Search eliminates the need to create new, redundant structures. In addition, Cloudera Search uses services provided by CDH and Cloudera Manager in a way that does not interfere with other tasks running in the same environment. This way, you can reuse existing infrastructure without the cost and problems associated with running multiple services in the same set of systems.

Cloudera Search and Other Cloudera Components

Cloudera Search interacts with other Cloudera components to solve different problems. The following table lists Cloudera components that contribute to the Search process and describes how they interact with Cloudera Search:

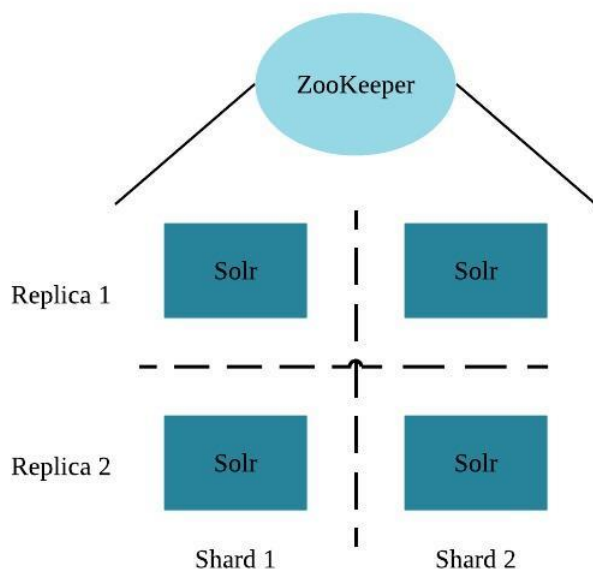
Component	Contribution	Applicable To
HDFS	Stores source documents. Search indexes source documents to make them searchable. Files that support Cloudera Search, such as Lucene index files and write-ahead logs, are also stored in HDFS. Using HDFS provides simpler provisioning on a larger base, redundancy, and fault tolerance. With HDFS, Cloudera Search servers are essentially stateless, so host failures have minimal consequences. HDFS also provides snapshotting, inter-cluster replication, and disaster recovery.	All cases
MapReduce	Search includes a pre-built MapReduce-based job. This job can be used for on-demand or scheduled indexing of any supported data set stored in HDFS. This job uses cluster resources for scalable batch indexing.	Many cases
Flume	Search includes a Flume sink that enables writing events directly to indexers deployed on the cluster, allowing data indexing during ingestion.	Many cases
Hue	Search includes a Hue front-end search application that uses standard Solr APIs. The application can interact with data indexed in HDFS. The application provides support for the Solr standard query language, visualization of faceted search functionality, and a typical full text search GUI-based.	Many cases
Morphlines	A morphline is a rich configuration file that defines an ETL transformation chain. Morphlines can consume any kind of data from any data source, process the data, and load the results into Cloudera Search. Morphlines run in a small, embeddable Java runtime system, and can be used for near real-time applications such as the flume agent as well as batch processing applications such as a Spark job.	Many cases
ZooKeeper	Coordinates distribution of data and metadata, also known as shards. It provides automatic failover to increase service resiliency.	Many cases
Spark	The CrunchIndexerTool can use Spark to move data from HDFS files into Apache Solr, and run the data through a morphline for extraction and transformation.	Some cases
HBase	Supports indexing of stored data, extracting columns, column families, and key information as fields. Because HBase does not use secondary indexing, Cloudera Search can complete full-text searches of content in rows and tables in HBase.	Some cases
Cloudera Manager	Deploys, configures, manages, and monitors Cloudera Search processes and resource utilization across services on the cluster. Cloudera Manager helps simplify Cloudera Search administration, but it is not required.	Some cases
Cloudera Navigator	Cloudera Navigator provides governance for Hadoop systems including support for auditing Search operations.	Some cases
Sentry	Sentry enables role-based, fine-grained authorization for Cloudera Search. Sentry can apply a range of restrictions to various tasks, such as accessing data, managing configurations through config objects, or creating collections. Restrictions are consistently applied, regardless of the way users attempt to complete actions. For example, restricting access to data in a collection restricts that access whether queries come from the command line, a browser, Hue, or through the admin console.	Some cases

Component	Contribution	Applicable To
Oozie	Automates scheduling and management of indexing jobs. Oozie can check for new data and begin indexing jobs as required.	Some cases
Impala	Further analyzes search results.	Some cases
Hive	Further analyzes search results.	Some cases
Parquet	Provides a columnar storage format, enabling especially rapid result returns for structured workloads such as Impala or Hive. Morphlines provide an efficient pipeline for extracting data from Parquet.	Some cases
Avro	Includes metadata that Cloudera Search can use for indexing.	Some cases
Kafka	Search uses this message broker project to increase throughput and decrease latency for handling real-time data.	Some cases
Sqoop	Ingests data in batch and enables data availability for batch indexing.	Some cases
Mahout	Applies machine-learning processing to search results.	Some cases

Cloudera Search Architecture

Cloudera Search runs as a distributed service on a set of servers, and each server is responsible for a portion of the entire set of content to be searched. The entire set of content is split into smaller pieces, copies are made of these pieces, and the pieces are distributed among the servers. This provides two main advantages:

- **Dividing** the content into smaller pieces distributes the task of indexing the content among the servers.
- **Duplicating** the pieces of the whole allows queries to be scaled more effectively and enables the system to provide higher levels of availability.



Each Cloudera Search server can handle requests for information. As a result, a client can send requests to index documents or perform searches to any Search server, and that server routes the request to the correct server.

Each search deployment requires:

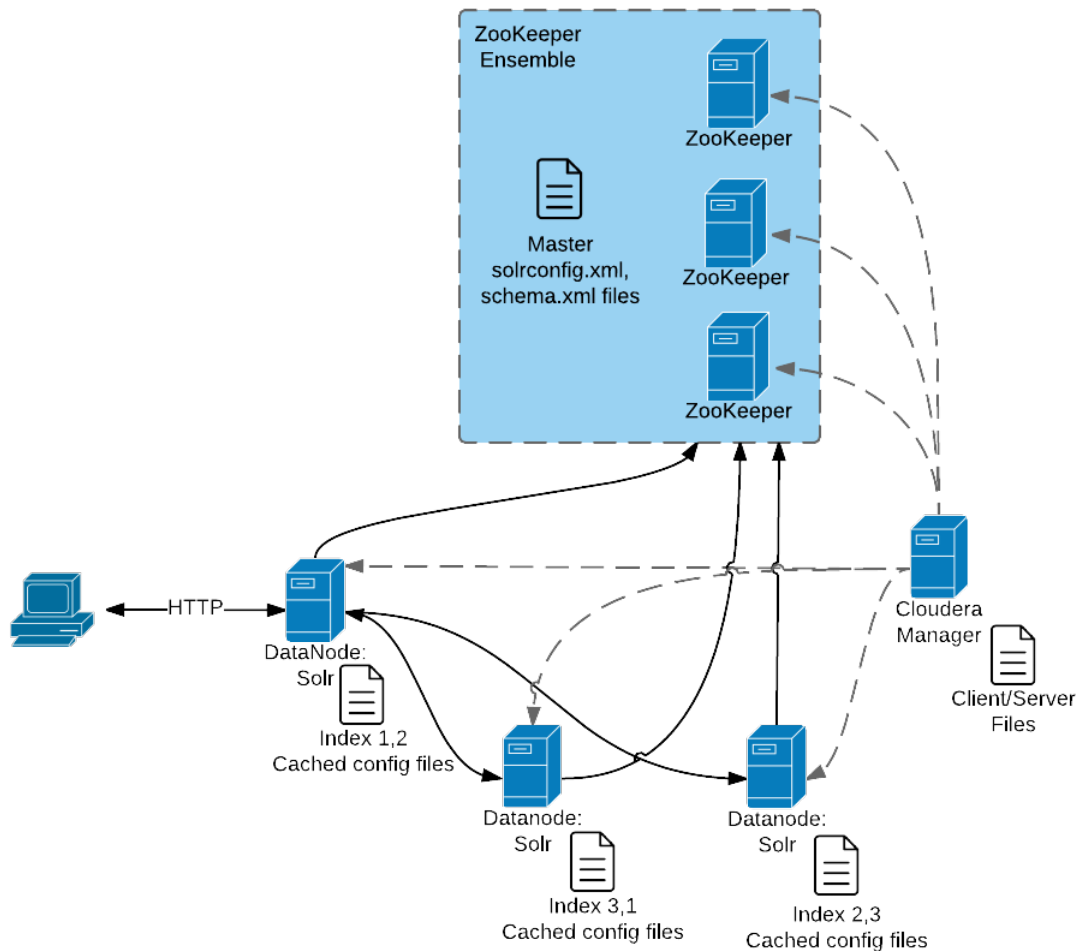
- ZooKeeper on one host. You can install ZooKeeper, Search, and HDFS on the same host.
- HDFS on at least one but as many as all hosts. HDFS is commonly installed on all hosts.

- Solr on at least one but as many as all hosts. Solr is commonly installed on all hosts.

More hosts with Solr and HDFS provides benefits of:

- More search host installations doing work.
- More search and HDFS collocation increasing the degree of data locality. More local data provides faster performance and reduces network traffic.

The following graphic illustrates some of the key elements in a typical deployment.



This graphic illustrates:

1. A client submit a query over HTTP.
2. The response is received by the NameNode and then passed to a DataNode.
3. The DataNode distributes the request among other hosts with relevant shards.
4. The results of the query are gathered and returned to the client.

Also notice that the:

- Cloudera Manager provides client and server configuration files to other servers in the deployment.
- ZooKeeper server provides information about the state of the cluster and the other hosts running Solr.

The information a client must send to complete jobs varies:

- For queries, a client must have the hostname of the Solr server and the port to use.

- For actions related to collections, such as adding or deleting collections, the name of the collection is required as well.
- Indexing jobs, such as MapReduceIndexer jobs, use a MapReduce driver that starts a MapReduce job. These jobs can also process morphlines, indexing the results to add to Solr.

Cloudera Search Configuration Files

Files on which the configuration of a Cloudera Search deployment are based include:

Solr files stored in ZooKeeper. Copies of these files exist on all Solr servers.

- `solrconfig.xml`: Contains the parameters for configuring Solr.
- `schema.xml`: Contains all of the details about which fields your documents can contain, and how those fields should be dealt with when adding documents to the index, or when querying those fields.

Files are copied from `hadoop-conf` in HDFS configurations to Solr servers:

- `core-site.xml`
- `hdfs-site.xml`
- `ssl-client.xml`
- `hadoop-env.sh`
- `topology.map`
- `topology.py`

Cloudera Manager manages the following configuration files:

- `cloudera-monitor.properties`
- `cloudera-stack-monitor.properties`

The following files are used for logging and security configuration:

- `log4j.properties`
- `jaas.conf`
- `solr.keytab`
- `sentry-site.xml`

Search can be deployed using parcels or packages. Some files are always installed to the same location and some files are installed to different locations based on whether the installation is completed using parcels or packages.

Client Files

Client files are always installed to the same location and are required on any host where corresponding services are installed. In a Cloudera Manager environment, Cloudera Manager manages settings. In an unmanaged deployment, all files can be manually edited. All files are found in a subdirectory of `/etc/`. Client configuration file types and their locations are:

- `/etc/solr/conf` for Solr client settings files
- `/etc/hadoop/conf` for HDFS, MapReduce, and YARN client settings files
- `/etc/zookeeper/conf` for ZooKeeper configuration files

Server Files

Server configuration file locations vary based on how services are installed.

- Cloudera Manager environments store configuration all files in `/var/run/`.
- Unmanaged environments store configuration files in `/etc/svc/conf`. For example:
 - `/etc/solr/conf`
 - `/etc/zookeeper/conf`
 - `/etc/hadoop/conf`

Cloudera Search Tasks and Processes

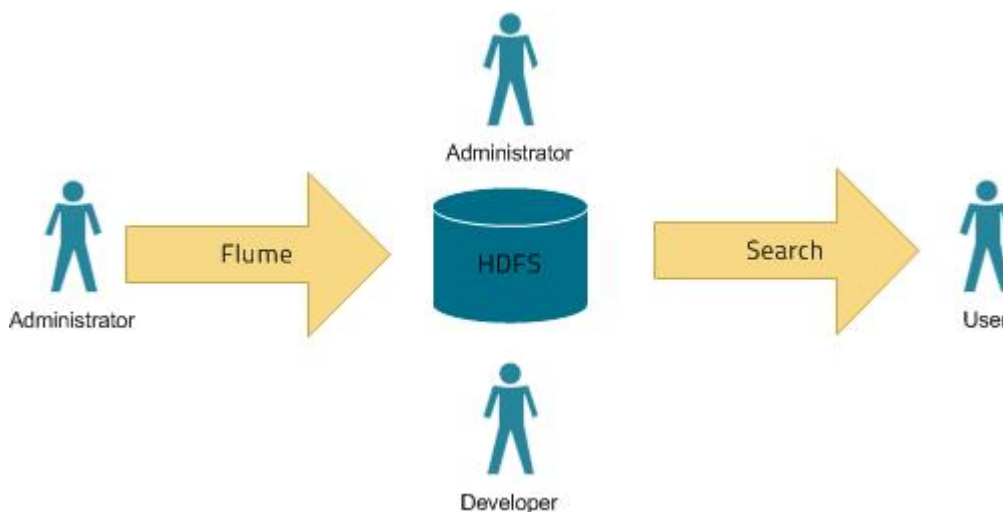
For content to be searchable, it must exist in CDH and be indexed. Content can either already exist in CDH and be indexed on demand, or it can be updated and indexed continuously. To make content searchable, first ensure that it is ingested or stored in CDH.

Ingestion

You can move content to CDH by using:

- Flume, a flexible, agent-based data ingestion framework.
- A copy utility such as `distcp` for HDFS.
- Sqoop, a structured data ingestion connector.
- `fuse-dfs`.

In a typical environment, administrators establish systems for search. For example, HDFS is established to provide storage; Flume or `distcp` are established for content ingestion. After administrators establish these services, users can use ingestion tools such as file copy utilities or Flume sinks.



Indexing

Content must be indexed before it can be searched. Indexing comprises the following steps:

1. Extraction, transformation, and loading (ETL) - Use existing engines or frameworks such as Apache Tika or Cloudera Morphlines.
 - a. Content and metadata extraction
 - b. Schema mapping
2. Create indexes using Lucene.
 - a. Index creation
 - b. Index serialization

Indexes are typically stored on a local file system. Lucene supports additional index writers and readers. One HDFS-based interface implemented as part of Apache Blur is integrated with Cloudera Search and has been optimized for CDH-stored indexes. All index data in Cloudera Search is stored in and served from HDFS.

You can index content in three ways:

Batch indexing using MapReduce

To use MapReduce to index documents, run a MapReduce job on content in HDFS to produce a Lucene index. The Lucene index is written to HDFS, and this index is subsequently used by search services to provide query results.

Batch indexing is most often used when bootstrapping a search cluster. The Map component of the MapReduce task parses input into indexable documents, and the Reduce component contains an embedded Solr server that indexes the documents produced by the Map. You can also configure a MapReduce-based indexing job to use all assigned resources on the cluster, utilizing multiple reducing steps for intermediate indexing and merging operations, and then writing the reduction to the configured set of shard sets for the service. This makes the batch indexing process as scalable as MapReduce workloads.

Near real-time (NRT) indexing using Flume

Flume events are typically collected and written to HDFS. Although any Flume event can be written, logs are most common.

Cloudera Search includes a Flume sink that enables you to write events directly to the indexer. This sink provides a flexible, scalable, fault-tolerant, near real-time (NRT) system for processing continuous streams of records to create live-searchable, free-text search indexes. Typically, data ingested using the Flume sink appears in search results in seconds, although you can tune this duration.

The Flume sink meets the needs of identified use cases that rely on NRT availability. Data can flow from multiple sources through multiple flume hosts. These hosts, which can be spread across a network, route this information to one or more Flume indexing sinks. Optionally, you can split the data flow, storing the data in HDFS while writing it to be indexed by Lucene indexes on the cluster. In that scenario, data exists both as data and as indexed data in the same storage infrastructure. The indexing sink extracts relevant data, transforms the material, and loads the results to live Solr search servers. These Solr servers are immediately ready to serve queries to end users or search applications.

This flexible, customizable system scales effectively because parsing is moved from the Solr server to the multiple Flume hosts for ingesting new content.

Search includes parsers for standard data formats including Avro, CSV, Text, HTML, XML, PDF, Word, and Excel. You can extend the system by adding additional custom parsers for other file or data formats in the form of Tika plug-ins. Any type of data can be indexed: a record is a byte array of any format, and custom ETL logic can handle any format variation.

In addition, Cloudera Search includes a simplifying ETL framework called Cloudera Morphlines that can help adapt and pre-process data for indexing. This eliminates the need for specific parser deployments, replacing them with simple commands.

Cloudera Search is designed to handle a variety of use cases:

- Search supports routing to multiple Solr collections to assign a single set of servers to support multiple user groups (multi-tenancy).
- Search supports routing to multiple shards to improve scalability and reliability.
- Index servers can be collocated with live Solr servers serving end-user queries, or they can be deployed on separate commodity hardware, for improved scalability and reliability.
- Indexing load can be spread across a large number of index servers for improved scalability and can be replicated across multiple index servers for high availability.

This flexible, scalable, highly available system provides low latency data acquisition and low latency querying. Instead of replacing existing solutions, Search complements use cases based on batch analysis of HDFS data using MapReduce. In many use cases, data flows from the producer through Flume to both Solr and HDFS. In this system, you can use NRT ingestion and batch analysis tools.

NRT indexing using some other client that uses the NRT API

Other clients can complete NRT indexing. This is done when the client first writes files directly to HDFS and then triggers indexing using the Solr REST API. Specifically, the API does the following:

1. Extract content from the document contained in HDFS, where the document is referenced by a URL.
2. Map the content to fields in the search schema.
3. Create or update a Lucene index.

This is useful if you index as part of a larger workflow. For example, you could trigger indexing from an Oozie workflow.

Querying

After data is available as an index, the query API provided by the search service allows direct queries to be completed or to be facilitated through a command-line tool or graphical interface. Cloudera Search provides a simple UI application that can be deployed with Hue, or you can create a custom application based on the standard Solr API. Any application that works with Solr is compatible and runs as a search-serving application for Cloudera Search, because Solr is the core.

Apache Sentry Overview

Apache Sentry is a granular, role-based authorization module for Hadoop. Sentry provides the ability to control and enforce precise levels of privileges on data for authenticated users and applications on a Hadoop cluster. Sentry currently works out of the box with Apache Hive, Hive Metastore/HCatalog, Apache Solr, Impala, and HDFS (limited to Hive table data).

Sentry is designed to be a pluggable authorization engine for Hadoop components. It allows you to define authorization rules to validate a user or application's access requests for Hadoop resources. Sentry is highly modular and can support authorization for a wide variety of data models in Hadoop.

For more information, see [Authorization With Apache Sentry](#).

Apache Spark Overview

[Apache Spark](#) is a general framework for distributed computing that offers high performance for both batch and interactive processing. It exposes APIs for Java, Python, and Scala and consists of Spark core and several related projects:

- [Spark SQL](#) - Module for working with structured data. Allows you to seamlessly mix SQL queries with Spark programs.
- [Spark Streaming](#) - API that allows you to build scalable fault-tolerant streaming applications.
- [MLlib](#) - API that implements common [machine learning](#) algorithms.
- [GraphX](#) - API for graphs and graph-parallel computation.

You can run Spark applications locally or distributed across a cluster, either by using an [interactive shell](#) or by [submitting an application](#). Running Spark applications interactively is commonly performed during the data-exploration phase and for ad hoc analysis.

To run applications distributed across a cluster, Spark requires a cluster manager. Cloudera supports two cluster managers: YARN and Spark Standalone. When run on YARN, Spark application processes are managed by the YARN ResourceManager and NodeManager roles. When run on Spark Standalone, Spark application processes are managed by Spark Master and Worker roles.

Unsupported Features

The following Spark features are not supported:

- Spark SQL:
 - Thrift JDBC/ODBC server
 - Spark SQL CLI
- Spark Dataset API
- SparkR
- GraphX
- Spark on Scala 2.11
- Mesos cluster manager

Related Information

- [Managing Spark](#)
- [Monitoring Spark Applications](#)
- [Spark Authentication](#)
- [Spark Encryption](#)
- [Cloudera Spark forum](#)
- [Apache Spark documentation](#)

File Formats and Compression

CDH supports all standard Hadoop file formats. For information about the file formats, see the File-Based Data Structures section of the Hadoop I/O chapter in [Hadoop: The Definitive Guide](#).

The file format has a significant impact on performance. Use [Avro](#) if your use case typically scans or retrieves all of the fields in a row in each query. [Parquet](#) is a better choice if your dataset has many columns, and your use case typically involves working with a subset of those columns instead of entire records. For more information, see this [Parquet versus Avro benchmark study](#).

All file formats include support for [compression](#), which affects the size of data on the disk and, consequently, the amount of I/O and CPU resources required to serialize and deserialize data.

Using Apache Parquet Data Files with CDH

[Apache Parquet](#) is a [columnar storage](#) format available to any component in the Hadoop ecosystem, regardless of the data processing framework, data model, or programming language. The Parquet file format incorporates several features that support data warehouse-style operations:

- Columnar storage layout - A query can examine and perform calculations on all values for a column while reading only a small fraction of the data from a data file or table.
- Flexible compression options - Data can be compressed with any of several codecs. Different data files can be compressed differently.
- Innovative encoding schemes - Sequences of identical, similar, or related data values can be represented in ways that save disk space and memory. The encoding schemes provide an extra level of space savings beyond overall compression for each data file.
- Large file size - The layout of Parquet data files is optimized for queries that process large volumes of data, with individual files in the multimegabyte or even gigabyte range.

Parquet is automatically installed when you install CDH, and the required libraries are automatically placed in the classpath for all CDH components. Copies of the libraries are in `/usr/lib/parquet` or `/opt/cloudera/parcels/CDH/lib/parquet`.

CDH lets you use the component of your choice with the Parquet file format for each phase of data processing. For example, you can read and write Parquet files using Pig and MapReduce jobs. You can convert, transform, and query Parquet tables through Hive, Impala, and Spark. And you can interchange data files between all of these components.

Compression for Parquet Files

For most CDH components, by default Parquet data files are not compressed. Cloudera recommends enabling compression to reduce disk usage and increase read and write performance.

You do not need to specify configuration to read a compressed Parquet file. However, to write a compressed Parquet file, you must specify the compression type. The supported compression types, the compression default, and how you specify compression depends on the CDH component writing the files.

Using Parquet Files in HBase

Parquet files in HBase is a common use case. See [Using a Custom MapReduce Job](#).

Using Parquet Tables in Hive

To create a table named `PARQUET_TABLE` that uses the Parquet format, use a command like the following, substituting your own table name, column names, and data types:

```
hive> CREATE TABLE parquet_table_name (x INT, y STRING) STORED AS PARQUET;
```



Note:

- Once you create a Parquet table, you can query it or insert into it through other components such as Impala and Spark.
- Set `dfs.block.size` to 256 MB in `hdfs-site.xml`.

If the table will be populated with data files generated outside of Impala and Hive, you can create the table as an external table pointing to the location where the files will be created:

```
hive> create external table parquet_table_name (x INT, y STRING)
ROW FORMAT SERDE 'parquet.hive.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT "parquet.hive.DeprecatedParquetInputFormat"
OUTPUTFORMAT "parquet.hive.DeprecatedParquetOutputFormat"
LOCATION '/test-warehouse/tinytable';
```

To populate the table with an `INSERT` statement, and to read the table with a `SELECT` statement, see [Using the Parquet File Format with Impala Tables](#).

To set the compression type to use when writing data, configure the `parquet.compression` property:

```
set parquet.compression=GZIP;
INSERT OVERWRITE TABLE tinytable SELECT * FROM texttable;
```

The supported compression types are `UNCOMPRESSED`, `GZIP`, and `SNAPPY`.

Using Parquet Tables in Impala

Impala can create tables that use Parquet data files, insert data into those tables, convert the data into Parquet format, and query Parquet data files produced by Impala or other components. The only syntax required is the `STORED AS PARQUET` clause on the `CREATE TABLE` statement. After that, all `SELECT`, `INSERT`, and other statements recognize the Parquet format automatically. For example, a session in the `impala-shell` interpreter might look as follows:

```
[localhost:21000] > create table parquet_table (x int, y string) stored as parquet;
[localhost:21000] > insert into parquet_table select x, y from some_other_table;
Inserted 50000000 rows in 33.52s
[localhost:21000] > select y from parquet_table where x between 70 and 100;
```

Once you create a Parquet table this way in Impala, you can query it or insert into it through either Impala or Hive.

The Parquet format is optimized for working with large data files. In Impala 2.0 and higher, the default size of Parquet files written by Impala is 256 MB; in lower releases, 1 GB. Avoid using the `INSERT ... VALUES` syntax, or partitioning the table at too granular a level, if that would produce a large number of small files that cannot use Parquet optimizations for large data chunks.

Inserting data into a partitioned Impala table can be a memory-intensive operation, because each data file requires a memory buffer to hold the data before it is written. Such inserts can also exceed HDFS limits on simultaneous open files, because each node could potentially write to a separate data file for each partition, all at the same time. Make sure table and column statistics are in place for any table used as the source for an `INSERT ... SELECT` operation into a Parquet table. If capacity problems still occur, consider splitting insert operations into one `INSERT` statement per partition.

Impala can query Parquet files that use the `PLAIN`, `PLAIN_DICTIONARY`, `BIT_PACKED`, and `RLE` encodings. Currently, Impala does not support `RLE_DICTIONARY` encoding. When creating files outside of Impala for use by Impala, make

sure to use one of the supported encodings. In particular, for MapReduce jobs, `parquet.writer.version` must not be defined (especially as `PARQUET_2_0`) for writing the configurations of Parquet MR jobs. Use the default version (or format). The default format, 1.0, includes some enhancements that are compatible with older versions. Data using the 2.0 format might not be consumable by Impala, due to use of the `RLE_DICTIONARY` encoding.

If you use Sqoop to convert RDBMS data to Parquet, be careful with interpreting any resulting values from `DATE`, `DATETIME`, or `TIMESTAMP` columns. The underlying values are represented as the Parquet `INT64` type, which is represented as `BIGINT` in the Impala table. The Parquet values represent the time in milliseconds, while Impala interprets `BIGINT` as the time in seconds. Therefore, if you have a `BIGINT` column in a Parquet table that was imported this way from Sqoop, divide the values by 1000 when interpreting as the `TIMESTAMP` type.

For complete instructions and examples, see [Using the Parquet File Format with Impala Tables](#).

Using Parquet Files in MapReduce

MapReduce requires Thrift in its `CLASSPATH` and in `libjars` to access Parquet files. It also requires `parquet-format` in `libjars`. Set up the following before running MapReduce jobs that access Parquet data files:

```
if [ -e /opt/cloudera/parcels/CDH ] ; then
    CDH_BASE=/opt/cloudera/parcels/CDH
else
    CDH_BASE=/usr
fi
THRIFTJAR=`ls -l $CDH_BASE/lib/hive/lib/libthrift*.jar | awk '{print $9}' | head -1`
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$THRIFTJAR
export LIBJARS=`echo "$CLASSPATH" | awk 'BEGIN { RS = ":" } { print }' | grep
parquet-format | tail -1`
export LIBJARS=$LIBJARS,$THRIFTJAR

hadoop jar my-parquet-mr.jar -libjars $LIBJARS
```

Reading Parquet Files in MapReduce

Using the Example helper classes in the Parquet JAR files, a simple map-only MapReduce job that reads Parquet files can use the `ExampleInputFormat` class and the `Group` value class. The following example demonstrates how to read a Parquet file in a MapReduce job; portions of code specific to Parquet are shown in bold.

```
import static java.lang.Thread.sleep;
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import parquet.Log;
import parquet.example.data.Group;
import parquet.hadoop.example.ExampleInputFormat;

public class TestReadParquet extends Configured
    implements Tool {
    private static final Log LOG =
        Log.getLog(TestReadParquet.class);

    /*
     * Read a Parquet record
     */
```

```

    public static class MyMap extends
        Mapper<LongWritable, Group, NullWritable, Text> {

        @Override
        public void map(LongWritable key, Group value, Context context) throws IOException,
            InterruptedException {
            NullWritable outKey = NullWritable.get();
            String outputRecord = "";
            // Get the schema and field values of the record
            String inputRecord = value.toString();
            // Process the value, create an output record
            // ...
            context.write(outKey, new Text(outputRecord));
        }
    }

    public int run(String[] args) throws Exception {

        Job job = new Job(getConf());

        job.setJarByClass(getClass());
        job.setJobName(getClass().getName());
        job.setMapOutputKeyClass(LongWritable.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        job.setMapperClass(MyMap.class);
        job.setNumReduceTasks(0);

        job.setInputFormatClass(ExampleInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
        return 0;
    }

    public static void main(String[] args) throws Exception {
        try {
            int res = ToolRunner.run(new Configuration(), new TestReadParquet(), args);
            System.exit(res);
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(255);
        }
    }
}

```

Writing Parquet Files in MapReduce

When writing Parquet files, you must provide a schema. Specify the schema in the `run` method of the job before submitting it; for example:

```

...
import parquet.Log;
import parquet.example.data.Group;
import parquet.hadoop.example.GroupWriteSupport;
import parquet.hadoop.example.ExampleInputFormat;
import parquet.hadoop.example.ExampleOutputFormat;
import parquet.hadoop.metadata.CompressionCodecName;
import parquet.hadoop.ParquetFileReader;
import parquet.hadoop.metadata.ParquetMetadata;
import parquet.schema.MessageType;
import parquet.schema.MessageTypeParser;
import parquet.schema.Type;
...
public int run(String[] args) throws Exception {
    ...

```



```
String writeSchema = "message example {\n" +
    "required int32 x;\n" +
    "required int32 y;\n" +
    "}";
ExampleOutputFormat.setSchema(
    job,
    MessageTypeParser.parseMessageType(writeSchema));

job.submit();
```

If input files are in Parquet format, the schema can be extracted using the `getSchema` method:

```
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.LocatedFileStatus;
import org.apache.hadoop.fs.RemoteIterator;
...

public int run(String[]
    args) throws Exception {
    ...

    String inputFile = args[0];
    Path parquetFilePath = null;
    // Find a file in case a directory was passed

    RemoteIterator<LocatedFileStatus> it = FileSystem.get(getConf()).listFiles(new
    Path(inputFile), true);
    while(it.hasNext()) {
        FileStatus fs = it.next();

        if(fs.isFile()) {
            parquetFilePath = fs.getPath();
            break;
        }
    }
    if(parquetFilePath == null) {
        LOG.error("No file found for " + inputFile);
        return 1;
    }
    ParquetMetadata readFooter =
        ParquetFileReader.readFooter(getConf(), parquetFilePath);
    MessageType schema =
        readFooter.getFileMetaData().getSchema();
    GroupWriteSupport.setSchema(schema, getConf());

    job.submit();
```

You can then write records in the mapper by composing a `Group` value using the `Example` classes and no key:

```
protected void map(LongWritable key, Text value,
    Mapper<LongWritable, Text, Void, Group>.Context context)
    throws java.io.IOException, InterruptedException {
    int x;
    int y;
    // Extract the desired output values from the input text
    //
    Group group = factory.newGroup()
        .append("x", x)
        .append("y", y);
    context.write(null, group);
}
}
```

To set the compression type before submitting the job, invoke the `setCompression` method:

```
ExampleOutputFormat.setCompression(job, compression_type);
```

The supported compression types are `CompressionCodecName.UNCOMPRESSED`, `CompressionCodecName.GZIP`, and `CompressionCodecName.SNAPPY`.

Using Parquet Files in Pig

Reading Parquet Files in Pig

If the external table is created and populated, the Pig instruction to read the data is:

```
grunt> A = LOAD '/test-warehouse/tinytable' USING parquet.pig.ParquetLoader AS (x: int,
y: int);
```

Writing Parquet Files in Pig

Create and populate a Parquet file with the `ParquetStorer` class:

```
grunt> store A into '/test-warehouse/tinytable' USING parquet.pig.ParquetStorer;
```

To set the compression type, configure the `parquet.compression` property before the first `store` instruction in a Pig script:

```
SET parquet.compression gzip;
```

The supported compression types are `uncompressed`, `gzip`, and `snappy` (the default).

Using Parquet Files in Spark

See [Accessing External Storage](#) and [Accessing Parquet Files From Spark SQL Applications](#).

Parquet File Interoperability

Impala has always included Parquet support, using high-performance code written in C++ to read and write Parquet files. The Parquet JARs for use with Hive, Pig, and MapReduce are available with CDH 4.5 and higher. Using the Java-based Parquet implementation on a CDH release lower than CDH 4.5 is not supported.

A Parquet table created by Hive can typically be accessed by Impala 1.1.1 and higher with no changes, and vice versa. Before Impala 1.1.1, when Hive support for Parquet was not available, Impala wrote a dummy SerDe class name into each data file. These older Impala data files require a one-time `ALTER TABLE` statement to update the metadata for the SerDe class name before they can be used with Hive. See [Apache Impala \(incubating\) Incompatible Changes and Limitations](#) for details.

A Parquet file written by Hive, Impala, Pig, or MapReduce can be read by any of the others. Different defaults for file and block sizes, compression and encoding settings, and so on might cause performance differences depending on which component writes or reads the data files. For example, Impala typically sets the HDFS block size to 256 MB and divides the data files into 256 MB chunks, so that each I/O request reads an entire data file.

In CDH 5.5 and higher, non-Impala components that write Parquet files include extra padding to ensure that the Parquet row groups are aligned with HDFS data blocks. The maximum amount of padding is controlled by the `parquet.writer.max-padding` setting, specified as a number of bytes. By default, up to 8 MB of padding can be added to the end of each row group. This alignment helps prevent remote reads during Impala queries. The setting does not apply to Parquet files written by Impala, because Impala always writes each Parquet file as a single HDFS data block.

Each release may have limitations. The following are current limitations in CDH:

- The `TIMESTAMP` data type in Parquet files is not supported in Hive, Pig, or MapReduce in CDH 4. Attempting to read a Parquet table created with Impala that includes a `TIMESTAMP` column fails.
- Parquet has not been tested with HCatalog. Without HCatalog, Pig cannot correctly read dynamically partitioned tables; this is true for all file formats.
- Impala supports table columns using nested data types or complex data types such as `map`, `struct`, or `array` only in Impala 2.3 (corresponding to CDH 5.5) and higher. Impala 2.2 (corresponding to CDH 5.4) can query only the scalar columns of Parquet files containing such types. Lower releases of Impala cannot query any columns from Parquet data files that include such types.

- Cloudera supports some but not all of the object models from the upstream Parquet-MR project. Currently supported object models are:
 - `parquet-avro` (recommended for Cloudera users)
 - `parquet-thrift`
 - `parquet-protobuf`
 - `parquet-pig`
 - The Impala and Hive object models built into those components, not available in external libraries. (CDH does not include the `parquet-hive` module of the `parquet-mr` project, because recent versions of Hive have Parquet support built in.)

Parquet File Structure

To examine the internal structure and data of Parquet files, you can use the `parquet-tools` command that comes with CDH. Make sure this command is in your `$PATH`. (Typically, it is symlinked from `/usr/bin`; sometimes, depending on your installation setup, you might need to locate it under a CDH-specific `bin` directory.) The arguments to this command let you perform operations such as:

- `cat`: Print a file's contents to standard out. In CDH 5.5 and higher, you can use the `-j` option to output JSON.
- `head`: Print the first few records of a file to standard output.
- `schema`: Print the Parquet schema for the file.
- `meta`: Print the file footer metadata, including key-value properties (like Avro schema), compression ratios, encodings, compression used, and row group information.
- `dump`: Print all data and metadata.

Use `parquet-tools -h` to see usage information for all the arguments. Here are some examples showing `parquet-tools` usage:

```
$ # Be careful doing this for a big file! Use parquet-tools head to be safe.
$ parquet-tools cat sample.parq
year = 1992
month = 1
day = 2
dayofweek = 4
dep_time = 748
crs_dep_time = 750
arr_time = 851
crs_arr_time = 846
carrier = US
flight_num = 53
actual_elapsed_time = 63
crs_elapsed_time = 56
arrdelay = 5
depdelay = -2
origin = CMH
dest = IND
distance = 182
cancelled = 0
diverted = 0

year = 1992
month = 1
day = 3
...
```

```
$ parquet-tools head -n 2 sample.parq
year = 1992
month = 1
day = 2
dayofweek = 4
dep_time = 748
crs_dep_time = 750
```

```

arr_time = 851
crs_arr_time = 846
carrier = US
flight_num = 53
actual_elapsed_time = 63
crs_elapsed_time = 56
arrdelay = 5
depdelay = -2
origin = CMH
dest = IND
distance = 182
cancelled = 0
diverted = 0

year = 1992
month = 1
day = 3
...

```

```

$ parquet-tools schema sample.parq
message schema {
  optional int32 year;
  optional int32 month;
  optional int32 day;
  optional int32 dayofweek;
  optional int32 dep_time;
  optional int32 crs_dep_time;
  optional int32 arr_time;
  optional int32 crs_arr_time;
  optional binary carrier;
  optional int32 flight_num;
  ...

```

```

$ parquet-tools meta sample.parq
creator:      impala version 2.2.0-cdh5.4.3 (build
517bb0f71cd604a00369254ac6d88394df83e0f6)

```

```
file schema:      schema
```

```

-----
year:             OPTIONAL INT32 R:0 D:1
month:            OPTIONAL INT32 R:0 D:1
day:              OPTIONAL INT32 R:0 D:1
dayofweek:        OPTIONAL INT32 R:0 D:1
dep_time:         OPTIONAL INT32 R:0 D:1
crs_dep_time:     OPTIONAL INT32 R:0 D:1
arr_time:         OPTIONAL INT32 R:0 D:1
crs_arr_time:     OPTIONAL INT32 R:0 D:1
carrier:          OPTIONAL BINARY R:0 D:1
flight_num:       OPTIONAL INT32 R:0 D:1
...

```

```
row group 1:      RC:20636601 TS:265103674
```

```

-----
year:             INT32 SNAPPY DO:4 FPO:35 SZ:10103/49723/4.92 VC:20636601
ENC:PLAIN_DICTIONARY,RLE,PLAIN
month:            INT32 SNAPPY DO:10147 FPO:10210 SZ:11380/35732/3.14 VC:20636601
ENC:PLAIN_DICTIONARY,RLE,PLAIN
day:              INT32 SNAPPY DO:21572 FPO:21714 SZ:3071658/9868452/3.21 VC:20636601
ENC:PLAIN_DICTIONARY,RLE,PLAIN
dayofweek:        INT32 SNAPPY DO:3093276 FPO:3093319 SZ:2274375/5941876/2.61
VC:20636601 ENC:PLAIN_DICTIONARY,RLE,PLAIN
dep_time:         INT32 SNAPPY DO:5367705 FPO:5373967 SZ:28281281/28573175/1.01
VC:20636601 ENC:PLAIN_DICTIONARY,RLE,PLAIN
crs_dep_time:     INT32 SNAPPY DO:33649039 FPO:33654262 SZ:10220839/11574964/1.13
VC:20636601 ENC:PLAIN_DICTIONARY,RLE,PLAIN
arr_time:         INT32 SNAPPY DO:43869935 FPO:43876489 SZ:28562410/28797767/1.01
VC:20636601 ENC:PLAIN_DICTIONARY,RLE,PLAIN

```

```

crs_arr_time:          INT32 SNAPPY DO:72432398 FPO:72438151 SZ:10908972/12164626/1.12
VC:20636601 ENC:PLAIN_DICTIONARY,RLE,PLAIN
carrier:              BINARY SNAPPY DO:83341427 FPO:83341558 SZ:114916/128611/1.12
VC:20636601 ENC:PLAIN_DICTIONARY,RLE,PLAIN
flight_num:          INT32 SNAPPY DO:83456393 FPO:83488603 SZ:10216514/11474301/1.12
VC:20636601 ENC:PLAIN_DICTIONARY,RLE,PLAIN
...

```

Examples of Java Programs to Read and Write Parquet Files

You can find full examples of Java code at the Cloudera [Parquet examples](#) GitHub repository.

The [TestReadWriteParquet.java](#) example demonstrates the “identity” transform. It reads any Parquet data file and writes a new file with exactly the same content.

The [TestReadParquet.java](#) example reads a Parquet data file, and produces a new text file in CSV format with the same content.

Using Apache Avro Data Files with CDH

[Apache Avro](#) is a serialization system. Avro supports rich data structures, a compact binary encoding, and a container file for sequences of Avro data (often referred to as **Avro data files**). Avro is language-independent and there are several language bindings for it, including Java, C, C++, Python, and Ruby.

Avro data files have the `.avro` extension. Make sure the files you create have this extension, because some tools use it to determine which files to process as Avro (for example, `AvroInputFormat` and `AvroAsTextInputFormat` for MapReduce and streaming).

Avro does not rely on generated code, so processing data imported from Flume or Sqoop 1 is simpler than using Hadoop Writables in SequenceFiles, where you must ensure that the generated classes are on the processing job classpath. Pig and Hive cannot easily process SequenceFiles with custom Writables, so users often revert to using text, which has disadvantages in compactness and compressibility. Generally, you cannot split compressed text, which makes it difficult to process efficiently using MapReduce.

All components in CDH that produce or consume files support Avro data files.

Compression for Avro Data Files

By default Avro data files are not compressed, but Cloudera recommends enabling compression to reduce disk usage and increase read and write performance. Avro data files support [Deflate](#) and [Snappy](#) compression. Snappy is faster, but Deflate is slightly more compact.

You do not need to specify configuration to read a compressed Avro data file. However, to write an Avro data file, you must specify the type of compression. How you specify compression depends on the component.

Using Avro Data Files in Flume

The [HDFSEventSink](#) used to serialize event data onto HDFS supports plug-in implementations of the [EventSerializer](#) interface. Implementations of this interface have full control over the serialization format and can be used in cases where the default serialization format provided by the sink is insufficient.

An abstract implementation of the `EventSerializer` interface, called [AbstractAvroEventSerializer](#), is provided with Flume. This class can be extended to support custom schemas for Avro serialization over HDFS. The [FlumeEventAvroEventSerializer](#) class provides a simple implementation that maps the events to a representation of a String header map and byte payload in Avro. Use this class by setting the serializer property of the sink as follows:

```
agent-name.sinks.sink-name.serializer = AVRO_EVENT
```

Using Avro Data Files in Hive

The following example demonstrates how to create a Hive table backed by Avro data files:

```
CREATE TABLE doctors
ROW FORMAT
SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
TBLPROPERTIES ('avro.schema.literal'='{
  "namespace": "testing.hive.avro.serde",
  "name": "doctors",
  "type": "record",
  "fields": [
    {
      "name": "number",
      "type": "int",
      "doc": "Order of playing the role"
    },
    {
      "name": "first_name",
      "type": "string",
      "doc": "first name of actor playing role"
    },
    {
      "name": "last_name",
      "type": "string",
      "doc": "last name of actor playing role"
    },
    {
      "name": "extra_field",
      "type": "string",
      "doc": "an extra field not in the original file",
      "default": "fishfingers and custard"
    }
  ]
}');
```

```
LOAD DATA LOCAL INPATH '/usr/share/doc/hive-0.7.1+42.55/examples/files/doctors.avro'
INTO TABLE doctors;
```

You can also create an Avro backed Hive table by using an Avro schema file:

```
CREATE TABLE my_avro_table(notused INT)
ROW FORMAT SERDE
'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
WITH SERDEPROPERTIES (
  'avro.schema.url'='file:///tmp/schema.avsc')
STORED as INPUTFORMAT
'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT
'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat';
```

`avro.schema.url` is a URL (here a `file://` URL) pointing to an Avro schema file used for reading and writing. It could also be an `hdfs://` URL; for example, `hdfs://hadoop-namenode-uri/examplefile`.

To enable Snappy compression on output files, run the following before writing to the table:

```
SET hive.exec.compress.output=true;
SET avro.output.codec=snappy;
```

Also include the `snappy-java` JAR in `--auxpath`, which is located at `/usr/lib/hive/lib/snappy-java-1.0.4.1.jar` or `/opt/cloudera/parcels/CDH/lib/hive/lib/snappy-java-1.0.4.1.jar`.

[Haivvreo SerDe](#) has been merged into Hive as [AvroSerDe](#) and is no longer supported in its original form. `schema.url` and `schema.literal` have been changed to `avro.schema.url` and `avro.schema.literal` as a result of the merge. If you were using [Haivvreo SerDe](#), you can use the Hive [AvroSerDe](#) with tables created with the [Haivvreo](#)

Serde. For example, if you have a table `my_avro_table` that uses the `Haivvreo SerDe`, add the following to make the table use the new `AvroSerDe`:

```
ALTER TABLE my_avro_table SET SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe';

ALTER TABLE my_avro_table SET FILEFORMAT
INPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.AvroContainerOutputFormat';
```

Using Avro Data Files in MapReduce

The Avro MapReduce API is an Avro module for running MapReduce programs that produce or consume Avro data files.

If you are using Maven, add the following dependency to your POM:

```
<dependency>
  <groupId>org.apache.avro</groupId>
  <artifactId>avro-mapred</artifactId>
  <version>1.7.3</version>
  <classifier>hadoop2</classifier>
</dependency>
```

Then write your program, using the [Avro MapReduce javadoc](#) for guidance.

At run time, include the `avro` and `avro-mapred` JARs in the `HADOOP_CLASSPATH` and the `avro`, `avro-mapred` and `paranamer` JARs in `-libjars`.

To enable Snappy compression on output, call `AvroJob.setOutputCodec(job, "snappy")` when configuring the job. You must also include the `snappy-java` JAR in `-libjars`.

Using Avro Data Files in Pig

CDH provides `AvroStorage` for Avro integration in Pig.

To use it, first register the `piggybank` JAR file and supporting libraries:

```
REGISTER piggybank.jar
REGISTER lib/avro-1.7.3.jar
REGISTER lib/json-simple-1.1.jar
REGISTER lib/snappy-java-1.0.4.1.jar
```

Then load Avro data files as follows:

```
a = LOAD 'my_file.avro' USING org.apache.pig.piggybank.storage.avro.AvroStorage();
```

Pig maps the Avro schema to a corresponding Pig schema.

You can store data in Avro data files with:

```
store b into 'output' USING org.apache.pig.piggybank.storage.avro.AvroStorage();
```

With `store`, Pig generates an Avro schema from the Pig schema. You can override the Avro schema by specifying it literally as a parameter to `AvroStorage` or by using the same schema as an existing Avro data file. See the [Pig wiki](#) for details.

To store two relations in one script, specify an index to each `store` function. For example:

```
set1 = load 'input1.txt' using PigStorage() as ( ... );
store set1 into 'set1' using org.apache.pig.piggybank.storage.avro.AvroStorage('index',
'1');

set2 = load 'input2.txt' using PigStorage() as ( ... );
store set2 into 'set2' using org.apache.pig.piggybank.storage.avro.AvroStorage('index',
'2');
```

For more information, search for "index" in the [AvroStorage wiki](#).

To enable Snappy compression on output files, do the following before issuing the `STORE` statement:

```
SET mapred.output.compress true
SET mapred.output.compression.codec org.apache.hadoop.io.compress.SnappyCodec
SET avro.output.codec snappy
```

For more information, see the [Pig wiki](#). The version numbers of the JAR files to register are different on that page, so adjust them as shown above.

Importing Avro Data Files in Sqoop 1

On the command line, use the following option to import Avro data files:

```
--as-avrodatafile
```

Sqoop 1 automatically generates an Avro schema that corresponds to the database table being exported from.

To enable Snappy compression, add the following option:

```
--compression-codec snappy
```



Note: Sqoop 2 does not currently support Avro.

Using Avro Data Files in Spark

See [Accessing External Storage](#) and [Accessing Avro Data Files From Spark SQL Applications](#).

Using Avro Data Files in Streaming Programs

To read from Avro data files from a streaming program, specify `org.apache.avro.mapred.AvroAsTextInputFormat` as the input format. This format converts each datum in the Avro data file to a string. For a "bytes" schema, this is the raw bytes; in general cases, this is a single-line [JSON](#) representation.

To write to Avro data files from a streaming program, specify `org.apache.avro.mapred.AvroTextOutputFormat` as the output format. This format creates Avro data files with a "bytes" schema, where each datum is a tab-delimited key-value pair.

At run time, specify the `avro`, `avro-mapred`, and `paranamer` JARs in `-libjars` in the streaming command.

To enable Snappy compression on output files, set the property `avro.output.codec` to `snappy`. You must also include the `snappy-java` JAR in `-libjars`.

Data Compression

Data compression and compression formats can have a significant impact on performance. Three important places to consider data compression are in MapReduce and Spark jobs, data stored in HBase, and Impala queries. For the most part, the principles are similar for each.

You must balance the processing capacity required to compress and uncompress the data, the disk IO required to read and write the data, and the network bandwidth required to send the data across the network. The correct balance of these factors depends upon the characteristics of your cluster and your data, as well as your usage patterns.

Compression is not recommended if your data is already compressed (such as images in JPEG format). In fact, the resulting file can sometimes be larger than the original.

For more information about compression algorithms in Hadoop, see the Compression section of the Hadoop I/O chapter in [Hadoop: The Definitive Guide](#).

Compression Types

Hadoop supports the following compression types and codecs:

- `gzip` - `org.apache.hadoop.io.compress.GzipCodec`
- `bzip2` - `org.apache.hadoop.io.compress.BZip2Codec`
- `LZO` - `com.hadoop.compression.lzo.LzopCodec`
- `Snappy` - `org.apache.hadoop.io.compress.SnappyCodec`
- `Deflate` - `org.apache.hadoop.io.compress.DeflateCodec`

Different file types and CDH components support different compression types. For details, see [Using Apache Avro Data Files with CDH](#) on page 29 and [Using Apache Parquet Data Files with CDH](#) on page 21.

For guidelines on choosing compression types and configuring compression, see [Choosing and Configuring Data Compression](#).

Snappy Compression

[Snappy](#) is a compression/decompression library. It optimizes for very high-speed compression and decompression, and moderate compression instead of maximum compression or compatibility with other compression libraries.

Snappy is supported for all CDH components. How you specify compression depends on the component.

Using Snappy with HBase

If you install Hadoop and HBase from RPM or Debian packages, Snappy requires no HBase configuration.

Using Snappy with Hive

To enable Snappy compression for Hive output when creating `SequenceFile` outputs, use the following settings:

```
SET hive.exec.compress.output=true;
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;
SET mapred.output.compression.type=BLOCK;
```

Using Snappy with MapReduce

Enabling MapReduce intermediate compression can make jobs run faster without requiring application changes. Only the temporary intermediate files created by Hadoop for the shuffle phase are compressed; the final output may or may not be compressed. Snappy is ideal in this case because it compresses and decompresses very quickly compared to other compression algorithms, such as Gzip. For information about choosing a compression format, see [Choosing and Configuring Data Compression](#).

To enable Snappy for MapReduce intermediate compression for the whole cluster, set the following properties in `mapred-site.xml`:

- MRv1

```
<property>
  <name>mapred.compress.map.output</name>
  <value>true</value>
</property>
<property>
  <name>mapred.map.output.compression.codec</name>
  <value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

- YARN

```
<property>
  <name>mapreduce.map.output.compress</name>
  <value>true</value>
</property>
<property>
  <name>mapred.map.output.compress.codec</name>
```

```
<value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

You can also set these properties on a per-job basis.

Use the properties in the following table to compress the final output of a MapReduce job. These are usually set on a per-job basis.

MRv1 Property	YARN Property	Description
mapred.output.compress	mapreduce.output.fileoutputformat.compress	Whether to compress the final job outputs (true or false).
mapred.output.compression.codec	mapreduce.output.fileoutputformat.compress.codec	If the final job outputs are to be compressed, the codec to use. Set to <code>org.apache.hadoop.io.compress.SnappyCodec</code> for Snappy compression.
mapred.output.compression.type	mapreduce.output.fileoutputformat.compress.type	For SequenceFile outputs, the type of compression to use (NONE, RECORD, or BLOCK). Cloudera recommends BLOCK.



Note: The MRv1 property names are also supported (but deprecated) in YARN. You do not need to update them in this release.

Using Snappy with Pig

Set the same properties for Pig as for MapReduce.

Using Snappy with Spark SQL

To enable Snappy compression for Spark SQL when writing tables, specify the `snappy` codec in the `spark.sql.parquet.compression.codec` configuration:

```
sqlContext.setConf("spark.sql.parquet.compression.codec", "snappy")
```

Using Snappy Compression with Sqoop 1 and Sqoop 2 Imports

- **Sqoop 1** - On the command line, use the following option to enable Snappy compression:

```
--compression-codec org.apache.hadoop.io.compress.SnappyCodec
```

Cloudera recommends using the `--as-sequentialfile` option with this compression option.

- **Sqoop 2** - When you create a job (`sqoop:000> create job`), choose 7 (SNAPPY) as the compression format.

External Documentation

Cloudera provides documentation for CDH as a whole, whether your CDH cluster is managed by Cloudera Manager or not. In addition, you may find it useful to refer to documentation for the individual components included in CDH. Where possible, these links point to the main documentation for a project, in the Cloudera release archive. This ensures that you are looking at the correct documentation for the version of a project included in CDH. Otherwise, the links may point to the project's main site.

- [Apache Avro](#)
- [Apache Crunch](#)
- [Apache DataFu](#)
- [Apache Flume](#)

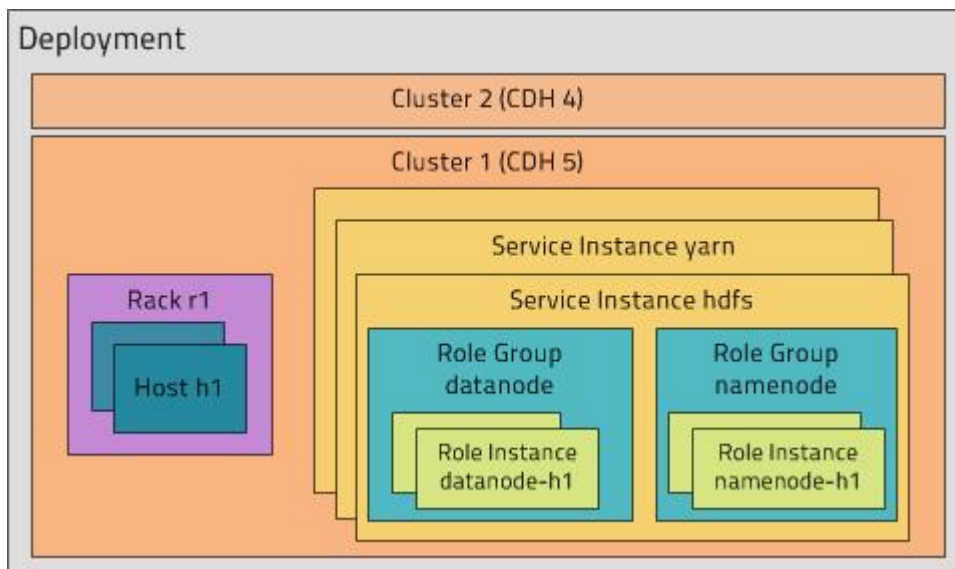
- [Apache Hadoop](#)
- [Apache HBase](#)
- [Apache Hive](#)
- [Hue](#)
- [Kite](#)
- [Apache Mahout](#)
- [Apache Oozie](#)
- [Apache Parquet](#)
- [Apache Pig](#)
- [Apache Sentry](#)
- [Apache Solr](#)
- [Apache Spark](#)
- [Apache Sqoop](#)
- [Apache Sqoop2](#)
- [Apache Whirr](#)
- [Apache ZooKeeper](#)

Cloudera Manager 5 Overview

Cloudera Manager is an end-to-end application for managing CDH clusters. Cloudera Manager sets the standard for enterprise deployment by delivering granular visibility into and control over every part of the CDH cluster—empowering operators to improve performance, enhance quality of service, increase compliance and reduce administrative costs. With Cloudera Manager, you can easily deploy and centrally operate the complete CDH stack and other managed services. The application automates the installation process, reducing deployment time from weeks to minutes; gives you a cluster-wide, real-time view of hosts and services running; provides a single, central console to enact configuration changes across your cluster; and incorporates a full range of reporting and diagnostic tools to help you optimize performance and utilization. This primer introduces the basic concepts, structure, and functions of Cloudera Manager.

Terminology

To effectively use Cloudera Manager, you should first understand its terminology. The relationship between the terms is illustrated below and their definitions follow:



Some of the terms, such as cluster and service, will be used without further explanation. Others, such as role group, gateway, host template, and parcel are expanded upon in later sections.

A common point of confusion is the overloading of the terms **service** and **role** for both types and instances; Cloudera Manager and this section sometimes uses the same term for type and instance. For example, the Cloudera Manager Admin Console **Home** > **Status** tab and **Clusters** > **ClusterName** menu lists service instances. This is similar to the practice in programming languages where for example the term "string" may indicate either a type (`java.lang.String`) or an instance of that type ("hi there"). When it's necessary to distinguish between types and instances, the word "type" is appended to indicate a type and the word "instance" is appended to explicitly indicate an instance.

deployment

A configuration of Cloudera Manager and all the clusters it manages.

dynamic resource pool

In Cloudera Manager, a named configuration of resources and a policy for scheduling the resources among YARN applications or Impala queries running in the pool.

cluster

- A set of computers or racks of computers that contains an [HDFS](#) filesystem and runs [MapReduce](#) and other processes on that data. A pseudo-distributed cluster is a [CDH](#) installation run on a single machine and useful for demonstrations and individual study.
- In Cloudera Manager, a logical entity that contains a set of hosts, a single version of CDH installed on the hosts, and the service and role instances running on the hosts. A host can belong to only one cluster. Cloudera Manager can manage multiple CDH clusters, however each cluster can only be associated with a single Cloudera Manager Server or [Cloudera Manager HA pair](#).

host

In Cloudera Manager, a physical or virtual machine that runs role instances. A host can belong to only one cluster.

rack

In Cloudera Manager, a physical entity that contains a set of physical hosts typically served by the same switch.

service

- A Linux command that runs a System V init script in `/etc/init.d/` in as predictable an environment as possible, removing most environment variables and setting the current working directory to `/`.
- A category of managed functionality in Cloudera Manager, which may be distributed or not, running in a cluster. Sometimes referred to as a service type. For example: MapReduce, HDFS, YARN, Spark, and Accumulo. In traditional environments, multiple services run on one host; in distributed systems, a service runs on many hosts.

service instance

In Cloudera Manager, an instance of a service running on a cluster. For example: "HDFS-1" and "yarn". A service instance spans many role instances.

role

In Cloudera Manager, a category of functionality within a service. For example, the HDFS service has the following roles: NameNode, SecondaryNameNode, DataNode, and Balancer. Sometimes referred to as a role type. See also [user role](#).

role instance

In Cloudera Manager, an instance of a role running on a host. It typically maps to a Unix process. For example: "NameNode-h1" and "DataNode-h1".

role group

In Cloudera Manager, a set of configuration properties for a set of role instances.

host template

A set of role groups in Cloudera Manager. When a template is applied to a host, a role instance from each role group is created and assigned to that host.

gateway

In Cloudera Manager, role that designates a host that should receive a client configuration for a service when the host does not have any role instances for that service running on it.

parcel

A binary distribution format that contains compiled code and meta-information such as a package description, version, and dependencies.

static service pool




















In Cloudera Manager, a static partitioning of total cluster resources—CPU, memory, and I/O weight—across a set of services.

Cluster Example

Consider a cluster **Cluster 1** with four hosts as shown in the following listing from Cloudera Manager:

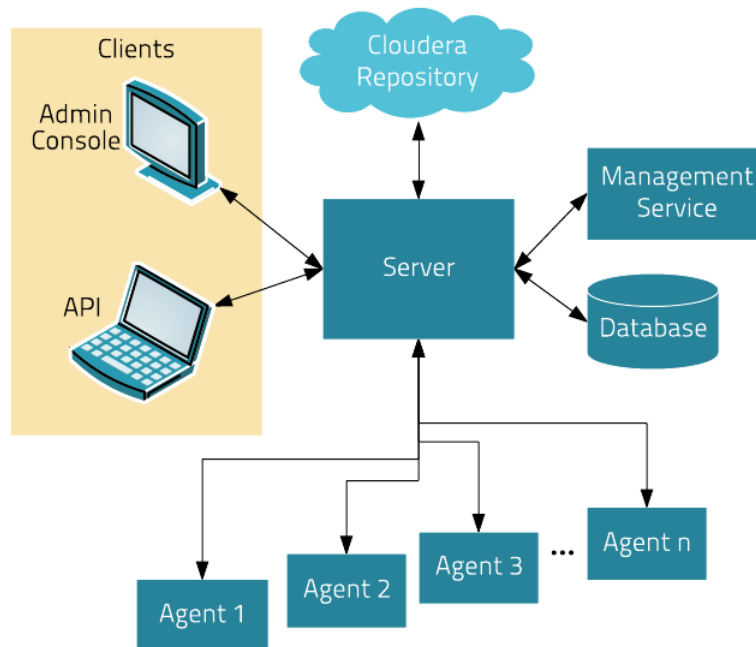
<input type="checkbox"/>	⚡	⬆ Name	⚡ IP	⚡ Roles	⚡ Load Average	⚡ Disk Usage	⚡ Physical Memory	⚡ Swap Space
<input type="checkbox"/>	🟢	tcdn501-1.ent.cloudera.com	10.20.195.240	➤ 21 Role(s)	0.04 0.15 0.26	<div><div></div>11.3 GiB / 57 GiB</div>	<div><div></div>6.3 GiB / 9.7 GiB</div>	<div><div></div>4.7 MiB / 2 GiB</div>
<input type="checkbox"/>	🟢	tcdn501-2.ent.cloudera.com	10.20.81.81	➤ 7 Role(s)	0.07 0.07 0.05	<div><div></div>8.9 GiB / 57 GiB</div>	<div><div></div>2 GiB / 9.7 GiB</div>	<div><div></div>0 B / 2 GiB</div>
<input type="checkbox"/>	🟢	tcdn501-3.ent.cloudera.com	10.20.190.234	➤ 7 Role(s)	0.08 0.11 0.04	<div><div></div>8.9 GiB / 57 GiB</div>	<div><div></div>2 GiB / 9.7 GiB</div>	<div><div></div>0 B / 2 GiB</div>
<input type="checkbox"/>	🟢	tcdn501-4.ent.cloudera.com	10.20.195.243	➤ 7 Role(s)	0.06 0.23 0.23	<div><div></div>8.9 GiB / 57 GiB</div>	<div><div></div>2 GiB / 9.7 GiB</div>	<div><div></div>0 B / 2 GiB</div>
<div><div>First</div><div>Previous</div><div>1</div><div>Next</div><div>Last</div></div>								

The host **tcdn501-1** is the "master" host for the cluster, so it has many more role instances, 21, compared with the 7 role instances running on the other hosts. In addition to the CDH "master" role instances, **tcdn501-1** also has Cloudera Management Service roles:

Service	Instance	Name
None	None	deploy-client-config
 HBase	Master	hbase-MASTER
 HDFS	NameNode	hdfs-NAMENODE
 HDFS	SecondaryNameNode	hdfs-SECONDARYNAMENODE
 Hive	Hive Metastore Server	hive-HIVEMETASTORE
 Hive	HiveServer2	hive-HIVESERVER2
 Hue	Hue Server	hue-HUE_SERVER
 Impala	Impala Catalog Server	impala-CATALOGSERVER
 Impala	Impala StateStore	impala-STATESTORE
 Cloudera Management Service	Alert Publisher	cloudera-mgmt-ALERTPUBLISHER
 Cloudera Management Service	Event Server	cloudera-mgmt-EVENTSERVER
 Cloudera Management Service	Host Monitor	cloudera-mgmt-HOSTMONITOR
 Cloudera Management Service	Navigator Audit Server	cloudera-mgmt-NAVIGATOR
 Cloudera Management Service	Navigator Metadata Server	cloudera-mgmt-NAVIGATORMETASERVER
 Cloudera Management Service	Reports Manager	cloudera-mgmt-REPORTSMANAGER
 Cloudera Management Service	Service Monitor	cloudera-mgmt-SERVICEMONITOR
 Oozie	Oozie Server	oozie-OOZIE_SERVER
 Spark	Master	spark-SPARK_MASTER
 YARN (MR2 Included)	JobHistory Server	yarn-JOBHISTORY
 YARN (MR2 Included)	ResourceManager	yarn-RESOURCEMANAGER

Architecture

As depicted below, the heart of Cloudera Manager is the Cloudera Manager Server. The Server hosts the Admin Console Web Server and the application logic, and is responsible for installing software, configuring, starting, and stopping services, and managing the cluster on which the services run.



The Cloudera Manager Server works with several other components:

- **Agent** - installed on every host. The agent is responsible for starting and stopping processes, unpacking configurations, triggering installations, and monitoring the host.
- **Management Service** - a service consisting of a set of roles that perform various monitoring, alerting, and reporting functions.
- **Database** - stores configuration and monitoring information. Typically, multiple logical databases run across one or more database servers. For example, the Cloudera Manager Server and the monitoring roles use different logical databases.
- **Cloudera Repository** - repository of software for distribution by Cloudera Manager.
- **Clients** - are the interfaces for interacting with the server:
 - **Admin Console** - Web-based UI with which administrators manage clusters and Cloudera Manager.
 - **API** - API with which developers create custom Cloudera Manager applications.

Heartbeating

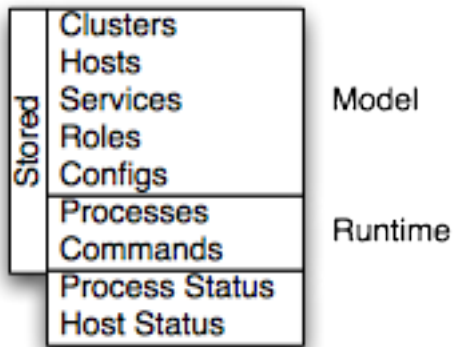
Heartbeats are a primary communication mechanism in Cloudera Manager. By default Agents send heartbeats every 15 seconds to the Cloudera Manager Server. However, to reduce user latency the frequency is increased when state is changing.

During the heartbeat exchange, the Agent notifies the Cloudera Manager Server of its activities. In turn the Cloudera Manager Server responds with the actions the Agent should be performing. Both the Agent and the Cloudera Manager Server end up doing some reconciliation. For example, if you start a service, the Agent attempts to start the relevant processes; if a process fails to start, the Cloudera Manager Server marks the start command as having failed.

State Management

The Cloudera Manager Server maintains the state of the cluster. This state can be divided into two categories: "model" and "runtime", both of which are stored in the Cloudera Manager Server database.

State Maintained by CM Server



Cloudera Manager models CDH and managed services: their roles, configurations, and inter-dependencies. *Model state* captures what is supposed to run where, and with what configurations. For example, model state captures the fact that a cluster contains 17 hosts, each of which is supposed to run a DataNode. You interact with the model through the Cloudera Manager Admin Console configuration screens and API and operations such as "Add Service".

Runtime state is what processes are running where, and what commands (for example, rebalance HDFS or run a Backup/Disaster Recovery schedule or rolling restart or stop) are currently running. The runtime state includes the exact configuration files needed to run a process. When you select Start in the Cloudera Manager Admin Console, the server gathers up all the configuration for the relevant services and roles, validates it, generates the configuration files, and stores them in the database.

When you update a configuration (for example, the Hue Server web port), you have updated the model state. However, if Hue is running while you do this, it is still using the old port. When this kind of mismatch occurs, the role is marked as having an "outdated configuration". To resynchronize, you restart the role (which triggers the configuration re-generation and process restart).

While Cloudera Manager models all of the reasonable configurations, some cases inevitably require special handling. To allow you to work around, for example, a bug or to explore unsupported options, Cloudera Manager supports an "[advanced configuration snippet](#)" mechanism that lets you add properties directly to the configuration files.

Configuration Management








Cloudera Manager defines configuration at several levels:

- The service level may define configurations that apply to the entire service instance, such as an HDFS service's default replication factor (`dfs.replication`).
- The [role group](#) level may define configurations that apply to the member roles, such as the DataNodes' handler count (`dfs.datanode.handler.count`). This can be set differently for different groups of DataNodes. For example, DataNodes running on more capable hardware may have more handlers.
- The role instance level may override configurations that it inherits from its role group. This should be used sparingly, because it easily leads to configuration divergence within the role group. One example usage is to temporarily enable debug logging in a specific role instance to troubleshoot an issue.
- Hosts have configurations related to monitoring, software management, and resource management.
- Cloudera Manager itself has configurations related to its own administrative operations.

Role Groups

You can set configuration at the service instance (for example, HDFS) or role instance (for example, the DataNode on host17). An individual role inherits the configurations set at the service level. Configurations made at the role level override those inherited from the service level. While this approach offers flexibility, configuring a set of role instances in the same way can be tedious.

Cloudera Manager supports role groups, a mechanism for assigning configurations to a group of role instances. The members of those groups then inherit those configurations. For example, in a cluster with heterogeneous hardware, a DataNode role group can be created for each host type and the DataNodes running on those hosts can be assigned to their corresponding role group. That makes it possible to set the configuration for all the DataNodes running on the same hardware by modifying the configuration of one role group. The HDFS service discussed earlier has the following role groups defined for the service's roles:

<input type="checkbox"/>  Role Name	State	Host	Role Group
<input type="checkbox"/>  Balancer	N/A	tcdn501-1.ent.cloudera.com	Balancer Default Group
<input type="checkbox"/>  DataNode	Started	tcdn501-2.ent.cloudera.com	DataNode Default Group
<input type="checkbox"/>  DataNode	Started	tcdn501-3.ent.cloudera.com	DataNode Default Group
<input type="checkbox"/>  DataNode	Started	tcdn501-4.ent.cloudera.com	DataNode Default Group
<input type="checkbox"/>  NameNode (Active)	Started	tcdn501-1.ent.cloudera.com	NameNode Default Group
<input type="checkbox"/>  SecondaryNameNode	Started	tcdn501-1.ent.cloudera.com	SecondaryNameNode Default Group

In addition to making it easy to manage the configuration of subsets of roles, role groups also make it possible to maintain different configurations for experimentation or managing shared clusters for different users or workloads.

Host Templates

In typical environments, sets of hosts have the same hardware and the same set of services running on them. A host template defines a set of role groups (at most one of each type) in a cluster and provides two main benefits:

- Adding new hosts to clusters easily - multiple hosts can have roles from different services created, configured, and started in a single operation.
- Altering the configuration of roles from different services on a set of hosts easily - which is useful for quickly switching the configuration of an entire cluster to accommodate different workloads or users.

Server and Client Configuration

Administrators are sometimes surprised that modifying `/etc/hadoop/conf` and then restarting HDFS has no effect. That is because service instances started by Cloudera Manager do not read configurations from the default locations. To use HDFS as an example, when not managed by Cloudera Manager, there would usually be one HDFS configuration per host, located at `/etc/hadoop/conf/hdfs-site.xml`. Server-side daemons and clients running on the same host would all use that same configuration.

Cloudera Manager distinguishes between server and client configuration. In the case of HDFS, the file `/etc/hadoop/conf/hdfs-site.xml` contains only configuration relevant to an HDFS client. That is, by default, if you run a program that needs to communicate with Hadoop, it will get the addresses of the NameNode and JobTracker, and other important configurations, from that directory. A similar approach is taken for `/etc/hbase/conf` and `/etc/hive/conf`.

In contrast, the HDFS role instances (for example, NameNode and DataNode) obtain their configurations from a private per-process directory, under `/var/run/cloudera-scm-agent/process/unique-process-name`. Giving each process its own private execution and configuration environment allows Cloudera Manager to control each process independently. For example, here are the contents of an example `879-hdfs-NAMENODE` process directory:

```
$ tree -a /var/run/cloudera-scm-agent/process/879-hdfs-NAMENODE /
/var/run/cloudera-scm-agent/process/879-hdfs-NAMENODE/
  cloudera_manager_agent_fencer.py
  cloudera_manager_agent_fencer_secret_key.txt
  cloudera-monitor.properties
  core-site.xml
  dfs_hosts_allow.txt
  dfs_hosts_exclude.txt
  event-filter-rules.json
  hadoop-metrics2.properties
```

```

hdfs.keytab
hdfs-site.xml
log4j.properties
logs
  stderr.log
  stdout.log
topology.map
topology.py

```

Distinguishing between server and client configuration provides several advantages:

- Sensitive information in the server-side configuration, such as the password for the Hive Metastore RDBMS, is not exposed to the clients.
- A service that depends on another service may deploy with customized configuration. For example, to get good HDFS read performance, Impala needs a specialized version of the HDFS client configuration, which may be harmful to a generic client. This is achieved by separating the HDFS configuration for the Impala daemons (stored in the per-process directory mentioned above) from that of the generic client (`/etc/hadoop/conf`).
- Client configuration files are much smaller and more readable. This also avoids confusing non-administrator Hadoop users with irrelevant server-side properties.

Deploying Client Configurations and Gateways

A client configuration is a zip file that contains the relevant configuration files with the settings for a service. Each zip file contains the set of configuration files needed by the service. For example, the MapReduce client configuration zip file contains copies of `core-site.xml`, `hadoop-env.sh`, `hdfs-site.xml`, `log4j.properties`, and `mapred-site.xml`. Cloudera Manager supports a **Download Client Configuration** action to enable distributing the client configuration file to users outside the cluster.

Cloudera Manager can deploy client configurations within the cluster; each applicable service has a **Deploy Client Configuration** action. This action does not necessarily deploy the client configuration to the entire cluster; it only deploys the client configuration to all the hosts that this service has been assigned to. For example, suppose a cluster has 10 hosts, and a MapReduce service is running on hosts 1-9. When you use Cloudera Manager to deploy the MapReduce client configuration, host 10 will not get a client configuration, because the MapReduce service has no role assigned to it. This design is intentional to avoid deploying conflicting client configurations from multiple services.

To deploy a client configuration to a host that does not have a role assigned to it you use a gateway. A **gateway** is a marker to convey that a service should be accessible from a particular host. Unlike all other roles it has no associated process. In the preceding example, to deploy the MapReduce client configuration to host 10, you assign a MapReduce gateway role to that host.

Gateways can also be used to customize client configurations for some hosts. Gateways can be placed in role groups and those groups can be configured differently. However, unlike role instances, there is no way to override configurations for gateway instances.

In the cluster we discussed earlier, the three hosts (**tcdn501-[2-5]**) that do not have Hive role instances have Hive gateways:

<input type="checkbox"/>	↑ Role Name	↑ State	↑ Host	↑ Role Group
<input type="checkbox"/>	Gateway	N/A	tcdn501-2.ent.cloudera.com	Gateway Default Group
<input type="checkbox"/>	Gateway	N/A	tcdn501-3.ent.cloudera.com	Gateway Default Group
<input type="checkbox"/>	Gateway	N/A	tcdn501-4.ent.cloudera.com	Gateway Default Group
<input type="checkbox"/>	Gateway	N/A	tcdn501-1.ent.cloudera.com	Gateway Default Group
<input type="checkbox"/>	Hive Metastore Server	Started	tcdn501-1.ent.cloudera.com	Hive Metastore Server Default Group
<input type="checkbox"/>	HiveServer2	Started	tcdn501-1.ent.cloudera.com	HiveServer2 Default Group

Process Management

In a non-Cloudera Manager managed cluster, you most likely start a role instance process using an `init` script, for example, `service hadoop-hdfs-datanode start`. Cloudera Manager does not use `init` scripts for the daemons it manages; in a Cloudera Manager managed cluster, starting and stopping services using `init` scripts will not work.

In a Cloudera Manager managed cluster you can only start or stop role instance processes using Cloudera Manager. Cloudera Manager uses an open source process management tool called `supervisord`, that starts processes, takes care of redirecting log files, notifying of process failure, setting the effective user ID of the calling process to the right user, and so on. Cloudera Manager supports automatically restarting a crashed process. It will also flag a role instance with a bad health flag if its process crashes repeatedly right after start up.

Stopping the Cloudera Manager Server and the Cloudera Manager Agents will not bring down your services; any running role instances keep running.

The Agent is started by `init.d` at start-up. It, in turn, contacts the Cloudera Manager Server and determines which processes should be running. The Agent is monitored as part of Cloudera Manager's host monitoring: if the Agent stops heartbeating, the host is marked as having bad health.

One of the Agent's main responsibilities is to start and stop processes. When the Agent detects a new process from the Server heartbeat, the Agent creates a directory for it in `/var/run/cloudera-scm-agent` and unpacks the configuration. It then contacts `supervisord`, which starts the process.

These actions reinforce an important point: a Cloudera Manager process never travels alone. In other words, a process is more than just the arguments to `exec()` — it also includes configuration files, directories that need to be created, and other information.

Software Distribution Management

A major function of Cloudera Manager is to install CDH and managed service software. Cloudera Manager installs software for new deployments and to upgrade existing deployments. Cloudera Manager supports two software distribution formats: packages and parcels.

A **package** is a binary distribution format that contains compiled code and meta-information such as a package description, version, and dependencies. Package management systems evaluate this meta-information to allow package searches, perform upgrades to a newer version, and ensure that all dependencies of a package are fulfilled. Cloudera Manager uses the native system package manager for each supported OS.

A **parcel** is a binary distribution format containing the program files, along with additional metadata used by Cloudera Manager. The important differences between parcels and packages are:

- Parcels are self-contained and installed in a versioned directory, which means that multiple versions of a given parcel can be installed side-by-side. You can then designate one of these installed versions as the active one. With packages, only one package can be installed at a time so there's no distinction between what's installed and what's active.
- Parcels can be installed at any location in the filesystem and by default are installed in `/opt/cloudera/parcels`. In contrast, packages are installed in `/usr/lib`.
- Parcel handling automatically downloads, distributes, and activates the correct parcel for the operating system running on each host in the cluster. All CDH hosts that make up a logical cluster need to run on the same major OS release to be covered by Cloudera Support. Cloudera Manager needs to run on the same OS release as one of the CDH clusters it manages, to be covered by Cloudera Support. The risk of issues caused by running different minor OS releases is considered lower than the risk of running different major OS releases. Cloudera recommends running the same minor release cross-cluster, because it simplifies issue tracking and supportability.

Because of their unique properties, parcels offer the following advantages over packages:

- **Distribution of CDH as a single object** - Instead of having a separate package for each part of CDH, parcels have just a single object to install. This makes it easier to distribute software to a cluster that is not connected to the Internet.

- **Internal consistency** - All CDH components are matched, eliminating the possibility of installing parts from different versions of CDH.
- **Installation outside of `/usr`** - In some environments, Hadoop administrators do not have privileges to install system packages. These administrators needed to use CDH tarballs, which do not provide the infrastructure that packages do. With parcels, administrators can install to `/opt`, or anywhere else, without completing the additional manual steps of regular tarballs.



Note: With parcels, the path to the CDH libraries is `/opt/cloudera/parcels/CDH/lib` instead of the usual `/usr/lib`. Do not link `/usr/lib/` elements to parcel-deployed paths, because the links may cause scripts that distinguish between the two paths to not work.

- **Installation of CDH without `sudo`** - Parcel installation is handled by the Cloudera Manager Agent running as root or another user, so you can install CDH without `sudo`.
- **Decoupled distribution from activation** - With side-by-side install capabilities, you can stage a new version of CDH across the cluster before switching to it. This allows the most time-consuming part of an upgrade to be done ahead of time without affecting cluster operations, thereby reducing downtime.
- **Rolling upgrades** - Packages require you to shut down the old process, upgrade the package, and then start the new process. Any errors in the process can be difficult to recover from, and upgrading requires extensive integration with the package management system to function seamlessly. With parcels, when a new version is staged side-by-side, you can switch to a new minor version by simply changing which version of CDH is used when restarting each process. You can then perform upgrades with [rolling restarts](#), in which service roles are restarted in the correct order to switch to the new version with minimal service interruption. Your cluster can continue to run on the existing installed components while you stage a new version across your cluster, without impacting your current operations. Major version upgrades (for example, CDH 4 to CDH 5) require full service restarts because of substantial changes between the versions. Finally, you can upgrade individual parcels or multiple parcels at the same time.
- **Upgrade management** - Cloudera Manager manages all the steps in a CDH version upgrade. With packages, Cloudera Manager only helps with initial installation.
- **Additional components** - Parcels are not limited to CDH. Cloudera Impala, Cloudera Search, LZO, Apache Kafka, and [add-on service](#) parcels are also available.
- **Compatibility with other distribution tools** - Cloudera Manager works with other tools you use for download and distribution. For example, you can use Puppet. Or, you can download the parcel to Cloudera Manager Server manually if your cluster has no Internet connectivity and then have Cloudera Manager distribute the parcel to the cluster.

Host Management

Cloudera Manager provides several features to manage the hosts in your Hadoop clusters. The first time you run Cloudera Manager Admin Console you can search for hosts to add to the cluster and once the hosts are selected you can map the assignment of CDH roles to hosts. Cloudera Manager automatically deploys all software required to participate as a managed host in a cluster: JDK, Cloudera Manager Agent, CDH, Impala, Solr, and so on to the hosts.

Once the services are deployed and running, the Hosts area within the Admin Console shows the overall status of the managed hosts in your cluster. The information provided includes the version of CDH running on the host, the cluster to which the host belongs, and the number of roles running on the host. Cloudera Manager provides operations to manage the lifecycle of the participating hosts and to add and delete hosts. The Cloudera Management Service Host Monitor role performs health tests and collects host metrics to allow you to monitor the health and performance of the hosts.

Resource Management

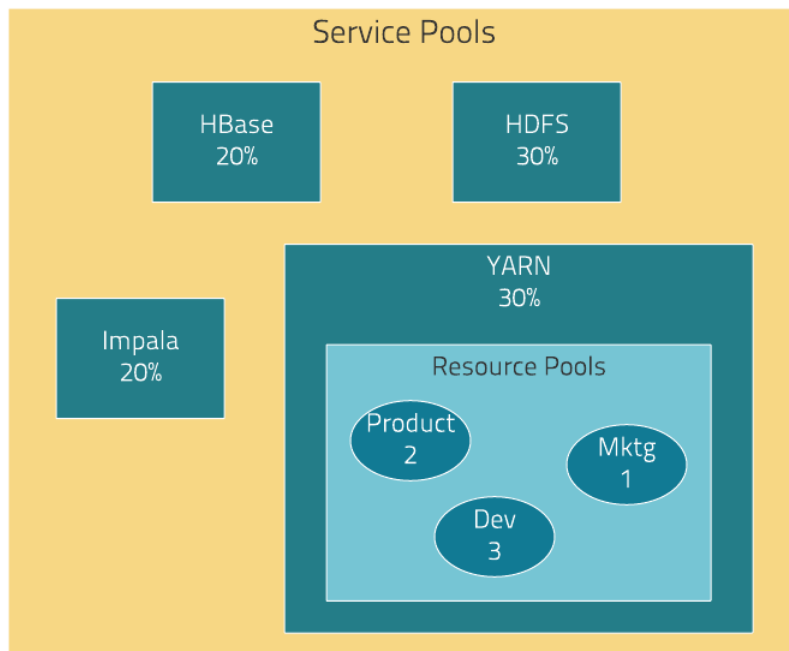
Resource management helps ensure predictable behavior by defining the impact of different services on cluster resources. Use resource management to:

- Guarantee completion in a reasonable time frame for critical workloads.
- Support reasonable cluster scheduling between groups of users based on fair allocation of resources per group.
- Prevent users from depriving other users access to the cluster.

With Cloudera Manager 5, statically allocating resources using cgroups is configurable through a single *static service pool* [wizard](#). You allocate services as a percentage of total resources, and the wizard configures the cgroups.

Static service pools isolate the services in your cluster from one another, so that load on one service has a bounded impact on other services. Services are allocated a static percentage of total resources—CPU, memory, and I/O weight—which are not shared with other services. When you configure static service pools, Cloudera Manager computes recommended memory, CPU, and I/O configurations for the worker roles of the services that correspond to the percentage assigned to each service. Static service pools are implemented per role group within a cluster, using [Linux control groups \(cgroups\)](#) and cooperative memory limits (for example, Java maximum heap sizes). Static service pools can be used to control access to resources by HBase, HDFS, Impala, MapReduce, Solr, Spark, YARN, and [add-on](#) services. Static service pools are not enabled by default.

For example, the following figure illustrates static pools for HBase, HDFS, Impala, and YARN services that are respectively assigned 20%, 30%, 20%, and 30% of cluster resources.



You can dynamically apportion resources that are statically allocated to YARN and Impala by using *dynamic resource pools*.

Depending on the version of CDH you are using, dynamic resource pools in Cloudera Manager support the following scenarios:

- **YARN (CDH 5)** - YARN manages the virtual cores, memory, running applications, and scheduling policy for each pool. In the preceding diagram, three dynamic resource pools—Dev, Product, and Mktg with weights 3, 2, and 1 respectively—are defined for YARN. If an application starts and is assigned to the Product pool, and other applications are using the Dev and Mktg pools, the Product resource pool receives $30\% \times \frac{2}{6}$ (or 10%) of the total cluster resources. If no applications are using the Dev and Mktg pools, the YARN Product pool is allocated 30% of the cluster resources.
- **Impala (CDH 5 and CDH 4)** - Impala manages memory for pools running queries and limits the number of running and queued queries in each pool.

User Management

Access to Cloudera Manager features is controlled by user accounts. A user account identifies how a user is authenticated and determines what privileges are granted to the user.

Cloudera Manager provides several mechanisms for authenticating users. You can configure Cloudera Manager to authenticate users against the Cloudera Manager database or against an [external authentication service](#). The external authentication service can be an LDAP server (Active Directory or an OpenLDAP compatible directory), or you can specify another external service. Cloudera Manager also supports using the Security Assertion Markup Language (SAML) to enable single sign-on.

For information about the privileges associated with each of the Cloudera Manager user roles, see [Cloudera Manager User Roles](#).

Security Management

Cloudera Manager strives to consolidate security configurations across several projects.

Authentication

The purpose of authentication in Hadoop, as in other systems, is simply to prove that a user or service is who he or she claims to be.

Typically, authentication in enterprises is managed through a single distributed system, such as a Lightweight Directory Access Protocol (LDAP) directory. LDAP authentication consists of straightforward username/password services backed by a variety of storage systems, ranging from file to database.

A common enterprise-grade authentication system is Kerberos. Kerberos provides strong security benefits including capabilities that render intercepted authentication packets unusable by an attacker. It virtually eliminates the threat of impersonation by never sending a user's credentials in cleartext over the network.

Several components of the Hadoop ecosystem are converging to use Kerberos authentication with the option to manage and store credentials in LDAP or AD. For example, Microsoft's Active Directory (AD) is an LDAP directory that also provides Kerberos authentication for added security.

Authorization

Authorization is concerned with who or what has access or control over a given resource or service. Since Hadoop merges together the capabilities of multiple varied, and previously separate IT systems as an enterprise data hub that stores and works on all data within an organization, it requires multiple authorization controls with varying granularities. In such cases, Hadoop management tools simplify setup and maintenance by:

- Tying all users to groups, which can be specified in existing LDAP or AD directories.
- Providing role-based access control for similar interaction methods, like batch and interactive SQL queries. For example, Apache Sentry permissions apply to Hive (HiveServer2) and Impala.

CDH currently provides the following forms of access control:

- Traditional POSIX-style permissions for directories and files, where each directory and file is assigned a single owner and group. Each assignment has a basic set of permissions available; file permissions are simply read, write, and execute, and directories have an additional permission to determine access to child directories.
- [Extended Access Control Lists](#) (ACLs) for HDFS that provide fine-grained control of permissions for HDFS files by allowing you to set different permissions for specific named users or named groups.
- Apache HBase uses ACLs to authorize various operations (READ, WRITE, CREATE, ADMIN) by column, column family, and column family qualifier. HBase ACLs are granted and revoked to both users and groups.
- Role-based access control with [Apache Sentry](#).

Encryption

The goal of encryption is to ensure that only authorized users can view, use, or contribute to a data set. These security controls add another layer of protection against potential threats by end-users, administrators, and other malicious actors on the network. Data protection can be applied at a number of levels within Hadoop:

- **OS Filesystem-level** - Encryption can be applied at the Linux operating system filesystem level to cover all files in a volume. An example of this approach is [Cloudera Navigator Encrypt](#) (formerly Gazzang zNcrypt) which is available for Cloudera customers licensed for Cloudera Navigator. Navigator Encrypt operates at the Linux volume level, so it can encrypt cluster data inside and outside HDFS, such as temp/spill files, configuration files and metadata databases (to be used only for data related to a CDH cluster). Navigator Encrypt must be used with [Cloudera Navigator Key Trustee Server](#) (formerly Gazzang zTrustee).

CDH components, such as Impala, MapReduce, YARN, or HBase, also have the ability to encrypt data that lives temporarily on the local filesystem outside HDFS. To enable this feature, see [Configuring Encryption for Data Spills](#).

- **Network-level** - Encryption can be applied to encrypt data just before it gets sent across a network and to decrypt it just after receipt. In Hadoop, this means coverage for data sent from client user interfaces as well as service-to-service communication like remote procedure calls (RPCs). This protection uses industry-standard protocols such as TLS/SSL.



Note: Cloudera Manager and CDH components support either TLS 1.0, TLS 1.1, or TLS 1.2, but not SSL 3.0. References to SSL continue only because of its widespread use in technical jargon.

- **HDFS-level** - Encryption applied by the HDFS client software. [HDFS Transparent Encryption](#) operates at the HDFS folder level, allowing you to encrypt some folders and leave others unencrypted. HDFS transparent encryption cannot encrypt any data outside HDFS. To ensure reliable key storage (so that data is not lost), use Cloudera Navigator Key Trustee Server; the default Java keystore can be used for test purposes. For more information, see [Enabling HDFS Encryption Using Cloudera Navigator Key Trustee Server](#).

Unlike OS and network-level encryption, HDFS transparent encryption is end-to-end. That is, it protects data at rest and in transit, which makes it more efficient than implementing a combination of OS-level and network-level encryption.

Cloudera Management Service

The Cloudera Management Service implements various management features as a set of roles:

- Activity Monitor - collects information about activities run by the MapReduce service. This role is not added by default.
- Host Monitor - collects health and metric information about hosts
- Service Monitor - collects health and metric information about services and activity information from the YARN and Impala services
- Event Server - aggregates relevant Hadoop events and makes them available for alerting and searching
- Alert Publisher - generates and delivers alerts for certain types of events
- Reports Manager - generates reports that provide an historical view into disk utilization by user, user group, and directory, processing activities by user and YARN pool, and HBase tables and namespaces. This role is not added in Cloudera Express.




In addition, for certain editions of the Cloudera Enterprise license, the Cloudera Management Service provides the [Navigator Audit Server](#) and [Navigator Metadata Server](#) roles for [Cloudera Navigator](#).

Health Tests

Cloudera Manager monitors the health of the services, roles, and hosts that are running in your clusters using **health tests**. The Cloudera Management Service also provides health tests for its roles. Role-based health tests are enabled by default. For example, a simple health test is whether there's enough disk space in every NameNode data directory.

A more complicated health test may evaluate when the last checkpoint for HDFS was compared to a threshold or whether a DataNode is connected to a NameNode. Some of these health tests also aggregate other health tests: in a distributed system like HDFS, it's normal to have a few DataNodes down (assuming you've got dozens of hosts), so we allow for setting thresholds on what percentage of hosts should color the entire service down.

Health tests can return one of three values: **Good**, **Concerning**, and **Bad**. A test returns **Concerning** health if the test falls below a warning threshold. A test returns **Bad** if the test falls below a critical threshold. The overall health of a service or role instance is a roll-up of its health tests. If any health test is **Concerning** (but none are **Bad**) the role's or service's health is **Concerning**; if any health test is **Bad**, the service's or role's health is **Bad**.

In the Cloudera Manager Admin Console, health tests results are indicated with colors: **Good** , **Concerning** , and **Bad** .

One common question is whether monitoring can be separated from configuration. One of the goals for monitoring is to enable it without needing to do additional configuration and installing additional tools (for example, Nagios). By having a deep model of the configuration, Cloudera Manager is able to know which directories to monitor, which ports to use, and what credentials to use for those ports. This tight coupling means that, when you install Cloudera Manager all the monitoring is enabled.

Metric Collection and Display

To perform monitoring, the Service Monitor and Host Monitor collects metrics. A **metric** is a numeric value, associated with a name (for example, "CPU seconds"), an entity it applies to ("host17"), and a timestamp. Most metric collection is performed by the Agent. The Agent communicates with a supervised process, requests the metrics, and forwards them to the Service Monitor. In most cases, this is done once per minute.


A few special metrics are collected by the Service Monitor. For example, the Service Monitor hosts an HDFS canary, which tries to write, read, and delete a file from HDFS at regular intervals, and measure whether it succeeded, and how long it took. Once metrics are received, they're aggregated and stored.

Using the Charts page in the Cloudera Manager Admin Console, you can query and explore the metrics being collected. Charts display **time series**, which are streams of metric data points for a specific entity. Each metric data point contains a timestamp and the value of that metric at that timestamp.

Some metrics (for example, `total_cpu_seconds`) are counters, and the appropriate way to query them is to take their rate over time, which is why a lot of metrics queries contain the `dt0` function. For example, `dt0(total_cpu_seconds)`. (The `dt0` syntax is intended to remind you of derivatives. The 0 indicates that the rate of a monotonically increasing counter should never have negative rates.)

Events, Alerts, and Triggers

An **event** is a record that something of interest has occurred – a service's health has changed state, a log message (of the appropriate severity) has been logged, and so on. Many events are enabled and configured by default.

An **alert** is an event that is considered especially noteworthy and is triggered by a selected event. Alerts are shown with an  badge when they appear in a list of [events](#). You can configure the Alert Publisher to send alert notifications by email or by SNMP trap to a trap receiver.

A **trigger** is a statement that specifies an action to be taken when one or more specified conditions are met for a service, role, role configuration group, or host. The conditions are expressed as a [tsquery statement](#), and the action to be taken is to change the health for the service, role, role configuration group, or host to either Concerning (yellow) or Bad (red).

Overview of EMC DSSD D5 Integration

The EMC DSSD D5 provides a high-speed, low-latency storage solution based on flash media. It has been optimized for use as storage for DataNodes in the Cloudera CDH distribution. The DataNode hosts connect directly to the DSSD D5 using a PCIe card interface. In a CDH cluster, only the DataNodes use the DSSD D5 for storage; all other hosts use standard disks.

To manage clusters that use DSSD D5 storage, enable **DSSD Mode** in Cloudera Manager. All other Hadoop components operate normally. When this mode is enabled, Cloudera Manager can only manage clusters with DSSD D5 DataNodes; you cannot mix cluster types (a cluster that uses only DSSD D5 DataNodes and a cluster that uses regular DataNodes). All DataNodes must connect to the DSSD D5; you cannot mix DataNode types within a cluster.

You can connect multiple instances of a DSSD D5 appliance to a single cluster by defining each DSSD D5 as a "rack." See [Configuring Multiple DSSD D5 Appliances in a Cluster](#).

Cloudera Manager Admin Console



Cloudera Manager Admin Console is the web-based UI that you use to configure, manage, and monitor CDH.

If no services are configured when you log into the Cloudera Manager Admin Console, the Cloudera Manager installation wizard displays. If services have been configured, the Cloudera Manager top navigation bar:



and [Home](#) page display. The Cloudera Manager Admin Console top navigation bar provides the following tabs and menus:

- **Clusters > *cluster_name***
 - **Services** - Display individual services, and the Cloudera Management Service. In these pages you can:
 - View the status and other details of a service instance or the role instances associated with the service
 - Make configuration changes to a service instance, a role, or a specific role instance
 - Add and delete a service or role
 - Stop, start, or restart a service or role.
 - View the commands that have been run for a service or a role
 - View an audit event history
 - Deploy and download client configurations
 - Decommission and recommission role instances
 - Enter or exit maintenance mode
 - Perform actions unique to a specific type of service. For example:
 - Enable HDFS high availability or NameNode federation
 - Run the HDFS Balancer
 - Create HBase, Hive, and Sqoop directories
 - **Hosts** - Displays the hosts in the cluster.
 - **Dynamic Resource Pools** - Manage dynamic allocation of cluster resources to YARN and Impala services by specifying the relative weights of named pools.
 - **Static Service Pools** - Manage static allocation of cluster resources to HBase, HDFS, Impala, MapReduce, and YARN services.
 - **Reports** - Create reports about the HDFS, MapReduce, YARN, and Impala usage and browse HDFS files, and manage quotas for HDFS directories.
 - **Impala_service_name Queries** - Query information about Impala queries running on your cluster.
 - **MapReduce_service_name Jobs** - Query information about MapReduce jobs running on your cluster.
 - **YARN_service_name Applications** - Query information about YARN applications running on your cluster.
- **Hosts** - Display the hosts managed by Cloudera Manager. In this page you can:
 - View the status and a variety of detail metrics about individual hosts
 - Make configuration changes for host monitoring
 - View all the processes running on a host
 - Run the Host Inspector
 - Add and delete hosts
 - Create and manage host templates


- Manage parcels
- Decommission and recommission hosts
- Make rack assignments
- Run the host upgrade wizard
- **Diagnostics** - Review logs, events, and alerts to diagnose problems. The subpages are:
 - **Events** - Search for and displaying events and alerts that have occurred.
 - **Logs** - Search logs by service, role, host, and search phrase as well as log level (severity).
 - **Server Log** - Display the Cloudera Manager Server log.
- **Audits** - Query and filter audit events across clusters, including logins, across clusters.
- **Charts** - Query for metrics of interest, display them as charts, and display personalized chart dashboards.
- **Backup** - Manage replication schedules and snapshot policies.
- **Administration** - Administer Cloudera Manager. The subpages are:
 - **Settings** - Configure Cloudera Manager.
 - **Alerts** - Display when alerts will be generated, configure alert recipients, and send test alert email.
 - **Users** - Manage Cloudera Manager users and user sessions.
 - **Kerberos** - Generate Kerberos credentials and inspect hosts.
 - **License** - Manage Cloudera licenses.
 - **Language** - Set the language used for the content of activity events, health events, and alert email messages.
 - **Peers** - Connect multiple instances of Cloudera Manager.
- **Parcel Icon**  link to the **Hosts > Parcels** page.
- **Running Commands Indicator**  displays the number of commands currently running for all services or roles.
- **Search** - Supports searching for services, roles, hosts, configuration properties, and commands. You can enter a partial string and a drop-down list with up to sixteen entities that match will display.
- **Support** - Displays various support actions. The subcommands are:
 - **Send Diagnostic Data** - Sends data to Cloudera Support to support troubleshooting.
 - **Support Portal (Cloudera Enterprise)** - Displays the Cloudera Support portal.
 - **Mailing List (Cloudera Express)** - Displays the Cloudera Manager Users list.
 - **Scheduled Diagnostics: Weekly** - Configure the frequency of automatically collecting diagnostic data and sending to Cloudera support.
 - The following links open the latest documentation on the Cloudera web site:
 - **Help**
 - **Installation Guide**
 - **API Documentation**
 - **Release Notes**
 - **About** - Version number and build details of Cloudera Manager and the current date and time stamp of the Cloudera Manager server.
- **Logged-in User Menu** - The currently logged-in user. The subcommands are:
 - **Change Password** - Change the password of the currently logged in user.
 - **Logout**

Starting and Logging into the Admin Console

1. In a web browser, enter `http://Server host:7180`, where *Server host* is the FQDN or IP address of the host where the Cloudera Manager Server is running.

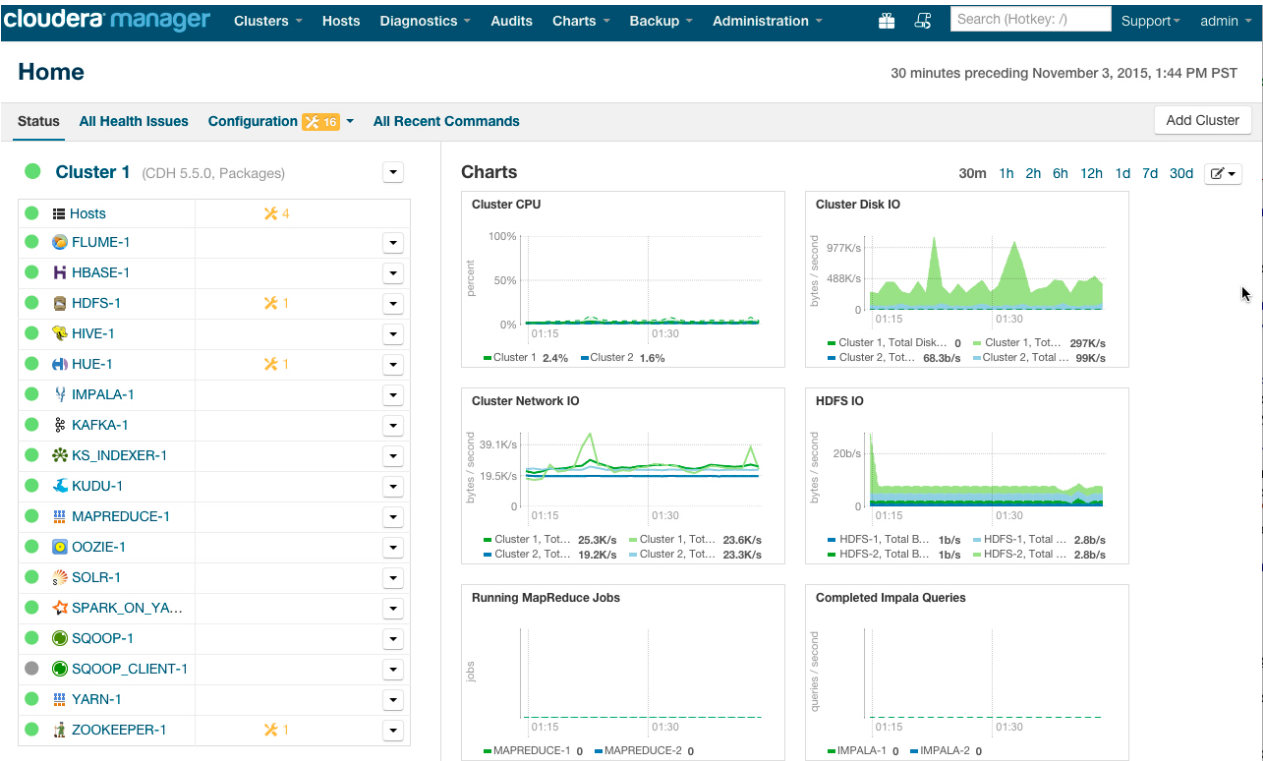
The login screen for Cloudera Manager Admin Console displays.

2. Log into Cloudera Manager Admin Console using the [credentials](#) assigned by your administrator. User accounts are assigned [roles](#) that constrain the features available to you.

**Note:** You can configure the Cloudera Manager Admin Console to automatically log out a user after a configurable period of time. See [Automatic Logout](#) on page 55.

Cloudera Manager Admin Console Home Page

When you start the [Cloudera Manager Admin Console](#) on page 50, the **Home > Status** tab displays.



You can also go to the **Home > Status** tab by clicking the Cloudera Manager logo in the top navigation bar.

Status


The Status tab contains:






- **Clusters** - The clusters being managed by Cloudera Manager. Each cluster is displayed either in summary form or in full form depending on the configuration of the **Administration > Settings > Other > Maximum Cluster Count Shown In Full** property. When the number of clusters exceeds the value of the property, only cluster summary information displays.
 - **Summary Form** - A list of links to cluster status pages. Click **Customize** to jump to the **Administration > Settings > Other > Maximum Cluster Count Shown In Full** property.
 - **Full Form** - A separate section for each cluster containing a link to the cluster status page and a table containing links to the Hosts page and the status pages of the services running in the cluster.

Each service row in the table has a menu of actions that you select by clicking



and can contain one or more of the following indicators:

Indicator	Meaning	Description
 2	Health issue	Indicates that the service has at least one health issue. The indicator shows the number of health issues at the highest severity level. If there are Bad

Indicator	Meaning	Description
		<p>health test results, the indicator is red. If there are no Bad health test results, but Concerning test results exist, then the indicator is yellow. No indicator is shown if there are no Bad or Concerning health test results.</p> <div>  Important: If there is one Bad health test result and two Concerning health results, there will be three health issues, but the number will be one. </div> <p>Click the indicator to display the Health Issues pop-up dialog box.</p> <p>By default only Bad health test results are shown in the dialog box. To display Concerning health test results, click the Also show <i>n</i> concerning issue(s) link. Click the link to display the Status page containing with details about the health test result.</p>
✖ 4	Configuration issue	<p>Indicates that the service has at least one configuration issue. The indicator shows the number of configuration issues at the highest severity level. If there are configuration errors, the indicator is red. If there are no errors but configuration warnings exist, then the indicator is yellow. No indicator is shown if there are no configuration notifications.</p> <div>  Important: If there is one configuration error and two configuration warnings, there will be three configuration issues, but the number will be one. </div> <p>Click the indicator to display the Configuration Issues pop-up dialog box.</p> <p>By default only notifications at the Error severity level are listed, grouped by service name are shown in the dialog box. To display Warning notifications, click the Also show <i>n</i> warning(s) link. Click the message associated with an error or warning to be taken to the configuration property for which the notification has been issued where you can address the issue. See Managing Services.</p>
 Restart Needed  Refresh Needed	Configuration modified	<p>Indicates that at least one of a service's roles is running with a configuration that does not match the current configuration settings in Cloudera Manager.</p> <p>Click the indicator to display the Stale Configurations page. To bring the cluster up-to-date, click the Refresh or Restart button on the Stale Configurations page or follow the instructions in Refreshing a Cluster, Restarting a Cluster, or Restarting Services and Instances after Configuration Changes.</p>
	Client configuration redeployment required	<p>Indicates that the client configuration for a service should be redeployed.</p> <p>Click the indicator to display the Stale Configurations page. To bring the cluster up-to-date, click the Deploy Client Configuration button on the Stale Configurations page or follow the instructions in Manually Redeploying Client Configuration Files.</p>

- **Cloudera Management Service** - A table containing a link to the Cloudera Manager Service. The Cloudera Manager Service has a menu of actions that you select by clicking



- **Charts** - A set of charts ([dashboard](#)) that summarize resource utilization (IO, CPU usage) and processing metrics.

Click a line, stack area, scatter, or bar chart to expand it into a full-page view with a legend for the individual charted entities as well more fine-grained axes divisions.

By default the time scale of a dashboard is 30 minutes. To change the time scale, click a duration link

[30m](#) [1h](#) [2h](#) [6h](#) [12h](#) [1d](#) [7d](#) [30d](#) at the top-right of the dashboard.

To set the dashboard type, click  and select one of the following:

- **Custom** - displays a custom dashboard.
- **Default** - displays a default dashboard.
- **Reset** - resets the custom dashboard to the predefined set of charts, discarding any customizations.

All Health Issues


Displays all health issues by cluster. The number badge has the same semantics as the per service health issues reported on the Status tab.

- By default only Bad health test results are shown in the dialog box. To display Concerning health test results, click the **Also show *n* concerning issue(s)** link.
- To group the health test results by entity or health test, click the buttons on the **Organize by Entity/Organize by Health Test** switch.
- Click the link to display the Status page containing with details about the health test result.

All Configuration Issues

Displays all configuration issues by cluster. The number badge has the same semantics as the per service configuration issues reported on the Status tab. By default only notifications at the Error severity level are listed, grouped by service name are shown in the dialog box. To display Warning notifications, click the **Also show *n* warning(s)** link. Click the message associated with an error or warning to be taken to the configuration property for which the notification has been issued where you can address the issue.

All Recent Commands

Displays all commands run recently across the clusters. A badge  indicates how many recent commands are still running. Click the command link to display details about the command and child commands. See also [Viewing Running and Recent Commands](#).

Starting and Logging into the Cloudera Manager Admin Console

1. In a web browser, enter `http://Server host:7180`, where *Server host* is the FQDN or IP address of the host where the Cloudera Manager Server is running.

The login screen for Cloudera Manager Admin Console displays.

2. Log into Cloudera Manager Admin Console using the [credentials](#) assigned by your administrator. User accounts are assigned [roles](#) that constrain the features available to you.



Note: You can configure the Cloudera Manager Admin Console to automatically log out a user after a configurable period of time. See [Automatic Logout](#) on page 55.

Displaying Cloudera Manager Documentation

To display Cloudera Manager documentation:

1. Open the Cloudera Manager Admin Console.
2. Select **Support > Help, Installation Guide, API Documentation, or Release Notes**. By default, the Help and Installation Guide files from the Cloudera web site are opened. This is because local help files are not updated after installation. You can configure Cloudera Manager to open either the latest Help and Installation Guide from the Cloudera web site (this option requires Internet access from the browser) or locally-installed Help and Installation

Guide by configuring the **Administration > Settings > Support > Open latest Help files from the Cloudera website** property.

Displaying the Cloudera Manager Server Version and Server Time

To display the version, build number, and time for the Cloudera Manager Server:

1. Open the Cloudera Manager Admin Console.
2. Select **Support > About**.

Automatic Logout

For security purposes, Cloudera Manager automatically logs out a user session after 30 minutes. You can change this session logout period.

To configure the timeout period:

1. Click **Administration > Settings**.
2. Click **Category > Security**.
3. Edit the **Session Timeout** property.
4. Click **Save Changes** to commit the changes.

When the timeout is one minute from triggering, the user sees the following message:

Automatic Logout for Your Protection



Due to inactivity, your current work session is about to expire. For your security, Cloudera Manager sessions automatically end after 30 minutes of inactivity.

Your current session will expire in **1 minute**.
Press any key or click anywhere to continue.

If the user does not click the mouse or press a key, the user is logged out of the session and the following message appears:

Automatic Log Out Due to Inactivity

You are now logged out of your account.

We hadn't heard from you for about 30 minute(s), so for your security Cloudera Manager automatically logged you out of your account. Log back in below to continue.

☐ Remember me

EMC DSSD D5 Storage Appliance Integration for Hadoop DataNodes

Overview of EMC DSSD D5 Integration

The EMC DSSD D5 provides a high-speed, low-latency storage solution based on flash media. It has been optimized for use as storage for DataNodes in the Cloudera CDH distribution. The DataNode hosts connect directly to the DSSD D5 using a PCIe card interface. In a CDH cluster, only the DataNodes use the DSSD D5 for storage; all other hosts use standard disks.

To manage clusters that use DSSD D5 storage, enable **DSSD Mode** in Cloudera Manager. All other Hadoop components operate normally. When this mode is enabled, Cloudera Manager can only manage clusters with DSSD D5 DataNodes; you cannot mix cluster types (a cluster that uses only DSSD D5 DataNodes and a cluster that uses regular DataNodes). All DataNodes must connect to the DSSD D5; you cannot mix DataNode types within a cluster.

You can connect multiple instances of a DSSD D5 appliance to a single cluster by defining each DSSD D5 as a "rack." See [Configuring Multiple DSSD D5 Appliances in a Cluster](#).

Installing CDH with DSSD DataNodes

Use Cloudera Manager to install a DSSD D5-enabled cluster. You can install Cloudera Manager in several ways, and you can use Cloudera Manager to install agents and other software on all hosts in your cluster. Installing CDH with DSSD D5 DataNodes is similar to non-DSSD D5 installation, except for the following:

- You cannot install a DSSD D5 cluster using a Cloudera Manager instance that is already managing a cluster.
- You set a single property to enable **DSSD Mode**.
- You set several DSSD D5-specific properties.
- When installing CDH and other services from Cloudera Manager, only *parcel* installations are supported. *Package* installations are not supported. See [Managing Software Installation Using Cloudera Manager](#).

See [Installation and Upgrade with the EMC DSSD D5](#) for complete installation instructions.

Cloudera Manager API

The Cloudera Manager API provides configuration and service lifecycle management, service health information and metrics, and allows you to configure Cloudera Manager itself. The API is served on the same host and port as the [Cloudera Manager Admin Console](#) on page 50, and does not require an extra process or extra configuration. The API supports HTTP Basic Authentication, accepting the same users and credentials as the Cloudera Manager Admin Console.

Resources

- [Quick Start](#)
- [Cloudera Manager API tutorial](#)
- [Cloudera Manager API documentation](#)
- [Python client](#)
- [Using the Cloudera Manager Java API for Cluster Automation](#) on page 59

Obtaining Configuration Files

1. Obtain the list of a service's roles:

```
http://cm_server_host:7180/api/v14/clusters/clusterName/services/serviceName/roles
```

2. Obtain the list of configuration files a process is using:

```
http://cm_server_host:7180/api/v14/clusters/clusterName/services/serviceName/roles/roleName/process
```


3. Obtain the content of any particular file:

```
http://cm_server_host:7180/api/v14/clusters/clusterName/services/serviceName/roles/roleName/process/
configFiles/configFileName
```

For example:

```
http://cm_server_host:7180/api/v14/clusters/Cluster%201/services/OOZIE-1/roles/
OOZIE-1-OOZIE_SERVER-e121641328fcb107999f2b5fd856880d/process/configFiles/oozie-site.xml
```

Retrieving Service and Host Properties

To update a service property using the Cloudera Manager APIs, you'll need to know the name of the property, not just the display name. If you know the property's display name but not the property name itself, retrieve the documentation by requesting any configuration object with the query string `view=FULL` appended to the URL. For example:

```
http://cm_server_host:7180/api/v14/clusters/Cluster%201/services/service_name/config?view=FULL
```

Search the results for the display name of the desired property. For example, a search for the display name **HDFS Service Environment Advanced Configuration Snippet (Safety Valve)** shows that the corresponding property name is `hdfs_service_env_safety_valve`:

```
{
  "name" : "hdfs_service_env_safety_valve",
  "require" : false,
  "displayName" : "HDFS Service Environment Advanced Configuration Snippet (Safety
Valve)",
  "description" : "For advanced use only, key/value pairs (one on each line) to be
inserted into a roles
environment. Applies to configurations of all roles in this service except client
configuration.",
  "relatedName" : "",
  "validationState" : "OK"
}
```

Similar to finding service properties, you can also find host properties. First, get the host IDs for a cluster with the URL:

```
http://cm_server_host:7180/api/v14/hosts
```

This should return host objects of the form:

```
{
  "hostId" : "2c2e951c-aaf2-4780-a69f-0382181f1821",
  "ipAddress" : "10.30.195.116",
  "hostname" : "cm_server_host",
  "rackId" : "/default",
  "hostUrl" :
"http://cm_server_host:7180/cm/hosts/hostRedirect/2c2e951c-adf2-4780-a69f-0382181f1821",
  "maintenanceMode" : false,
  "maintenanceOwners" : [ ],
  "commissionState" : "COMMISSIONED",
  "numCores" : 4,
  "totalPhysMemBytes" : 10371174400
}
```

Then obtain the host properties by including one of the returned host IDs in the URL:

```
http://cm_server_host:7180/api/v14/hosts/2c2e951c-adf2-4780-a69f-0382181f1821?view=FULL
```

Backing Up and Restoring the Cloudera Manager Configuration

You can use the Cloudera Manager REST API to export and import all of its configuration data. The API exports a JSON document that contains configuration data for the Cloudera Manager instance. You can use this JSON document to back up and restore a Cloudera Manager deployment.

Minimum Required Role: [Cluster Administrator](#) (also provided by **Full Administrator**)

Exporting the Cloudera Manager Configuration

1. Log in to the Cloudera Manager server host as the `root` user.
2. Run the following command:

```
# curl -u admin_username:admin_pass "http://cm_server_host:7180/api/v14/cm/deployment" > path_to_file/cm-deployment.json
```

Where:

- `admin_username` is a username with either the Full Administrator or Cluster Administrator role.
- `admin_pass` is the password for the `admin_username` username.
- `cm_server_host` is the hostname of the Cloudera Manager server.
- `path_to_file` is the path to the file where you want to save the configuration.

Redacting Sensitive Information from the Exported Configuration

The exported configuration may contain passwords and other sensitive information. You can configure redaction of the sensitive items by specifying a JVM parameter for Cloudera Manager. When you set this parameter, API calls to Cloudera Manager for configuration data do not include the sensitive information.



Important: If you configure this redaction, you cannot use an exported configuration to restore the configuration of your cluster due to the redacted information.

To configure redaction for the API:

1. Log in the Cloudera Manager server host.
2. Edit the `/etc/default/cloudera-scm-server` file by adding the following property (separate each property with a space) to the line that begins with `export CMF_JAVA_OPTS`:

```
-Dcom.cloudera.api.redaction=true
```

For example:

```
export CMF_JAVA_OPTS="-Xmx2G -Dcom.cloudera.api.redaction=true"
```

3. Restart Cloudera Manager:

```
$ sudo service cloudera-scm-server restart
```

Restoring the Cloudera Manager Configuration



Important: This feature is available only with a Cloudera Enterprise license. It is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see [Managing Licenses](#).

Using a previously saved JSON document that contains the Cloudera Manager configuration data, you can restore that configuration to a running cluster.

1. Using the Cloudera Manager Administration Console, stop all running services in your cluster:

a. On the **Home > Status** tab, click



to the right of the cluster name and select **Stop**.

b. Click **Stop** in the confirmation screen. The **Command Details** window shows the progress of stopping services.

When **All services successfully stopped** appears, the task is complete and you can close the **Command Details** window.



Warning: If you do not stop the cluster before making this API call, the API call will stop *all* cluster services before running the job. Any running jobs and data are lost.

2. Log in to the Cloudera Manager server host as the `root` user.

3. Run the following command:

```
# curl --upload-file path_to_file/cm-deployment.json
-u admin_username:admin_pass
http://cm_server_host:7180/api/v14/cm/deployment?deleteCurrentDeployment=true
```

Where:

- `admin_username` is a username with either the Full Administrator or Cluster Administrator role.
- `admin_pass` is the password for the `admin_username` username.
- `cm_server_host` is the hostname of the Cloudera Manager server.
- `path_to_file` is the path to the file containing the JSON configuration file.

Using the Cloudera Manager Java API for Cluster Automation

One of the complexities of Apache Hadoop is the need to deploy clusters of servers, potentially on a regular basis. If you maintain hundreds of test and development clusters in different configurations, this process can be complex and cumbersome if not automated.

Cluster Automation Use Cases

Cluster automation is useful in various situations. For example, you might work on many versions of CDH, which works on a wide variety of OS distributions (RHEL 5 and RHEL 6, Ubuntu Precise and Lucid, Debian Wheezy, and SLES 11). You might have complex configuration combinations—highly available HDFS or simple HDFS, Kerberized or non-secure, YARN or MRv1, and so on. With these requirements, you need an easy way to create a new cluster that has the required setup. This cluster can also be used for integration, testing, customer support, demonstrations, and other purposes.

You can install and configure Hadoop according to precise specifications using the Cloudera Manager [REST API](#). Using the API, you can add hosts, install CDH, and define the cluster and its services. You can also tune heap sizes, set up HDFS HA, turn on Kerberos security and generate keytabs, and customize service directories and ports. Every configuration available in Cloudera Manager is exposed in the API.

The API also provides access to management functions:

- Obtaining logs and monitoring the system
- Starting and stopping services
- Polling cluster events
- Creating a disaster recovery replication schedule

For example, you can use the API to retrieve logs from HDFS, HBase, or any other service, without knowing the log locations. You can also stop any service with no additional steps.

Use scenarios for the Cloudera Manager API for cluster automation might include:

- OEM and hardware partners that deliver Hadoop-in-a-box appliances using the API to set up CDH and Cloudera Manager on bare metal in the factory.
- Automated deployment of new clusters, using a combination of Puppet and the Cloudera Manager API. Puppet does the OS-level provisioning and installs the software. The Cloudera Manager API sets up the Hadoop services and configures the cluster.
- Integrating the API with reporting and alerting infrastructure. An external script can poll the API for health and metrics information, as well as the stream of events and alerts, to feed into a custom dashboard.

Java API Examples

This example covers the Java API client.

To use the Java client, add this dependency to your project's `pom.xml`:

```
<project>
  <repositories>
    <repository>
      <id>cdh.repo</id>
      <url>https://repository.cloudera.com/groups/cloudera-repos</url>
      <name>Cloudera Repository</name>
    </repository>
    ...
  </repositories>
  <dependencies>
    <dependency>
      <groupId>com.cloudera.api</groupId>
      <artifactId>cloudera-manager-api</artifactId>
      <version>4.6.2</version>      <!-- Set to the version of Cloudera Manager you use -->
    </dependency>
    ...
  </dependencies>
  ...
</project>
```

The Java client works like a proxy. It hides from the caller any details about REST, HTTP, and JSON. The entry point is a handle to the root of the API:

```
RootResourceV14 apiRoot = new ClouderaManagerClientBuilder().withHost("cm.cloudera.com")
    .withUsernamePassword("admin", "admin").build().getRootV14();
```

From the root, you can traverse down to all other resources. (It's called "v14" because that is the current Cloudera Manager API version, but the same builder will also return a root from an earlier version of the API.) The tree view shows some key resources and supported operations:

- `RootResourceV14`
 - `ClustersResourceV14` - host membership, start cluster
 - `ServicesResourceV14` - configuration, get metrics, HA, service commands
 - `RolesResource` - add roles, get metrics, logs
 - `RoleConfigGroupsResource` - configuration
 - `ParcelsResource` - parcel management
 - `HostsResource` - host management, get metrics
 - `UsersResource` - user management

For more information, see the [Javadoc](#).

The following example lists and starts a cluster:

```
// List of clusters
ApiClusterList clusters = apiRoot.getClustersResource().readClusters(DataView.SUMMARY);
```

```

for (ApiClient cluster : clusters) {
    LOG.info("{}: {}", cluster.getName(), cluster.getVersion());
}

// Start the first cluster
ApiClient cmd = apiRoot.getClustersResource().startCommand(clusters.get(0).getName());
while (cmd.isActive()) {
    Thread.sleep(100);
    cmd = apiRoot.getCommandsResource().readCommand(cmd.getId());
}
LOG.info("Cluster start {}", cmd.getSuccess() ? "succeeded" : "failed " +
cmd.getResultMessage());

```

To see a full example of cluster deployment using the Java client, see [whirr-cm](#). Go to `CmServerImpl#configure` to see the relevant code.

Extending Cloudera Manager

In addition to the set of software packages and services managed by Cloudera Manager, you can also define and add new types of services using [custom service descriptors](#). When you deploy a custom service descriptor, the implementation is delivered in a Cloudera Manager [parcel](#) or other software package. For information on the extension mechanisms provided by Cloudera Manager for creating custom service descriptors and parcels, see [Cloudera Manager Extensions](#).

Cloudera Navigator 2 Overview

Cloudera Navigator is a fully integrated data-management and security system for the Hadoop platform. Cloudera Navigator enables you to work effectively with data at scale and helps various stakeholders answer the following questions:

- Compliance groups
 - Who accessed the data, and what did they do with it?
 - Are we prepared for an audit?
 - Is our sensitive data protected?
- Hadoop administrators and DBAs
 - How can we boost productivity and cluster performance?
 - How is data being used?
 - How can data be optimized for future workloads?
- Data stewards and curators
 - How can data assets be managed and organized?
 - What is the lifecycle of the data?
- Data scientists and Business Intelligence users
 - Where is the most important data?
 - Is this data trustworthy?
 - What is the relationship between data sets?

Cloudera Navigator provides the following components to help you answer these questions and meet data-management and security requirements.

- **Data Management** - Provides visibility into and control over the data in Hadoop datastores, and the computations performed on that data. Hadoop administrators, data stewards, and data scientists can use Cloudera Navigator to:
 - Audit data access and verify access privileges - The goal of auditing is to capture a complete and immutable record of all activity within a system. Cloudera Navigator [auditing](#) adds secure, real-time audit components to key data and access frameworks. Compliance groups can use Cloudera Navigator to configure, collect, and view audit events that show who accessed data, and how.
 - Search metadata and visualize lineage - Cloudera Navigator [metadata management](#) allows DBAs, data stewards, business analysts, and data scientists to define, search for, amend the properties of, and tag data entities and view relationships between datasets.
 - Policies - Data stewards can use Cloudera Navigator [policies](#) to define automated actions, based on data access or on a schedule, to add metadata, create alerts, and move or purge data.
 - Analytics - Hadoop administrators can use Cloudera Navigator [analytics](#) to examine data usage patterns and create policies based on those patterns.
- **Data Encryption** - Data encryption and key management provide a critical layer of protection against potential threats by malicious actors on the network or in the datacenter. Encryption and key management are also requirements for meeting key compliance initiatives and ensuring the integrity of your enterprise data. The following Cloudera Navigator components enable compliance groups to manage encryption:
 - [Cloudera Navigator Encrypt](#) transparently encrypts and secures data at rest without requiring changes to your applications and ensures there is minimal performance lag in the encryption or decryption process.
 - [Cloudera Navigator Key Trustee Server](#) is an enterprise-grade virtual safe-deposit box that stores and manages cryptographic keys and other security artifacts.

- [Cloudera Navigator Key HSM](#) allows Cloudera Navigator Key Trustee Server to seamlessly integrate with a hardware security module (HSM).

You can install Cloudera Navigator data management and data encryption components independently.

Related Information

- [Installing the Cloudera Navigator Data Management Component](#)
- [Upgrading the Cloudera Navigator Data Management Component](#)
- [Cloudera Navigator Data Management Component Administration](#)
- [Cloudera Data Management](#)
- [Configuring Authentication in the Cloudera Navigator Data Management Component](#)
- [Configuring TLS/SSL for the Cloudera Navigator Data Management Component](#)
- [Cloudera Navigator Data Management Component User Roles](#)

Cloudera Navigator Data Management Overview

The section describes basic features of Cloudera Navigator data management.

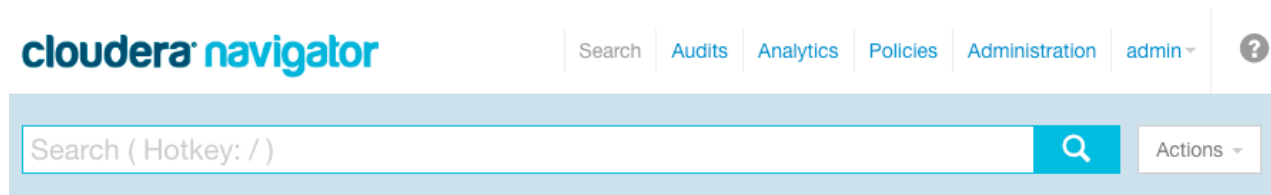
Cloudera Navigator Data Management UI

Managing data in your cluster presents a number of challenges:

- The volume of data can be very large.
- Multiple people can access that data. How do you know who is access what data, and how it is being used?
- Tracking data can be challenging. Where did the data originate? Has it been altered? In how many places is that data being used? In regulated industries, answers to these questions must be precise and provable. For example, how can you make sure that you are complying with industry or government requirements for the deletion or preservation of data?

Cloudera Navigator is designed to help you manage and monitor vast amounts of data in your cluster. You can use the Cloudera Navigator data management UI to:

- Create and view audit reports
- Search entity metadata, view entity lineage, and modify custom metadata
- Define policies for modifying custom metadata and sending notifications when entities are extracted
- View metadata analytics
- Assign user roles to groups



Navigator auditing, metadata, lineage, policies, and analytics all support multi-cluster deployments managed by a single Cloudera Manager instance. So if you have five clusters, all centrally managed by a single Cloudera Manager, you see all this information in a single Navigator data management UI. In the metadata part of the UI, Navigator provides technical metadata that tracks the specific cluster from which the data is derived.

Starting and Logging into the Cloudera Navigator Data Management UI

1. Do one of the following:

- Enter the URL of the Navigator UI in a browser: `http://Navigator_Metadata_Server_host:port/`, where `Navigator_Metadata_Server_host` is the name of the host on which you are running the Navigator

Metadata Server role and *port* is the port configured for the role. The default port of the Navigator Metadata Server is 7187. To change the port, follow the instructions in [Configuring the Navigator Metadata Server Port](#).

- Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Navigator**.
 - Navigate from the Navigator Metadata Server role:
 1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the **Home > Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
 2. Click the **Instances** tab.
 3. Click the **Navigator Metadata Server** role.
 4. Click the **Cloudera Navigator** link.
- 2. Log into Cloudera Navigator UI using the [credentials](#) assigned by your administrator.


Cloudera Navigator Data Management API

The Cloudera Navigator data management API provides access to the same features as the UI.

The API available at `http://Navigator_Metadata_Server_host:port/api/v9`, where *Navigator_Metadata_Server_host* is the name of the host on which you are running the Navigator Metadata Server role and *port* is the port configured for the role. The default port of the Navigator Metadata Server is 7187. To change the port, follow the instructions in [Configuring the Navigator Metadata Server Port](#). The API supports HTTP Basic Authentication, accepting the same users and credentials as the UI.

To get a listing of the API calls invoked by the UI, see [Downloading a Debug File](#) on page 65.

Accessing API Documentation

For API documentation, select  > **API Documentation** or go to `Navigator_Metadata_Server_host:port/api-console/index.html`. The Cloudera Navigator API documentation displays in a new window. The API is structured into resource categories. Click a category to display the resource endpoints.

To view an API tutorial, click the **Tutorial** link at the top of the API documentation or go to `Navigator_Metadata_Server_host:port/api-console/tutorial.html`

Capturing and Downloading API Calls

To capture API calls made from the Cloudera Navigator data management UI, enable debug mode. You can then download a file containing the captured calls and send it to Cloudera.

Enabling and Disabling Debug Mode

To enable debug mode:

1. [Start and log into the Cloudera Navigator data management component UI](#).
2. In the top right, select **username > Enable Debug Mode**. A red box with the following message displays at the bottom right of the UI.

Debug mode enabled. Captured 0 calls.

3. Reload the page so that all API calls are captured.

To disable debug mode, do one of the following:

- In the top right, select **username > Disable Debug Mode**.
- Click **Disable** in the red box at the bottom right of the UI.

The red box at the bottom right of the UI disappears.


Downloading a Debug File

In debug mode, the *n* in the string "Captured *n* calls." is incremented with the number of calls of the Cloudera Navigator data management API as you interact with the Cloudera Navigator data management UI. To download a file containing information about the API calls, click **Download debug file**. A file named `api-data-Navigator_Metadata_Server_host-UTC timestamp.json` is downloaded. For example:

```
{
  "href": "http://Navigator Metadata Server
hostname:port/?view=detailsView&id=7f44221738670c98baf0799aa6abd330&activeView=lineage&b=ImMka",
  "userAgent": ...
  "windowSize": ...
},
{
  "timestamp": 1456795776671,
  "calls": [
    {
      "type": "POST",
      "url": "/api/v6/interactive/entities?limit=0&offset=0",
      "data": ...,
      "page": "http://Navigator Metadata Server
hostname:port/?view=resultsView&facets=%7B%22type%22%3A%5B%22database%22%5D%7D",
      "timestamp": 1456795762472
    },
    {
      "type": "GET",
      "url": "/api/v3/entities?query=type%3Asource",
      "status": 200,
      "responseText": ...,
      "page": "http://Navigator Metadata Server
hostname:port/?view=resultsView&facets=%7B%22type%22%3A%5B%22database%22%5D%7D",
      "timestamp": 1456795763233
    },
    ...
  ]
}
```

Displaying Cloudera Navigator Data Management Documentation

To display Cloudera Navigator data management documentation:

1. [Start and log into the Cloudera Navigator data management component UI.](#)
2. Select  > **Help**. The Cloudera Navigator data management online documentation displays in a new window.

Displaying the Cloudera Navigator Data Management Component Version

To display the version and build number for the Cloudera Navigator data management component:

1. [Start and log into the Cloudera Navigator data management component UI.](#)
2. Select  > **About**.

Cloudera Navigator Data Encryption Overview



Warning: Encryption transforms coherent data into random, unrecognizable information for unauthorized users. It is *absolutely critical* that you follow the documented procedures for encrypting and decrypting data, and that you regularly back up the encryption keys and configuration files. Failure to do so can result in irretrievable data loss. See [Backing Up and Restoring Key Trustee Server and Clients](#) for more information.

Do not attempt to perform any operations that you do not understand. If you have any questions about a procedure, contact Cloudera Support before proceeding.

Cloudera Navigator includes a turnkey encryption and key management solution for data at rest, whether data is stored in HDFS or on the local Linux filesystem. Cloudera Navigator encryption comprises the following components:

- [Cloudera Navigator Key Trustee Server](#)

Key Trustee Server is an enterprise-grade virtual safe-deposit box that stores and manages cryptographic keys. With Key Trustee Server, encryption keys are separated from the encrypted data, ensuring that sensitive data is protected in the event that unauthorized users gain access to the storage media.

- [Cloudera Navigator Key HSM](#)

Key HSM is a service that allows Key Trustee Server to integrate with a hardware security module (HSM). Key HSM enables Key Trustee Server to use an HSM as the root of trust for cryptographic keys, taking advantage of Key Trustee Server's policy-based key and security asset management capabilities while satisfying existing internal security requirements regarding treatment of cryptographic materials.

- [Cloudera Navigator Encrypt](#)

Navigator Encrypt is a client-side service that transparently encrypts data at rest without requiring changes to your applications and with minimal performance lag in the encryption or decryption process. Advanced key management with Key Trustee Server and process-based access controls in Navigator Encrypt enable organizations to meet compliance regulations and ensure unauthorized parties or malicious actors never gain access to encrypted data.

- Key Trustee KMS

For [HDFS Transparent Encryption](#), Cloudera provides Key Trustee KMS, a customized [Keystores and the Hadoop Key Management Server](#) that uses Key Trustee Server for robust and scalable encryption key storage and management instead of the file-based Java KeyStore used by the default Hadoop KMS.

Cloudera Navigator encryption provides:

- High-performance transparent data encryption for files, databases, and applications running on Linux
- Separation of cryptographic keys from encrypted data
- Centralized management of cryptographic keys
- Integration with hardware security modules (HSMs) from Thales and SafeNet
- Support for Intel AES-NI cryptographic accelerator for enhanced performance in the encryption and decryption process
- Process-Based Access Controls

Cloudera Navigator encryption can be deployed to protect different assets, including (but not limited to):

- Databases
- Log files
- Temporary files
- Spill files
- HDFS data

For planning and deployment purposes, this can be simplified to two types of data that Cloudera Navigator encryption can secure:

1. HDFS data
2. Local filesystem data

The following table outlines some common use cases and identifies the services required.

Table 2: Encrypting Data at Rest

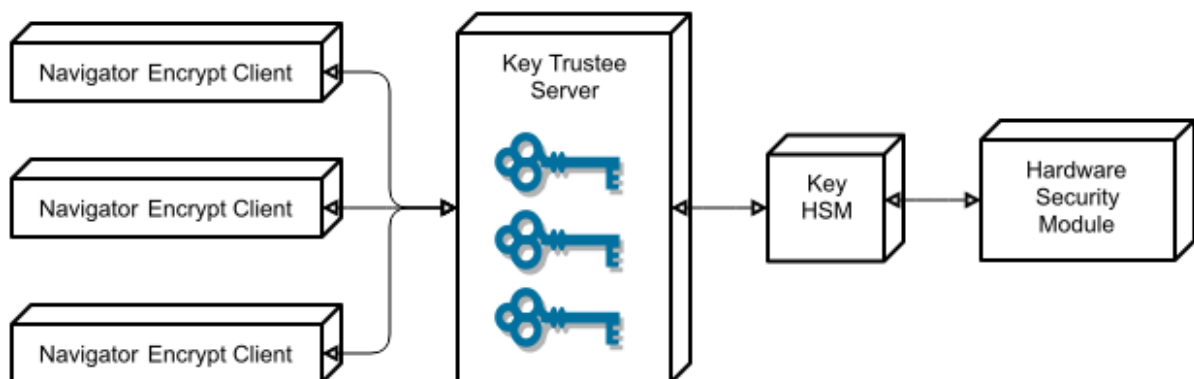
Data Type	Data Location	Key Management	Additional Services Required
HDFS	HDFS	Key Trustee Server	Key Trustee KMS

Data Type	Data Location	Key Management	Additional Services Required
Metadata databases, including: <ul style="list-style-type: none"> Hive Metastore Cloudera Manager Cloudera Navigator Data Management Sentry 	Local filesystem	Key Trustee Server	Navigator Encrypt
Temp/spill files for CDH components with native encryption: <ul style="list-style-type: none"> Impala YARN MapReduce Flume HBase Accumulo 	Local filesystem	N/A (temporary keys are stored in memory only)	None (enable native temp/spill encryption for each component)
Temp/spill files for CDH components without native encryption: <ul style="list-style-type: none"> Spark Kafka Sqoop2 HiveServer2 	Local filesystem	Key Trustee Server	Navigator Encrypt
Log files	Local filesystem	Key Trustee Server	Navigator Encrypt Log Redaction

For instructions on using Navigator Encrypt to secure local filesystem data, see [Cloudera Navigator Encrypt](#).

Cloudera Navigator Encryption Architecture

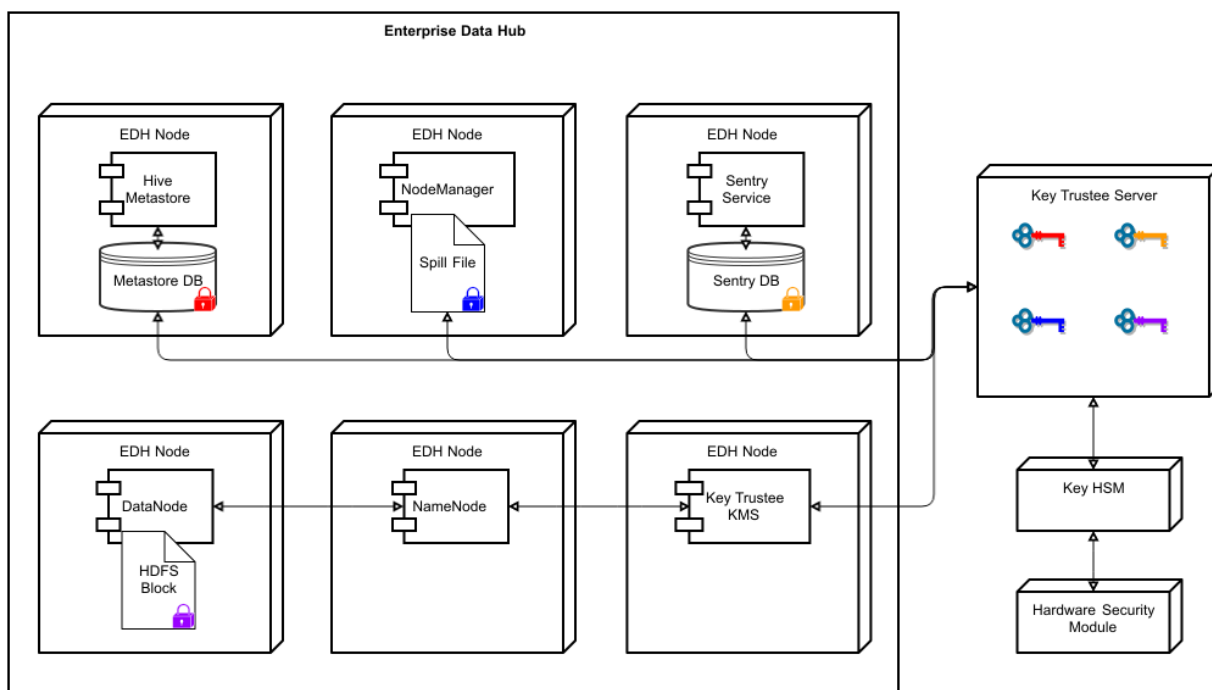
The following diagram illustrates how the Cloudera Navigator encryption components interact with each other:



Key Trustee clients include Navigator Encrypt and Key Trustee KMS. Encryption keys are created by the client and stored in Key Trustee Server.

Cloudera Navigator Encryption Integration with an EDH

The following diagram illustrates how the Cloudera Navigator encryption components integrate with an Enterprise Data Hub (EDH):



For more details on the individual components of Cloudera Navigator encryption, continue reading:

Cloudera Navigator Key Trustee Server Overview

Cloudera Navigator Key Trustee Server is an enterprise-grade virtual safe-deposit box that stores and manages cryptographic keys and other security artifacts. With Navigator Key Trustee Server, encryption keys are separated from the encrypted data, ensuring that sensitive data is still protected if unauthorized users gain access to the storage media.

Key Trustee Server protects these keys and other critical security objects from unauthorized access while enabling compliance with strict data security regulations. For added security, Key Trustee Server can integrate with a [hardware security module](#) (HSM). See [Cloudera Navigator Key HSM Overview](#) on page 69 for more information.

In conjunction with the Key Trustee KMS, Navigator Key Trustee Server can serve as a backing key store for [HDFS Transparent Encryption](#), providing enhanced security and scalability over the file-based Java KeyStore used by the default Hadoop Key Management Server.

Cloudera Navigator Encrypt also uses Key Trustee Server for key storage and management.

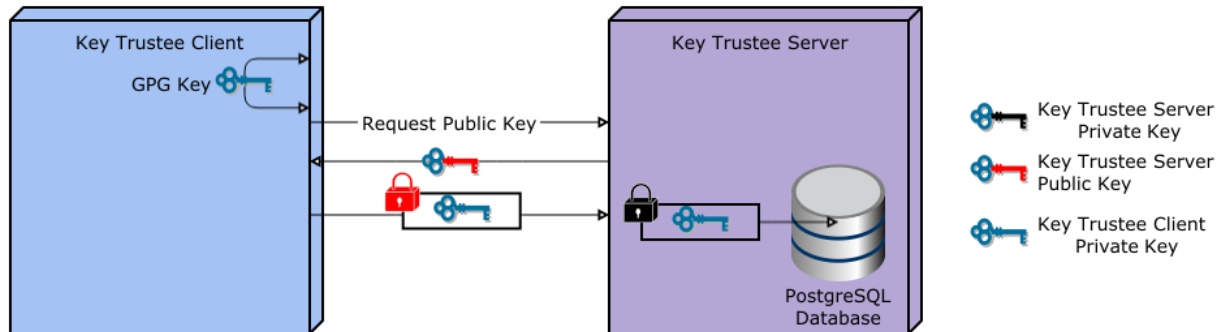
For instructions on installing Navigator Key Trustee Server, see [Installing Cloudera Navigator Key Trustee Server](#). For instructions on configuring Navigator Key Trustee Server, see [Initializing Standalone Key Trustee Server](#) or [Cloudera Navigator Key Trustee Server High Availability](#).

Key Trustee Server Architecture

Key Trustee Server is a secure object store. Clients register with Key Trustee Server, and are then able to store and retrieve objects with Key Trustee Server. The most common use case for Key Trustee Server is storing encryption keys to simplify key management and enable compliance with various data security regulations, but Key Trustee Server is agnostic about the actual objects being stored.

All interactions with Key Trustee Server occur over a TLS-encrypted HTTPS connection.

Key Trustee Server does not generate encryption keys for clients. Clients generate encryption keys, encrypt them with their private key, and send them over a TLS-encrypted connection to the Key Trustee Server. When a client needs to decrypt data, it retrieves the appropriate encryption key from Key Trustee Server and caches it locally to improve performance. This process is demonstrated in the following diagram:



The most common Key Trustee Server clients are Navigator Encrypt and Key Trustee KMS.

When a Key Trustee client registers with Key Trustee Server, it generates a unique fingerprint. All client interactions with the Key Trustee Server are authenticated with this fingerprint. You must ensure that the file containing this fingerprint is secured with appropriate Linux file permissions. The file containing the fingerprint is `/etc/navencrypt/keytrustee/ztrustee.conf` for Navigator Encrypt clients, and `/var/lib/kms-keytrustee/keytrustee/.keytrustee/keytrustee.conf` for Key Trustee KMS.

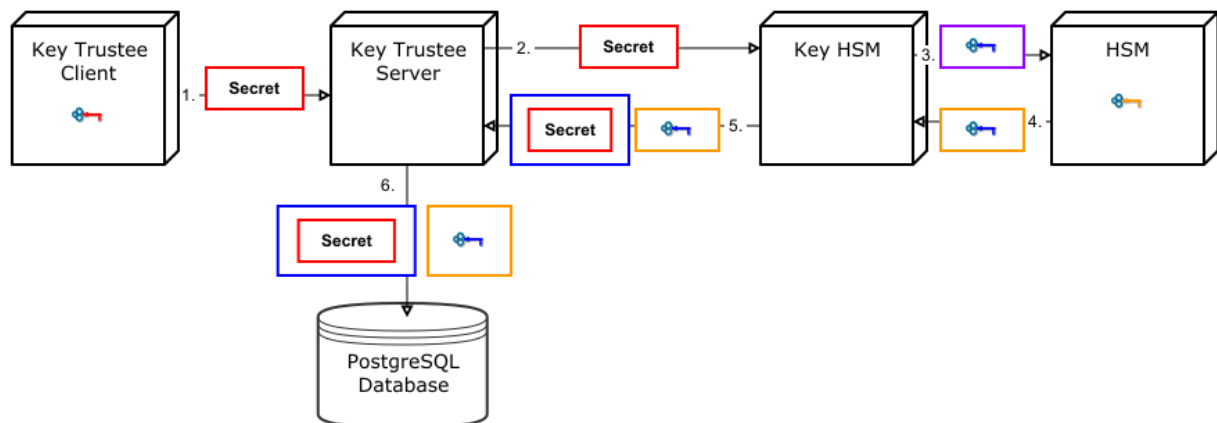
Many clients can use the same Key Trustee Server to manage security objects. For example, you can have several Navigator Encrypt clients using a Key Trustee Server, and also use the same Key Trustee Server as the backing store for Key Trustee KMS (used in HDFS encryption).

Cloudera Navigator Key HSM Overview

Cloudera Navigator Key HSM allows Cloudera Navigator Key Trustee Server to seamlessly integrate with a hardware security module (HSM). Key HSM enables Key Trustee Server to use an HSM as a root of trust for cryptographic keys, taking advantage of Key Trustee Server's policy-based key and security asset management capabilities while satisfying existing, internal security requirements for treatment of cryptographic materials.

Key HSM adds an additional layer of encryption to Key Trustee Server deposits, and acts as a root of trust. If a key is revoked on the HSM, any Key Trustee Server deposits encrypted with that key are rendered irretrievable.

The following diagram demonstrates the flow of storing a deposit in Key Trustee Server when Key HSM is used:



Cloudera Navigator 2 Overview

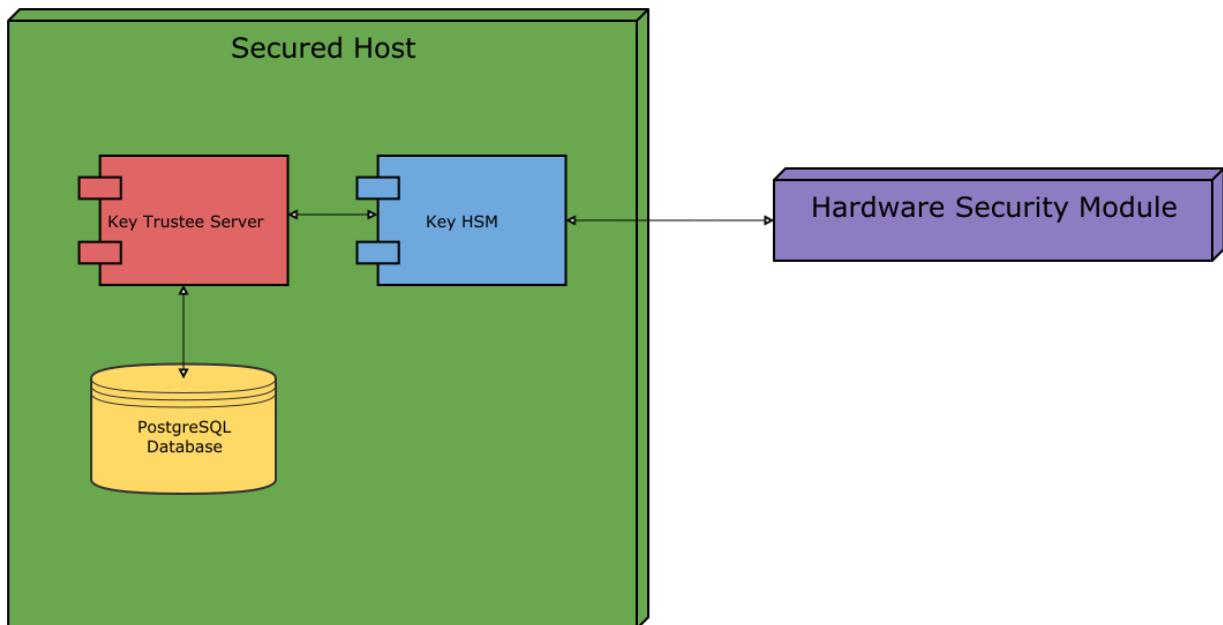
1. A Key Trustee client (for example, Navigator Encrypt or Key Trustee KMS) sends an encrypted secret to Key Trustee Server.
2. Key Trustee Server forwards the encrypted secret to Key HSM.
3. Key HSM generates a symmetric encryption key and sends it to the HSM over an encrypted channel.
4. The HSM generates a new key pair and encrypts the symmetric key and returns the encrypted symmetric key to Key HSM.
5. Key HSM encrypts the original client-encrypted secret with the symmetric key, and returns the twice-encrypted secret, along with the encrypted symmetric key, to Key Trustee Server. Key HSM discards its copy of the symmetric key.
6. Key Trustee Server stores the twice-encrypted secret along with the encrypted symmetric key in its PostgreSQL database.

The only way to retrieve the original encrypted secret is for Key HSM to request the HSM to decrypt the encrypted symmetric key, which is required to decrypt the twice-encrypted secret. If the key has been revoked on the HSM, it is not possible to retrieve the original secret.

Key HSM Architecture

For increased security, Key HSM should always be installed on the same host running the Key Trustee Server. This reduces the attack surface of the system by ensuring that communication between Key Trustee Server and Key HSM stays on the same host, and never has to traverse a network segment.

The following diagram displays the recommended architecture for Key HSM:



For instructions on installing Navigator Key HSM, see [Installing Cloudera Navigator Key HSM](#). For instructions on configuring Navigator Key HSM, see [Initializing Navigator Key HSM](#).

Cloudera Navigator Encrypt Overview

Cloudera Navigator Encrypt transparently encrypts and secures data at rest without requiring changes to your applications and ensures minimal performance lag in the encryption or decryption process. Advanced key management with [Cloudera Navigator Key Trustee Server](#) and process-based access controls in Navigator Encrypt enable organizations to meet compliance regulations and prevent unauthorized parties or malicious actors from gaining access to encrypted data.

For instructions on installing Navigator Encrypt, see [Installing Cloudera Navigator Encrypt](#). For instructions on configuring Navigator Encrypt, see [Registering Cloudera Navigator Encrypt with Key Trustee Server](#).

Navigator Encrypt features include:

- Automatic key management: Encryption keys are stored in Key Trustee Server to separate the keys from the encrypted data. If the encrypted data is compromised, it is useless without the encryption key.
- Transparent encryption and decryption: Protected data is encrypted and decrypted seamlessly, with minimal performance impact and no modification to the software accessing the data.
- Process-based access controls: Processes are authorized individually to access encrypted data. If the process is modified in any way, access is denied, preventing malicious users from using customized application binaries to bypass the access control.
- Performance: Navigator Encrypt supports the Intel AES-NI cryptographic accelerator for enhanced performance in the encryption and decryption process.
- Compliance: Navigator Encrypt enables you to comply with requirements for HIPAA-HITECH, PCI-DSS, FISMA, EU Data Protection Directive, and other data security regulations.
- Multi-distribution support: Navigator Encrypt supports Debian, Ubuntu, RHEL, CentOS, and SLES.
- Simple installation: Navigator Encrypt is distributed as RPM and DEB packages, as well as SLES KMPs.
- Multiple mountpoints: You can separate data into different mountpoints, each with its own encryption key.

Navigator Encrypt can be used with many kinds of data, including (but not limited to):

- Databases
- Temporary files (YARN containers, spill files, and so on)
- Log files
- Data directories
- Configuration files

Navigator Encrypt uses `dmccrypt` for its underlying cryptographic operations. Navigator Encrypt uses several different encryption keys:

- Master Key: The master key can be a single passphrase, dual passphrase, or RSA key file. The master key is stored in Key Trustee Server and cached locally. This key is used when registering with a Key Trustee Server and when performing administrative functions on Navigator Encrypt clients.
- Mount Encryption Key (MEK): This key is generated by Navigator Encrypt using `openssl rand` by default, but it can alternatively use `/dev/urandom`. This key is generated when preparing a new mount point. Each mount point has its own MEK. This key is uploaded to Key Trustee Server.
- `dmccrypt` Device Encryption Key (DEK): This key is not managed by Navigator Encrypt or Key Trustee Server. It is managed locally by `dmccrypt` and stored in the header of the device.

Process-Based Access Control List

The access control list (ACL) controls access to specified data. The ACL uses a process fingerprint, which is the SHA256 hash of the process binary, for authentication. You can create rules to allow a process to access specific files or directories. The ACL file is encrypted with the client master key and stored locally for quick access and updates.

Here is an example rule:

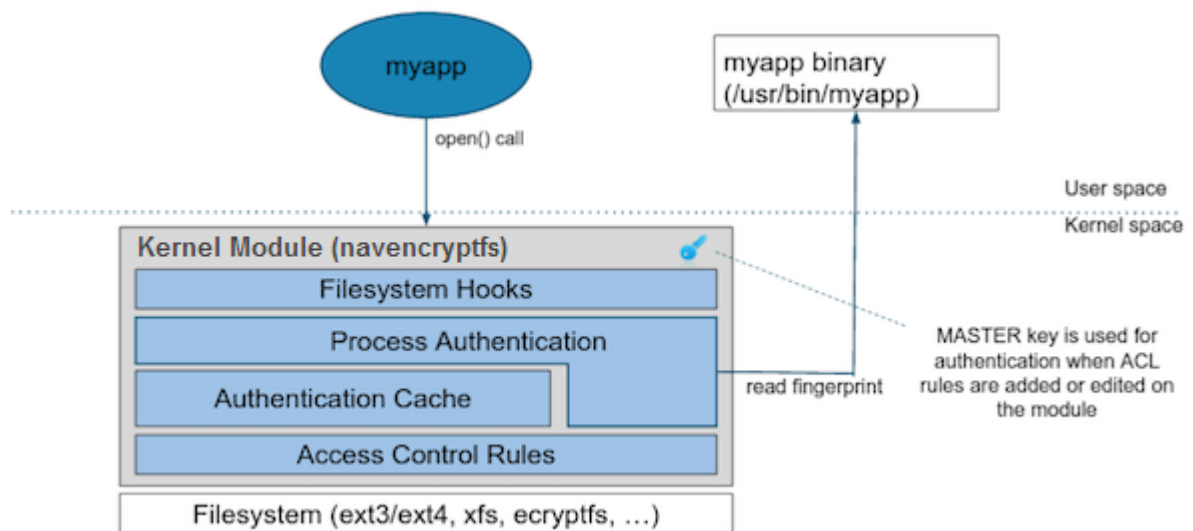
```
"ALLOW @mydata * /usr/bin/myapp"
```

This rule allows the `/usr/bin/myapp` process to access any encrypted path (*) that was encrypted under the category `@mydata`.

Navigator Encrypt uses a kernel module that intercepts any input/output (I/O) sent to an encrypted and managed path. The Linux module filename is `navencryptfs.ko` and it resides in the kernel stack, injecting filesystem hooks. It also authenticates and authorizes processes and caches authentication results for increased performance.

Because the kernel module intercepts and does not modify I/O, it supports any filesystem (`ext3`, `ext4`, `xfs`, and so on).

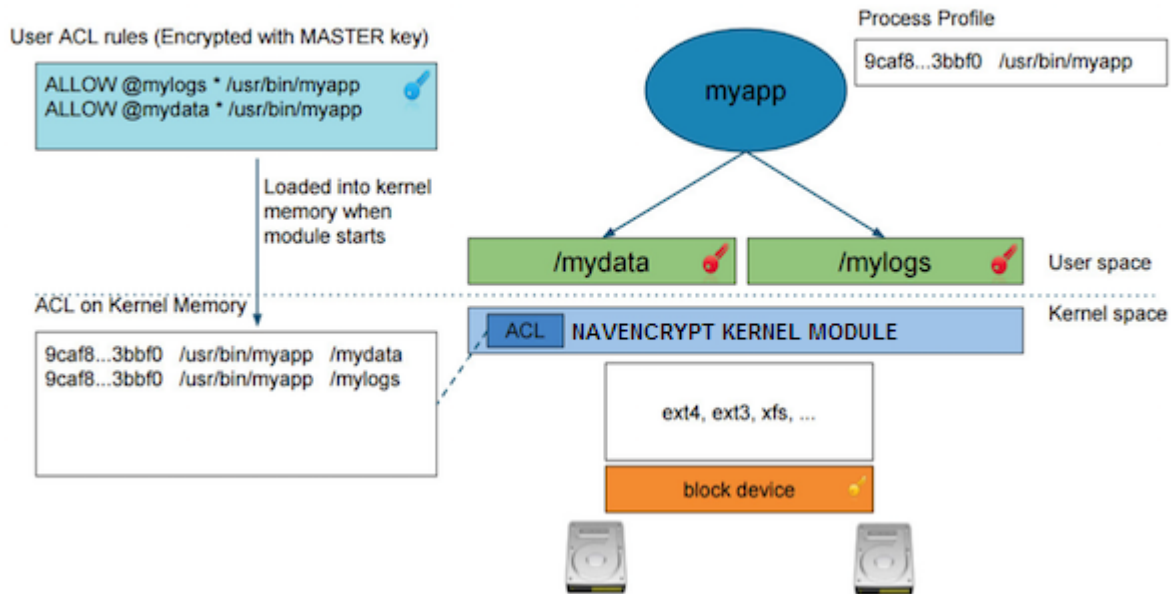
The following diagram shows `/usr/bin/myapp` sending an `open()` call that is intercepted by `navencrypt-kernel-module` as an open hook:



The kernel module calculates the process fingerprint. If the authentication cache already has the fingerprint, the process is allowed to access the data. If the fingerprint is not in the cache, the fingerprint is checked against the ACL. If the ACL grants access, the fingerprint is added to the authentication cache, and the process is permitted to access the data.

When you add an ACL rule, you are prompted for the master key. If the rule is accepted, the ACL rules file is updated as well as the `navencrypt-kernel-module` ACL cache.

The next diagram illustrates different aspects of Navigator Encrypt:



The user adds a rule to allow `/usr/bin/myapp` to access the encrypted data in the category `@mylogs`, and adds another rule to allow `/usr/bin/myapp` to access encrypted data in the category `@mydata`. These two rules are loaded into the `navencrypt-kernel-module` cache after restarting the kernel module.

The `/mydata` directory is encrypted under the `@mydata` category and `/mylogs` is encrypted under the `@mylogs` category using `dmccrypt` (block device encryption).

When `myapp` tries to issue I/O to an encrypted directory, the kernel module calculates the fingerprint of the process (`/usr/bin/myapp`) and compares it with the list of authorized fingerprints in the cache.

Encryption Key Storage and Management

The master key and mount encryption keys are securely deposited in Key Trustee Server. One MEK per mount point is stored locally for offline recovery and rapid access. The locally-stored MEKs are encrypted with the master key.

The connection between Navigator Encrypt and Key Trustee Server is secured with TLS/SSL certificates.

The following diagram demonstrates the communication process between Navigator Encrypt and Key Trustee Server:



The master key is encrypted with a local GPG key. Before being stored in the Key Trustee Server database, it is encrypted again with the Key Trustee Server GPG key. When the master key is needed to perform a Navigator Encrypt operation, Key Trustee Server decrypts the stored key with its server GPG key and sends it back to the client (in this case, Navigator Encrypt), which decrypts the deposit with the local GPG key.

All communication occurs over TLS-encrypted connections.

Cloudera Navigator Optimizer

Cloudera Navigator Optimizer profiles and analyzes the query text in SQL workloads. This analysis helps you gain in-depth understanding of your workloads, identify and offload queries best-suited for Hadoop, and optimize your workloads already running on Hive or Impala.

Use Navigator Optimizer to

- **Identify workloads to move to Hadoop:** Offloading non-operational enterprise data warehouse (EDW) workloads to Hadoop can help you increase the efficiency of your EDW system.
- **Optimize workloads already running on Hadoop:** Use Navigator Optimizer to see what is happening with your workloads running on Hadoop and get recommendations for optimizing them.
- **Slice and dice workloads by custom attributes:** Add custom attribute information to your query workload file and Navigator Optimizer can pivot its analysis based on those custom attributes. For example, you can identify data access patterns for specific users, applications, or reports.
- **Get risk alerts based on Hive and Impala best practices:** Before you offload a workload to Hive or Impala, Navigator Optimizer can assess the risk based on compatibility and complexity of these queries and suggest fixes for them.

For more information about Navigator Optimizer, see www.cloudera.com/products/cloudera-navigator-optimizer.html

Frequently Asked Questions About Cloudera Software

Cloudera Express and Cloudera Enterprise Features

Features available with Cloudera Express and Cloudera Enterprise are summarized in the following table.

Feature	Cloudera Express	Cloudera Enterprise
Cluster Management		
Number of hosts supported	Unlimited	Unlimited
Host inspector for determining CDH readiness	■	■
Multi-cluster management	■	■
Centralized view of all running commands	■	■
Resource management	■	■
Global time control for historical diagnosis	■	■
Cluster-wide configuration	■	■
Cluster-wide event management	■	■
Cluster-wide log search	■	■
Aggregate UI		■
Deployment		
Support for CDH 4 and CDH 5	■	■
Automated deployment and readiness checks	■	■
Installation from local repositories	■	■
Rolling upgrade of CDH		■
Service and Configuration Management		
Manage Accumulo, Flume, HBase, HDFS, Hive, Hue, Impala, Isilon, Kafka, Kudu, MapReduce, Oozie, Sentry, Solr, Spark, Sqoop, YARN, and ZooKeeper services	■	■
Manage Key Trustee and Cloudera Navigator		■
Manage add-on services	■	■
Rolling restart of services		■
High availability (HA) support <ul style="list-style-type: none"> CDH 4 - HDFS and MapReduce JobTracker (CDH 4.2) 	■	■

Frequently Asked Questions About Cloudera Software

Feature	Cloudera Express	Cloudera Enterprise
<ul style="list-style-type: none"> CDH 5 - HDFS, Hive Metastore, Hue, Impala Llama ApplicationMaster, MapReduce JobTracker, Oozie, YARN ResourceManager 		
HBase co-processor support	■	■
Configuration audit trails	■	■
Client configuration management	■	■
Workflows (add, start, stop, restart, delete, and decommission services, hosts, and role instances)	■	■
Role groups	■	■
Host templates	■	■
Configuration versioning and history		■
Restoring a configuration using the API		■
Security		
Kerberos authentication	■	■
LDAP authentication for CDH	■	■
LDAP authentication for Cloudera Manager		■
SAML authentication		■
Encrypted communication between Server and host Agents (TLS)	■	■
Sentry role-based access control	■	■
Password redaction	■	■
Data encryption with KMS		■
Cloudera Manager user roles		■
Monitoring and Diagnostics		
Service, host, and activity monitoring	■	■
Proactive health tests	■	■
Health history	■	■
Advanced filtering and charting of metrics	■	■
Job monitoring for MapReduce jobs, YARN applications, and Impala queries	■	■
Similar activity performance for MapReduce jobs	■	■
Support for terminating activities	■	■
Alert Management		

Feature	Cloudera Express	Cloudera Enterprise
Alert by email	■	■
Alert by SNMP		■
User-defined triggers	■	■
Custom alert publish scripts		■
Advanced Management Features		
Automated backup and disaster recovery		■
File browsing, searching, and disk quota management		■
HBase, MapReduce, Impala, and YARN usage reports		■
Support integration		■
Operational reports		■
Cloudera Navigator Data Management		
Metadata management and augmentation		■
Ingest policies		■
Analytics		■
Auditing		■
Lineage		■

Cloudera Manager 5 Frequently Asked Questions

This guide answers frequently asked questions about Cloudera Manager.

General Questions

What are the new features of Cloudera Manager 5?

For a list of new features in Cloudera Manager 5, see [New Features and Changes in Cloudera Manager 5](#).

What operating systems are supported?

See [CDH and Cloudera Manager Supported Operating Systems](#) for more detailed information on which operating systems are supported.

What databases are supported?

See [CDH and Cloudera Manager Supported Databases](#) for more detailed information on which database systems are supported.

What version of CDH is supported for Cloudera Manager 5?

See [CDH Requirements for Cloudera Manager](#) for detailed information.

Frequently Asked Questions About Cloudera Software

What are the differences between the Cloudera Express and the Cloudera Enterprise versions of Cloudera Manager?

Cloudera Express includes a free version of Cloudera Manager. The Cloudera Enterprise version of Cloudera Manager provides additional functionality. Both the Cloudera Express and Cloudera Enterprise versions automate the installation, configuration, and monitoring of CDH 4 or CDH 5 on an entire cluster. See the matrix at [Cloudera Express and Cloudera Enterprise Features](#) on page 75 for a comparison of the two versions.

The Cloudera Enterprise version of Cloudera Manager is available as part of the Cloudera Enterprise subscription offering, and requires a license. You can also choose a Cloudera Enterprise Enterprise Data Hub Edition Trial that is valid for 60 days.

If you are not an existing Cloudera customer, contact Cloudera Sales using this [form](#) or call 866-843-7207 to obtain a Cloudera Enterprise license. If you are already a Cloudera customer and you need to upgrade from Cloudera Express to Cloudera Enterprise, contact [Cloudera Support](#) to obtain a license.

Are there different types of Cloudera Enterprise licenses?

Cloudera Enterprise is available on a subscription basis in five editions, each designed around how you use the platform:

- **Basic Edition** provides superior support and advanced management for core Apache Hadoop.
- **Data Engineering Edition** for programmatic data preparation and predictive modeling.
- **Operational Database Edition** for online applications with real-time serving needs.
- **Analytic Database Edition** for BI and high-performance, SQL-based analytics.
- **Enterprise Data Hub Edition** provides for complete use of the platform.

All editions are available in your environment of choice: cloud, on-premise, or a hybrid deployment. For more information, see the [Cloudera Enterprise Data Sheet](#).

Can I upgrade CDH using Cloudera Manager?

You can upgrade to CDH 4.1.2 and higher from within the Cloudera Manager Admin Console using parcels. Furthermore, once you have installed or upgraded CDH using parcels, you can perform rolling upgrades on your CDH services. If you have HDFS high availability configured and enabled, you can perform a rolling upgrade on your cluster without taking the entire cluster down.



Warning:

- Cloudera Manager 4 and CDH 4 have reached End of Maintenance (EOM) on August 9, 2015. Cloudera does not support or provide updates for Cloudera Manager 4 and CDH 4 releases.
- Cloudera Manager 3 and CDH 3 have reached End of Maintenance (EOM) on June 20, 2013. Cloudera does not support or provide updates for Cloudera Manager 3 and CDH 3 releases.

For instructions on upgrading CDH 4 to CDH 5, see [Upgrading CDH 4 to CDH 5](#). For instructions on upgrading CDH 4 to a newer version, see [Upgrading CDH 4](#). For instructions on upgrading from CDH 3 to CDH 4, see [Upgrading CDH 3 to CDH 4 in a Cloudera Manager Deployment](#).

What version of CDH does Cloudera Manager 5 install?

Cloudera Manager 5 allows you to install any version of CDH 4 and a version of CDH 5 with the same minor version or lower as Cloudera Manager. For more information, see [CDH 5 and Cloudera Manager 5 Requirements and Supported Versions](#).

Where are CDH libraries located when I distribute CDH using parcels?

With parcel software distribution, the path to the CDH libraries is `/opt/cloudera/parcels/CDH/lib/` instead of the usual `/usr/lib/`.

What upgrade paths are available for Cloudera Manager, and what's involved?

For instructions about upgrading, see [Cloudera Upgrade](#).

How do I install Cloudera Manager 5 in a walled-off environment (no Internet access)?

You can set up a local repository and use it in the installer. For instructions, see [Understanding Custom Installation Solutions](#).

Do worker hosts need access to the Cloudera public repositories for an install with Cloudera Manager?

You can perform an installation or upgrade using the parcel format and when using parcels, only the Cloudera Manager Server requires access to the Cloudera public repositories. Distribution of the parcels to worker hosts is done between the Cloudera Manager Server and the worker hosts. See [Parcels](#) for more information. If you want to install using the traditional packages, hosts only require access to the installation files.

For both parcels and packages, it is also possible to create local repositories that serve these files to the hosts that are being upgraded. If you have established local repositories, no access to the Cloudera public repository is required. For more information, see [Creating and Using a Package Repository for Cloudera Manager](#).

Can I use the service monitoring features of Cloudera Manager without the Cloudera Management Service?

No. To understand the desired state of the system, Cloudera Manager requires the global configuration that the Cloudera Management Service roles gather and provide. The Cloudera Manager Agent doubles as both the agent for supervision and for monitoring.

Can I run the Cloudera Management Service and the Hadoop services on the host where the Cloudera Manager Server is running?

Yes. This is especially common in deployments that have a small number of hosts.

Does Cloudera Manager Support an API?

Yes. A comprehensive set of APIs for the various features is supported in this version of Cloudera Manager. For more information about the Cloudera Manager API, see [Cloudera Manager API](#) on page 56. You can download this [Cloudera Manager API example](#) that shows how to integrate with Nagios or other systems.

Cloudera Navigator 2 Frequently Asked Questions

Is Cloudera Navigator a module of Cloudera Manager?

Cloudera Navigator and Cloudera Manager complement each other. Cloudera Manager helps you manage services and Cloudera Navigator helps you manage the data stored in those services. Cloudera Navigator provides the following components:

- **Data Management** - Provides visibility into and control over the data in Hadoop datastores, and the computations performed on that data. Hadoop administrators, data stewards, and data scientists can use Cloudera Navigator to:
 - Audit data access and verify access privileges - The goal of auditing is to capture a complete and immutable record of all activity within a system. Cloudera Navigator [auditing](#) adds secure, real-time audit components to key data and access frameworks. Compliance groups can use Cloudera Navigator to configure, collect, and view audit events that show who accessed data, and how.
 - Search metadata and visualize lineage - Cloudera Navigator [metadata management](#) allows DBAs, data stewards, business analysts, and data scientists to define, search for, amend the properties of, and tag data entities and view relationships between datasets.
 - Policies - Data stewards can use Cloudera Navigator [policies](#) to define automated actions, based on data access or on a schedule, to add metadata, create alerts, and move or purge data.
 - Analytics - Hadoop administrators can use Cloudera Navigator [analytics](#) to examine data usage patterns and create policies based on those patterns.

Frequently Asked Questions About Cloudera Software

- **Data Encryption** - Data encryption and key management provide a critical layer of protection against potential threats by malicious actors on the network or in the datacenter. Encryption and key management are also requirements for meeting key compliance initiatives and ensuring the integrity of your enterprise data. The following Cloudera Navigator components enable compliance groups to manage encryption:
 - [Cloudera Navigator Encrypt](#) transparently encrypts and secures data at rest without requiring changes to your applications and ensures there is minimal performance lag in the encryption or decryption process.
 - [Cloudera Navigator Key Trustee Server](#) is an enterprise-grade virtual safe-deposit box that stores and manages cryptographic keys and other security artifacts.
 - [Cloudera Navigator Key HSM](#) allows Cloudera Navigator Key Trustee Server to seamlessly integrate with a hardware security module (HSM).

The Cloudera Navigator data management component is implemented as two roles in the [Cloudera Management Service](#): Navigator Audit Server and Navigator Metadata Server. You can add Cloudera Navigator data management roles while installing Cloudera Manager for the first time or into an existing Cloudera Manager installation. For information on compatible Cloudera Navigator and Cloudera Manager versions, see the [Product Compatibility Matrix for Cloudera Navigator](#) product compatibility matrix.

Is Cloudera Navigator included with a Cloudera Enterprise Enterprise Data Hub Edition license?

Yes. Cloudera Navigator is included with a Cloudera Enterprise Enterprise Data Hub Edition license and can be selected as a choice with a Cloudera Enterprise Flex Edition license.

Can Cloudera Navigator be purchased standalone—that is, without Cloudera Manager?

Cloudera Navigator components are managed by Cloudera Manager. Therefore, Cloudera Manager is a prerequisite for Cloudera Navigator.

What Cloudera Manager, CDH, and Impala releases does Cloudera Navigator 2 work with?

See [Product Compatibility Matrix for Cloudera Navigator](#).

Is Cloudera Navigator open source or closed source?

Cloudera Navigator is a closed-source management tool and part of the Cloudera suite of management capabilities for Hadoop.

How are Cloudera Navigator logs different from Cloudera Manager logs?

Cloudera Navigator tracks and aggregates only the accesses to the data stored in CDH services and used for audit reports and analysis. Cloudera Manager monitors and logs all the activity performed by CDH services that helps administrators maintain the health of the cluster. Together these logs provide better visibility into both the data access and system activity for an enterprise cluster.

Impala Frequently Asked Questions

Here are the categories of frequently asked questions for Impala, the interactive SQL engine included with CDH.

Trying Impala

How do I try Impala out?

To look at the core features and functionality on Impala, the easiest way to try out Impala is to download the Cloudera QuickStart VM and start the Impala service through Cloudera Manager, then use `impala-shell` in a terminal window or the Impala Query UI in the Hue web interface.

To do performance testing and try out the management features for Impala on a cluster, you need to move beyond the QuickStart VM with its virtualized single-node environment. Ideally, download the Cloudera Manager software to set up the cluster, then install the Impala software through Cloudera Manager.

Does Cloudera offer a VM for demonstrating Impala?

Cloudera offers a demonstration VM called the QuickStart VM, available in VMWare, VirtualBox, and KVM formats. For more information, see [the Cloudera QuickStart VM](#). After booting the QuickStart VM, many services are turned off by default; in the Cloudera Manager UI that appears automatically, turn on Impala and any other components that you want to try out.

Where can I find Impala documentation?

Starting with Impala 1.3.0, Impala documentation is integrated with the CDH 5 documentation, in addition to the standalone Impala documentation for use with CDH 4. For CDH 5, the core Impala developer and administrator information remains in the associated [Impala documentation](#) portion. Information about Impala release notes, installation, configuration, startup, and security is embedded in the corresponding CDH 5 guides.

- [New features](#)
- [Known and fixed issues](#)
- [Incompatible changes](#)
- [Installing Impala](#)
- [Upgrading Impala](#)
- [Configuring Impala](#)
- [Starting Impala](#)
- [Security for Impala](#)
- [CDH Version and Packaging Information](#)

Information about the latest CDH 4-compatible Impala release remains at the [Impala for CDH 4 Documentation](#) page.

Where can I get more information about Impala?

More product information is available here:

- O'Reilly introductory e-book: [Cloudera Impala: Bringing the SQL and Hadoop Worlds Together](#)
- O'Reilly getting started guide for developers: [Getting Started with Impala: Interactive SQL for Apache Hadoop](#)
- Blog: [Cloudera Impala: Real-Time Queries in Apache Hadoop, For Real](#)
- Webinar: [Introduction to Impala](#)
- Product website page: [Cloudera Enterprise RTQ](#)

To see the latest release announcements for Impala, see the [Cloudera Announcements](#) forum.

How can I ask questions and provide feedback about Impala?

- Join the [Impala discussion forum](#) and the [Impala mailing list](#) to ask questions and provide feedback.
- Use the [Impala Jira project](#) to log bug reports and requests for features.

Where can I get sample data to try?

You can get scripts that produce data files and set up an environment for TPC-DS style benchmark tests from [this Github repository](#). In addition to being useful for experimenting with performance, the tables are suited to experimenting with many aspects of SQL on Impala: they contain a good mixture of data types, data distributions, partitioning, and relational data suitable for join queries.

Impala System Requirements

What are the software and hardware requirements for running Impala?

For information on Impala requirements, see [Impala Requirements](#). Note that there is often a minimum required level of Cloudera Manager for any given Impala version.

How much memory is required?

Although Impala is not an in-memory database, when dealing with large tables and large result sets, you should expect to dedicate a substantial portion of physical memory for the `impalad` daemon. Recommended physical memory for an Impala node is 128 GB or higher. If practical, devote approximately 80% of physical memory to Impala.

The amount of memory required for an Impala operation depends on several factors:

- The file format of the table. Different file formats represent the same data in more or fewer data files. The compression and encoding for each file format might require a different amount of temporary memory to decompress the data for analysis.
- Whether the operation is a `SELECT` or an `INSERT`. For example, Parquet tables require relatively little memory to query, because Impala reads and decompresses data in 8 MB chunks. Inserting into a Parquet table is a more memory-intensive operation because the data for each data file (potentially hundreds of megabytes, depending on the value of the `PARQUET_FILE_SIZE` query option) is stored in memory until encoded, compressed, and written to disk.
- Whether the table is partitioned or not, and whether a query against a partitioned table can take advantage of partition pruning.
- Whether the final result set is sorted by the `ORDER BY` clause. Each Impala node scans and filters a portion of the total data, and applies the `LIMIT` to its own portion of the result set. In Impala 1.4.0 and higher, if the sort operation requires more memory than is available on any particular host, Impala uses a temporary disk work area to perform the sort. The intermediate result sets are all sent back to the coordinator node, which does the final sorting and then applies the `LIMIT` clause to the final result set.

For example, if you execute the query:

```
select * from giant_table order by some_column limit 1000;
```

and your cluster has 50 nodes, then each of those 50 nodes will transmit a maximum of 1000 rows back to the coordinator node. The coordinator node needs enough memory to sort (`LIMIT * cluster_size`) rows, although in the end the final result set is at most `LIMIT` rows, 1000 in this case.

Likewise, if you execute the query:

```
select * from giant_table where test_val > 100 order by some_column;
```

then each node filters out a set of rows matching the `WHERE` conditions, sorts the results (with no size limit), and sends the sorted intermediate rows back to the coordinator node. The coordinator node might need substantial memory to sort the final result set, and so might use a temporary disk work area for that final phase of the query.

- Whether the query contains any join clauses, `GROUP BY` clauses, analytic functions, or `DISTINCT` operators. These operations all require some in-memory work areas that vary depending on the volume and distribution of data. In Impala 2.0 and later, these kinds of operations utilize temporary disk work areas if memory usage grows too large to handle. See [SQL Operations that Spill to Disk](#) for details.
- The size of the result set. When intermediate results are being passed around between nodes, the amount of data depends on the number of columns returned by the query. For example, it is more memory-efficient to query only the columns that are actually needed in the result set rather than always issuing `SELECT *`.

- The mechanism by which work is divided for a join query. You use the `COMPUTE STATS` statement, and query hints in the most difficult cases, to help Impala pick the most efficient execution plan. See [Performance Considerations for Join Queries](#) for details.

See [Hardware Requirements](#) for more details and recommendations about Impala hardware prerequisites.

What processor type and speed does Cloudera recommend?

Impala makes use of SSE 4.1 instructions.

What EC2 instances are recommended for Impala?

For large storage capacity and large I/O bandwidth, consider the `hs1.8xlarge` and `cc2.8xlarge` instance types. Impala I/O patterns typically do not benefit enough from SSD storage to make up for the lower overall size. For performance and security considerations for deploying CDH and its components on AWS, see [Cloudera Enterprise Reference Architecture for AWS Deployments](#).

Supported and Unsupported Functionality In Impala

What are the main features of Impala?

- A large set of SQL statements, including [SELECT](#) and [INSERT](#), with [joins](#), [Subqueries in Impala SELECT Statements](#), and [Impala Analytic Functions](#). Highly compatible with HiveQL, and also including some vendor extensions. For more information, see [Impala SQL Language Reference](#).
- Distributed, high-performance queries. See [Tuning Impala for Performance](#) for information about Impala performance optimizations and tuning techniques for queries.
- Using Cloudera Manager, you can deploy and manage your Impala services. Cloudera Manager is the best way to get started with Impala on your cluster.
- Using Hue for queries.
- Appending and inserting data into tables through the [INSERT](#) statement. See [How Impala Works with Hadoop File Formats](#) for the details about which operations are supported for which file formats.
- ODBC: Impala is certified to run against MicroStrategy and Tableau, with restrictions. For more information, see [Configuring Impala to Work with ODBC](#).
- Querying data stored in HDFS and HBase in a single query. See [Using Impala to Query HBase Tables](#) for details.
- In Impala 2.2.0 and higher, querying data stored in the Amazon Simple Storage Service (S3). See [Using Impala with the Amazon S3 Filesystem](#) for details.
- Concurrent client requests. Each Impala daemon can handle multiple concurrent client requests. The effects on performance depend on your particular hardware and workload.
- Kerberos authentication. For more information, see [Overview of Impala Security](#).
- Partitions. With Impala SQL, you can create partitioned tables with the `CREATE TABLE` statement, and add and drop partitions with the `ALTER TABLE` statement. Impala also takes advantage of the partitioning present in Hive tables. See [Partitioning for Impala Tables](#) for details.

What features from relational databases or Hive are not available in Impala?

- Querying streaming data.
- Deleting individual rows. You delete data in bulk by overwriting an entire table or partition, or by dropping a table.
- Indexing (not currently). LZO-compressed text files can be indexed outside of Impala, as described in [Using LZO-Compressed Text Files](#).
- Full text search on text fields. The Cloudera Search product is appropriate for this use case.
- Custom Hive Serializer/Deserializer classes (SerDes). Impala supports a set of common native file formats that have built-in SerDes in CDH. See [How Impala Works with Hadoop File Formats](#) for details.
- Checkpointing within a query. That is, Impala does not save intermediate results to disk during long-running queries. Currently, Impala cancels a running query if any host on which that query is executing fails. When one or more hosts are down, Impala reroutes future queries to only use the available hosts, and Impala detects when

Frequently Asked Questions About Cloudera Software

the hosts come back up and begins using them again. Because a query can be submitted through any Impala node, there is no single point of failure. In the future, we will consider adding additional work allocation features to Impala, so that a running query would complete even in the presence of host failures.

- Encryption of data transmitted between Impala daemons.
- Hive indexes.
- Non-Hadoop data stores, such as relational databases.

For the detailed list of features that are different between Impala and HiveQL, see [SQL Differences Between Impala and Hive](#).

Does Impala support generic JDBC?

Impala supports the HiveServer2 JDBC driver.

Is Avro supported?

Yes, Avro is supported. Impala has always been able to query Avro tables. You can use the Impala `LOAD DATA` statement to load existing Avro data files into a table. Starting with Impala 1.4, you can create Avro tables with Impala. Currently, you still use the `INSERT` statement in Hive to copy data from another table into an Avro table. See [Using the Avro File Format with Impala Tables](#) for details.

How do I?

How do I prevent users from seeing the text of SQL queries?

For instructions on making the Impala log files unreadable by unprivileged users, see [Securing Impala Data and Log Files](#).

For instructions on password-protecting the web interface to the Impala log files and other internal server information, see [Securing the Impala Web User Interface](#).

In Impala 2.2 / CDH 5.4 and higher, you can use the log redaction feature to obfuscate sensitive information in Impala log files. See [Sensitive Data Redaction](#) for details.

How do I know how many Impala nodes are in my cluster?

The Impala statestore keeps track of how many `impalad` nodes are currently available. You can see this information through the statestore web interface. For example, at the URL `http://statestore_host:25010/metrics` you might see lines like the following:

```
statestore.live-backends:3
statestore.live-backends.list:[host1:22000, host1:26000, host2:22000]
```

The number of `impalad` nodes is the number of list items referring to port 22000, in this case two. (Typically, this number is one less than the number reported by the `statestore.live-backends` line.) If an `impalad` node became unavailable or came back after an outage, the information reported on this page would change appropriately.

Impala Performance

Are results returned as they become available, or all at once when a query completes?

Impala streams results whenever they are available, when possible. Certain SQL operations (aggregation or `ORDER BY`) require all of the input to be ready before Impala can return results.

Why does my query run slowly?

There are many possible reasons why a given query could be slow. Use the following checklist to diagnose performance issues with existing queries, and to avoid such issues when writing new queries, setting up new nodes, creating new tables, or loading data.

- Immediately after the query finishes, issue a `SUMMARY` command in `impala-shell`. You can check which phases of execution took the longest, and compare estimated values for memory usage and number of rows with the actual values.
- Immediately after the query finishes, issue a `PROFILE` command in `impala-shell`. The numbers in the `BytesRead`, `BytesReadLocal`, and `BytesReadShortCircuit` should be identical for a specific node. For example:

```
- BytesRead: 180.33 MB
- BytesReadLocal: 180.33 MB
- BytesReadShortCircuit: 180.33 MB
```

If `BytesReadLocal` is lower than `BytesRead`, something in your cluster is misconfigured, such as the `impalad` daemon not running on all the data nodes. If `BytesReadShortCircuit` is lower than `BytesRead`, short-circuit reads are not enabled properly on that node; see [Post-Installation Configuration for Impala](#) for instructions.

- If the table was just created, or this is the first query that accessed the table after an `INVALIDATE METADATA` statement or after the `impalad` daemon was restarted, there might be a one-time delay while the metadata for the table is loaded and cached. Check whether the slowdown disappears when the query is run again. When doing performance comparisons, consider issuing a `DESCRIBE table_name` statement for each table first, to make sure any timings only measure the actual query time and not the one-time wait to load the table metadata.
- Is the table data in uncompressed text format? Check by issuing a `DESCRIBE FORMATTED table_name` statement. A text table is indicated by the line:

```
InputFormat: org.apache.hadoop.mapred.TextInputFormat
```

Although uncompressed text is the default format for a `CREATE TABLE` statement with no `STORED AS` clauses, it is also the bulkiest format for disk storage and consequently usually the slowest format for queries. For data where query performance is crucial, particularly for tables that are frequently queried, consider starting with or converting to a compact binary file format such as Parquet, Avro, RCFile, or SequenceFile. For details, see [How Impala Works with Hadoop File Formats](#).

- If your table has many columns, but the query refers to only a few columns, consider using the Parquet file format. Its data files are organized with a column-oriented layout that lets queries minimize the amount of I/O needed to retrieve, filter, and aggregate the values for specific columns. See [Using the Parquet File Format with Impala Tables](#) for details.
- If your query involves any joins, are the tables in the query ordered so that the tables or subqueries are ordered with the one returning the largest number of rows on the left, followed by the smallest (most selective), the second smallest, and so on? That ordering allows Impala to optimize the way work is distributed among the nodes and how intermediate results are routed from one node to another. For example, all other things being equal, the following join order results in an efficient query:

```
select some_col from
  huge_table join big_table join small_table join medium_table
where
  huge_table.id = big_table.id
  and big_table.id = medium_table.id
  and medium_table.id = small_table.id;
```

See [Performance Considerations for Join Queries](#) for performance tips for join queries.

- Also for join queries, do you have table statistics for the table, and column statistics for the columns used in the join clauses? Column statistics let Impala better choose how to distribute the work for the various pieces of a join query. See [Table and Column Statistics](#) for details about gathering statistics.
- Does your table consist of many small data files? Impala works most efficiently with data files in the multi-megabyte range; Parquet, a format optimized for data warehouse-style queries, uses large files (originally 1 GB, now 256 MB in Impala 2.0 and higher) with a block size matching the file size. Use the `DESCRIBE FORMATTED table_name` statement in `impala-shell` to see where the data for a table is located, and use the `hadoop fs -ls` or `hdfs dfs -ls` Unix commands to see the files and their sizes. If you have thousands of small data files, that is a signal that you should consolidate into a smaller number of large files. Use an `INSERT ... SELECT` statement to copy the data to a new table, reorganizing into new data files as part of the process. Prefer to construct large data files and import them in bulk through the `LOAD DATA` or `CREATE EXTERNAL TABLE` statements, rather than issuing

many `INSERT ... VALUES` statements; each `INSERT ... VALUES` statement creates a separate tiny data file. If you have thousands of files all in the same directory, but each one is megabytes in size, consider using a partitioned table so that each partition contains a smaller number of files. See the following point for more on partitioning.

- If your data is easy to group according to time or geographic region, have you partitioned your table based on the corresponding columns such as `YEAR`, `MONTH`, and/or `DAY`? Partitioning a table based on certain columns allows queries that filter based on those same columns to avoid reading the data files for irrelevant years, postal codes, and so on. (Do not partition down to too fine a level; try to structure the partitions so that there is still sufficient data in each one to take advantage of the multi-megabyte HDFS block size.) See [Partitioning for Impala Tables](#) for details.

Why does my `SELECT` statement fail?

When a `SELECT` statement fails, the cause usually falls into one of the following categories:

- A timeout because of a performance, capacity, or network issue affecting one particular node.
- Excessive memory use for a join query, resulting in automatic cancellation of the query.
- A low-level issue affecting how native code is generated on each node to handle particular `WHERE` clauses in the query. For example, a machine instruction could be generated that is not supported by the processor of a certain node. If the error message in the log suggests the cause was an illegal instruction, consider turning off native code generation temporarily, and trying the query again.
- Malformed input data, such as a text data file with an enormously long line, or with a delimiter that does not match the character specified in the `FIELDS TERMINATED BY` clause of the `CREATE TABLE` statement.

Why does my `INSERT` statement fail?

When an `INSERT` statement fails, it is usually the result of exceeding some limit within a Hadoop component, typically HDFS.

- An `INSERT` into a partitioned table can be a strenuous operation due to the possibility of opening many files and associated threads simultaneously in HDFS. Impala 1.1.1 includes some improvements to distribute the work more efficiently, so that the values for each partition are written by a single node, rather than as a separate data file from each node.
- Certain expressions in the `SELECT` part of the `INSERT` statement can complicate the execution planning and result in an inefficient `INSERT` operation. Try to make the column data types of the source and destination tables match up, for example by doing `ALTER TABLE ... REPLACE COLUMNS` on the source table if necessary. Try to avoid `CASE` expressions in the `SELECT` portion, because they make the result values harder to predict than transferring a column unchanged or passing the column through a built-in function.
- Be prepared to raise some limits in the HDFS configuration settings, either temporarily during the `INSERT` or permanently if you frequently run such `INSERT` statements as part of your ETL pipeline.
- The resource usage of an `INSERT` statement can vary depending on the file format of the destination table. Inserting into a Parquet table is memory-intensive, because the data for each partition is buffered in memory until it reaches 1 gigabyte, at which point the data file is written to disk. Impala can distribute the work for an `INSERT` more efficiently when statistics are available for the source table that is queried during the `INSERT` statement. See [Table and Column Statistics](#) for details about gathering statistics.

Does Impala performance improve as it is deployed to more hosts in a cluster in much the same way that Hadoop performance does?

Yes. Impala scales with the number of hosts. It is important to install Impala on all the DataNodes in the cluster, because otherwise some of the nodes must do remote reads to retrieve data not available for local reads. Data locality is an important architectural aspect for Impala performance. See [this Impala performance blog post](#) for background. Note that this blog post refers to benchmarks with Impala 1.1.1; Impala has added even more performance features in the 1.2.x series.

Is the HDFS block size reduced to achieve faster query results?

No. Impala does not make any changes to the HDFS or HBase data sets.

The default Parquet block size is relatively large (256 MB in Impala 2.0 and later; 1 GB in earlier releases). You can control the block size when creating Parquet files using the [PARQUET_FILE_SIZE](#) query option.

Does Impala use caching?

Impala does not cache table data. It does cache some table and file metadata. Although queries might run faster on subsequent iterations because the data set was cached in the OS buffer cache, Impala does not explicitly control this.

Impala takes advantage of the HDFS caching feature in CDH 5. You can designate which tables or partitions are cached through the `CACHED` and `UNCACHED` clauses of the `CREATE TABLE` and `ALTER TABLE` statements. Impala can also take advantage of data that is pinned in the HDFS cache through the `hdfsccacheadmin` command. See [Using HDFS Caching with Impala \(CDH 5.1 or higher only\)](#) for details.

Impala Use Cases

What are good use cases for Impala as opposed to Hive or MapReduce?

Impala is well-suited to executing SQL queries for interactive exploratory analytics on large data sets. Hive and MapReduce are appropriate for very long running, batch-oriented tasks such as ETL.

Is MapReduce required for Impala? Will Impala continue to work as expected if MapReduce is stopped?

Impala does not use MapReduce at all.

Can Impala be used for complex event processing?

For example, in an industrial environment, many agents may generate large amounts of data. Can Impala be used to analyze this data, checking for notable changes in the environment?

Complex Event Processing (CEP) is usually performed by dedicated stream-processing systems. Impala is not a stream-processing system, as it most closely resembles a relational database.

Is Impala intended to handle real time queries in low-latency applications or is it for ad hoc queries for the purpose of data exploration?

Ad-hoc queries are the primary use case for Impala. We anticipate it being used in many other situations where low-latency is required. Whether Impala is appropriate for any particular use-case depends on the workload, data size and query volume. See [Impala Benefits](#) on page 10 for the primary benefits you can expect when using Impala.

Questions about Impala And Hive

How does Impala compare to Hive and Pig?

Impala is different from Hive and Pig because it uses its own daemons that are spread across the cluster for queries. Because Impala does not rely on MapReduce, it avoids the startup overhead of MapReduce jobs, allowing Impala to return results in real time.

Can I do transforms or add new functionality?

Impala adds support for UDFs in Impala 1.2. You can write your own functions in C++, or reuse existing Java-based Hive UDFs. The UDF support includes scalar functions and user-defined aggregate functions (UDAs). User-defined table functions (UDTFs) are not currently supported.

Impala does not currently support an extensible serialization-deserialization framework (SerDes), and so adding extra functionality to Impala is not as straightforward as for Hive or Pig.

Can any Impala query also be executed in Hive?

Yes. There are some minor differences in how some queries are handled, but Impala queries can also be completed in Hive. Impala SQL is a subset of HiveQL, with some functional limitations such as transforms. For details of the Impala

Frequently Asked Questions About Cloudera Software

SQL dialect, see [Impala SQL Statements](#). For the Impala built-in functions, see [Impala Built-In Functions](#). For the detailed list of differences between Impala and HiveQL, see [SQL Differences Between Impala and Hive](#).

Can I use Impala to query data already loaded into Hive and HBase?

There are no additional steps to allow Impala to query tables managed by Hive, whether they are stored in HDFS or HBase. Make sure that Impala is configured to access the Hive metastore correctly and you should be ready to go. Keep in mind that `impalad`, by default, runs as the `impala` user, so you might need to adjust some file permissions depending on how strict your permissions are currently.

See [Using Impala to Query HBase Tables](#) for details about querying data in HBase.

Is Hive an Impala requirement?

The Hive metastore service is a requirement. Impala shares the same metastore database as Hive, allowing Impala and Hive to access the same tables transparently.

Hive itself is optional, and does not need to be installed on the same nodes as Impala. Currently, Impala supports a wider variety of read (query) operations than write (insert) operations; you use Hive to insert data into tables that use certain file formats. See [How Impala Works with Hadoop File Formats](#) for details.

Impala Availability

Is Impala production ready?

Impala has finished its beta release cycle, and the 1.0, 1.1, and 1.2 GA releases are production ready. The 1.1.x series includes additional security features for authorization, an important requirement for production use in many organizations. The 1.2.x series includes important performance features, particularly for large join queries. Some Cloudera customers are already using Impala for large workloads.

The Impala 1.3.0 and higher releases are bundled with corresponding levels of CDH 5. The number of new features grows with each release. See [What's New in Apache Impala \(incubating\)](#) for a full list.

How do I configure Hadoop high availability (HA) for Impala?

You can set up a proxy server to relay requests back and forth to the Impala servers, for load balancing and high availability. See [Using Impala through a Proxy for High Availability](#) for details.

You can enable HDFS HA for the Hive metastore. See the [CDH5 High Availability Guide](#) for details.

What happens if there is an error in Impala?

There is not a single point of failure in Impala. All Impala daemons are fully able to handle incoming queries. If a machine fails however, all queries with fragments running on that machine will fail. Because queries are expected to return quickly, you can just rerun the query if there is a failure. See [Impala Concepts and Architecture](#) for details about the Impala architecture.

The longer answer: Impala must be able to connect to the Hive metastore. Impala aggressively caches metadata so the metastore host should have minimal load. Impala relies on the HDFS NameNode, and, in CDH4, you can configure HA for HDFS. Impala also has centralized services, known as the [statestore](#) and [catalog](#) services, that run on one host only. Impala continues to execute queries if the statestore host is down, but it will not get state updates. For example, if a host is added to the cluster while the statestore host is down, the existing instances of `impalad` running on the other hosts will not find out about this new host. Once the statestore process is restarted, all the information it serves is automatically reconstructed from all running Impala daemons.

What is the maximum number of rows in a table?

There is no defined maximum. Some customers have used Impala to query a table with over a trillion rows.

Can Impala and MapReduce jobs run on the same cluster without resource contention?

Yes. See [Controlling Impala Resource Usage](#) for how to control Impala resource usage using the Linux cgroup mechanism, and [Resource Management for Impala](#) for how to use Impala with the YARN resource management framework. Impala is designed to run on the DataNode hosts. Any contention depends mostly on the cluster setup and workload.

For a detailed information about configuring a cluster to share resources between Impala queries and MapReduce jobs, see [How To Create a Multitenant Enterprise Data Hub](#) and [How to Configure Resource Management for Impala](#).

Impala Internals

On which hosts does Impala run?

Cloudera strongly recommends running the `impalad` daemon on each DataNode for good performance. Although this topology is not a hard requirement, if there are data blocks with no Impala daemons running on any of the hosts containing replicas of those blocks, queries involving that data could be very inefficient. In that case, the data must be transmitted from one host to another for processing by “remote reads”, a condition Impala normally tries to avoid. See [Impala Concepts and Architecture](#) for details about the Impala architecture. Impala schedules query fragments on all hosts holding data relevant to the query, if possible.

In cases where some hosts in the cluster have much greater CPU and memory capacity than others, or where some hosts have extra CPU capacity because some CPU-intensive phases are single-threaded, some users have run multiple `impalad` daemons on a single host to take advantage of the extra CPU capacity. This configuration is only practical for specific workloads that rely heavily on aggregation, and the physical hosts must have sufficient memory to accommodate the requirements for multiple `impalad` instances.

How are joins performed in Impala?

By default, Impala automatically determines the most efficient order in which to join tables using a cost-based method, based on their overall size and number of rows. (This is a new feature in Impala 1.2.2 and higher.) The `COMPUTE STATS` statement gathers information about each table that is crucial for efficient join performance. Impala chooses between two techniques for join queries, known as “broadcast joins” and “partitioned joins”. See [Joins in Impala SELECT Statements](#) for syntax details and [Performance Considerations for Join Queries](#) for performance considerations.

How does Impala process join queries for large tables?

Impala utilizes multiple strategies to allow joins between tables and result sets of various sizes. When joining a large table with a small one, the data from the small table is transmitted to each node for intermediate processing. When joining two large tables, the data from one of the tables is divided into pieces, and each node processes only selected pieces. See [Joins in Impala SELECT Statements](#) for details about join processing, [Performance Considerations for Join Queries](#) for performance considerations, and [Query Hints in Impala SELECT Statements](#) for how to fine-tune the join strategy.

What is Impala's aggregation strategy?

Impala currently only supports in-memory hash aggregation. In Impala 2.0 and higher, if the memory requirements for a join or aggregation operation exceed the memory limit for a particular host, Impala uses a temporary work area on disk to help the query complete successfully.

How is Impala metadata managed?

Impala uses two pieces of metadata: the catalog information from the Hive metastore and the file metadata from the NameNode. Currently, this metadata is lazily populated and cached when an `impalad` needs it to plan a query.

The `REFRESH` statement updates the metadata for a particular table after loading new data through Hive. The `INVALIDATE METADATA` statement refreshes all metadata, so that Impala recognizes new tables or other DDL and DML changes performed through Hive.

In Impala 1.2 and higher, a dedicated `catalogd` daemon broadcasts metadata changes due to Impala DDL or DML statements to all nodes, reducing or eliminating the need to use the `REFRESH` and `INVALIDATE METADATA` statements.

Frequently Asked Questions About Cloudera Software

What load do concurrent queries produce on the NameNode?

The load Impala generates is very similar to MapReduce. Impala contacts the NameNode during the planning phase to get the file metadata (this is only run on the host the query was sent to). Every `impalad` will read files as part of normal processing of the query.

How does Impala achieve its performance improvements?

These are the main factors in the performance of Impala versus that of other Hadoop components and related technologies.

Impala avoids MapReduce. While MapReduce is a great general parallel processing model with many benefits, it is not designed to execute SQL. Impala avoids the inefficiencies of MapReduce in these ways:

- Impala does not materialize intermediate results to disk. SQL queries often map to multiple MapReduce jobs with all intermediate data sets written to disk.
- Impala avoids MapReduce start-up time. For interactive queries, the MapReduce start-up time becomes very noticeable. Impala runs as a service and essentially has no start-up time.
- Impala can more naturally disperse query plans instead of having to fit them into a pipeline of map and reduce jobs. This enables Impala to parallelize multiple stages of a query and avoid overheads such as sort and shuffle when unnecessary.

Impala uses a more efficient execution engine by taking advantage of modern hardware and technologies:

- Impala generates runtime code. Impala uses LLVM to generate assembly code for the query that is being run. Individual queries do not have to pay the overhead of running on a system that needs to be able to execute arbitrary queries.
- Impala uses available hardware instructions when possible. Impala uses the supplemental SSE3 (SSSE3) instructions which can offer tremendous speedups in some cases. (Impala 2.0 and 2.1 required the SSE4.1 instruction set; Impala 2.2 and higher relax the restriction again so only SSSE3 is required.)
- Impala uses better I/O scheduling. Impala is aware of the disk location of blocks and is able to schedule the order to process blocks to keep all disks busy.
- Impala is designed for performance. A lot of time has been spent in designing Impala with sound performance-oriented fundamentals, such as tight inner loops, inlined function calls, minimal branching, better use of cache, and minimal memory usage.

What happens when the data set exceeds available memory?

Currently, if the memory required to process intermediate results on a node exceeds the amount available to Impala on that node, the query is cancelled. You can adjust the memory available to Impala on each node, and you can fine-tune the join strategy to reduce the memory required for the biggest queries. We do plan on supporting external joins and sorting in the future.

Keep in mind though that the memory usage is not directly based on the input data set size. For aggregations, the memory usage is the number of rows *after* grouping. For joins, the memory usage is the combined size of the tables *excluding* the biggest table, and Impala can use join strategies that divide up large joined tables among the various nodes rather than transmitting the entire table to each node.

What are the most memory-intensive operations?

If a query fails with an error indicating “memory limit exceeded”, you might suspect a memory leak. The problem could actually be a query that is structured in a way that causes Impala to allocate more memory than you expect, exceeded the memory allocated for Impala on a particular node. Some examples of query or table structures that are especially memory-intensive are:

- `INSERT` statements using dynamic partitioning, into a table with many different partitions. (Particularly for tables using Parquet format, where the data for each partition is held in memory until it reaches the full block size in size before it is written to disk.) Consider breaking up such operations into several different `INSERT` statements, for example to load data one year at a time rather than for all years at once.

- `GROUP BY` on a unique or high-cardinality column. Impala allocates some handler structures for each different value in a `GROUP BY` query. Having millions of different `GROUP BY` values could exceed the memory limit.
- Queries involving very wide tables, with thousands of columns, particularly with many `STRING` columns. Because Impala allows a `STRING` value to be up to 32 KB, the intermediate results during such queries could require substantial memory allocation.

When does Impala hold on to or return memory?

Impala allocates memory using [tcmalloc](#), a memory allocator that is optimized for high concurrency. Once Impala allocates memory, it keeps that memory reserved to use for future queries. Thus, it is normal for Impala to show high memory usage when idle. If Impala detects that it is about to exceed its memory limit (defined by the `-mem_limit` startup option or the `MEM_LIMIT` query option), it deallocates memory not needed by the current queries.

When issuing queries through the JDBC or ODBC interfaces, make sure to call the appropriate close method afterwards. Otherwise, some memory associated with the query is not freed.

SQL

Is there an `UPDATE` statement?

Impala does not currently have an `UPDATE` statement, which would typically be used to change a single row, a small group of rows, or a specific column. The HDFS-based files used by typical Impala queries are optimized for bulk operations across many megabytes of data at a time, making traditional `UPDATE` operations inefficient or impractical.

You can use the following techniques to achieve the same goals as the familiar `UPDATE` statement, in a way that preserves efficient file layouts for subsequent queries:

- Replace the entire contents of a table or partition with updated data that you have already staged in a different location, either using `INSERT OVERWRITE`, `LOAD DATA`, or manual HDFS file operations followed by a `REFRESH` statement for the table. Optionally, you can use built-in functions and expressions in the `INSERT` statement to transform the copied data in the same way you would normally do in an `UPDATE` statement, for example to turn a mixed-case string into all uppercase or all lowercase.
- To update a single row, use an HBase table, and issue an `INSERT ... VALUES` statement using the same key as the original row. Because HBase handles duplicate keys by only returning the latest row with a particular key value, the newly inserted row effectively hides the previous one.

Can Impala do user-defined functions (UDFs)?

Impala 1.2 and higher does support UDFs and UDAs. You can either write native Impala UDFs and UDAs in C++, or reuse UDFs (but not UDAs) originally written in Java for use with Hive. See [Impala User-Defined Functions \(UDFs\)](#) for details.

Why do I have to use `REFRESH` and `INVALIDATE METADATA`, what do they do?

In Impala 1.2 and higher, there is much less need to use the `REFRESH` and `INVALIDATE METADATA` statements:

- The new `impala-catalog` service, represented by the `catalogd` daemon, broadcasts the results of Impala DDL statements to all Impala nodes. Thus, if you do a `CREATE TABLE` statement in Impala while connected to one node, you do not need to do `INVALIDATE METADATA` before issuing queries through a different node.
- The catalog service only recognizes changes made through Impala, so you must still issue a `REFRESH` statement if you load data through Hive or by manipulating files in HDFS, and you must issue an `INVALIDATE METADATA` statement if you create a table, alter a table, add or drop partitions, or do other DDL statements in Hive.
- Because the catalog service broadcasts the results of `REFRESH` and `INVALIDATE METADATA` statements to all nodes, in the cases where you do still need to issue those statements, you can do that on a single node rather than on every node, and the changes will be automatically recognized across the cluster, making it more convenient to load balance by issuing queries through arbitrary Impala nodes rather than always using the same coordinator node.

Why is space not freed up when I issue DROP TABLE?

Impala deletes data files when you issue a `DROP TABLE` on an internal table, but not an external one. By default, the `CREATE TABLE` statement creates internal tables, where the files are managed by Impala. An external table is created with a `CREATE EXTERNAL TABLE` statement, where the files reside in a location outside the control of Impala. Issue a `DESCRIBE FORMATTED` statement to check whether a table is internal or external. The keyword `MANAGED_TABLE` indicates an internal table, from which Impala can delete the data files. The keyword `EXTERNAL_TABLE` indicates an external table, where Impala will leave the data files untouched when you drop the table.

Even when you drop an internal table and the files are removed from their original location, you might not get the hard drive space back immediately. By default, files that are deleted in HDFS go into a special trashcan directory, from which they are purged after a period of time (by default, 6 hours). For background information on the trashcan mechanism, see <https://archive.cloudera.com/cdh4/cdh/4/hadoop/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>. For information on purging files from the trashcan, see <https://archive.cloudera.com/cdh4/cdh/4/hadoop/hadoop-project-dist/hadoop-common/FileSystemShell.html>.

When Impala deletes files and they are moved to the HDFS trashcan, they go into an HDFS directory owned by the `impala` user. If the `impala` user does not have an HDFS home directory where a trashcan can be created, the files are not deleted or moved, as a safety measure. If you issue a `DROP TABLE` statement and find that the table data files are left in their original location, create an HDFS directory `/user/impala`, owned and writeable by the `impala` user. For example, you might find that `/user/impala` is owned by the `hdfs` user, in which case you would switch to the `hdfs` user and issue a command such as:

```
hdfs dfs -chown -R impala /user/impala
```

Is there a DUAL table?

You might be used to running queries against a single-row table named `DUAL` to try out expressions, built-in functions, and UDFs. Impala does not have a `DUAL` table. To achieve the same result, you can issue a `SELECT` statement without any table name:

```
select 2+2;
select substr('hello',2,1);
select pow(10,6);
```

Partitioned Tables

How do I load a big CSV file into a partitioned table?

To load a data file into a partitioned table, when the data file includes fields like year, month, and so on that correspond to the partition key columns, use a two-stage process. First, use the `LOAD DATA` or `CREATE EXTERNAL TABLE` statement to bring the data into an unpartitioned text table. Then use an `INSERT ... SELECT` statement to copy the data from the unpartitioned table to a partitioned one. Include a `PARTITION` clause in the `INSERT` statement to specify the partition key columns. The `INSERT` operation splits up the data into separate data files for each partition. For examples, see [Partitioning for Impala Tables](#). For details about loading data into partitioned Parquet tables, a popular choice for high-volume data, see [Loading Data into Parquet Tables](#).

Can I do INSERT ... SELECT * into a partitioned table?

When you use the `INSERT ... SELECT *` syntax to copy data into a partitioned table, the columns corresponding to the partition key columns must appear last in the columns returned by the `SELECT *`. You can create the table with the partition key columns defined last. Or, you can use the `CREATE VIEW` statement to create a view that reorders the columns: put the partition key columns last, then do the `INSERT ... SELECT *` from the view.

HBase

What kinds of Impala queries or data are best suited for HBase?

HBase tables are ideal for queries where normally you would use a key-value store. That is, where you retrieve a single row or a few rows, by testing a special unique key column using the `=` or `IN` operators.

HBase tables are not suitable for queries that produce large result sets with thousands of rows. HBase tables are also not suitable for queries that perform full table scans because the `WHERE` clause does not request specific values from the unique key column.

Use HBase tables for data that is inserted one row or a few rows at a time, such as by the `INSERT . . . VALUES` syntax. Loading data piecemeal like this into an HDFS-backed table produces many tiny files, which is a very inefficient layout for HDFS data files.

If the lack of an `UPDATE` statement in Impala is a problem for you, you can simulate single-row updates by doing an `INSERT . . . VALUES` statement using an existing value for the key column. The old row value is hidden; only the new row value is seen by queries.

HBase tables are often wide (containing many columns) and sparse (with most column values `NULL`). For example, you might record hundreds of different data points for each user of an online service, such as whether the user had registered for an online game or enabled particular account features. With Impala and HBase, you could look up all the information for a specific customer efficiently in a single query. For any given customer, most of these columns might be `NULL`, because a typical customer might not make use of most features of an online service.

Cloudera Search Frequently Asked Questions

This section includes answers to questions commonly asked about Search for CDH. Questions are divided into the following categories:

General

The following are general questions about Cloudera Search and the answers to those questions.

What is Cloudera Search?

Cloudera Search is [Apache Solr](#) integrated with CDH, including Apache Lucene, Apache SolrCloud, Apache Flume, Apache Tika, and Apache Hadoop MapReduce and HDFS. Cloudera Search also includes valuable integrations that make searching more scalable, easy to use, and optimized for both near-real-time and batch-oriented indexing. These integrations include Cloudera Morphlines, a customizable transformation chain that simplifies loading any type of data into Cloudera Search.

What is the difference between Lucene and Solr?

Lucene is a low-level search library that is accessed by a Java API. Solr is a search server that runs in a servlet container and provides structure and convenience around the underlying Lucene library.

What is Apache Tika?

The Apache Tika toolkit detects and extracts metadata and structured text content from various documents using existing parser libraries. Using the `solrCell` morphline command, the output from Apache Tika can be mapped to a Solr schema and indexed.

How does Cloudera Search relate to web search?

Traditional web search engines crawl web pages on the Internet for content to index. Cloudera Search indexes files and data that are stored in HDFS and HBase. To make web data available through Cloudera Search, it needs to be downloaded and stored in [Cloudera Enterprise](#).

Frequently Asked Questions About Cloudera Software

How does Cloudera Search relate to enterprise search?

Enterprise search connects with different backends (such as RDBMS and filesystems) and indexes data in all those systems. Cloudera Search is intended as a full-text search capability for data in CDH. Cloudera Search is a tool added to the Cloudera data processing platform and does not aim to be a stand-alone search solution, but rather a user-friendly interface to explore data in Hadoop and HBase.

How does Cloudera Search relate to custom search applications?

Custom and specialized search applications are an excellent complement to the Cloudera data-processing platform. Cloudera Search is not designed to be a custom application for niche vertical markets. However, Cloudera Search does include a simple search GUI through a plug-in application for Hue. The Hue plug-in application is based on the Solr API and allows for easy exploration, along with all of the other Hadoop frontend applications in Hue.

Do Search security features use Kerberos?

Yes, Cloudera Search includes support for Kerberos authentication. Search continues to use simple authentication with the anonymous user as the default configuration, but Search now supports changing the authentication scheme to Kerberos. All required packages are installed during the installation or upgrade process. Additional configuration is required before Kerberos is available in your environment.

Do I need to configure Sentry restrictions for each access mode, such as for the admin console and for the command line?

Sentry restrictions are consistently applied regardless of the way users attempt to complete actions. For example, restricting access to data in a collection consistently restricts that access, whether queries come from the command line, from a browser, or through the admin console.

Does Search support indexing data stored in JSON files and objects?

Yes, you can use the `readJson` and `extractJsonPaths` morphline commands that are included with the CDK to access JSON data and files. For more information, see [cdk-morphlines-json](#).

How can I set up Cloudera Search so that results include links back to the source that contains the result?

You can use stored results fields to create links back to source documents. For information on data types, including the option to set results fields as stored, see the Solr Wiki page on [SchemaXml](#).

For example, with `MapReduceIndexerTool` you can take advantage of fields such as `file_path`. See [MapReduceIndexerTool Metadata](#) for more information. The output from the `MapReduceIndexerTool` includes file path information that can be used to construct links to source documents.

If you use the [Hue UI](#), you can link to data in HDFS by inserting links of the form:

```
<a href="/filebrowser/download/{{file_path}}?disposition=inline">Download</a>
```

Why do I get an error "no field name specified in query and no default specified via 'df' param" when I query a Schemaless collection?

Schemaless collections initially have no default or `df` setting. As a result, simple searches that might succeed on non-Schemaless collections may fail on Schemaless collections.

When a user submits a search, it must be clear which field Cloudera Search should query. A default field, or `df`, is often specified in `solrconfig.xml`, and when this is the case, users can submit queries that do not specify fields. In such situations, Solr uses the `df` value.

When a new collection is created in Schemaless mode, there are initially no fields defined, so no field can be chosen as the `df` field. As a result, when query request handlers do not specify a `df`, errors can result. This issue can be addressed in several ways:

- Queries can specify any valid field name on which to search. In such a case, no `df` is required.
- Queries can specify a default field using the `df` parameter. In such a case, the `df` is specified in the query.

- You can uncomment the `df` section of the generated schemaless `solrconfig.xml` file and set the `df` parameter to the desired field. In such a case, all subsequent queries can use the `df` field in `solrconfig.xml` if no field or `df` value is specified.

Performance and Fail Over

The following are questions about performance and fail over in Cloudera Search and the answers to those questions.

How large of an index does Cloudera Search support per search server?

This question includes too many variables to provide a single answer. Typically, a server can host from 10 to 300 million documents, with the underlying index as large as hundreds of gigabytes. To determine a reasonable maximum document quantity and index size for servers in your deployment, prototype with realistic data and queries.

What is the response time latency I can expect?

Many factors affect how quickly responses are returned. Some factors that contribute to latency include whether the system is also completing indexing, the type of fields you are searching, whether the search results require aggregation, and whether there are sufficient resources for your search services.

With appropriately-sized hardware, if the query results are found in memory, they may be returned within milliseconds. Conversely, complex queries requiring results aggregation over huge indexes may take a few seconds.

The time between when Search begins to work on indexing new data and when that data can be queried can be as short as a few seconds, depending on your configuration.

This high performance is supported by custom caching optimizations that Cloudera has added to the Solr/HDFS integration. These optimizations allow for rapid read and writes of files in HDFS, performing at or above the speeds of stand-alone Solr reading and writing from local disk.

What happens when a write to the Lucene indexer fails?

Cloudera Search provides two configurable, highly available, and fault-tolerant data ingestion schemes: near real-time ingestion using the Flume Solr Sink and MapReduce-based batch ingestion using the MapReduceIndexerTool. These approaches are discussed in more detail in [Search High Availability](#).

What hardware or configuration changes can I make to improve Search performance?

Search performance can be constrained by CPU limits. If you're seeing bottlenecks, consider allocating more CPU to Search.

Are there settings that can help avoid out of memory (OOM) errors during data ingestion?

A data ingestion pipeline can be made up of many connected segments, each of which may need to be evaluated for sufficient resources. A common limiting factor is the relatively small default amount of permgen memory allocated to the flume JVM. Allocating additional memory to the Flume JVM may help avoid OOM errors. For example, for JVM settings for Flume, the following settings are often sufficient:

```
-Xmx2048m -XX:MaxPermSize=256M
```

How can I redistribute shards across a cluster?

You can move shards between hosts using the process described in [Migrating Solr Replicas](#).

Can I adjust replication levels?

For information on adjusting replication levels, see [Replication Settings](#). Do not adjust HDFS replication settings for Solr in most cases.

Schema Management

The following are questions about schema management in Cloudera Search and the answers to those questions.

Frequently Asked Questions About Cloudera Software

If my schema changes, will I need to re-index all of my data and files?

When you change the schema, Cloudera recommends re-indexing. For example, if you add a new field to the index, apply the new field to all index entries through re-indexing. Re-indexing is required in such a case because existing documents do not yet have the field. Cloudera Search includes a MapReduce batch-indexing solution for re-indexing and a GoLive feature that assures updated indexes are dynamically served.

While you should typically re-index after adding a new field, this is not necessary if the new field applies only to new documents or data. This is because, were indexing to be completed, existing documents would still have no data for the field, making the effort unnecessary.

For schema changes that only apply to queries, re-indexing is not necessary.

Can I extract fields based on regular expressions or rules?

Cloudera Search supports limited regular expressions in Search queries. For details, see [Lucene Regular Expressions](#).

On data ingestion, Cloudera Search supports easy and powerful extraction of fields based on regular expressions. For example the `grok` morphline command supports field extraction using regular expressions.

Cloudera Search also includes support for rule directed ETL with an extensible rule engine, in the form of the `tryRules` morphline command.

Can I use nested schemas?

Cloudera Search does not support nesting documents in this release. Cloudera Search assumes documents in the Cloudera Search repository have a flat structure.

What is Apache Avro and how can I use an Avro schema for more flexible schema evolution?

To learn more about Avro and Avro schemas, see the [Avro Overview page](#) and the [Avro Specification page](#).

To see examples of how to implement inheritance, backwards compatibility, and polymorphism with Avro, see this [InfoQ article](#).

Supportability

The following are questions about supportability in Cloudera Search and the answers to those questions.

Does Cloudera Search support multiple languages?

Cloudera Search supports approximately 30 languages, including most Western European languages, as well as Chinese, Japanese, and Korean.

Which file formats does Cloudera Search support for indexing? Does it support searching images?

Cloudera Search uses the [Apache Tika](#) library for indexing many standard document formats. In addition, Cloudera Search supports indexing and searching Avro files and a wide variety of other file types such as log files, Hadoop Sequence Files, and CSV files. You can add support for indexing custom file formats using a morphline command plug-in.

Getting Support

This section describes how to get support.

Cloudera Support

Cloudera can help you install, configure, optimize, tune, and run CDH for large scale data processing and analysis. Cloudera supports CDH whether you run it on servers in your own datacenter or on hosted infrastructure services, such as Amazon Web Services, Microsoft Azure, or Google Compute Engine.

If you are a Cloudera customer, you can:

- Register for an account to create a support ticket at the [support site](#).
- Visit the [Cloudera Knowledge Base](#).

If you are not a Cloudera customer, learn how [Cloudera](#) can help you.

Information Required for Logging a Support Case

Before you log a support case, ensure you have either part or all of the following information to help Support investigate your case:

- If possible, provide a diagnostic data bundle following the instructions in [Collecting and Sending Diagnostic Data to Cloudera](#).
- For security issues, see [Logging a Security Support Case](#).
- Provide details about the issue such as what was observed and what the impact was.
- Provide any error messages that were seen, using screen capture if necessary & attach to the case.
- If you were running a command or performing a series of steps, provide the commands and the results, captured to a file if possible.
- Specify whether the issue took place in a new install or a previously-working cluster.
- Mention any configuration changes made in the follow-up to the issue being seen.
- Specify the type of release environment the issue is taking place in, such as sandbox, development, or production.
- The severity of the impact and whether it is causing outage.

Community Support

There are several vehicles for community support. You can:

- Register for the [Cloudera forums](#).
- If you have any questions or comments about CDH, you can visit the [Using the Platform forum](#).
- If you have any questions or comments about Cloudera Manager, you can
 - Visit the [Cloudera Manager forum](#) forum.
 - Cloudera Express users can access the Cloudera Manager support mailing list from within the Cloudera Manager Admin Console by selecting **Support > Mailing List**.
 - Cloudera Enterprise customers can access the [Cloudera Support Portal](#) from within the Cloudera Manager Admin Console, by selecting **Support > Cloudera Support Portal**. From there you can register for a support account, create a support ticket, and access the Cloudera Knowledge Base.
- If you have any questions or comments about Cloudera Navigator, you can visit the [Cloudera Navigator forum](#).
- Find more documentation for specific components by referring to [External Documentation](#) on page 34.

Get Announcements about New Releases

To get information about releases and updates for all products, visit the [Release Announcements](#) forum.

Report Issues

Your input is appreciated, but before filing a request:

- Search the [Cloudera issue tracker](#), where Cloudera tracks software and documentation bugs and enhancement requests for CDH.
- Search the [CDH Manual Installation](#), [Using the Platform](#), and [Cloudera Manager](#) forums.