cloudera®

Cloudera Introduction

Important Notice

Cloudera, Inc.
395 Page Mill Road
Palo Alto, CA 94306
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: Cloudera Enterprise 6.2.x
Date: August 2, 2021

Table of Contents

Overview of Cloudera and the Cloudera Documentation Set

Important: As of February 1, 2021, all downloads of CDH and Cloudera Manager require a username and password and use a modified URL. You must use the modified URL, including the username and password when downloading the repository contents described below. You may need to upgrade Cloudera Manager to a newer version that uses the modified URLs.

This can affect new installations, upgrades, adding new hosts to a cluster, downloading a new parcel, and adding a cluster.

For more information, see Updating an existing CDH/Cloudera Manager deployment to access downloads with authentication.

Cloudera Enterprise is a modern platform for machine learning and analytics, optimized for the cloud to be:

- **Unified**

  Bring your data warehouse, data science, data engineering, and operational database workloads together on a single integrated platform. The Cloudera Shared Data Experience (SDX) enables these diverse analytic processes to operate against a shared data catalog that preserves business context like security and governance policies and schema. This common services framework persists even in transient cloud environments and makes it easier for IT departments to set and enforce policies while enabling business access to self-service analytics.

- **Hybrid**

  Work where and how it's most convenient, affordable, and effective. Cloudera Enterprise can read directly from and write directly to cloud object stores like Amazon S3 (AWS S3) and Azure Data Lake Store (Microsoft ADLS) as well as on-premises storage environments, or HDFS and Kudu on IaaS (infrastructure as a service). This provides flexibility to work on the data that you want wherever it lives, with no copies and no moves. Cloudera Enterprise also provides the most popular data warehouse and machine learning engines that can run on any compute resource for ultimate deployment flexibility. Cloudera hybrid control means users can self-service by way of a PaaS (platform as a service) offering, or choose more options to configure and manage the platform by way of an IaaS offering, private cloud, or an on-premises deployment.

- **Enterprise-grade**

  Cloudera Enterprise is where the scale and performance required for today's modern data workloads meets the security and governance demanded by today's IT departments. This modern platform makes it easy to bring more users -- thousands -- to petabytes of diverse data and provides industry-leading engines to process and query data, and develop and serve data models quickly. The platform also provides several layers of fine-grained security and complete audit capability that prevents unauthorized data access and demonstrates accountability for actions taken.

Cloudera Enterprise provides the following solutions:

- Data Warehouse
- Data Science
- Data Engineering
- Operational Database
- Run Everything in the Cloud, Multi-Cloud, or on a Hybrid "Cloud / On-Premises" Deployment

This Getting Started guide provides a general overview of Cloudera enterprise solutions and their documentation. The same set of integrated enterprise products and tools offered for on-premises deployments are also offered in the cloud with Cloudera Altus.

Data Warehouse

Cloudera's modern Data Warehouse powers high-performance BI and data warehousing in both on-premises deployments and as a cloud service. Business users can explore and iterate on data quickly, run new reports and workloads, or access interactive dashboards without assistance from the IT department. In addition, IT can eliminate the inefficiencies of "data silos" by consolidating data marts into a scalable analytics platform to better meet business needs. With its open architecture, data can be accessed by more users and more tools, including data scientists and data engineers, providing more value at a lower cost.

POWERED BY…	
Apache Impala	Distributed interactive SQL query engine for BI and SQL analytics on data in cloud object stores (AWS S3, Microsoft ADLS), Apache Kudu (for updating data), or on HDFS.
Apache Hive on Spark	Provides the fastest ETL/ELT at scale so you can prepare data for BI and reporting.
SQL Development Workbench (HUE)	Supports thousands of SQL developers, running millions of queries each week.
Workload XM	Provides unique insights on workloads to support predictable offloads, query analysis and optimizations, and efficient utilization of cluster resources.
Cloudera Navigator	Enables trusted data discovery and exploration, and curation based on usage needs.

Data Science

Only Cloudera offers a modern enterprise platform, tools and expert guidance to help you unlock business value with machine learning and AI. Cloudera's modern platform for machine learning and analytics, optimized for the cloud, lets you build and deploy AI solutions at scale, efficiently and securely, anywhere you want. Cloudera Fast Forward Labs expert guidance helps you realize your AI future, faster.

POWERED BY…	
Cloudera Data Science Workbench (CDSW)	Accelerate data science from research to production on a collaborative platform for machine learning and AI. CDSW provides on-demand access to runtimes for R, Python, and Scala, plus high-performance integration with Apache Spark with secure connectivity to CDH. For deep learning and other demanding data science techniques, CDSW supports GPU-accelerated computing, so data scientists can use deep learning frameworks like TensorFlow, Apache MXNet, Keras, and more.
Apache Spark	Provides flexible, in-memory data processing, reliable stream processing, and rich machine learning tooling for Hadoop.
Cloudera Fast Forward Labs	Cloudera Fast Forward Labs helps you design and execute your enterprise machine learning strategy, enabling rapid, practical application of emerging machine learning technologies to your business. In addition, Cloudera Professional Services offer proven delivery of scalable, production-grade machine learning systems.

Data Engineering

Cloudera Data Engineering supports the foundational workloads of your big data journey: Fast and flexible ETL data processing workloads, and workloads that train machine learning models at scale. These workloads can be deployed on-premises or in the cloud.

POWERED BY…	
Apache Spark, Spark Streaming, Spark MLlib, Spark SQL, and Hive on Spark	Cloudera offers a modern platform for fast, flexible data processing of batch, real-time, and streaming workloads. Utilizing Apache Spark, which ingests all data, performs analytics on it, and then writes out data to the disk in one operation, advanced processing jobs can be completed in times that are significantly faster than traditional technology.
Altus Data Engineering	Cloudera Enterprise is the comprehensive platform for data science and data engineering in the public cloud whether users are launching multiple workloads in a multi-tenant environment or designing jobs that leverage cloud infrastructure for specific job like ETL and exploratory data science.
Workload XM	Provides unique insights on workloads to support predictable offloads, query analysis and optimizations, and efficient utilization of cluster resources.
Cloudera Navigator	Provides governance and data management, including auditing, lineage, discovery, and policy lifecycle management.
Cloudera Navigator Encrypt & Key Trustee	Provides transparent encryption of data at rest without requiring changes to your application and advanced key management.
HDFS, YARN, MapReduce, Hive, Pig, HUE, Sentry, Flume, Sqoop, Oozie, Kafka, Cloudera Manager, and Cloudera Altus Director	Provides the basic Hadoop platform, management tools, and cloud deployment tools that supports data engineering workloads on-premises and in the cloud.

Operational Database

Cloudera's operational database delivers a secure, low-latency, high-concurrency experience that can extract the insights in real-time that you need from constantly changing data. Operational database brings together and processes more data of all types from more sources, including IoT, to drive business insights within a single platform designed for web scale. Real-time, batch, and interactive processing frameworks give developers a variety of tools to ensure they deliver the value your business is looking for. As data sets, data-driven applications, and data users grow, Cloudera's operational database offers linear scalability in performance at a manageable cost.

POWERED BY…	
Apache Spark	Provides flexible, in-memory data processing, reliable stream processing, and rich machine learning tooling for Hadoop.
Apache Kudu	Kudu is Hadoop-native storage for fast analytics on fast data. It complements the capabilities of HDFS and HBase by providing a simplified architecture for building real-time analytic applications. It is designed to take advantage of next-generation hardware developments from Intel for even faster analytic performance. Combined with Apache Impala, they provide a high-performance analytic database solution; however, Kudu integrates with other frameworks within Cloudera Enterprise.
Apache HBase	Provides a high performance, NoSQL database built on Hadoop. Similar to HDFS, it offers flexible data storage to store any type of data in any format. HBase is designed for fast, random read/write access and can be used for real-time data serving when you have many users who need low-latency read/write capabilities. It can also be used for real-time data capture and analysis due to its semi-structured row format, high performance, and

POWERED BY…	
	its ability to store all raw and refined data. Finally, since HBase is an integrated part of the Cloudera Enterprise platform, you can manage it with Cloudera Manager and it includes security features (including table, column, and cell-level security) that make it compliance-ready.

Run Everything in the Cloud, Multi-Cloud, or on a Hybrid "Cloud / On-Premises" Deployment

Public clouds present a compelling opportunity to make analytics more agile and self-service. However, to reduce risk and costs, it makes sense to pursue hybrid- and multi-cloud environments. Cloudera Enterprise complements public cloud services and preserves your ability to pick and choose. Our solutions offer easy job-focused features and enterprise-grade qualities like unified security and governance. In addition, our cloud solutions efficiently deliver machine learning and analytic capabilities that you can use to leverage the power of your data.

POWERED BY…	
Cloudera Altus and Cloudera Data Engineering	Provides a platform-as-a-service (PaaS) for machine learning and analytics on Amazon Web Services and Microsoft Azure. Targets foundational data processing jobs like ETL and pipeline development, enabling data engineers to focus on their jobs while removing the burden of cluster management.
Cloudera Altus Director	Provision and manage cloud environments for Data Engineering, Data Warehouse, Operational Database, or run CDSW in the cloud. The Cloudera Shared Data Experience provides unified and persistent controls for the data catalog, governance, and security both on-premises and in multiple clouds.

Documentation Overview

The following guides are included in the Cloudera enterprise documentation set:

Guide	Description
Getting Started	Provides an introduction to Cloudera solutions and their associated documentation. Also includes a section describing how to create a Proof-of-Concept Installation where you can test applications before you deploy.
Enterprise Release Guide	Comprehensive release notes, requirements, supported versions, packaging and download information, and deprecated items for the Cloudera enterprise solutions.
Installation	Provides instructions for installing Cloudera software, including Cloudera Manager, CDH, and other managed services.
Upgrade	Provides a complete upgrade guide for upgrading CDH and all the supporting platform software, such as the operating system, the JDK, and underlying databases.
Cluster Management	Describes how to configure and manage clusters in a Cloudera enterprise deployment using Cloudera Manager. In addition, this guide shows you how to use Cloudera Manager to monitor the health of your Cloudera deployment, diagnose issues as they occur, and use/view logs and reports to troubleshoot issues related to configuration, operation, and compliance.
Security	Provides information about securing your cluster by using data encryption, user authentication, and authorization techniques.

Guide	Description
Governance and Metadata Management	Provides information about using Cloudera Navigator Data Management for comprehensive data governance, compliance, data stewardship, and other data management tasks.
Component Guides	Provides how-to and best practice information for the CDH components: • Interactive SQL Engine (Apache Impala) • Run workloads on fast-changing data (Apache Kudu) • Intelligent SQL workbench (HUE) • Operational database (Apache HBase, Apache Spark) • Rapid ETL/ELT processing (Apache Hive on Spark) • Optimize search & discovery (Apache Solr) • Ingest data (Apache Flume) and schedule jobs (Apache Oozie) • Highly performant streaming message platform (Apache Kafka) • File formats and compression

Cloudera Primary User Personas

Cloudera has defined the following set of personas described in this topic. These personas are characters based on real people, where each persona represents a user type. This collection of personas helps define the goals and activities of typical users of Cloudera products. Defining personas for software products is a *moving target* because user types change over time. This collection is the result of a 2018 study collecting data from about fifteen leaders in Cloudera product management and engineering. These primary personas are being validated with some customers to ensure their accuracy and will be updated as needed.

Infrastructure

The personas in this group use either Cloudera Manager or Altus to manage CDH clusters on-premises or in the cloud.

Jim — Senior Hadoop Administrator

Skills and Background

- Very strong knowledge of HDFS and Linux administration
- Understanding of:
  - Distributed/grid computing
  - VMs and their capabilities
  - Racks, disk topologies, and RAID
  - Hadoop architecture
- Proficiency in Java

Tools:

Cloudera

- Cloudera Manager/CDH
- Navigator
- BDR
- Workload XM

Third-party Tools: Configuration management tools, log monitoring tools, for example, Splunk, Puppet, Chef, Ganglia, or Grafana

Goals:

- Achieve consistent high availability and performance on Hadoop clusters
- User administration, including creating new users and updating access control rights upon demand

Typical Tasks:

- Monitor cluster performance to ensure high percentage up time
- Back up and replicate appropriate files to ensure disaster recovery

- Schedule and perform cluster upgrades
- Security: enable and check status of security services and configurations
- Analyze query performance with Workload XM to ensure optimum cluster performance
- Provision new clusters

Jen — Junior Hadoop Administrator

Skills and Background

- Basic knowledge of HDFS
- Limited knowledge of Linux (shell scripting mostly)
- General understanding of:
  - Distributed/grid computing
  - VMs and their capabilities
  - Racks, disk topologies, and RAID
  - Hadoop architecture

Tools:

Cloudera

- Cloudera Manager/CDH
- Navigator
- Workload XM

Third-party Tools: Configuration management tools, log monitoring tools, for example, Splunk, Puppet, Chef, Ganglia, or Grafana

Goals:

- Maintain high availability and performance of Hadoop clusters

Typical Tasks:

- Perform basic procedures to ensure clusters are up and running
- Perform maintenance work flows

Sarah — Cloud Administrator

Skills and Background

- Understands public cloud primitives (Virtual Private Cloud)

- Understands security access policies (Identity Access Management)
- Proficiency in Java

Tools:

Cloudera

- Altus

Third-party Tools: Amazon Web Services, Microsoft Azure

Goals:

- Maintain correct access to cloud resources
- Maintain correct resource allocation to cloud resources, such as account limits

Typical Tasks:

- Create the Altus environment for the organization

Data Ingest, ETL, and Metadata Management

The personas in this group typically use Navigator, Workload XM, HUE, Hive, Impala, and Spark.

Terence — Enterprise Data Architect or Modeler

Skills and Background

- Experience with:
  - ETL process
  - Data munging
  - Wide variety of data wrangling tools

Tools:

Cloudera

- Navigator
- Workload XM
- HUE
- Hive
- Impala
- Spark

Third-party Tools: ETL and other data wrangling tools

Goals:

- Maintain organized/optimized enterprise data architecture to support the business needs
- Ensure that data models support improved data management and consumption
- Maintain efficient schema design

Typical Tasks:

- *Organize data at the macro level:* set architectural principles, create data models, create key entity diagrams, and create a data inventory to support business processes and architecture
- *Organize data at the micro level:* create data models for specific applications
- Map organization use cases to execution engines (Impala, Spark, Hive)
- Provide logical data models for the most important data sets, consuming applications, and data quality rules
- Provide data entity descriptions
- *Ingest new data into the system:* use ingest tools, monitor ingestion rate, data formatting, and partitioning strategies

Kara — Data Steward and Data Curator

Skills and Background

- Experience with:
  - ETL process
  - Data wrangling tools

Tools:

Cloudera

- Navigator
- HUE data catalog

Third-party Tools: ETL and other data wrangling tools

Goals:

- Maintain metadata (technical and custom)
- Maintain data policies to support business processes
- Maintain data lifecycle at Hadoop scale
- Maintain data access permissions

Typical Tasks:

- Manage technical metadata
- Classify data at Hadoop scale
- Create and manage custom and business metadata using policies or third-party tools that integrate with Navigator

Analytics and Machine Learning

The personas in this group typically use Cloudera Data Science Workbench (CDSW), HUE, HDFS, and HBase.

Song — Data Scientist

Skills and Background

- Statistics
- Related scripting tools, for example R
- Machine learning models
- SQL
- Basic programming

Tools:

Cloudera

- CDSW
- HUE to build and test queries before adding to CDSW
- HDFS
- HBase

Third-party Tools: R, SAS, SPSS, and others. Command-line scripting languages such as Scala, Python, Tableau, Qlik, and some Java

Goals:

- Solve business problems by applying advanced analytics and machine learning in an ad hoc manner

Typical Tasks:

- Access, explore, and prepare data by joining and cleaning it
- Define data features and variables to solve business problems as in data feature engineering
- Select and adapt machine learning models or write algorithms to answer business questions
- Tune data model features and hyper parameters while running experiments
- Publish the optimized model for wider use as an API for BI Analysts or Data Owners to use as part of their reporting
- Publish data model results to answer business questions for consumption by Data Owners and BI Analysts

Jason — Machine Learning Engineer

Skills and Background

- Machine learning and big data skills
- Software engineering

Tools:

Cloudera

- Spark
- HUE to build and test queries before adding to application
- CDSW

Third-party Tools: Java

Goals:

- Build and maintain production machine learning applications

Typical Tasks:

- Set up big data machine learning projects at companies such as Facebook

Cory — Data Engineer

Skills and Background

- Software engineering
- SQL mastery
- ETL design and big data skills
- Machine learning skills

Tools:

Cloudera

- CDSW
- Spark/MapReduce
- Hive
- Oozie
- Altus Data Engineering
- HUE
- Workload XM

Third-party Tools: IDE, Java, Python, Scala

Goals:

- Create data pipelines (about 40% of working time)
- Maintain data pipelines (about 60% of working time)

Typical Tasks:

- Create data workflow paths
- Create code repository check-ins
- Create XML workflows for production system launches

Sophie — Application Developer

Skills and Background

- Deep knowledge of software engineering to build real-time applications

Tools:

Cloudera

- HBase

Third-party Tools: Various software development tools

Goals:

- Applications developed run and successfully send workloads to the cluster. For example, connects a front-end to HBase on the cluster.

Typical Tasks:

- Develops application features, but does not write the SQL workload. Rather writes the application that sends the workloads to the cluster.
- Tests applications to ensure they run successfully

Abe — SQL Expert/SQL Developer

Skills and Background

- Deep knowledge of SQL dialects and schemas

Tools:

Cloudera

- HUE
- Cloudera Manager to monitor Hive queries
- Hive via command line or HUE
- Impala via HUE, another BI tool, or the command line
- Navigator via HUE
- Sentry via HUE
- Workload XM via HUE

Third-party Tools: SQL Studio, TOAD

Goals:

- Create workloads that perform well and that return the desired results

Typical Tasks:

- Create query workloads that applications send to the cluster
- Ensure optimal performance of query workloads by monitoring the query model and partitioning strategies
- Prepare and test queries before they are added to applications

Kiran — SQL Analyst/SQL User

Skills and Background

- Has high-level grasp of SQL concepts, but prefers to drag and drop query elements
- Good at data visualization, but prefers pre-populated tables and queries

Tools:

Cloudera

- HUE
- Cloudera Manager to monitor queries
- Oozie to schedule workloads
- Impala (rather than Hive)

Third-party Tools: Reporting and business intelligence tools like Cognos, Crystal Reports

Goals:

- To answer business questions and problems based on data

Typical Tasks:

- Create query workloads that applications send to the cluster
- Ensure optimal performance of queries (query model, partitioning strategies)

Christine — BI Analyst

Skills and Background

- Ability to:
  - View reports and drill down into results of interest
  - Tag, save, share reports and results

Tools:

Cloudera

- HUE
- Navigator via HUE

Third-party Tools: SQL query tools, Tableau, Qlik, Excel

Goals:

- Apply data preparation and analytic skills to solve recurrent business problems. For example, to create a weekly sales report.
- Provide reports for the Business/Data Owner

Typical Tasks:

- Access, explore, and prepare data by joining and cleaning it
- Create reports to satisfy requests from business stakeholders to solve business problems

Reference Architectures

Reference architectures illustrate sample cluster configurations and certified partner products. They are not replacements for official statements of supportability, rather they're guides to assist with deployment and sizing options. Statements regarding supported configurations in the RAs are informational and should be cross-referenced with the latest documentation.

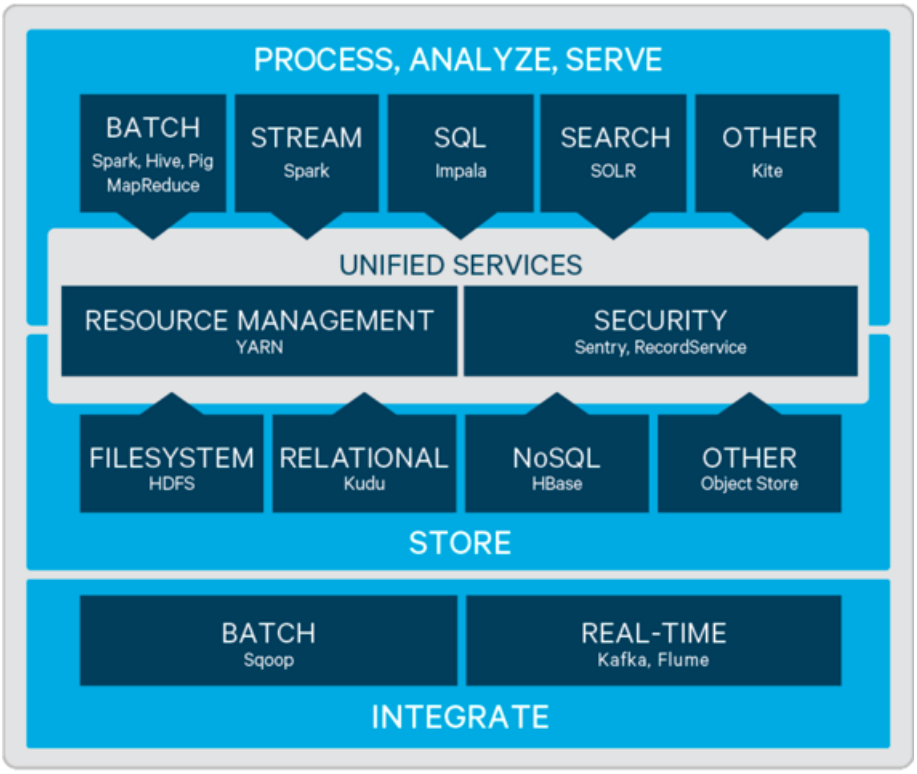
To view the reference architectures, see Cloudera Reference Architectures.

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 28.

Apache Hive Overview in CDH

Hive data warehouse software enables reading, writing, and managing large datasets in distributed storage. Using the Hive query language (HiveQL), which is very similar to SQL, queries are converted into a series of jobs that execute on a Hadoop cluster through MapReduce or Apache Spark.

Users can run batch processing workloads with Hive while also analyzing the same data for interactive SQL or machine-learning workloads using tools like Apache Impala or Apache Spark—all within a single platform.

As part of CDH, Hive also benefits from:

- Unified resource management provided by YARN
- Simplified deployment and administration provided by Cloudera Manager
- Shared security and governance to meet compliance requirements provided by Apache Sentry and Cloudera Navigator

Use Cases for Hive

Because Hive is a petabyte-scale data warehouse system built on the Hadoop platform, it is a good choice for environments experiencing phenomenal growth in data volume. The underlying MapReduce interface with HDFS is hard to program directly, but Hive provides an SQL interface, making it possible to use existing programming skills to perform data preparation.

Hive on MapReduce or Spark is best-suited for batch data preparation or ETL:

- You must run scheduled batch jobs with very large ETL sorts with joins to prepare data for Hadoop. Most data served to BI users in Impala is prepared by ETL developers using Hive.
- You run data transfer or conversion jobs that take many hours. With Hive, if a problem occurs partway through such a job, it recovers and continues.
- You receive or provide data in diverse formats, where the Hive SerDes and variety of UDFs make it convenient to ingest and convert the data. Typically, the final stage of the ETL process with Hive might be to a high-performance, widely supported format such as Parquet.

Hive Components

Hive consists of the following components:

The Metastore Database

The metastore database is an important aspect of the Hive infrastructure. It is a separate database, relying on a traditional RDBMS such as MySQL or PostgreSQL, that holds metadata about Hive databases, tables, columns, partitions, and Hadoop-specific information such as the underlying data files and HDFS block locations.

The metastore database is shared by other components. For example, the same tables can be inserted into, queried, altered, and so on by both Hive and Impala. Although you might see references to the “Hive metastore”, be aware that the metastore database is used broadly across the Hadoop ecosystem, even in cases where you are not using Hive itself.

The metastore database is relatively compact, with fast-changing data. Backup, replication, and other kinds of management operations affect this database. See Configuring the Hive Metastore for CDH for details about configuring the Hive metastore.

Cloudera recommends that you deploy the Hive metastore, which stores the metadata for Hive tables and partitions, in "remote mode." In this mode the metastore service runs in its own JVM process and other services, such as HiveServer2, HCatalog, and Apache Impala communicate with the metastore using the Thrift network API.

See Starting the Hive Metastore in CDH for details about starting the Hive metastore service.

HiveServer2

HiveServer2 is a server interface that enables remote clients to submit queries to Hive and retrieve the results. HiveServer2 supports multi-client concurrency, capacity planning controls, Sentry authorization, Kerberos authentication, LDAP, SSL, and provides support for JDBC and ODBC clients.

HiveServer2 is a container for the Hive execution engine. For each client connection, it creates a new execution context that serves Hive SQL requests from the client. It supports JDBC clients, such as the Beeline CLI, and ODBC clients. Clients connect to HiveServer2 through the Thrift API-based Hive service.

See Configuring HiveServer2 for CDH for details on configuring HiveServer2 and see Starting, Stopping, and Using HiveServer2 in CDH for details on starting/stopping the HiveServer2 service and information about using the Beeline

CLI to connect to HiveServer2. For details about managing HiveServer2 with its native web user interface (UI), see Using HiveServer2 Web UI in CDH.

How Hive Works with Other Components

Hive integrates with other components, which serve as query execution engines or as data stores:

Hive on Spark

Hive traditionally uses MapReduce behind the scenes to parallelize the work, and perform the low-level steps of processing a SQL statement such as sorting and filtering. Hive can also use Spark as the underlying computation and parallelization engine. See Running Apache Hive on Spark in CDH for details about configuring Hive to use Spark as its execution engine and see Tuning Apache Hive on Spark in CDH for details about tuning Hive on Spark.

Hive and HBase

Apache HBase is a NoSQL database that supports real-time read/write access to large datasets in HDFS. See Using Apache Hive with HBase in CDH for details about configuring Hive to use HBase. For information about running Hive queries on a secure HBase server, see Using Hive to Run Queries on a Secure HBase Server.

Hive on Amazon S3

Use the Amazon S3 filesystem to efficiently manage transient Hive ETL (extract-transform-load) jobs. For step-by-step instructions to configure Hive to use S3 and multiple scripting examples, see Configuring Transient Hive ETL Jobs to Use the Amazon S3 Filesystem. To optimize how Hive writes data to and reads data from S3-backed tables and partitions, see Tuning Hive Performance on the Amazon S3 Filesystem. For information about setting up a shared Amazon Relational Database Service (RDS) as your Hive metastore, see Configuring a Shared Amazon RDS as an HMS for CDH.

Hive on Microsoft Azure Data Lake Store

In CDH 5.12 and higher, both Hive on MapReduce2 and Hive on Spark can access tables on Microsoft Azure Data Lake store (ADLS). In contrast to Amazon S3, ADLS more closely resembles native HDFS behavior, providing consistency, file directory structure, and POSIX-compliant ACLs. See Configuring ADLS Gen1 Connectivity for information about configuring and using ADLS with Hive on MapReduce2.

Impala

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 graduated from the Apache Incubator on November 15, 2017. In places where the documentation formerly referred to “Cloudera Impala”, now the official name is “Apache Impala”.

Impala Benefits

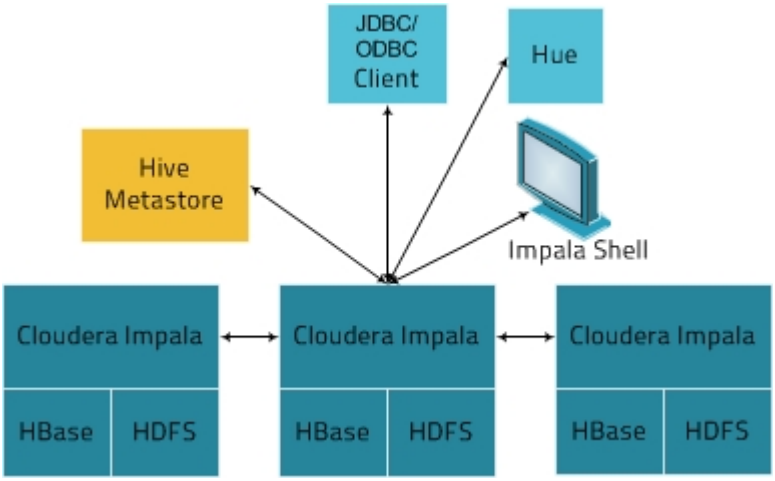
Impala provides:

- 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

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.

Apache Kudu Overview

Apache Kudu is a columnar storage manager developed for the Hadoop platform. Kudu shares the common technical properties of Hadoop ecosystem applications: It runs on commodity hardware, is horizontally scalable, and supports highly available operation.

Apache Kudu is a top-level project in the Apache Software Foundation.

Kudu's benefits include:

- Fast processing of OLAP workloads.
- Integration with MapReduce, Spark, Flume, and other Hadoop ecosystem components.
- Tight integration with Apache Impala, making it a good, mutable alternative to using HDFS with Apache Parquet.
- Strong but flexible consistency model, allowing you to choose consistency requirements on a per-request basis, including the option for strict serialized consistency.
- Strong performance for running sequential and random workloads simultaneously.
- Easy administration and management through Cloudera Manager.
- High availability. Tablet Servers and Master use the Raft consensus algorithm, which ensures availability as long as more replicas are available than unavailable. Reads can be serviced by read-only follower tablets, even in the event of a leader tablet failure.
- Structured data model.

By combining all of these properties, Kudu targets support applications that are difficult or impossible to implement on currently available Hadoop storage technologies. Applications for which Kudu is a viable solution include:

- Reporting applications where new data must be immediately available for end users
- Time-series applications that must support queries across large amounts of historic data while simultaneously returning granular queries about an individual entity
- Applications that use predictive models to make real-time decisions, with periodic refreshes of the predictive model based on all historical data

Kudu-Impala Integration

Apache Kudu has tight integration with Apache Impala, allowing you to use Impala to insert, query, update, and delete data from Kudu tablets using Impala's SQL syntax, as an alternative to using the Kudu APIs to build a custom Kudu application. In addition, you can use JDBC or ODBC to connect existing or new applications written in any language, framework, or business intelligence tool to your Kudu data, using Impala as the broker.

- **`CREATE/ALTER/DROP TABLE`** - Impala supports creating, altering, and dropping tables using Kudu as the persistence layer. The tables follow the same internal/external approach as other tables in Impala, allowing for flexible data ingestion and querying.
- **`INSERT`** - Data can be inserted into Kudu tables from Impala using the same mechanisms as any other table with HDFS or HBase persistence.
- **`UPDATE/DELETE`** - Impala supports the `UPDATE` and `DELETE` SQL commands to modify existing data in a Kudu table row-by-row or as a batch. The syntax of the SQL commands is designed to be as compatible as possible with existing solutions. In addition to simple `DELETE` or `UPDATE` commands, you can specify complex joins in the `FROM` clause of the query, using the same syntax as a regular `SELECT` statement.
- **Flexible Partitioning** - Similar to partitioning of tables in Hive, Kudu allows you to dynamically pre-split tables by hash or range into a predefined number of tablets, in order to distribute writes and queries evenly across your

cluster. You can partition by any number of primary key columns, with any number of hashes, a list of split rows, or a combination of these. A partition scheme is required.

- **Parallel Scan** - To achieve the highest possible performance on modern hardware, the Kudu client used by Impala parallelizes scans across multiple tablets.
- **High-efficiency queries** - Where possible, Impala pushes down predicate evaluation to Kudu, so that predicates are evaluated as close as possible to the data. Query performance is comparable to Parquet in many workloads.

Example Use Cases

Streaming Input with Near Real Time Availability

A common business challenge is one where new data arrives rapidly and constantly, and the same data needs to be available in near real time for reads, scans, and updates. Kudu offers the powerful combination of fast inserts and updates with efficient columnar scans to enable real-time analytics use cases on a single storage layer.

Time-Series Application with Widely Varying Access Patterns

A time-series schema is one in which data points are organized and keyed according to the time at which they occurred. This can be useful for investigating the performance of metrics over time or attempting to predict future behavior based on past data. For instance, time-series customer data might be used both to store purchase click-stream history and to predict future purchases, or for use by a customer support representative. While these different types of analysis are occurring, inserts and mutations might also be occurring individually and in bulk, and become available immediately to read workloads. Kudu can handle all of these access patterns simultaneously in a scalable and efficient manner.

Kudu is a good fit for time-series workloads for several reasons. With Kudu's support for hash-based partitioning, combined with its native support for compound row keys, it is simple to set up a table spread across many servers without the risk of "hotspotting" that is commonly observed when range partitioning is used. Kudu's columnar storage engine is also beneficial in this context, because many time-series workloads read only a few columns, as opposed to the whole row.

In the past, you might have needed to use multiple datastores to handle different data access patterns. This practice adds complexity to your application and operations, and duplicates your data, doubling (or worse) the amount of storage required. Kudu can handle all of these access patterns natively and efficiently, without the need to off-load work to other datastores.

Predictive Modeling

Data scientists often develop predictive learning models from large sets of data. The model and the data might need to be updated or modified often as the learning takes place or as the situation being modeled changes. In addition, the scientist might want to change one or more factors in the model to see what happens over time. Updating a large set of data stored in files in HDFS is resource-intensive, as each file needs to be completely rewritten. In Kudu, updates happen in near real time. The scientist can tweak the value, re-run the query, and refresh the graph in seconds or minutes, rather than hours or days. In addition, batch or incremental algorithms can be run across the data at any time, with near-real-time results.

Combining Data In Kudu With Legacy Systems

Companies generate data from multiple sources and store it in a variety of systems and formats. For instance, some of your data might be stored in Kudu, some in a traditional RDBMS, and some in files in HDFS. You can access and query all of these sources and formats using Impala, without the need to change your legacy systems.

Related Information

- Apache Kudu Concepts and Architecture
- Apache Kudu Installation and Upgrade
- Kudu Security Overview
- More Resources for Apache Kudu

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

> **Note:**
>
> This page contains information related to Spark 2.x, which is included with CDH beginning with CDH 6. This information supercedes the documentation for the separately available parcel for CDS Powered By Apache Spark.

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.

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. In CDH 6, Cloudera supports only the YARN cluster manager. When run on YARN, Spark application processes are managed by the YARN ResourceManager and NodeManager roles. Spark Standalone is no longer supported.

For detailed API information, see the Apache Spark project site.

> **Note:** Although this document makes some references to the external Spark site, not all the features, components, recommendations, and so on are applicable to Spark when used on CDH. Always cross-check the Cloudera documentation before building a reliance on some aspect of Spark that might not be supported or recommended by Cloudera. In particular, see Apache Spark Known Issues for components and features to avoid.

The Apache Spark 2 service in CDH 6 consists of Spark core and several related projects:

Module for working with structured data. Allows you to seamlessly mix SQL queries with Spark programs.

API that allows you to build scalable fault-tolerant streaming applications.

API that implements common machine learning algorithms.

The Cloudera Enterprise product includes the Spark features roughly corresponding to the feature set and bug fixes of Apache Spark 2.4. The Spark 2.x service was previously shipped as its own parcel, separate from CDH.

In CDH 6, the Spark 1.6 service does not exist. The port of the Spark History Server is 18088, which is the same as formerly with Spark 1.6, and a change from port 18089 formerly used for the Spark 2 parcel.

Unsupported Features

The following Spark features are not supported:

- Apache Spark experimental features/APIs are not supported unless stated otherwise.

- Using the JDBC Datasource API to access Hive or Impala is not supported
- ADLS not Supported for All Spark Components. Microsoft Azure Data Lake Store (ADLS) is a cloud-based filesystem that you can access through Spark applications. Spark with Kudu is not currently supported for ADLS data. (Hive on Spark is available for ADLS in CDH 5.12 and higher.)
- IPython / Jupyter notebooks is not supported. The IPython notebook system (renamed to Jupyter as of IPython 4.0) is not supported.
- Certain Spark Streaming features not supported. The `mapWithState` method is unsupported because it is a nascent unstable API.
- Thrift JDBC/ODBC server is not supported
- Spark SQL CLI is not supported
- GraphX is not supported
- SparkR is not supported
- Structured Streaming is supported, but the following features of it *are not*:
  - Continuous processing, which is still experimental, is not supported.
  - Stream static joins with HBase have not been tested and therefore are not supported.
- Spark cost-based optimizer (CBO) not supported.

Consult Apache Spark Known Issues for a comprehensive list of Spark 2 features that are not supported with CDH 6.

Related Information

- Managing Spark
- Monitoring Spark Applications
- Spark Authentication
- Spark Encryption
- Cloudera Spark forum
- Apache Spark documentation
-
-

External Documentation

Note: This page contains references to CDH 5 components or features that have been removed from CDH 6. These references are only applicable if you are managing a CDH 5 cluster with Cloudera Manager 6. For more information, see Deprecated Items.

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 Flume
- Apache Hadoop
- Apache HBase
- Apache Hive
- Hue
- Kite
- Apache Oozie

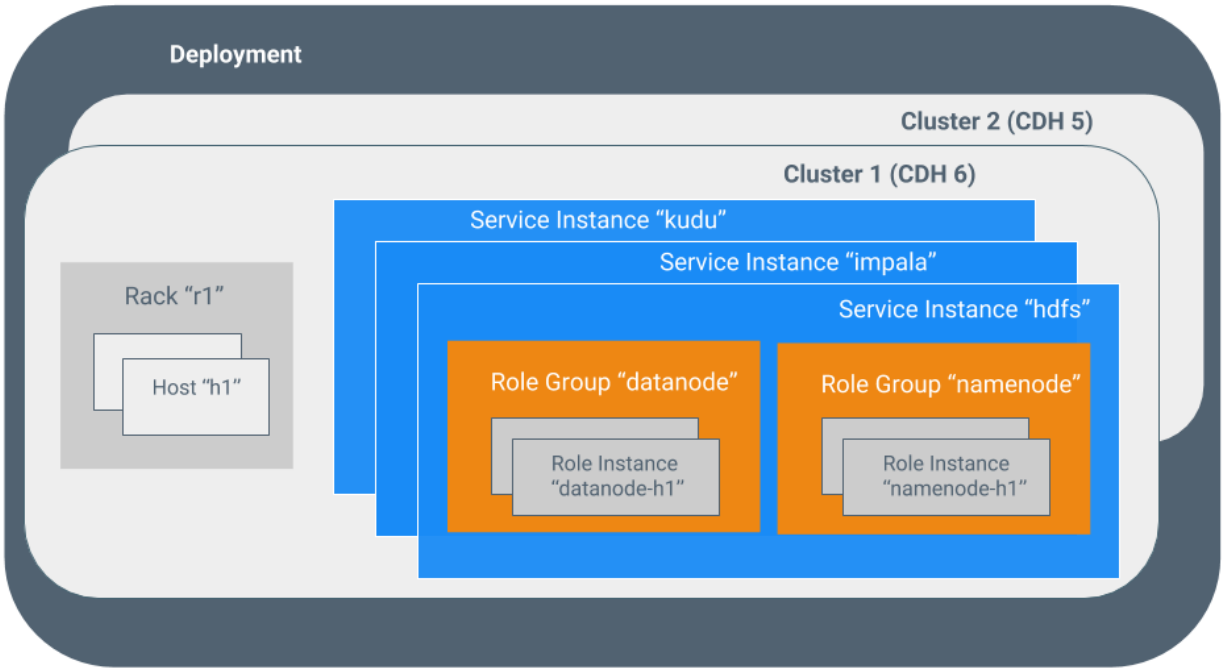
- Apache Parquet
- Apache Pig
- Apache Sentry
- Apache Solr
- Apache Spark
- Apache Sqoop
- Apache ZooKeeper

Cloudera Manager 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, are used without further explanation. Other terms, such as role group, gateway, host template, and parcel are explained in the below sections.

Sometimes the terms ***service*** and ***role*** are used to refer to both *types* and *instances*, which can be confusing. Cloudera Manager and this section sometimes use the same term for *type* and *instance*. For example, the Cloudera Manager Admin Console **Home** > **Status** tab and the **Clusters** > ***ClusterName*** menu list service instances. This is similar to the practice in programming languages where the term "string" might indicate a type (`java.lang.String`) or an instance of that type ("hi there"). Where it is 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

A type of role that typically provides client access to specific cluster services. For example, HDFS, Hive, Kafka, MapReduce, Solr, and Spark each have gateway roles to provide access for their clients to their respective services. Gateway roles

do not always have "gateway" in their names, nor are they exclusively for client access. For example, Hue Kerberos Ticket Renewer is a gateway role that proxies tickets from Kerberos.

The node supporting one or more gateway roles is sometimes referred to as the ***gateway node*** or ***edge node***, with the notion of "edge" common in network or cloud environments. In terms of the Cloudera cluster, the gateway nodes in the cluster receive the appropriate client configuration files when **Deploy Client Configuration** is selected from the Actions menu in Cloudera Manager Admin Console.

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:

Name	IP	Roles	Load Average	Disk Usage	Physical Memory	Swap Space
tcdn501-1.ent.cloudera.com	10.20.195.240	21 Role(s)	0.04 0.15 0.26	11.3 GiB / 57 GiB	6.3 GiB / 9.7 GiB	4.7 MiB / 2 GiB
tcdn501-2.ent.cloudera.com	10.20.81.81	7 Role(s)	0.07 0.07 0.05	8.9 GiB / 57 GiB	2 GiB / 9.7 GiB	0 B / 2 GiB
tcdn501-3.ent.cloudera.com	10.20.190.234	7 Role(s)	0.08 0.11 0.04	8.9 GiB / 57 GiB	2 GiB / 9.7 GiB	0 B / 2 GiB
tcdn501-4.ent.cloudera.com	10.20.195.243	7 Role(s)	0.06 0.23 0.23	8.9 GiB / 57 GiB	2 GiB / 9.7 GiB	0 B / 2 GiB

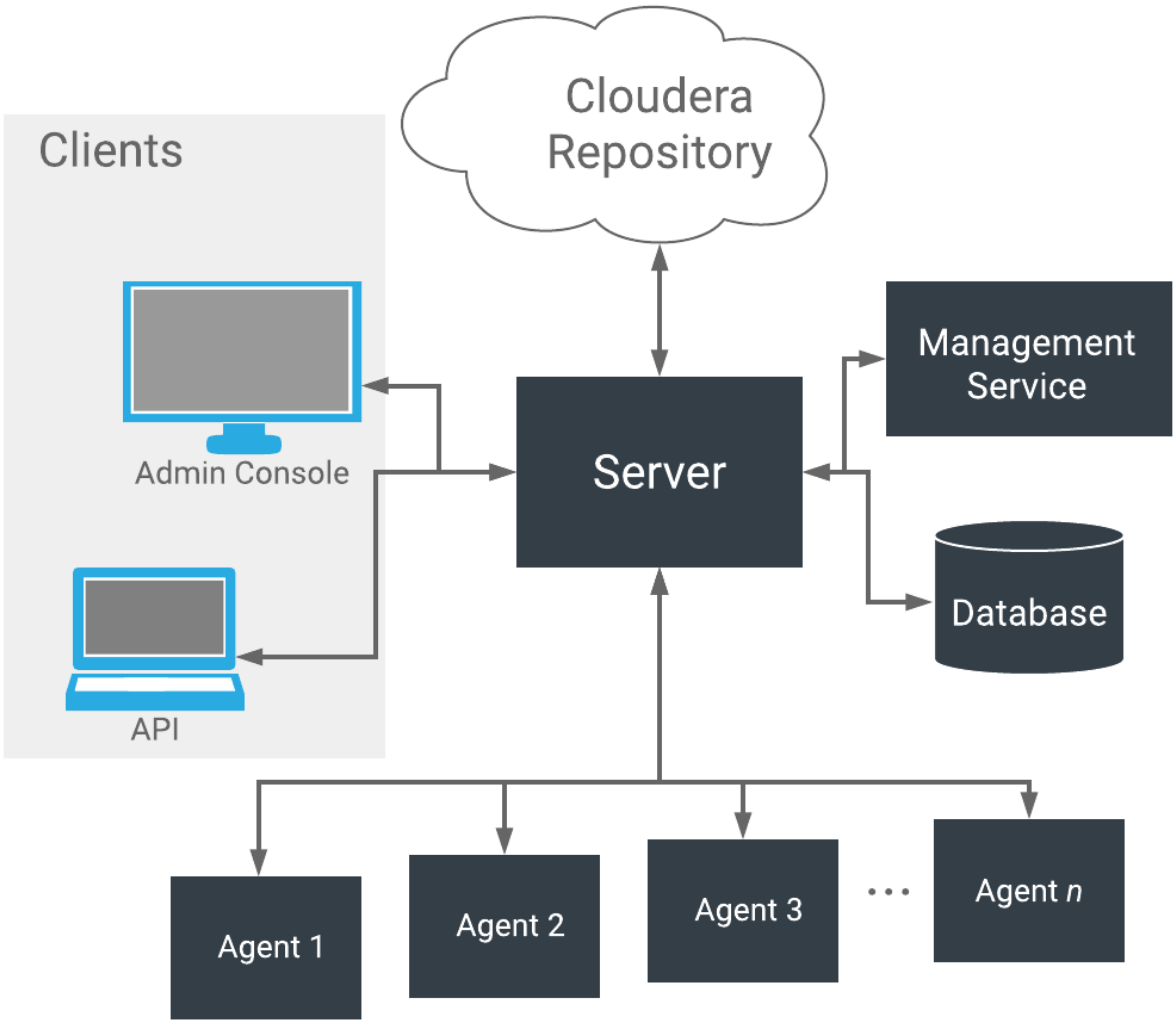
First | Previous | 1 | Next | Last

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.

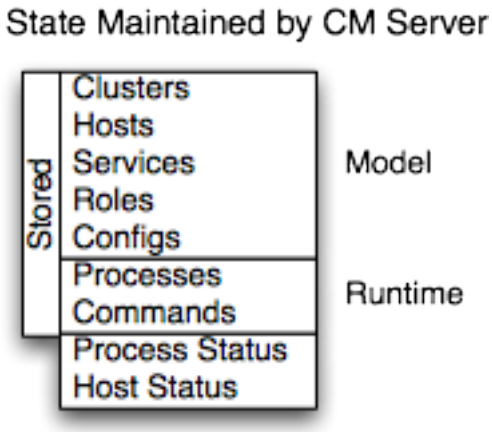
Heartbeating

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.



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 workaround, 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:

Role Name	State	Host	Role Group
Balancer	N/A	tcdn501-1.ent.cloudera.com	Balancer Default Group
DataNode	Started	tcdn501-2.ent.cloudera.com	DataNode Default Group
DataNode	Started	tcdn501-3.ent.cloudera.com	DataNode Default Group
DataNode	Started	tcdn501-4.ent.cloudera.com	DataNode Default Group
NameNode (Active)	Started	tcdn501-1.ent.cloudera.com	NameNode Default Group
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 contain 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:

Role Name	State	Host	Role Group
Gateway	N/A	tcdn501-2.ent.cloudera.com	Gateway Default Group
Gateway	N/A	tcdn501-3.ent.cloudera.com	Gateway Default Group
Gateway	N/A	tcdn501-4.ent.cloudera.com	Gateway Default Group
Gateway	N/A	tcdn501-1.ent.cloudera.com	Gateway Default Group
Hive Metastore Server	Started	tcdn501-1.ent.cloudera.com	Hive Metastore Server Default Group
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 and upgrade CDH and other managed services. 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 is no distinction between what is installed and what is active.
- Parcels are required for rolling upgrades.

- You can install parcels at any location in the filesystem. They are installed by default in `/opt/cloudera/parcels`. In contrast, packages are installed in `/usr/lib`.
- When you install from the **Parcels** page, Cloudera Manager 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 must run on the same major OS release to be covered by Cloudera Support. Cloudera Manager must run on the same major OS release as at least 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. For information about supported operating systems, see Operating System Requirements.

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 component of CDH, parcels are distributed as a single object. 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 components from different CDH versions.
- **Installation outside of `/usr`** - In some environments, Hadoop administrators do not have privileges to install system packages. With parcels, administrators can install to `/opt`, or anywhere else.

  > **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 can 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** - Using packages requires you to shut down the old process, upgrade the package, and then start the new process. Errors 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 5 to CDH 6) 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. LZO 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, such as 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

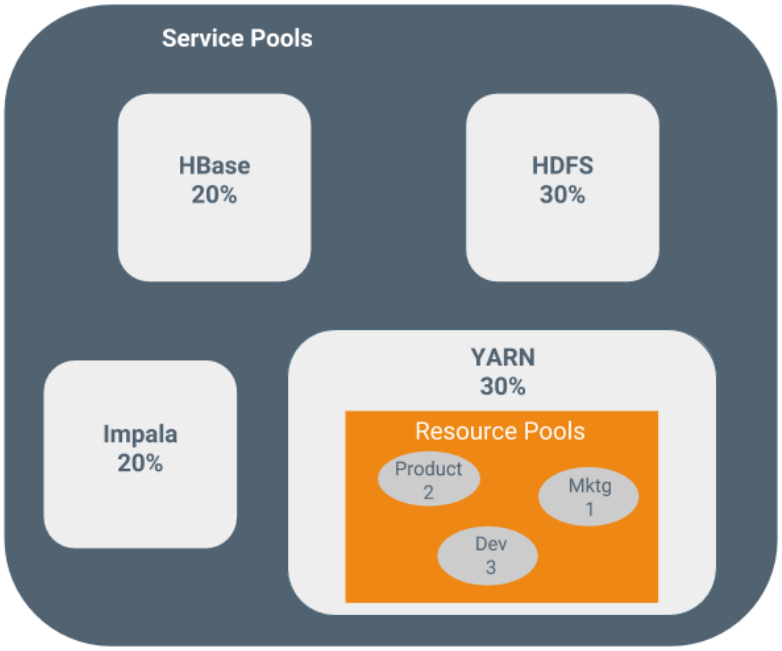
Resource management helps ensure predictable behavior by defining the impact of different services on cluster resources. Use resource management to:

- 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.

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** - YARN manages the virtual cores, memory, running applications, maximum resources for undeclared children (for parent pools), 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% x 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** - 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

Authentication is a process that requires users and services to prove their identity when trying to access a system resource. Organizations typically manage user identity and authentication through various time-tested technologies, including Lightweight Directory Access Protocol (LDAP) for identity, directory, and other services, such as group management, and Kerberos for authentication.

Cloudera clusters support integration with both of these technologies. For example, organizations with existing LDAP directory services such as Active Directory (included in Microsoft Windows Server as part of its suite of Active Directory Services) can leverage the organization's existing user accounts and group listings instead of creating new accounts throughout the cluster. Using an external system such as Active Directory or OpenLDAP is required to support the user role authorization mechanism implemented in Cloudera Navigator.

For authentication, Cloudera supports integration with MIT Kerberos and with Active Directory. Microsoft Active Directory supports Kerberos for authentication in addition to its identity management and directory functionality, that is, LDAP.

These systems are not mutually exclusive. For example, Microsoft Active Directory is an LDAP directory service that also provides Kerberos authentication services, and Kerberos credentials can be stored and managed in an LDAP directory service. Cloudera Manager Server, CDH nodes, and Cloudera Enterprise components, such as Cloudera Navigator, Apache Hive, Hue, and Impala, can all make use of Kerberos authentication.

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

Data at rest and data in transit encryption function at different technology layers of the cluster:

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 Cloudera Manager Software Management

A major function of Cloudera Manager is to install and upgrade CDH and other managed services. 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 is no distinction between what is installed and what is active.
- Parcels are required for rolling upgrades.
- You can install parcels at any location in the filesystem. They are installed by default in `/opt/cloudera/parcels`. In contrast, packages are installed in `/usr/lib`.
- When you install from the **Parcels** page, Cloudera Manager 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 must run on the same major OS release to be covered by Cloudera Support. Cloudera Manager must run on the same major OS release as at least 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. For information about supported operating systems, see Operating System Requirements.

Important: You cannot install software using both parcels and packages in the same cluster.

Parcels

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

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 is no distinction between what is installed and what is active.
- Parcels are required for rolling upgrades.
- You can install parcels at any location in the filesystem. They are installed by default in `/opt/cloudera/parcels`. In contrast, packages are installed in `/usr/lib`.
- When you install from the **Parcels** page, Cloudera Manager 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 must run on the same major OS release to be covered by Cloudera Support. Cloudera Manager must run on the same major OS release as at least 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. For information about supported operating systems, see Operating System Requirements.

Important: Cloudera Manager manages parcels without the need for users to manipulate parcels in the filesystem. You might cause failures or unexpected behaviors in your cluster if you perform any of the following unsupported actions:

- Installing parcels within custom RPM packages and saving them to the Cloudera Manager parcel directory.
- Downloading parcels and manually placing them in the Cloudera Manager parcel directory.
- Manually adding, modifying, or deleting files within the root parcels directory or its subdirectories.

For detailed installation instructions using parcels, and other methods, see Cloudera Installation Guide.

Parcels are available for CDH, for other managed services, and for Sqoop Connectors.

Important: You cannot install software using both parcels and packages in the same cluster.

Advantages of Parcels

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 component of CDH, parcels are distributed as a single object. 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 components from different CDH versions.
- **Installation outside of `/usr`** - In some environments, Hadoop administrators do not have privileges to install system packages. With parcels, administrators can install to `/opt`, or anywhere else.

  > **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 can 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** - Using packages requires you to shut down the old process, upgrade the package, and then start the new process. Errors 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 5 to CDH 6) 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. LZO 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, such as 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.

Parcel Life Cycle

To enable upgrades and additions with minimal disruption, parcels have following phases:

- **Downloaded** -The parcel software is copied to a local parcel directory on the Cloudera Manager Server, where it is available for distribution to other hosts in any of the clusters managed by this Cloudera Manager Server. You can have multiple parcels for a product downloaded to your Cloudera Manager Server. After a parcel has been downloaded to the Server, it is available for distribution on all clusters managed by the Server. A downloaded parcel appears in the cluster-specific section for every cluster managed by this Cloudera Manager Server.
- **Distributed** - The parcel is copied to the cluster hosts, and components of the parcel are unpacked. Distributing a parcel does not upgrade the components running on your cluster; the current services continue to run unchanged. You can have multiple parcels distributed on your cluster. Distributing parcels does not require Internet access; the Cloudera Manager Agent on each cluster member downloads the parcels from the local parcel repository on the Cloudera Manager Server.
- **Activated** - Links to the parcel components are created. Activation does not automatically stop the current services or perform a restart. You can restart services after activation, or the system administrator can determine when to perform those operations.

  If you are upgrading CDH or managed services when you activate a parcel, follow the instructions in Upgrading the CDH Cluster to complete the upgrade.

- **In Use** - The parcel components on the cluster hosts are in use when you start or restart the services that use those components.
- **Deactivated** - The links to the parcel components are removed from the cluster hosts.
- **Removed** - The parcel components are removed from the cluster hosts.
- **Deleted** - The parcel is deleted from the local parcel repository on the Cloudera Manager Server.

Cloudera Manager detects when new parcels are available. You can configure Cloudera Manager to download and distribute parcels automatically. See Configuring Cloudera Manager Server Parcel Settings on page 52.

Parcel Locations

The default location for the local parcel directory on the Cloudera Manager Server is `/opt/cloudera/parcel-repo`. To change this location, follow the instructions in Configuring Cloudera Manager Server Parcel Settings on page 52.

The default location for the distributed parcels on managed hosts is `/opt/cloudera/parcels`. To change this location, set the `parcel_dir` property in `/etc/cloudera-scm-agent/config.ini` file of the Cloudera Manager Agent and restart the Cloudera Manager Agent or by following the instructions in Configuring the Host Parcel Directory on page 53.

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 can cause scripts that distinguish between the two paths to not work.

Managing Parcels

On the Parcels page in Cloudera Manager, you can manage parcel installation and activation and determine which parcel versions are running across your clusters. The Parcels page displays a list of parcels managed by Cloudera Manager. Cloudera Manager displays the name, version, and status of each parcel and provides available actions on the parcel.

Accessing the Parcels Page

Access the Parcels page by doing one of the following:

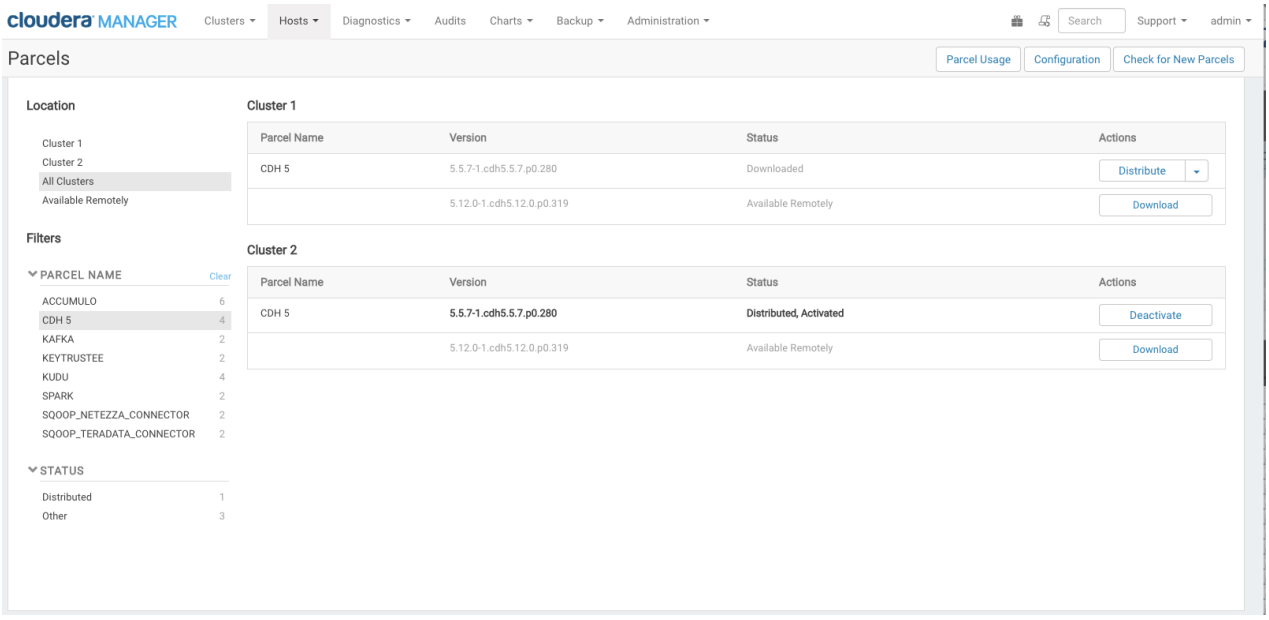
- Click the parcel icon in the top navigation bar.
- Click the **Hosts** in the top navigation bar, then the **Parcels** tab.

Use the selectors on the left side of the console to filter the displayed parcels:

- **Location** selector - View only parcels that are available remotely, only parcels pertaining to a particular cluster, or parcels pertaining to all clusters. When you access the Parcels page, the selector is set to Available Remotely.
- **Error Status** section of the **Filters** selector - Limit the list of displayed parcels by error status.
- **Parcel Name** section of the **Filters** selector - Limit the list of displayed parcels by parcel name.
- **Status** section of the **Filters** selector - Limit the list to parcels that have been distributed, parcels that have not been distributed (**Other**), or all parcels.

When you download a parcel, it appears in the list for each cluster managed by Cloudera Manager, indicating that the parcel is available for distribution on those clusters. Only one copy of the downloaded parcel resides on the Cloudera Manager Server. After you distribute the parcel, Cloudera Manager copies the parcel to the hosts in that cluster.

For example, if Cloudera Manager is managing two clusters, the rows in the All Clusters page list the information about the parcels on the two clusters. The Status column displays the current status of the parcels. The Version column displays version information about the parcel. Click the information icon to view the release notes for the parcel. The Actions column shows actions you can perform on the parcels, such as download, distribute, delete, deactivate, and remove from host.



Downloading a Parcel

1. Go to the Parcels page. In the Location selector, click ***ClusterName*** or **Available Remotely**. Parcels that are available for download display the Available Remotely status and a Download button.

   If the parcel you want is not shown here—for example, you want to upgrade to a version of CDH that is not the most current version—you can make additional remote parcel repositories available. You can also configure the location of the local parcel repository and other settings. See Parcel Configuration Settings on page 52.

   If a parcel version is too new to be supported by the Cloudera Manager version, the parcel appears with a red background and error message:

   Such parcels are also listed when you select the Error status in the Error Status section of the Filters selector.

2. Click the **Download** button of the parcel you want to download to your local repository. The status changes to Downloading.

After a parcel has been downloaded, it is removed from the Available Remotely page.

Distributing a Parcel

Downloaded parcels can be distributed to the hosts in your cluster and made available for activation. Parcels are downloaded to the Cloudera Manager Server, so with multiple clusters, the downloaded parcels are shown as available to *all* clusters managed by the Cloudera Manager Server. However, you select distribution to a specific cluster's hosts on a cluster-by-cluster basis.

1. From the Parcels page, in the Location selector, select the cluster where you want to distribute the parcel, or select **All Clusters**. (The first cluster in the list is selected by default when you open the Parcels page.)
2. Click **Distribute** for the parcel you want to distribute. The status changes to **Distributing**. During distribution, you can:
   - Click the **Details** link in the Status column to view the **Parcel Distribution Status** page.
   - Click **Cancel** to cancel the distribution. When the Distribute action completes, the button changes to **Activate**, and you can click the **Distributed** status link to view the status page.

Distribution does not require Internet access; the Cloudera Manager Agent on each cluster member downloads the parcel from the local parcel repository hosted on the Cloudera Manager Server.

If you have a large number of hosts to which parcels must be distributed, you can control how many concurrent uploads Cloudera Manager performs. See Parcel Configuration Settings on page 52.

To delete a parcel that is ready to be distributed, click the triangle at the right end of the **Distribute** button and select **Delete**. This deletes the parcel from the local parcel repository.

Distributing parcels to the hosts in the cluster does not affect the current running services.

Activating a Parcel

Parcels that have been distributed to the hosts in a cluster are ready to be activated.

1. From the Parcels page, in the Location selector, choose ***ClusterName*** or **All Clusters**, and click the **Activate** button for the parcel you want to activate. This updates Cloudera Manager to point to the new software, which is ready to run the next time a service is restarted. A pop-up indicates which services must be restarted to use the new parcel.
2. Choose one of the following:
   - **Restart** - Activate the parcel and restart services affected by the new parcel.
   - **Activate Only** - Active the parcel. You can restart services at a time that is convenient. If you do not restart services as part of the activation process, you must restart them at a later time. Until you restart services, the current parcel continues to run.
3. Click **OK**.

Activating a new parcel also deactivates the previously active parcel for the product you just upgraded. However, until you restart the services, the previously active parcel displays a status of **Still in use** because the services are using that parcel, and you cannot remove the parcel until it is no longer being used.

If the parcel you activate updates the software for only a subset of services, even if you restart all of that subset, the previously active parcel displays **Still in use** until you restart the remaining services. For example, if you are running HDFS, YARN, Oozie, Hue, Impala, and Spark services, and you activate a parcel that updates only the Oozie service, the pop-up that displays instructs you to restart only the Oozie and Hue services. Because the older parcel is still in use by the HDFS, YARN, Impala, and Spark services, the parcel page shows that parcel as **Still in use** until you restart these remaining services.

Sometimes additional upgrade steps may be required. In this case, instead of **Activate**, the button will say **Upgrade**. When you click the **Upgrade** button, the upgrade wizard starts. See Upgrading the CDH Cluster.

Deactivating a Parcel

You can deactivate an active parcel; this updates Cloudera Manager to point to the previous software version, which is ready to run the next time a service is restarted. From the Parcels page, choose ***ClusterName*** or **All Clusters** in the Location selector, and click the **Deactivate** button on an activated parcel.

To use the previous version of the software, restart your services.

Important: If you originally installed from parcels, and one version of the software is installed (that is, no packages, and no previous parcels have been activated and started), when you attempt to restart after deactivating the current version, your roles will be stopped and will not be able to restart.

Removing a Parcel

From the Parcels page, in the Location selector, choose ***ClusterName*** or **All Clusters**, click the  to the right of an **Activate** button, and select **Remove from Hosts**.

Deleting a Parcel

From the Parcels page, in the Location selector, choose ***ClusterName*** or **All Clusters**, and click the  to the right of a **Distribute** button, and select **Delete**.

Changing the Parcel Directory

The default location of the parcel directory is `/opt/cloudera/parcels`. To relocate distributed parcels to a different directory, do the following:

1. Stop all services.
2. Deactivate all in-use parcels.
3. Shut down the Cloudera Manager Agent on all hosts.
4. Move the existing parcels to the new location.
5. Configure the host parcel directory.
6. Start the Cloudera Manager Agents.
7. Activate the parcels.
8. Start all services.

Troubleshooting

If you experience an error while performing parcel operations, click the red 'X' icons on the parcel page to display a message that identifies the source of the error.

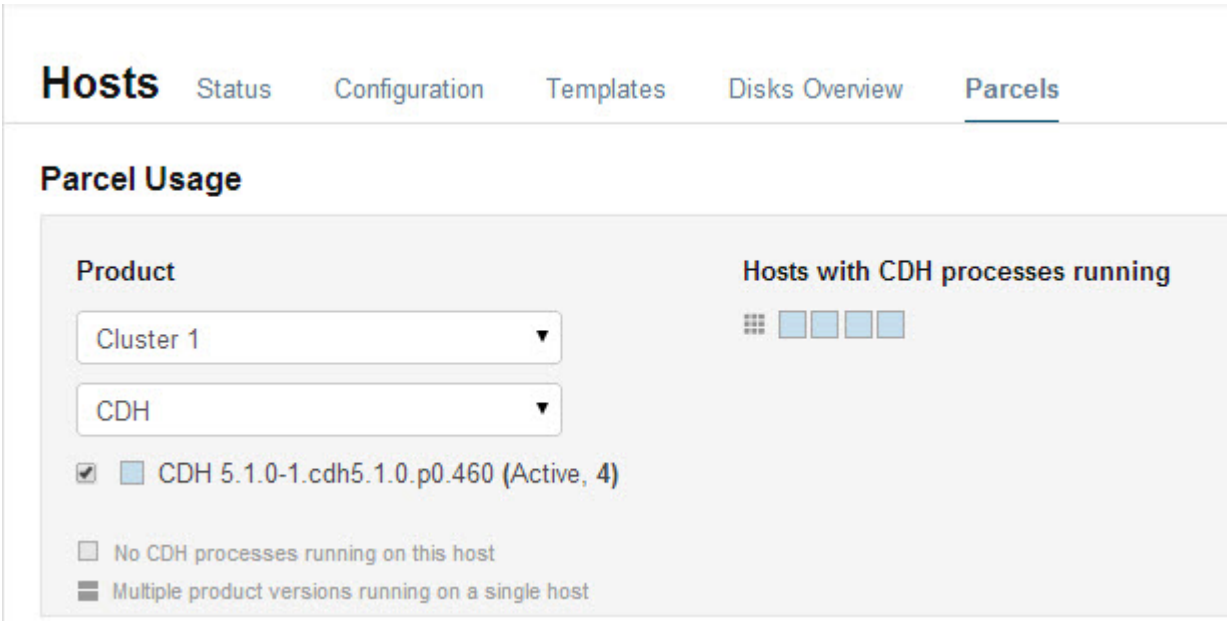
If a parcel is being distributed but never completes, make sure you have enough free space in the parcel download directories, because Cloudera Manager will try to download and unpack parcels even if there is insufficient space.

Viewing Parcel Usage

The **Parcel Usage** page shows parcels in current use in your clusters. In a large deployment, this makes it easier to keep track of different versions installed across the cluster, especially if some hosts were not available when you performed an installation or upgrade, or were added later. To display the Parcel Usage page:

1. Do one of the following:
   - Click the parcel icon in the top navigation bar.
   - Click **Hosts** in the top navigation bar and click the **Parcels** tab.
2. Click the **Parcel Usage** button.

This page only shows the usage of parcels, not components that were installed as packages. If you select a cluster running packages, the cluster is not displayed, and instead you see a message indicating the cluster is not running parcels.



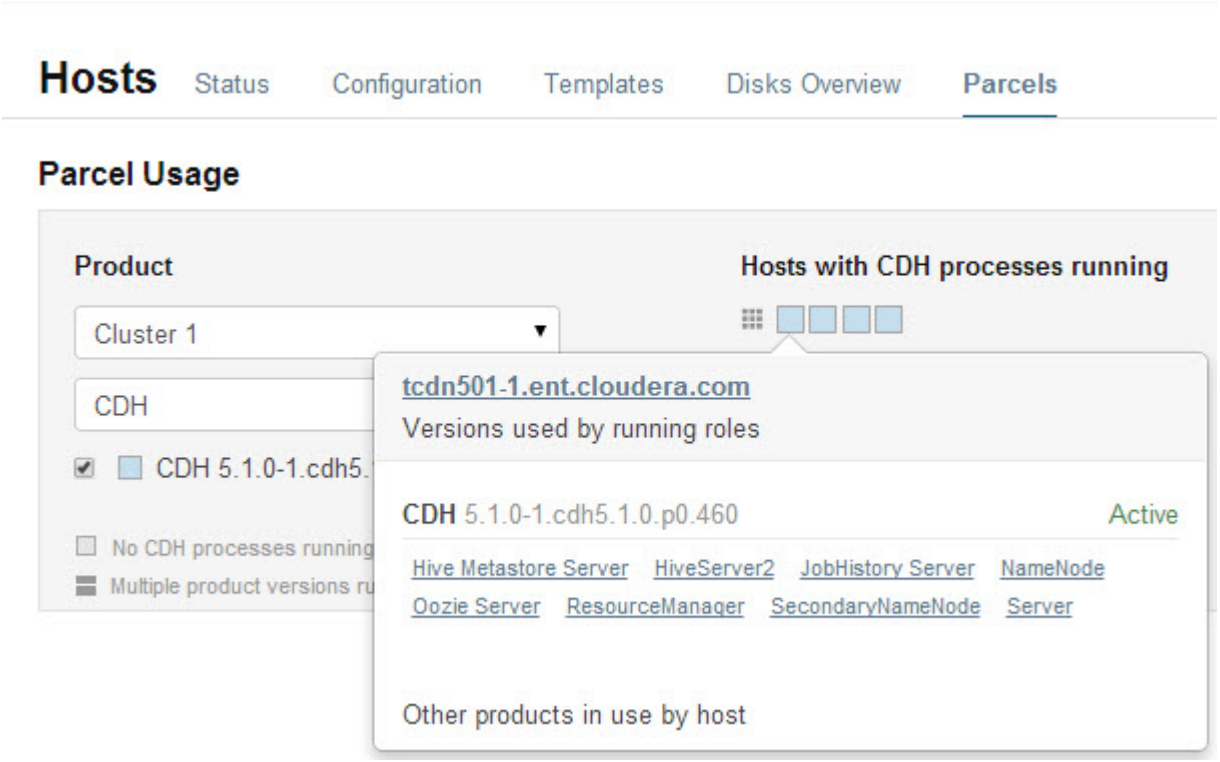
You can view parcel usage by cluster or by product.

You can also view just the hosts running only the active parcels, or just hosts running older parcels (not the currently active parcels), or both.

The host map at the right shows each host in the cluster, with the status of the parcels on that host. If the host is running the processes from the currently activated parcels, the host is indicated in blue. A black square indicates that a parcel has been activated, but that all the running processes are from an earlier version of the software. This occurs, for example, if you have not restarted a service or role after activating a new parcel. If you have individual hosts running components installed as packages, the square is empty.

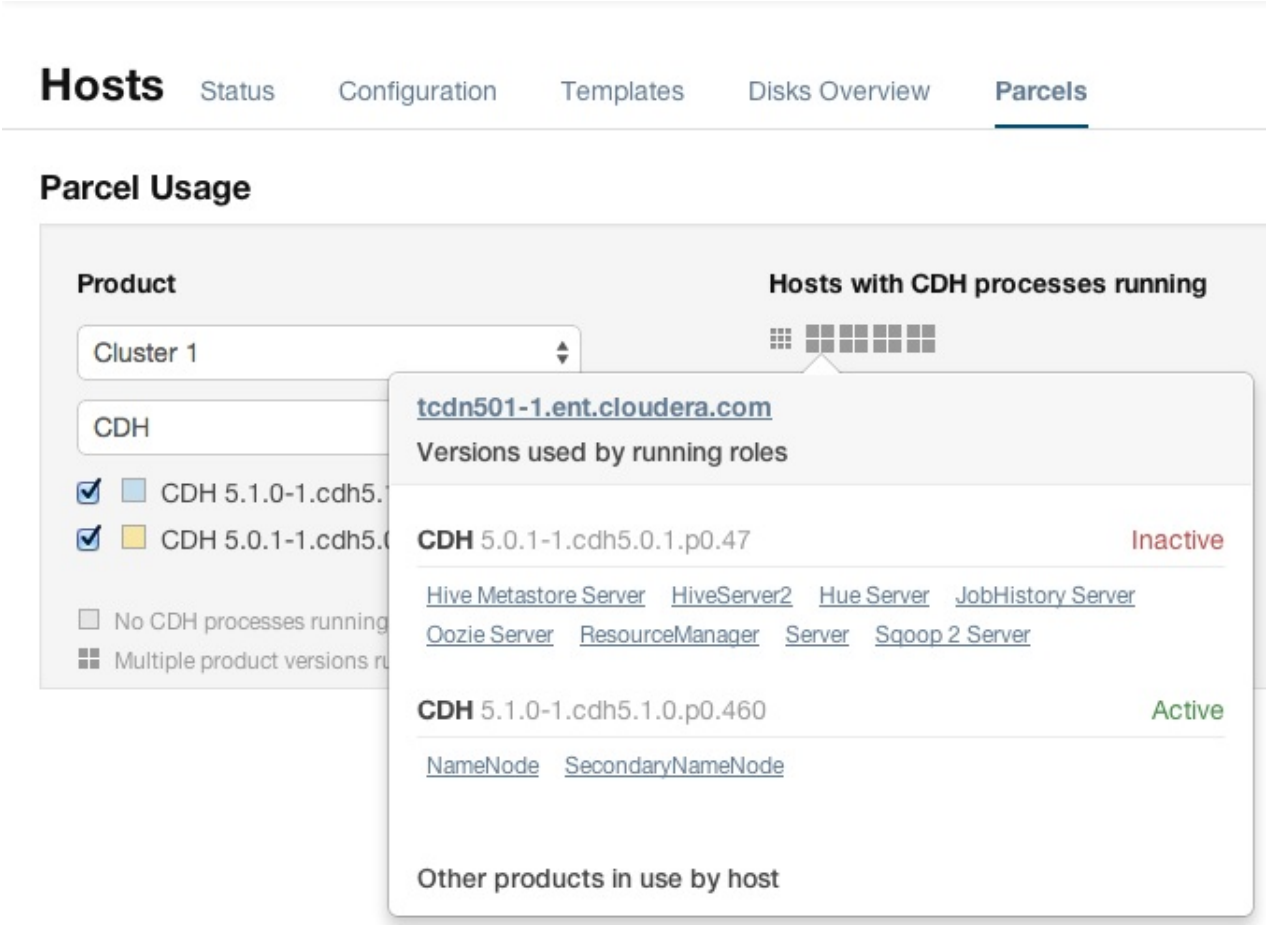
Move the cursor over the grid icon to see the rack to which the hosts are assigned. Hosts on different racks are displayed in separate rows.

To view the exact versions of the software running on a given host, click the square representing the host. This displays the parcel versions installed on that host.



The pop-up lists the roles running on the selected host that are part of the listed parcel. Clicking a role opens the Cloudera Manager page for that role. It also shows whether the parcel is active or not.

If a host is running various software versions, the square representing the host is a four-square grid icon. When you move the cursor over that host, both the active and inactive components are shown. For example, in the image below, the older CDH parcel has been deactivated, but only the HDFS service has been restarted.



Parcel Configuration Settings

You can configure where parcels are stored on the Cloudera Manager Server host, the URLs of parcel repositories, the properties of a proxy server through which parcels are downloaded, and where parcels distributed to cluster hosts are stored.

Configuring Cloudera Manager Server Parcel Settings

1. Use one of the following methods to open the parcel settings page:
   - **Navigation bar**
     1. Click the parcel icon in the top navigation bar or click **Hosts** and click the **Parcels** tab.
     2. Click the **Configuration** button.
   - **Menu**
     1. Select **Administration** > **Settings**.
     2. Select **Category** > **Parcels**.
2. Specify a property:
   - **Local Parcel Repository Path** defines the path on the Cloudera Manager Server host where downloaded parcels are stored.
   - **Remote Parcel Repository URLs** is a list of repositories that Cloudera Manager checks for parcels. Initially this points to the latest released CDH 5 and CDH 6repositories, but you can add your own repository locations to the list. Use this mechanism to add Cloudera repositories that are not listed by default, such as older versions of CDH. You can also use this to add your own custom repositories. The locations of the Cloudera parcel repositories are `https://archive.cloudera.com/`*`product`*`/parcels/`*`version`*, where *product*

is a product name and *version* is a specific product version, `latest`, or the substitution variable `{latest_supported}`. The substitution variable appears after the parcel for the CDH version with the same major number as the Cloudera Manager version to enable substitution of the latest supported maintenance version of CDH.

To add a parcel repository:

1. In the **Remote Parcel Repository URLs** list, click the addition symbol to open an additional row.
2. Enter the path to the repository.

3. Click **Save Changes**.

You can also:

- Set the frequency with which Cloudera Manager checks for new parcels.
- Configure a proxy to access to the remote repositories.
- Configure whether downloads and distribution of parcels should occur automatically when new ones are detected. If automatic downloading and distribution are not enabled (the default), go to the **Parcels** page to initiate these actions.
- Control which products can be downloaded if automatic downloading is enabled.
- Control whether to retain downloaded parcels.
- Control whether to retain old parcel versions and how many parcel versions to retain

You can tune the parcel distribution load on your network by configuring the bandwidth limits and the number of concurrent uploads. The defaults are up to 50 MiB/s aggregate bandwidth and 50 concurrent parcel uploads.

- Theoretically, the concurrent upload count (**Maximum Parcel Uploads**) is unimportant if all hosts have the same speed Ethernet. Fifty concurrent uploads is acceptable in most cases. However, if the server has more bandwidth (for example, 10 GbE, and the normal hosts are using 1 GbE), then the count is important to maximize bandwidth. It should be at least the difference in speeds (10x in this case).
- The bandwidth limit (**Parcel Distribution Rate Limit**) should be your Ethernet speed (in MiB/seconds) divided by approximately 16. You can use a higher limit if you have QoS configured to prevent starving other services, or if you can accept the risk associated with higher bandwidth load.

Configuring a Proxy Server

To configure a proxy server through which parcels are downloaded, follow the instructions in Configuring Network Settings.

Configuring the Host Parcel Directory

Important: If you modify the parcel directory location, make sure that all hosts use the same location. Using different locations on different hosts can cause unexpected problems.

To configure the location of distributed parcels:

1. Click **Hosts** in the top navigation bar.
2. Click the **Configuration** tab.
3. Select **Category** > **Parcels**.
4. Configure the value of the **Parcel Directory** property. The setting of the `parcel_dir` property in the Cloudera Manager Agent configuration file overrides this setting.
5. Enter a **Reason for change**, and then click **Save Changes** to commit the changes.
6. Restart the Cloudera Manager Agent on all hosts.

Configuring Peer-to-Peer Distribution of Parcels

Cloudera Manager uses a peer-to-peer service to efficiently distribute parcels to cluster hosts. The service is enabled by default and is configured to run on port 7191. You can change this port number, and you can disable peer-to-peer distribution.

To modify peer-to-peer distribution of parcels:

1. Open Cloudera Manager and select **Hosts** > **All Hosts** > **Configuration**.
2. Change the value of the **P2P Parcel Distribution Port** property to the new port number.

   Set the value to `0` to disable peer-to-peer distribution of parcels.
3. Enter a **Reason for change**, and then click **Save Changes** to commit the changes.

Cloudera Navigator Data Management Overview

Cloudera Navigator Data Management is a complete solution for data governance, auditing, and related data management tasks that is fully integrated with the Hadoop platform. Cloudera Navigator lets compliance groups, data stewards, administrators, and others work effectively with data at scale. This brief introduction includes the following topics:

For further information on using and administering Cloudera Navigator Data Management see the Navigator product guide.

Data Management Challenges

As Hadoop clusters have become ubiquitous in organizations large and small, the ability to store and analyze all types of data at any scale brings with it some management challenges for reasons such as these:

- Data volumes are extremely large and continue to grow, with increased velocity while comprising various data types.
- Data ingested by the cluster at one point in time is transformed by many different processes, users, or applications. The result is that the provenance of any data entity can be difficult at best to trace.
- Multi-user and multi-tenant access to the data is the norm. Each user group may need different levels of access to the data, at varying degrees of granularity.

These characteristics make it difficult to answer questions such as these:

- Where did the data originate? Has it been altered, and if so, by whom or by what process?
- Are downstream processes using that data, and if so, how?
- Have unauthorized people been trying to get to the data? Can we verify all accesses and attempted access to our data?
- Are we prepared for an audit? For example, can we prove that the values shown in the organization's G/L have not been mishandled? Will the bank examiners approve our data governance operations? In general, can the data's validity be trusted, and better yet, proven?
- Are data retention mandates being met? Are we complying with all industry and government regulations for preservation and deletion of data? Can we automate the life cycle of our data?
- Where is the most important data? Is there a way to get an at-a-glance view about overall cluster activity, over various time periods, by data type?

Enabling organizations to quickly answer questions such as these and many more is one of the chief benefits of Cloudera Navigator.

Cloudera Navigator Data Management Capabilities

As an organization's clusters consume more and more data, its business users want self-service access to that data. For example, business users might want to find data relevant for a current market analysis by searching using the nomenclature common to their field of expertise rather than needing to know all the file types that might contain such data.

At the same time, the organization's security team wants to know about all attempts to access any and all data. They want to be able to quickly identify confidential data and to track any data entity to its source (provenance). The

compliance group wants to be audit-ready at all times. And everyone, organization wide, wants to completely trust the integrity of the data they are using.

These are the kinds of data management challenges that Cloudera Navigator data management was designed to meet head on. Data stewards, administrators, business analysts, data scientists, and developers can obtain better value from the vast amounts of data stored in Hadoop clusters through the growing set of features provided by Cloudera Navigator data management.

The following capabilities are all available through the Cloudera Navigator console.

Analytics

The Cloudera Navigator analytics system leverages the metadata system, policies, and auditing features to provide a starting place for most users, from data stewards to administrators. Get at-a-glance overviews of all cluster data across various dimensions and using a variety of interactive tools to easily filter and drill down into specific data objects. For example, Data Stewardship Analytics has a Dashboard and Data Explorer that provide comprehensive yet intuitive tools for interacting with all data in the system. The HDFS Analytics page displays histograms of technical metadata for HDFS files (file size, block size, and so on). Use the mouse and brush over any histogram to display the lower level details, along with selectors and other filters for digging further below the surface of the data objects.

Auditing

In addition to meeting the needs of data stewards, Cloudera Navigator also meets governance needs by providing secure real-time auditing. The Navigator Audit Server creates a complete and immutable record of cluster activity (in its own database) that is easy to retrieve when needed. Compliance groups can configure, collect, and view audit events that show who accessed data, when, and how.

Metadata

The analytics, lineage, and search capabilities of Cloudera Navigator rely on metadata, both the technical metadata inherent in the data object itself (date, file type, and so on) or the metadata you define (managed and user-defined properties) to characterize data objects so they can be easily found by data stewards, data scientists, and other business users.

For example, Navigator Metadata Server captures information about table, file, and database activity, including file and table creation and modification trends, all of which is displayed in visually clean renderings on the Data Stewardship dashboard.

Policies

Cloudera Navigator policies let you automate actions based on data access or on a schedule, to add metadata, create alerts, move data, or purge data.

Search

By default, the Cloudera Navigator console opens to the Search menu. Use this sophisticated (yet simple) filtering scheme to find all types of entities that meet your criteria, then drill down to explore a specific entity's lineage. High level diagrams can be filtered further using the interactive tools provided, letting you follow the data upstream or downstream and in myriad other ways.

Getting Started with Cloudera Navigator

Cloudera Navigator features and functions are available through the Cloudera Navigator console, a web-based user interface provided by the host configured with the Navigator Metadata Server role (daemon). The Cloudera Navigator console invokes the REST APIs exposed by the web service provided by the Navigator Metadata Server role instance. The APIs can be called directly by in-house, third-party, and other developers who want to further automate tasks or implement functionality not currently provided by the Cloudera Navigator console. This topic discusses both of these interfaces to the Cloudera Navigator metadata component.

Note: Navigator Metadata Server is one of the roles (daemons) that comprises Cloudera Navigator. The other role is Navigator Audit Server. See Cloudera Navigator Data Management for details.

The following steps assume that the Cloudera Navigator data management component is running, and that you know the host name and port of the node in the cluster configured with the Navigator Metadata Server role. See Cloudera Installation and Cloudera Administration guides for more information about setting up Cloudera Navigator.

Cloudera Navigator Console

The Cloudera Navigator console is the web-based user interface that provides data stewards, compliance groups, auditing teams, data engineers, and other business users access to Cloudera Navigator features and functions.

The Cloudera Navigator console provides a unified view of auditing, lineage, and other metadata management capabilities across all clusters managed by a given Cloudera Manager instance. That is, if the Cloudera Manager instance manages five clusters, the Cloudera Navigator console can obtain auditing, lineage, and other metadata for the same five clusters. Cloudera Navigator uses its technical metadata gathering feature to keep track of which cluster originates the data.

Accessing the Cloudera Navigator Console

Open the Cloudera Navigator console in a browser at port 7187 on the host where the Navigator Metadata Server instance is running:

```
http://fqdn-1.example.com:7187/login.html
```

In this example, node 1 of the cluster is running the Navigator Metadata Server role, hosted on the default port 7187. The login page displays. Log in to the Cloudera Navigator console using the credentials assigned by your administrator.

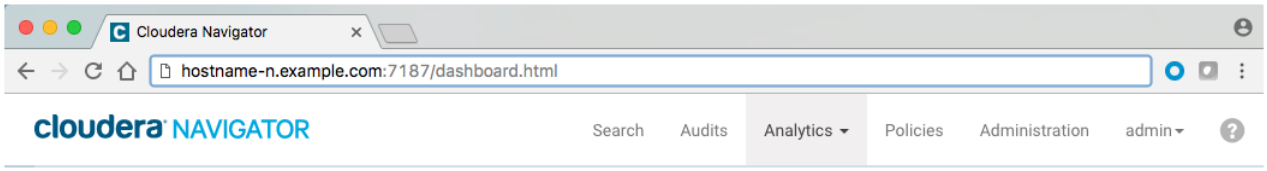
Cloudera Manager users with the role of Navigator Administrator (or Full Administrator) can also access Cloudera Navigator console from the Cloudera Manager Admin Console.

From the Cloudera Manager Admin Console:

1. Open your browser.
2. Navigate to the Cloudera Manager Admin Console.
3. Log in as either Navigator Administrator or Full Administrator.
4. From the menu, select **Clusters** > ***Cluster-n***.
5. Select **Cloudera Navigator** from the menu.
6. Log in to the Cloudera Navigator console.

Menu and Icon Overview

The Cloudera Navigator console opens to an empty Search results page and displays the following menu bar.



The *user_name* and Help menus display for all users. However, other available menu choices depend on the Cloudera Navigator user role of the logged in account. In this example, the **admin** account is logged in. That account has Navigator Administrator privileges which means that all menu items display and the user has full access to all system features. For a user account limited to Policy Editor privileges, only the Policies menu item displays.

This table provides a reference to all menu choices.

Icon or menu	Description
Search	Displays sets of filters that can be applied to find specific entities by source type, type, owner, cluster name, and tags.
Audits	Displays summary list of current audit events for the selected filter. Define filters for audit events, create reports based on selected time frames and filters, export selected audit details as CSV or JSON files.
Analytics	Menu for HDFS Analytics and Data Stewardship Analytics selections. This is your entry point for several dashboards that display the overall state of cluster data and usage trends: • Data Stewardship Analytics displays Dashboard and Data Explorer tabs. • HDFS Analytics displays Metadata and Audit tabs.
Policies	Displays a list of existing policies and enables you to create new policies.
Administration	Displays tabs for Managed Metadata, Role Management, and Purge schedule settings.
user_name	Displays account (**admin**, for example) currently logged in to Cloudera Navigator. Menu for Command Actions, My Roles, Enable Debug Mode, and Logout selections.
	Help icon and menu. Provides access to Help, API Documentation, and About selections. Select Help to display Cloudera Navigator documentation. Select About to display version information.

This is just an overview of the menu labels in the Cloudera Navigator console and pointers to some of the sub-menus. See the Cloudera Navigator Data Management guide for details.

Displaying Documentation

You can open the Cloudera Data Management guide from the Cloudera Navigator console from the Help menu.

> **Note:** You can also view the Cloudera Data Management guide outside the scope of a running instance of Cloudera Navigator.

Displaying Version Information

To display the version and build number for the Cloudera Navigator data management component, click About in the  menu.

The versions for both Cloudera Navigator and the associated Cloudera Manager instance display.

Cloudera Navigator APIs

Cloudera Navigator exposes REST APIs on the host running the Navigator Metadata Server role. In-house, third-party, and other developers can use the APIs to provide functionality not currently available through the Cloudera Navigator console or to customize their users' experience.

The APIs are used by the Cloudera Navigator console at runtime as users interact with the system. The Cloudera Navigator console has a debug mode that captures API calls and responses to a file that can then be downloaded and analyzed. See Using Debug Mode on page 59 for details.

Documentation and a brief tutorial for the Cloudera Navigator APIs are available on the same node of the cluster that runs the Navigator Metadata Server role. These are also accessible from the Cloudera Navigator console. Example pathnames for direct access (outside the Cloudera Navigator console) are shown in the table.

Resource	Default location
APIs	http://*navigator-host:port*/api/*vn/operation*
API Documentation	http://*navigator-host:port*/api-console/index.html

Resource	Default location
API Usage Tutorial	http://*navigator-host:port*/api-console/tutorial.html

Operations on the API are invoked using the following general pattern:

```
http://navigator-host:port/api/vn/operation
```

The *n* in v*n* for the APIs represents the version number, for example, v11 for the Cloudera Navigator version 2.11.x. The API version number is located at the bottom of the Cloudera Navigator API documentation page.

The API uses HTTP Basic Authentication and accepts the same users and credentials that have access to the Cloudera Navigator console.

The resources listed in the table are aimed at technical and general audiences for Cloudera Navigator Data Management. The GitHub repository is aimed at developers.

Resource	Default location
GitHub cloudera/navigator-sdk	Cloudera Navigator SDK is a client library that provides functionality to help users extract metadata from Navigator and to enrich the metadata in Navigator with custom metadata models, entities, and relationships.
Cloudera Community Data Management forum	Cloudera Community > Data Management > Data Discovery, Optimization forum for Cloudera Navigator users and developers.
Cloudera Engineering Blog	In-depth technical content provided by Cloudera engineers about components and specific subject areas.

Using Debug Mode

Cloudera Navigator console debug mode captures API responses and requests during interaction with the server. Use debug mode to facilitate development efforts or when asked by Cloudera Support to troubleshoot issues.

Tip: Exploring debug files can be a good way to learn about the APIs.

To follow these steps you must be logged in to the Cloudera Navigator console.

Use Debug Mode to Collect Files for Troubleshooting

The general process for obtaining debug files to send to Cloudera Support is as follows:

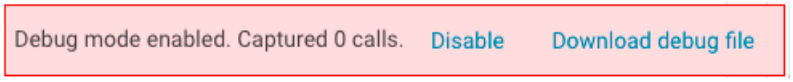
1. Navigate to the Cloudera Navigator console menu item or page for which you want to obtain debug information.
2. Enable debug mode.
3. Reload the page and perform the specific action. For example, if you need to send a debug file to Cloudera Support for an error message that displays when you click on an entity in one of the charts, perform that action to raise the error message.
4. Download the debug file or files. Each time the counter changes, the content of the debug file content changes, so you should download each time the counter updates.
5. Disable debug mode.

Enable Debug Mode

To enable debug mode:

1. Log in to the Cloudera Navigator console.
2. Select ***user_name*** > **Enable Debug Mode**.

   The Cloudera Navigator consoles displays a notification message in the bottom right of the page:



Notice the links to Disable and Download debug file, which you can use at any time to do either. Do not disable debug mode until you have downloaded the debug file you need.

3. Reload the browser page.

The counter *n* displays the number of API calls captured in the debug file for the current session interaction. Here is an example from a downloaded debug file showing one of the responses after reloading the page:


```
{
  "type": "GET",
  "url": "/api/v10/dashboard/clusters",
  "status": 200,
  "responseText": "{\n  \"clusterInfos\" : [ {\n    \"clusterId\" : \"Cluster 1\"
    ,\n    \"clusterDisplayText\" : \"Cluster 1\"\n  } ]\n}",
  "page": "http://fqdn-1.example.com:7187/dashboard.html",
  "timestamp": 1497719524407
},
```


Download Debug File or Files

As you perform actions with the Cloudera Navigator console, the counter refreshes and keeps tracks of current calls only (stateless nature of REST). In other words, download the debug file for any given action or selection that you want to examine.

To download the debug file:

- Enable Debug Mode if you haven't already.
- Click the **Download debug file** link in the debug-enabled notification message, bottom right corner of the page.
- Save the file to your local workstation.

Debug files use a naming convention that includes the prefix "api-data" followed by the name of the host and the UTC timestamp of the download time and the JSON file extension. The pattern is as follows:

```
api-data-navigator-host-yyyy-mm-ddThh-mm-ss.json
```

For example:

```
api-data-node01.example.com-2017-06-15T09-00-32.json
```

Disable Debug Mode

When you are finished capturing and downloading debug files, you can disable debug mode.

- Click the **Disable debug mode** link in the notification message box.

The message box is dismissed and debug mode is disabled.

Or, use the Cloudera Navigator console menu:

- Select ***user_name*** > **Disable Debug Mode**.

Set Debug Logging Levels

The Cloudera Navigator Debug Mode allows you to configure the logging levels for individual methods in the application code. For example, to increase the logging detail for API calls, you would set the logging level for the `org.apache.cxf.interceptor` class to "DEBUG" level. If you are collecting debugging information for a specific issue, Cloudera support will provide the class name.

To set the logging level for a Cloudera Navigator application class:

1. Log in to the Cloudera Navigator console.

2. Add /debug to the Navigator URL:

```
http://navigator-host:7187/debug
```

   The debugging console appears.

3. Click **Configure Logging**.
4. In the **Logger** drop-down menu, select a Navigator class, such as `org.apache.cxf.interceptor` for API calls.
5. Set the logging level for the class.

   For example, for the API call logging, select **DEBUG**.

6. Click **Submit**.

When changed in Debug Mode, logging levels will return to their default values after restarting the Navigator Metadata Server. To permanently change the logging levels, use Cloudera Manager to add an entry in the Navigator Metadata Server Logging Advanced Configuration Snippet. Create the entry as follows:

```
log4j.logger.<classname>=<level>
```

For example, to set the API logging level to "DEBUG", include the following string in the Advanced Configuration Snippet:

```
log4j.logger.org.apache.cxf.interceptor=DEBUG
```

See Modifying Configuration Properties Using Cloudera Manager.

Cloudera Navigator Frequently Asked Questions

Cloudera Navigator offers components for security, data management, and data optimization, as follows:

- Cloudera Navigator Data Encryption is a data-at-rest encryption and key management suite that includes Cloudera Navigator Encrypt, Cloudera Navigator Key Trustee Server, among other components.
- Cloudera Navigator Data Management is a comprehensive auditing, data governance, compliance, data stewardship, and data lineage discovery component that is fully integrated with Hadoop.

This Navigator YouTube video gives a light-hearted look at the Cloudera Navigator brand and helps identify the value of each of the Navigator components:

https://youtu.be/jl3cBU2lHeY

The FAQs below discuss the Cloudera Navigator Data Management component only.

For further information on using and administering Cloudera Navigator Data Management see the Navigator product guide.

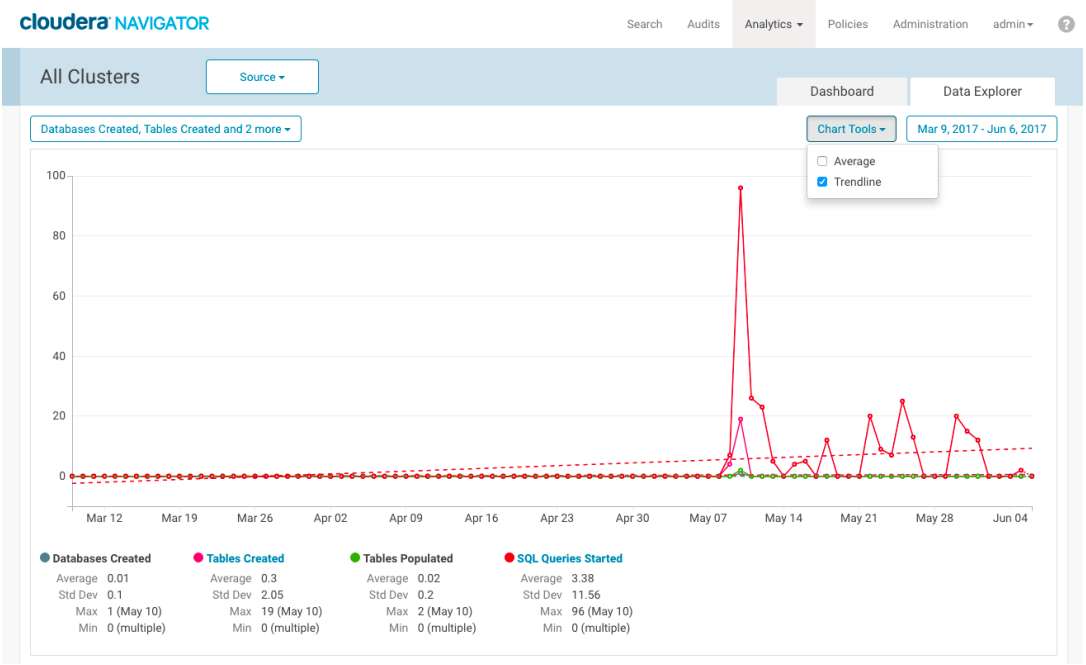
Is Cloudera Navigator a module of Cloudera Manager?

Not exactly. Cloudera Navigator is installed separately, after Cloudera Manager is installed, and it interacts behind the scenes with Cloudera Manager to deliver some of its core functionality. Cloudera Manager is used by cluster administrators to manage the cluster and all its services. Cloudera Navigator is used by administrators but also by security and governance teams, data stewards, and others to audit, trace data lineage from source raw data through final form, and perform other comprehensive data governance and stewardship tasks.

Does Cloudera Navigator have a user interface?

Yes. The Cloudera Navigator console is a browser-based user interface hosted on the node running an instance of the Navigator Metadata Server role. It provides a unified view of clusters associated with a given Cloudera Manager instance and lets you customize metadata, trace lineage, view audit events, and so on.

For example, the screenshot below shows a Data Explorer view of databases and tables created, tables populated, and SQL queries started within the specified timeframe (May 9, 2017 to June 6, 2017), with a trendline (the dashed line) displayed in the context of the results:



Clicking anywhere within the chart lets you drill down into the entities and begin identifying sources of spikes and trends and open lineage diagrams.

- See the Cloudera Navigator Data Management Oveview to learn more about the types of questions that Cloudera Navigator is designed to answer.
- See Getting Started with Cloudera Navigator for an overview of the Cloudera Manager console.

How can I access the Cloudera Navigator console?

The Cloudera Navigator console can be accessed from the Cloudera Manager Admin Console or directly on the Navigator Metadata Server instance. Using the Cloudera Manager Admin Console as a starting point requires the Cloudera Manager roles of either Navigator Administrator or Full Administrator.

From the Cloudera Manager Admin Console:

1. Select **Clusters** > ***Cluster-n***.
2. Select **Cloudera Navigator** from the menu.

To access the Cloudera Navigator console directly:

1. Open the browser to the host name of the node in the cluster that runs the Cloudera Navigator Metadata Server. For example, if node 2 in the cluster is running the Navigator Metadata Server role, the URL to the Cloudera Navigator console (assuming the default port of 7187) might look as follows:

```
http://fqdn-2.example.com:7187/login.html
```

2. Log in to the Cloudera Navigator console using the credentials assigned by your administrator.

What are the requirements for Cloudera Navigator?

Cloudera Navigator data management is a standard component of the Cloudera Enterprise edition. It is not included with Cloudera Express and requires an enterprise license. The Cloudera Enterprise license is an annual subscription that provides a complete portfolio of support and services. See Cloudera Enterprise Data Sheet for details.

Can Cloudera Navigator be purchased separately?

No. Cloudera Navigator auditing and metadata subsystems interact with various subsystems of Cloudera Manager. For companies without Kerberos or other external means of authentication, Cloudera Navigator uses Cloudera Manager for user authentication. In short, Cloudera Manager is a pre-requisite for Cloudera Navigator, and must be installed before you can install Cloudera Navigator.

What versions of Cloudera Manager, CDH, and Impala does Cloudera Navigator support?

See Product Compatibility Matrix for Cloudera Navigator.

Is Cloudera Navigator an open source project?

No. Cloudera Navigator is closed source software that is fully integrated with the Hadoop platform. Cloudera Navigator complements Cloudera Manager and part of the Cloudera suite of management capabilities for Hadoop clusters.

Do Cloudera Navigator logs differ from Cloudera Manager logs?

Yes. For example, Cloudera Navigator tracks and aggregates accesses to the data stored in CDH services and is 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 data access and system activity for an enterprise cluster.

How can I learn more about Cloudera Navigator?

See Cloudera Navigator Data Management Overview for a more detailed introduction to Cloudera Navigator. Other Cloudera Navigator information resources are listed in the table. Installation, upgrade, administration, and security guides are all aimed at administrators.

User Guide	Cloudera Navigator Data Management guide shows data stewards, compliance officers, and business users how to use Cloudera Navigator for data governance, compliance, data stewardship, and other tasks. Topics include Auditing, Metadata, Lineage Diagrams, Cloudera Navigator and the Cloud, Services and Security Management, and more.
Installation	Installing the Cloudera Navigator Data Management Component
Upgrade	Cloudera Navigator upgrades along with Cloudera Manager. See Upgrading Cloudera Manager for details.
Security	Configuring Authentication for Cloudera Navigator
	Configuring TLS/SSL for Navigator Audit Server
	Configuring TLS/SSL for Navigator Metadata Server

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 data 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 key management server (KMS) 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 HSM KMS

  Also for HDFS Transparent Encryption, Navigator HSM KMS provides a customized key management server (KMS) that uses third-party HSMs to provide the highest level of key isolation, storing key material on the HSM. When using the Navigator HSM KMS, encryption zone key material originates on the HSM and never leaves the HSM. While Navigator HSM KMS allows for the highest level of key isolation, it also requires some overhead for network calls to the HSM for key generation, encryption and decryption operations.

- Cloudera Navigator HSM KMS Services and HA

  Navigator HSM KMSs running on a single node fulfill the functional needs of users, but do not provide the non-functional qualities of service necessary for production deployment (primarily key data high availability and key data durability). You can achieve high availability (HA) of key material through the HA mechanisms of the backing HSM. However, metadata cannot be stored on the HSM directly, so the HSM KMS provides for high availability of key metadata via a built-in replication mechanism between the metadata stores of each KMS role instance. This release supports a two-node topology for high availability. When deployed using this topology,

there is a durability guarantee enforced for key creation and roll such that a key create or roll operation will fail if it cannot be successfully replicated between the two nodes.

Cloudera Navigator data 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 data 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 data encryption can secure:

1. HDFS data
2. Local filesystem data

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

Table 1: Encrypting Data at Rest

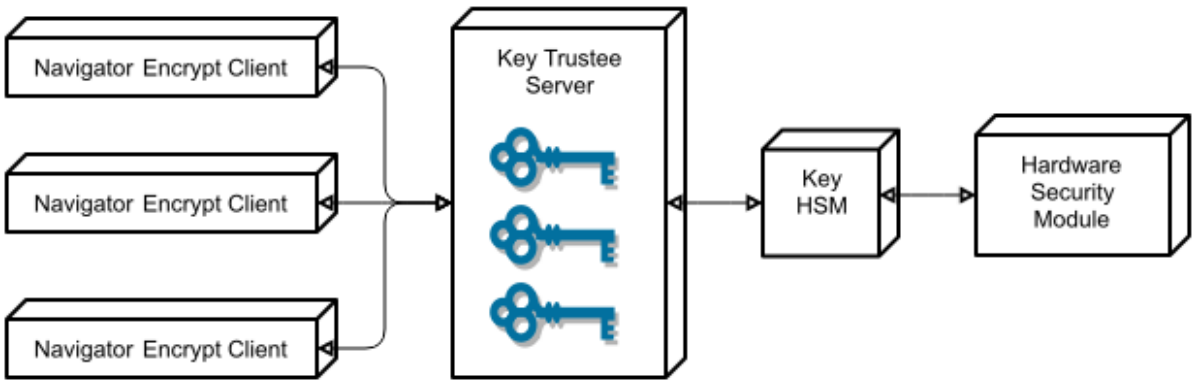
Data Type	Data Location	Key Management	Additional Services Required
HDFS	HDFS	Key Trustee Server	Key Trustee KMS
Metadata databases, including: • Hive Metastore • Cloudera Manager • Cloudera Navigator Data Management • Sentry	Local filesystem	Key Trustee Server	Navigator Encrypt
Temp/spill files for CDH components with native encryption: • 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:	Local filesystem	Key Trustee Server	Navigator Encrypt

Data Type	Data Location	Key Management	Additional Services Required
• Kafka • HiveServer2			
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 Data Encryption Architecture

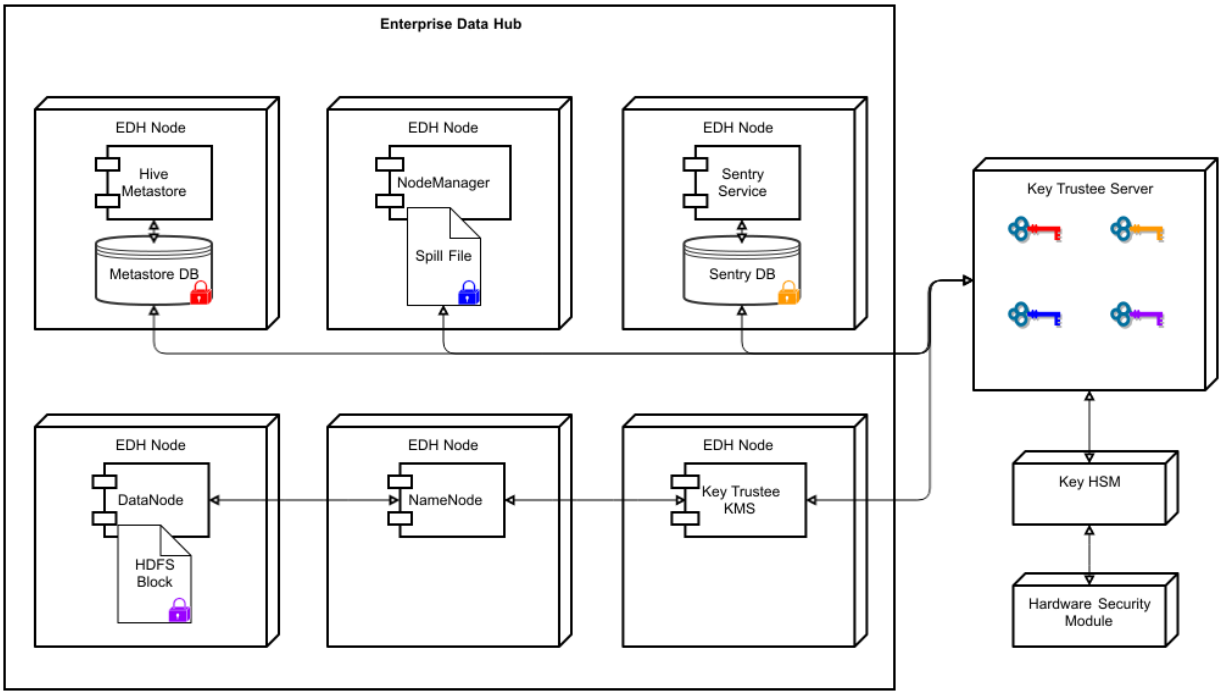
The following diagram illustrates how the Cloudera Navigator data encryption components interact with each other:



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

Cloudera Navigator Data Encryption Integration with an EDH

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



For more details on the individual components of Cloudera Navigator data 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 68 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.

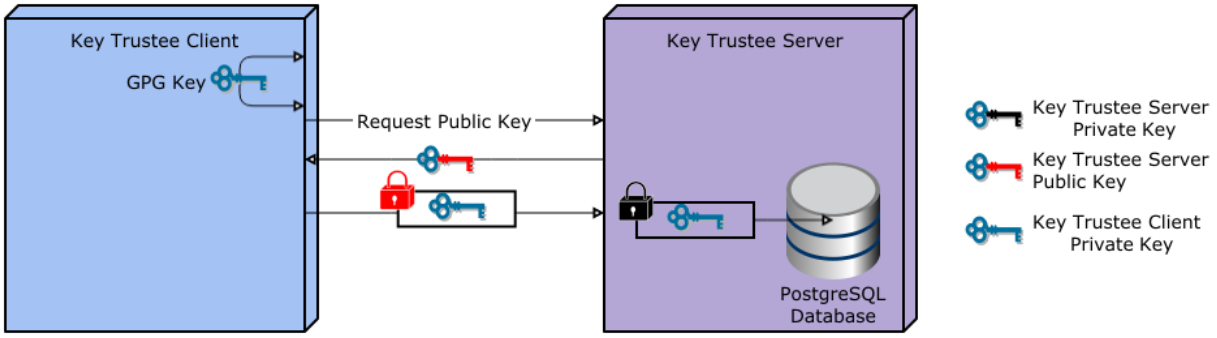
For instructions on installing Navigator Key Trustee Server, see Installing Cloudera Navigator Key Trustee Server. For instructions on configuring Navigator Key Trustee Server, see Initializing Standalone Key Trustee Server or Cloudera Navigator Key Trustee Server High Availability.

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.

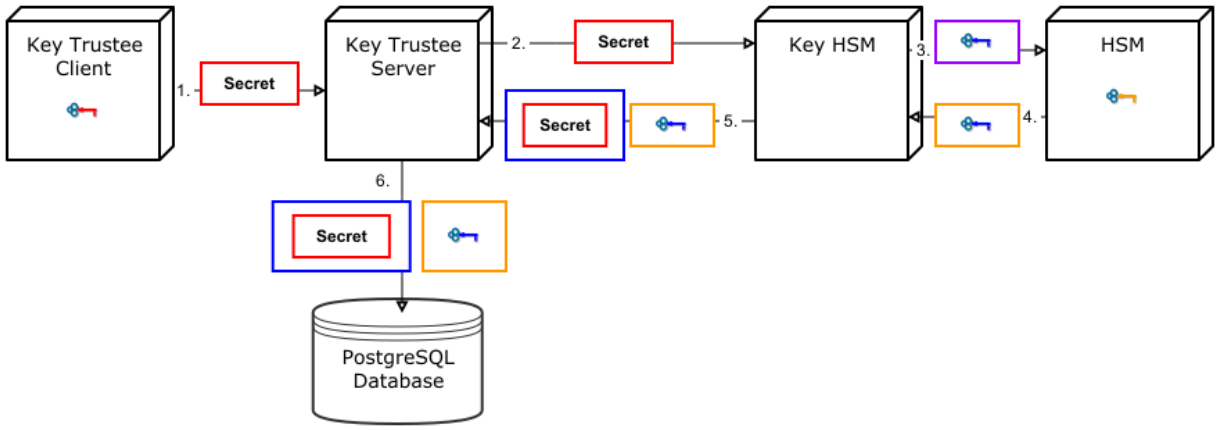
Many clients can use the same Key Trustee Server to manage security objects. For example, you can have several Navigator Encrypt clients using a Key Trustee Server, and also use the same Key Trustee Server as the backing store for Key Trustee KMS (used in HDFS encryption).

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:



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.

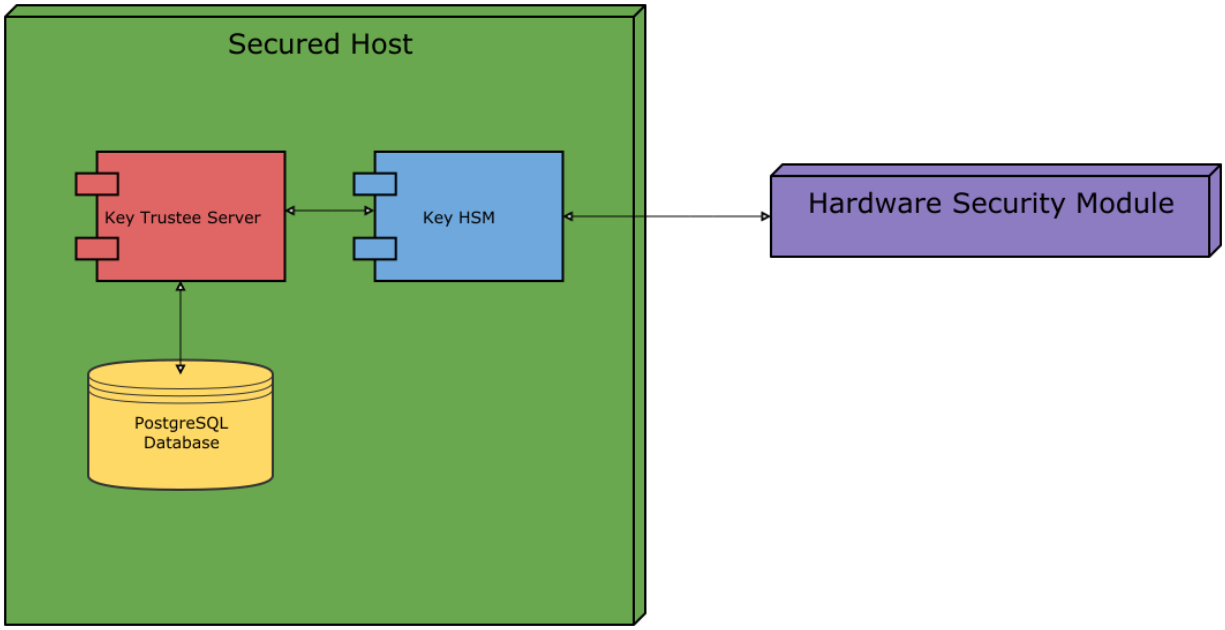
4. The HSM generates a new key pair and encrypts the symmetric key and returns the encrypted symmetric key to Key HSM.
5. Key HSM encrypts the original client-encrypted secret with the symmetric key, and returns the twice-encrypted secret, along with the encrypted symmetric key, to Key Trustee Server. 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 HSM KMS Overview

In addition to existing Navigator Key Trustee KMS (KT KMS) solutions (see Configuring the Key Management Server (KMS)), Cloudera offers the Navigator Hardware Security Module Key Management Server (HSM KMS) solution, which provides an additional encryption model (other encryption models include the Java Keystore KMS and KT KMS) to ensure the greatest level of encryption security for an enterprise's HDFS data.

Specifically, the Navigator HSM KMS solution provides the following key security benefits:

- Encryption zone keys (EZ keys) *originate on* and *never leave* the HSM. This is a significant reason to select an HSM KMS over the existing KMS implementations, as it provides the highest level of EZ key security.
- The HSM KMS root of trust model differs from the model found in the existing KT KMS. KT KMS employs a *client-based trust* model, where there is an exchange of GPG keys that enable the client and server to identify and

communicate with each other privately. For some, this client-based trust level of GPG key exchange can be a bit of a challenge to manage and administer. So, the HSM KMS model can be easier to manage in terms of the exchange of root of trust because the HSM is the root of trust and always secures the EZ key(s)

For details on the client prerequisites and installation instructions for the available HSM KMS solutions, see:

- Installing Navigator HSM KMS Backed by Thales HSM
- Installing Navigator HSM KMS Backed by Luna HSM

For details about resource planning for HSM KMS, see Navigator HSM KMS HA Planning.

Data Privacy with HSM KMS: Encryption Zone Key Creation and Management

As mentioned previously, in the Navigator HSM KMS architecture the EZ key *originates in and never* leaves the HSM.

With the HSM KMS, EZ key generation occurs in the HSM device, meaning that the actual key being used to encrypt data encryption keys for HDFS encryption is created within the HSM, and never leaves the HSM. All EDEK encryption and decryption operations happen within the HSM, providing the greatest level of encryption zone key protection.

For additional details about encryption keys and zones, see Managing Encryption Keys and Zones.

HSM KMS High Availability (HA) Support

The HSM KMS supports HA using a two-node, leaderless HA model. Both nodes in the HSM KMS configuration are active, and both nodes are first-class providers of the KMS proxy service, meaning one node can be down while the other node continues to operate as normal.

The HSM KMS HA model also provides a durability guarantee for the two-node topology. When an EZ key is created or rolled, the information is written to at least two nodes before returning to the client. In this way, if there is a failure of one node, you can rest assured that there is always at least one copy of the EZ key on the other node that was created. So, when there are two nodes, data is written to both nodes before returning to the client.

Essentially, the HA model dictates writes to the number of nodes required by the durability guarantee, and then the remainder of synchronization operations occur in the background.

For details about how to enable HSM KMS HA, refer to Enabling Navigator HSM KMS High Availability.

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 `dmcrypt` 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.
- `dmcrypt` Device Encryption Key (DEK): This key is not managed by Navigator Encrypt or Key Trustee Server. It is managed locally by `dmcrypt` 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`.

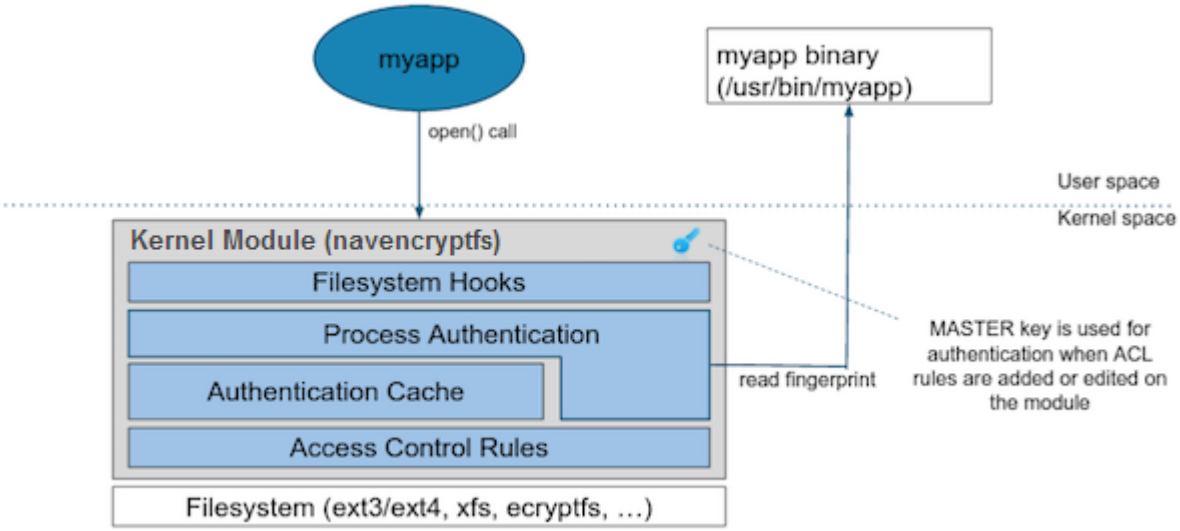
Note: You have the option of using wildcard characters when defining process-based ACLs. The following example shows valid wildcard definitions:

```
"ALLOW @* * *"
"ALLOW @* path/* /path/to/process"
```

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).

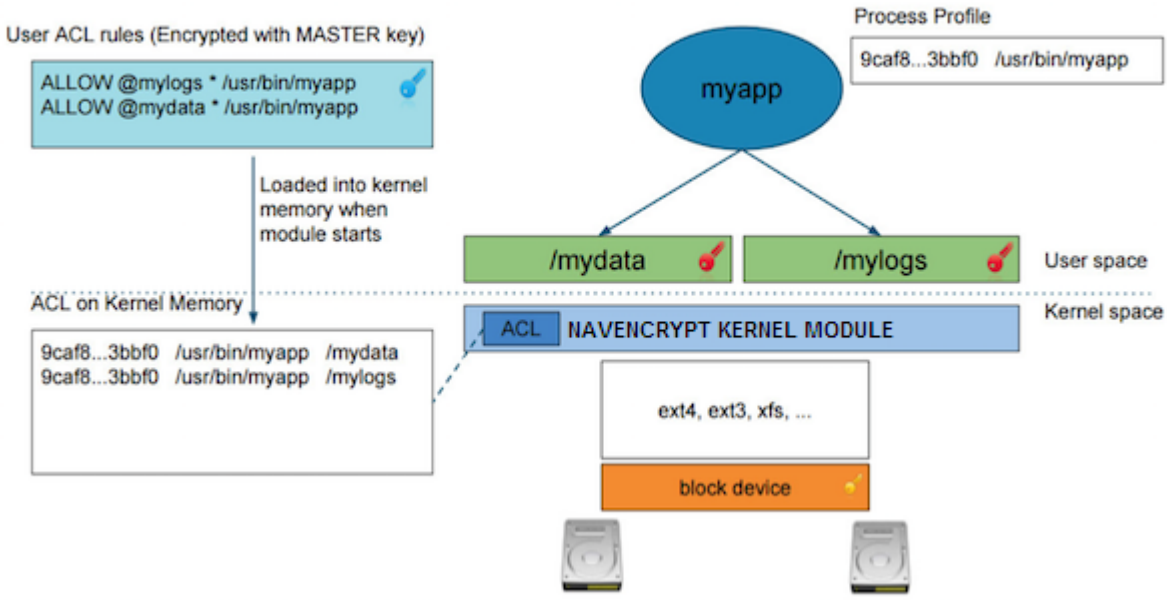
The following diagram shows `/usr/bin/myapp` sending an `open()` call that is intercepted by `navencrypt-kernel-module` as an open hook:



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 `dmcrypt` (block device encryption).

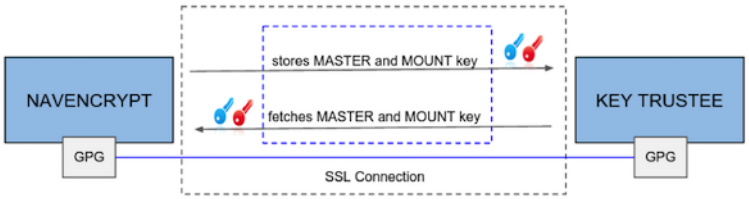
When `myapp` tries to issue I/O to an encrypted directory, the kernel module calculates the fingerprint of the process (`/usr/bin/myapp`) and compares it with the list of authorized fingerprints in the cache.

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.

Appendix: Apache License, Version 2.0

SPDX short identifier: Apache-2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims

licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```