CDP Private Cloud Base 7.1.5

Cloudera Security Overview

Date published: 2020-10-09
Date modified: 2020-11-25

https://docs.cloudera.com/

Legal Notice

Contents

Cloudera Security Overview

A brief introduction to Cloudera security features.

As a system designed to support vast amounts and types of data, Cloudera clusters must meet ever-evolving security requirements imposed by regulating agencies, governments, industries, and the general public. Cloudera clusters comprise both Hadoop core and ecosystem components, all of which must be protected from a variety of threats to ensure the confidentiality, integrity, and availability of all the cluster's services and data. This overview provides introductions to:

Related Information

Authentication Overview

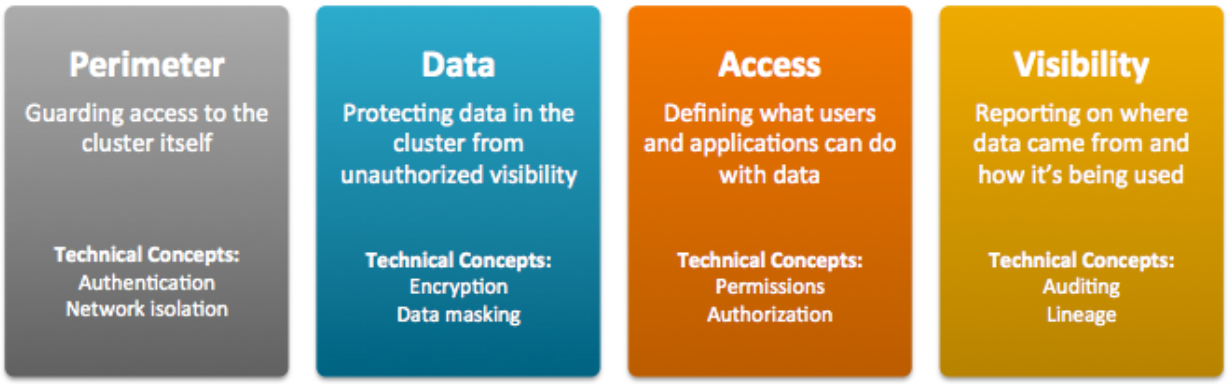
Encryption Overview

Authorization Overview

Using metadata for cluster governance

Security Requirements

Goals for data management systems, such as confidentiality, integrity, and availability, require that the system be secured across several dimensions. These can be characterized in terms of both general operational goals and technical concepts, as shown in the figure below:

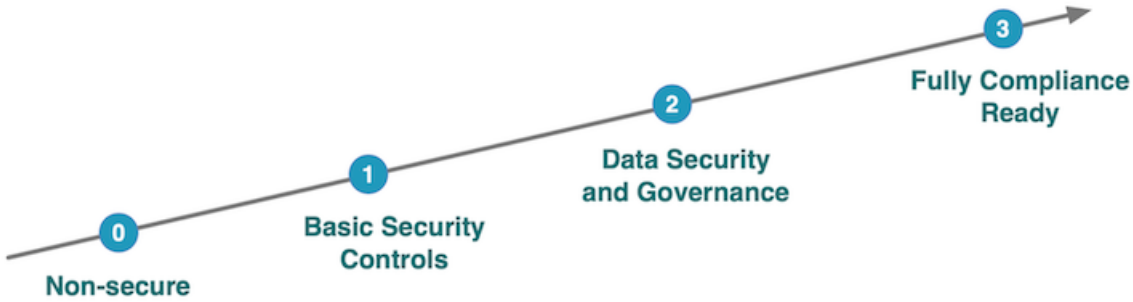


- Perimeter Access to the cluster must be protected from a variety of threats coming from internal and external networks and from a variety of actors. Network isolation can be provided by proper configuration of firewalls, routers, subnets, and the proper use of public and private IP addresses, for example. Authentication mechanisms ensure that people, processes, and applications properly identify themselves to the cluster and prove they are who they say they are, before gaining access to the cluster.
- Data Data in the cluster must always be protected from unauthorized exposure. Similarly, communications between the nodes in the cluster must be protected. Encryption mechanisms ensure that even if network packets are intercepted or hard-disk drives are physically removed from the system by bad actors, the contents are not usable.
- Access Access to any specific service or item of data within the cluster must be specifically granted. Authorization mechanisms ensure that once users have authenticated themselves to the cluster, they can only see the data and use the processes to which they have been granted specific permission.
- Visibility Visibility means that the history of data changes is transparent and capable of meeting data governance policies. Auditing mechanisms ensure that all actions on data and its lineage—source, changes over time, and so on—are documented as they occur.

Securing the cluster to meet specific organizational goals involves using security features inherent to the Hadoop ecosystem as well as using external security infrastructure. The various security mechanisms can be applied in a range of levels.

Security Levels

The figure below shows the range of security levels that can be implemented for a Cloudera cluster, from non-secure (0) to compliance ready (3). As the sensitivity and volume of data on the cluster increases, so should the security level you choose for the cluster.



The table below describes the levels in more detail:

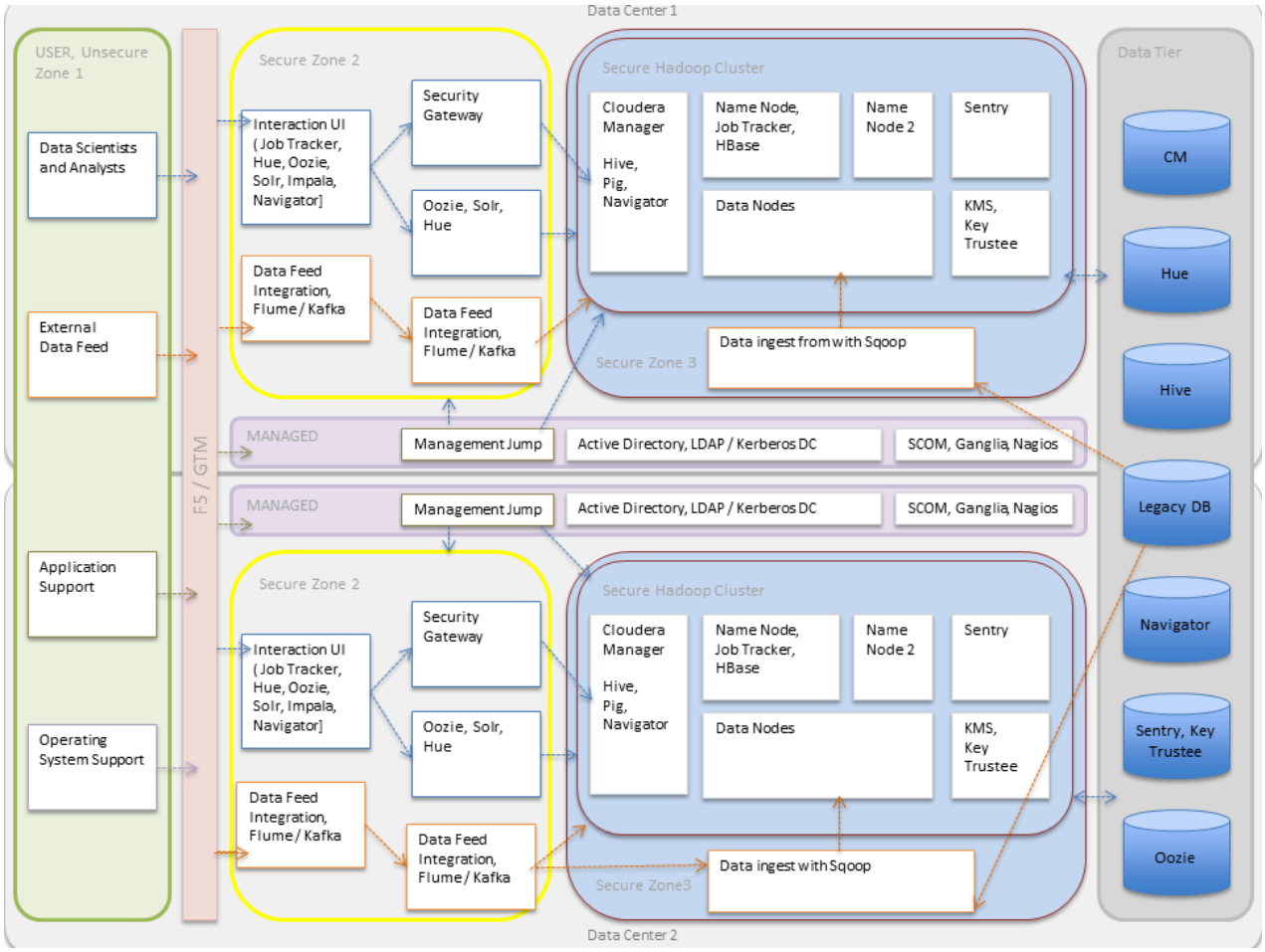
Level	Security	Characteristics
0	Non-secure	No security configured. Non-secure clusters should never be used in production environments because they are vulnerable to any and all attacks and exploits. Recommended only for proof of concept, demo, and disposable platforms.
1	Basic security	Configured for authentication, authorization, and auditing. Authentication is first configured to ensure that users and services can access the cluster only after proving their identities. Next, authorization mechanisms are applied to assign privileges to users and user groups. Auditing procedures keep track of who accesses the cluster (and how). Recommended for development platforms.
2	Data Security and Governance	Sensitive data is encrypted. Key management systems handle encryption keys. Auditing has been setup for data in meta stores. System metadata is reviewed and updated regularly. Ideally, cluster has been setup so that lineage for any data object can be traced (data governance). Recommended for staging and production.
3	Compliance Ready	The secure CDP cluster is one in which all data, both data-at-rest and data-in-transit, is encrypted and the key management system is fault-tolerant. Auditing mechanisms comply with industry, government, and regulatory standards (PCI, HIPAA, NIST, for example), and extend from CDP to the other systems that integrate with it. Cluster administrators are well-trained, security procedures have been certified by an expert, and the cluster can pass technical review. Recommended for all production platforms.

Hadoop Security Architecture

The figure below is an example of some of the many components at work in a production Cloudera enterprise cluster. The figure highlights the need to secure clusters that may ingest data from both internal and external data feeds, and across possibly multiple datacenters. Securing the cluster requires applying authentication and access controls throughout these many inter- and intra-connections, as well as to all users who want to query, run jobs, or even view the data held in the cluster.

- External data streams are authenticated by mechanisms in place for Flume and Kafka. Data from legacy databases is ingested using Sqoop. Data scientists and BI analysts can use interfaces such as Hue to work with data on Impala or Hive, for example, to create and submit jobs. Kerberos authentication can be leveraged to protect all these interactions.
- Encryption can be applied to data at-rest using transparent HDFS encryption with an enterprise-grade Key Trustee Server.

- Authorization policies can be enforced using Ranger (for services such as Hive, Impala, and Search) as well as HDFS Access Control Lists.
- Auditing capabilities can be provided by using Apache Ranger.



Authentication Overview

Overview of Cloudera Manager authentication.

Authentication is a basic security requirement for any computing environment. In simple terms, users and services must prove their identity (authenticate) to the system before they can use system features to the degree authorized. Authentication and authorization work hand-in-hand to protect system resources. Authorization is handled in many different ways, from access control lists (ACLs), to HDFS extended ACLs, to role-based access controls (RBAC) using Ranger.

Several different mechanisms work together to authenticate users and services in a cluster. These vary depending on the services configured on the cluster. Most CDP components, including Apache Hive, Hue, and Apache Impala can use Kerberos for authentication. Both MIT and Microsoft Active Directory Kerberos implementations can be integrated for use with Cloudera clusters.

In addition, Kerberos credentials can be stored and managed in the LDAP-compliant identity service, such as OpenLDAP and Microsoft Active Directory, a core component of Windows Server.

This section provides a brief overview with special focus on different deployment models available when using Microsoft Active Directory for Kerberos authentication or when integrating MIT Kerberos and Microsoft Active Directory.

Cloudera does not provide a Kerberos implementation. Cloudera clusters can be configured to use Kerberos for authentication, either MIT Kerberos or Microsoft Server Active Directory Kerberos, specifically the Key Distribution Center or KDC. The Kerberos instance must be setup and operational before you can configure the cluster to use it.

Gathering all the configuration details about the KDC—or having the Kerberos administrator available to help during the setup process—is an important preliminary task involved with integrating the cluster and Kerberos regardless of the deployment model.

Related Information

Enabling Kerberos Using the Wizard
How to Configure Clusters to Use Kerberos for Authentication
Encrypting Data at Rest
Encrypting Data in Transit

Kerberos Overview

In simple terms, Kerberos is an authentication protocol that relies on cryptographic mechanisms to handle interactions between a requesting client and server, greatly reducing the risk of impersonation. Passwords are not stored locally nor sent over the network in the clear. The password users enter when logging in to their systems is used to unlock a local mechanism that is then used in a subsequent interaction with a trusted third-party to grant a user a ticket (with a limited lifetime) that is used to authenticate with requested services. After the client and server processes prove their respective identities to each other, communications are encrypted to ensure privacy and data integrity.

The trusted third-party is the Kerberos Key Distribution Center (KDC), the focal point for Kerberos operations which also provides the Authentication Service and the Ticket Granting Service (TGS) for the system. Briefly, the TGS issues a ticket to the requesting user or service which is then presented to the requested service that proves the user (or service) identity for the ticket lifetime (by default, 10 hours). There are many nuances to Kerberos, including defining the principals that identify users and services for the system, ticket renewal, delegated token handling, to name a few.

Furthermore, these processes occur for the most part completely transparently. For example, business users of the cluster simply enter their password when they log in, and the ticket-handling, encryption, and other details take place automatically, behind the scenes. Additionally, users are authenticated not only to a single service target, but to the network as a whole thanks to the tickets and other mechanisms at work in the Kerberos infrastructure.

Kerberos Deployment Models

Credentials needed for Kerberos authentication can be stored and managed in an LDAP-compliant identity/directory service, such as OpenLDAP or Microsoft Active Directory.

At one time a stand-alone service offering from Microsoft, Active Directory services are now packaged as part of the Microsoft Server Domain Services. In the early 2000s, Microsoft replaced its NT LAN Manager authentication mechanism with Kerberos. That means that sites running Microsoft Server can integrate their clusters with Active Directory for Kerberos and have the credentials stored in the LDAP directory on the same server.

This section provides overviews of the different deployment models available for integrating Kerberos authentication with Cloudera clusters, with some of the advantages and disadvantages of the available approaches.

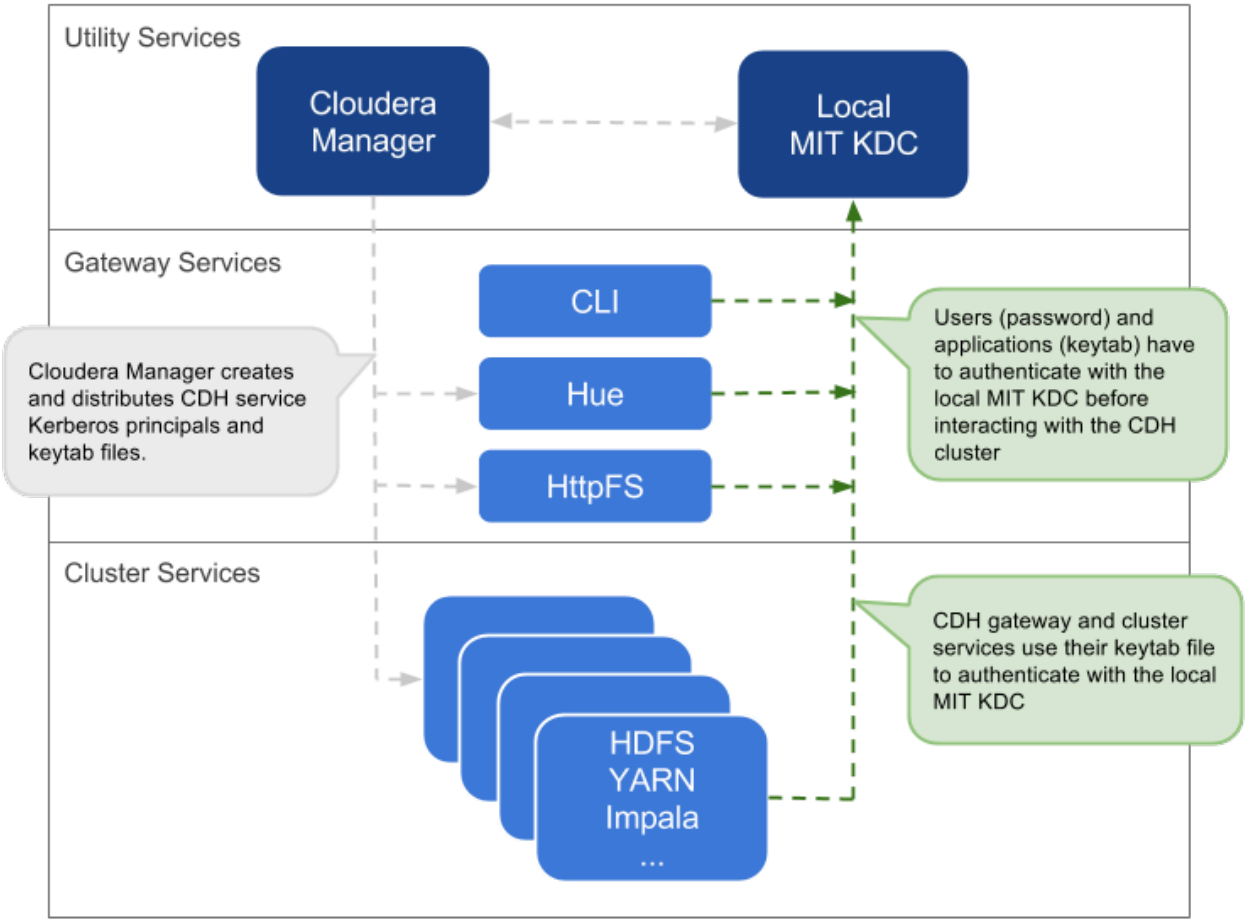
Local MIT KDC

This approach uses an MIT KDC that is local to the cluster. Users and services authenticate to the local KDC before they can interact with the CDP components on the cluster.

Architecture Summary

- An MIT KDC and a separate Kerberos realm is deployed locally to the CDP cluster. The local MIT KDC is typically deployed on a Utility host. Additional replicated MIT KDCs for high-availability are optional.

- All cluster hosts must be configured to use the local MIT Kerberos realm using the krb5.conf file.
- All service and user principals must be created in the local MIT KDC and Kerberos realm.
- The local MIT KDC will authenticate both the service principals (using keytab files) and user principals (using passwords).
- Cloudera Manager connects to the local MIT KDC to create and manage the principals for the CDP services running on the cluster. To do this Cloudera Manager uses an admin principal and keytab that is created during the setup process. This step has been automated by the Kerberos wizard.
- The local MIT KDC administrator typically creates all other user principals. However, the Cloudera Manager Kerberos wizard can create the principals and keytab files automatically.



Pros	Cons
The authentication mechanism is isolated from the rest of the enterprise.	This mechanism is not integrated with central authentication system.
This is fairly easy to setup, especially if you use the Cloudera Manager Kerberos wizard that automates creation and distribution of service principals and keytab files.	User and service principals must be created in the local MIT KDC, which can be time-consuming.
	The local MIT KDC can be a single point of failure for the cluster unless replicated KDCs can be configured for high-availability.
	The local MIT KDC is yet another authentication system to manage.

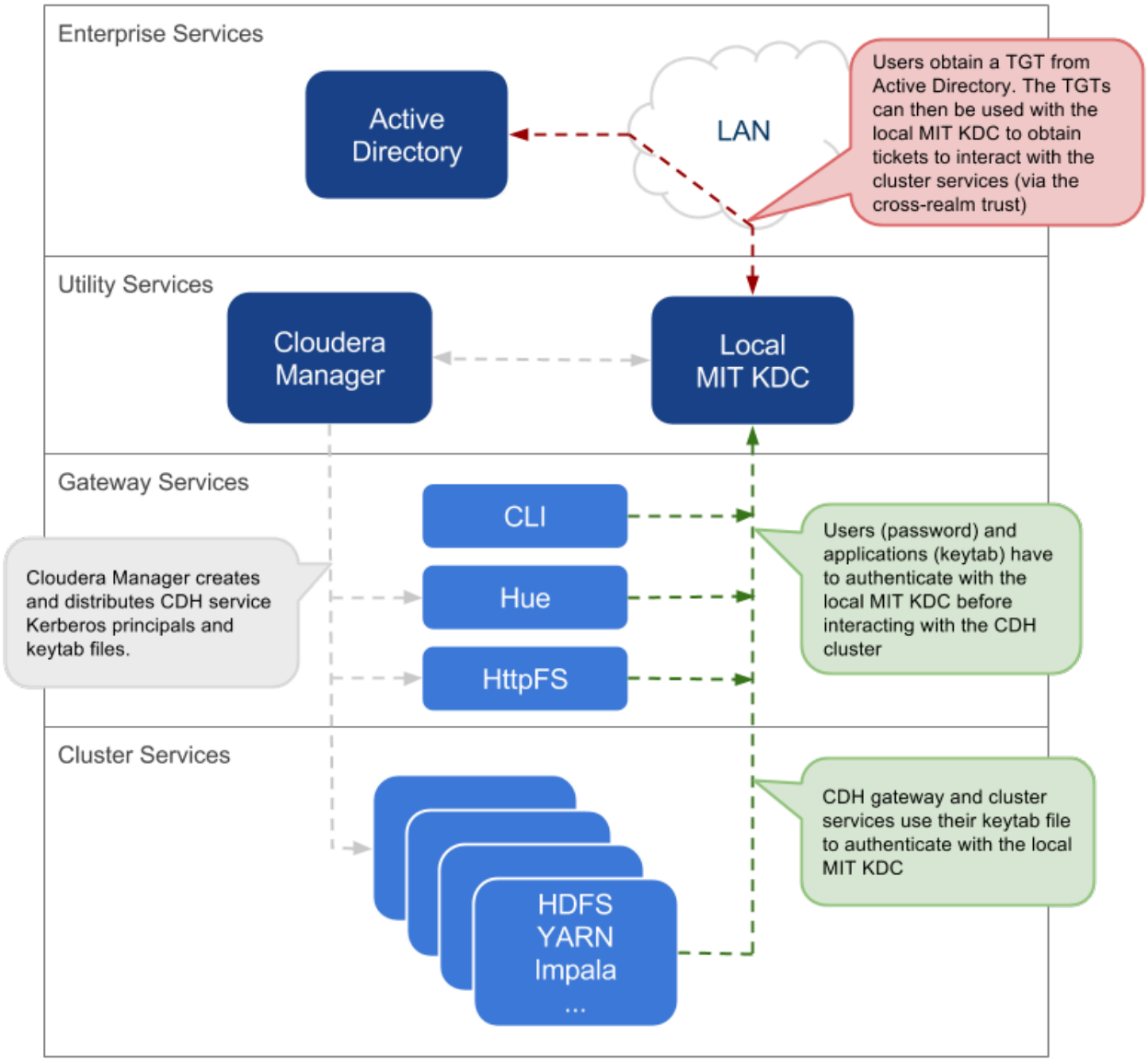
Local MIT KDC with Active Directory Integration

This approach uses an MIT KDC and Kerberos realm that is local to the cluster. However, Active Directory stores the user principals that will access the cluster in a central realm. Users will have to authenticate with this central AD

realm to obtain TGTs before they can interact with CDP services on the cluster. Note that CDP service principals reside only in the local KDC realm.

Architecture Summary

- An MIT KDC and a distinct Kerberos realm is deployed locally to the CDP cluster. The local MIT KDC is typically deployed on a Utility host and additional replicated MIT KDCs for high-availability are optional.
- All cluster hosts are configured with both Kerberos realms (local and central AD) using the krb5.conf file. The default realm should be the local MIT Kerberos realm.
- Service principals should be created in the local MIT KDC and the local Kerberos realm. Cloudera Manager connects to the local MIT KDC to create and manage the principals for the CDP services running on the cluster. To do this, Cloudera Manager uses an admin principal and keytab that is created during the security setup. This step has been automated by the Kerberos wizard.
- A one-way, cross-realm trust must be set up from the local Kerberos realm to the central AD realm containing the user principals that require access to the CDP cluster. There is no need to create the service principals in the central AD realm and no need to create user principals in the local realm.



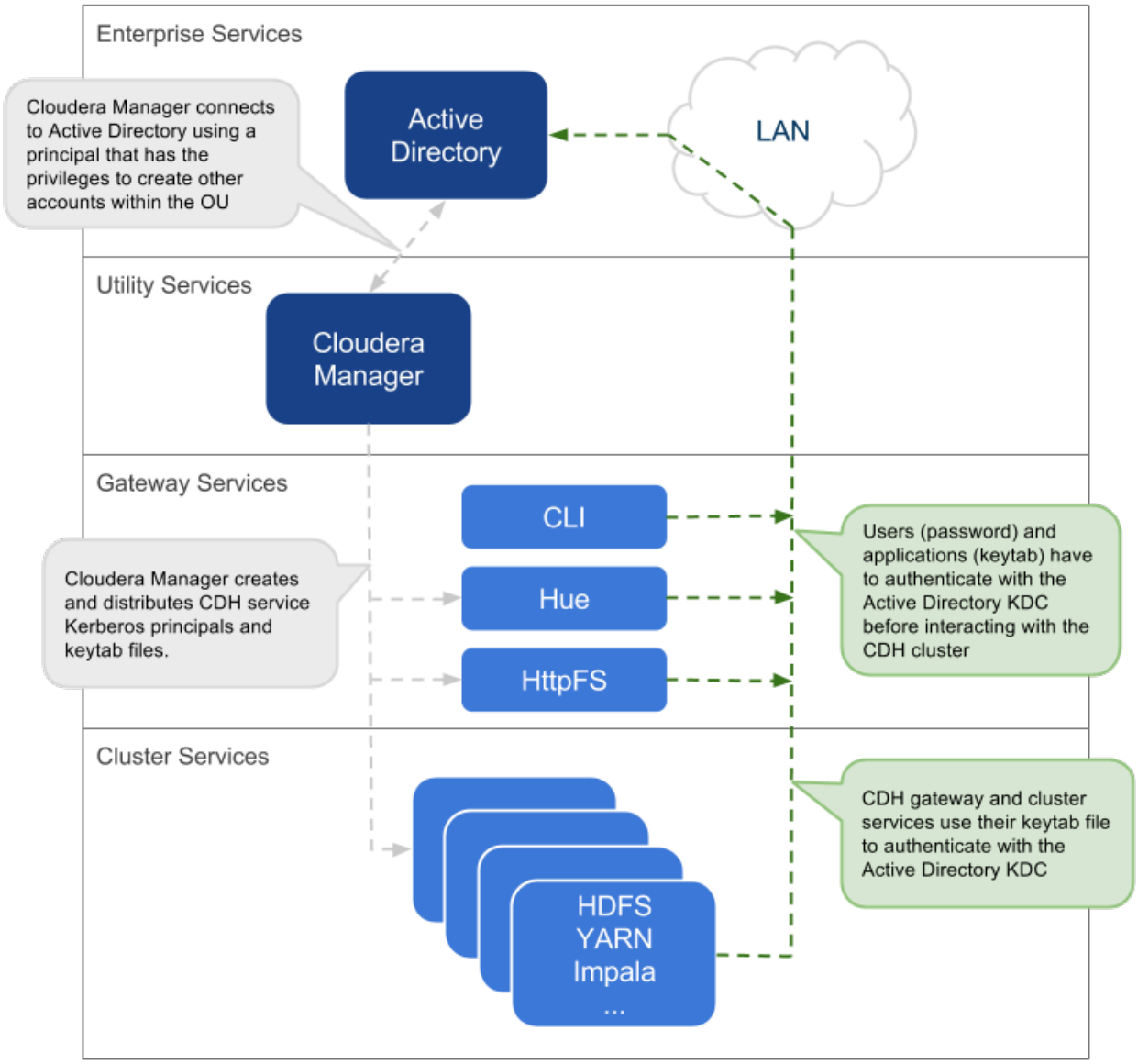
Pros	Cons
The local MIT KDC serves as a shield for the central Active Directory from the many hosts and services in a CDP cluster. Service restarts in a large cluster create many simultaneous authentication requests. If Active Directory is unable to handle the spike in load, then the cluster can effectively cause a distributed denial of service (DDOS) attack.	The local MIT KDC can be a single point of failure (SPOF) for the cluster. Replicated KDCs can be configured for high-availability.
This is fairly easy to setup, especially if you use the Cloudera Manager Kerberos wizard that automates creation and distribution of service principals and keytab files. Active Directory administrators will only need to be involved to configure the cross-realm trust during setup.	The local MIT KDC is yet another authentication system to manage.
Integration with central Active Directory for user principal authentication results in a more complete authentication solution.	
Allows for incremental configuration. Hadoop security can be configured and verified using local MIT KDC independently of integrating with Active Directory.	

Using a Centralized Active Directory Service

This approach uses the central Active Directory as the KDC. No local KDC is required. Before you decide upon an AD KDC deployment, make sure you are aware of the following possible ramifications of that decision.

Architecture Summary

- All service and user principals are created in the Active Directory KDC.
- All cluster hosts are configured with the central AD Kerberos realm using krb5.conf.
- Cloudera Manager connects to the Active Directory KDC to create and manage the principals for the CDP services running on the cluster. To do this, Cloudera Manager uses a principal that has the privileges to create other accounts within the given Organisational Unit (OU). (This step has been automated by the Kerberos wizard.)
- All service and user principals are authenticated by the Active Directory KDC.



Note: If it is not possible to create the Cloudera Manager admin principal with the required privileges in the Active Directory KDC, then the CDP services principals will need to be created manually. The corresponding keytab files should then be stored securely on the Cloudera Manager Server host. Cloudera Manager's Custom Kerberos Keytab Retrieval script can be used to retrieve the keytab files from the local filesystem.

Recommendations for Active Directory KDC

Several different subsystems are involved in servicing authentication requests, including the Key Distribution Center (KDC), Authentication Service (AS), and Ticket Granting Service (TGS). The more nodes in the cluster and the more services provided, the heavier the traffic between these services and the services running on the cluster.

As a general guideline, Cloudera recommends using a dedicated Active Directory instance (Microsoft Server Domain Services) for every 100 nodes in the cluster. However, this is just a loose guideline. Monitor utilization and deploy additional instances as needed to meet the demand.

By default, Kerberos uses TCP for client/server communication which guarantees delivery but is not as fast at delivering packets as UDP. To override this setting and let Kerberos try UDP before TCP, modify the Kerberos configuration file (krb5.conf) as follows:

```
[libdefaults]
```

```
udp_preference_limit = 1
...
```

This is especially useful if the domain controllers are not on the same subnet as the cluster or are separated by firewalls.

In general, Cloudera recommends setting up the Active Directory domain controller (Microsoft Server Domain Services) on the same subnet as the cluster and never over a WAN connection. Separating the cluster from the KDC running on the Active Directory domain controller results in considerable latency and affects cluster performance.

Troubleshooting cluster operations when Active Directory is being used for Kerberos authentication requires administrative access to the Microsoft Server Domain Services instance. Administrators may need to enable Kerberos event logging on the Microsoft Server KDC to resolve issues.

Deleting Cloudera Manager roles or nodes requires manually deleting the associate Active Directory accounts. Cloudera Manager cannot delete entries from Active Directory.

Identity Integration with Active Directory

A core requirement for enabling Kerberos security in the platform is that users have accounts on all cluster processing nodes. Commercial products such as Centrify or Quest Authentication Services (QAS) provide integration of all cluster hosts for user and group resolution to Active Directory. These tools support automated Kerberos authentication on login by users to a Linux host with AD. For sites not using Active Directory, or sites wanting to use an open source solution, the Site Security Services Daemon (SSSD) can be used with either AD or OpenLDAP compatible directory services and MIT Kerberos for the same needs.

For third-party providers, you may have to purchase licenses from the respective vendors. This procedure requires some planning as it takes time to procure these licenses and deploy these products on a cluster. Care should be taken to ensure that the identity management product does not associate the service principal names (SPNs) with the host principals when the computers are joined to the AD domain. For example, Centrify by default associates the HTTP SPN with the host principal. So the HTTP SPN should be specifically excluded when the hosts are joined to the domain.

You will also need to complete the following setup tasks in AD:

- Active Directory Organizational Unit (OU) and OU user - A separate OU in Active Directory should be created along with an account that has privileges to create additional accounts in that OU.
- Enable SSL for AD - Cloudera Manager should be able to connect to AD on the LDAPS (TCP 636) port.
- Principals and Keytabs - In a direct-to-AD deployment that is set up using the Kerberos wizard, by default, all required principals and keytabs will be created, deployed and managed by Cloudera Manager. However, if for some reason you cannot allow Cloudera Manager to manage your direct-to-AD deployment, then unique accounts should be manually created in AD for each service running on each host and keytab files must be provided for the same. These accounts should have the AD User Principal Name (UPN) set to service/fqdn@REALM, and the Service Principal Name (SPN) set to service/fqdn. The principal name in the keytab files should be the UPN of the account. The keytab files should follow the naming convention: servicename_fqdn.keytab. The following principals and keytab files must be created for each host they run on:
- AD Bind Account - Create an AD account that will be used for LDAP bindings in Hue, Cloudera Manager and Cloudera Navigator.
- AD Groups for Privileged Users - Create AD groups and add members for the authorized users, HDFS admins and HDFS superuser groups.
    - Authorized users – A group consisting of all users that need access to the cluster
    - HDFS admins – Groups of users that will run HDFS administrative commands
    - HDFS super users – Group of users that require superuser privilege, that is, read/wwrite access to all data and directories in HDFS

      Putting regular users into the HDFS superuser group is not recommended. Instead, an account that administrators escalate issues to, should be part of the HDFS superuser group.

- AD Groups for Role-Based Access to Cloudera Manager and Cloudera Navigator - Create AD groups and add members to these groups so you can later configure role-based access to Cloudera Manager and Cloudera Navigator.
- AD Test Users and Groups - At least one existing AD user and the group that the user belongs to should be provided to test whether authorization rules work as expected.

Using TLS/SSL for Secure Keytab Distribution

The Kerberos keytab file is transmitted among the hosts in the Cloudera Manager cluster, between Cloudera Manager Server and Cloudera Manager Agent hosts. To keep this sensitive data secure, configure Cloudera Manager Server and the Cloudera Manager Agent hosts for encrypted communications using TLS/SSL.

Using the Wizard or Manual Process to Configure Kerberos Authentication

Cloudera does not provide a Kerberos implementation but uses an existing Kerberos deployment to authenticate services and users. The Kerberos server may be set up exclusively for use by the cluster (for example, Local MIT KDC on page 7) or may be a distributed Kerberos deployment used by other applications in the organization.

Regardless of the deployment model, the Kerberos instance must be operational before the cluster can be configured to use it. In addition, the cluster itself should also be operational and ideally, configured to use TLS/SSL for Cloudera Manager Server and Cloudera Manager Agent hosts, as mentioned above.

When you are ready to integrate the cluster with your organization's MIT KDC or Active Directory KDC, you can do so using the wizard provided in Cloudera Manager Server or by following a manual process, as follows:

- Enabling Kerberos Authentication Using the Wizard
- How to Configure Clusters to Use Kerberos for Authentication

Authentication Mechanisms used by Cluster Components

Component or Product	Authentication Mechanism Supported
Accumulo	Kerberos (partial)
Backup and Disaster Recovery	Kerberos (used to authenticate Cloudera Manager to Kerberos-protected services), LDAP, SAML
Cloudera Manager	Kerberos (used to authenticate Cloudera Manager to Kerberos-protected services), LDAP, SAML
Cloudera Navigator	Active Directory, OpenLDAP, SAML
HBase	Kerberos, user-based authentication required for HBase Thrift and REST clients
HDFS	Kerberos, SPNEGO (HttpFS)
HiveServer	None
HiveServer2	Kerberos, LDAP, Custom/pluggable authentication
Hive Metastore	Kerberos
Hue	Kerberos, LDAP, SAML, Custom/pluggable authentication
Impala	Kerberos, LDAP, SPNEGO (Impala Web Console)
Kudu	Kerberos
MapReduce	Kerberos (also see HDFS)
Oozie	Kerberos, SPNEGO
Pig	Kerberos
Search	Kerberos, SPNEGO
Spark	Kerberos

Component or Product	Authentication Mechanism Supported
Sqoop	Kerberos
YARN	Kerberos (also see HDFS)
Zookeeper	Kerberos

Encryption Overview

Encryption is a process that uses digital keys to encode various components—text, files, databases, passwords, applications, or network packets, for example—so that only the appropriate entity (user, system process, and so on) can decode (decrypt) the item and view, modify, or add to the data. Cloudera provides encryption mechanisms to protect data persisted to disk or other storage media (*data at rest encryption* or simply, data encryption) and as it moves over the network (*data in transit encryption*).

Data encryption is mandatory in government, health, finance, education, and many other environments. For example, the Federal Information Security Management Act (FISMA) governs patient privacy concerns and the Payment Card Industry Data Security Standard (PCI DSS) regulates information security for credit-card processors. These are just two examples.

The vast quantity of data contained in Cloudera clusters, deployed using many different components, must nonetheless support whatever degree of privacy, confidentiality, and data integrity is required by the use case. The encryption mechanisms supported by Cloudera and discussed in this overview aim to do just that.

Related Information

Encrypting Data in Transit

Encrypting Data at Rest

Protecting Data At-Rest

Protecting data at rest typically means encrypting the data when it is stored on disk and letting authorized users and processes—and only authorized users and processes—to decrypt the data when needed for the application or task at hand. With data-at-rest encryption, encryption keys must be distributed and managed, keys should be rotated or changed on a regular basis (to reduce the risk of having keys compromised), and many other factors complicate the process.

However, encrypting data alone may not be sufficient. For example, administrators and others with sufficient privileges may have access to personally identifiable information (PII) in log files, audit data, or SQL queries. Depending on the specific use case—in hospital or financial environment, the PII may need to be redacted from all such files, to ensure that users with privileges on the logs and queries that might contain sensitive data are nonetheless unable to view that data when they should not.

Cloudera provides complementary approaches to encrypting data at rest, and provides mechanisms to mask PII in log files, audit data, and SQL queries.

Encryption Options Available

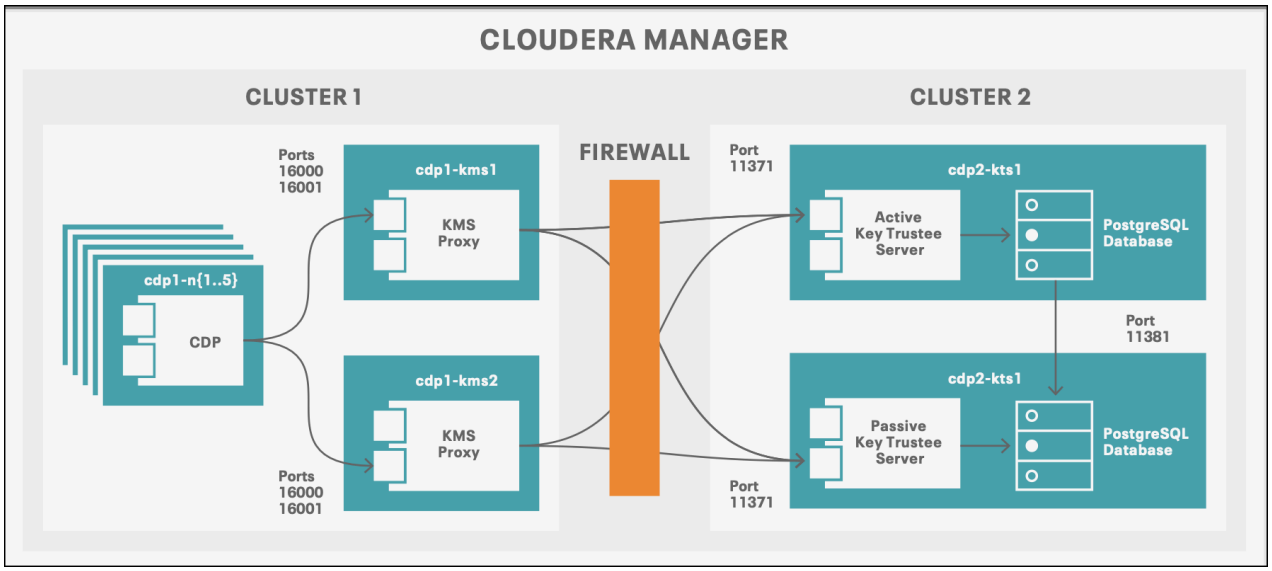
Cloudera provides several mechanisms to ensure that sensitive data is secure. CDP provides transparent HDFS encryption, ensuring that all sensitive data is encrypted before being stored on disk. HDFS encryption when combined with the enterprise-grade encryption key management of Key Trustee enables regulatory compliance for most enterprises. For Cloudera Enterprise, HDFS encryption can be augmented by Navigator Encrypt to secure metadata, in addition to data. Cloudera clusters that use these solutions run as usual and have very low performance impact, given that data nodes are encrypted in parallel. As the cluster grows, encryption grows with it.

Additionally, this transparent encryption is optimized for the Intel chipset for high performance. Intel chipsets include AES-NI co-processors, which provide special capabilities that make encryption workloads run extremely fast. Cloudera leverages the latest Intel advances for even faster performance.

The Ranger Key Management System (KMS), used in conjunction with Key Trustee Server and Key HSM, provides HSM-based protection of stored key material. The Ranger KMS generates encryption zone key material locally on the KMS and then encrypts this key material using an HSM-generated key. The Ranger KMS remains the recommended key management solution for HDFS encryption for most production scenarios.

The following figure shows an example deployment that uses Cloudera Transparent HDFS Encryption to encrypt data stored on HDFS.

Figure 1: Logical Architecture for Encrypting Data at Rest



In addition to applying encryption to the data layer of a Cloudera cluster, encryption can also be applied at the network layer, to encrypt communications among nodes of the cluster. See Encryption Mechanisms Overview on page 17 for more information.

Encryption does not prevent administrators with full access to the cluster from viewing sensitive data. To obfuscate sensitive data, including PII, the cluster can be configured for data redaction.

Data Redaction for Cloudera Clusters

Redaction is a process that obscures data. It can help organizations comply with industry regulations and standards, such as PCI (Payment Card Industry) and HIPAA, by obfuscating personally identifiable information (PII) so that is not usable except by those whose jobs require such access. For example, HIPAA legislation requires that patient PII not be available to anyone other than appropriate physician (and the patient), and that any patient's PII cannot be used to determine or associate an individual's identity with health data. Data redaction is one process that can help ensure this privacy, by transforming PII to meaningless patterns—for example, transforming U.S. social security numbers to XXX-XX-XXXX strings.

Data redaction works separately from Cloudera encryption techniques, which do not preclude administrators with full access to the cluster from viewing sensitive user data. It ensures that cluster administrators, data analysts, and others cannot see PII or other sensitive data that is not within their job domain and at the same time, it does not prevent users with appropriate permissions from accessing data to which they have privileges.

See How to Enable Sensitive Data Redaction for details.

Protecting Data In-Transit

For data-in-transit, implementing data protection and encryption is relatively easy. Wire encryption is built into the Hadoop stack, such as SSL, and typically does not require external systems. This data-in-transit encryption is built using session-level, one-time keys, by means of a session handshake with immediate and subsequent transmission. Thus, data-in-transit avoids much of the key management issues associated with data-at-rest due the temporal nature of the keys, but it does rely on proper authentication; a certificate compromise is an issue with authentication, but can compromise wire encryption. As the name implies, data-in-transit covers the secure transfer and intermediate storage of data. This applies to all process-to-process communication, within the same node or between nodes. There are three primary communication channels:

- HDFS Transparent Encryption: Data encrypted using HDFS Transparent Encryption is protected end-to-end. Any data written to and from HDFS can only be encrypted or decrypted by the client. HDFS does not have access to the unencrypted data or the encryption keys. This supports both, at-rest encryption as well as in-transit encryption.
- Data Transfer: The first channel is data transfer, including the reading and writing of data blocks to HDFS. Hadoop uses a SASL-enabled wrapper around its native direct TCP/IP-based transport, called DataTranspor tProtocol, to secure the I/O streams within an DIGEST-MD5 envelope. This procedure also employs secured HadoopRPC (see Remote Procedure Calls) for the key exchange. The HttpFS REST interface, however, does not provide secure communication between the client and HDFS, only secured authentication using SPNEGO.

  For the transfer of data between DataNodes during the shuffle phase of a MapReduce job (that is, moving intermediate results between the Map and Reduce portions of the job), Hadoop secures the communication channel with HTTP Secure (HTTPS) using Transport Layer Security (TLS).
- Remote Procedure Calls: The second channel is system calls to remote procedures (RPC) to the various systems and frameworks within a Hadoop cluster. Like data transfer activities, Hadoop has its own native protocol for RPC, called HadoopRPC, which is used for Hadoop API client communication, intra-Hadoop services communication, as well as monitoring, heartbeats, and other non-data, non-user activity. HadoopRPC is SASL-enabled for secured transport and defaults to Kerberos and DIGEST-MD5 depending on the type of communication and security settings.
- User Interfaces: The third channel includes the various web-based user interfaces within a Hadoop cluster. For secured transport, the solution is straightforward; these interfaces employ HTTPS.

TLS/SSL Certificates Overview

Certificates can be signed in one three different ways:

Type	Usage Note
Public CA-signed certificates	Recommended. Using certificates signed by a trusted public CA simplifies deployment because the default Java client already trusts most public CAs. Obtain certificates from one of the trusted well-known (public) CAs, such as Symantec and Comodo.
Internal CA-signed certificates	Obtain certificates from your organization's internal CA if your organization has its own. Using an internal CA can reduce costs (although cluster configuration may require establishing the trust chain for certificates signed by an internal CA, depending on your IT infrastructure).
Self-signed certificates	Not recommended for production deployments. Using self-signed certificates requires configuring each client to trust the specific certificate (in addition to generating and distributing the certificates). However, self-signed certificates are fine for non-production (testing or proof-of-concept) deployments.

TLS/SSL Encryption for CDP Components

Cloudera recommends securing a cluster using Kerberos authentication before enabling encryption such as SSL on a cluster. If you enable SSL for a cluster that does not already have Kerberos authentication configured, a warning will be displayed.

Hadoop services differ in their use of SSL as follows:

- HDFS, MapReduce, and YARN daemons act as both SSL servers and clients.
- HBase daemons act as SSL servers only.

- Oozie daemons act as SSL servers only.
- Hue acts as an SSL client to all of the above.

Daemons that act as SSL servers load the keystores when starting up. When a client connects to an SSL server daemon, the server transmits the certificate loaded at startup time to the client, which then uses its truststore to validate the server's certificate.

For information on setting up SSL/TLS for CDP services, see the applicable component guide.

Data Protection within Hadoop Projects

The table below lists the various encryption capabilities that can be leveraged by CDP components and Cloudera Manager.

Project	Encryption for Data-in-Transit	Encryption for Data-at-Rest (HDFS Encryption + Navigator Encrypt + Key Trustee)
HDFS	SASL (RPC), SASL (DataTransferProtocol)	Yes
MapReduce	SASL (RPC), HTTPS (encrypted shuffle)	Yes
YARN	SASL (RPC)	Yes
Accumulo	Partial - Only for RPCs and Web UI (Not directly configurable in Cloudera Manager)	Yes
HBase	SASL - For web interfaces, inter-component replication, the HBase shell and the REST, Thrift 1 and Thrift 2 interfaces	Yes
HiveServer2	SASL (Thrift), SASL (JDBC), TLS (JDBC, ODBC)	Yes
Hue	TLS	Yes
Impala	TLS or SASL between impalad and clients, but not between daemons	
Oozie	TLS	Yes
Search	TLS	Yes
Ranger	SASL (RPC)	Yes
Spark	None	Yes
Sqoop	Partial - Depends on the RDBMS database driver in use	Yes
ZooKeeper	SASL (RPC)	No
Cloudera Manager	TLS - Does not include monitoring	Yes
Backup and Disaster Recovery	TLS - Also see Cloudera Manager	Yes

Encryption Mechanisms Overview

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

Layer	Description
Application	Applied by the HDFS client software, HDFS Transparent Encryption lets you encrypt specific folders contained in HDFS. To securely store the required encryption keys, Cloudera recommends using the Ranger Key Trustee Server in conjunction with HDFS encryption. Data stored temporarily on the local filesystem outside HDFS by CDP components (including Impala, MapReduce, YARN, or HBase) can also be encrypted.
Operating System	At the Linux OS file system layer, encryption can be applied to an entire volume.

Layer	Description
Network	Network communications between client processes and server processes (HTTP, RPC, or TCP/IP services) can be encrypted using industry-standard TLS/SSL.

Here are some good starting places for more information about encryption for Cloudera clusters:

- Encrypting data at rest
- Encrypting data in transit

Related Information

Encrypting Data at Rest
Encrypting Data in Transit

Authorization Overview

Authorization is one of the fundamental security requirements of any computing environment. Its goal is to ensure that only the appropriate people or processes can access, view, use, control, or change specific resources, services, or data. In any cluster deployed to meet specific workloads using various CDH components (Hive, HDFS, Impala, and so on), different authorization mechanisms can ensure that only authorized users or processes can access data, systems, and other resources as needed. Ideally, authorization mechanisms can leverage the authentication mechanisms, so that when users login to a system—a cluster, for example—they are transparently authorized based on their identity across the system for the applications, data, and other resources they are authorized to use.

For example, Cloudera CDH clusters can be configured to leverage the user and group accounts that exist in the organization's Active Directory (or other LDAP-accessible directory) instance.

Note: Authorization through Apache Ranger is just one element of a secure production cluster: Cloudera supports Ranger only when it runs on a cluster where Kerberos is enabled to authenticate users.

The various possible configurations and integrations are discussed later in this guide.

Related Information

Using Ranger to Provide Authorization in CDP

Authorization Mechanisms in Hadoop

Hadoop supports several authorization mechanisms, including:

- Traditional POSIX-style permissions on files and directories. Each directory and file has a single owner and group with basic permissions that can be set to read, write, execute (at the file level). Directories have an additional permission that enables access to child directories.
- Access Control Lists (ACL) for management of services and resources. For example, 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 users and groups. Fine-grained permissions can be applied to HDFS files using Apache HDFS ACLs to set permissions for specific named users and named groups.
- Apache Ranger manages access control through and ensures consistent policy administration across cluster services. See Using Ranger to Provide Authorization in CDP for more information.
- Apache Ranger also provides a centralized framework for collecting access audit history and reporting data, including filtering on various parameters. See Managing Auditing with Ranger for more information.

POSIX Permissions

Most services running on Hadoop clusters, such as the command-line interface (CLI) or client applications that use Hadoop API, directly access data stored within HDFS. HDFS uses POSIX-style permissions for directories and files; each directory and file is assigned a single owner and group. Each assignment has a basic set of permissions

available; file permissions are read, write, and execute, and directories have an additional permission to determine access to child directories.

Ownership and group membership for a given HDFS asset determines a user's privileges. If a given user fails either of these criteria, they are denied access. For services that may attempt to access more than one file, such as MapReduce, Cloudera Search, and others, data access is determined separately for each file access attempt. File permissions in HDFS are managed by the NameNode.

Access Control Lists

Hadoop also maintains general access controls for the services themselves in addition to the data within each service and in HDFS. Service access control lists (ACL) are typically defined within the global hadoop-policy.xml file and range from NameNode access to client-to-DataNode communication. In the context of MapReduce and YARN, user and group identifiers form the basis for determining permission for job submission or modification.

In addition, with MapReduce and YARN, jobs can be submitted using queues controlled by a scheduler, which is one of the components comprising the resource management capabilities within the cluster. Administrators define permissions to individual queues using ACLs. ACLs can also be defined on a job-by-job basis. Like HDFS permissions, local user accounts and groups must exist on each executing server, otherwise the queues will be unusable except by superuser accounts.

Apache HBase also uses ACLs for data-level authorization. HBase ACLs 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. Local user accounts are required for proper authorization, similar to HDFS permissions.

Apache ZooKeeper also maintains ACLs to the information stored within the DataNodes of a ZooKeeper data tree.

Integration with Authentication Mechanisms for Identity Management

Like many distributed systems, Hadoop projects and workloads often consist of a collection of processes working in concert. In some instances, the initial user process conducts authorization throughout the entirety of the workload or job's lifecycle. But for processes that spawn additional processes, authorization can pose challenges. In this case, the spawned processes are set to execute as if they were the authenticated user, that is, setuid, and thus only have the privileges of that user. The overarching system requires a mapping to the authenticated principal and the user account must exist on the local host system for the setuid to succeed.

Important:

- Cloudera strongly recommends against using Hadoop's LdapGroupsMapping provider. LdapGroupsMapping should only be used in cases where OS-level integration is not possible. Production clusters require an identity provider that works well with all applications, not just Hadoop. Hence, often the preferred mechanism is to use tools such as SSSD, VAS or Centrify to replicate LDAP groups.
- Cloudera does not support the use of Winbind in production environments. Winbind uses an inefficient approach to user/group mapping, which may lead to low performance or cluster failures as the size of the cluster, and the number of users and groups increases.

Irrespective of the mechanism used, user/group mappings must be applied consistently across all cluster hosts for ease with maintenance.

System and Service Authorization - Several Hadoop services are limited to inter-service interactions and are not intended for end-user access. These services do support authentication to protect against unauthorized or malicious users. However, any user or, more typically, another service that has login credentials and can authenticate to the service is authorized to perform all actions allowed by the target service. Examples include ZooKeeper, which is used by internal systems such as YARN, Cloudera Search, and HBase, and Flume, which is configured directly by Hadoop administrators and thus offers no user controls.

The authenticated Kerberos principals for these "system" services are checked each time they access other services such as HDFS, HBase, and MapReduce, and therefore must be authorized to use those resources. Thus, the fact that Flume does not have an explicit authorization model does not imply that Flume has unrestricted access to HDFS and

other services; the Flume service principals still must be authorized for specific locations of the HDFS file system. Hadoop administrators can establish separate system users for a services such as Flume to segment and impose access rights to only the parts of the file system for a specific Flume application.

Authorization within Hadoop Projects

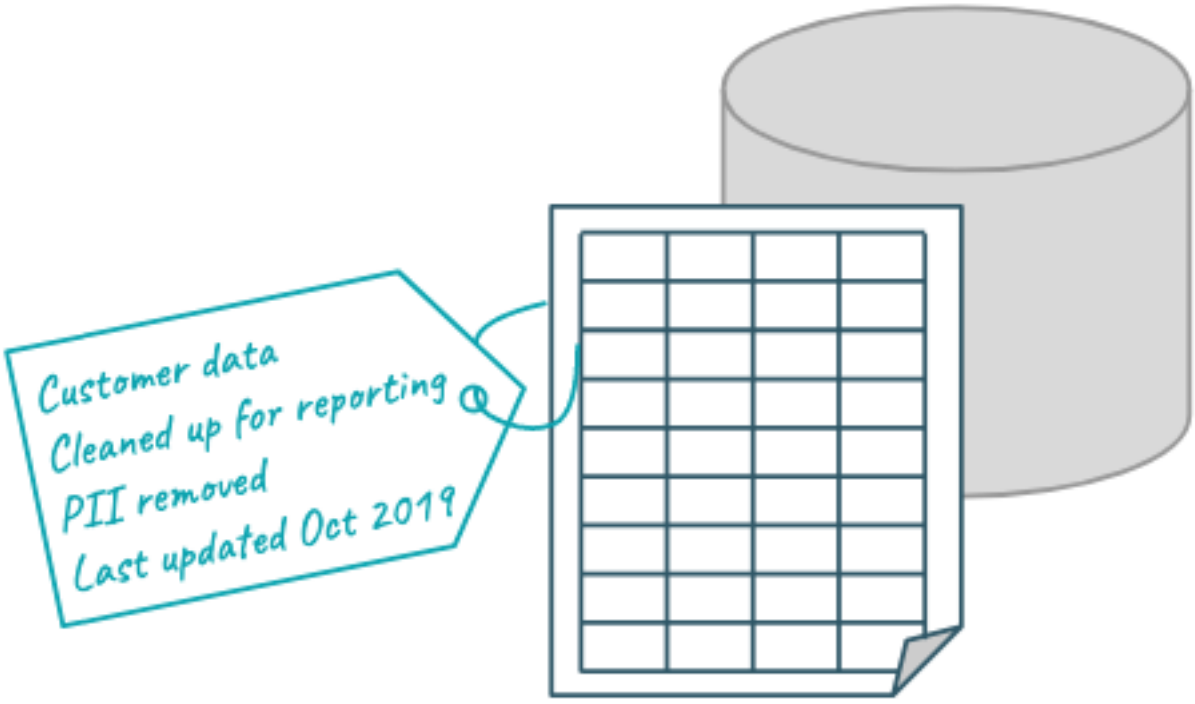
Project	Authorization Capabilities
HDFS	File Permissions, Ranger
MapReduce	File Permissions, Ranger
YARN	File Permissions, Ranger
Accumulo	Ranger
HBase	HBase ACLs, Ranger
HiveServer2	File Permissions, Ranger
Hue	Hue authorization mechanisms (assigning permissions to Hue apps)
Impala	Ranger
Oozie	ACLs
Pig	File Permissions
Search	File Permissions
Spark	File Permissions, Ranger
Sqoop	N/A
ZooKeeper	ACLs
Cloudera Manager	Cloudera Manager roles
Backup and Disaster Recovery	N/A

Using metadata for cluster governance

Concepts for collecting, creating, and using metadata.

What is Apache Atlas?

Atlas is a metadata management and governance system designed to help you find, organize, and manage data assets. Atlas creates “entities” or metadata representations of objects and operations in your data lake. You can add business metadata to these entities so you can use business vocabulary to make it easier to search for specific assets.



Apache Atlas uses metadata to create lineage relationships

Atlas reads the content of the metadata it collects to build relationships among data assets. When Atlas receives query information, it notes the input and output of the query and generates a lineage map that traces how data is used and transformed over time. This visualization of data transformations allows governance teams to quickly identify the source of data and to understand the impact of data and schema changes.

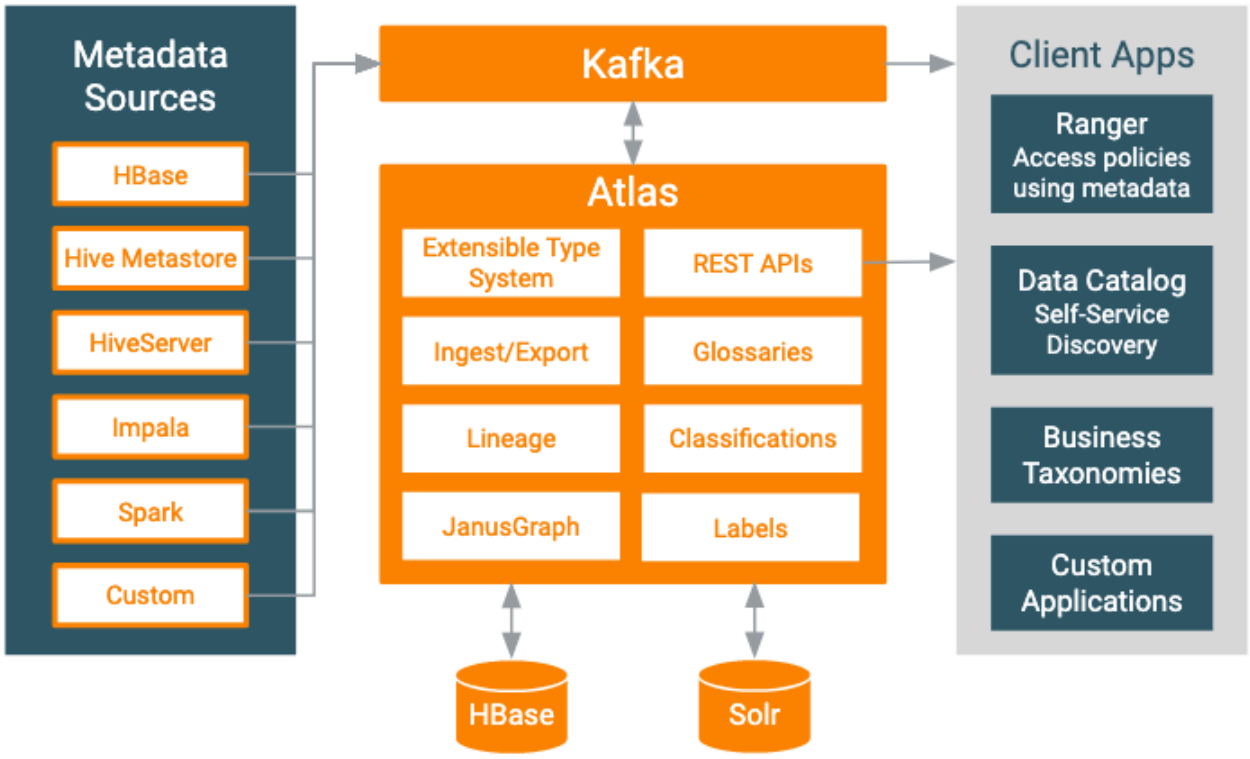
Adding to entity metadata makes searching easier

Atlas manages *classifications* and *labels* that you create and use to enhance the metadata for data assets. You can create and organize classifications and labels to use for anything from identifying data cleansing stages to recording user comments and insights on specific data assets. When you use classifications, the Atlas Dashboard makes it easy to search, group, report, and further annotate the entities you label. Classifications themselves can be organized into hierarchies to make them easier to manage.

Atlas also provides an infrastructure to create and maintain business ontologies to label your data assets. Atlas' "glossaries" include "terms" so you can build agreed-upon lists for department- or organization-wide vocabulary to identify and manage data. Adding a term gives you a single-click report of entities identified by that term.

Apache Atlas architecture

Atlas runs as an independent service in a Hadoop environment. Many Hadoop data processing and storage services include Atlas add-ons that publish metadata for the services' activities to a Kafka message topic. Atlas reads the messages and stores them in JanusGraph to model the relationships among entities. The datastore behind JanusGraph is HBase. Atlas stores a search index in Solr to take advantage of Solr's search functionality.



Pre-defined hooks exist for Hive, Impala, Kafka, NiFi, Spark, and Sqoop.

Atlas also provides “bridges” that import metadata for all of the existing data assets in a given source. For example, if you start Atlas after you’ve already created databases and tables in Hive, you can import metadata for the existing data assets using the Hive bridge. Bridges use the Atlas API to import the metadata rather than publishing messages to Kafka.

If you need a hook or bridge to automate collecting metadata from another source, use the Atlas Java API to create a custom Atlas addon.

Note: Governance through Apache Atlas is just one element of a secure production cluster: Cloudera supports Atlas only when it runs on a cluster where Kerberos is enabled to authenticate users.

Related Information

Governance