Cloudera Data Platform Disaster Recovery

Cloudera Disaster Recovery

Date published: 2022-06-21
Date modified: 2022-06-21

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2026. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Abstract

This document brings together a series of recommendations, best-practices and links to resources that help in the planning and understanding of disaster recovery solutions for Cloudera.

It is written for practitioners involved in the planning and implementation of business continuity, disaster recovery and backup procedures. It outlines a set of definitions and recommendations aligned with open industry standards, provides high-level recommendations and a checklist of resources and documents that can be used for informed decision-making. The document does not make specific recommendations with respect to measures or controls because these vary on a case by case basis.

Definitions

Before you use Cloudera disaster recovery solutions, you might want to understand the definitions of certain terms that are related to disaster recovery operations.

Disaster recovery and business continuity planning

Disaster Recovery (DR)

Disaster recovery concerns the collected set of tools, equipment, policies, and procedures required to recover a business environment in the event of some disaster. This disaster can be a natural or environmental event such as flood or fire, it can be a technical issue such as power loss or critical equipment failure, or can be caused by human factors (inadvertent or otherwise). Disaster recovery solutions are often designed with two or more geographically-separated environments to facilitate returning business processes back to a normal standard of operation, even in the event that one environment is unrecoverable.

Business Continuity Planning (BCP)

Business continuity planning is a superset of the disaster recovery design process. With business continuity planning, the goal is to identify the items that minimally need to be running as soon as possible after a disaster-level event has occurred. While all business processes may need to be restored after an event, each process has a particular priority that defines how soon that must occur and what level of data loss is tolerable from that event. Two metrics that help define this are the Recovery Point Objective (RPO) and Recovery Time Objective (RTO).

Seven tiers for disaster recovery

The seven tiers for disaster recovery are used to provide a concise definition. The tiers that are most relevant to this document are tier 4 and 5.

Tier 4: Point-in-time recovery

In this tier, point-in-time copies of storage volumes, files or dates are created at a specific moment, such as end-of-day. The copies are stored in an active-secondary site, either company-owned or provided by a third party. Copies are stored at both the primary and secondary sites, each site backing up the other. Disk and solid-state storage are used in this tier, as well as cloud storage technology.

Cloudera backup and disaster recovery is designed to deliver a tier 4 capability between one or more separate clusters in different regions.

Tier 5: Two-site commit/transaction integrity

At this tier, data is continuously transmitted to primary and alternate backup sites. Organizations need substantial network bandwidth, often provided through the internet, to maintain a constant flow of data. Local as well as remote storage can be used in this situation. Cloud storage is often a fit at this tier.

If higher levels of disaster recovery are required then certain services and applications can be configured to achieve it. For example Kafka in combination with Streams Replication or HBase in an active-active configuration can achieve these requirements. However HDFS can only typically achieve tier 4.

For more information, see Seven tiers of disaster recovery.

Recovery Point Objective and Recovery Time Objective

Recovery Point Objective (RPO)

The Recovery Point Objective defines how much data loss is tolerated as a direct outcome of an incident. Another way to think of this is how recent your data needs to be on the secondary site to permit recovery operations. This metric is measured over a particular time period for recovery purposes. For example, if you can only tolerate five minutes worth of data loss, having a disaster recovery strategy that synchronizes data hourly is insufficient. The RPO directly affects how frequently your disaster recovery environments must be synchronized or backed up to survive an incident. Additionally, this also drives the design of the disaster recovery environment. As you move closer to near- or real-time data synchronization, depending on your storage mechanism of choice, you may need to switch from periodic data synchronizing to dual-ingestion to achieve your RPO requirements.

Recovery Time Objective (RTO)

The Recovery Time Objective is a separate, but a related metric. This metric defines the amount of time you have to restore normal service to your business systems before you begin negatively impacting the business processes. In other words, how long after the incident can the system be down before the normal recovery process is impacted, such as requiring additional manual work to re-run processes.

High availability

When we talk about high availability (HA), we think about it in terms of both intra- and inter-cluster behavior. Additionally, availability requirements must be considered for the entire stack, from the Cloudera platform up to all applications built on top.

To minimize the risk of single components taking out an individual environment, we look at the single points of failure and attempt to reasonably address those within that system. As an example, we enable multiple HDFS NameNodes within Cloudera Base on premises to allow the system to survive individual service issues. This helps reduce the need to fail-over to an entire disaster recovery environment, which further reduces the operational work required to recover from an individual service issue.

Additionally, we can think about high availability across equivalently configured clusters through the use of load balancers and suitable data, metadata, and workload synchronization. Dual-ingesting data into multiple clusters can help facilitate this mode of HA. In some cases, such as cost minimization, you may consider reducing availability within a cluster because overall availability is sufficiently provided across clusters.

Cloudera recommends considering high availability at all levels for all environments because the operational and runtime benefits far outweigh the cost reductions.

Backups

Backups are the processes, procedures, and equipment necessary to have a recoverable snapshot of the data, metadata, and related environmental information. Backing up data, metadata, and workload information is a form of disaster recovery preparation, but the time scales associated with restoring service may preclude it from being a sole source of recovery (that is, the recovery time is higher). Relying only on backups misses critical parts of the architecture such as replacement equipment, time to provision new equipment and environments, and so on.

Where backups come into play with disaster recovery is in the case of managing data corruption or data deletion. Without backups, the risk in a paired set of disaster recovery clusters is data corruption or deletion propagating between the two.

Environment naming

Disaster recovery clusters are traditionally designed in pairs or more. Cloudera recommends describing them in terms of Left/Right, A/B, or through regional monikers such as East/West. When naming environments like Production/Production-DR, you can get into some confusion post-failover, when Production-DR effectively becomes “Production”, and the original environment finally comes back online.

Active-Active and Active-Passive

Disaster Recovery architectures are designed in terms of Active-Active or Active-Passive.

Note: Unlike, for example an RDBMS, the Cloudera clusters do not have an explicit passive mode. Even in an Active-Passive architecture, the clusters are always Active. In this section, we are explicitly referring to Active and Passive application architectures, rather than clusters.

Active-Active

In Active-Active designs, writes can go into each side of the disaster recovery pair, with the expectation that new or changed data is correctly and quickly replicated. Read requests can go to either side of the pair with some understanding that they may not be consistent within the bounds of replication latency behavior. For example, replication happens asynchronously in Cloudera. If a client writes to cluster A and immediately attempts to read that from cluster B because of a failover incident, the data may not look consistent to the client. Data consistency is only guaranteed within a single cluster environment.

Active-Passive

In Active-Passive designs, one half of the pair is declared the primary. All reads and writes go to the primary, the passive cluster being configured as a read-only replica to reduce risks of inconsistencies. All data changes are replicated to the secondary environment. Generally, clients are never exposed to the secondary environment unless an incident occurs that requires failover between clusters. In some Active-Passive designs, clients may be directed to read from secondary environments when higher data latency can be tolerated. This allows you to utilize the additional idle capacity on the secondary cluster with the understanding that those use cases may need to be halted during a failure event.

Stretch clusters

A stretch cluster is a single logical cluster that is placed across multiple physical locations in a way to take advantage of cluster high availability mechanics. In general, stretch clusters are not well suited for disaster recovery scenarios, despite the reliance upon the cluster’s high-availability components. This cluster implementation type has several notable disadvantages including minimum location count, significant impact on performance due to latency, and incapability of all cluster or cluster management components of high availability operation. Additionally, you may need to provide more spare capacity to account for cluster behavior during failure situations, such as re-replication activities that occur within HDFS or Ozone when nodes go offline.

Minimum location set

Stretch clusters can only operate redundantly with three distinct locations because of the architectural design of lower level structural components needed by HDFS. Attempts to operate with two locations leave the cluster prone to failure when quorum operations occur during a critical event such as losing the dominant half of odd-numbered quorum services.

Latency

Lower level services, such as HDFS, YARN, and Zookeeper, are prone to significant negative performance impact as you stretch network latency. HDFS, for example, must wait for the data block write pipeline to complete before providing a consistent response to client write operations. Individual YARN jobs can see poor read performance if tasks are dispatched to nodes not co-located with the required data. The larger the network latency, the lower the overall read and write performance.

Planning overview (Aligned with NIST)

While NIST Policies are generally directed toward US Federal Organizations, their definitions, policies, and processes can be aligned with and adopted by other organizations in other industries. The open and accessible nature of the standards also make them easily accessed and referenced. NIST 800-34 provides guidance on contingency planning for Federal Organizations and outlines seven stages of contingency planning with associated samples.

Develop the contingency planning policy

This phase identifies the roles and responsibilities, the scope of application to platforms and organizational functions, the resource, training requirements, and the schedules for exercising, testing and maintenance.

Business impact analysis

This phase helps determine the mission, business processes and their recovery criticality. At this stage the downtime and impact is considered. The analysis of the resources required to resume the mission and business processes, and finally an understanding of the order and priorities and the sequence of recovery.

This analysis has a direct impact on the cost and budgeting required to restore an enterprise to functionality. Classify business processes and workloads by how quickly they need to be restored to functionality after an event. Some workloads might need to be continuous throughout the event. Some can tolerate being restored days, weeks, or even months after. Understanding this drives the design decisions around the practical length for RPO and RTO, as well as influences sizing and operational requirements around your disaster recovery environment.

Identify preventive controls

Implement preventative measures such as implementing commodity hardware that includes enterprise features such as hot swap drives and power supplies. Select a data center that provides resilient networking and fire protection. Take regular backups and implement high-availability or stretched clusters to reduce the risk of a disaster event .

Create contingency strategies and plan

This includes the implementation of regular and comprehensive backup and recovery, and the methods for backup and strategy for local or off-site storage. Also taken into consideration is the use of multiple geographic sites. If a disaster occurs, then consider the method for replacement of equipment and the lead times involved. This should all be considered against overall cost and budget. Once suitable contingencies are identified, then consider the teams who are assigned to implement them.

Plan testing, training, and exercise

Organizations should conduct test, training and exercise regularly to validate the plans, changes in those plans or changes to the systems involved. Each event should be documented and used to inform the update and maintenance of the plans.

Disaster recovery testing frequency depends entirely on the customer’s requirements and tolerance for impact to the business. All failover events are disruptive to the business. How disruptive they are depends entirely on how automated the procedure is and how rigorously documented and how tolerant the business process is to potential gaps in data, metadata, schema, policy, and governance information.

Cloudera recommends that customers regularly exercise failover procedures and become comfortable operating in either A or B environments.

Plan maintenance

The plan must be maintained in a ready state that reflects the system and procedures, organizational structure and policies. As these change based on changing business needs, system upgrades or new use case deployment, the plan must be updated and maintained.

Prerequisites

This document assumes that you have undertaken a series of reviews aligned with industry best practice in contingency planning, backup, and disaster recovery. We have included the section above to help align with the NIST guidance.

This document assumes that technically you have created a resilient, highly available cluster design that implements appropriate high availability and load balancing choices for the workloads associated with that cluster.

This document assumes that clusters are managed equivalently, minimizing configuration drift between the two environments that may make failover operations risky or prone to failure.

Recommendations

When undertaking Business Impact Analysis, consider the workloads that run the business, being more important than ones that assist the business.

Not all workloads need to be replicated, allowing for smaller disaster recovery environments, with the option to scale up post-disaster, or offload to Cloudera on cloud.

Some environments require full workload replication, necessitating duplicate equipment, configuration, and change management procedures for each half of the disaster recovery pair.

Cloudera does not recommend using hybrid replication from on premises to on cloud for only data replication. Simply backing up data to the cloud may be insufficient for providing reliable disaster recovery capabilities. Both the data and workload must be replicated in both form factors. This ensures that during a disaster recovery event, the workload can still operate on the data in either form factor. Cloudera provides a common runtime so that applications can run consistently in both form factors. For that to happen the application and data have to be available.

Types of failure

When thinking about disaster recovery and identifying preventative controls, or contingency planning, you need to assess your environment in terms of risk and impact when it comes to particular types of failures. Failures can happen all the time, at all layers of the stack, from physical hardware up to individual service components. Not every failure requires triggering your disaster recovery procedure.

Failure states and associated impacts

The Hadoop Files System (HDFS) and Ozone are architected from the ground up to be reliable and robust. It is designed with the expectation of failure. As clusters get larger and the volume of data being managed grows, then that data is replicated across many more disks. Eventually a disk, node or rack will fail.

The following table outlines the failure states and associated impacts:

Failure Type	Impact	Failover
Single disk failure Leader node	Low to Moderate	Intra-cluster failover between high availability services
Single disk failure Worker node	Low	Allow automated Cloudera cluster recovery features to trigger
Multi-disk failure Leader node	Low to Moderate	Intra-cluster failover between high availability services

Failure Type	Impact	Failover
Multi-disk failure Worker node	Low	Allow automated Cloudera cluster recovery features to trigger
Single node failure Leader node	Low to Moderate	Intra-cluster failover between high availability services
Single node failure Worker node	Low	Allow automated Cloudera cluster recovery features to trigger
Multi-node failure Leader node	Moderate to High	Consider disaster recovery failover if not recoverable by RTO period
Multi-node failure Worker node	Low to Moderate	Allow automated Cloudera cluster recovery features to trigger; if significant number of nodes are affected, consider triggering disaster recovery failover if not recoverable by RTO period
Network-link between sites	Moderate to High	Consider disaster recovery failover if not recoverable by RTO period
Single-site failure	High	Trigger disaster recovery failover to secondary site
Multi-site failure	Very High	Recover single site and trigger disaster recovery failover

As the radius of impact increases during a failure event, the likelihood of a failover increases. When considering how to mitigate the event, take this impact into account. For example, a single-node failure can be mitigated within a cluster using available replicas or, in the case of leader nodes, enabling high availability services. As you move into higher impact events, towards single- or multi-site failure, the need to trigger a full failover operation with your disaster recovery plan becomes larger.

Types of failure

Failing and recovering gracefully

When considering an Active-Passive architecture, how the environment fails drives the mechanism and planning for how you recover to a normal operational state after the disaster. Given most failover events are triggered as part of disaster recovery testing, it is important to determine how you interchange the role of primary and secondary sites.

Fail Over/Fail Back

Fail Over/Fail Back means you have a designated primary and secondary at all times. These designations rarely change. Your primary environment is a complete unit of data and workloads. The secondary site may be a complete unit of data and workloads. Failing over to the secondary is considered a time-limited event.

Fail Forward

Fail Forward means you have a time-oriented primary and secondary. The designation changes whenever a failure event occurs. The primary environment is a complete unit of data and workloads. The current Active environment is always the primary, meaning the primary designation can interchange between your environments. The current Passive environment is the secondary.

Fail Over versus Fail Forward

The two methods have different recovery challenges. With Fail Forward, your environments need to be sized equivalently. Resynchronizing the primary to the secondary is less disruptive to workloads because you always push new or changed data to the secondary. This limits the need to have a prolonged workload outage when interchanging the Active environment.

With Fail Over, your secondary environment can be smaller because it may be focused only on those business critical functions. Resynchronizing data back to the primary is difficult because you

need to pause updates to both environments, synchronize the changes, and coordinate moving back the workloads. Workloads may be offline for a prolonged period to facilitate the resynchronizing of data back to the primary.

Periodic revalidation

As part of your normal operating procedure, you should periodically revalidate that the data that exists in the primary environment also exists in the secondary environment. This helps to identify gaps, corruption, or general differences in data between the two environments that may negatively impact you during a critical event. This process is aligned with the developing preventative controls such as validating replication logs and tracking record counts across both clusters. The Data Catalog and Data Profiling tools can assist with this.

Disaster recovery checklist and resources

A suitable disaster recovery plan consists of multiple items. These include high availability mitigations, backup and recovery, as well as data and workload replication.

A Cloudera environment consists of several services, components, and data storage mechanisms and dependencies. Some of these live outside the envelope of the cluster, such as the database systems used by services like Hue and Hive. When considering if and how to replicate components within the entire stack, take into account the capabilities of the service. In some cases, you may want to use the native replication capability within the service, such as HBase. In other cases, you may need to use an external tool, such as Cloudera Replication Manager to coordinate the movement of data between HDFS on your disaster recovery clusters.

Workload

A workload or application consists of the services, data, schema and business metadata, security and governance policy, and supporting workload scripts and workflows that allow a business application to run correctly.

When planning the disaster recovery architecture, you must account for each direct or indirect dependency within a workload to ensure it works correctly after a failure event.

Data

Data is maintained on a file system or object store service such as HDFS or Ozone. Data may also exist in higher level abstractions maintained by a service such as HBase or Solr, which create their own file formats on HDFS. In other cases, for services such as Kudu or Kafka, the data is placed and maintained directly on a local disk.

File systems

Cloudera has multiple deployment methodologies on cloud and on premises. These form factors have multiple, interrelated mechanisms for storing data at various organizational levels within the cluster. Each layer has specific requirements related to how and whether data should be replicated. These layers include local file storage on individual nodes, HDFS, and Ozone.

HDFS

HDFS must be configured for high availability (HA). Three components that enable high availability of data on HDFS are Namenode, Journalnode, and Datanode services. Multiple cluster leader nodes run both Namenode and JournalNode services to maintain the filesystem metadata. Datanode services run on each worker node, maintaining the individual data blocks and providing data availability through block replication.

For information about enabling high availability on HDFS, see Enabling HA on HDFS with Cloudera Manager.

HDFS utilizes the DistCp tool to replicate data between clusters. Cloudera utilizes Cloudera Replication Manager within Cloudera Manager to manage the replication policies for a given directory structure in HDFS. Cloudera

Replication Manager utilizes a version of the DistCp tool specific for Cloudera, to integrate with Cloudera cluster management. Replication policies are triggered on a periodic basis, providing an asynchronous replication path for data on HDFS. HDFS supports replicating encrypted data safely across clusters. HDFS Transparent Data Encryption must be enabled for encrypting data at rest.

For information about utilizing Cloudera Replication Manager for HDFS data replication, see HDFS Replication

Ozone

Ozone must be configured for high availability. Three components that enable high availability for data on Ozone are Ozone Manager, Storage Container Manager, and Datanode services. The Ozone Manager and Storage Container Manager run on multiple leader nodes to maintain availability of the object store metadata and background operational tasks. Datanode services run on each Ozone worker node, maintaining individual data blocks and providing data availability through replication.

For information about enabling high availability in Ozone, see Storage Container Manager HA and Ozone Manager HA.

Backup can be achieved by using DistCp to replicate data between separate clusters. Currently, this is facilitated through direct use of the DistCp tool and manually scheduling these through an orchestration tool like Oozie.

Local file system

You may have local file system synchronization requirements between the two disaster recovery clusters. These areas are outside the scope of this document and Cloudera replication capabilities do not cover this use case.

Data Stores

Separate data stores might exist on top of HDFS and Ozone or at the local file system level. These data stores might have their own replication or backup and restore mechanisms used in conjunction with traditional Hadoop replication strategies. Services might have additional availability capabilities and configurations to help establish intra-cluster high availability. This creates a robust and fault tolerant environment when paired with cross-cluster replication.

Data Lake

For the Cloudera on cloud form factor, it is possible to backup and restore the metadata maintained in the Data Lake services. The backup and restore operation creates a comprehensive backup that improves the likelihood of data in the backup to be synchronized across all the services running in the Data Lake.

There is a downtime when a Data Lake backup is performed, as some Data Lake services are stopped. Additionally, access to the HMS/Ranger databases are blocked for the duration of the backup.

Note: A Data Lake backup includes metadata about your cluster workloads and does not include the data itself. For more information, see Backup a Cloudera on cloud Data Lake

HBase

HBase should be deployed in High Availability (HA) mode by configuring backup leader nodes. You must configure high availability when HBase applications require low-latency queries and can tolerate minimal (near-zero-second) staleness for read operations.

For information about enabling HBase with High Availability, see Enabling HBase HA using Cloudera Manager

HBase has a built-in replication mechanism for replicating writes between clusters. This is one of the few services that can work in an Active-Active configuration. HBase utilizes the Write Ahead Log (WAL) to track changes that need to be shipped to the peer cluster. Once the peer relationship is established, HBase automatically manages shipping WAL changes to minimize administrative overhead, but can require administrator intervention in certain conditions where the peer becomes unavailable. When planning, pay close attention to replication WAL backlogs when a cluster is offline. These WAL backlogs can fill HDFS, causing additional issues in environments with significant data churn.

Peer replication status is monitored through built-in HBase status commands. Integration into alerting systems requires additional customer integration.

When Phoenix is implemented in conjunction with HBase, use the native HBase replication mechanism to synchronize data across disaster recovery environments.

For information about enabling HBase replication, see Deploying HBase Replication

Kudu

Kudu maintains its own in-memory state and uses local disks to maintain persistent copies of data. Kudu has no built-in cluster-to-cluster replication. When planning disaster recovery for Kudu, consider two instances and a dual ingest pattern. It is possible to backup Kudu tables but these are one-off snapshots at a single point in time.

Review the Cloudera documentation for Kudu recovery Backing up and Recovering Apache Kudu

Cloudera Search (Solr)

Cloudera Search includes a backup and restore mechanism primarily designed to provide disaster recovery capability for Apache Solr. You can create a backup of a Solr collection by creating and exporting a snapshot and restoring from this backup to recover from scenarios, such as accidental or malicious data modification or deletion. When planning for disaster recovery for Cloudera Search, consider two instances and a dual ingest pattern. It is possible to backup Solr tables but these are one off snapshots at a single point in time.

Review the Cloudera documentation for Search recovery Backing up and restoring data using Cloudera Search

Hive and Impala data

Hive and Impala data generally reside on HDFS. Hive and Impala replication enables you to copy your Hive metastore and data from one cluster to another. You can synchronize the Hive metastore and data on the destination cluster with the source, based on a specified replication policy.

There are two mechanisms to replicate Hive data on HDFS. The mechanism you choose depends on whether your tables are defined as managed or external.

The following table lists the table type and the replication method you can use to replicate data:

Table Type	Replication Style
Hive 3 Managed Transactional	Hive built-in Replication
Hive 3 External	Cloudera Replication Manager Hive replication policies

Note: Cloudera Replication Manager currently copies the Hive schema and data details. It does not copy the surrounding metadata and authorization policy in Apache Ranger and Apache Atlas. If you require to synchronize this information, you must do this manually. Both Apache Ranger and Apache Atlas provide import and export APIs to facilitate this. Refer to their respective sections below for more information.

In cases where Hive and Impala utilize non-HDFS storage backends to manage data, such as Kudu or Phoenix on HBase, you must utilize the underlying storage mechanism to manage replication directly. If the underlying storage mechanism has no-native replication capability, such as Kudu, then dual-ingest solutions can be used to keep disaster recovery environments in sync.

For information about using Cloudera Replication Manager to create Hive replication policies, see Replicating Hive and Impala.

Metadata

Metadata refers to the schema and data required for correctly running Hadoop SQL workloads on top of Hive, Impala, or SparkSQL. Additionally, it also refers to business metadata created within Atlas to assign additional context to datasets. Hive, Impala, and SparkSQL share a metadata catalog for databases, tables, and schemas. When replicating or backing up data managed by these services, you need to consider both the data and catalog metadata together.

Hive Metastore

The Hive Metastore (HMS) manages the metadata related to databases and table schema. Additionally, it tracks where the underlying table data resides on HDFS or Ozone.

HMS should be deployed in a High Available (HA) mode by configuring an additional HMS instance. This provides failover capabilities within a cluster to a secondary Hive Metastore if your primary instance goes down.

For more information about the Hive Metastore, see Configuring the HMS for High Availability.

Metadata within the HMS is generally co-replicated with the underlying Hive and Impala data when using Cloudera Replication Manager. This activity is configurable using the Cloudera Replication Manager replication policy associated with the Hive data set.

Atlas

Atlas should be deployed in High Availability mode by configuring additional Atlas service instances. This provides failover capabilities within a cluster if the primary instance goes down.

Atlas depends upon HBase, Kafka, and the Infrastructure Solr. When enabling Atlas HA, these services must also be enabled for high availability support.

For more information about Atlas High Availability, see About Atlas High Availability.

Business metadata generated and stored within Atlas is used for data tagging and lineage. This information is not currently synchronized between disaster recovery environments. Should this be required, the Atlas metadata can be exported from one environment and imported to the other using Atlas REST API calls.

Streaming and event processing

The Stream Replication Engine is a next generation multi-cluster and cross data center replication engine for Kafka clusters.

Streams Replication Manager is an enterprise-grade replication solution that enables fault tolerant, scalable and robust cross-cluster Kafka topic replication. Streams Replication Manager provides the ability to dynamically change configurations and keeps the topic properties in sync across clusters at high performance. For more information about Streams Replication Manager, see Streams Replication Manager OverviewStreams Replication Manager Overview. For information about configuration and planning, see Streams Replication Manager RequirementsStreams Replication Manager Requirements.

Security and governance

Cloudera implements security and governance using Kerberos for authentication, Apache Ranger for policy enforcement and auditing, and Apache Atlas for metadata tagging and lineage.

Authentication

Cloudera Base on premises clusters in a disaster recovery pair utilize Kerberos to mutually authenticate each other for data replication and service access purposes. The clusters must exist in a common Kerberos realm, or two realms with some form of trust. One-way or cross-realm Kerberos trusts are acceptable. Refer to your operating system manuals for details on implementing Kerberos trusts.

Authorization

Cloudera utilizes Apache Ranger to implement a common authorization policy framework across services and tracks user activity within those services. Services across a cluster interface with Ranger using a plugin framework. When a service uses Ranger policies, the service loads a plugin module that manages the authorization decisions, local service auditing, and syncing policy updates from the Ranger Admin interface.

Ranger Admin UI and Ranger Service plugins

Note: Ranger and RangerKMS are separate, but related services. This section deals with specifics for Ranger. The Encryption Support topic provides details about Ranger KMS.

For clusters that have multiple users and production, or production-like availability requirements, you must enable Ranger high availability. The two components that must be configured in each running service are the Ranger Admin UI and the Ranger plugins. Ranger high availability can be implemented with or without a load balancer. A load balancer can be utilized to provide a common accessible Ranger Admin UI host name for operators and administrators. When Ranger high availability is enabled without a load balancer, the Ranger plugins are configured to be aware of all Ranger Admin instances.

In the case of one or more Ranger Admin instances going offline, Ranger plugins keep a local cache of downloaded policy so the authorization decisions continue. The Ranger plugin automatically tracks changes to these policies and downloads updates when the Ranger Admin instances return to service.

For information about Ranger high availability, see Configuring Apache Ranger High Availability.

Ranger policy synchronization

In a complex disaster recovery environment, authorization policies must be synchronized across both clusters. This ensures that the workloads seamlessly move across disaster recovery cluster pairs in the event of a failure. Policy synchronization is currently implemented through export and import operations with the Ranger Admin API, or through external management of policy configurations that are pushed into the Ranger Admin service for each disaster recovery cluster pair.

Because HDFS high availability requires unique Name Service identifiers for all clusters, care must be taken when crafting Ranger policies to account for embedded HDFS service names. In particular, HDFS policies and Hadoop SQL policies can utilize HDFS path URIs, where Name Service identifiers can be found. When exporting and importing policies across clusters, review and convert policy exports are required to account for these name differences.

Auditing

Ranger auditing is managed on a per-cluster basis. Ranger plugins log the audit details to both HDFS for long-term storage and to the infrastructure Solr instance for supporting audit search in the Ranger Admin UI. In the event that a Ranger plugin cannot write an audit log to either service, the plugin locally caches the audit event and attempts to replay it at a later time.

When you enable Ranger high availability (HA), you must also enable high availability on the HDFS and infrastructure Solr dependencies.

Cloudera makes no attempt to synchronize audit details across clusters in a disaster recovery setup. If unified audits are required, consider exporting the audit details to an external mechanism, such as a security information and event management (SIEM) framework.

Encryption support

Cloudera implements both in-flight and at-rest encryption for data. In-flight encryption is enabled using TLS. In Cloudera on premises, at-rest encryption is enabled in HDFS through integration with a Key Management Server (KMS) and a Key Trustee Server (KTS). In some cases, a Hardware Security Module (HSM) might be integrated with the KTS to enhance key storage functionality. In Cloudera on cloud, server-side encryption (SSE) is used.

Encryption key access high availability

Cloudera utilizes the Ranger KMS to manage key requests between a cluster and the keystore. Cloudera supports using either a Key Trustee Server or database to act as the keystore.

To enable fault tolerant access between the cluster and the keystore, you should enable Ranger KMS in high availability mode. If utilizing a RangerKMS with database as the keystore, you must also implement appropriate

database-level high availability to ensure keys are accessible. If utilizing the Ranger KMS with KTS, you must also implement the KTS service as a highly available pair. Implementing Ranger KMS with KTS is the preferred method for configuring HDFS Transparent Data Encryption.

KTS high availability applies to read operations only. If either KTS fails, the client automatically retries fetching keys from the functioning server. New write operations (for example, creating new encryption keys) are not allowed unless both KTS are operational.

Encryption key material is intended to be implemented on a per-cluster basis. Replication of keys across clusters is not supported. When replicating encrypted data between clusters, Replication Manager re-encrypts data in-flight using appropriate keys from the destination cluster.

If utilizing an HSM, ensure that the HSM is implemented in a highly available way. This is dependent upon each HSM manufacturer.

For information about Ranger KMS and KTS, see Configuring Ranger KMS for High Availability and Setting up Key Trustee Server High Availability.

For information about setting up robust data protection, security, and governance for individual clusters, see Cloudera Security Reference Architecture.

Encryption key backup and recovery

In case of service failure causing keystore corruption, you should regularly back up the keystore. For Ranger KMS with database, you need to back up the underlying keystore database. For Ranger KMS with KTS, you must back up the KTS databases and configuration files. You must also back up client configuration files and keys for KTS clients, such as Navigator Encrypt.

Note: Cloudera recommends regularly backing up Ranger KMS keystore databases and Key Trustee Server databases and configuration files. Because these backups contain encryption keys and encrypted deposits, you must ensure that your backup repository is as secure as the Key Trustee Server.

For information about backing up encryption key, see Backing Up Key Trustee Server and Clients and Restoring Key Trustee Server.

Operations and management

You can configure Hue in a high availability mode. Cloudera Manager does not currently support high availability mode within Cloudera. Therefore, create a Cloudera Manager instance for each side of a disaster recovery cluster pair and periodically back up the configuration details, the Cloudera Manager database, and associated metric data stored on the Cloudera Manager server.

Hue

Hue can be configured in a high availability (HA) mode by using multiple Hue servers and a load balancer in front of the Hue service. Hue configuration and state can be maintained in an external RDBMS which provides its own backup and high availability.

For information about enabling high availability in Hue, see Configuring High Availability for Hue.

Hue utilizes a database for storing internal metadata about users interacting with various services within Cloudera. This database can be backed up using Hue's built-in export functions. This creates a point-in-time backup that can be stored offline.

For information about backing up and restoring Hue data, see Backing up the Hue database.

Hue has no built-in replication capabilities. Periodically restoring a point-in-time backup is sufficient in many cases.

Cloudera Manager

Cloudera recommends creating a Cloudera Manager instance for each side of a disaster recovery cluster pair. While a single Cloudera Manager instance can manage multiple clusters, in the event of a disaster, your Cloudera Manager might be on the wrong side of the incident, leaving you without the ability to manage anything post-failover.

Because of this, keeping cluster configurations in sync between two Cloudera Manager instances can be a challenge. Adopting an appropriate change management and update procedure between the two clusters may be required. The cloudera-playbook Ansible framework can help normalize this by letting you apply a common set of configurations to individual clusters. See the cloudera-playbook details in the Cloudera Labs Github repository for more information about using Ansible.

For more information about the Cloudera Playbook Ansible Framework, see Cloudera-playbook on Cloudera Labs and Automating Cloudera on premises Installations with Ansible.

Cloudera Manager does not currently support high availability mode within Cloudera. Because of this, it is important to periodically back up configuration details, the Cloudera Manager database, and associated metric data stored on the Cloudera Manager server.

For information about Cloudera Manager API, see Backing up the Cloudera Manager Configuration and Restoring the Cloudera Manager Configuration.

RDBMS for Services

In most cases, the RDBMS backing a particular service needs to be treated as a tightly coupled component of the cluster it is attached to. Individual services might maintain cluster-specific information within the databases that make it difficult to safely replicate the databases directly. As an example, the Hive Metastore database might refer to fully qualified HDFS paths that only make sense for one half of the disaster recovery pair. This would require you to use Hive-level replication and export mechanisms to transfer and transform Hive Schema details from one side of the disaster recovery pair to the other.

System Configuration

Operating system and infrastructure level configuration that sit outside the envelope of Cloudera products should be managed through a common configuration management infrastructure such as Ansible. This ensures configuration consistency and enforcement between your disaster recovery pair. It is highly recommended to keep your disaster recovery clusters as closely configured as possible from an architecture and design perspective.

Aligning with business continuity standards

ISO 27031:2011 provides guidance for business continuity and recovery. This standard outlines the policies and procedures that apply during a disaster. The core areas of the standard help define the Plan, Do, Check, and Act stages of the processes and procedures. This has been expanded to include some preliminary discovery and broader areas of consideration.

NIST 800-34 provides guidance on contingency planning for federal organizations and outlines the seven stages to contingency planning with associated samples. For more information, see https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-34r1.pdf.

References and acknowledgements

The document includes several links and references to understand disaster recovery operations.

References

The following list contains a short list of the key references:

- Seven tiers for Disaster Recovery and The tiers of disaster recovery, explained.
- NIST 800-34 in PDF format.
- Cloudera Base on premises Reference Architecture in PDF format.
- Enabling High Availability for HDFS.
- Cloudera Base on premises Replication Manager.

Acknowledgements

Thanks to Travis Campbell and Christopher Royles for help with review, feedback, and creation of this document.