# CLOUDERA

# Cloudera Data Platform Data Center (CDP-DC) Reference Architecture

## Important Notice

# Table of Contents

# Abstract

Cloudera Data Platform (CDP) Data Center is the on-premises version of the Cloudera Data Platform. This new product combines the best of Cloudera Enterprise Data Hub and Hortonworks Data Platform Enterprise along with new features and enhancements across the stack. This unified distribution is a scalable and customizable platform where you can securely run many types of workloads.

CDP Data Center supports a variety of hybrid solutions where compute tasks are separated from data storage and where data can be accessed from remote clusters. This hybrid approach provides a foundation for containerized applications by managing storage, table schema, authentication, authorization, and governance.

CDP Data Center comprises of a variety of components such as Apache HDFS, Apache Hive 3, Apache HBase, and Apache Impala, along with many other components for specialized workloads. You can select any combination of these services to create clusters that address your business requirements and workloads. Several pre-configured packages of services are also available for common workloads. These include:

- Data Engineering
  Ingest, transform, and analyze data.
  Services: HDFS, YARN, YARN Queue Manager, Ranger, Atlas, Hive metastore, Hive on Tez, Spark, Oozie, Hue, and Data Analytics Studio
- Data Mart
  Browse, query, and explore your data in an interactive way.
  Services: HDFS, YARN, YARN Queue Manager, Ranger, Atlas, Hive metastore, Impala, and Hue
- Operational Database
  Low latency writes, reads, and persistent access to data for Online Transactional Processing (OLTP) use cases.
  Services: HDFS, Ranger, Atlas, and HBase

When installing a CDP Data Center cluster, you install a single parcel called Cloudera Runtime that contains all of the components. For a complete list of the included components, see Cloudera Runtime Component Versions.

In addition to the Cloudera Runtime components, the CDP Data Center includes powerful tools to manage, govern, and secure your cluster.
    CDP - Data Center Component Documentation
    Links to documentation for Cloudera Runtime components.

Cloudera Reference Architecture documents illustrate example cluster configurations and certified partner products. The reference architecture documents (RAs) are not replacements for official statements of supportability, rather they provide guidance to assist with deployment and sizing options. Statements regarding supported configurations in the RA are informational and should be cross-referenced with the latest documentation.

This document is a high-level design and best-practices guide for deploying Cloudera Data Platform Data Center, in customer data centers.

# Infrastructure

## System Architecture Best Practices

This section describes Cloudera's recommendations and best practices applicable to Hadoop cluster system architecture.

The following few sections cover the requirements for various infrastructure sub-components.

### Operating Systems

To be covered by Cloudera Support:

- All Runtime hosts in a logical cluster must run on the same major OS release.
- Cloudera supports a temporarily mixed OS configuration during an OS upgrade project.
- Cloudera Manager must run on the same OS release as one of the clusters it manages.

Cloudera recommends running the same minor release on all cluster nodes. However, the risk caused by running different minor OS releases is considered lower than the risk of running different major OS releases.

Points to note:

- Cloudera does not support Runtime cluster deployments in Docker containers.
- Cloudera Enterprise is supported on platforms with Security-Enhanced Linux (SELinux) enabled and in enforcing mode. Cloudera is not responsible for policy support or policy enforcement. If you experience issues with SELinux, contact your OS provider.

See the CDP-DC operating systems requirements page for the latest information for compatibility and special considerations.

### Databases

Cloudera Manager and Runtime come packaged with an embedded PostgreSQL database for use in non-production environments. The embedded PostgreSQL database is not supported in production environments. For production environments, you *must* configure your cluster to use dedicated external databases.

Cloudera recommends that for most purposes you use the default versions of databases that correspond to the operating system of your cluster nodes. See the operating system's documentation to verify support if you choose to use a database other than the default. Enabling backups for these databases should be addressed by the customer's IT department.

Data Analytics Studio requires PostgreSQL version 9.6, while RHEL 7.6 provides PostgreSQL 9.2. Use UTF8 encoding for all custom databases.

Check the CDP-DC database requirements page for the latest information in terms of compatibility and special considerations.

### Java
Only 64 bit JDKs are supported.

Unless specifically excluded, Cloudera supports later updates to a major JDK release from the release that support was introduced. Cloudera excludes or removes support for select Java updates when security is jeopardized.

Running Cloudera Runtime nodes within the same cluster on different JDK releases is not supported. All cluster hosts must use the same JDK update level.

See the CDP-DC Java Requirements page for the latest information in terms of compatibility and special considerations.

> **NOTE:** The Cloudera Data Platform Private Cloud architecture might significantly affect the node and network sizing considerations. This reference architecture is appropriate for aggregated workload clusters running the Cloudera Runtime.

### Right-Size Server Configurations
Cloudera recommends deploying up to four machine types into production:
- **Master Nodes -** Run the Hadoop master daemons such as NameNode, Standby NameNode, YARN Resource Manager and History Server, the HBase Master daemon, Ranger server, Atlas Server, and the Impala StateStore Server and Catalog Server. Master nodes are also the location where Zookeeper and JournalNodes are installed. The daemons can share a single pool of servers. Depending on the cluster size, the roles can be run on a dedicated server. Kudu Master Servers should also be deployed on master nodes.
- **Worker Nodes -** Run the HDFS DataNode, YARN NodeManager, HBase RegionServer, Impala impalad, Search worker daemons and Kudu Tablet Servers.
- **Utility Nodes -** Runs Cloudera Manager and the Cloudera Management Services.  It can also host a MariaDB (or another supported) database instance, which is used by Cloudera Manager, Hive, Ranger and other Hadoop-related projects.
- **Edge Nodes -**  Contain all client-facing configurations and services, including gateway configurations for HDFS, YARN, Impala, Hive, and HBase. The edge node is also a good place for Hue, Oozie, HiveServer2, and Impala HAProxy. HiveServer2 and Impala HAProxy serve as a gateway to external applications such as the business intelligence (BI) tools. Edge nodes are also known as Gateway nodes.

For more information see Runtime Cluster Hosts and Role Assignments.

## Deployment Topology

The graphic below shows a cluster deployed across several racks (Rack 1, Rack 2, … Rack *n*).
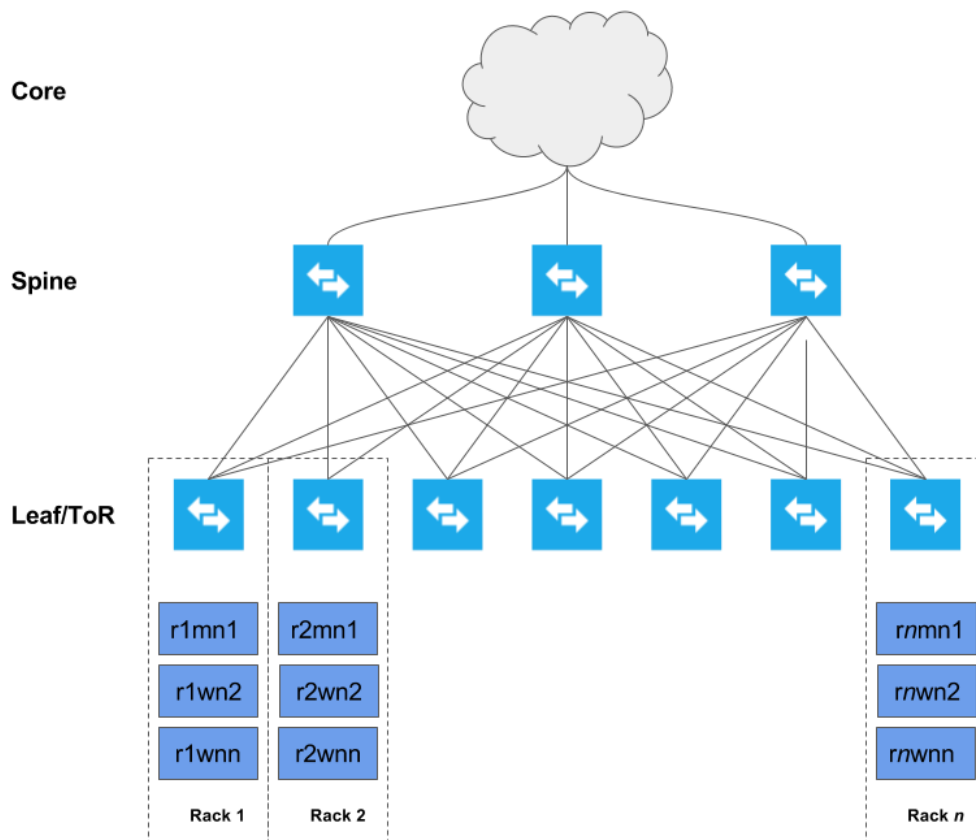


Each host is connected to two top-of-rack (TOR) switches which are in turn connected to a collection of spine switches which are then connected to the enterprise network. This deployment model allows each host to maximum throughput, minimize latency, while encouraging scalability. The specifics of the network topology are described in the subsequent sections.

**NOTE:** In the graphic above, the nomenclature used is meant to represent the rack number, the role of a node in the cluster and its ordinality in the rack. For example: r1mn1 would represent Rack1, Master Node 1, and so on. Every rack need not have a master node. It is a good practice to spread all master nodes of a cluster across different racks to avoid single point of failure.

Gateway and utility nodes also reside within these racks (not annotated in the diagram).

## Physical Component List

The table describes the physical components recommended to deploy a Cloudera Runtime cluster.

| Component | Configuration | Description | Quantity |
|---|---|---|---|
| Physical servers | Two-socket, 8-14 cores per socket > 2 GHz; minimum 128 GB RAM. | Host the various cluster components. | Based on cluster design. |
| NICs | 10 Gbps Ethernet NICs (minimum required) | Provide the data network services for the cluster. | At least one per server, although two NICs can be bonded for additional throughput.[1] |
| Internal HDDs | 500 GB HDD or SSD recommended for operating system and logs; HDD for data disks (size varies with data volume requirements) | Ensure continuity of service on server resets and contain the cluster data. | 10-24 disks per physical server. The largest storage density currently supported is 100 TB per DataNode. |
| Ethernet ToR/leaf switches | Minimum 10 Gbps switches with sufficient port density to accommodate the cluster. These require enough ports to create a realistic spine-leaf topology providing ISL bandwidth above a 1:4 oversubscription ratio (preferably 1:1). | Although most enterprises have mature data network practices, consider building a dedicated data network for the Hadoop cluster. | At least two per rack. |
| Ethernet spine switches | Minimum 40 Gbps switches with sufficient port density to accommodate incoming ISL links and ensure required throughput over the spine (for inter-rack traffic). | Same considerations as for ToR switches. | Depends on the number of racks. |

---

[1] Cloudera does not support multi-homing. Multiple network adapters can be used, but they must be aggregated to provide a single host IP per node.

## Network Specification

### *Dedicated Network Hardware*

Hadoop can consume all available network bandwidth. For this reason, Cloudera recommends that Hadoop be placed in a separate physical network with its own core switches.

### *Switch Per Rack*

Hadoop supports the concept of rack locality and takes advantage of the network topology to minimize network congestion. Ideally, nodes in one rack should connect to a single physical switch. Two top- of-rack (ToR) switches can be used for high availability. Each rack switch (ToR switch) uplinks to a core switch with a significantly bigger backplane. Cloudera recommends 10 GbE (or faster) connections between the servers and ToR switches. ToR uplink bandwidth to the core switch (two switches in a HA configuration) will often be oversubscribed.

### *Uplink Oversubscription*

How much oversubscription is appropriate depends on the workload. Cloudera's recommendation is that the ratio between the total access port bandwidth and uplink bandwidth be as close to 1:1 as is possible. This is especially important for heavy ETL workloads, and MapReduce jobs that have a lot of data sent to reducers.

Oversubscription ratios up to 4:1 are generally fine for balanced workloads, but network monitoring is needed to ensure uplink bandwidth is not the bottleneck for Hadoop. The following table provides some examples as a point of reference:

| Access Port Bandwidth (In Use) | Uplink Port Bandwidth (Bonded) | Ratio |
|---|---|---|
| 48 x 1 GbE = 48 Gbit/s | 4 x 10 GbE = 40 Gbit/s | 1.2:1 |
| 24 x 10 GbE = 240 Gbit/s | 2 x 40 Gig CFP = 80 Gbit/s | 3:1 |
| 48 x 10 GbE = 480 Gbit/s | 4 x 40 Gig CFP = 160 Gbit/s | 3:1 |

> **Important:**
>
> Do not exceed 4:1 oversubscription ratio. For example, if a ToR has 20 x 10 GbE ports used, the uplink should be <u>at least</u> 50 Gbps. Different switches have dedicated uplink ports of specific bandwidth (often 40 Gbps or 100 Gbps) and therefore careful planning needs to be done to choose the right switch types.

### *Redundant Network Switches*

Having redundant core switches in a full mesh configuration allow the cluster to continue operating in the event of a core switch failure. Redundant ToR switches prevent the loss of an entire rack of processing and storage capacity in the event of a ToR switch failure. General cluster availability can still be maintained in the event of the loss of a rack, as long as master nodes are distributed across multiple racks.

### *Accessibility*

The accessibility of your Cloudera Enterprise cluster is defined by the network configuration and depends on the security requirements and the workload. Typically, there are edge/client nodes that have direct

access to the cluster. Users go through these edge nodes through the client applications to interact with the cluster and the data residing there. These edge nodes could be running a web application for real-time serving workloads, BI tools, or simply the Hadoop command-line client used to submit or interact with HDFS.

Cloudera recommends allowing access to the Cloudera Enterprise cluster through edge nodes only. You can configure this in the security groups for the hosts that you provision. The rest of this document describes the various options in detail.

### *Internet Connectivity*

Clusters that do not require heavy data transfer between the internet or services outside of the immediate network and HDFS might need access to services like software repositories for updates or other low-volume outside data sources.

Customers who intend to leverage the multi-cloud/hybrid-cloud functionality in CDP, must ensure adequate network bandwidth be present between their data centers and the public cloud vendors' networks. Details on this topic are out of scope of this document. Engage with your Cloud vendor's technical sales team and Cloudera Sales engineering team to determine the requirements in such scenarios.

If you completely disconnect the cluster from the Internet, you block access for software updates which makes maintenance difficult.

### Cloudera Manager

Cloudera strongly recommends installing Cloudera Runtime using the Cloudera Manager (CM). During the Cloudera Runtime installation using the CM there is the choice to install using parcels or native packages. A parcel is a binary distribution format. Parcels offer a number of benefits including consistency, flexible installation location, installation without sudo, reduced upgrade downtime, rolling upgrades, and easy downgrades. Cloudera recommends using parcels, though using packages is supported.

## Cluster Sizing Best Practices

Each worker node typically has several physical disks dedicated to raw storage for Hadoop. This number is used to calculate the total available storage for each cluster. Also, the calculations listed below assume 10% disk space allocated for YARN temporary storage. Cloudera recommends allocating 10-25% of the raw disk space for temporary storage as a general guideline. This can be changed within Cloudera Manager and should be adjusted after analyzing production workloads. For example, MapReduce jobs that send less data to reducers allow for adjusting this number percentage down considerably.

The following table contains example calculations for a cluster that contains 17 worker nodes. Each server has twelve 3 TB drives available for use by Hadoop. The table below outlines the Hadoop storage available based upon the number of worker nodes:

### Default Replication Factor

| | |
|---|---|
| Raw Storage | 612 TB |
| HDFS Storage (configurable) | 550.8 TB |
| HDFS Unique Storage (default replication factor) | 183.6 TB |
| MapReduce Intermediate Storage (configurable) | 61.2 TB |

### Erasure Coding RS-6-3

| Raw Storage | 612 TB |
|---|---|
| HDFS Storage (configurable) | 550.8 TB |
| HDFS Unique Storage (EC RS-6-3 -- 1.5x overhead) | 367.2 TB |
| MapReduce Intermediate Storage (configurable) | 61.2 TB |

### Erasure Coding RS-10-4

| Raw Storage | 612 TB |
|---|---|
| HDFS Storage (configurable) | 550.8 TB |
| HDFS Unique Storage (EC RS-10-4 -- 1.4x overhead) | 393.4 TB |
| MapReduce Intermediate Storage (configurable) | 61.2 TB |

> **Note:**
>
> HDFS Unique Storage varies depending on the amount of data stored in EC directories and the RS policies. The tables above are examples of how different policies can affect HDFS Unique Storage.

> **Note:**
>
> Compressing raw data can effectively increase HDFS storage capacity.

While Cloudera Manager provides tools such as Static Resource Pools, which utilize Linux Cgroups, to allow multiple components to share hardware, in high volume production clusters it can be beneficial to allocate dedicated hosts for roles such as Solr, HBase, and Kafka.

## Cluster Hardware Selection Best Practices

This section gives a high-level overview of how different hardware component selections impact the performance of a Hadoop cluster.

See the Hardware Requirements Guide for detailed workload-specific practices.

### Number of Drives

Traditionally Hadoop has been thought of as a large I/O platform.  While there are many new types of workloads being run on Cloudera clusters that may not be as I/O bound as traditional MapReduce applications, it is still useful to consider the I/O performance when designing a Cloudera cluster.

Unlike the number of cores in a CPU and the density of RAM, the speed at which data can be read from a spinning hard drive (spindle) has not changed much in the last 10 years[2]. To counter the limited

---

[2] SSDs have dramatically changed the persistent storage performance landscape, but the price per GB of spinning disks is still significantly less than that of SSD storage. As SSDs come down in cost and

performance of hard drive read/write operations, Hadoop reads and writes from many drives in parallel. Every additional spindle added to a node increases the overall read/write speed of the cluster.

Additional drives also come with the likelihood of more network traffic in the cluster. For the majority of cases, network traffic between nodes is generally limited by how fast data can be written to or read from a node. Therefore, the rule normally follows that with more drives, network speed requirements increase.

Generally speaking, the more drives a node has, the lower the cost per TB. However, the larger the quantity of data stored on one node, the longer will the re-replication time be, if that node goes down.  Hadoop clusters are designed to have many nodes. It is generally better to have more average nodes than fewer super nodes. This has a lot to do with both data protection, as well as increased parallelism for distributed computation engines such as MapReduce and Spark.

Lastly, the number of drives per node impacts the number of YARN containers configured for a node. YARN configuration and performance tuning is a complicated topic, but for I/O bound applications, the number of physical drives per host may be a limiting factor in determining the number of container slots configured per node.

Kafka clusters are often run on dedicated servers that do not run HDFS data nodes or processing components such as YARN and Impala. Because Kafka is a message-based system, fast storage and network I/O is critical to performance. Although Kafka does persist messages to disk, it is not generally necessary to store the entire contents of a Kafka topic log on the Kafka cluster indefinitely. Kafka brokers should be configured with dedicated spinning hard drives for the log data directories. Using SSDs instead of spinning disks has not shown to provide a significant performance improvement for Kafka.

Kafka drives should also be configured as RAID 10 because the loss of a single drive on a Kafka broker will cause the broker to experience an outage.

### Disk Layout
For Master nodes, the following layout is recommended:
- 2 x Disks (capacity at least 500 GB) in RAID 1 (software or hardware) for OS and logs
- 4 x Disks (>= 1 TB each) in RAID 10 for Database data (see Note)
- 2 x Disks (capacity at least 1 TB) in RAID 1 (software or hardware) for NameNode metadata
- 1 x Disk JBOD/RAID 0 for ZooKeeper (>= 1TB) (see Note)
- 1 x Disk JBOD/RAID 0 for Quorum JournalNode (>= 1 TB)

> **Note:**
> Ideally databases should be run on an external host rather than running on the master node(s).

> **Note:**

technologies such as Intel's Optane™ enter the market, workloads may swing back towards being CPU bound. Most Cloudera customers are still deploying clusters that store data on spinning hard disks.

> If a customer has experienced fsync delays and other I/O related issues with ZooKeeper, ZooKeeper's `dataDir` and `dataLogDir` can be configured to use separate disks. It is hard to determine ahead of time whether this will be necessary; even a small cluster can result in heavy ZooKeeper activity.

For Worker nodes, the following layout is recommended:
- 2x Disks (capacity at least 500 GB) in RAID 1 (software or hardware) for OS and logs
- 12-24 SATA Disks JBOD mode (or as multiple single-drive RAID 0 volumes if using a RAID controller incapable of doing JBOD passthrough) no larger than 4 TB in capacity. If the RAID controller has cache, use it for write caching (preferably with battery backup) and disable read caching. Follow your hardware vendor's best practices where available.
- For a higher performance profile, use 10K RPM SATA or faster SAS drives.These often have lower capacity, but capacity considerations can be offset by adding more data nodes.

RAID controllers should be configured to disable any optimization settings for the RAID 0 volumes[3].

## Data Density Per Drive

Hard drives today come in many sizes. Popular drive sizes are 1-4 TB, although larger drives are becoming more common. When picking a drive size the following points need to be considered.

- **Lower Cost Per TB** – The larger the drive, the cheaper the cost per TB, which makes for lower TCO.
- **Replication Storms** – Larger drives means drive failures will produce larger re-replication storms, which can take longer and saturate the network while impacting in-flight workloads.
- **Cluster Performance** – In general, drive size has little impact on cluster performance.  The exception is when drives have different read/write speeds and a use case that leverages this gain. MapReduce is designed for long sequential reads and writes, so latency timings are generally not as important.  HBase can potentially benefit from faster drives, but that is dependent on a variety of factors, such as HBase access patterns and schema design; this also implies acquisition of more nodes.  Impala and Cloudera Search workloads can also potentially benefit from faster drives, but for those applications the ideal architecture is to maintain as much data in memory as possible.

Cloudera does not support more than 100 TB per data node. You could use 12 x 8 TB spindles or 24 x 4 TB spindles. Cloudera does not support drives larger than 8 TB.[4]

> **Warning:**
>
> Running CDP DC on storage platforms other than direct-attached physical disks can provide suboptimal performance. Cloudera Runtime and the majority of the Hadoop platform are optimized to provide high performance by distributing work across a cluster that can utilize data locality and fast

---

[3] DO NOT create multi-disk RAID 0 volumes.

[4] Larger disks offer increased capacity but not increased I/O throughput. Clusters with larger disks can easily result in capacities exceeding 100 TB per-worker, contributing to replication storms mentioned above. Clusters with larger disks that observe the 100 TB limit end up having fewer spindles which reduces HDFS throughput.

local I/O. See the [Cloudera Enterprise Storage Device Acceptance Criteria Guide](#) for more information about using non-local storage.

## Number of Cores and Multithreading

Other than cost, there is no negative for buying more and better CPUs. However, the ROI on additional CPU power must be evaluated carefully[5]. Here are some points to consider:

- **Cluster Bottleneck** – In general, CPU resources (and lack thereof) do not bottleneck MapReduce and HBase. The bottleneck almost always is drive and/or network performance. There are certainly exceptions to this, such as inefficient Hive queries. Other compute frameworks like Impala, Spark, and Cloudera Search may be CPU-bound depending on the workload.
- **Additional Cores/Threads** – Within a given MapReduce job, a single task typically uses one thread at a time. With Spark this is different, as a single task might use multiple threads in parallel. As outlined earlier, the number of slots allocated per node may be a function of the number of drives in the node[6]. As long as there is no huge disparity in the number of cores (threads) and the number of drives, there is no need  for additional cores. In addition, a MapReduce task is going to be I/O bound for typical jobs. Thus, a given thread used by the task will have a large amount of idle time while waiting for an I/O response.
- **Clock Speed** – Because Cloudera clusters often begin with a small number of use cases and associated workloads and grow over time, it makes sense to purchase the fastest CPUs available. Actual CPU usage is use case and workload dependent.For instance, computationally intensive Spark jobs benefit more from faster CPUs than I/O bound MapReduce applications.

> **Important:**
>
> Allocate two vCPUs for the operating system and other non-Hadoop use (although this amount may need to be higher if additional non-Hadoop applications are running on the cluster nodes, such as third-party active monitoring/alerting tools). The more services you are running, the more vCPUs will be required; you will need to use more capable hosts to accommodate these needs..

For worker nodes, a mid-range 12-14 core CPU running at 2.4-2.5 GHz provides a good cost/performance tradeoff. For master nodes, a mid-range 8 core CPU with a faster clock speed (For example 2.6 GHz) would suffice. Where available, Simultaneous Multi-Threading implementations should be enabled (for example Intel's HyperThreading). BIOS settings for CPU and memory should be set to Maximum Performance mode or equivalent.

See the [Hardware Requirements Guide](#) for detailed workload-specific practices.

## RAM

More memory is always good and it is recommended to purchase as much as the budget allows. Applications such as Impala and Cloudera Search are often configured to use large amounts of heap, and

---

[5] It might help to consider the [latest cloudera licensing model](#) as different models might fit for different use-cases.

[6] A more careful evaluation is required in scenarios where the [Virtual Private Cluster](#) model is being considered. In such scenarios, if storage and compute are clearly distinct roles, then requirements might vary significantly.

a mixed workload cluster supporting both services should have sufficient RAM to allow all required services.

See the Hardware Requirements Guide for detailed workload-specific practices.

> **Important:**
>
> Allocate at least 4 GB memory for the operating system and other non-Hadoop use (although this amount may need to be higher if additional non-Hadoop applications are running on the cluster nodes, such as third-party active monitoring/alerting tools). The more services you are running, the more memory will be required; you will need to use more capable hosts to accommodate these needs.

It is critical to performance that the total memory allocated to all Hadoop-related processes (including processes such as HBase) is less than the total memory on the node, taking into account the operating system and non-Hadoop processes. Oversubscribing the memory on a system can lead to the Linux kernel's out-of-memory process killer being invoked and important processes being terminated. Performance might be affected by over-allocation of memory to a Hadoop process, as this can lead to long Java garbage collection pauses. For processes such as HBase, factor in aspects such as off heap bucket cache configuration.

For optimum performance, confer with your hardware vendor for defining optimal memory configuration layout.

Whilst 128 GB RAM can be accommodated, this typically constrains the amount of memory allocated to services such as YARN and Impala, thereby reducing the query capacity of the cluster. 256 GB RAM is recommended with higher values also possible.

### Power Supplies

Hadoop software is designed around the expectation that nodes will fail. Redundant hot-swap power supplies are not necessary for worker nodes, but should be used for master, utility, and edge nodes.

If using single power supplies on worker nodes, confer with your data center team to alternate the power feeds for each rack. For example, Rack 1 goes on Feed A, Rack 2 goes on Feed B, Rack 3 goes on Feed C, repeating the pattern.

## Operating System Best Practices

Cloudera currently supports running the platform only on Linux distributions with RHEL/CentOS/OL with RHCK kernel, with the release version being RHEL 7.6 or higher (or equivalent). To receive support from Cloudera, a supported version of the operating system must be in use. The Requirements and Supported Versions guide lists the supported operating systems for each version of Cloudera Manager and CDP Runtime.

The CDP DC documentation provides a planning page. Review this information prior to starting an installation.
See the CDP DC Runtime cluster hosts and role assignments page for service layout guidelines.

## Hostname Naming Convention

Cloudera recommends using a hostname convention that allows for easy recognition of roles and/or physical connectivity. This is especially important for configuring rack awareness within Cloudera Manager. Using a project name identifier, followed by the rack ID, the machine class, and a machine ID is an easy way to encode useful information about the cluster. For example:

```
acme-test-r01m01
```

This hostname represents the ACME customer's test project, rack #1, and master node #1.

## Hostname Resolution

Cloudera recommends using DNS for hostname resolution. The usage of `/etc/hosts` becomes cumbersome quickly, and routinely is the source of hard to diagnose problems. `/etc/hosts` should contain an entry for 127.0.0.1, and `localhost` should be the only name that resolves to it. The file should also contain an entry for the system's IP address, FQDN, and shortname. The machine name must not resolve to the 127.0.0.1 address. All hosts in the cluster must have forward and reverse lookups to be the inverse of each other. An easy test to perform on the hosts to ensure proper DNS resolution is to execute:

```
dig <hostname>
dig -x <ip_address_returned_from_hostname_lookup)
```

For example:

```
dig themis.apache.org
themis.apache.org.    1758  IN   A     140.211.11.105

dig -x 140.211.11.105
105.11.211.140.in-addr.arpa. 3513 IN      PTR   themis.apache.org.
```

This is the acceptable behavior for every host in the cluster.

Also enable nscd with only host name caching enabled for a 30-60 second period. This further reduces heavy DNS impact. This is a mitigation technique for preventing the overload of AD DNS, for example, which can fall over during high DNS load.

```
enable-cache passwd no
enable-cache group no
enable-cache hosts yes
positive-time-to-live hosts 60
negative-time-to-live hosts 20
```

### Functional Accounts

Cloudera Manager and CDH use dedicated functional accounts for the associated daemon processes. By default, these accounts are created as local accounts on every machine in the cluster that needs them if they do not already exist (locally or from a directory service, such as LDAP). The CDP-DC Installation guide includes a table showing the UNIX user and group associated with each service.

> **Note:**
>
> Kerberos deployment models (including identity integration with Active Directory) are covered in detail within the Authentication documentation.

### Time

All machines in the cluster need to have the same time and date settings, including time zones. Use of the Network Time Protocol (NTP) is recommended. Many cluster services are sensitive to time (For example HBase, Kudu, ZooKeeper) and troubleshooting is easier if time is consistent across all hosts. Doing the following enables the NTP daemon:

(RHEL/CentOS 7)

```
systemctl start ntpd.service
systemctl enable ntpd.service
```

> **Note:**
>
> Chrony may be preferred on newer operating systems.

### Name Service Caching

It is recommended that name service caching be enabled, particularly for clusters that use non-local Hadoop functional accounts, such as the `hdfs` and `yarn` users.  This becomes critical in the case where the latter is combined with using Kerberos. Many difficult-to-diagnose problems can arise when name service lookups time out or fail during heavy cluster utilization. Doing the following enables the Name Service Cache Daemon (nscd):

(RHEL/CentOS 7)

```
systemctl start nscd.service
systemctl enable nscd.service
```

If you are running Red Hat SSSD you must modify the nscd configuration to not cache the `passwd`, `group`, or `netgroup` information.

### SELinux

Cloudera Enterprise (without Cloudera Navigator) is supported on platforms with Security-Enhanced Linux (SELinux) enabled.However Cloudera recommends SELinux be disabled on all machines in the Hadoop cluster until the cluster is up and running.

The Linux command `getenforce` returns the status of SELinux.

SELinux can be disabled on RHEL/CentOS by editing `/etc/selinux/config` and setting `SELINUX=disabled`. This change must be done as root (or with proper sudo access), and requires a reboot.

### IPv6

Hadoop does not support IPv6. IPv6 configurations should be removed, and IPv6-related services should be stopped.

### iptables

Cloudera recommends disabling host-based firewalls on the cluster, until the cluster is up and running. Many problems that are difficult to diagnose result from incorrect/conflicting iptables entries that interfere with normal cluster communication. Doing the following disables iptables for both IPv4 and IPv6:

(RHEL/CentOS 7)

```
systemctl stop firewalld.service
systemctl disable firewalld.service
```

For those who must restrict access using host-based firewalls, see the list of ports used by Cloudera Manager, Cloudera Runtime components, managed services, and third-party components.

### Startup Services

As with any production server, unused services should be removed and/or disabled. Some example services that are enabled by default and not needed by the CDH are:

- bluetooth
- cups
- iptables
- ip6tables
- postfix[7]

This list is not exhaustive. To view the list of services that are configured to start during system startup, execute the following command:

(RHEL/CentOS 7)

---

[7] While not needed by CDH, postfix (or other MTA) may be required by other services to deliver generated notices/alerts from the system.

```
systemctl list-unit-files --type service | grep enabled
```

### Process Memory

The memory on each node is allocated out to the various Hadoop processes. This predictability reduces the chance of Hadoop processes inadvertently running out of memory and paging to disk, which in turn leads to severe degradation in performance. See the section on *Kernel and OS Tuning* for further information.

A minimum of 4 GB of memory should be reserved on all nodes for operating system and other non-Hadoop use. This amount may need to be higher if additional non-Hadoop applications are running on the cluster nodes, such as third-party active monitoring/alerting tools.

Memory requirements and allocation for Hadoop components are discussed in further detail in other sections of this document.

### Kernel and OS Tuning

The Cloudera Runtime platform depends on a tuned underlying host operating system for optimal performance. Cloudera suggests setting the vm.swappiness and transparent hugepage compaction kernel parameters. The Cloudera Administration manual has additional background information and suggested settings: Optimizing Performance in Cloudera Runtime.

#### *Entropy*

Cryptographic operations require entropy to ensure randomness. The Cloudera Security guide explains how to check available entropy and how to ensure sufficient entropy is available: Entropy Requirements.

#### *Networking Parameters*

The Transmit and Receive ring buffer sizes on the ethernet interfaces of every node of the cluster should be adjusted to ensure higher throughput rates --

Check existing ring buffer sizes by running the following command:

```
$ ethtool -g eth0
Ring parameters for eth0:
Pre-set maximums:
RX:      4096
RX Mini: 0
RX Jumbo:    0
TX:      4096
Current hardware settings:
RX:      256
RX Mini: 0
RX Jumbo:    0
TX:       256
```

After checking the preset maximum values and the current hardware settings, use the following commands to resize the ring buffers:

```
# ethtool –G <interface> rx <newsize>
```

- or -

```
# ethtool –G <interface> tx <newsize>
```

Most modern enterprise-grade network adapters have several performance optimization features, such as offload capabilities and large segmentation offload, which reduce load on host CPU by handling these functions at the network interface level which can be explored as part of performance optimization initiatives.  An iterative approach is recommended -- while applying a standard load generator such as the [Terasuite benchmarks](#),  test the effect of enabling said features. Performance optimization parameters should not be applied indiscriminately, without thorough testing, and should be applied only based on genuine need.

The following parameters are to be added[8] to `/etc/sysctl.conf` to optimize various network behaviors:

- Disable TCP timestamps to improve CPU utilization (this is an optional parameter and will depend on your NIC vendor):

  ```
  net.ipv4.tcp_timestamps=0
  ```

- Enable TCP sacks to improve throughput:

  ```
  net.ipv4.tcp_sack=1
  ```

- Increase the maximum length of processor input queues:

  ```
  net.core.netdev_max_backlog=250000
  ```

- Increase the TCP max and default buffer sizes using `setsockopt()`:

  ```
  net.core.rmem_max=4194304
  net.core.wmem_max=4194304
  net.core.rmem_default=4194304
  net.core.wmem_default=4194304
  net.core.optmem_max=4194304
  ```

- Increase memory thresholds to prevent packet dropping:

  ```
  net.ipv4.tcp_rmem="4096 87380 4194304"
  net.ipv4.tcp_wmem="4096 65536 4194304"
  ```

---

[8] Note that performance tuning guidelines provided in this document are meant to be applied in an iterative manner, along with sufficient testing. Not all parameters specified may be applicable. These are general best practices but details may vary based on infrastructure being used as well as application workload patterns. When in doubt, see the vendor documentation relevant for your equipment.

## Swappiness

The kernel parameter that determines the tendency of the kernel to swap pages in and out of the main memory is vm.swappiness. By default, on RHEL 7 and variants, the value is set to 60 (range 0-99). This is not appropriate for hadoop clusters. This needs to be tuned down to 1 to avoid unnecessary page outs and long garbage collection pauses.

```
sudo sysctl -w vm.swappiness=1
```

To make this parameter persist across reboots, an entry needs to be made in /etc/sysctl.conf file as follows --

```
vm.swappiness=1
```

## Filesystems

In Linux there are several choices for formatting and organizing drives. That being said, only a few choices are optimal for Hadoop.

In RHEL/CentOS, the Logical Volume Manager (LVM) should not be used for data drives. It is not optimal and can lead to combining multiple drives into one logical disk, which is in complete contrast to how Hadoop manages fault tolerance across HDFS. It is beneficial to keep LVM enabled on the OS drives. Any performance impact that may occur is countered by the improvement to manageability of the system. Using LVM on the OS drives enables the admin to avoid over-allocating space on partitions. Space needs can change over time and the ability to dynamically grow a file system is better than having to rebuild a system. Do not use LVM to stripe or span logical volumes across multiple physical volumes to mimic RAID.

Cloudera recommends using an extent-based file system. This includes `ext3`, `ext4`, and `xfs`. Most new Hadoop clusters use the `ext4` filesystem by default. Red Hat Enterprise Linux 7 uses `xfs` as its default filesystem.

> **Important:**
>
> If using Kudu ensure that filesystem hole punching is a capability of the filesystem. Hole punching is the use of the `fallocate()` system call with the `FALLOC_FL_PUNCH_HOLE` option set. Newer versions of `ext4` and `xfs`  support Hole Punching. `ext3`  does not support hole punching and unpatched RHEL prior to 6.4 does not support this facility. Older versions of `ext4` and `xfs` that do not support hole punching cause Kudu to fail to start as Kudu provides a pre-start test for this facility. Without the hole punching support, the block manager is unsafe to use as claimed blocks will never be released and more disk space will be consumed.

### *Filesystem Creation Options*

When creating `ext4` filesystems for use with Hadoop **data volumes**, Cloudera recommends reducing the superuser block reservation from 5% to 1% for root (using the `-m1` option) as well as setting the following options:

- use one inode per 1 MB (largefile)
- minimize the number of super block backups (sparse_super)
- enable journaling (has_journal)
- use b-tree indexes for directory trees (dir_index)
- use extent-based allocations (extent)

Creating such an `ext4` filesystem might look like this:

```
mkfs -t ext4 -m 1 -O sparse_super,dir_index,extent,has_journal
/dev/sdb1
```

Creating an `xfs` filesystem might look like this:

```
mkfs -t xfs /dev/sdb1
```

> **Note:**
>
> When creating `xfs` filesystems, no options are required.

### *Disk Mount Options*

The HDFS by design is a fault-tolerant file system. All drives used by DataNode machines for data need to be mounted without the use of RAID Drives should be mounted in `/etc/fstab` using the `noatime` option (which also implies `nodiratime`). In case of SSD or flash turn on [TRIM](#) by specifying the `discard` option when mounting, this reduces premature SSD wear and device failures, while primarily avoiding long garbage collection pauses.

In `/etc/fstab` ensure that the appropriate file systems have the `noatime` mount option specified:

```
/dev/sda1    /            ext4     noatime               0 0
```

To enable TRIM, edit `/etc/fstab` and set mount option discard as well.

```
/dev/sdb1    /data        ext4     noatime,discard       0 0
```

### *Disk Mount Naming Convention*

For ease of administration, it is recommended to mount all of the disks on the DataNode machines with a naming pattern, for example:

```
/data1
/data2
/data3
/data4
/data5
/data6
```

# Cluster Configuration

This section contains information about how the cluster is configured, including Cloudera recommendations specific to the customer hardware being used, best practices, and general recommendations for each Cloudera Runtime service. This section is not an exhaustive description of every configuration, rather focuses on important configurations and those that have been changed from the default setting.

## Teragen and Terasort Performance Baseline

The `teragen` and `terasort` benchmarking tools are part of the standard Apache Hadoop distribution and are included with the Cloudera distribution. In the course of a cluster installation or certification, Cloudera recommends running several `teragen` and `terasort` jobs to obtain a performance baseline for the cluster. The intention is not to demonstrate the maximum performance possible for the hardware or to compare with externally published results, as tuning the cluster for this may be at odds with actual customer operational workloads. Rather the intention is to run a real workload through YARN to functionally test the cluster as well as obtain baseline numbers that can be used for future comparison, such as in evaluating the performance overhead of enabling encryption features or in evaluating whether operational workloads are performance limited by the I/O hardware. Running the benchmarks provides an indication of cluster performance and may also identify and help diagnose hardware or software configuration problems by isolating hardware components, such as disks and network, and subjecting them to a higher than normal load.

The `teragen` job generates an arbitrary amount of data, formatted as 100-byte records of random data, and stores the result in HDFS.  Each record has a random key and value.  The `terasort` job sorts the data generated by `teragen` and writes the output to HDFS.

During the first iteration of the `teragen` job, the goal is to obtain a performance baseline on the disk I/O subsystem. The HDFS replication factor should be overridden from the default value 3 and set to 1 so that the data generated by `teragen` is not replicated to additional data nodes. Replicating the data over the network would obscure the raw disk performance with potential network bandwidth constraints.

Once the first `teragen` job has been run, a second iteration should be run with the HDFS replication factor set to the default value. This applies a high load on the network, and deltas between the first run and second run can provide an indication of network bottlenecks in the cluster.

While the `teragen` application can generate any amount of data, 1 TB is standard. For larger clusters, it may be useful to also run 10 TB or even 100 TB, as the time to write 1 TB may be negligible compared to the startup overhead of the YARN job. Another `teragen` job should be run to generate a dataset that is 3 times the RAM size of the entire cluster.  This ensures you are not seeing page cache effects and are exercising the disk I/O subsystem.

The number of mappers for the `teragen` and `terasort` jobs should be set to the maximum number of disks in the cluster. This is  less than the total number of YARN `vcores` available, so it is advisable to temporarily lower the `vcores` available per YARN worker node to the number of disk spindles to ensure an even distribution of the workload. An additional `vcore` is needed for the YARN ApplicationMaster.

The `terasort` job should also be run with the HDFS replication factor set to 1 as well as with the default replication factor. The `terasort` job hardcodes a DFS replication factor of 1, but it can be overridden or set explicitly by specifying the *mapreduce.terasort.output.replication* parameter as shown below.

### Teragen and Terasort Command Examples

### *Teragen Command to Generate 1 TB of Data With HDFS Replication Set to 1*

```
EXAMPLES_PATH=/opt/cloudera/parcels/CDH/lib/hadoop-mapreduce

yarn jar ${EXAMPLES_PATH}/hadoop-mapreduce-examples.jar \
  teragen -Ddfs.replication=1 -Dmapreduce.job.maps=360 \
  10000000000 TS_input1
```

The above command generates 1 TB of data with an HDFS replication factor of 1, using 360 mappers. This command is appropriate for a cluster with 360 disks.

### *Teragen Command to Generate 1 TB of Data With HDFS Default Replication*

```
EXAMPLES_PATH=/opt/cloudera/parcels/CDH/lib/hadoop-mapreduce

yarn jar ${EXAMPLES_PATH}/hadoop-mapreduce-examples.jar \
  teragen -Dmapreduce.job.maps=360 \
  10000000000 TS_input2
```

The above command generates 1 TB of data with the default HDFS replication factor (usually 3), using 360 mappers. This command is appropriate for a cluster with 360 disks.

### *Terasort Command to Sort Data With HDFS Replication Set to 1*

```
EXAMPLES_PATH=/opt/cloudera/parcels/CDH/lib/hadoop-mapreduce

yarn jar ${EXAMPLES_PATH}/hadoop-mapreduce-examples.jar \
  terasort -Ddfs.replication=1 \
  -Dmapreduce.job.maps=360 \
  TS_input1 TS_output1
```

The above command sorts the data generated by `terasort` using 360 mappers and writes the sorted output to HDFS with a replication factor of 1. This is appropriate for a cluster with 360 disks.

***Terasort Command to Sort Data With HDFS Replication Set to 3***

```
EXAMPLES_PATH=/opt/cloudera/parcels/CDH/lib/hadoop-mapreduce

yarn jar ${EXAMPLES_PATH}/hadoop-mapreduce-examples.jar \
  terasort -Dmapreduce.job.maps=360 -Ddfs.replication=3 \
  TS_input2 TS_output2
```

The above command sorts the data generated by `terasort` using 360 mappers and writes the sorted output to HDFS with a replication factor of 3 (a typical default). This would be appropriate for a cluster with 360 disks.

***Teragen and Terasort Results***

| Command | HDFS Replication | Number of Mappers | Run Time |
|---|---|---|---|
| Teragen for 1 TB data set | 1 | | |
| Teragen for 3x cluster RAM data set | 1 | | |
| Terasort for 1 TB data set | 1 | | |
| Terasort for 3x cluster RAM data set | 1 | | |
| Teragen for 1 TB data set | 3 | | |
| Teragen for 3x cluster RAM data set | 3 | | |
| Terasort for 1 TB data set | 3 | | |
| Terasort for 3x cluster RAM data set | 3 | | |

## Cluster Configuration Best Practices

### ZooKeeper

ZooKeeper is sensitive to disk latency. While it only uses a modest amount of resources, having ZooKeeper swap out or wait for a disk operation can result in that ZooKeeper node being considered 'dead' by its quorum peers. For this reason, Cloudera recommends against deploying ZooKeeper on worker nodes where loads are unpredictable and are prone to spikes. It is acceptable to deploy Zookeeper on master nodes where load is more uniform and predictable (or on any node where it can have unobstructed access to disk).

### HDFS

***Java Heap Sizes***

The NameNode memory should be increased over time as HDFS has more files and blocks stored. Cloudera Manager can monitor and alert on memory usage. A rough estimate is that the NameNode needs 1 GB of memory for every 1 million files. Setting the heap size too large when it is not needed leads to inefficient Java garbage collection, which can lead to erratic behavior that is hard to diagnose. NameNode and Standby NameNode heap sizes must always be the same, and must be adjusted together.

### NameNode Metadata Locations

When a quorum-based high availability HDFS configuration is used, JournalNodes handle the storage of metadata writes. The NameNode daemons require a local location to store metadata. Cloudera recommends that only a single directory be used if the underlying disks are configured as RAID, or two directories on different disks if the disks are mounted as JBOD.

### Block Size

HDFS stores files in blocks that are distributed over the cluster. A block is typically stored contiguously on disk to provide high read throughput. The choice of block size influences how long these high throughput reads run for, and over how many nodes a file is distributed. When reading the many blocks of a single file, a too low block size spends more overall time in slow disk seek, and a large block size has reduced parallelism. Data processing that is I/O heavy benefits from larger block sizes, and data processing that is CPU heavy benefits from smaller block sizes.

The default provided by Cloudera Manager is 128MB. The block size can also be specified by an HDFS client on a per-file basis.

### Replication

Bottlenecks can occur on a small number of nodes when only small subsets of files on HDFS are being heavily accessed. Increasing the replication factor of the files so that their blocks are replicated over more nodes can alleviate this. This is done at the expense of storage capacity on the cluster. This can be set on individual files, or recursively on directories with the -R parameter, by using the Hadoop shell command `hadoop fs -setrep`. By default, the replication factor is 3.

### Erasure Coding

Erasure Coding (EC) is an alternative to the 3x replication scheme. See Data Durability for details on how EC works.

EC reconstruction is nearly free computationally with the Intel ISA-L codec. Performance improvements can be obtained by speeding up the computations through the use of the following instruction sets: AES-NI, SSE, AVX, AVX2, and AVX 512. To determine whether a node has ISA-L instruction sets available, check  if any of the following CPU flags are present.

```
$ grep flags /proc/cpuinfo
aes
sse
avx
avx2
avx512f
avx512cd
```

It is important that edge nodes and client gateways have codec support so that they can do the calculations.

Erasure Coding levies additional demands on the number of nodes or racks required to achieve fault tolerance:

- **node-level**: number of <u>DataNodes</u> needed to equal or exceed the data stripe width
- **rack-level**: number of <u>racks</u> needed to equal or exceed the data stripe width

For example, for a RS-10-4 policy to be rack-failure tolerant, you need at least 14 racks (10 for data blocks, 4 for parity blocks); for host-failure tolerance you need at least 14 nodes.

Erasure Coding will observe rack topology, but the resulting block placement policy (BPP) differs from replication. With EC, the BPP tries to place all blocks as evenly on all racks as possible. Cloudera recommends that racks have a consistent number of nodes. Racks with fewer DataNodes are busier and fill faster than racks with more DataNodes.

> **Warning:**
>
> Impala and HBase queries fail if they attempt to access Erasure Coded data. For more information, see <u>Data Durability</u>.

### *Rack Awareness*

Hadoop optimizes performance and redundancy when <u>rack awareness is configured</u> for clusters that span across multiple racks, and Cloudera recommends doing so. Rack assignments for nodes can be configured within the Cloudera Manager.

When setting up a multi-rack environment, place each master node on a different rack. In the event of a rack failure, the cluster continues to operate using the remaining master(s).

### *DataNode Failed Volumes Tolerated*

By default, Cloudera Manager sets the HDFS datanode failed volume threshold to half of the data drives in a datanode. If each datanode has eight drives dedicated to data storage, this threshold would be set to four, meaning that HDFS marks the datanode dead on the fifth drive failure. This number may need to be adjusted up or down depending on internal policies regarding hard drive replacements, or because of evaluating what behavior is actually seen on the cluster under normal operating conditions. Setting the value too high has a negative impact on the Hadoop cluster. Specifically for YARN, the number of total containers available on the node with many drive failures is the same as nodes without drive failures, meaning data locality is less likely on the former, leading to more network traffic and slower performance.

> **Important:**
>
> Multiple drive failures in a short amount of time might be indicative of a larger problem with the machine, such as a failed disk controller.

### *DataNode Max Transfer Threads Count*

This parameter replaces the deprecated dfs.datanode.max.xcievers parameter, and would need to be adjusted for workloads like <u>HBase to ensure that the DataNodes serve adequate number of files at any one time</u>. Failure to do so can result in error messages about exceeding the number of transfer threads or missing blocks.

### Balancing

HDFS  spreads data evenly across the cluster to optimize read access, MapReduce performance, and node utilization. Over time it is possible that the data distribution in the cluster can become out of balance due to a variety of reasons. Hadoop can help mitigate this by rebalancing data across the cluster using the balancer tool. Running the balancer is a manual process that can be executed from within Cloudera Manager as well as from the command line. By default, Cloudera Manager configures the balancer to rebalance a datanode when its utilization is 10% more or less from the average utilization across the cluster. Individual datanode utilization can be viewed from within Cloudera Manager.

By default the maximum bandwidth a datanode uses for rebalancing is set to 1 MB/second (8 Mbit/second). This can be increased but network bandwidth used by rebalancing could potentially impact production cluster application performance. Changing the balancer bandwidth setting within Cloudera Manager requires a restart of the HDFS service, however this setting can also be made instantly across all nodes without a configuration change by running the command:

```
hdfs dfsadmin -setBalancerBandwidth <bytes_per_second>
```

This command must be run as an HDFS superuser. This is a convenient way to change the setting without restarting the cluster, but since it is a dynamic change, it does not persist if the cluster is restarted. The [Recommended](#) configurations for the Balancer page provides more insights into scenarios and tunables with suggested values.

> **Important:**
>
> Cloudera does not recommend running the balancer on an HBase cluster as it affects data locality for the RegionServers, which can reduce performance. Unfortunately, when HBase and YARN services are colocated and heavy usage is expected on both, there is not a good way to ensure the cluster is optimally balanced.

You can configure HDFS to distribute writes on each DataNode in a manner that balances out available storage among that DataNode's disk volumes.

By default, a DataNode writes new block replicas to disk volumes solely on a round-robin basis. You can configure a volume-choosing policy that causes the DataNode to take into account how much space is available on each volume when deciding where to place a new replica.

For more information, see [Configuring Storage Balancing for DataNodes](#).

### YARN

The YARN service manages MapReduce and Spark tasks. Applications run in YARN containers, which use Linux Cgroups for resource management and process isolation. The Cloudera Installation and Upgrade manual has a [section on YARN tuning guidance](#).

### Impala

The Impala service is a distributed, MPP database engine for interactive performance SQL queries over large data sets. Impala performs best when it can operate on data in memory; as a result, Impala is often configured with a very large heap size.

Impala daemon processes must be colocated with HDFS data nodes to use HDFS local reads, which also improve performance.

Impala does not provide any built-in load balancing, so a production Impala deployment should be deployed behind a load balancer for performance and high availability. The Cloudera Impala product documentation contains detailed information on configuring Impala with a load balancer: Using Impala through a Proxy for High Availability.

The Cloudera Impala product documentation also contains a section on Impala performance tuning that should be reviewed prior to a production deployment: Tuning Impala for Performance.

### Spark

Cloudera supports Spark on YARN-managed deployments for a more flexible and consistent resource management approach. When running under Spark, the number of executors (YARN containers) can be specified when submitting the job. As of CDH 5.5, dynamic allocation is enabled by default but can be disabled either by setting `spark.dynamicAllocation.enabled=false`. If `--num-executors` is specified in the job, dynamic allocation is disabled implicitly. Additional information on Spark configuration and management is available in the Cloudera Administration manual: Managing Spark.

Spark standalone mode is not supported.

### HBase

#### *Automatic Major Compaction*

By default major compactions happen every 7 days. The next major compaction happens 7 days after the last one has finished.  This means that the actual time that major compaction happens can impact production processes, which is not ideal if it is desired to run compactions at a specific known off-peak hour (For example, 3 AM).  Cloudera recommends disabling automatic major compaction by setting the interval to zero (hbase.hregion.major.compaction = 0).  Major compactions should then be run through cron calling the HBase admin tool.

### Search

Cloudera Search, based on Apache Solr, provides a distributed search engine service. Search engines are often expected to provide fast, interactive performance so it is important to allocate sufficient RAM to the search service.

If other resource intensive applications, such as Impala, are deployed on the same cluster, it is important to use the resource management facilities in Cloudera Manager. In some cases, it may also be preferable to avoid colocating the search service with other services.

### Oozie

Writing Oozie XML configuration files can be tedious and error-prone. Cloudera recommends using the Oozie editor provided by Hue for workflow creation, scheduling, and execution.

### Kafka

Kafka's default configuration with Cloudera Manager is suited to start development quickly. Several default settings should be changed before deploying a Cloudera Kafka cluster in production.

The default ZooKeeper Kafka root is `/`. However, Cloudera recommends changing this to `/kafka`. This is the location within ZooKeeper where the znodes for the Kafka cluster are stored. As long as a single Kafka cluster is using the ZooKeeper service, using `/kafka` is advised. If multiple Kafka clusters (For example:. dev, test, qa) are sharing a ZooKeeper service, each Kafka instance should have a unique ZooKeeper root (For example: `/kafka-dev`, `/kafka-test`, `/kafka-qa`).

Cloudera Manager enables Kafka topic auto-creation by default. any data written to a topic causes the creation of that topic if it does not already exist. While this may be convenient in prototyping and development, it should not be used in production as it leads to cases where arbitrary topics can be created and data may be written to the wrong topic in the case of an improper application configuration without leading to an error.

Cloudera Manager sets the default minimum number of in-sync replicas (ISR) to 1. This should generally be increased to a minimum of 2 in a production cluster to prevent data loss.

The Kafka Maximum Process File Descriptors setting may need to be increased in some cases in production deployments. This value can be monitored in Cloudera Manager and increased if usage requires a larger value than the default 64 k limit.

The default data retention time for Kafka is often acceptable for production but should be reviewed for use case suitability.

### Kudu

#### *Limitations*

Current versions of Kudu come with a number of [usage limitations](#):

- Kudu does not currently support multi-row transactions. Operations that affect multiple rows will not roll back if the operation fails part way through. This should be mitigated by exploiting the primary key uniqueness constraint to make operations idempotent.

#### *Partitioning Guidelines*

Kudu supports partitioning tables by RANGE and HASH partitions. The RANGE and HASH partitions can be combined to create effective partitioning strategies. It is also possible to utilize non-covering RANGE partitions.

For large tables, such as fact tables, aim for as many tablets as you have cores in the cluster.

For small tables, such as dimension tables, aim for a large number of tablets and each tablet is of at least 1 GB in size.

Further details can be found in the [Apache Kudu Guide](#).

> **Note:**
>
> In general, be mindful of the number of tablets, limit the parallelism of reads, in the current implementation. Increasing the number of tablets significantly beyond the number of cores is likely to have diminishing returns.

## What's new in the stack?

### Atlas

Apache Atlas provides data governance capabilities for Hadoop. Apache Atlas serves as a common metadata store that is designed to exchange metadata both within and outside of the Hadoop stack. Close integration of Atlas with Apache Ranger enables you to define, administer, and manage security and compliance policies consistently across all components of the Hadoop stack. Atlas provides metadata and lineage to Data Steward Studio to support curating data across enterprise data.

### Ranger

Cloudera Data Platform security components enable you to control access to CDP services, and also provide access to auditing and reporting.

### DAS

Data Analytics Studio (DAS) is an application that provides diagnostic tools and intelligent recommendations to make the business analysts self-sufficient and productive with Hive.

DAS helps you to perform operations on Hive tables and provides recommendations for optimizing the performance of your queries.

### Phoenix

Apache Phoenix is an add-on for Apache HBase that provides a programmatic ANSI SQL interface. Apache Phoenix implements best-practice optimizations to enable software engineers to develop HBase based next-generation applications that operationalize big data. Using Phoenix, you can create and interact with tables in the form of typical DDL/DML statements using the Phoenix standard JDBC API.

### Ozone (Tech Preview)

Ozone is a scalable, redundant, and distributed object store, optimized for big data workloads. Apart from scaling to billions of objects of varying sizes, Ozone can function effectively in containerized environments such as Kubernetes and YARN.

Ozone consists of three important storage elements: volumes, buckets, and keys. Each key is part of a bucket, which, in turn, belongs to a volume. Only an administrator can create volumes. Depending on their requirements, users can create buckets in volumes. Ozone stores data as keys inside these buckets.

When a key is written to Ozone, the associated data is stored on the DataNodes in chunks called blocks. Therefore, each key is associated with one or more blocks. Within the DataNodes, a series of unrelated blocks are stored in a container, allowing many blocks to be managed as a single entity.

> **Important**
>
> Ozone is available for technical preview and considered to be under development. Do not use this component in your production systems. If you have questions regarding Ozone, contact support by logging a case on the Cloudera Support Portal.

Some key considerations for Ozone Deployments are provided below --

- The Ozone implementation is a separate instance (from HDFS). Think of it as having two independent HDFS services installed on the same servers. The disks can be shared but it is recommended to use separate/dedicated disks for Ozone.
- Ozone has two master servers (Ozone Manager, and Storage Container Manager). They use rocksdb to persist some of the metadata on the disk, it is recommended to use SSD there.
- Datanode requires storage for replication (log of RAFT consensus protocol) and the storage (raw data + rocksdb metadata). It is recommended to use SSD for the Ratis metadata, as for correctness, it requires frequent sync to the disk. Storage partitions can be placed on spinning disks (hdds.datanode.dir).
- Similar to HDFS you can use multiple drives to store the data (to spread the io pressure between multiple disks).
- There is one stateless service s3 gateway which can be started in multiple instances in case of high load (this transforms the s3 REST protocol to the internal Ozone protocol).
- Recon server is a stateful server to do historical data analysis and it contains  a generic unified web UI.

# Security Integration

The Cloudera Security guide is intended for system administrators who want to secure a cluster using data encryption, user authentication, and authorization techniques.

It provides conceptual overviews and how-to information about setting up various Hadoop components for optimal security, including how to set up a gateway to restrict access. The guide assumes that you have basic knowledge of Linux and systems administration practices.

# Common Questions

## Multiple Data Centers

CDP Data Center deployments are restricted to data centers within a geographic region. Single clusters spanning large geographic regions are not supported. See Appendix A: Spanning Multiple Data Centers for more information.

## Operating Systems

A list of supported operating systems for Cloudera Runtime and Cloudera Manager can be found [here.](here.)

## Storage Options and Configuration

The HDFS data directories should use local storage, which provides all the benefits of keeping compute resources close to the storage and not reading remotely over the network.

The root device size for Cloudera Enterprise clusters should be at least 500 GB to allow parcels and logs to be stored.

To guard against data center failures, you can set up a scheduled distcp operation to persist data to a supported cloud object storage platform (see the examples in the [distcp documentation](distcp documentation)) or leverage Cloudera Manager's [Backup and Data Recovery (BDR)](Backup and Data Recovery (BDR)) features to backup data to another running cluster.

## Relational Databases

Cloudera Enterprise deployments require relational databases for the following components: Cloudera Manager, Hive metastore, Hue, Ranger, Atlas, Oozie, and others. The database credentials are required during Cloudera Enterprise installation.

See [Cloudera Manager and Managed Service Datastores](Cloudera Manager and Managed Service Datastores) for more information.

> **Important:**
>
> The embedded PostgreSQL database is not supported for use in production systems.

# Appendix A: Spanning Multiple Data Centers

Spanning a Cloudera cluster across multiple data centers can provide highly available services and further protect data against host, rack, and data center failures.

Coudera recommends the following deployment methodology when spanning a Cloudera cluster across multiple data centers.

## Networking Requirements

Deploying across multiple data centers can introduce new challenges. Network throughput -- all but guaranteed on local networks -- becomes a more limited and often more expensive commodity. Similarly, cross-site links are subject to higher and variable latency as network traffic travels greater distances.

Consult the [Network Specification](Network Specification) earlier in this document. The deployment topology remains the same, except that the network core spans numerous data centers and is connected by WAN links. Cloudera recommends 10 Gbps switches with sufficient port density to accommodate cross-site links. Redundant links connecting each site are recommended.

Network throughput between sites depends on the number of nodes at each data center. Oversubscription ratios up to 4:1 are generally fine for balanced workloads, but network monitoring is needed to ensure uplink bandwidth is not the bottleneck for Hadoop.

Network latency between sites should not exceed 10 ms.

## Provisioning

Provision hosts within different data centers, each data center with a dedicated subnet. In this way the entire cluster can exist within a handful of subnets and cross-site security ACLs can be more easily managed to permit traffic to and from each data center.

Deploy across three (3) data centers within a single geographical region. This might not be possible within your preferred region due to availability or network requirements.

Note: Network latency tends to be higher and less predictable across geographic regions. Cloudera does not recommend or support spanning clusters across geographic regions.

## Cloudera Runtime Deployment

Deploy HDFS NameNode in high availability mode with Quorum Journal nodes, with each master placed in a different data center. For example, if you have deployed the primary NameNode to a datacenter in Virginia you might deploy your standby NameNode to a data center in Maryland or Pennsylvania. You should place a QJN in each data center.

Although HDFS currently supports only two NameNodes, the cluster can continue to operate if any one host, rack, or data center fails:

- lose active NameNode, standby NameNode takes over
- lose standby NameNode, active is still active; promote third data center master to be new standby NameNode
- lose data center without any NameNode, still have two viable NameNodes

Deploy YARN ResourceManager nodes in a similar fashion.

Deploy a three node ZooKeeper quorum, one located in each data center.

Deploy edge nodes to all three data centers and configure client application access to all three.

Configure rack awareness, one rack per data center. For example, your rack IDs might look like this:

- /va-ashburn
- /va-herndon
- /va-sterling

If you are using Erasure Coding, you must have a consistent number of DataNodes in each data center. The requirements for rack-level tolerance also apply.You need to have a number of data centers equal to or exceeding data width of the RS policy.

## Considerations

There may be data transfer costs associated with host network data sent between data centers, not limited to setup fees, monthly charges, ongoing costs for personnel and maintenance, and hardware refresh.

DFS throughput will be less than if cluster nodes were provisioned within a single data center.

Network throughput and latency can vary based on WAN link. Network throughput and latency must be verified for suitability before deploying to production and routinely throughout the life of the project.

# References

## Cloudera Data Platform Data Center

Product Documentation
>            Hardware Requirements Guide
>            Requirements and Supported Versions
>            Security Guide

Cloudera Services & Support

.

# Acknowledgements

Thanks to Travis Campbell, Alex Moundalexis, Marton Elek, Jay O'Leary,  J P Player, Ali Bajwa, David Fowler, Andreas Nemeth, Dwai Lahiri for their help with the review/feedback on, and creation of this document.

Thanks to Asha Kadam and other members of the Cloudera Documentation team for their editorial support.