Hortonworks Data Platform

(May 26, 2015)

docs.hortonworks.com

Hortonworks Data Platform: Getting Started Guide

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including YARN, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper and Ambari. In Release 2.1, HDP expands to include Falcon, Sqoop, Flume, Tez and Storm.

Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

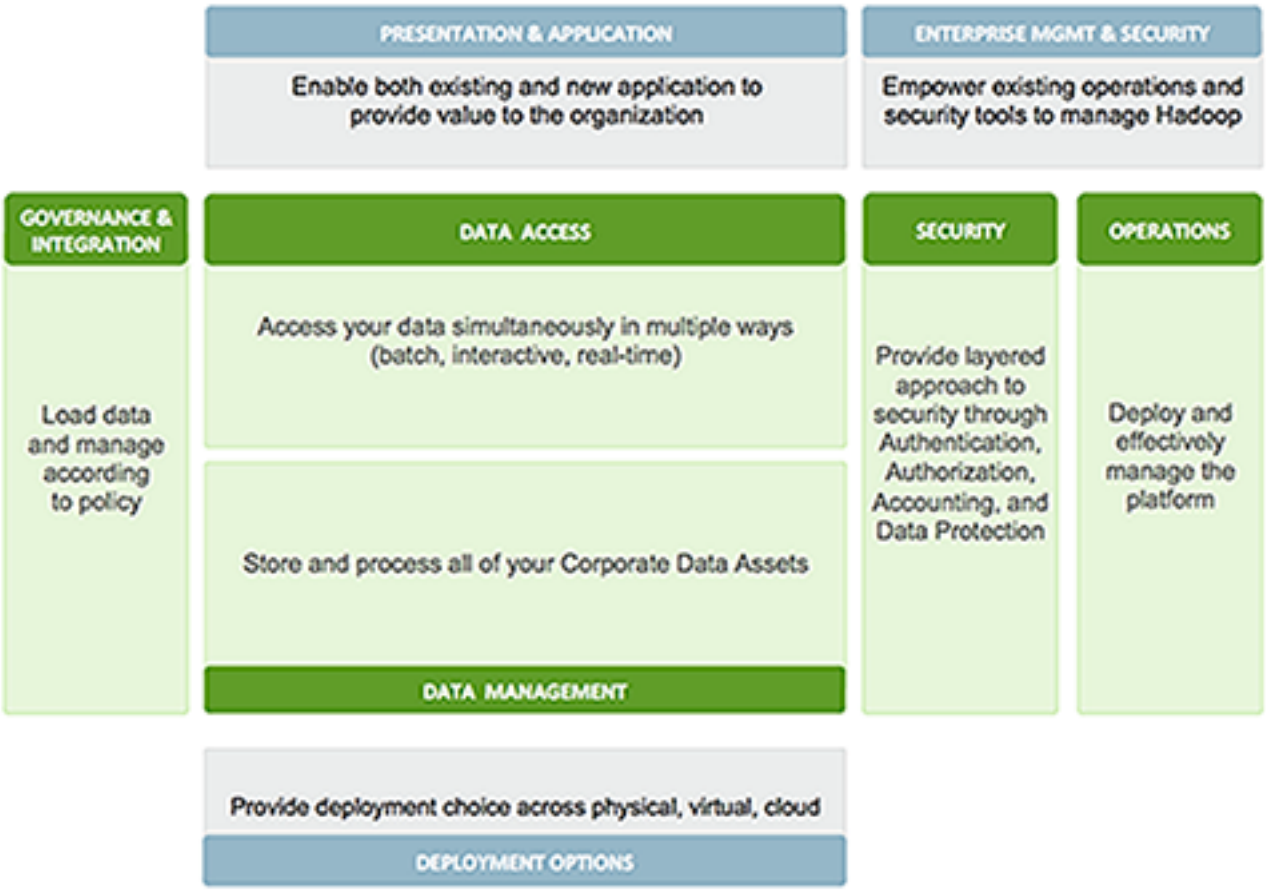
Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. *All of our technology is, and will remain, free and open source.*

For more information on Hortonworks technology, visit the Hortonworks Data Platform page. For more information on Hortonworks services, visit either the Support or Training page. Feel free to contact us directly to discuss your specific needs.

Table of Contents

1. About Hortonworks Data Platform

Hortonworks Data Platform (HDP) is an open source distribution powered by Apache Hadoop. HDP provides you with the actual Apache-released versions of the components with all the latest enhancements to make the components interoperable in your production environment, and an installer that deploys the complete Apache Hadoop stack to your entire cluster.



All components are official Apache releases of the most recent stable versions available. Hortonworks' philosophy is to assure interoperability of the components by patching only when absolutely necessary. Consequently, there are very few patches in the HDP, and they are all fully documented. Each of the HDP components have been tested rigorously prior to the actual Apache release.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. *All of our technology is, and will remain, free and open source.*

To learn more about the distribution details and the component versions, see the Release Notes.

To learn more about the testing strategy adopted at Hortonworks, Inc., see Delivering high-quality Apache Hadoop releases.

1.1. What is Hadoop?

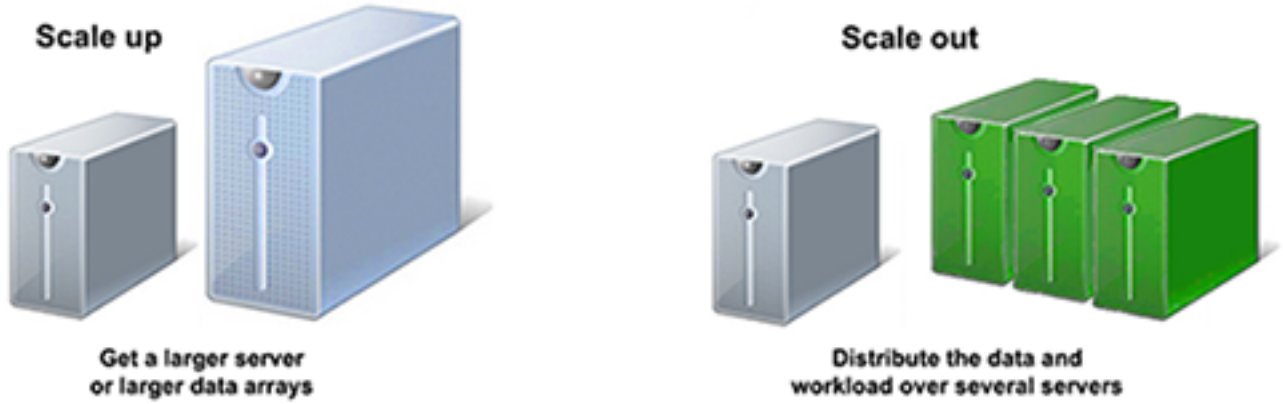
Hadoop is an open source framework for large-scale data processing. Hadoop enables companies to retain and make use of all the data they collect, performing complex analysis quickly and storing results securely over a number of distributed servers.

Traditional large-scale data processing was performed on a number of large computers. When demand increased, the enterprise would 'scale up', replacing existing servers with larger servers or storage arrays. The advantage of this approach was that the increase in the size of the servers had no affect on the overall system architecture. The disadvantage was that scaling up was expensive.

Moreover, there's a limit on how much you can grow a system set up to process terabytes of information, and still keep the same system architecture. Sooner or later, when the amount of data collected becomes hundreds of terabytes or even petabytes, scaling up comes to a hard stop.

Scale-up strategies remain a good option for businesses that require intensive processing of data with strong internal cross-references and a need for transactional integrity. However, for growing enterprises that want to mine an ever-increasing flood of customer data, there is a better approach.

Using a Hadoop framework, large-scale data processing can respond to increased demand by "scaling out": if the data set doubles, you distribute processing over two servers; if the data set quadruples, you distribute processing over four servers. This eliminates the strategy of growing computing capacity by throwing more expensive hardware at the problem.



When individual hosts each possess a subset of the overall data set, the idea is for each host to work on their own portion of the final result independent of the others. In real life, it's likely that the hosts will need to communicate between each other, or that some pieces of data will be required by multiple hosts. This creates the potential for bottlenecks and increased risk of failure.

Therefore, designing a system where data processing is spread out over a number of servers, requires designing for self-healing distributed storage, fault-tolerant distributed

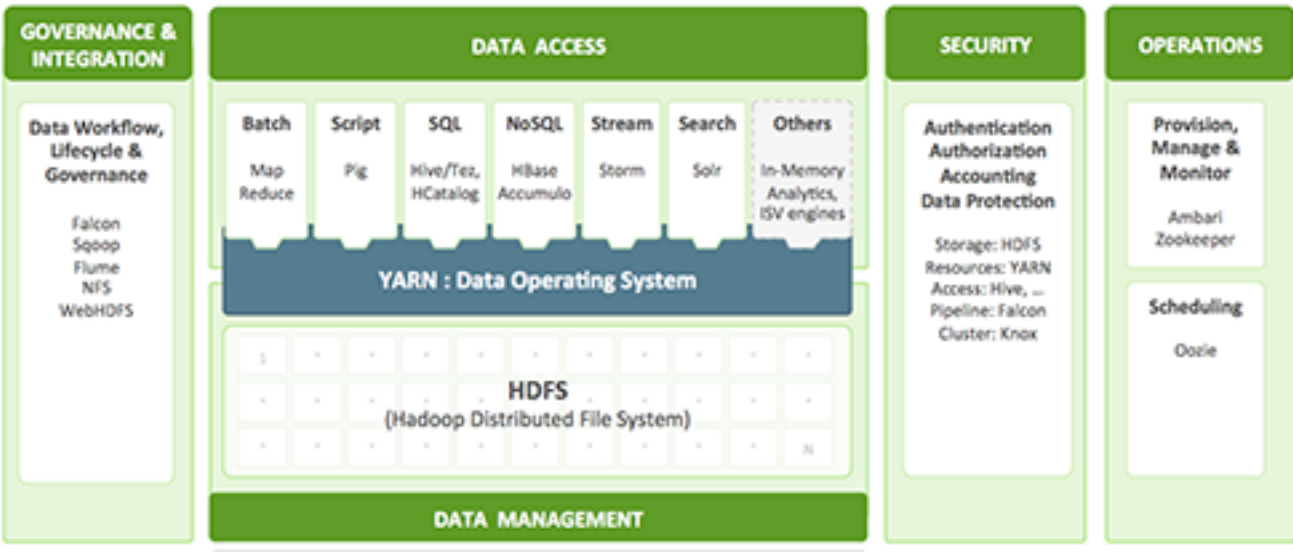
computing, and abstraction for parallel processing. Since Hadoop is not actually a single product but a collection of several components, this design is generally accomplished through deployment of various components.

The Hadoop "ecosystem" includes components that support critical enterprise requirements such as Security, Operations and Data Governance. To learn more about how critical enterprise requirements are implemented within the Hadoop ecosystem, see Understanding the Hadoop Ecosystem.

For a look at a typical Hadoop cluster, see A Typical Hadoop Cluster.

2. Understanding the Hadoop Ecosystem

HDP provides a number of core capabilities that ingetrate Hadoop with your existing data center technologies and team capabilities, creating a data architecture that can scale across the enterprise:

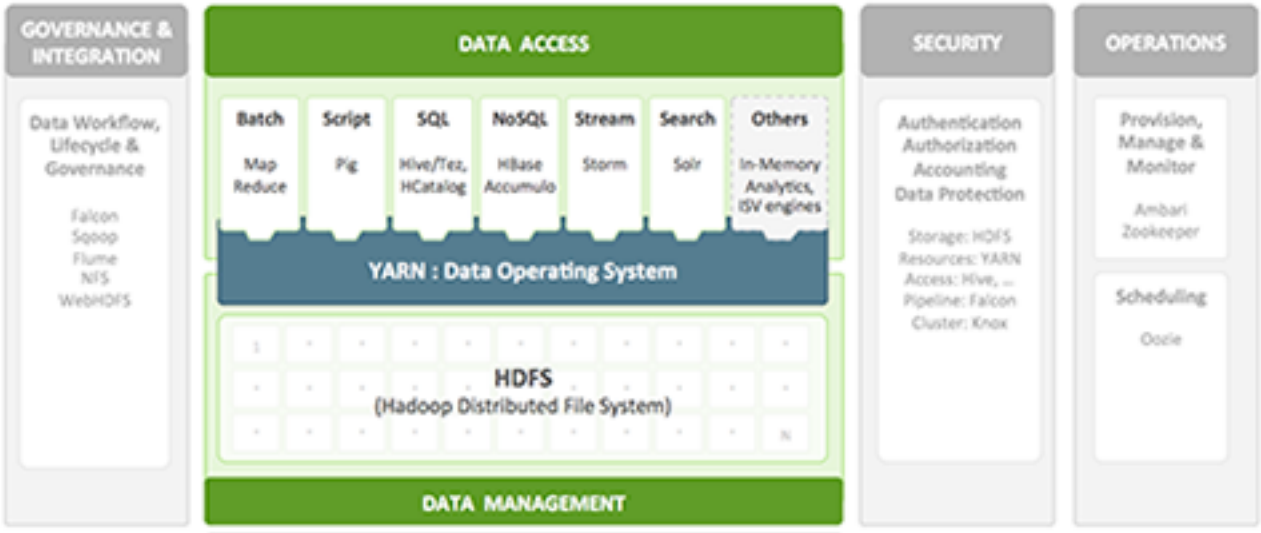


We discuss this architecture in the following sections:

- Data access and data management
- Security
- Operations
- Governance and integration

2.1. Data access and data management

As noted in What Is Hadoop, the Apache Hadoop framework enables the distributed processing of large data sets across clusters of commodity computers using a simple programming model. Hadoop is designed to scale up from single servers to thousands of machines, each providing computation and storage. Rather than rely on hardware to deliver high availability, the framework itself is designed to detect and handle failures at the application layer. This enables it to deliver a highly-available service *on top of* a cluster of computers, any of which might be prone to failure.



The core components of HDP are YARN and the Hadoop Distributed File System (HDFS).

As part of Hadoop 2.0, **YARN** handles the process resource management functions previously performed by MapReduce.

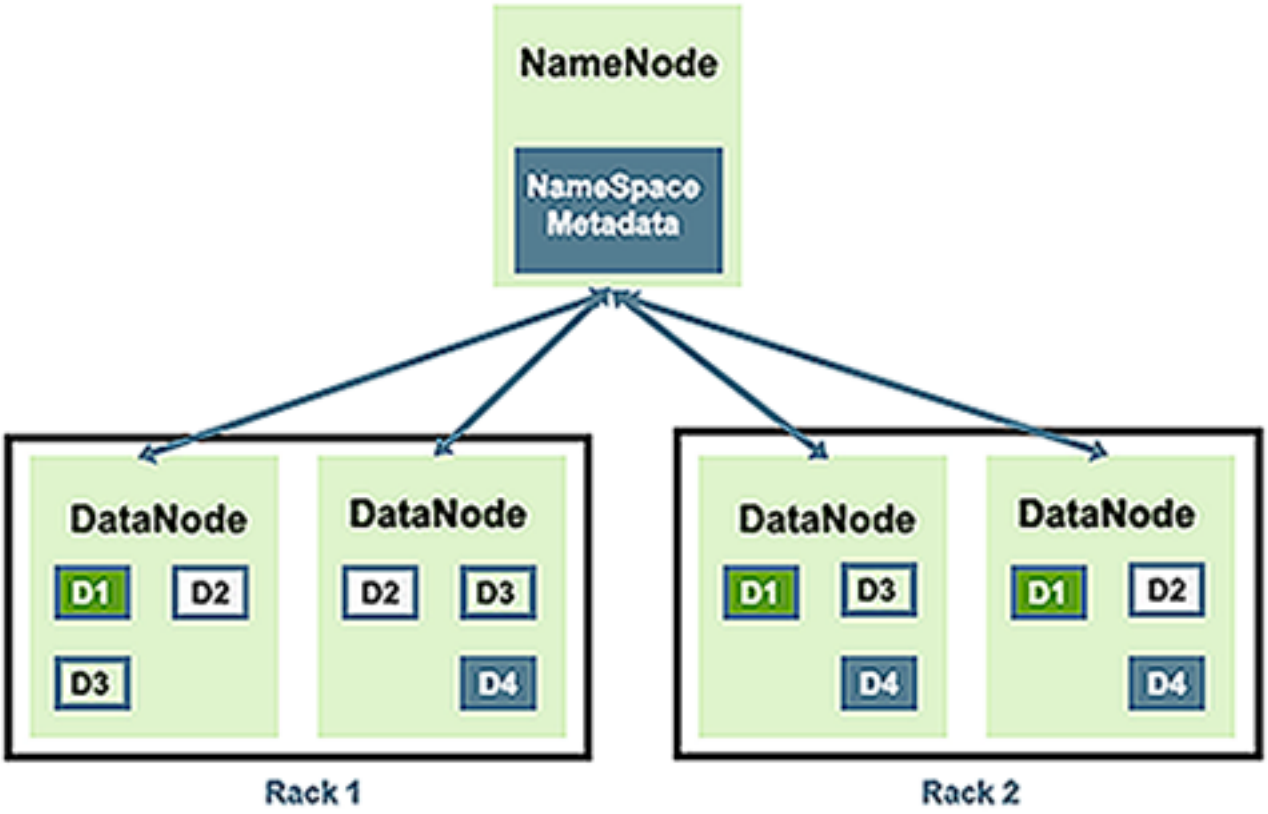
YARN is designed to be co-deployed with HDFS such that there is a single cluster, providing the ability to move the computation resource to the data, not the other way around. With YARN, the storage system need not be physically separate from the processing system.

YARN provides a broader array of possible interaction patterns for data stored in HFDFS, beyond MapReduce; thus, multiple applications can be run in Hadoop, sharing a common resource management. This frees MapReduce to process data.

HDFS (storage) works closely with MapReduce (data processing) to provide scalable, fault-tolerant, cost-efficient storage for big data. By distributing storage and computation across many servers, the combined storage resource can grow with demand while remaining economical at every size.

HDFS can support file systems with up to 6,000 nodes, handling up to 120 Petabytes of data. It's optimized for streaming reads/writes of very large files. HDFS data redundancy

enables it to tolerate disk and node failures. HDFS also automatically manages the addition or removal of nodes. One operator can handle up to 3,000 nodes.



We discuss how HDFS works, and describe a typical HDFS cluster, in A Typical Hadoop Cluster.

MapReduce is a framework for writing applications that process large amounts of structured and unstructured data in parallel, across a cluster of thousands of machines, in a reliable and fault-tolerant manner. The *Map* function divides the input into ranges by the InputFormat and creates a map task for each range in the input. The JobTracker distributes those tasks to the worker nodes. The output of each map task is partitioned into a group of key-value pairs for each reduce. The *Reduce* function then collects the various results and combines them to answer the larger problem the master node was trying to solve.

Each reduce pulls the relevant partition from the machines where the maps executed, then writes its output back into HDFS. Thus, the reduce can collect the data from all of the maps for the keys it is responsible for and combine them to solve the problem.

Tez (*SQL*) leverages the MapReduce paradigm to enable the creation and execution of more complex Directed Acyclic Graphs (DAG) of tasks. Tez eliminates unnecessary tasks, synchronization barriers and reads-from and writes-to HDFS, speeding up data processing across both small-scale/low-latency and large-scale/high-throughput workloads.

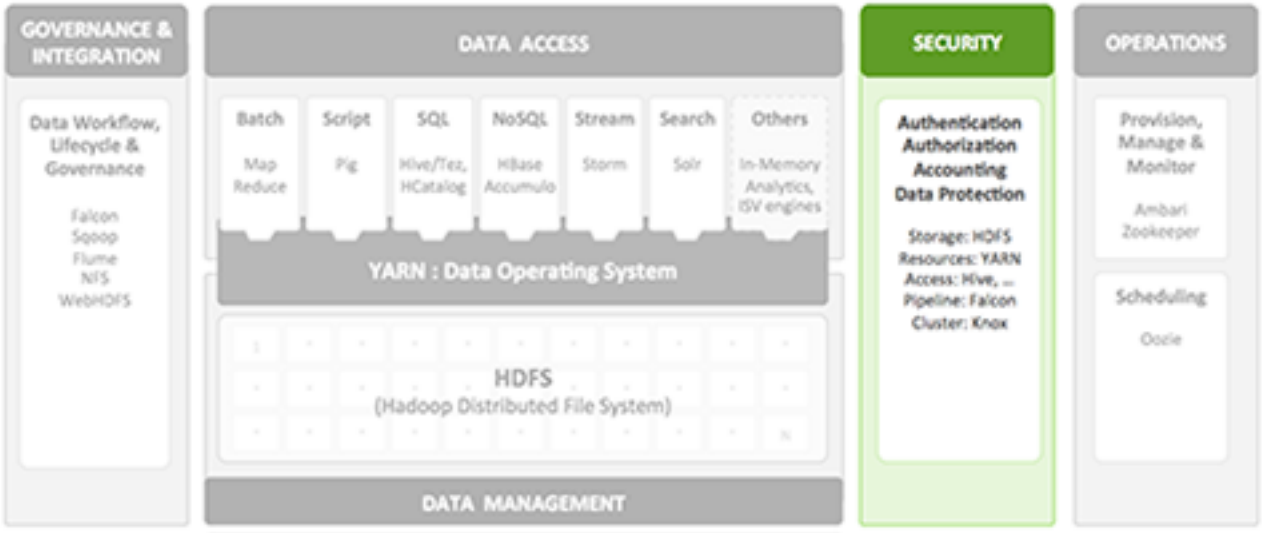
Tez also acts as the execution engine for Hive, Pig and others.

Because the MapReduce interface is suitable only for developers, HDP provides higher-level abstractions to accomplish common programming tasks:

- **Pig** (*scripting*). Platform for analyzing large data sets. It is comprised of a high-level language (Pig Latin) for expressing data analysis programs, and infrastructure for evaluating these programs (a compiler that produces sequences of Map-Reduce programs). Pig was designed for performing a long series of data operations, making it ideal for Extract-transform-load (ETL) data pipelines, research on raw data, and iterative processing of data. Pig Latin is designed to be easy to use, extensible and self-optimizing.
- **Hive** (*SQL*). Provides data warehouse infrastructure, enabling data summarization, ad-hoc query and analysis of large data sets. The query language, HiveQL (HQL), is similar to SQL.
- **HCatalog** (*SQL*). Table and storage management layer that provides users with Pig, MapReduce and Hive with a relational view of data in HDFS. Displays data from RCFile format, text or sequence files in a tabular view; provides REST APIs so that external systems can access these tables' metadata.
- **HBase** (*NoSQL*). Non-relational database that provides random real-time access to data in very large tables. HBase provides transactional capabilities to Hadoop, allowing users to conduct updates, inserts and deletes. HBase includes support for SQL through Phoenix.
- **Accumulo** (*NoSQL*). Provides cell-level access control for data storage and retrieval.
- **Storm** (*data streaming*). Distributed real-time computation system for processing fast, large streams of data. Storm topologies can be written in any programming language, to create low-latency dashboards, security alerts and other operational enhancements to Hadoop.

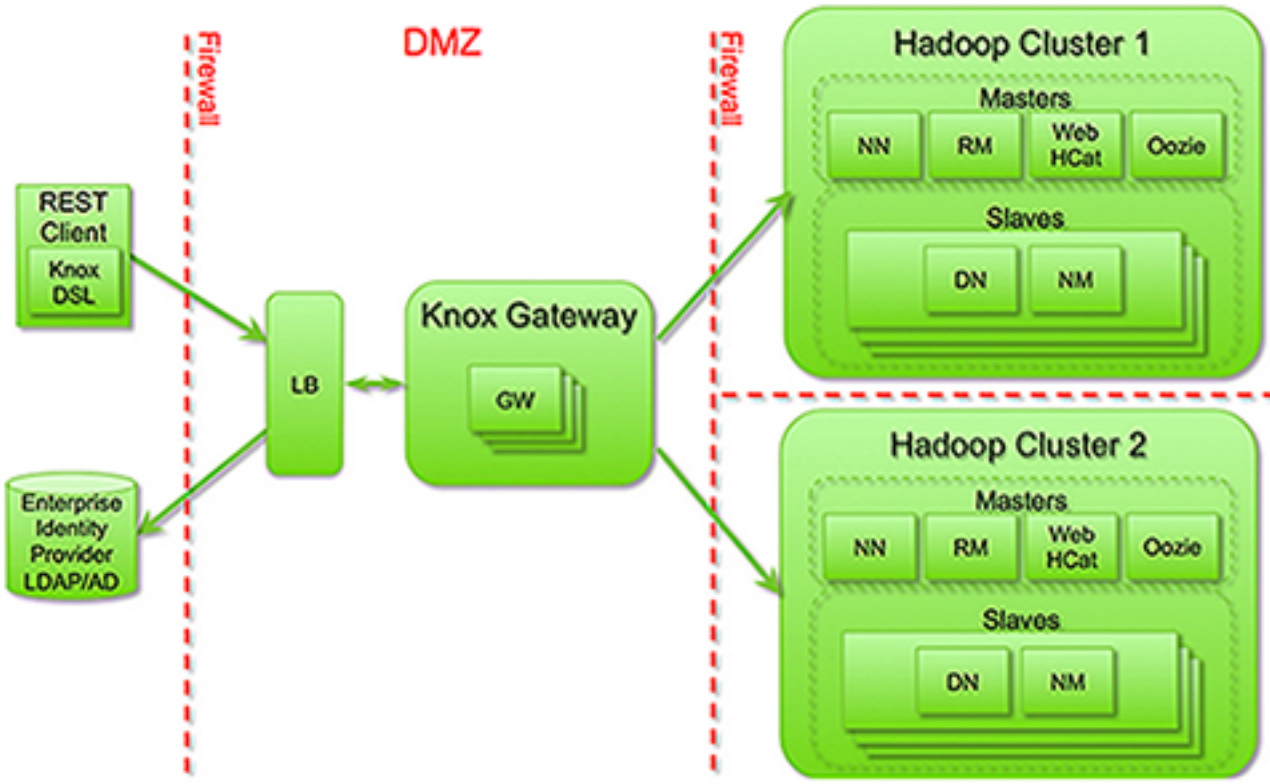
2.2. Security

Enterprise-level security generally requires four layers of security: authentication, authorization, accounting and data protection.



HDP and other members of the Apache open source community address security concerns as follows:

- **Authentication.** Verifying the identity of a system or user accessing the cluster. Hadoop provides both simple authentication and Kerberos (fully secure) authentication while relying on widely-accepted corporate user stores (such as LDAP or Active Directory), so a single source can be used for credentialling across the data processing network.
- **Authorization.** Specifying access privileges for a user or system is accomplished through Knox Gateway, which includes support for the lookup of enterprise group permissions plus the introduction of service-level access control.



- **Accounting.** For security compliance of forensics, HDFS and MapReduce provide base audit support; Hive metastore provides an audit trail showing who interacts with Hive and when such interactions occur; Oozie, the workflow engine, provides a similar audit trail for services.
- **Data protection.** In order to protect data in motion, ensuring privacy and confidentiality, HDP provides encryption capability for various channels through Remote Procedure Call (RPC), HTTP, HDBC/ODBC and Data Transfer Protocol (DTP). HDFS supports encryption at the operating system level.

Other Apache projects in a Hadoop distribution include their own access control features. For example, HDFS has file permissions for fine-grained authorization; MapReduce includes resource-level access control via ACL; HBase provides authorization with ACL on tables and column families; Accumulo extends this further for cell-level access control. Apache Hive provides coarse-grained access control on tables.

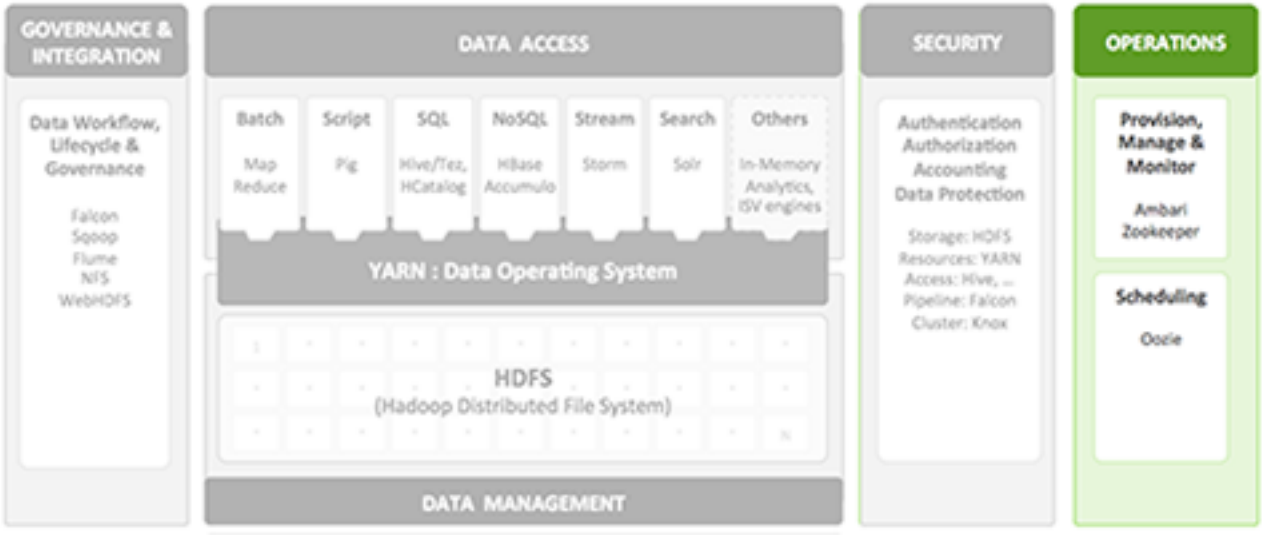
Applications that are deployed with HDFS, Knox and Kerberos to implement security include:

- **Hive** (*access*). Provides data warehouse infrastructure, enabling data summarization, ad-hoc query and analysis of large data sets. The query language, HiveQL (HQL), is similar to SQL.

- **Falcon** (*pipeline*). Server that simplifies the development and management of data processing pipelines by supplying simplified data replication handling, data lifecycle management, data lineage and traceability and process coordination and scheduling. Falcon enables the separation of business logic from application logic via the use of high-level data abstractions such as Clusters, Feeds and Processes. It utilizes existing services like Oozie to coordinate and schedule data workflows.

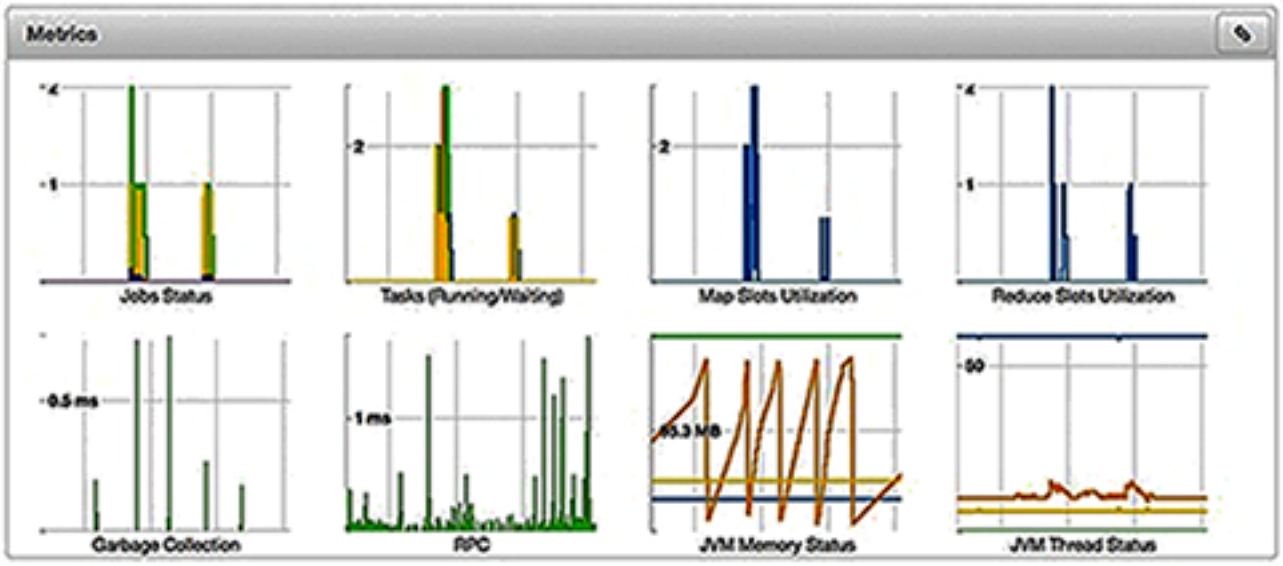
2.3. Operations

Operations components deploy and effectively manage the platform.



Deployment and management tasks include the following:

- **Provisioning, management, monitoring.** Ambari provides an open operational framework for provisioning, managing and monitoring Hadoop clusters. Ambari includes a web interface that enables administrators to start/stop/test services, change configurations, and manage the ongoing growth of the cluster.

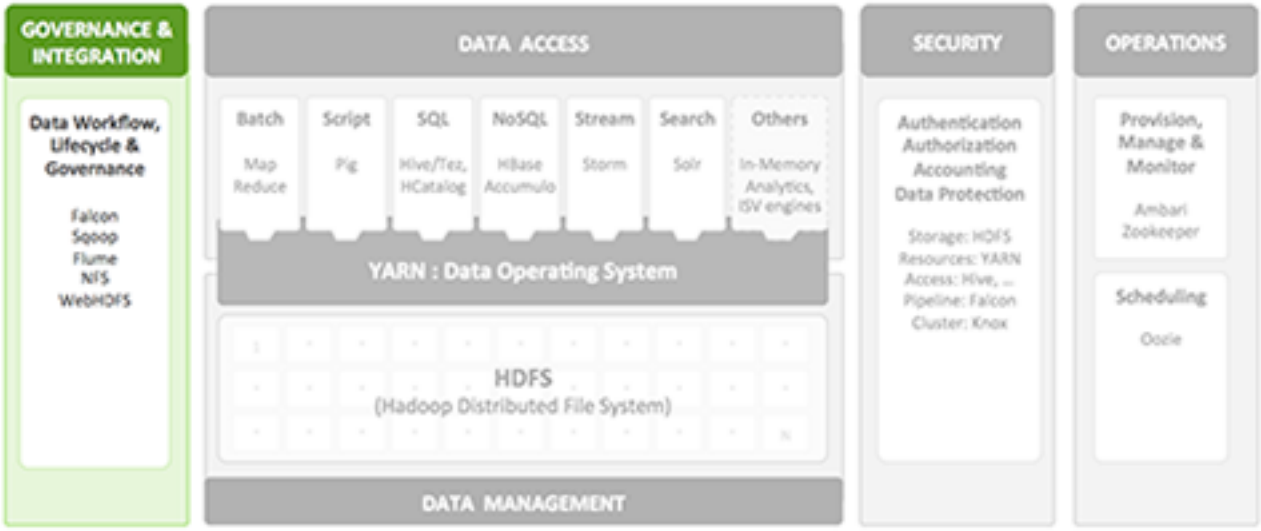


- **Integrating with other applications.** The Ambari RESTful API enables integration with existing tools, such as Microsoft System Center and Teradata Viewpoint. For deeper customization, Ambari also leverages standard technologies with Nagios and Ganglia. Operational services for Hadoop clusters, including a distributed configuration service, a synchronization service and a naming registry for distributed systems, is provided by Zookeeper. Distributed applications use Zookeeper to store and mediate updates to important configuration information.

- **Job scheduling.** Oozie is a Java Web application used to schedule Hadoop jobs. Oozie enables Hadoop administrators to build complex data transformations out of multiple component tasks, enabling greater control over complex jobs and also making it easier to schedule repetitions of those jobs.

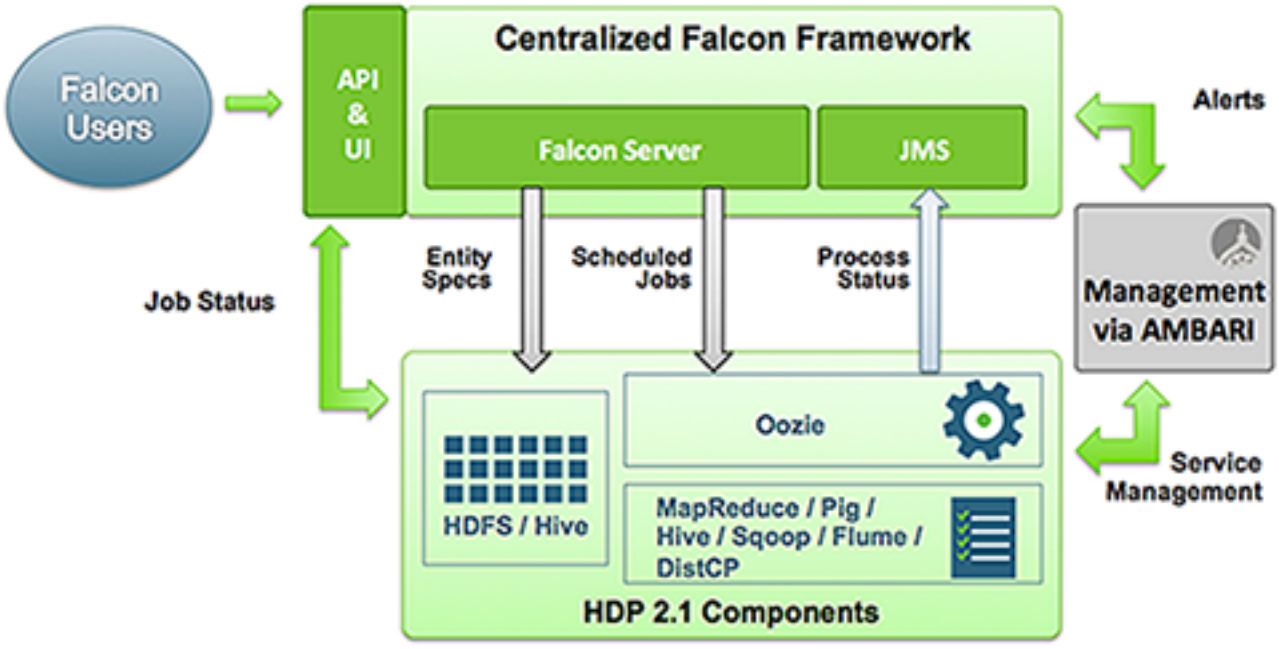
2.4. Governance and integration

Governance and integration components enable the enterprise to load data and manage it according to policy, controlling the workflow of data and the data lifecycle. Goals of data governance include improving data quality (timeliness of current data, "aging" and "cleansing"of data that is growing stale), the ability to quickly respond to government regulations and improving the mapping and profiling of customer data.

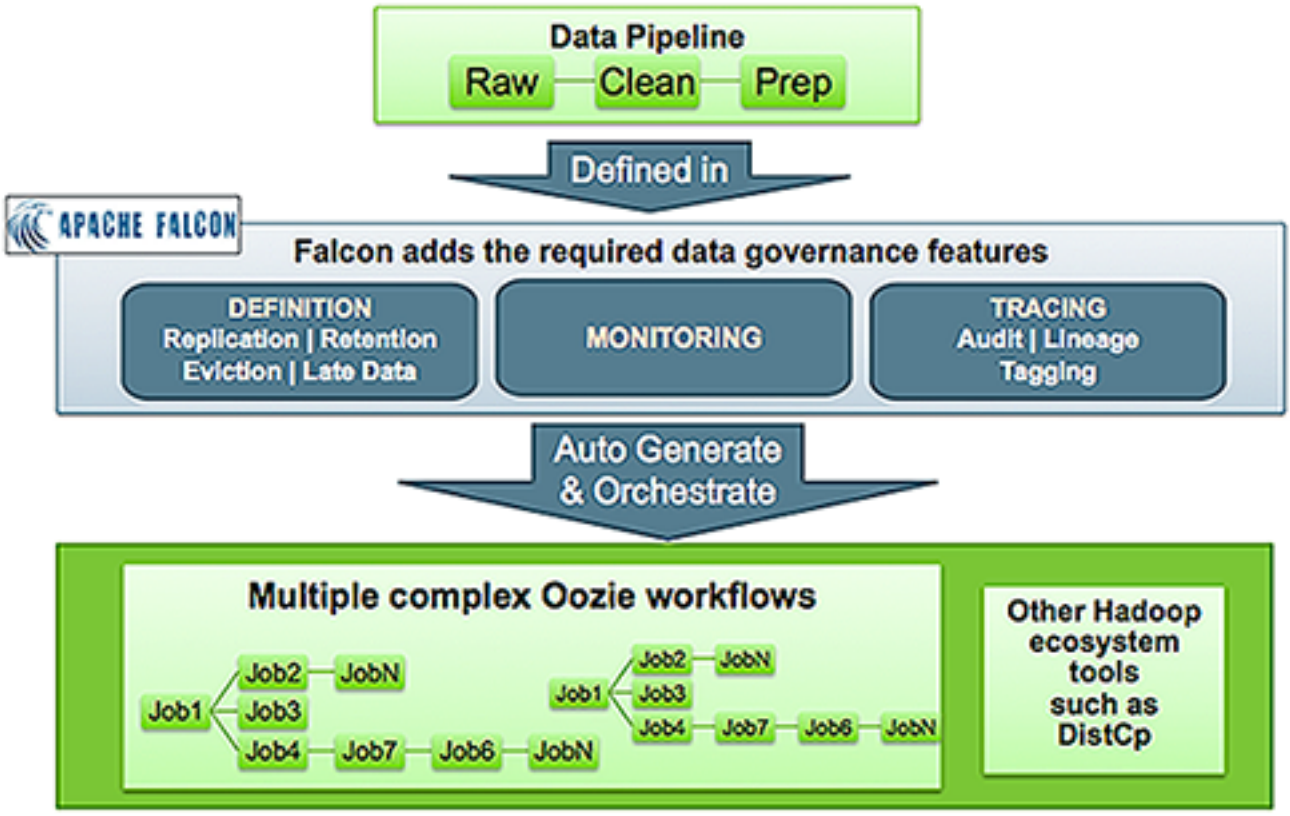


The primary mover of data governance in Hadoop is Apache Falcon. Falcon is a data governance engine that defines data pipelines, monitors those pipelines (in coordination with Ambari), and traces those pipelines for dependencies and audits.

Once Hadoop administrators define their data pipelines, Falcon uses those definition to auto-generate workflows in Apache Oozie, the Hadoop workflow scheduler.



Falcon helps to address data governance challenges that can result when a data processing system employs hundreds to thousands of data sets and process definitions, using high-level, reusable "entities" that can be defined once and reused many times. These entities include data management policies that are manifested as Oozie workflows.



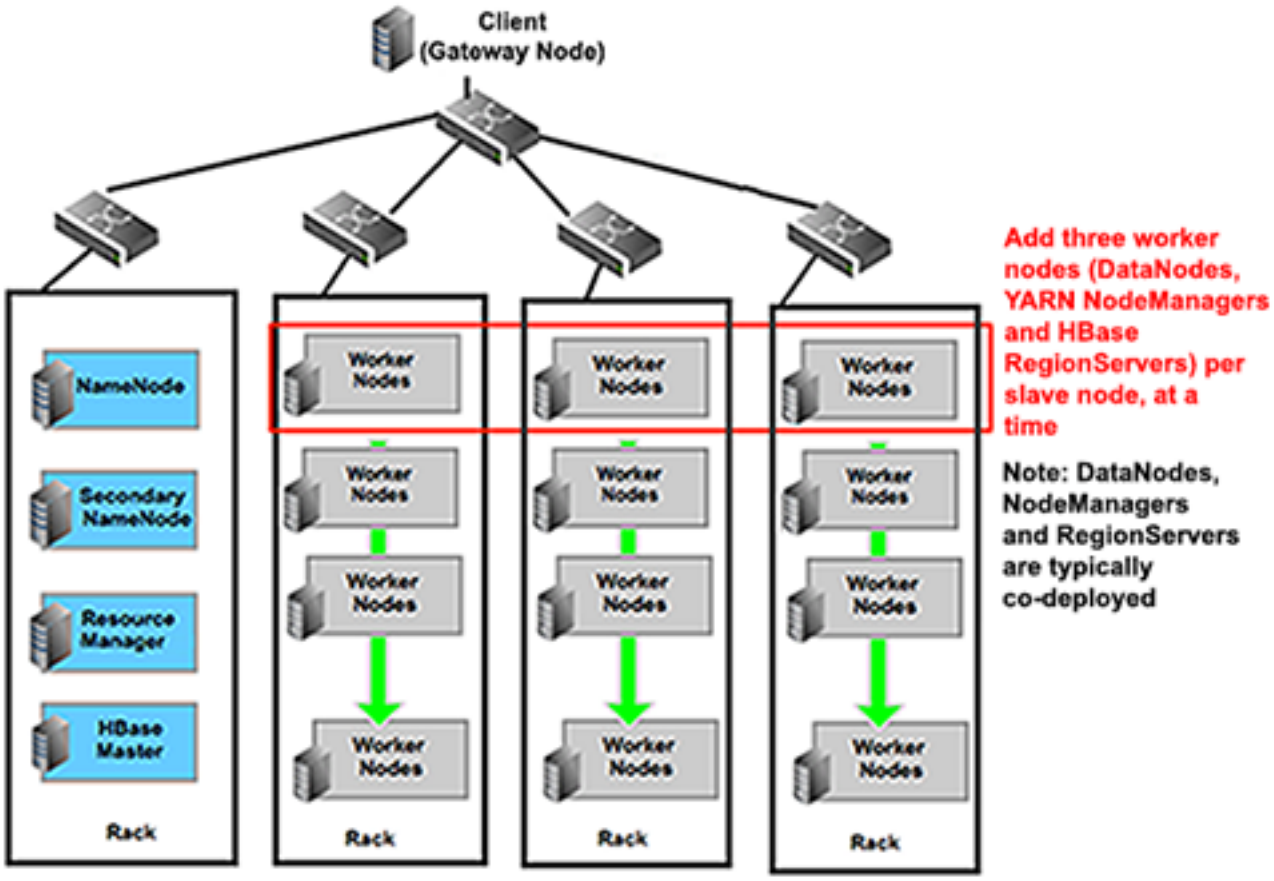
Other applications that are employed to implement Governance and Integration include:

- **Sqoop:** Tool that efficiently transfers bulk data between Hadoop and structured datastores such as relational databases.
- **Flume:** Distributed service that collects, aggregates and moves large amounts of streaming data to the Hadoop Distributed File System (HDFS).
- **WebHDFS:** Protocol that provides HTTP REST access to HDFS, enabling users to connect to HDFS from outside of a Hadoop cluster.

For a closer look at how data pipelines are defined and processed in Apache Falcon, see the hands-on tutorial in the HDP 2.1 Hortonworks Sandbox.

3. Typical Hadoop Cluster

A typical Hadoop cluster is comprised of masters, slaves and clients.



Hadoop and HBase clusters have two types of machines:

- **Masters** (HDFS NameNode, MapReduce JobTracker (HDP-1) or YARN ResourceManager (HDP-2), and HBase Master
- **Slaves** (the HBase RegionServers; HDFS DataNodes, MapReduce TaskTrackers in HDP-1; and YARN NodeManagers in HDP-2). The DataNodes, TaskTrackers/NodeManagers, and HBase RegionServers are co-located or co-deployed to optimize the locality of data that will be accessed frequently.

Hortonworks recommends separating master and slave nodes because task/application workloads on the slave nodes should be isolated from the masters. Slave nodes are frequently decommissioned for maintenance.

For evaluation purposes, it is possible to deploy Hadoop on a single node, with all master and the slave processes reside on the same machine. However, it's more common to configure a small cluster of two nodes, with one node acting as master (running both NameNode and JobTracker/ResourceManager) and the other node acting as the slave (running DataNode and TaskTracker/NodeMangers).

The smallest cluster can be deployed using a minimum of four machines, with one machine deploying all the master processes, and the other three co-deploying all the slave nodes (YARN NodeManagers, DataNodes, and RegionServers). Clusters of three or more machines typically use a single NameNode and JobTracker/ResourceManager with all the other nodes as slave nodes.

Typically, a medium-to-large Hadoop cluster consists of a two- or three-level architecture built with rack-mounted servers. Each rack of servers is interconnected using a 1 Gigabit Ethernet (GbE) switch. Each rack-level switch is connected to a cluster-level switch, which is typically a larger port-density 10GbE switch. These cluster-level switches may also interconnect with other cluster-level switches or even uplink to another level of switching infrastructure.

To maintain the replication factor of three, a minimum of three machines is required per slave node.

For further information, see Data Replication in Apache Hadoop.