Hortonworks Data Platform

2.0

(Mar 18, 2013)

docs.hortonworks.com

Hortonworks Data Platform 2.0 : 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. 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.

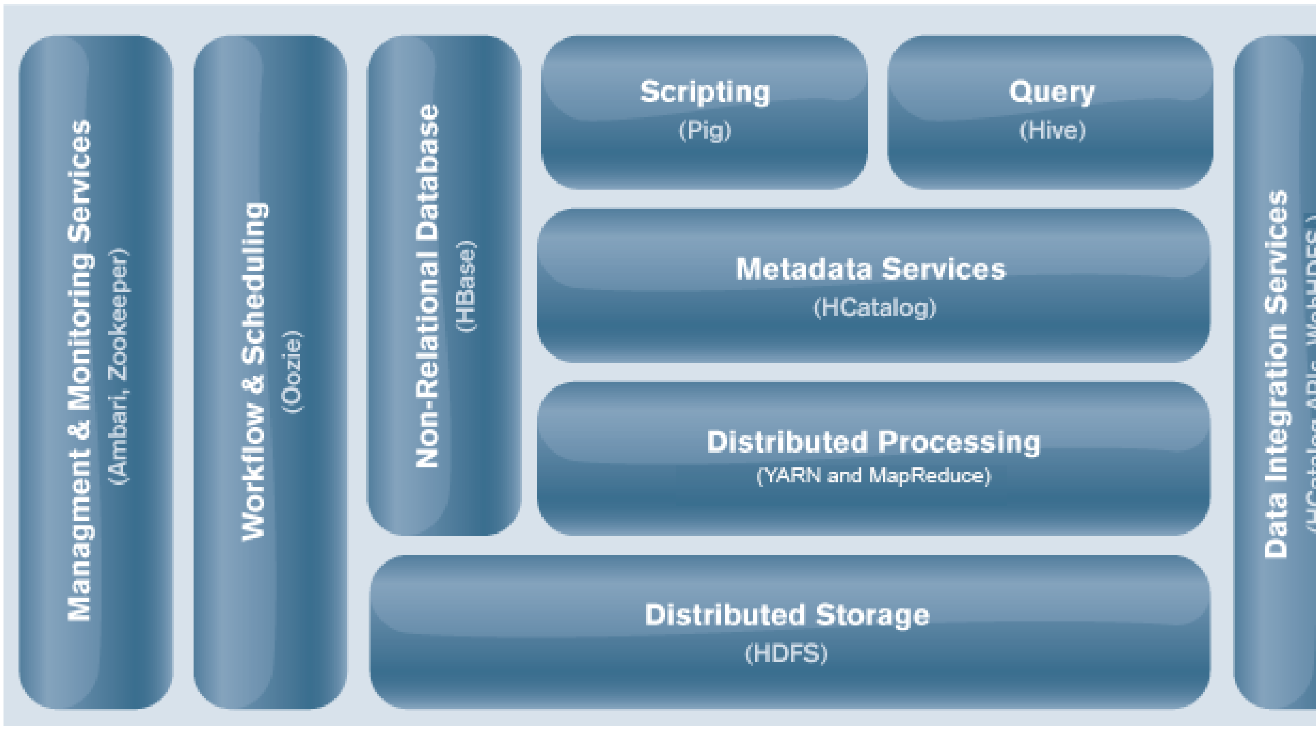
Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

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 necessary bug fixes to make all the components interoperable in your production environments. It is packaged with an easy to use installer (HDP Installer) that deploys the complete Apache Hadoop stack to your entire cluster. The HDP distribution consists of the following components:

1. Core Hadoop platform (Hadoop HDFS and Hadoop MapReduce)
2. Non-relational database (Apache HBase)
3. Metadata services (Apache HCatalog)
4. Scripting platform (Apache Pig)
5. Data access and query (Apache Hive, Tez)
6. Cluster coordination (Apache Zookeeper)
7. Data integration services (HCatalog APIs, WebHCatalog, WebHDFS)



To learn more about the distribution details and the component versions, see the Release Notes. All components are official Apache releases of the most recent stable versions available. Hortonworks' philosophy is to do patches only when absolutely necessary to assure interoperability of the components. 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. To learn more about the testing strategy adopted at Hortonworks, Inc., see: Delivering high-quality Apache Hadoop releases.

2. Understanding Hadoop Ecosystem

Hadoop is an open source project from Apache that has evolved rapidly into a major technology movement. It has emerged as the best way to handle massive amounts of data, including not only structured data but also complex, unstructured data as well.

Its popularity is due in part to its ability to store, analyze and access large amounts of data, quickly and cost effectively across clusters of commodity hardware. Apache Hadoop is not actually a single product but instead a collection of several components.

The following sections provide additional information on the individual components:

- Apache Hadoop core components
- Apache Pig
- Apache Hive
- Tez
- Apache HCatalog
- Apache HBase
- Apache Zookeeper

2.1. Apache Hadoop core components

Apache Hadoop is a framework that allows for the distributed processing of large data sets across clusters of commodity computers using a simple programming model. It 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, thus delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

HDFS (storage) and **YARN** (processing) are the two core components of Apache Hadoop. The most important aspect of Hadoop is that both HDFS and YARN are designed with each other in mind and are co-deployed such that there is a single cluster. This aspect provides the ability to move computation to the data not the other way around. Thus, the storage system is not physically separate from a processing system.

Hadoop Distributed File System (HDFS)

HDFS is a distributed file system that provides high-throughput access to data. It provides a limited interface for managing the file system to allow it to scale and provide high throughput.

HDFS creates multiple replicas of each data block and distributes them on computers throughout a cluster to enable reliable and rapid access.

Note

A file consists of many blocks (large blocks of 64MB and above).

The main components of HDFS are as described below:

- NameNode is the master of the system. It maintains the name system (directories and files) and manages the blocks which are present on the DataNodes.
- DataNodes are the slaves which are deployed on each machine and provide the actual storage. They are responsible for serving read and write requests for the clients.
- Secondary NameNode is responsible for performing periodic checkpoints. In the event of NameNode failure, you can restart the NameNode using the checkpoint.

YARN

The fundamental idea of YARN is to split up the two major responsibilities of the MapReduce - JobTracker i.e. resource management and job scheduling/monitoring, into separate daemons: a global **ResourceManager** and per-application **ApplicationMaster** (AM).

The ResourceManager and per-node slave, the **NodeManager** (NM), form the new, and generic, system for managing applications in a distributed manner.

The main components of YARN are as described below:

- **ResourceManager** is the ultimate authority that arbitrates resources among all the applications in the system.
- **ApplicationMaster**: The per-application ApplicationMaster is, in effect, a framework specific entity and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the component tasks.
- **NodeManager** is YARN’s per-node agent, and takes care of the individual compute nodes in a Hadoop cluster.

  This includes keeping up-to date with the ResourceManager (RM), overseeing containers’ life-cycle management; monitoring resource usage (memory, CPU) of individual containers, tracking node-health, log’s management and auxiliary services which may be exploited by different YARN applications. .
- **JobHistoryServer** is a daemon that serves historical information about completed applications. Typically, JobHistory server can be co-deployed with JobTracker, but we recommend running JobHistory server as a separate daemon.

2.2. Apache Pig

Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of MapReduce programs. Pig's language layer currently consists of a textual language called Pig Latin, which is easy to use, optimized, and extensible.

2.3. Apache Hive

Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems.

It provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. Hive also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

For more information, see Introducing Tez: Accelerating processing of data stored in HDFS

2.4. Tez

Tez provides a general-purpose, highly customizable framework that creates simplified data-processing tasks across both small scale (low-latency) and large-scale (high throughput) workloads in Hadoop.

Tez is the improved implementation of MapReduce Application that supports container reuse. This allows jobs to run faster on clusters that have limited resources per job. On smaller clusters, it reduces the time for a job to finish by efficiently using containers to run more than one task.

2.5. Apache HCatalog

HCatalog is a metadata abstraction layer for referencing data without using the underlying filenames or formats. It insulates users and scripts from how and where the data is physically stored.

WebHCat provides a REST-like web API for HCatalog and related Hadoop components. Application developers make HTTP requests to access the Hadoop MapReduce, Pig, Hive, and HCatalog DDL from within the applications. Data and code used by WebHCat are maintained in HDFS. HCatalog DDL commands are executed directly when requested. MapReduce, Pig, and Hive jobs are placed in queue by WebHCat and can be monitored for progress or stopped as required. Developers also specify a location in HDFS into which WebHCat should place Pig, Hive, and MapReduce results.

2.6. Apache HBase

HBase (Hadoop DataBase) is a distributed, column oriented database. HBase uses HDFS for the underlying storage. It supports both batch style computations using MapReduce and point queries (random reads).

The main components of HBase are as described below:

- HBase Master negotiates load balancing across all Region Servers and also maintains the state of the cluster. It is not part of the actual data storage or retrieval path.
- RegionServer is deployed on each machine and hosts data and processes I/O requests.

2.7. Apache Zookeeper

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services which are very useful for a variety of distributed systems. HBase is not operational without ZooKeeper.

3. Typical Hadoop Cluster

A typical Hadoop cluster comprises of machines based on the various machine roles (master, slaves, and clients). For more details, see: Machine Roles In A Typical Hadoop Cluster. The following illustration provides information about a typical Hadoop cluster. A smallest cluster can be deployed using a minimum of four machines (for evaluation purpose only). One machine can be used to deploy all the master processes (NameNode, Secondary NameNode, JobTracker, HBase Master), Hive Metastore, WebHCat Server, and ZooKeeper nodes. The other three machines can be used to co-deploy the slave nodes (YARN NodeManagers, DataNodes, and RegionServers).

A minimum of three machines is required for the slave nodes in order to maintain the replication factor of three. For more details, see: Data Replication in Apache Hadoop

3.1. Machine roles in a typical Hadoop cluster

In Hadoop and HBase, the following two types of machines are available:

- Masters (HDFS NameNode, Secondary NameNode, YARN ResourceManagers, and the HBase Master)

  Note

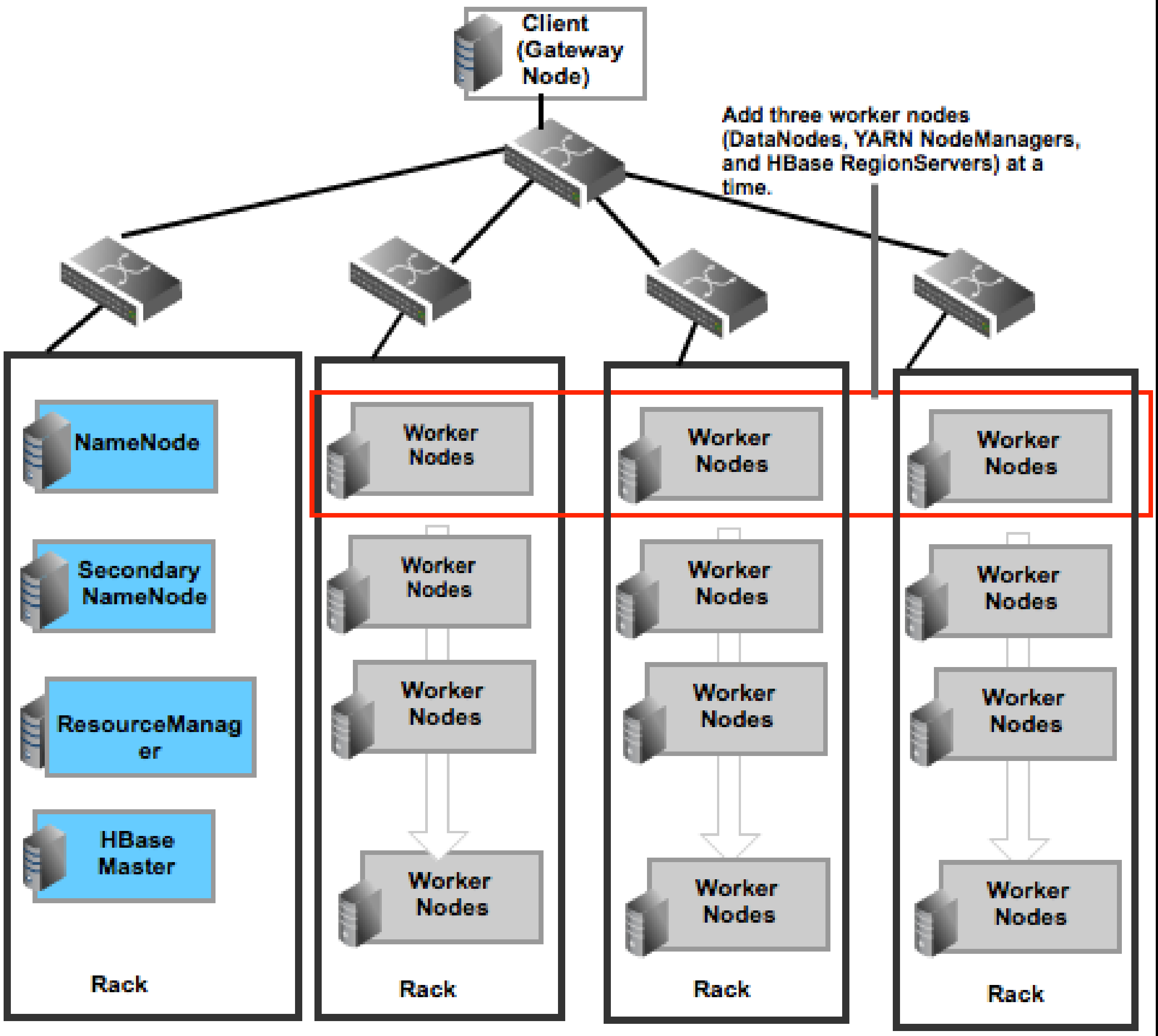
  It is recommended to add only limited number of disks to the master nodes, because the master nodes do not have high storage demands.

- Slaves (HDFS DataNodes, YARN NodeManagers, and HBase RegionServers)

Additionally, we strongly recommend that you use separate client machines for performing the following tasks:

- Load data in the HDFS cluster
- Submit YARN applications(describing how to process the data)
- Retrieve or view the results of the job after its completion
- Submit Pig or Hive queries

Based on the recommended settings for the client machines, the following illustration provides details of a typical Hadoop cluster:



NOTE: DataNodes, NodeManagers, and RegionServers are typically co-deployed.