

HDP Operations: Hadoop Administration 1

Student Guide

Rev 1.1





Copyright © 2012 - 2016 Hortonworks, Inc. All rights reserved.

The contents of this course and all its lessons and related materials, including handouts to audience members, are Copyright © 2012 - 2016 Hortonworks, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Hortonworks, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Hortonworks, Inc. Hortonworks, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

All other trademarks are the property of their respective owners.



Become a **Hortonworks Certified Professional** and establish your credentials:

- HDP Certified Developer: for Hadoop developers using frameworks like Pig, Hive, Sqoop and Flume.
- HDP Certified Administrator: for Hadoop administrators who deploy and manage Hadoop clusters.
- HDP Certified Developer: Java: for Hadoop developers who design, develop and architect Hadoop-based solutions written in the Java programming language.
- HDP Certified Developer: Spark: for Hadoop developers who write and deploy applications for the Spark framework.
- HDF Certified Professional: for DataFlow Operators responsible for building and deploying HDF workflows.

How to Register: Visit www.examslocal.com and search for “Hortonworks” to register for an exam. The cost of each exam is \$250 USD, and you can take the exam anytime, anywhere using your own computer. For more details, including a list of exam objectives and instructions on how to attempt our practice exams, visit <http://hortonworks.com/training/certification/>

Earn Digital Badges: Hortonworks Certified Professionals receive a digital badge for each certification earned. Display your badges proudly on your résumé, LinkedIn profile, email signature, etc.





Self Paced Learning Library

On Demand Learning

Hortonworks University Self-Paced Learning Library is an on-demand dynamic repository of content that is accessed using a Hortonworks University account. Learners can view lessons anywhere, at any time, and complete lessons at their own pace. Lessons can be stopped and started, as needed, and completion is tracked via the Hortonworks University Learning Management System.

Hortonworks University courses are designed and developed by Hadoop experts and provide an immersive and valuable real world experience. In our scenario-based training courses, we offer unmatched depth and expertise. We prepare you to be an expert with highly valued, practical skills and prepare you to successfully complete Hortonworks Technical Certifications.

Target Audience: Hortonworks University Self-Paced Learning Library is designed for those new to Hadoop, as well as architects, developers, analysts, data scientists, and IT decision makers. It is essentially for anyone who desires to learn more about Apache Hadoop and the Hortonworks Data Platform.

Duration: Access to the Hortonworks University Self-Paced Learning Library is provided for a 12-month period per individual named user. The subscription includes access to over 400 hours of learning lessons.

The online library accelerates time to Hadoop competency. In addition, the content is constantly being expanded with new material, on an ongoing basis.

Visit: <http://hortonworks.com/training/class/hortonworks-university-self-paced-learning-library/>

Table of Contents

Big Data, Hadoop, and the Hortonworks Data Platform	1
Lesson Objectives.....	1
What is Apache Hadoop	1
Hadoop Cluster Node Types.....	1
Hadoop Cluster Resource Management.....	2
Apache Software Frameworks	3
Data Management and Operations Frameworks	4
Data Access Frameworks	5
Governance and Integration Frameworks.....	7
Security Frameworks.....	8
HDP 2.3 Platform Versions.....	9
Example Cluster	9
Cluster Management Options.....	11
Management Features	12
Apache Ambari.....	13
Ambari Server Architecture.....	14
Ambari Web UI.....	15
Ambari Views	16
Knowledge Check Questions	18
Knowledge Check Answers.....	19
Summary	20
Managing Ambari Users and Groups	21
Lesson Objectives.....	21
Ambari Users Versus Hadoop Users.....	21
Hadoop Service Users	22
Ambari Users and Groups	23
Ambari User and Group Permissions	24
Managing Users, Groups and Permissions	25
Listing, Modifying, or Creating a User.....	26
Listing, Modifying, or Creating a Group	27
Managing User and Group Permissions.....	29
Knowledge Check Questions	30
Knowledge Check Answers.....	31

Summary	32
Managing Hadoop Services	33
Lesson Objectives.....	33
Core Hadoop Configuration Files.....	33
Configuration Precedence	34
Final Properties.....	35
Other Framework Configuration Files.....	35
Configuration Management Options	36
Ambari Web UI -Service Management Overview	37
Listing Hadoop Services	37
Service Quick Links	38
Maintenance Mode.....	40
Managing Hadoop Configuration Properties with Ambari	41
Viewing or Changing Service Configurations.....	41
Ambari Configuration Groups	46
Client Configuration Files	49
REST APIs.....	50
Manually Editing Configuration Files	51
Starting Services using the CLI.....	53
Stopping Services using the CLI	54
Knowledge Check Questions	55
Knowledge Check Answers.....	56
Summary	57
Using HDFS Storage.....	59
Lesson Objetives.....	59
Hadoop Distributed File System (HDFS).....	59
HDFS Characteristics	60
Accessing HDFS	61
NameNode and DataNodes Introduction	62
File and Directory Attributes	63
HDFS Permissions.....	64
HDFS Home Directories	65
HDFS Superuser.....	65
HDFS Shell Operations	66

Creating and Viewing Directories	66
Home and Current Working Directory	67
Creating Files.....	67
Overriding HDFS Default Properties.....	70
Ambari Files View	72
Browse HDFS Using the NameNode UI	75
Using WebHDFS	77
WebHDFS Features	77
WebHDFS Enabled by Default.....	78
WebHDFS Operations	79
WebHDFS Authentication	80
WebHDFS and HttpFS.....	80
Using HDFS Access Control Lists (ACLs).....	81
Enabling HDFS ACLs.....	82
Managing ACLs	83
Listing Default Directory ACL Entries	85
Creating a File in a Directory with a Default ACL.....	86
Creating a Directory in a Directory with a Default ACL	86
ACL Mask Types.....	87
Knowledge Check Questions	90
Knowledge Check Answers.....	92
Summary	94
Managing HDFS Storage.....	95
Lesson Objectives.....	95
HDFS Architecture and Operation.....	95
Writing to HDFS Storage.....	96
NameNode Startup.....	102
Reading Data	104
Managing HDFS Using UIs	109
Ambari HDFS Service Managemet	109
The NameNode UI	112
DataNode UI	113
Managing HDFS Manually and Using Command-Line Tools.....	114
Command Line Tools	114
<code>hdfs dfs</code>	114

HDFS fsck	115
dfsadmin	119
Manual Configuration.....	120
HDFS Quotas.....	121
Limiting Specific Users and Applications.....	122
Configuring Quotas	122
Viewing Quota Information.....	123
Knowledge Check Questions	124
Knowledge Check Answers.....	126
Summary	128
YARN Resource Management	129
Lesson Objectives.....	129
YARN Resource Management.....	129
YARN Architecture and Operation	130
YARN NodeManager	131
YARN ResourceManager	138
YARN Component Summary.....	139
YARN Management Options.....	139
Ambari UI	140
ResourceManager UI.....	149
Command Line	152
Manual Configure File Editing.....	155
REST API.....	155
YARN Component Failure Management.....	156
Monitoring.....	157
Configure Failure Monitoring	157
YARN Component Failure Recovery	160
YARN Work-Preserving Restarts.....	160
ResourceManager	161
NodeManager	161
Default Configuration	161
YARN Log Aggregation.....	162
Default Configuration	162
Knowledge Check Questions	163
Knowledge Check Answers.....	164

Summary	166
YARN Applications	167
Lesson Objectives.....	167
YARN Applications.....	167
Traditional MapReduce Applications	168
Apache Pig and Apache Hive.....	168
Apache Pig.....	168
Apache Hive.....	169
Summary	170
Adding, Deleting, and Replacing Worker Nodes.....	171
Lesson Objectives.....	171
Working with Cluster Nodes.....	171
Recommendations When Adding or Replacing Nodes	172
Adding a Worker Node Using the Ambari Web UI.....	172
Select Nodes for Ambari Agents	173
Assign Slaves and Clients.....	175
Choose Configuration Groups	176
Review and Deploy.....	177
Install, Start, and Test	178
The HDFS Balancer.....	178
Running the Balancer Using Ambari.....	179
Running the Balancer Using CLI.....	179
Monitoring DataNode Balance.....	180
Decommissioning and Re-commissioning a Worker Node	181
Turning on Maintenance Mode	182
Decommissioning	182
Performing Maintenance.....	184
Re-Commissioning	184
Rebalancing HDFS	185
Deleting Worker Nodes.....	185
Turning On Maintenance Mode	186
Decommissioning	187
Stopping All Components	188
Stopping the Ambari Agent.....	188
Deleting the Node.....	188

Manual Decommissioning and Recommissioning	189
Moving a Master Component.....	190
Knowledge Check Questions	191
Knowledge Check Answers.....	192
Summary	193
The YARN Capacity Scheduler	195
Lesson Objectives.....	195
Capacity Scheduler Operation	195
Scenario	195
Resource Allocation	199
Queues.....	204
Potential SLA Problem	205
Examining Queues.....	205
Queue Characteristics.....	212
Queue Access Control	227
Queue ACLs.....	228
Default Queue Mapping	231
ACLs vs. Default Queue Mappings.....	233
User Limits	233
Queue Administrators	238
Knowledge Check Questions	240
Knowledge Check Answers.....	241
Summary	243
Configuring Rack Awareness.....	245
Lesson Objectives.....	245
Why Rack Awareness	245
Benefits of Rack Awareness	245
Configuring Rack Awareness	246
The Set Rack Window	247
Checking for Rack Awareness.....	248
Viewing Rack Names.....	249
Knowledge Check Questions	250
Knowledge Check Answers.....	251
Summary	252

Configuring HDFS and YARN High Availability (HA)	253
Lesson Objectives.....	253
NameNode HA.....	253
NameNode HA Components.....	254
Automatic Failover Components	255
Recoverability Versus Availability	257
ZooKeeper Prerequisites	258
Configuring NameNode HA Using Ambari	258
Configure the Nameservice ID	259
Select Host Locations	260
Review the Configuration.....	261
Required Manual Steps for NameNode.....	261
Required Manual Steps for JournalNodes	263
Required Manual Steps for NameNode HA Metadata.....	264
Verifying the Configuration.....	265
Viewing Status Using the NameNode UI	265
Initiating a Failover	266
ResourceManager HA.....	266
ResourceManager Components.....	267
Working with the State Store	269
Configuring ResourceManager HA	270
Enabling RersourceManager HA	270
Select Host Location	271
Review the Configuration.....	272
Verify the Configuration.....	273
Initiating a Failover	273
Knowledge Check Questions	275
Knowledge Check Answers	276
Summary	277
Monitoring a Cluster	279
Lesson Objectives.....	279
Ambari Metrics	279
Ambari Metrics Components	280
Switching to Distributed Mode	280
Ambari Dashboard	281

Types of Widgets.....	282
Ambari Alerts.....	283
Alert Definitions and Alert Instances	284
Alert Types.....	285
Viewing Alert Definitions	286
Alerts, Alert Groups, and Notifications.....	290
Adding and Configuring an Ambari Alert	290
Creating a New Alert Group	291
Creating a New Notification	294
Add a Notification to an Alert Group	297
Knowledge Check Questions	298
Knowledge Check Answers.....	299
Summary	300
Protecting a Cluster with Backups	301
Lesson Objectives.....	301
Hadoop Backups	301
What to Back Up	302
Backup Considerations.....	303
Using HDFS Snapshots	304
HDFS Snapshot Operation.....	304
Snapshots Administrator Tasks.....	305
Snapshots User Tasks	305
Viewing Snapshots in NameNode UI.....	306
Using DistCp	306
Copying with DistCp.....	306
Updating and Overwriting	308
Static Mode.....	308
Dynamic Mode.....	309
DistCp Recommendations	310
Using Snapshots and DistCp Together.....	310
Restoring from a Backup	311
Knowledge Check Questions	312
Knowledge Check Answers.....	313
Summary	314

Installing the Hortonworks Data Platform	315
Lesson Objectives.....	315
Hadoop Deployment Options	315
Planning a Hadoop Cluster Deployment.....	316
Consider Processing Workload Type	317
Hardware Sizing	319
Supported Operating Systems	322
Required Software Packages	322
Supported Databases	324
The Ambari Installation Process.....	324
Ambari Interactive Installation Overview.....	325
Software Repositories	326
Ambari Stacks	328
Installing Ambari	328
Ambari Web UI Log In	329
Naming Your Cluster	330
Selecting the Stack Version	331
Choose Nodes for Ambari Agents	333
Install and Register Ambari Agents	334
Choose Services to Install	334
Configuration Warnings.....	337
Review and Deploy.....	338
Install, Start, and Test	338
The Ambari Dashboard	339
Knowledge Check Questions	340
Knowledge Check Answers.....	341
Summary	343

Big Data, Hadoop, and the Hortonworks Data Platform

Lesson Objectives

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe Apache Hadoop
- ✓ Summarize the purpose of the Hortonworks Data Platform software frameworks
- ✓ List Hadoop cluster management choices
- ✓ Describe Apache Ambari

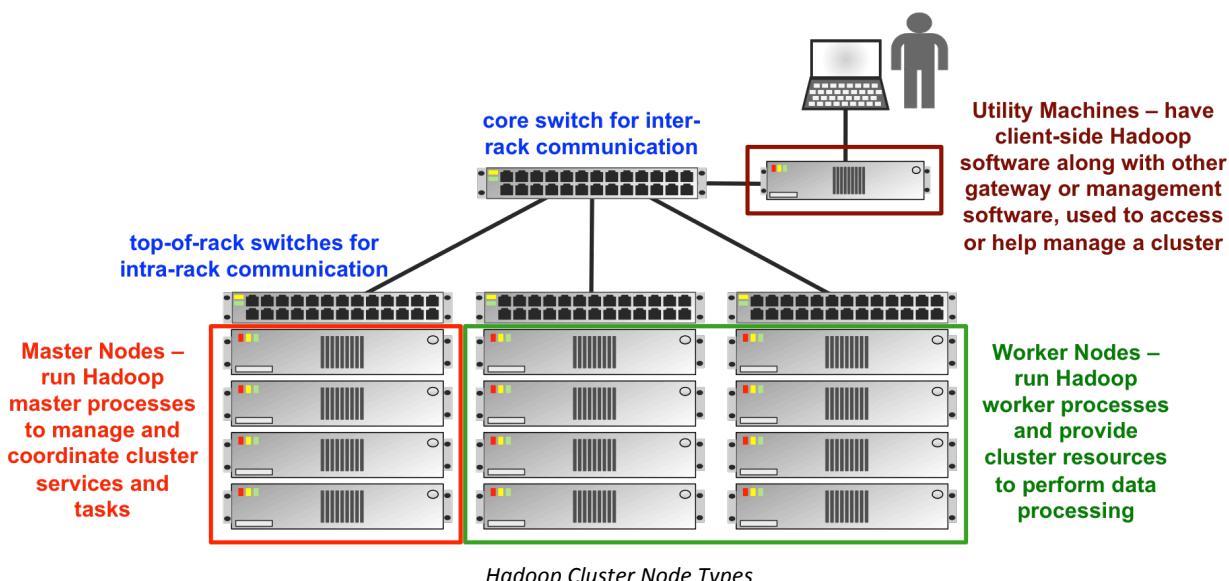
What is Apache Hadoop

Hadoop is a collection of open source software frameworks for the distributed storing and processing of large sets of data. Hadoop development is a community effort governed under the licensing of the Apache Software Foundation. Anyone can help to improve Hadoop by adding features, fixing software bugs, or improving performance and scalability.

Hadoop clusters are scalable, ranging from as few as one machine to literally thousands of machines. It is also fault tolerant. Hadoop services achieve fault tolerance through redundancy. Clusters are created using commodity, enterprise-grade hardware, which not only reduces the original purchase price, but potentially also reduces support costs too.

Hadoop also uses distributed storage and processing to achieve massive scalability. Large datasets are automatically split into smaller chunks, called blocks, and distributed across the cluster machines. Not only that, but each machine commonly processes its local block of data. This means that processing is distributed too, potentially across hundreds of CPUs and hundreds of gigabytes of memory.

Hadoop Cluster Node Types



This is an illustration of a Hadoop cluster. A Hadoop cluster comprises several components; the primary ones being the master and worker nodes. Worker nodes are sometimes called slave nodes.

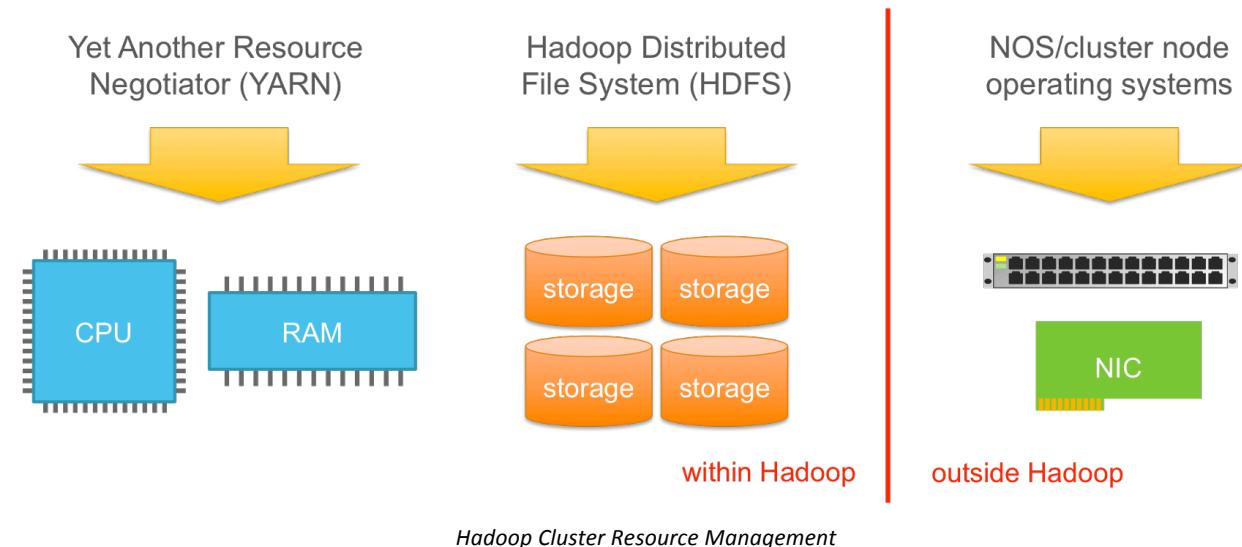
Master nodes manage and coordinate cluster services and tasks. They are master nodes because they have various Hadoop master processes running on them. For example, a master node runs the NameNode process that coordinates Hadoop storage operations. A single master machine can run all of the Hadoop master processes. However for better scalability, and higher availability, it is common to have the various Hadoop master processes spread across multiple master nodes.

Worker nodes provide the CPU, memory, and local disk resources to store and process data. They are worker nodes because they have various Hadoop worker processes running on them. For example, a worker node runs a DataNode process that works under the management of the NameNode. The DataNode does the actual work of reading and writing data blocks to storage. A Hadoop cluster is easily scaled up by adding additional worker machines.

A **utility machine** has the Hadoop client-side software installed on it. Client software is used to submit data processing jobs to the cluster. For the machines outside of the cluster, this machine can function as a gateway machine to the cluster. The utility machine might also have dedicated network access or cluster management software installed on it. For example, Apache Knox or Apache Ambari could be installed on a utility machine. Knox and Ambari are described in more detail later.

It is common for Hadoop to be deployed on rack-based servers. Many server rack and blade chassis configurations are possible. In the illustration, each rack of machines has a top-of-the-rack network switch for intra-rack communication. Each top-of-the-rack switch is also connected to a core switch used for inter-rack and client-to-cluster communications.

Hadoop Cluster Resource Management



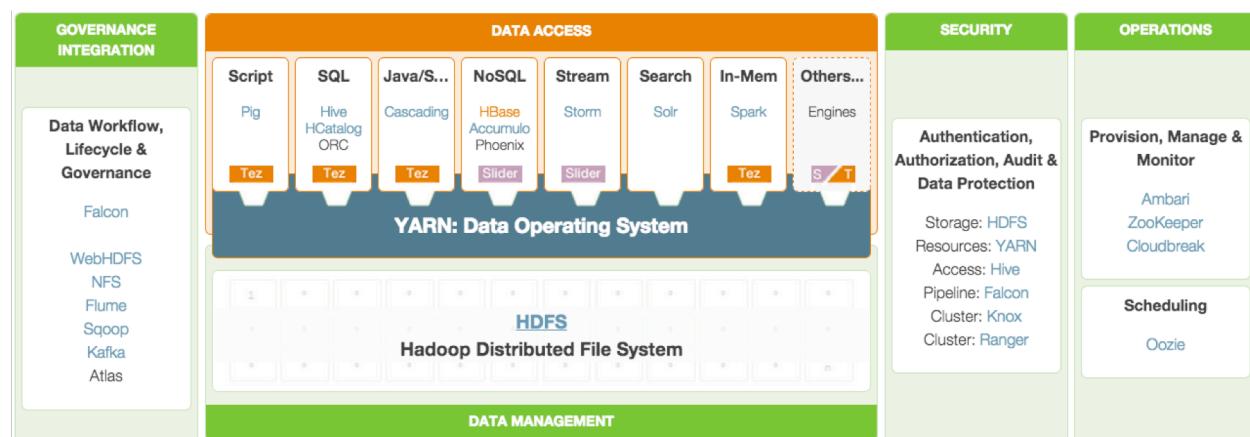
Worker nodes provide CPU, memory, storage, and network resources to a Hadoop cluster. These resources are used to process data.

The YARN (Yet Another Resource Negotiator) service manages worker node CPU and memory resources. Other lessons provide more detail about the architecture and operation of the YARN service.

The Hadoop Distributed File System (HDFS) manages cluster storage resources. Other lessons provide more detail about the architecture and operation of HDFS.

Worker node network resources are managed by the underlying operating system. Network resources are also managed at the switch level by the network operating system (NOS).

Apache Software Frameworks



Hadoop is a Collection of Frameworks

Hadoop is not a monolithic piece of software. It is a collection of software frameworks. Most of the frameworks are part of the Apache software ecosystem. The picture illustrates the Apache frameworks that are part of the Hortonworks Hadoop distribution.

So why does Hadoop have so many frameworks and tools? The reason is that each tool is designed for a specific purpose. The functionality of some tools overlap but typically one tool is going to be better than others when performing certain tasks.

For example, both Apache Storm and Apache Flume ingest data and perform real-time analysis. But Storm has more functionality and is more powerful for real-time data analysis.

NOTE

The following brief descriptions are provided for quick convenience. More detailed descriptions are available online or in other lessons and courses.

Data Management and Operations Frameworks

Framework	Description
Hadoop Distributed File System (HDFS)	A Java-based, distributed file system that provides scalable, reliable, high-throughput access to application data stored across commodity servers
Yet Another Resource Negotiator (YARN)	A framework for cluster resource management and job scheduling

Framework	Description
Ambari	A Web-based framework for provisioning, managing, and monitoring Hadoop clusters
ZooKeeper	A high-performance coordination service for distributed applications
Cloudbreak	A tool for provisioning and managing Hadoop clusters in the cloud
Oozie	A server-based workflow engine used to execute Hadoop jobs

Data Management and Operations Frameworks

Data Management Frameworks

There are two data management frameworks: HDFS and YARN.

HDFS

HDFS is a Java-based distributed file system that provides scalable, reliable, high-throughput access to application data stored across commodity servers. HDFS is similar to many conventional file systems. For example, it shares many similarities to the Linux file system. HDFS supports operations to read, write, and delete files. It supports operations to create, list, and delete directories. HDFS is described in more detail in another lesson.

YARN

YARN is a framework for cluster resource management and job scheduling. YARN is the architectural center of Hadoop that enables multiple data processing engines such as interactive SQL, real-time streaming, data science, and batch processing to co-exist on a single cluster. YARN is described in more detail in another lesson.

Operations Frameworks

The four operations frameworks are: Apache Ambari, Apache ZooKeeper, Cloudbreak, and Apache Oozie.

Ambari

Ambari is a completely open operational framework for provisioning, managing, and monitoring Hadoop clusters. It includes an intuitive collection of operator tools and a set of RESTful APIs that mask the complexity of Hadoop, simplifying the operation of clusters. The most visible Ambari component is the Ambari Web UI, a Web-based interface used to provision, manage, and monitor Hadoop clusters. The Ambari Web UI is the “face” of Hadoop management.

Zookeeper

Zookeeper is a coordination service for distributed applications and services. Coordination services are hard to build correctly, and are especially prone to errors such as race conditions and deadlock. In addition, a distributed system must be able to conduct coordinated operations while dealing with such things as scalability concerns, security concerns, consistency issues, network outages, bandwidth limitations, and synchronization issues. ZooKeeper is designed to help with these issues.

Cloudbreak

Cloudbreak is a cloud agnostic tool for provisioning, managing, and monitoring of on-demand clusters. It automates the launching of elastic Hadoop clusters with policy-based autoscaling on the major cloud infrastructure platforms including Microsoft Azure, Amazon Web Services, Google Cloud Platform, OpenStack, and Docker containers.

Oozie

Oozie is a server-based workflow engine used to execute Hadoop jobs. Oozie enables Hadoop users to build and schedule complex data transformations by combining MapReduce, Apache Hive, Apache Pig, and Apache Sqoop jobs into a single, logical unit of work. Oozie can also perform Java, Linux shell, distcp, SSH, email, and other operations.

Data Access Frameworks

Framework	Description
Pig	A high-level platform for extracting, transforming, or analyzing large datasets
Hive	A data warehouse infrastructure that supports ad hoc SQL queries
HCatalog	A table information, schema, and metadata management layer supporting Hive, Pig, MapReduce, and Tez processing
Cascading	An application development framework for building data applications, abstracting the details of complex MapReduce programming
HBase	A scalable, distributed NoSQL database that supports structured data storage for large tables
Phoenix	A client-side SQL layer over HBase that provides low-latency access to HBase data
Accumulo	A low-latency, large table data storage and retrieval system with cell-level security
Storm	A distributed computation system for processing continuous streams of real-time data
Solr	A distributed search platform capable of indexing petabytes of data
Spark	A fast, general purpose processing engine use to build and run sophisticated SQL, streaming, machine learning, or graphics applications.

Data Access Frameworks

Apache Pig is a high-level platform for extracting, transforming, or analyzing large datasets. Pig includes a scripted, procedural-based language that excels at building data pipelines to aggregate and add structure to data. Pig also provides data analysts with tools to analyze data.

Apache Hive is a data warehouse infrastructure built on top of Hadoop. It was designed to enable users with database experience to analyze data using familiar SQL-based statements. Hive includes support for SQL:2011 analytics. Hive and its SQL-based language enable an enterprise to utilize existing SQL skillsets to quickly derive value from a Hadoop deployment.

Apache HCatalog is a table information, schema, and metadata management system for Hive, Pig, MapReduce, and Tez. HCatalog is actually a module in Hive that enables non-Hive tools to access Hive metadata tables. It includes a REST API, named WebHCat, to make table information and metadata available to other vendors' tools.

Cascading is an application development framework for building data applications. Acting as an abstraction layer, Cascading converts applications built on Cascading into MapReduce jobs that run on top of Hadoop.

Apache HBase is a non-relational database. Sometimes a non-relational database is referred to as a NoSQL database. HBase was created for hosting very large tables with billions of rows and millions of columns. HBase provides random, real-time access to data. It adds some transactional capabilities to Hadoop, allowing users to conduct table inserts, updates, scans, and deletes.

Apache Phoenix is a client-side SQL skin over HBase that provides direct, low-latency access to HBase. Entirely written in Java, Phoenix enables querying and managing HBase tables using SQL commands.

Apache Accumulo is a low-latency, large table data storage and retrieval system with cell-level security. Accumulo is based on Google's Bigtable but it runs on YARN.

Apache Storm is a distributed computation system for processing continuous streams of real-time data. Storm augments the batch processing capabilities provided by MapReduce and Tez by adding reliable, real-time data processing capabilities to a Hadoop cluster.

Apache Solr is a distributed search platform capable of indexing petabytes of data. Solr provides user-friendly, interactive search to help businesses find data patterns, relationships, and correlations across petabytes of data. Solr ensures that all employees in an organization, not just the technical ones, can take advantage of the insights that Big Data can provide.

Apache Spark is an open source, general purpose processing engine that allows data scientists to build and run fast and sophisticated applications on Hadoop. Spark provides a set of simple and easy-to-understand programming APIs that are used to build applications at a rapid pace in Scala. The Spark Engine supports a set of high-level tools that support SQL-like queries, streaming data applications, complex analytics such as machine learning, and graph algorithms.

Governance and Integration Frameworks

Framework	Description
Falcon	A data governance tool providing workflow orchestration, data lifecycle management, and data replication services.
WebHDFS	A REST API that uses the standard HTTP verbs to access, operate, and manage HDFS
HDFS NFS Gateway	A gateway that enables access to HDFS as an NFS mounted file system
Flume	A distributed, reliable, and highly-available service that efficiently collects, aggregates, and moves streaming data
Sqoop	A set of tools for importing and exporting data between Hadoop and RDBM systems
Kafka	A fast, scalable, durable, and fault-tolerant publish-subscribe messaging system
Atlas	A scalable and extensible set of core governance services enabling enterprises to meet compliance and data integration requirements

Governance and Integration Frameworks

Apache Falcon is a data governance tool. It provides a workflow orchestration framework designed for data motion, coordination of data pipelines, lifecycle management, and data discovery. Falcon enables data stewards and Hadoop administrators to quickly onboard data and configure its associated processing and management on Hadoop clusters.

WebHDFS uses the standard HTTP verbs GET, PUT, POST, and DELETE to access, operate, and manage HDFS. Using WebHDFS, a user can create, list, and delete directories as well as create, read, append, and delete files. A user can also manage file and directory ownership and permissions. Administrators can manage HDFS.

The **HDFS NFS Gateway** allows access to HDFS as though it were part of an NFS client's local file system. The NFS client mounts the root directory of the HDFS cluster as a volume and then uses local command-line commands, scripts, or file explorer applications to manipulate HDFS files and directories.

Apache Flume is a distributed, reliable, and available service that efficiently collects, aggregates, and moves streaming data. It is a distributed service because it can be deployed across many systems. The benefits of a distributed system include increased scalability and redundancy. It is reliable because its architecture and components are designed to prevent data loss. It is highly-available because it uses redundancy to limit downtime.

Apache Sqoop is a collection of related tools. The primary tools are the import and export tools. Writing your own scripts or MapReduce program to move data between Hadoop and a database or an enterprise data warehouse is an error prone and non-trivial task. Sqoop import and export tools are designed to reliably transfer data between Hadoop and relational databases or enterprise data warehouse systems.

Apache Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system. Kafka is often used in place of traditional message brokers like Java Messaging Service (JMS) or Advance Message Queueing Protocol (AMQP) because of its higher throughput, reliability, and replication.

Apache Atlas is a scalable and extensible set of core foundational governance services that enable and enterprise to meet their compliance requirements within Hadoop and enables integration with the complete enterprise data ecosystem.

Security Frameworks

Framework	Description
HDFS	A storage management service providing file and directory permissions, even more granular file and directory access control lists, and transparent data encryption
YARN	A resource management service with access control lists controlling access to compute resources and YARN administrative functions
Hive	A data warehouse infrastructure service providing granular access controls to table columns and rows
Falcon	A data governance tool providing access control lists that limit who may submit Hadoop jobs
Knox	A gateway providing perimeter security to a Hadoop cluster
Ranger	A centralized security framework offering fine-grained policy controls for HDFS, Hive, HBase, Knox, Storm, Kafka, and Solr

Security Frameworks

HDFS also contributes security features to Hadoop. HDFS include file and directory permissions, access control lists, and transparent data encryption. Access to data and services often depends on having the correct HDFS permissions and encryption keys.

YARN also contributes security features to Hadoop. YARN includes access control lists that control access to cluster memory and CPU resources, along with access to YARN administrative capabilities.

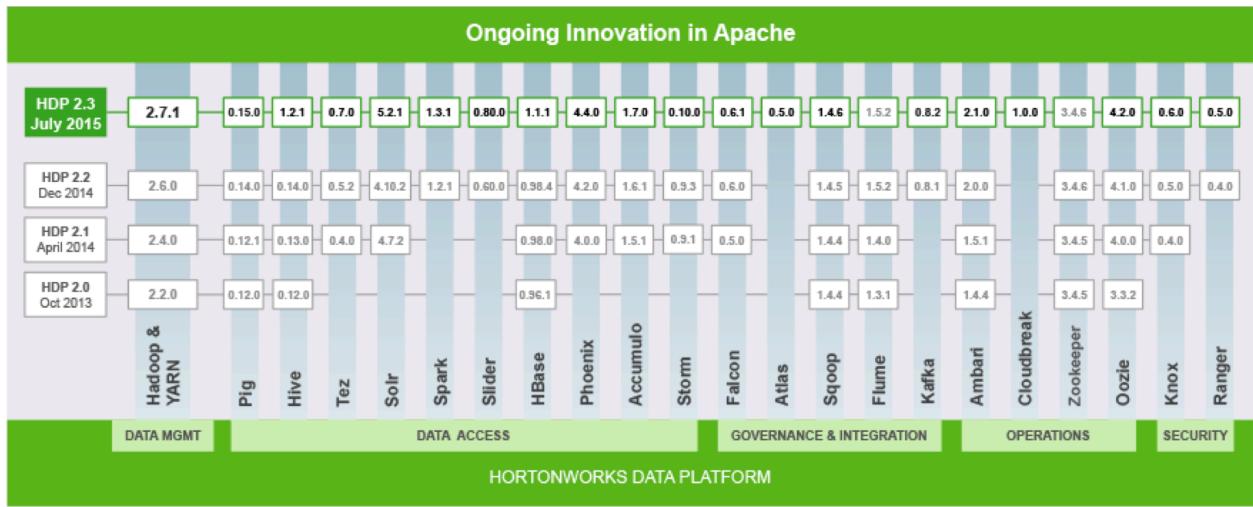
Hive can be configured to control access to table columns and rows.

Falcon is a data governance tool that also includes access controls that limit who may submit automated workflow jobs on a Hadoop cluster.

Apache Knox is a perimeter gateway protecting a Hadoop cluster. It provides a single point of authentication into a Hadoop cluster.

Apache Ranger is a centralized security framework offering fine-grained policy controls for HDFS, Hive, HBase, Knox, Storm, Kafka, and Solr. Using the Ranger Console, security administrators can easily manage policies for access to files, directories, databases, tables, and columns. These policies can be set for individual users or groups and then enforced within Hadoop.

HDP 2.3 Platform Versions



HDP 2.3 Platform Versions

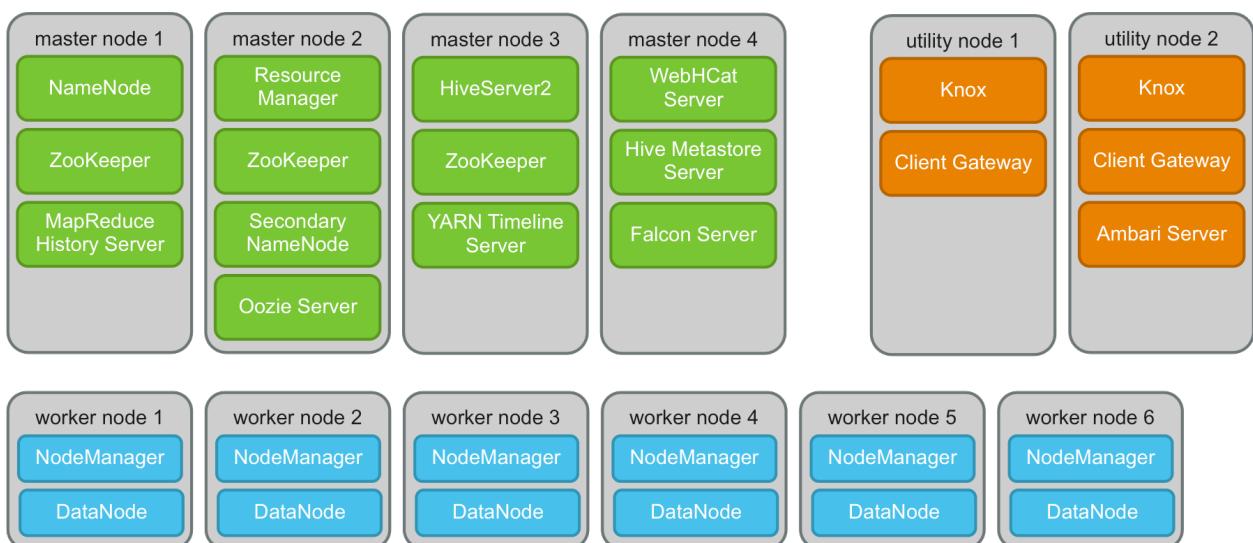
The Hortonworks Data Platform (HDP) is an open enterprise version of Hadoop distributed by Hortonworks. It includes a single installation utility that installs many of the Apache Hadoop software frameworks. Even the installer is pure Hadoop.

Benefits of HDP 2.3

The primary benefit is that Hortonworks has put HDP through a rigorous set of system, functional, and regression tests to ensure that versions of any frameworks included in the distribution work seamlessly together in a secure and reliable manner.

Because HDP is an open enterprise version of Hadoop, it is imperative that it uses the best combination of the most stable, reliable, secure, and current frameworks.

Example Cluster



Cluster Without High Availability

Here is a small example cluster with four master nodes, six worker nodes, and two utility nodes. The cluster is running various services, like YARN and HDFS. Services can be implemented by one or more service components. Service master and worker components illustrated here are:

Master Nodes

The four master nodes are running service master components. The six worker nodes are running service worker components, sometimes called slave components. The two utility nodes are running service components that provide access, security, and management services for the cluster.

Worker Nodes

The worker nodes are running service worker components. For example, the NodeManager is the YARN worker component while a DataNode is an HDFS worker component.

Utility Nodes

The utility nodes are running redundant Knox gateway components that provide perimeter security for the cluster. These nodes also have the Hadoop client software installed so that they can function as cluster gateway machines. For example, a user could use the SSH utility to log in from their machine to a utility node and then run HDFS commands. Utility Node 2 is also running the Ambari Server. The Ambari Server is used as a management platform for the cluster.

Cluster Services

Some cluster services are comprised of multiple master components. For example, the HDFS service has NameNode and Secondary NameNode master components. Some services, like Oozie, have only a single master component. Still other services run multiple master component instances for higher availability. For example, the ZooKeeper service here has three master server components, which increases ZooKeeper's fault tolerance. Other lessons will describe high availability configurations in more detail.

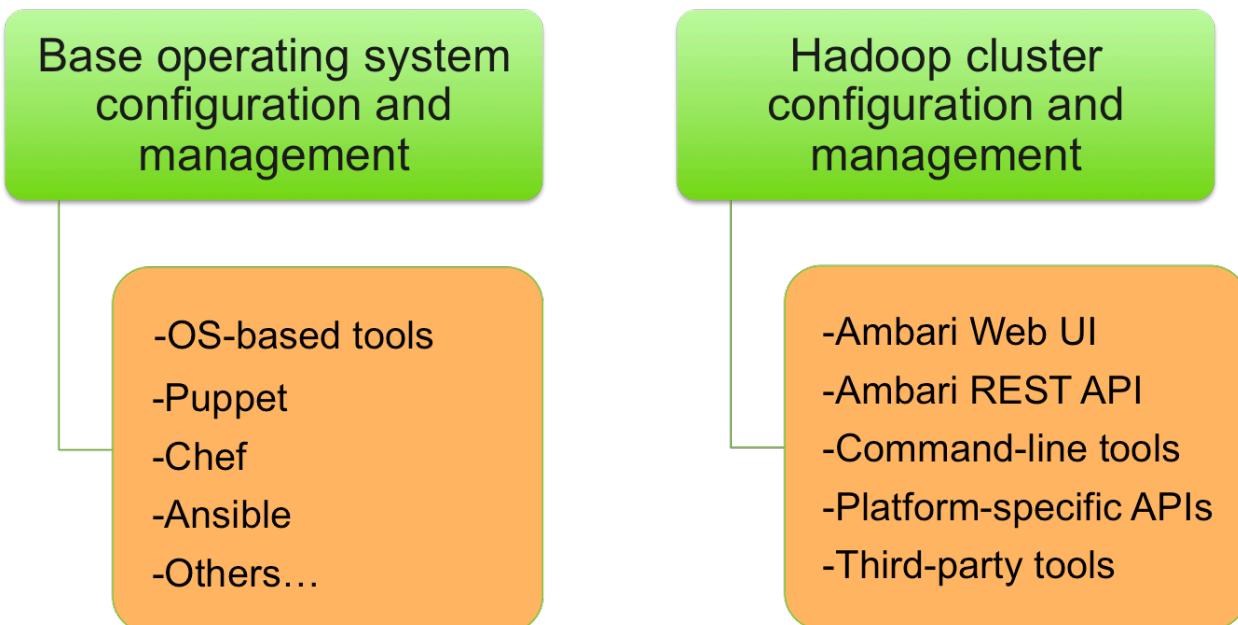
Components listed here include:

- **NameNode** – HDFS service master component
- **Secondary NameNode** – HDFS service master component
- **DataNode** – HDFS service worker component
- **Resource Manager** – YARN service master component
- **YARN Timeline Server** – YARN service master component
- **NodeManager** – YARN service worker component
- **MapReduce History Server** – MapReduce master component
- **ZooKeeper** – ZooKeeper service master component
- **Oozie** – Oozie service master component
- **HiveServer2** – Hive service master component
- **Hive Metastore Server** – Hive service master component
- **WebHCat Server** – Hive service master component
- **Falcon Server** – Falcon service master component
- **Knox** – Knox service master component
- **Ambari Server** – Ambari service master component

NOTE

This illustration does not show all services, service master, or service worker components. More detail is provided in other lessons.

Cluster Management Options



HDP Cluster Management Options

There are two primary facets to Hadoop cluster management:

- Managing the underlying base operating systems
- Managing the actual Hadoop cluster software

Managing Base Operation Systems

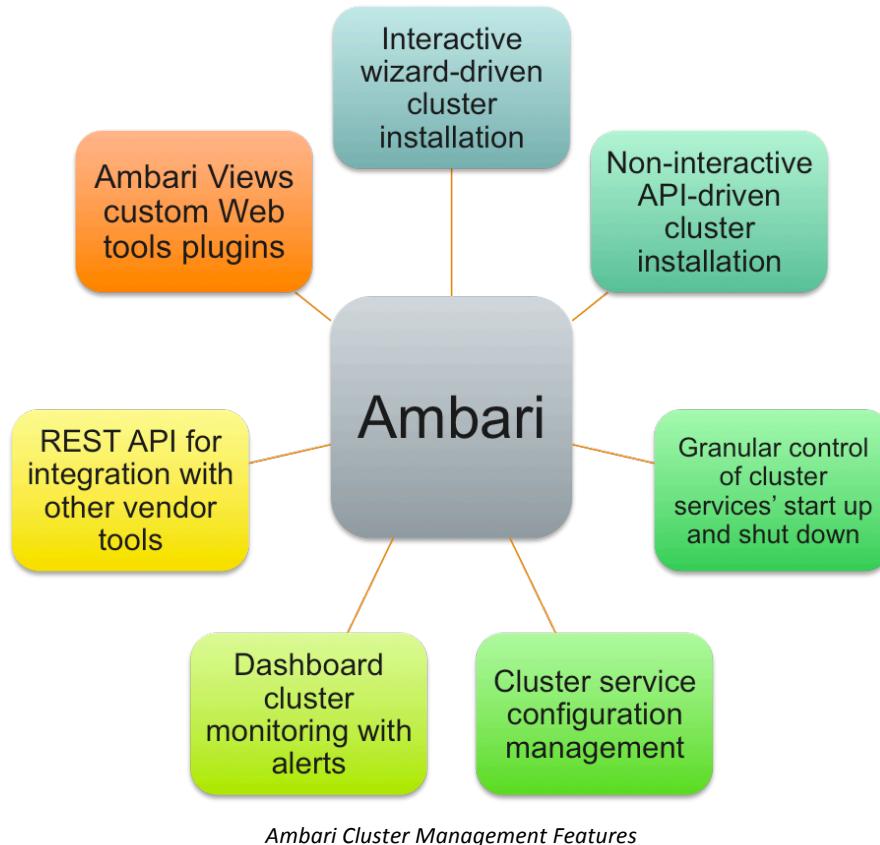
Manage the base operating systems using your favorite tools. These might include OS-based tools, or third-party tools like Puppet, Chef, Ansible, or others. Some tools offer more automation than others and this is important given that Hadoop clusters can span dozens, hundreds, or even thousands of systems.

Managing Hadoop Cluster Software

There are many options available for managing Hadoop cluster software. The Ambari Server features a built-in Web UI and offers a REST API if you choose to build your own Web-based management tools.

Each Hadoop software framework also provides its own management tools and APIs. These tools often include a command-line interface as well as Web-based tools and REST APIs. There are also a number of third-party tools available for cluster management. Once again, automation is important because each Hadoop service might have dozens, hundreds, or thousands of worker components.

Management Features



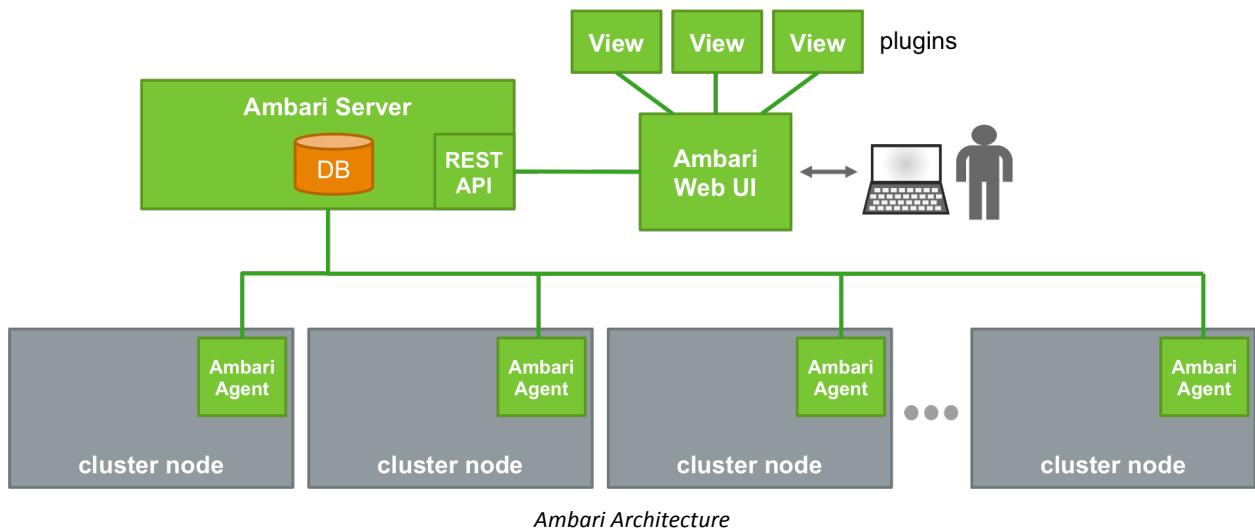
Ambari is the primary management interface in today's Hadoop cluster. Ambari provides a single control point with many important features.

Ambari:

- Enables interactive, wizard-driven installation of Hadoop across any number of hosts. Ambari automatically chooses how to distribute Hadoop services across cluster nodes but these choices can be user-modified prior to software installation.
- Provides a non-interactive, API-driven installation of Hadoop using Ambari Blueprints. Ambari Blueprints enables an administrator to create installation configuration templates that are executed by Ambari. These templates provide an administrator control over the installation process and result in consistent installation across multiple clusters.
- Enables granular control of Hadoop service start up and shut down. Ambari can start up and shut down individual services, groups of services, or all services. Ambari makes changes to service states in the proper sequence when multiple services are selected.
- Provides an administrator with an easy interface to view and update Hadoop service configurations. Once changes are made, Ambari edits the underlying Hadoop configuration files. Ambari also tracks configuration file versions in its database. When adding, deleting, or modifying service parameters, the service configuration windows in the Web-based interface provide on-screen help information. Ambari also alerts an administrator when a service must be restarted to effect a change.

- Offers a web-based interface that includes a dashboard window that enables simplified monitoring of cluster services. Automated service alerts are available in the interface and can also be emailed or sent by SNMP traps to a configurable list of recipients.
- Features advanced Web-based job diagnostics and heat maps to help simplify troubleshooting issues. Heat maps are a visualization tool that graphically depicts resource usage within a Hadoop cluster. This is useful when looking for areas of resource contention.
- Includes RESTful APIs for customization and integration with enterprise systems such as Microsoft System Center and Teradata Viewpoint.
- Includes Ambari Views which offer a systematic way to plug-in Web UI capabilities to enable custom visualization, management, and monitoring tools.

Apache Ambari



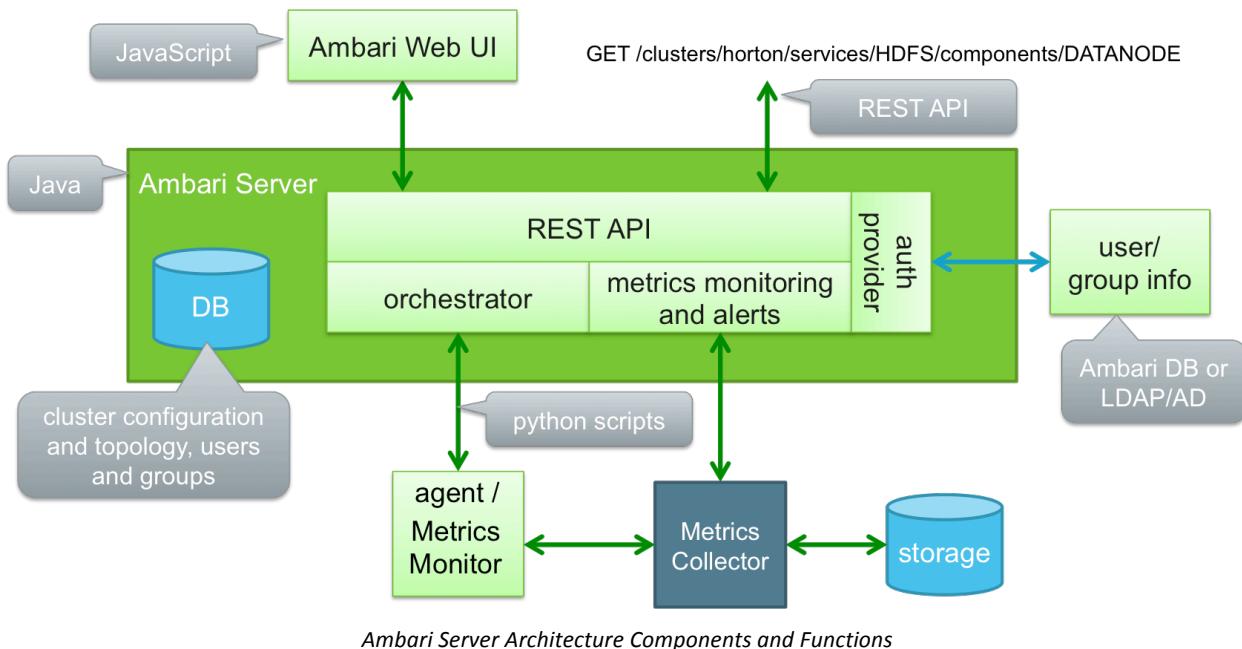
The Ambari Server serves as the collection point for data from across the cluster. The Ambari Server is accessible through a REST API. The Ambari Server relies on a database that maintains cluster topology and configuration information.

Each cluster node has a copy of the Ambari Agent that is either installed automatically by the installation wizard or installed manually by the administrator. The Ambari Agent enables the Ambari Server to monitor and control each cluster node.

The Ambari Web UI is a client-side JavaScript application, which accesses the Ambari Server through the REST API. Browser access to the Ambari Web interface is available at http://<ambari_server_host>:8080/. Access to Ambari is username and password controlled. Ambari also features administrator and non-administrator permissions.

Using the Ambari Views feature, custom Web-based tools can be plugged into Ambari Web. Several View plugins come with HDP, with more being added to new versions of the software.

Ambari Server Architecture



Ambari Server Architecture Components and Functions

The Ambari Server is written in Java. The Ambari Server features a REST API interface, and orchestrator function, a metrics monitoring and alerts function, and an authentication provider function.

The REST API is used by Ambari Web UI and any other program or utility that can issue REST API commands. The Ambari Web UI is written in JavaScript.

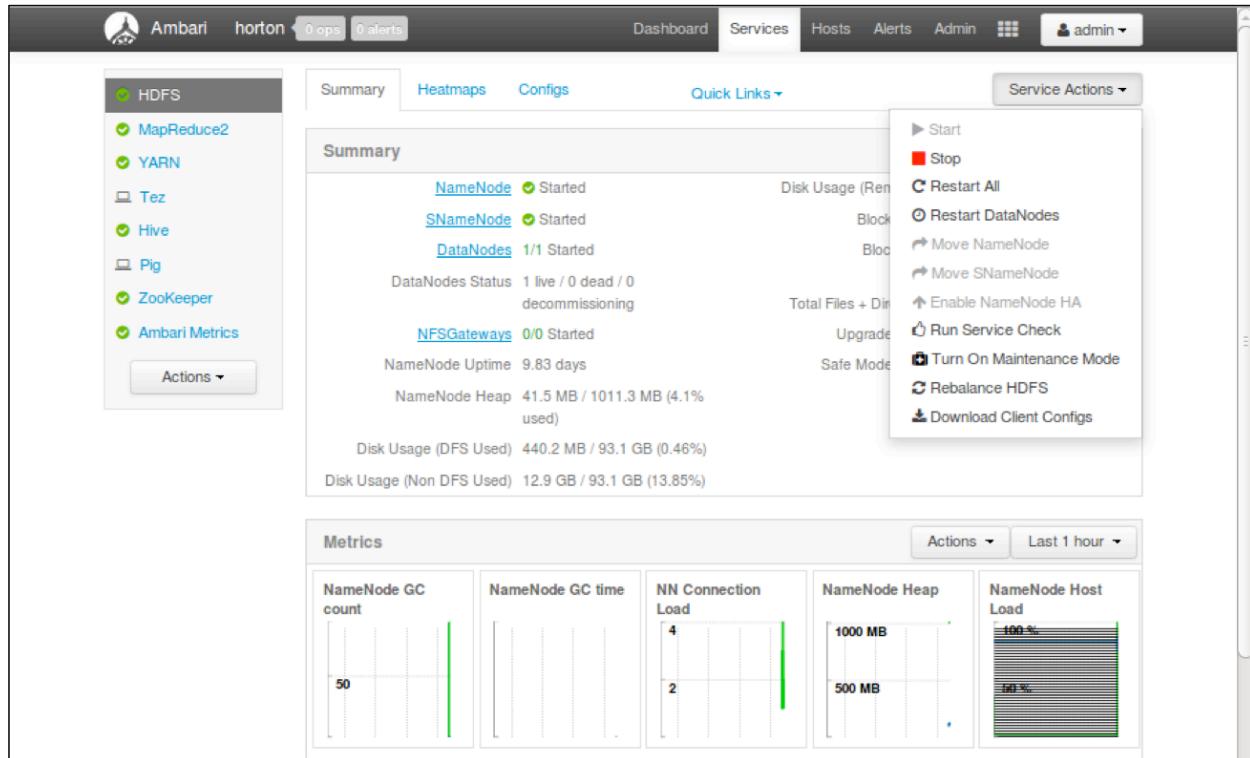
The orchestrator function is the interface to the Ambari agents installed on the cluster nodes. Ambari Server uses Python scripts to issue instructions to Ambari Agents.

Starting with Ambari 2.0, the Ambari Server includes built-in metrics monitoring and alerting functions. These enable an administrator to monitor and troubleshoot cluster issues. These functions remove the dependencies on Ganglia and Nagios that were necessary in earlier versions of Hadoop. A Metric Monitor is installed on each cluster node. It collects information and sends it to a standalone Metrics Collector machine. The Metrics Collector stores data on disk and forwards current metric data to the Ambari Server. The Ambari Server enables metric information to be displayed in the Ambari Web UI.

The authentication provider function enables Ambari Server to access user and group information from multiple sources. By default, Ambari Server maintains user and group information in its own database. Users logging in to Ambari would authenticate to the Ambari Server. However, an administrator may configure Ambari Server to access user and group information from LDAP or Active Directory. In this case, users logging in to Ambari would authenticate to either LDAP or Active Directory.

The Ambari database maintains cluster configuration and topology information, along with user and group information. By default, Ambari uses an internal Derby database. Ambari may be configured to use an external Oracle, MySQL, or PostgreSQL database.

Ambari Web UI



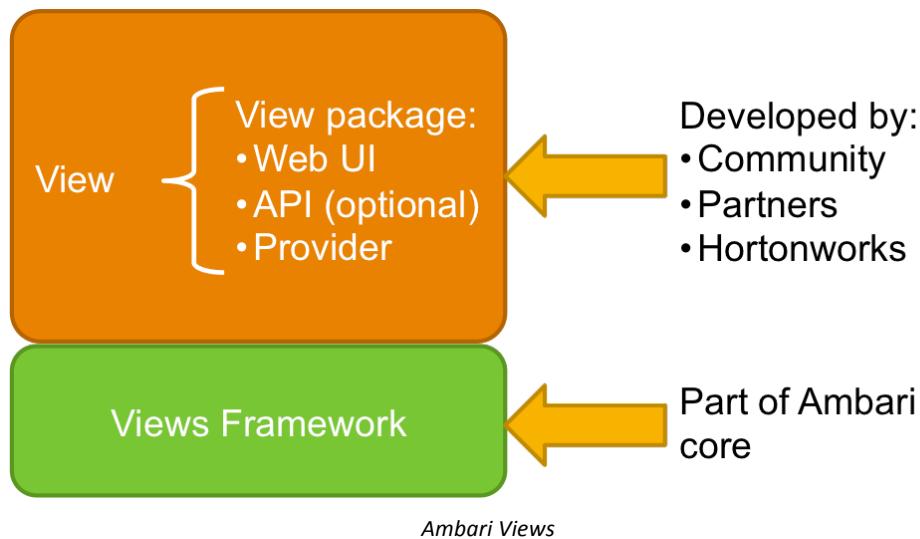
Ambari Web UI Interface

The Ambari Web UI is the primary management interface in a Hadoop cluster. The screen capture here is from the Ambari Web UI. It is included here just to briefly illustrate the graphical management and monitoring capabilities of Ambari.

This screen capture is specifically of one of the Ambari Web UI Services pages. The HDFS service is currently selected and summary and metric information for the HDFS service is displayed. The opened Service Actions menu shows some of the actions Ambari can perform on the HDFS service.

In addition to HDFS, a list of other services installed in the cluster is seen on the left-side panel. Each of these services can be monitored and managed as well.

Ambari Views



Ambari Views are Web applications that can be plugged into Ambari. The development and use of Views enables an organization to extend and customize the Ambari Web UI to meet specific needs.

Just like a typical Web application, a View can include server-side resources and client-side assets. Server-side resources, which are written in Java and sometimes referred to as providers, can integrate with external systems—such as cluster services—and expose REST end-points that are used by the View. Client-side assets—such as HTML/JavaScript/CSS—provide the user interface for the View that is rendered in the Ambari Web UI.

Administrators deploy a View package to the Ambari Server, which enables them to subsequently create specific View instances. A View instance acts like a new Web application running inside the Ambari Web UI. Users must be given access to a View instance by an administrator. View deployment and administration is described in another course.

Views Framework

The Views Framework is separate from Views themselves. The Views Framework is a core feature of Ambari and specific Views build on that Framework. The Framework exists to enable the development, deployment, creation, and management of Views. Ambari in the HDP software distribution includes some out-of-the-box Views. For example, the YARN Capacity Scheduler View is described in a later lesson.

Developers use the Views Framework to develop new Views. Developers create a View package that includes any server and client-side software components along with any optional APIs.

Example Ambari View

Ambari horton 0 ops 5 alerts

Dashboard Services Hosts 2 Alerts Admin

+ Add Queue Actions ▾

root (100%) ✓

default (100%) ✓

Scheduler ✓

Maximum Applications 10000

Maximum AM Resource 20 %

Node Locality Delay 40

Calculator org.apache.hadoop.yarn

Queue Mappings

Queue Mappings Override Disabled

Versions

v1 Current version1 load

Click on a queue to the left for details.

YARN Queue Manager
Tez View

Click to display available Ambari Views

Sample Ambari View

This is an example of an Ambari View instance. This is the YARN Queue Manager View.

Knowledge Check Questions

- 1) What are the three V's commonly used to describe Big Data?
- 2) What are the three Big Data formats?
- 3) List one example of structured data.
- 4) What are the goals of Hadoop?
- 5) Which type of Hadoop cluster nodes provide resources for data processing?
- 6) Which service manages cluster CPU and memory resources?
- 7) Which service manages cluster storage resources?
- 8) Which framework provides a high-performance coordination service for distributed applications?
- 9) Which framework provides provisioning, management, and monitoring capabilities?

Knowledge Check Answers

- 1) What are the three V's commonly used to describe Big Data?

Volume, velocity and variety

- 2) What are the three Big Data formats?

Structured, semi-structured, and unstructured

- 3) List one example of structured data.

A relational database or a data warehouse

- 4) What are the goals of Hadoop?

To leverage inexpensive enterprise-grade hardware to create large clusters, and to create massively scalable clusters through distributed storage and processing

- 5) Which type of Hadoop cluster nodes provide resources for data processing?

Worker nodes

- 6) Which service manages cluster CPU and memory resources?

YARN

- 7) Which service manages cluster storage resources?

HDFS

- 8) Which framework provides a high-performance coordination service for distributed applications?

ZooKeeper

- 9) Which framework provides provisioning, management, and monitoring capabilities?

Ambari

Summary

- **Big Data** is either **structured, semi-structured, or unstructured**.
- **Unstructured data accounts for the largest portion of Big Data** and includes text, audio, video, and other binary files.
- **Apache Hadoop is a scalable, fault tolerant, open source framework** for the distributed storing and processing of large sets of data on commodity hardware.
- **Master nodes** manage and coordinate cluster services and tasks.
- **Worker nodes** provide the CPU, memory, and local disk resources to store and process data.
- The **Hadoop frameworks** can be categorized by their function in data management, data access, operations, security, or governance and integration.
- **Hadoop management interfaces** include Ambari, command-line, and REST API options.
- The **Ambari Web UI** is the primary management interface.

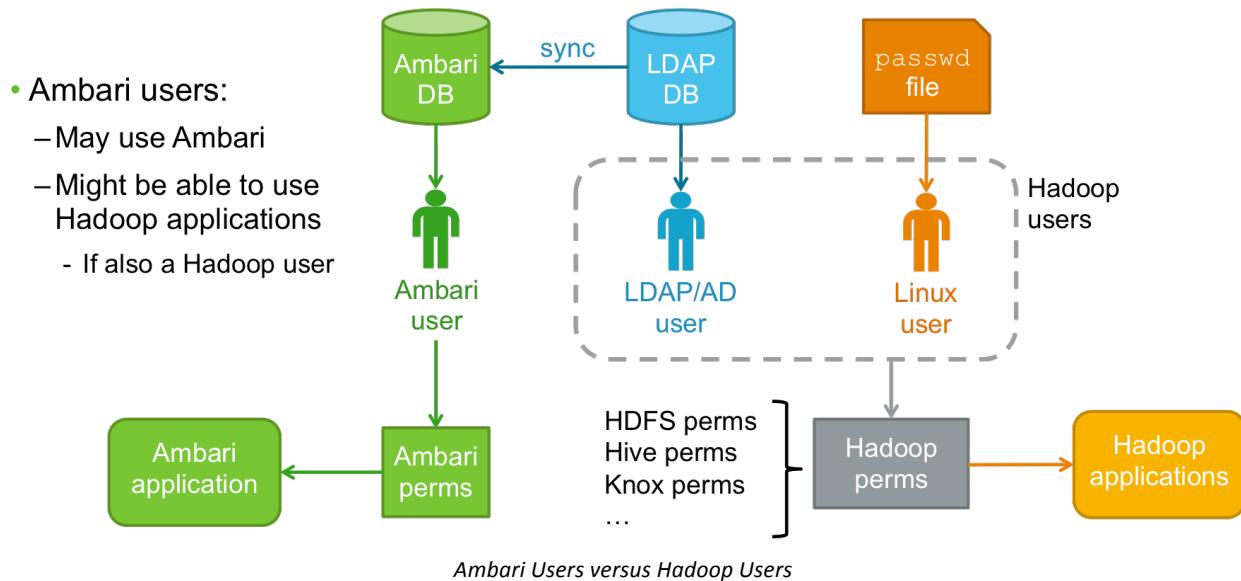
Managing Ambari Users and Groups

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe the differences between Hadoop users, Hadoop service owners, and Ambari users
- ✓ List and describe the two types of Ambari users and groups
- ✓ Manage Ambari users, groups, and permissions

Ambari Users Versus Hadoop Users



Ambari users are defined in the Ambari database. Ambari users may also be imported from an LDAP/AD database. Ambari users may use Ambari. What any specific Ambari user can do in Ambari is determined by their Ambari privileges.

Hadoop users can be defined in an LDAP/AD database or in the Linux /etc/passwd file. Hadoop users may use Hadoop cluster applications. What any specific Hadoop user can do in a cluster depends on their Hadoop permissions. Hadoop permissions are defined in a variety of places. For example, the Hadoop Distributed File System has permissions, Apache Hive defines permissions, Apache Knox defines permissions, and so on. Some permissions are described in other lessons in this course while other permissions are described in other courses.

LDAP/AD users can be imported into Ambari and also used by Linux. For this reason, an LDAP/AD user might be able to use both Ambari and various Hadoop cluster applications. Which actions a user can perform depends on their Ambari and Hadoop permissions.

Hadoop Service Users

Name	Value
Ambari Metrics User	ams
Smoke Test User	ambari-qa
Hadoop Group	hadoop
HDFS User	hdfs
Proxyuser Group	users
HBase User	hbase
HCat Client User	hcat
Hive User	hive
WebHCat User	hcat
MapReduce User	mapred
Oozie User	oozie
Sqoop User	sqoop
Tez User	tez
YARN User	yarn
ZooKeeper User	zookeeper

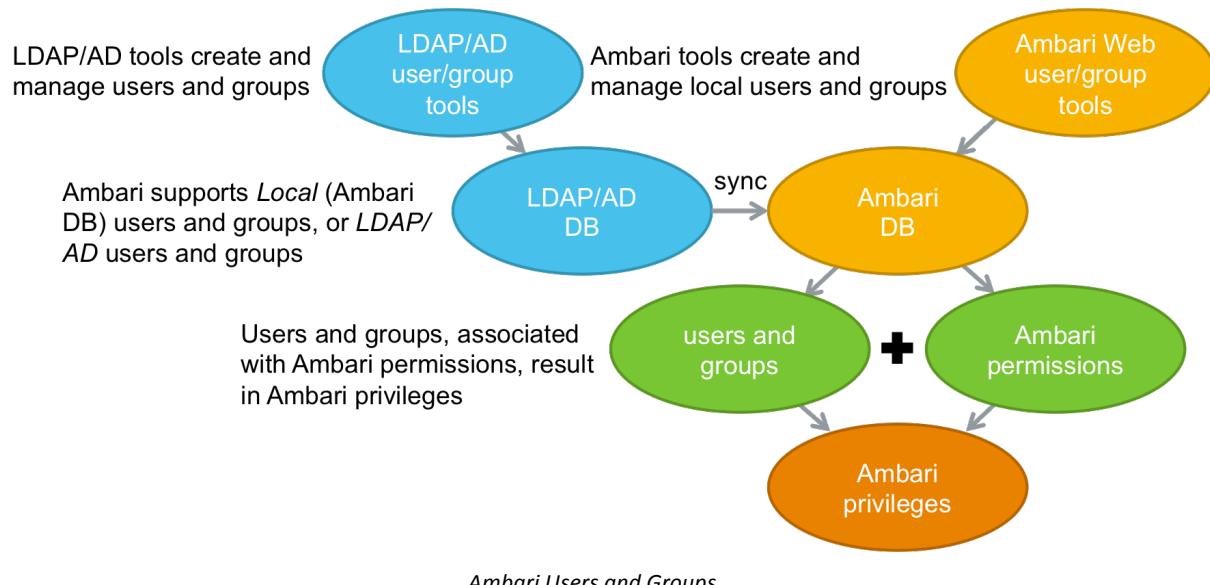
Hadoop Services Users versus Ambari Users

Each Hadoop service is run under the ownership of a corresponding Linux account. These accounts are known as service users. Service users are given special privileges for accessing and managing their service. For example, in another lesson you will see how the HDFS command `hdfs fsck /` fails if run as any user other than the hdfs service user.

In addition, there is a special service user for running smoke tests on components following installation. Smoke tests validate proper service operation. Smoke tests are also available on-demand for each service using **Run Service Check** from the **Service Actions** menu button in the Ambari Web UI.

Service users may be seen in the Ambari Web UI by selecting **Service Accounts** from the **admin** menu button. Service accounts also appear in the operating system files. For example, service accounts appear in the Linux `/etc/passwd` file.

Ambari Users and Groups



Ambari Users and Groups

Ambari supports two types of users and groups: Local and LDAP/AD. To access LDAP/AD users and groups, the Ambari Server must be configured to communicate with an LDAP server or an Active Directory domain. Configuring Ambari to access either LDAP or Active Directory is described in another course or can be found in the online documentation.

Local users and groups are created, managed, and deleted by Ambari administrators and maintained in the Ambari database. LDAP/AD users and groups are created, managed, and deleted by LDAP or Active Directory tools and maintained in an LDAP/AD directory database. LDAP/AD users and groups have only basic account and group membership information stored in the Ambari database. This information is transferred from LDAP/AD when an administrator runs an import and synchronization command.

Local users authenticate to Ambari during log in. LDAP/AD users authenticate to an LDAP or Active Directory Server during log in.

An Ambari administrator assigns Ambari permissions to users and groups. These Ambari permissions determine user and group Ambari privileges. Ambari privileges determine what a user or group is allowed to see or do when using Ambari.

Ambari User and Group Permissions

Ambari Permission Type*	Privileges
No permission (default)	Ambari log in permitted, but no access to information
Read-only	Ambari log in permitted, may view but not modify services and configurations
Operator	Ambari log in permitted, may start, restart, stop, and add new services, may modify or revert configurations
Admin	Ambari log in permitted, has full permissions for services and configurations, may manage user and group accounts and grant permissions

*In addition to these, users and groups may be assigned access to one or more Ambari Views. An Ambari View is a optional, custom “plug-in” tool added to the Ambari Web UI and designed to perform one or more specific tasks.

Ambari User and Group Permissions

Ambari features four levels of user and group permissions. They are no permission, Read-Only permission, Operator permission, and Admin permission. These permissions determine how a user can use Ambari to interact with Hadoop services and configurations. Services are such things as HDFS, YARN, Hive, and so on. Services, for example, can be stopped and started. Configurations affect running services and the cluster topology.

The default for newly created Ambari users and groups is no permissions. These users may log in to Ambari but cannot see any service or cluster information, or perform any actions.

Users or groups with Read-Only permission may log in to Ambari and view, but not modify, service and configuration information.

Those users and groups with Operator permission may log in to Ambari and view service and configuration information. They may also start, restart, and stop existing services as well as add new services. They may also modify current configurations or revert to previous configurations.

Ambari users with Admin permission may do anything. They may log in, view information, and manage services and configurations. In addition, they may also create new users and groups, manage group membership, and assign permissions to users and groups. They may even create other Ambari users with Admin permission. By default during installation, an Ambari admin user account is created and assigned Admin permission.

In addition to the four permission types, users and groups may be assigned access to one or more Ambari Views. An Ambari View is a optional, custom “plug-in” tool added to the Ambari Web UI and designed to perform one or more specific tasks.

Admin Privileges for Local Versus LDAP/AD Users

Ambari User with Admin Permission	Local User/Group	LDAP/AD User/Group
Change passwords	Available	Not Available*
Assign Admin permission	Available	Available
Change group membership	Available	Not Available*
Delete users	Available	Not Available*
Mark users as active or inactive	Available	Available

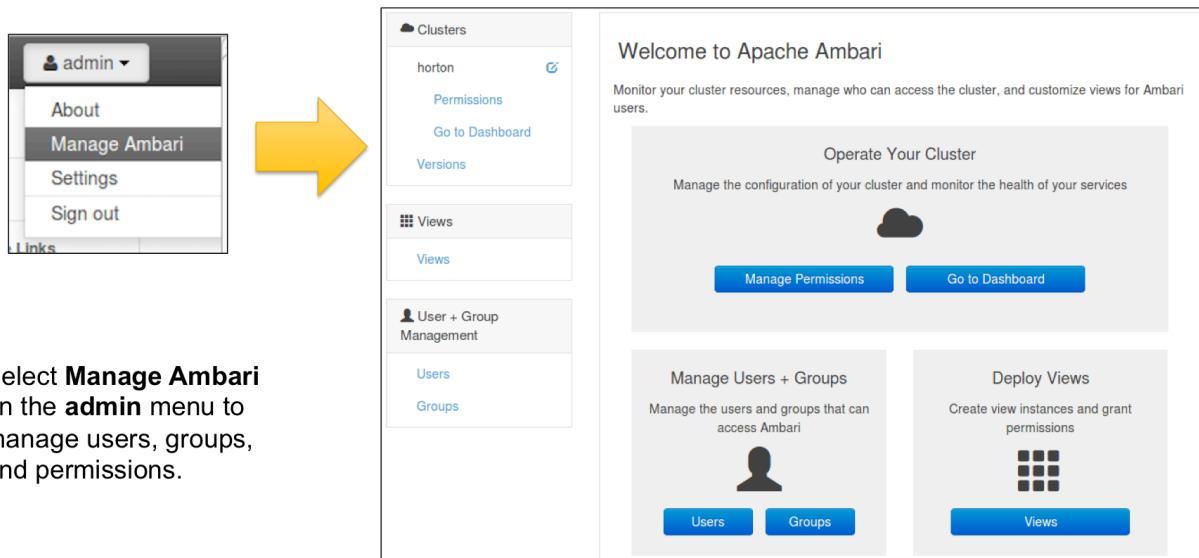
*Must perform the action using LDAP/AD tools or utilities.

Admin Privileges for Local Versus LDAP/AD Users

As an Ambari user with Admin permission, you can create new local users, delete local users, change local user passwords, and edit user settings. You can also control certain privileges for local and LDAP/AD users. However, many user and group management functions are not available to an Ambari administrator for LDAP/AD users and groups. This is because you need an LDAP/AD user or group management tool to manage LDAP/AD users and groups. Ambari can import these users and groups and can assign them Ambari permissions, but Ambari cannot administer the user and groups themselves.

The table shown here lists the privileges available as well as those not available to an Ambari administrator user.

Managing Users, Groups and Permissions



Select **Manage Ambari** on the **admin** menu button to open the browser page to manage users, groups, and permissions.

*Select **Manage Ambari** on the **admin** Menu*

Select **Manage Ambari** on the **admin** menu button to open the browser page to manage users, groups, and permissions.

Click the **Users** button or the **Users** link to manage users. Click the **Groups** button or the **Groups** link to manage groups. Click the **Manage Permissions** button or the **Permissions** link to manage user and group permissions.

Listing, Modifying, or Creating a User

Create a new user

List of existing users. Click any user to modify their settings or to delete the user.

Working with Users

Clicking the **Users** link opens the User page. This page lists any existing users. You may click any existing user to open the page to modify that user's settings. This includes changing the user's password.

Click the **Create Local User** button to create a new local user.

Creating a New User

Creating a Local User

The Create Local User page is used to create a new local user. LDAP/AD is not a choice because LDAP/AD users and groups must be created using LDAP/AD management tools.

Type the new user name.

The new user may be configured as Active or Inactive. Only active users may log in to Ambari. Click **Active** to switch to **Inactive**. Click **Inactive** to switch to **Active**.

The user may also be assigned Ambari Admin permission. The default is No. Click **No** to switch to **Yes**. Click **Yes** to switch to **No**.

Provide a password for the new user. Users may not change their own passwords. Only a user with Admin permissions may change passwords.

Click **Save** to add the new user.

Listing, Modifying, or Creating a Group

The screenshot shows the Ambari User + Group Management interface. On the left, there is a sidebar with the following links:

- Clusters
- horton
- Permissions
- Go to Dashboard
- Versions
- Views
- Views
- User + Group Management
- Users
- Groups** (this link is highlighted with a red rectangle)

On the right, the main area is titled "Groups". It features a search bar with "Group Name: Any", a dropdown for "Type" (set to "All"), and a "Members" column. Below this is a message: "No groups to display." At the bottom right are buttons for "10" (dropdown), "Previous", "1" (highlighted in blue), and "Next". A large callout bubble points to the "Groups" link in the sidebar with the text: "List of existing groups. Click any group to add or remove members or to delete the group." Another callout bubble points to the "+ Create Local Group" button at the top right with the text: "Create a new group".

Working with Groups

Clicking either the Groups button or the Groups link opens the Groups page. This page lists any existing groups. You may click any existing group to open the page to add or remove members or to delete the group.

Click the Create Local Group button to create a new Local group.

Creating a New Group

The screenshot shows the Ambari User Interface for managing groups. On the left, there's a sidebar with three main sections: 'Clusters' (which has 'horton' selected), 'Views', and 'User + Group Management'. Under 'User + Group Management', 'Groups' is selected. The main content area is titled 'Groups / Create Local Group'. It contains a 'Group name' input field with the value 'sales', and two buttons at the bottom right: 'Cancel' and 'Save'.

Creating a Local Group

Use the Create Local Group page to create a new local group. LDAP/AD is not a choice because LDAP/AD users and groups must be created using LDAP/AD management tools.

Type the new group name.

Click **Save** to complete adding the new group.

The new group will appear on the lists of groups displayed in the browser window. To add users as members of the group, click the group name. A browser page opens that enables you to add user names as group members.

Managing User and Group Permissions

horton Permissions

Permission	Grant permission to these users	Grant permission to these groups
Operator	steve	Add Group
Read-Only	janet	sales

Click here to add a the initial name

Cancel, delete and save buttons.

Click the gray area, and then the pencil, to add additional names or delete a name.

Managing User and Group Permissions

Clicking the **Permissions** link opens the cluster's Permissions page. Use this page to assign Operator or Read-Only permissions to any user or group.

Permissions are additive. A user assigned Read-Only permission who belongs to a group with Operator permission will get Operator privileges. More restrictive user-level permissions do not override more generous group-level permissions.

To add the first user or group to a field, click the **Add User** or **Add Group** box. To add a second user or group, click your mouse pointer in the gray area of a box and the pencil icon will be displayed. Click the **pencil** icon to add another user or group name. The **X** buttons are used to cancel an add operation or delete an existing user or group. The **check** button is used as a save button.

Knowledge Check Questions

- 1) List the two types of users supported by Ambari?
- 2) Can Ambari delete LDAP/AD users?
- 3) True or false? A user with Operator privileges can manage user accounts.
- 4) True or false? A non-admin user may change only their own password.
- 5) True or false? The ambari-qa service user is only used during installation to smoke test newly installed services.
- 6) True or false? Ambari updates a user's status from Active to Inactive after 30 days of non-use.
- 7) A user is assigned Read-Only permission but belongs to a group with Operator permission.
What are the user's effective permissions?

Knowledge Check Answers

- 1) List the two types of users supported by Ambari?

Local and LDAP/AD

- 2) Can Ambari delete LDAP/AD users?

No. You must use LDAP/AD-based tools.

- 3) True or false? A user with Operator privileges can manage user accounts.

False. You must have Admin privileges.

- 4) True or false? A non-admin user may change only his or her own password.

False. Only a user with Admin privilege can change user passwords.

- 5) True or false? The ambari-qa service user is only used during installation to smoke test newly installed services.

False. The ambari-qa service user is also used for on-demand service smoke tests.

- 6) True or false? Ambari updates a user's status from Active to Inactive after 30 days of non-use.

False. A user with Admin privilege must manually change a user's status from Active to Inactive.

- 7) A user is assigned Read-Only permission but belongs to a group with Operator permission.

What are the user's effective permissions?

Operator

Summary

- Ambari supports Local users defined in the Ambari database and LDAP/AD users defined in an LDAP or Active Directory directory service.
- Ambari can manage only Local users and groups.
- Local users authenticate to Ambari whereas LDAP/AD users authenticate to LDAP or Active Directory.
- The Ambari permissions model includes no permission, Read-Only permission, Operator permission, and Admin permission.
- User permissions are additive; effective permissions are a combination of the user's permissions combined with any relevant group permissions.
- Service users are given special privileges for accessing and managing their service.

Managing Hadoop Services

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Identify Hadoop configuration files
- ✓ List Hadoop service management options
- ✓ Summarize Operations of the Web UI tool
- ✓ Manage Hadoop service configuration properties using the Ambari Web UI
- ✓ Manage Ambari Configuration Groups
- ✓ Manage Hadoop service configuration properties using the command-line interface

Core Hadoop Configuration Files

File Name	File Format	File Purpose
core-site.xml	Hadoop configuration XML	Hadoop core configuration settings that can be used by HDFS, YARN, MapReduce, and others
hdfs-site.xml	Hadoop configuration XML	HDFS configuration settings (NameNode and DataNode)
yarn-site.xml	Hadoop configuration XML	YARN configuration settings
mapred-site.xml	Hadoop configuration XML	MapReduce configuration settings
hadoop-env.sh	Bash script	Environment variables used by various Hadoop scripts and programs
log4j.properties	Java properties	System log file configuration settings
hadoop-metrics2.properties	Java properties	Metrics publishing configuration settings

Ambari Installs Core Hadoop Configuration Files in /etc/hadoop/conf

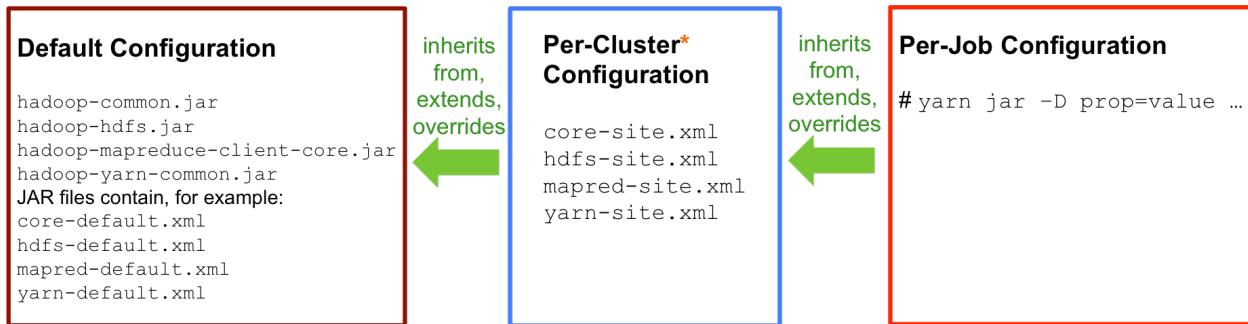
The core of the current Hadoop version consists primarily of HDFS and YARN, although MapReduce has historically been an important service within Hadoop. As a result, the core cluster configuration files define the configuration settings for HDFS, YARN, and MapReduce. These core services are primarily configured using files installed by Ambari in the /etc/hadoop/conf directory.

The core configuration files also define what should be recorded to log files and how to process those log files. Many Hadoop frameworks, including the core frameworks, implement Apache Log4j for their logging system.

Reference

For more information about Log4j, see <https://logging.apache.org/log4j/2.x/>.

Configuration Precedence



*Cluster nodes with different hardware configurations commonly need different *-site.xml files.

A Running Job's Configuration is a Combination of Sources

The actual configuration for any job running on a cluster is derived from a combination of sources. These sources include the default configuration, the per-cluster or per-node configuration, and the per-job configuration.

Hadoop software is installed with a set of default configuration settings. These settings are defined in various `*-default.xml` files that are embedded in various JAR files and are the same for any installation of the same version of Hadoop. These default configuration settings are not changed.

Each Hadoop installation includes a set of per-cluster configuration files that commonly include the word site in their file names. For example, there is a `core-site.xml`, an `hdfs-site.xml`, and others. The per-cluster configuration inherits configuration property settings from the default configuration. For example, any property that is set in the default configuration is used on the cluster even if that property is not found in the per-cluster configuration.

The per-cluster configuration can also extend the default configuration by adding additional configuration properties. The per-cluster configuration setting may also override a default configuration setting. Ambari, either during installation or any time after installation, can be used to modify these files to customize the configuration properties a specific cluster instance.

Some configuration property settings are based on the hardware configuration of master and worker nodes. For example, if all of the worker nodes share a common hardware specification then the same `*-site.xml` files could be used across the entire cluster. However, if there are two groups of worker nodes with different hardware specifications, then you would need two sets of the `*-site.xml` files, one for each group of worker nodes. Ambari can manage this for you using a feature named Configuration Groups, which is described later.

Finally, each job can be submitted with options requesting one or more specific configuration settings. These requests override the per-cluster and default configuration settings. For example, the user submitting a job could use the `-D` option to set a specific configuration property.

Final Properties

- *Final* properties cannot be overridden by user applications.
 - For example, no `-D prop=value`

Using Ambari Web UI

DataNode directories

/hadoop/hdfs/data

Click to toggle on or off (dark gray or light gray)

Editing the Configuration File

```
<property>
    <name>dfs.datanode.data.dir</name>
    <value>/hadoop/hdfs/data</value>
    <final>true</final>
</property>
```

*Using Ambari to view and modify property settings is described in more detail later in this lesson.

Hadoop Configuration Final Properties

User applications may specify their own configuration settings when they are submitted to a cluster. In some cases, a user could choose a configuration setting that unfairly consumes a resource and negatively effects the performance other user applications. To prevent this, an administrator can declare a configuration property value as final. This prevents any user application from overriding a property's value. The Ambari Web UI or a command-line editor can be used to make property settings final.

Other Framework Configuration Files

The other Hadoop frameworks also use files to define their configuration settings. They commonly use configuration files that have similar formats and naming conventions as those used for the core frameworks like HDFS or YARN. These naming conventions include such things as `*-default.xml`, `*-site.xml`, `*-env.sh`, and `*-log4j.properties` and similar derivatives.

The difference is that these configuration files are not typically found in `/etc/hadoop/conf`. Instead, each framework has its own dedicated configuration directory. Examples include:

- `/etc/ambari-server/conf`
- `/etc/ambari-agent/conf`
- `/etc/hive/conf`
- `/etc/pig/conf`
- `/etc/zookeeper/conf`
- and so on...

Configuration Management Options

Option	Description	Benefit
Ambari Web UI	Browser-based graphic user management interface	Ease of use, pre-built and ready to go
REST APIs: Ambari, WebHDFS, YARN, etc.	HTTP verb (GET, PUT, POST, DELETE) management interface	Integration with other web-based management interfaces, can be used for testing and troubleshooting cluster
Manual editing	Manually edit and distribute configuration files, manually restart services	No reliance on graphic user interface, no need to install Ambari, <i>not</i> compatible with Ambari management
Command-line	Per-framework command-line management utilities	Scriptable, no reliance on a graphic user interface

Hadoop Includes Several Options for Configuration Management

There are several available options for configuration management. Hortonworks recommends using either the Ambari Web UI or Ambari REST API.

The Ambari Web UI provides a browser-based graphical user interface for configuration management. The interface is designed not only to automate many configuration tasks, but also to help guide the user through management tasks. As an example of automation, Ambari ensures that any updated configuration files are automatically transferred to the appropriate cluster nodes. As an example of on-screen user guidance, any service or service components that must be restarted as a result of a configuration change are noted in the browser interface. The Web UI can be used to restart entire services or individual service components too.

REST API calls offer the same functionality as the various UI components of HDP. The difference is that the REST APIs enable the integration of Hadoop with other Web-based management platforms. Any management platform that can be extended by adding additional HTTP GET, PUT, POST, and DELETE operations can be used to manage the cluster. The REST APIs enable an organization to develop their own management interface.

An administrator may also manually edit a configuration file, transfer the modified file to the appropriate cluster nodes, and then manually restart the affected service or service components. While this does provide very low-level control, it is also a slower, and more error prone, method of configuration management.

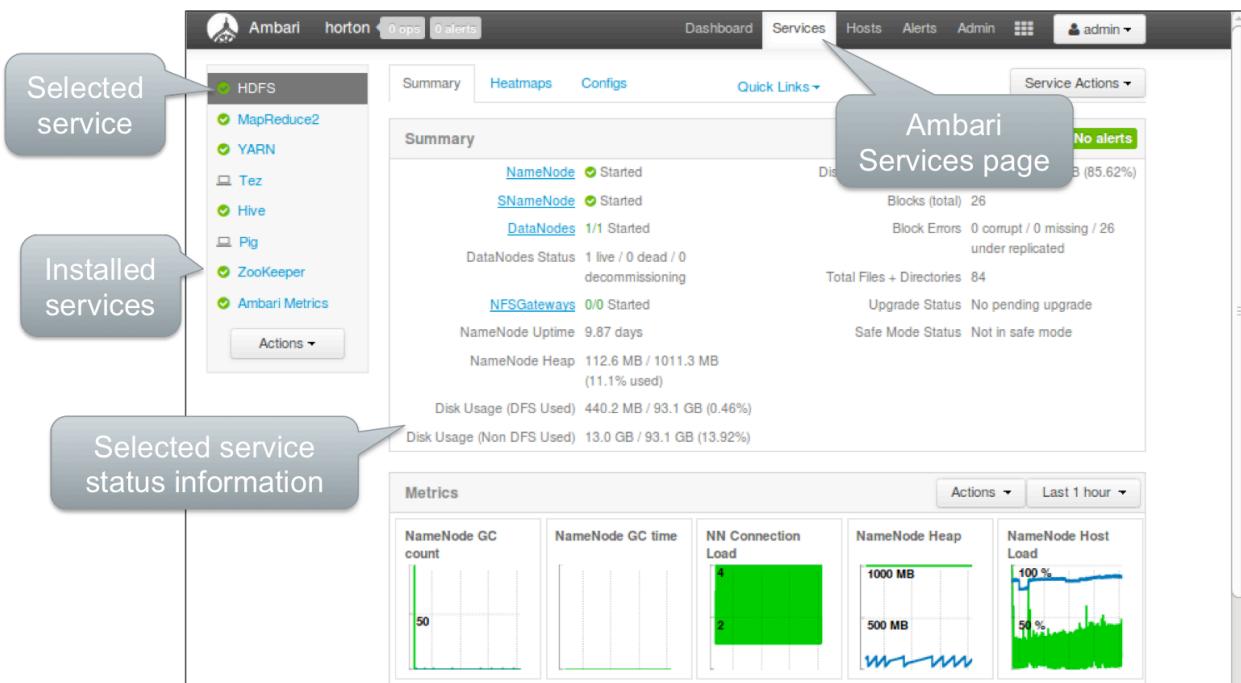
If you are managing your cluster configuration using Ambari, it is not recommended to regularly use other management methods as it will result in split-brain management issues. Ambari not only updates the configuration files as directed by an administrator, but it also makes corresponding updates in its database. If an administrator manually edits a configuration file and restarts services or service components, these changes are not reflected in the Ambari database. The Ambari database will overwrite the configuration text file with the information contained in its database.

Many of the Apache frameworks include their own command-line management interfaces. The command-line utilities can be used to change a service or service component's configuration. These utilities can be added to scripts to automate tasks, and these scripts do not depend on the availability of a graphic user environment. In an Ambari-managed cluster it is recommended to exclusively use Ambari.

Ambari Web UI -Service Management Overview

The Ambari Web UI is the primary tool recommended by Hortonworks for cluster administration. It is used extensively in this course. The majority of this lesson provides an overview of the Ambari Web UI service management features. Future lessons provide details about managing specific cluster services.

Listing Hadoop Services



Listing Hadoop Services

The **Services** page of the Ambari Web UI displays installed service information. The list of installed services always appears in the left-side panel. You may click any service to view status information about that service on its **Summary** tab.

Service Quick Links

Some (not all) services include their own Web UI tools. They can be launched from the service's **Quick Links** menu.

Service Actions

Quick Links ▾

- NameNode UI
- NameNode logs
- NameNode JMX
- Thread Stacks

No alerts

(Remaining) 79.7 GB / 93.1 GB (85.58%)

Blocks (total) 26

Block Errors 0 corrupt / 0 missing / 26 under replicated

Total Files + Directories 84

Upgrade Status No pending upgrade

Safe Mode Status Not in safe mode

DataNodes Status 1 live / 0 dead / 0 decommissioning

NFSGateways 0/0 Started

NameNode Uptime 9.88 days

NameNode Heap 120.6 MB / 1011.3 MB

Actions ▾

Service Quick Links

Several Hadoop services include their own Web UI tools. These tools primarily display service, service component, and service job information. Actual configuration changes are typically managed using Ambari or the services' command-line utilities.

If the service has its own Web UI tools, they commonly can be launched by selecting item from the service's **Quick Links** menu in the Ambari Web UI. If you know the Web UI tool's URL, you can type its URL directly into a browser window. Many Web UI tools will be described and used in later lessons.

Launching Service Actions

Actions menu is where you may start, stop, or restart an entire cluster.

Service Actions menu is context sensitive, therefore different for each service.

Service Actions ▾

- ▶ Start
- Stop
- ⟳ Restart All
- ⟲ Restart DataNodes
- ⤒ Move NameNode
- ⤒ Move SNameNode
- ⤒ Enable NameNode HA
- ⤒ Run Service Check
- ⤒ Turn On Maintenance Mode
- ⤒ Rebalance HDFS
- ⤒ Download Client Configs

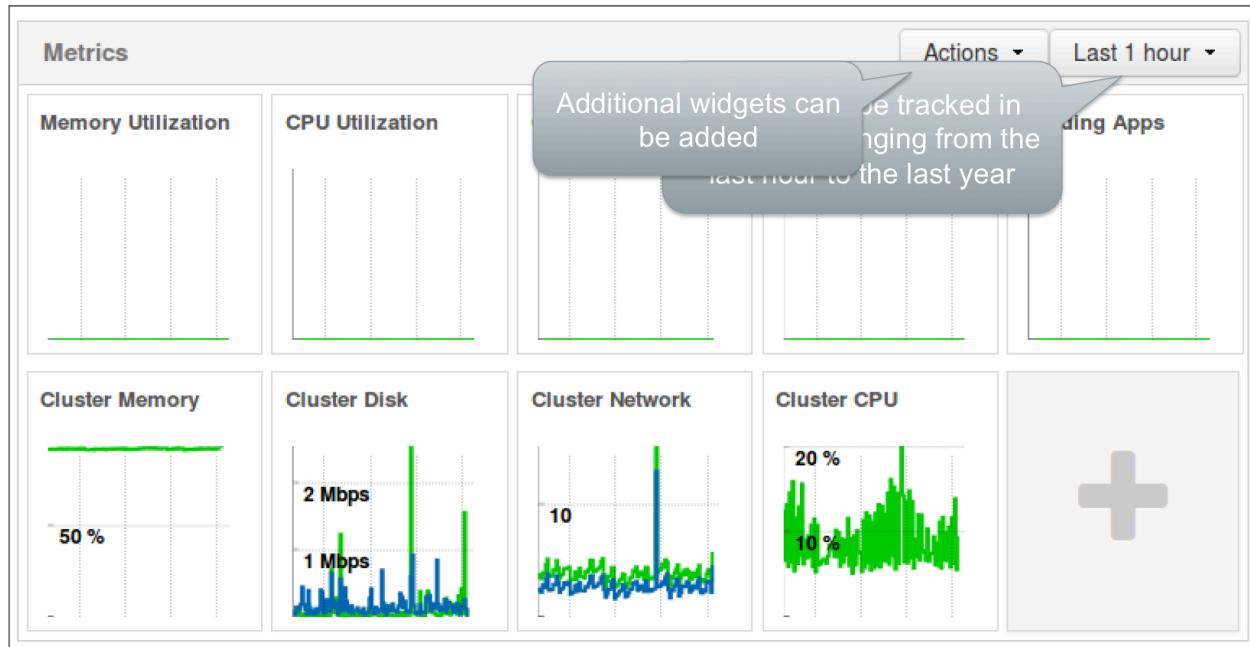
Context-sensitive Service Actions Menu

Each service in the Ambari Web UI has a context-sensitive **Service Actions** menu button. Selecting an action opens a wizard window where you control and view the action's operation. Because the **Service Actions** menu button is context sensitive, the menu choices are different for each service.

The **Service Actions** menu button is where you can start, stop, or restart an entire service or specific service component types. Other lessons provide more detail about a specific services and their service actions.

The **Actions** menu button is where you would start, stop, or restart an entire cluster. You can install additional services using the **Actions** menu button. For example, Falcon is not installed on this cluster and could be added using the **Actions** menu button.

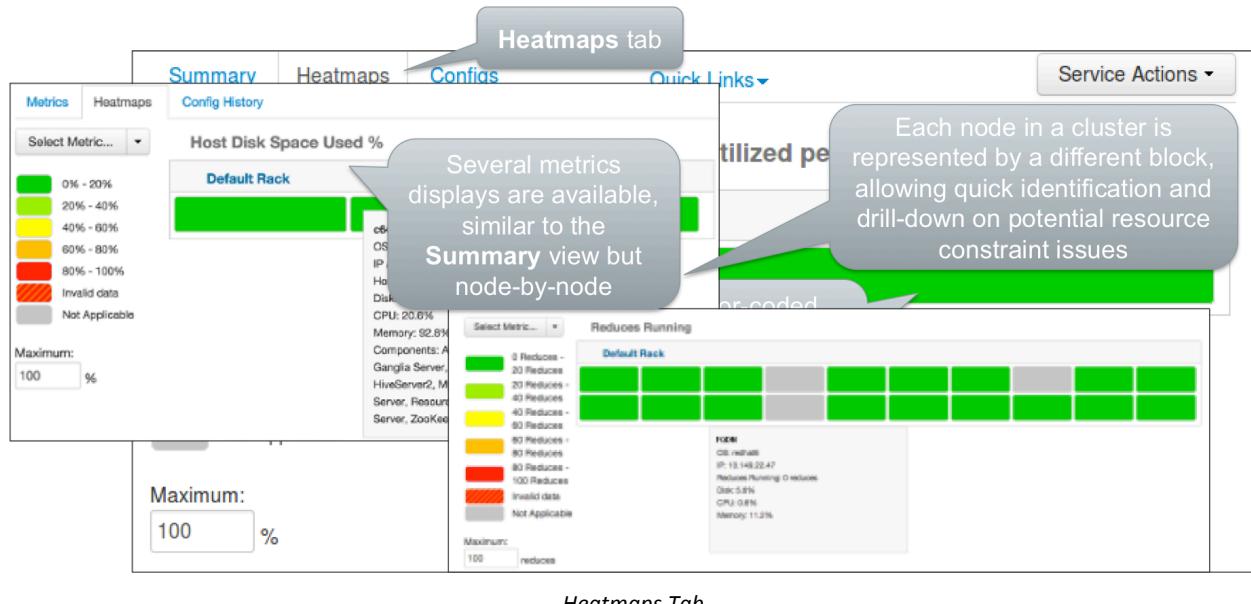
Service Resource Monitoring



Service Monitoring Widgets

At the bottom of each services' tab you will find a Metrics section that contains a collection of monitoring widgets. The widgets are active if the Ambari Metrics service is running. By default, resource usage is shown for the previous hour, but this can be adjusted to range up to a full year of data. Additional widgets can be added using the **Actions** menu button.

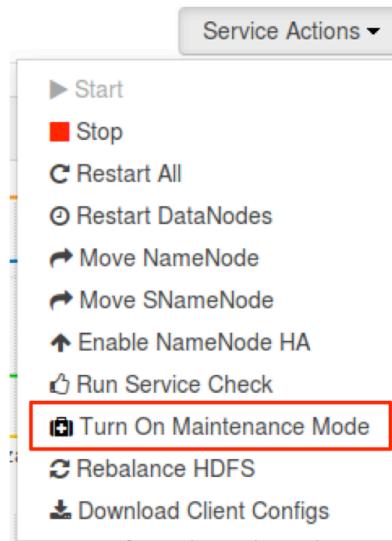
Service Heatmaps



Heatmaps Tab

The **Heatmaps** tab displays similar metrics to the **Summary** tab, but color-coded on a node-by node basis. Each node is represented by an individual block. Hovering over a block gives you relevant information about the node, including its name, operating system, IP address, rack location, resource utilization, and Hadoop components installed.

Maintenance Mode



Turning On Maintenance Mode

Maintenance Mode should be enabled when making cluster hardware or software changes. For example, Maintenance Mode is useful when changing a service configuration, shutting down a cluster node, removing a cluster node, or even during troubleshooting sessions. Explicitly turning on Maintenance Mode for a service implicitly turns on Maintenance Mode for components or hosts that run the service.

Maintenance Mode suppresses Ambari alerts, warnings, and status change indicators. It also exempts an object from host-level or service-level bulk operations. For example, if HDFS DataNode 8 were shut down for maintenance, Ambari would generate alerts and warning messages about that DataNode's "failure". Enabling Maintenance Mode on the DataNode prior to shutting it down would prevent this behavior. As another example, if DataNodes 11 and 12 are placed in Maintenance Mode, then they would be exempt from the bulk **Restart All** or **Restart DataNodes** actions on the Ambari HDFS Service Actions menu button.

While Maintenance Mode prevents bulk operations, an Ambari user may still use Ambari to explicitly start and stop a specific service, service component, or host in Maintenance Mode.

Managing Hadoop Configuration Properties with Ambari

Viewing or Changing Service Configurations

Not all services have both **Settings** and **Advanced** tabs.

Each service includes a **Configs** tab.

Float the mouse pointer over a property setting to get a description.

Customizing Service Configurations

Each cluster service has configuration property settings that can be customized per-cluster or per-node. To view these properties in the Ambari Web UI, select the **Services** page, select the service, and then select the service's **Config** tab.

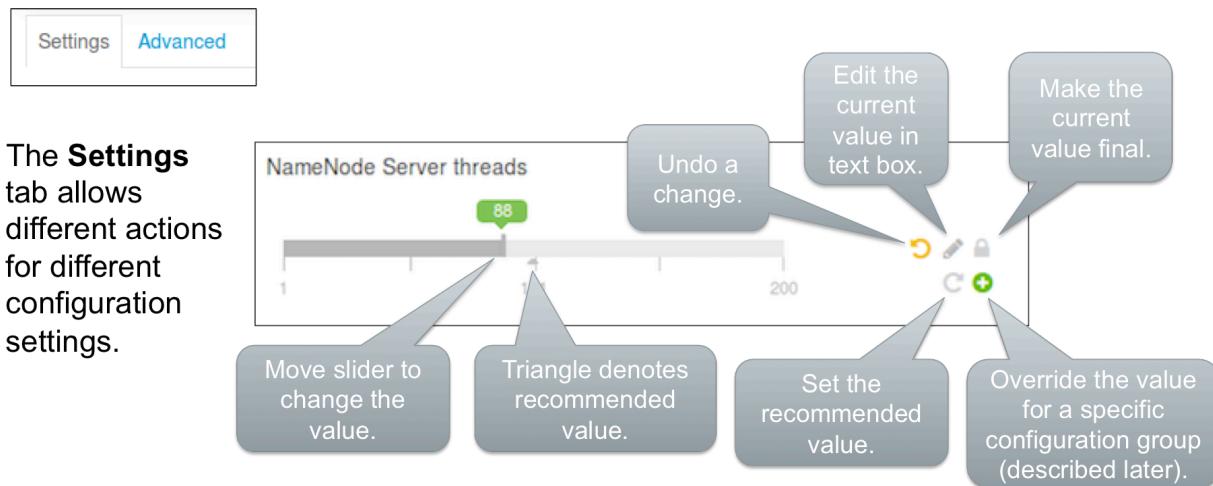
Ambari 2.1 includes a new feature named Guided Configuration. Guided Configuration uses the current physical resources of the cluster nodes along with built-in formulas to automatically calculate an optimum value for key and common configuration settings. The optimum value is visually displayed on a slider bar by a small triangle icon.

Settings controlled by Guided Configuration are interdependent. These interdependencies manifest as recommendations for other settings when you try to save a value change to a interdependent setting.

Every service has a **Configs** tab, but not all services have the **Settings** and **Advanced** sub-tabs. Services without the **Settings** and **Advanced** tabs simply list all configuration settings and their current values on the **Configs** page.

For all configuration settings no matter where they are displayed, short, on-screen descriptions are automatically displayed in pop-up windows by floating the mouse pointer over the configuration setting.

The Settings Tab



Settings Tab - Part of "Guided Configuration"

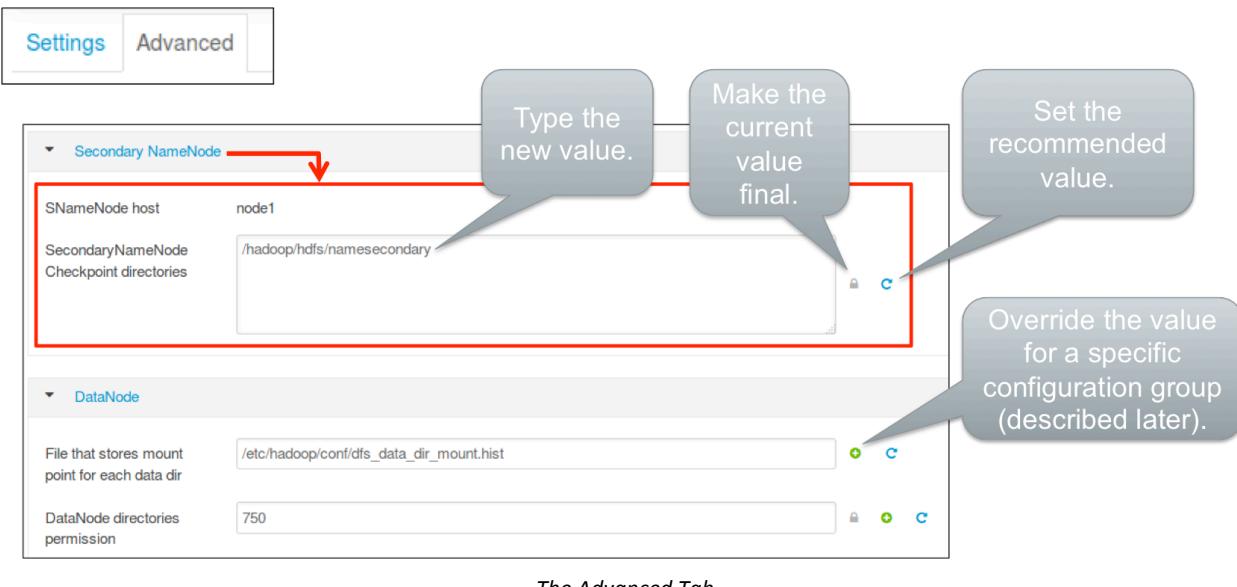
The **Settings** tab is new with Ambari 2.1 and is part of Guided Configuration. The **Settings** tab allows different actions for different configuration settings. The example here allows six different actions. The administrator may:

- Use a slider to change a current value (the triangle icon denotes the recommended value)
- Edit the current value in a text box, changing it to a preferred value
- Make the current value a final value
- Override the current value for a specific configuration group (configuration group's are described later)
- Set the value to the recommended value for this cluster (Guided Configuration)
- Undo a change

After making any change using any method, click the **Save** button (not shown).

The ability to change a value using a slider bar is the primary difference between making changes on the **Settings** tab versus making changes on the **Advanced** tab.

The Advanced Tab



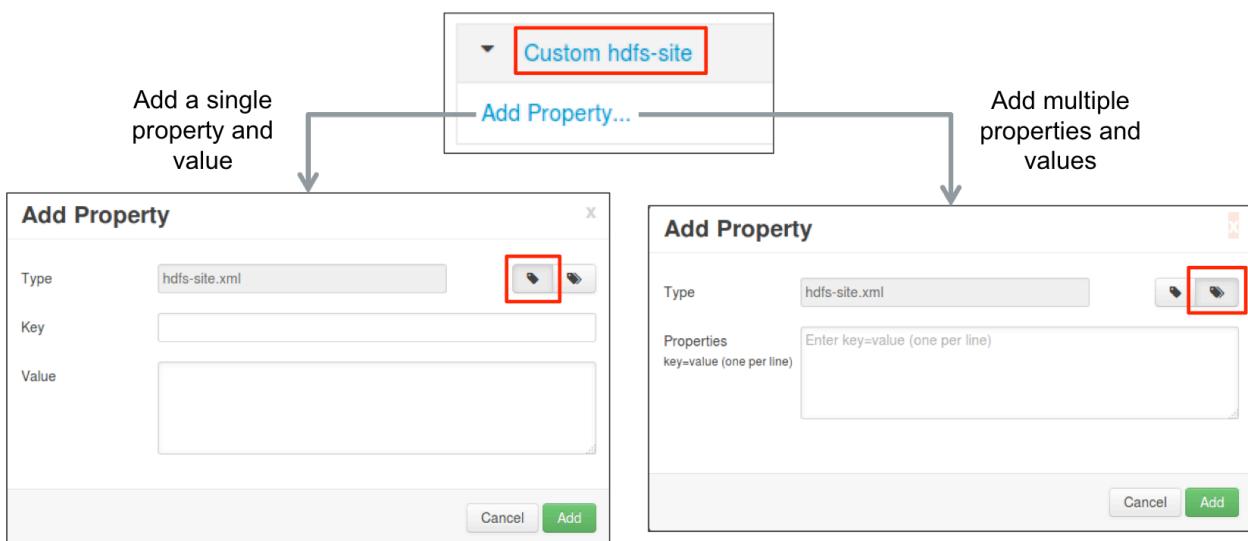
On the **Advanced** tab, related properties are grouped together. For example, properties that relate to the HDFS Secondary NameNode are grouped together under the Secondary NameNode heading. These groupings do not necessarily align to a single, underlying configuration file. This means that the grouped properties can span more than one configuration file.

As mentioned before, short, on-screen property descriptions are displayed by floating the mouse pointer over any property setting.

To manually change a property setting, click the current value, type in the new value.

You may also click the **Set Recommended** icon or make a value final. In all cases, click the **Save** button (not shown) in order to save the change.

Adding New Configuration Properties



Most services feature one or more configuration property groups with “custom” in their name. These groups include an **Add Property** link. The **Add Property** link opens a dialog box that enables you to add either a single property name and value or multiple property names and values.

Clicking the **single-tag** icon enables you to add a single new property and value. Clicking the **double-tag** icon enables you to add multiple properties and values. After you click **Add** you must still click **Save** in the main **Configs** tab panel.

Service Configuration Version History

The screenshot shows the Ambari interface for managing HDFS configurations. On the left, a sidebar lists various services: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, and ZooKeeper. The 'Configs' tab is selected. In the center, a message box says 'Restart Required: 8 Components on 3 Hosts' with a 'Restart' button. Below this, a section titled 'Manage Config Groups' shows a list of configuration versions:

Version	Author	Timestamp
V6	steve	a moment ago
V5	admin	about a day ago
V4	2...	ambari-upgra...
V3	2...	ambari-upgra...
V2	2...	ambari-upgra...
V1	admin	2 days ago

At the bottom, there are 'Discard' and 'Save' buttons. To the right of the screenshot, a yellow double-arrow icon is shown with the text '= Restart Icon'.

Service Configuration Version History

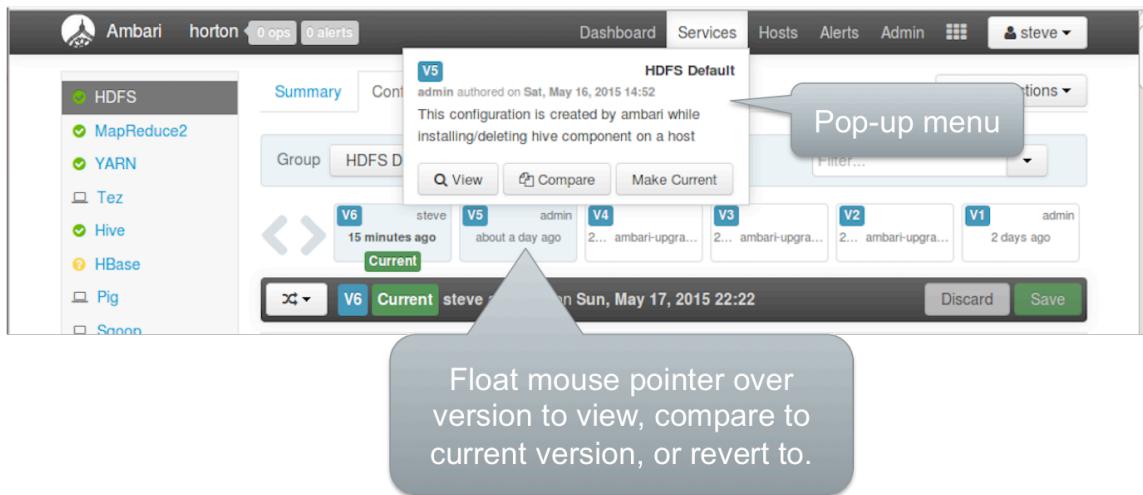
Ambari maintains a complete per-service configuration version history in the file `/var/log/ambari-server/ambari-config-changes.log`. Each time one or more service configuration properties are modified and saved, a new configuration version is generated. The configuration versions are displayed in Ambari, as seen here.

The current configuration is always tagged with the green **Current** label. Many service configuration property changes require that one or more service components, or even an entire service, be restarted. Ambari notifies you of any service that must be restarted by tagging that service with a yellow, double-arrow restart icon.

Services may be restarted using the **Restart** button shown here, or by selecting the service and choosing **Restart All** from its **Service Actions** menu button.

Viewing, Comparing, and Reverting Configurations

- View:** View a service configuration (V5 here)
- Compare:** Compare one service configuration to another (V5 to V6 here)
- Make Current:** Revert to a service configuration (V5 here), may require service restarts



Managing Configurations

The Ambari interface provides multiple methods to view a specific service configuration, compare two service configurations, and revert to a specific service configuration. The screen capture illustrates one method.

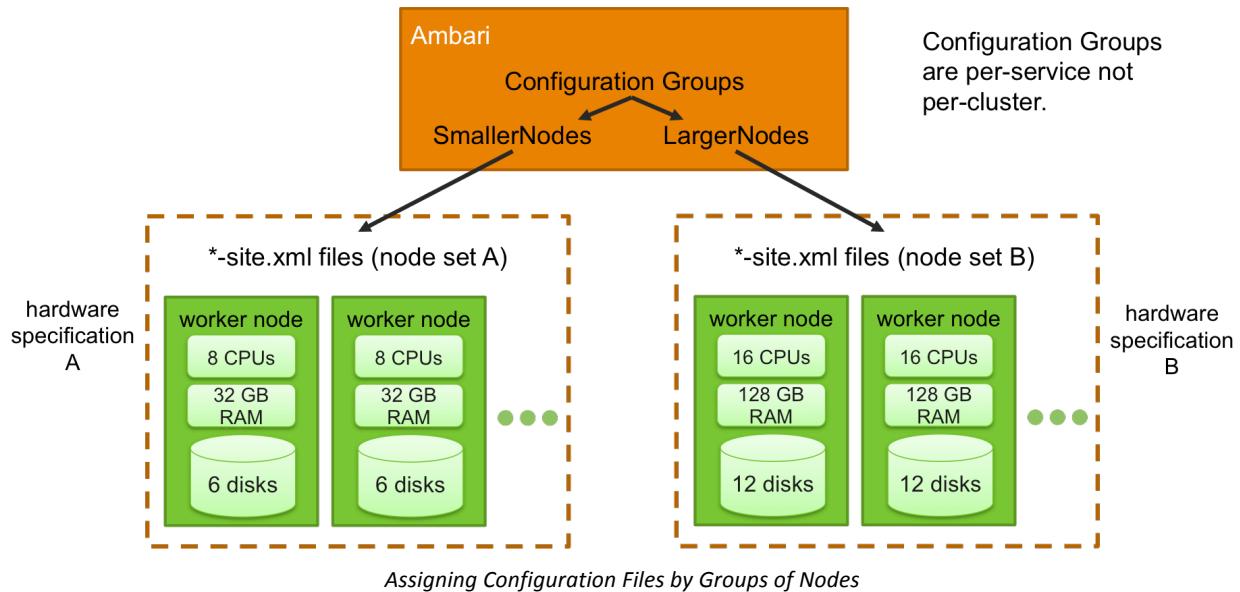
Floating the mouse pointer over a specific version opens a pop-up window. The pop-up window offers the choice to view the configuration, compare the differences between this version and another version, and make this version the current version.

If you click **View**, the **Configs** tab panel is updated to display the selected version. The same can be accomplished by clicking a specific version rather than just floating the mouse pointer over the version.

If you click **Compare**, the **Configs** tab panel is updated to display only those properties that are different between the two versions. The value of each of these properties is displayed so that they can be easily and quickly compared.

If you click **Make Current**, Ambari reverts to this version of the service configuration. One or more services or service components might have to be restarted after reverting to an older configuration. Restarting a service does impact cluster operations so it is best to schedule downtime for cluster changes.

Ambari Configuration Groups



Many service configuration property values are based on the hardware specification of the cluster node. If the hardware specification is homogeneous, then the same set of configuration files can often be used across all nodes. But what happens in the situation where the hardware specification is different across the nodes in a cluster? This commonly is the case where the number of cluster nodes has grown over time. As new hardware is ordered, the same hardware specification might not be available or desirable. Ambari Configuration Groups is designed to meet this challenge.

Ambari Configuration Groups enable a cluster administrator to create groups of cluster nodes, and assign each group their own set of configuration files. Each set of configuration files is customized to work with the hardware specification of a specific group of nodes.

Note

Configuration Groups are per-service and not per-cluster.

Managing Configuration Groups

Managing Configuration Groups

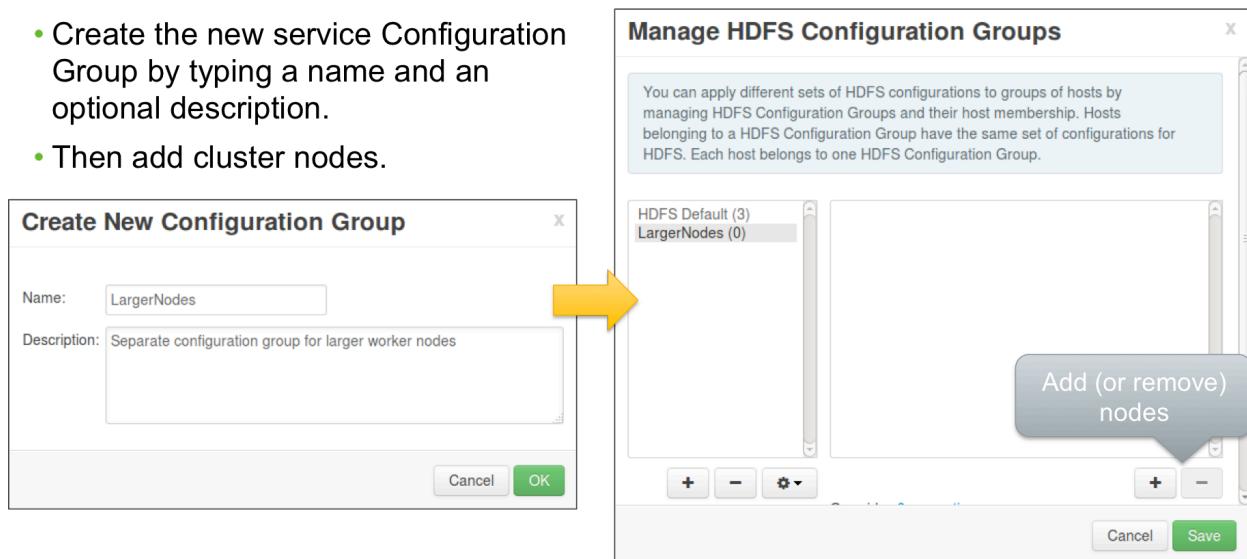
Click **Manage Config Groups** on a service's **Configs** tab panel to manage Configuration Groups.

All existing Configuration Groups for the service are listed in the Manage <Service_Name> Configuration Groups window. Any cluster nodes assigned to the Configuration Group are listed when a group is selected. By default, all nodes would be assigned to the default configuration group by the Ambari installation.

Use the plus (+) and minus (-) buttons below the Configuration Groups list to add and delete Configuration Groups. The **Tools** drop-down menu button is used to rename or duplicate the selected Configuration Group.

Creating a New Configuration Group

- Create the new service Configuration Group by typing a name and an optional description.
- Then add cluster nodes.



Creating a New Configuration Group

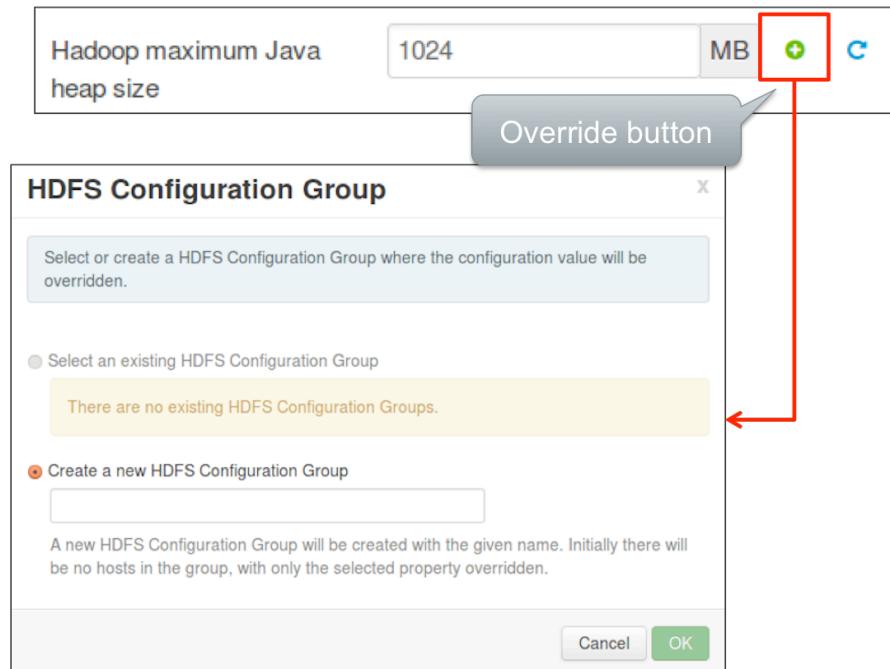
Creating a new service Configuration Group is a quick operation. In the Manage <Service_Name> Configuration Groups window, click the plus (+) button to open the Create New Configuration Group window. Type a name, an optional brief description, and click **OK**.

The new group appears back in the Manage <Service_Name> Configuration Groups window.

The plus (+) and minus (-) buttons below the nodes list are used to add and remove nodes from the selected Configuration Group. Depending on the service components running on the node, it might not be removable from a Configuration Group.

A node may only be a member of a single Configuration Group.

Overriding a Configuration Property Setting

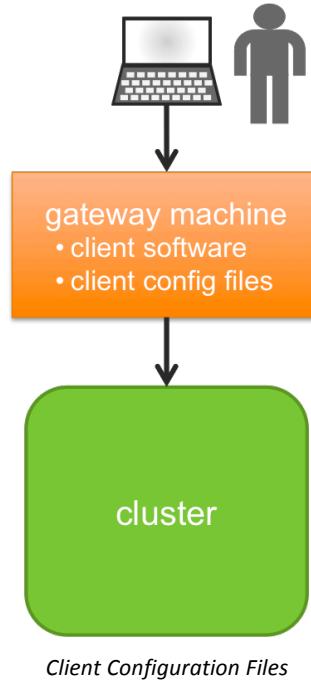


Overriding a Configuration Property Setting

Many configuration properties are supported in Configuration Groups. These properties include an override button in the Ambari Web UI. The **Override** button is a green-white plus symbol.

Clicking the **Override** button opens the <Service_Name> Configuration Group window. This window enables you to select an existing Configuration Group in which to place the modified property value. You may also create a new Configuration Group in which to place the modified property value.

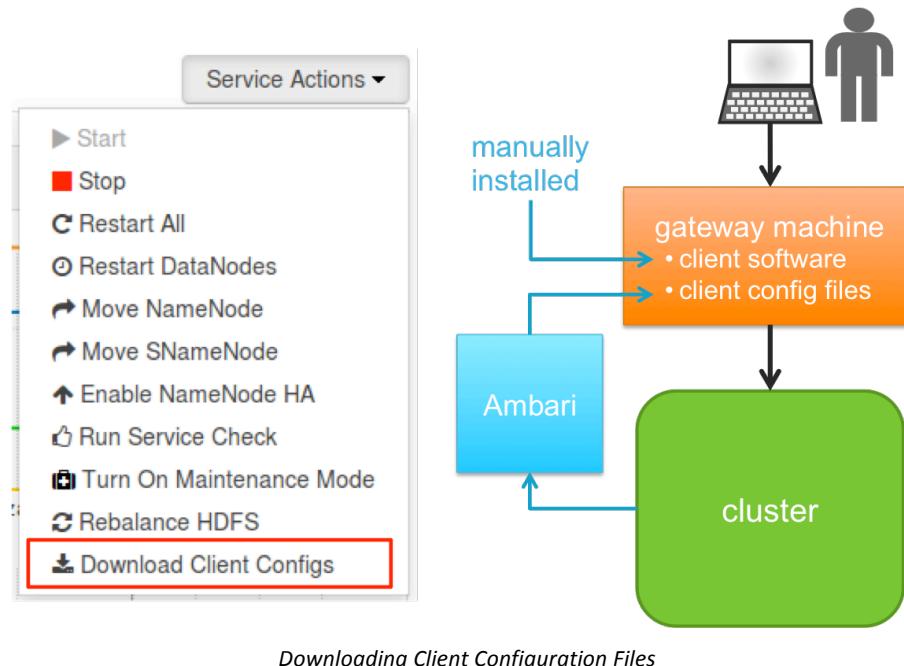
Client Configuration Files



Some machines have just client-side software installed. This is common for gateway machines that are outside of the cluster, but used by users and applications to access the cluster services. There are different client-side software packages. The Hadoop Client, for example, includes software to work with YARN, HDFS, Tez, and MapReduce.

Many services have their own client-side software. For example, the Hive Client includes Hive client-side software. In any case, the client-side software requires its corresponding set of client configuration files. All client configuration files must be properly configured in order to access and work with a specific cluster.

Downloading Client Configuration Files



Downloading Client Configuration Files

If a machine is managed by an Ambari agent, then Ambari can install both the client-side software and the corresponding client configuration files. However, if a machine is not managed by an Ambari agent then the client-side software would have to be manually installed. After installation, the client-side software would still require properly configured client configuration files. Rather than manually editing these files, they can be downloaded from the cluster using Ambari.

Ambari downloads the client-side files to the machine running the Ambari Web UI in a browser window. The client configuration files are downloaded per-service and not per-cluster. An Ambari agent is not required to download client configuration files.

REST APIs

REST (REpresentational State Transfer) APIs provide a way for application developers to write applications that interface with Hadoop clusters over HTTP. REST APIs use a relatively simple structure consisting of a small set of “verbs,” or actions that can be performed (examples include GET, PUT, POST, and DELETE.) Resources in the cluster that can be managed via REST APIs are assigned a URI (Uniform Resource Identifier) that allows the application to interact with the various components of the environment being accessed.

URIs are structured as follows: `http://<service address and port>/ws/<API version>/<resource>/<optional subset>/<optional subset>...`. The service address must include both the URL and port as part of the URI. The ws in the URI simply stands for “Web Services.” The API version must be specified (ex: v1). The resource and subset might be something representing the entire cluster, a specific node, a specific application, or historical information about applications or other resources.

An API call is typically an HTTP request that combines an appropriate verb with a URI. For example: using the API call `GET http://node1:8088/ws/v1/cluster/info` will access the YARN ResourceManager (default port 8088) on node1, using API version 1. It requests access to the cluster resource, and specifically requests general information. The Hadoop cluster would return information similar to what you might see by opening a web browser and pointing to `http://node1:8088/cluster/cluster`. Indeed, the YARN ResourceManager UI is actually using the REST API to gather the information it needs in order to provide the display.

Managing Hadoop Using REST APIs

APIs are not typically directly used by Hadoop Administrators to manage clusters. On rare occasions, however, it might be useful to use an API call to either test cluster functionality or assist a developer while troubleshooting an issue. Thus, being aware of APIs can be of some benefit.

Because an administrator would not typically write an application in order to access or test API calls, an easy way to access them for testing is to use the cURL command, which can be installed on either Linux or Windows machines. Thus, to actually see the information returned in the API call example called above, an administrator would run the following command:

```
curl -X GET http://node1:8088/ws/v1/cluster/info
```

The -X option (that's a capital X – the lower-case x performs a different function in cURL) specifies a custom request method. Because cURL is typically used to access URLs, it defaults to GET if nothing is specified. However, for some YARN API calls, the PUT or POST calls are used, and might require additional options depending on what is being done. Thus, it is a good practice to go ahead and use the -X option when using cURL for API call testing, even if you are doing a GET API call.

Advanced REST API Calls

POST, PUT, and DELETE call options often require additional options in the command and necessary parameters passed as part of the URI (normally designated by a “?” at the end of the URI followed by <info_type>=<info>).

Example: `http://<address:port>/ws/v1/<resource path>?user=root?password=hadoop`

It would not be common, however, to have to utilize this complex of an API call for administrative purposes. The Hadoop Administrator would typically use other tools or methods to accomplish these kinds of tasks.

Refer to <http://hortonworks.com/docs> for updated information and specific examples of how to manage applications and other cluster resources.

Manually Editing Configuration Files

It is possible to administer cluster and service configuration without Ambari. Modifying a configuration is three step process.

First, find and edit the correct file. There are many configuration files and hundreds of configuration properties. You must find the file containing the properties that must be changed and then make the changes. For example, what if you wanted to change the number of failed disks tolerated by HDFS DataNodes? You would have to find and change the `dfs.datanode.failed.volumes.tolerated` property in the `/etc/hadoop/conf/hdfs-site.xml` file.

Second, you must transfer the edited file to the appropriate cluster nodes. Extending the previous example, because you changed a DataNode property in the `hdfs-site.xml` file, you must transfer the updated file to all DataNode machines in the cluster.

Lastly, you must restart the service or service components in order to make the changes effective. This means that you must know the command-line commands to stop, start, or restart all services and service components. This also means that you would have to know service interdependencies. In some cases making a change to one service means that other services have to be restarted as well. Finishing the DataNode example, you would have to manually run the command to restart all the DataNode service components across the cluster.

All of this requires a higher level of cluster administration expertise. Even with the necessary expertise, the process can be time consuming and error prone. Scripting the process can help to mitigate these issues.

Be aware that manual administration is not compatible with Ambari administration. If you are managing your cluster configuration using Ambari, it is not recommended to regularly use other management methods. Ambari not only updates the configuration files as directed by an administrator, but it also makes corresponding updates in its database. If an administrator manually edits a configuration file these changes are not reflected in the Ambari database. Manually modified configuration files are overwritten by the information in the Ambari database when the modified service is restarted.

HDP Memory Configuration

Option	Description
<code>-c <num></code>	The number of CPU cores on each worker node.
<code>-m <num></code>	The amount of memory on each worker node, in GBs.
<code>-d <num></code>	The number of disks on each worker node.
<code>-k <boolean></code>	“True” if HBase is installed, otherwise “False”

yarn-utils.py Script Options

There are several configuration properties that should be modified either during installation or immediately following installation. In the absence of Ambari’s Guided Configuration feature, these properties must be manually calculated and configured. These properties determine the memory allocated to various YARN and MapReduce components. Improper configuration can negatively affect cluster performance.

Hortonworks provides an optional downloadable `hdp-configuration-utils.py` Python script to that can be run to help determine the initial values for these property settings. Hortonworks recommends using this script if manually configuring a cluster.

The script is not installed during installation but must be downloaded by a user. The `wget` command used to download the current version of the script is typically found in the HDP Manual Install guide at <http://docs.hortonworks.com>. This is a Python script and must be run on a system that has Python installed. The four script options are listed and described in the table.

Running the `hdp-configuration-utils.py` Script

Find and change these properties in the `yarn-site.xml` and `mapred-site.xml` files.

```
# python hdp-configuration-utils.py -c 12 -m 24 -d 8 -k True
Using cores=12 memory=24GB disks=8 hbase=True
Profile: cores=12 memory=16384MB reserved=8GB usableMem=16GB disks=8
Num Container=15
Container Ram=1024MB
Used Ram=15GB
Unused Ram=8GB
yarn.scheduler.minimum-allocation-mb=1024
yarn.scheduler.maximum-allocation-mb=15360
yarn.nodemanager.resource.memory-mb=15360
mapreduce.map.memory.mb=1024
mapreduce.map.java.opts=-Xmx819m
mapreduce.reduce.memory.mb=2048
mapreduce.reduce.java.opts=-Xmx1638m
yarn.app.mapreduce.am.resource.mb=2048
yarn.app.mapreduce.am.command-opts=-Xmx1638m
mapreduce.task.io.sort.mb=409
```

This is an example of running the `hdp-configuration-utils.py` script.

Using the script output, find and change the corresponding properties in the `yarn-site.xml` and `mapred-site.xml` files.

Starting Services using the CLI

All installed services must be starting in a specific order. The order is:

- 1) Knox
- 2) ZooKeeper
- 3) HDFS
- 4) YARN
- 5) HBase
- 6) Hive Metastore
- 7) HiveServer2
- 8) WebHCat
- 9) Oozie
- 10) Storm
- 11) Kafka

Some services have multiple components and each component has its own start-up command. All the the start-up commands are lengthy to type.

Reference

For examples of each start-up command, see <http://docs.hortonworks.com>.

Stopping Services using the CLI

Some services have multiple components and each component has its own start-up command. All running services must be stopped in a specific order. The order is:

- 1) Knox
- 2) Oozie
- 3) WebHCat
- 4) HiveServer2
- 5) Hive Metastore
- 6) HBase
- 7) YARN
- 8) HDFS
- 9) ZooKeeper
- 10) Storm
- 11) Kafka

All the stop commands are lengthy to type.

Reference

For examples of each stop command, see <http://docs.hortonworks.com>.

Knowledge Check Questions

- 1) Ambari installs the core Hadoop configuration files in which directory?
- 2) What is a purpose of the `*-site.xml` files?
- 3) What logging mechanism is commonly used by Apache frameworks?
- 4) How can you prevent a user application from overriding a per-cluster configuration property setting?
- 5) If Ambari installs Hive, where might the Hive configuration files be found?
- 6) True or false? The Ambari Web UI Services page displays all Hadoop services, installed or not.
- 7) If Maintenance Mode is enabled for DataNode 15, will it be restarted if **Restart All** is selected from the HDFS service's **Service Actions** menu?
- 8) Which tab on the Ambari Web UI Services page enables you to change service configuration?
- 9) Service configuration version history in the Ambari Web UI enables you to view, compare, and _____ configuration versions.
- 10) What is the purpose of Ambari Configuration Groups?
- 11) True or false? A cluster node may be a member of only a single Configuration Group for a single service.
- 12) What is the purpose of the Override button displayed next to configuration properties in the Ambari Web UI?
- 13) True or false? Download Client Configs in the Ambari Web UI is per-service and not per-cluster.

Knowledge Check Answers

- 1 Ambari installs the core Hadoop configuration files in which directory?
The /etc/hadoop/conf directory.
- 2 What is a purpose of the *-site.xml files?
They enable per-cluster configuration.
- 3 What logging mechanism is commonly used by Apache frameworks?
Apache Log4j
- 4 How can you prevent a user application from overriding a per-cluster configuration property setting?
By making the configuration property setting a final setting.
- 5 If Ambari installs Hive, where might the Hive configuration files be found?
In the /etc/hive/conf directory.
- 6 True or false? The Ambari Web UI Services page displays all Hadoop services, installed or not.
False. It only displays installed services.
- 7 If Maintenance Mode is enabled for DataNode 15, will it be restarted if **Restart All** is selected from the HDFS service's **Service Actions** menu?
No.
- 8 Which tab on the Ambari Web UI Services page enables you to change service configuration?
The Configs tab.
- 9 Service configuration version history in the Ambari Web UI enables you to view, compare, and _____ configuration versions.
revert (or Make Current)
- 10 What is the purpose of Ambari Configuration Groups?
They enable sets of cluster nodes to have different configuration property settings.
- 11 True or false? A cluster node may be a member of only a single Configuration Group for a single server.
True.
- 12 What is the purpose of the Override button displayed next to configuration properties in the Ambari Web UI?
It provides the ability to modify a property value and then place the modified value in a specific Configuration Group.
- 13 True or false? Download Client Configs in the Ambari Web UI is per-service and not per-cluster.
True.

Summary

- By default, Ambari installs core Hadoop configuration files in `/etc/hadoop/conf` and application services' configuration files in `/etc/<service_name>/conf`.
- A running job's actual configuration is a combination of the default, per-site, possibly per-node, and per-job configuration.
- Final properties cannot be overridden by user applications.
- The Ambari Web UI provides an easy-to-use cluster administration interface.
- Ambari maintains a service configuration history.
- Ambari Configuration Groups enable a cluster administrator to create groups of cluster nodes, and assign each group their own set of configuration files.
- Ambari, without an agent, can still download per-service client configuration files.
- Manual configuration file editing is not compatible with Ambari administration.

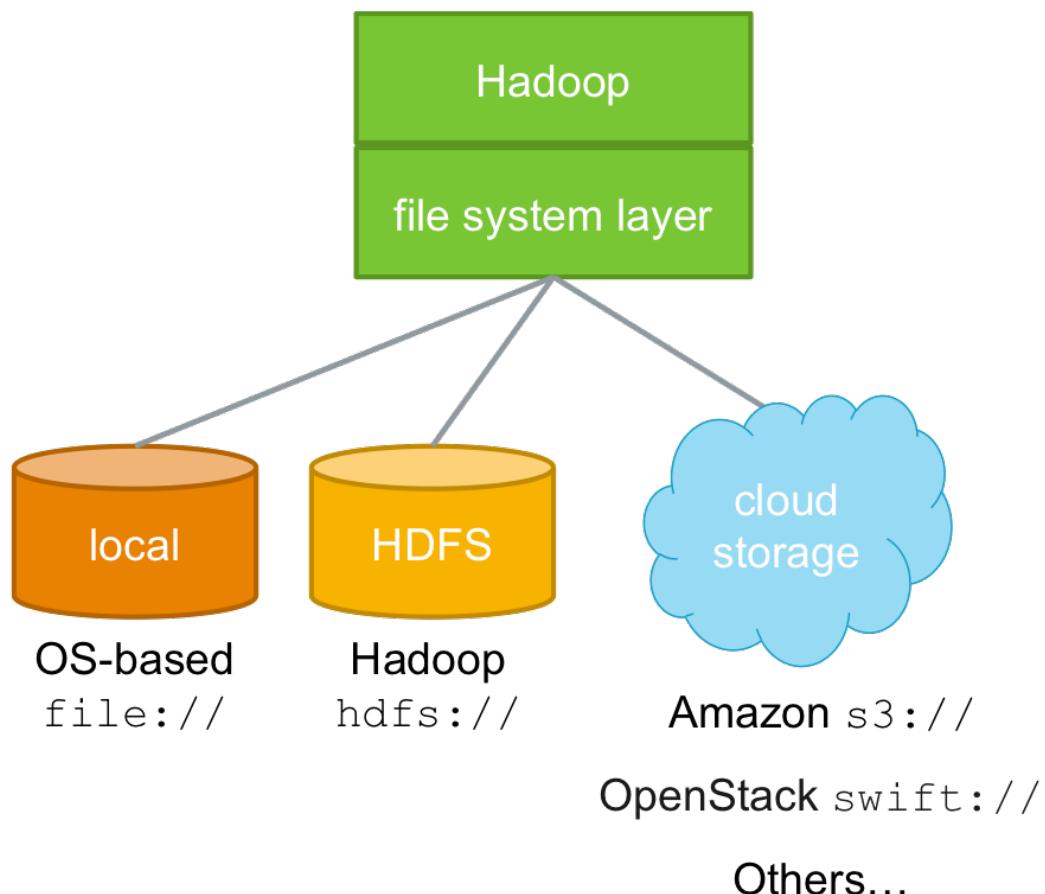
Using HDFS Storage

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe the Hadoop Distributed File System (HDFS)
- ✓ Perform HDFS Shell operations
- ✓ Use the Ambari Files View
- ✓ Use WebHDFS
- ✓ Protect data using HDFS Access Control Lists (ACLs)

Hadoop Distributed File System (HDFS)



Hadoop and Storage

Hadoop is extensible and designed to work with different file system types. Users or applications can access specific file system types by using specific URI prefixes on the command line or in program code. For example, the `hdfs dfs -ls` command is used to display the contents of a directory. The command may be used with different prefixes to access different file system types.

The command `hdfs dfs -ls file:///bin` displays the contents of the `/bin` directory in the local file system.

The command `hdfs dfs -ls hdfs:///root` displays the contents of the `/root` directory in HDFS.

The default file system type is configured by the `fs.defaultFS` property in the `core-site.xml` file. If it is set to `hdfs://<NameNode_host>:8020`, the HDFS client software will attempt to use RPC to contact the HDFS NameNode at port 8020 unless explicitly told otherwise. The command `hdfs dfs -ls /root` is simplified because the `hdfs://` prefix on the path name is no longer necessary. Even with the default file system type specified as HDFS, the command `hdfs dfs -ls file:///bin` still displays the contents of the `/bin` directory in the local file system.

HDFS Characteristics

Characteristic	Description
Hierarchical	Directories containing files are arranged in a series of parent-child relationships.
Distributed	File system storage spans multiple drives and hosts.
Replicated	The file system automatically maintains multiple copies of data blocks.
Write-once, read-many optimized	The file system is designed to write data once but read the data multiple times.
Sequential access	The file system is designed for large sequential writes and reads.
Multiple readers	Multiple HDFS clients may read data at the same time.
Single writer	To protect file system integrity, only a single writer at a time is allowed.
Append-only	Files may be appended, but existing data not updated.

HDFS Characteristics

Like all file system types, HDFS has specific characteristics. They are listed and described in the table.

Like the ext4 or NTFS file systems, HDFS is hierarchical. There is a top-level `/` directory, commonly with one or more child directories. Each child directory can have its own child directories, and so on. Each directory can contain zero or more files.

HDFS is a distributed file system. It can span multiple drives on multiple systems. Its distributed nature is what makes HDFS so massively scalable, and suitable for storing and reading extremely large data files. Large files are automatically distributed across multiple drives, which collectively enable high I/O write and read rates.

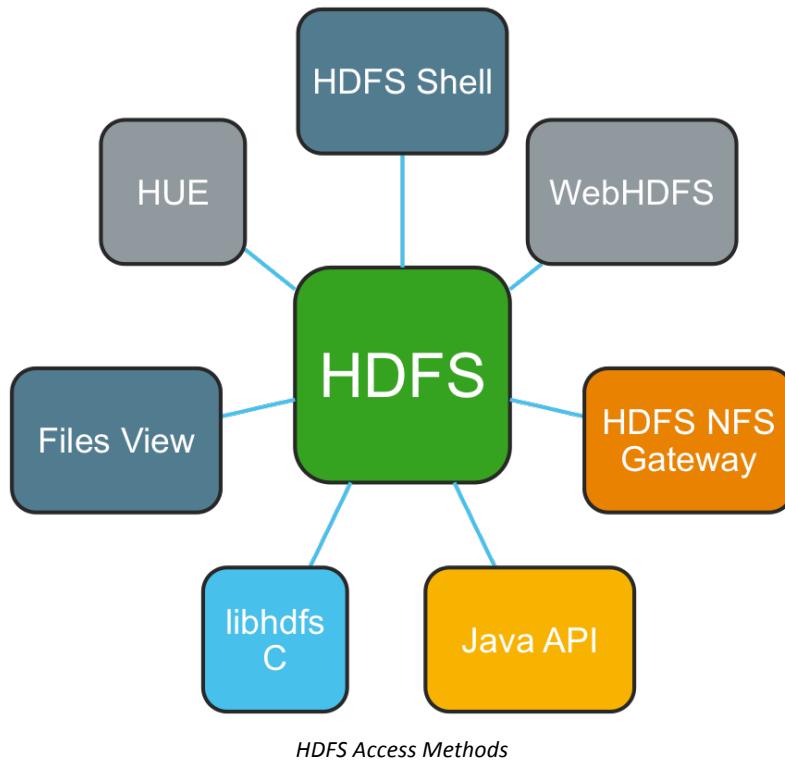
HDFS uses replication to protect data. Each data block is automatically replicated across multiple drives and multiple systems. The default replication factor is three, which instructs HDFS to keep each data block on three disks on three separate systems.

HDFS is designed as a write-once, read-many file system. Data is intended to be written once to disk, and then read multiple times as the data is either repeatedly analyzed or used to create new, transformed data in another file.

HDFS supports multiple readers. Because there are commonly three copies of each data block, it is possible for multiple readers to access the same data at the same time from different systems. However, the file system uses file locking to ensure that there is only a single writer at a time. This is used to protect the file system integrity.

Files in HDFS may be initially written, appended to, and deleted. Existing data within a file is immutable and cannot be updated.

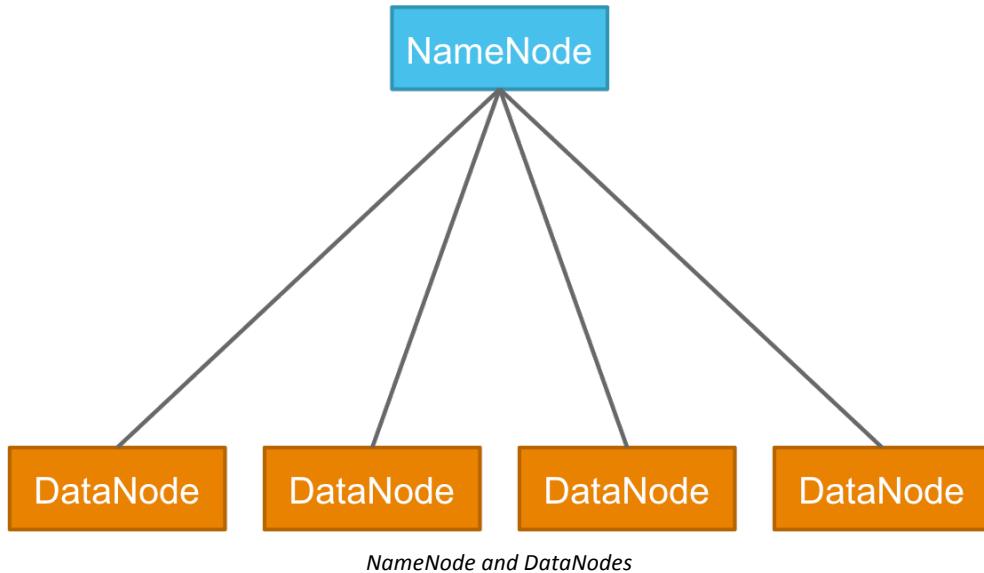
Accessing HDFS



Hadoop provides several ways to access, use, and manage HDFS.

- **HDFS Shell**
Includes various commands used to access and manage HDFS files and directories (uses Java API underneath)
- **WebHDFS**
An HTTP REST API used to access and manage HDFS files and directories
- **HDFS NFS Gateway**
Enables HDFS to be NFS mounted and accessed as part of a client's local file system
- **Java API**
Java classes used to access and manage file systems, including HDFS
- **libhdfs C**
A Java Native Interface (JNI) C API to access HDFS from Linux or Windows.
- **Ambari Files View**
Ambari View to graphically manage files and directories
- **HUE (Hadoop User Experience)**
Web-based UI used to access HDFS, Hive, Pig, and other Hadoop applications

NameNode and DataNodes Introduction



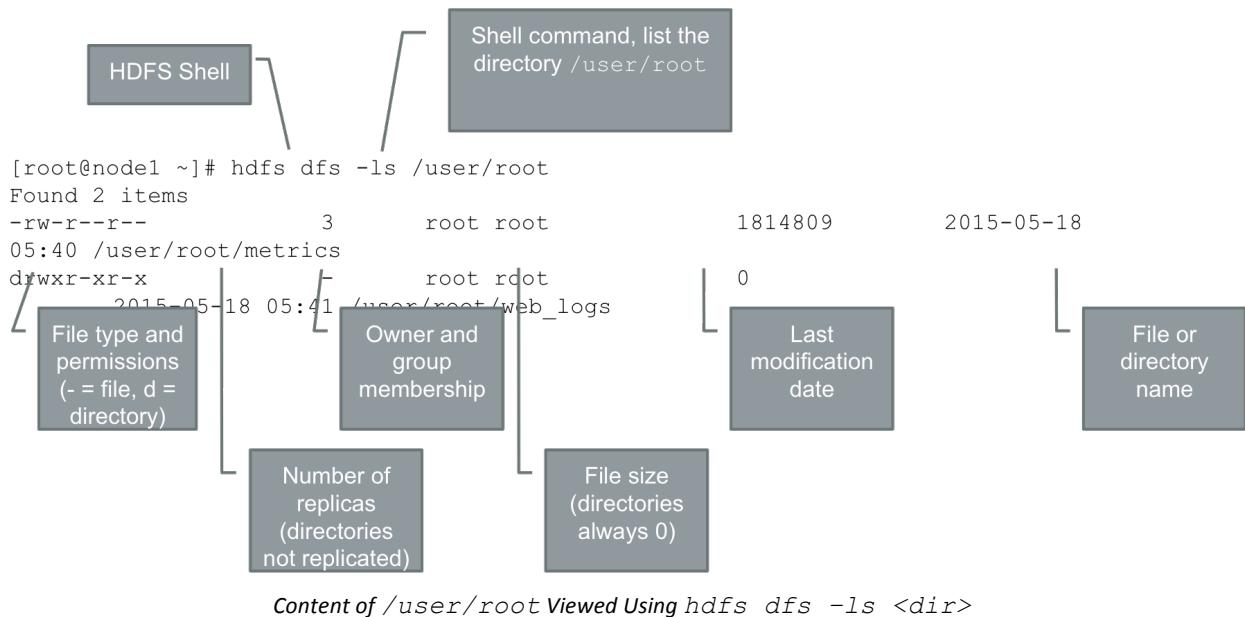
The NameNode and DataNode are components of the HDFS service. The NameNode is an HDFS master component while a DataNode is an HDFS worker component.

The NameNode maintains critical namespace and metadata information for the HDFS. This information includes such items as file names, directory names, the file system hierarchy, file and directory permissions and ownerships, file and directory last modification times, and access control lists. This information is so critical that HDFS includes a number of methods to protect this data using redundancy. Another lesson describes the NameNode and its operation in more detail.

The DataNodes contain only data blocks. By default, a data block is a maximum of 128 MB in size but this can be changed. Each block is assigned a unique ID number and maps to a file name maintained on the NameNode. Only the NameNode has information about which data blocks map to which files.

Data blocks are so critical that HDFS automatically replicates them across multiple DataNodes—three by default. Another lesson describes the DataNodes and their operation in more detail.

File and Directory Attributes



The HDFS Shell command `hdfs dfs -ls <dir>` is an easy way to view the attributes of files and directories. The example lists the contents of the directory `/user/root`.

The first character of the first column denotes the type of file. A dash character indicates a file while the character d indicates a directory. The remaining nine characters in the first column denote the permissions. Permissions are described later.

The second column lists the number of replicas for a file. Files are replicated while directories are not. This is a result of how, and specifically where, file and directory information is stored. File blocks are stored on the HDFS DataNodes (worker nodes) while directory information is stored as file system metadata on the HDFS NameNodes (master nodes).

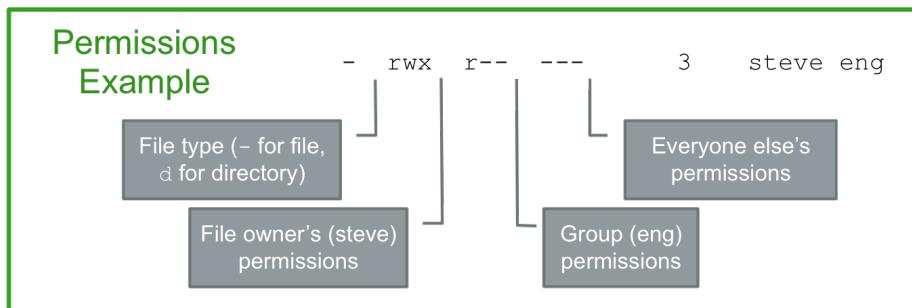
The next column is the file size, in bytes. Directories are always zero because directory information is stored as file system metadata on the HDFS NameNodes, and not on the HDFS DataNodes.

The next column indicates the file or directory's last modification time. This can be used, for example, by the HDFS shell command `hdfs dfs -ls -t <dir>` to sort the contents of a directory by modification times.

The final column lists the file or directory name.

HDFS Permissions

Permission	Authorized Directory Actions	Authorized File Actions
r = read	View (list) directory contents	View file contents
w = write	Create or delete files or subdirectories	Write, or append to, file contents
x = execute	Access a directory	Ignored for HDFS



Permissions are applied according to the most specific user class applicable to a user.

HDFS permissions are similar to Linux or UNIX file system permissions. There are read, write, and execute permissions and these are applied to files and directories. However, permissions on files and directories authorize different actions because files and directories are different.

Read permissions on a directory allow a user to view the directory's contents. The contents would be a list of files and subdirectories. Write permissions on a directory allow a user to create or delete files and subdirectories. Execute permissions on a directory allow the user to access the directory. For example, consider /dir1/dir2 where a user has no execute permission on dir1. The user would not be able to access dir1, which also means that the user could not "move through" dir1 to get to dir2 either.

Read permissions on a file allow a user to view the contents of a file. Write permissions allow a user to write, or append, contents to a file. Write permissions on a file do not permit a user to delete a file. Execute permissions for files in HDFS are not applicable because HDFS does not permit executable files. If a file is given execute permission, it is ignored by HDFS.

HDFS maintains three sets of read-write-execute permissions, one for each distinct user class. There are three user classes, owner, group, and other. The first set of permissions applies to the file or directory's owner. The second set applies to the group owner, while the third set applies to everyone else. HDFS applies permissions according to the most specific user class applicable to a specific user.

In the example, the user root would have read, write, and execute permissions on the file. Anyone who is a member of the group eng would have only read permission. Everyone else would not have any permissions. If a user was both root and a member of the group eng, the more specific file owner permissions would apply.

HDFS Home Directories

Only the Hive application can write to its home directory.

Only Saad can write to his home directory.

```
[hdfs@node1 ~]$ hdfs dfs -ls /user
Found 9 items
drwxrwx---  - ambari-qa  hdfs          0 2015-05-15 12:40 /user/ambari-qa
drwxr-xr-x  - hcat      hdfs          0 2015-05-15 12:41 /user/hcat
drwx-----  - hive       hdfs          0 2015-05-15 12:37 /user/hive
drwxr-xr-x  - jason     eng           0 2015-05-18 07:58 /user/jason
drwxr-xr-x  - may       sales          0 2015-05-18 07:58 /user/may
drwxrwxr-x  - oozie     hdfs          0 2015-05-15 12:38 /user/oozie
drwxr-xr-x  - root      root           0 2015-05-18 05:43 /user/root
drwxr-xr-x  - saad      sales          0 2015-05-18 07:58 /user/saad
drwxr-xr-x  - steve    eng            0 2015-05-18 07:58 /user/steve
```

Members of the sales group have read-only access to Saad's home directory.

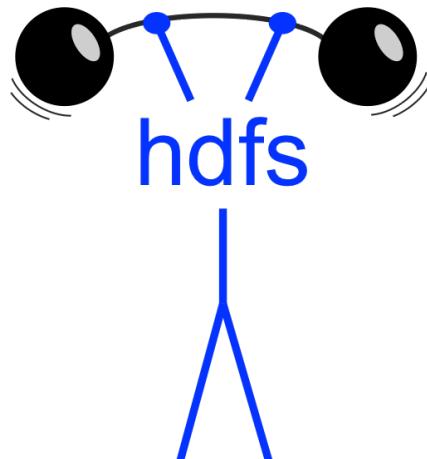
User and Application Home Directories

Home directories are used in concert with HDFS permissions to control data access. An HDFS administrator can create a home directory for each user. Some applications, like Hive, also create home directories.

In the example, only Saad has read-write access his home directory. This means that only the HDFS superuser and Saad can create files or subdirectories in Saad's home directory. The HDFS superuser is described later.

Once Saad has his own home directory, he can create files and subdirectories with different permissions that permit or deny access to other users and groups.

HDFS Superuser



Superuser has Special Admin Privileges

Many services in Hadoop have the concept of a superuser. This user is given special privileges for a specific service. It is analogous to the root account in Linux or the Administrator account in Windows. The common HDFS service superuser is the hdfs user account. The hdfs user can perform any HDFS action.

The HDFS superuser account is determined by the `dfs.cluster.administrators` property in the `hdfs-site.xml` file. To assume HDFS superuser privileges, use the `su - hdfs` command. The `su` command opens a new sub-shell in which the user has the `hdfs` user account privileges. Depending on the user, the `su` command might prompt for the `hdfs` account's password. By default the `hdfs` account does not have a password, which means only the Linux root account may use `su` to assume HDFS superuser privileges.

To return to normal user privileges, use the `exit` command. The `exit` command closes the sub-shell and returns to user to their normal shell with their normal privileges.

HDFS Shell Operations

The HDFS Client software includes the HDFS Shell. The HDFS Shell is used to manage HDFS files and directories. It can also manage files and directories in several other file system types. File and directory path arguments can be prefaced by URLs like `file://`, `swift://`, or `s3://` in order to manage files and directories in these file systems.

The HDFS Shell is accessed using either the older `hadoop fs` command syntax or the newer `hdfs dfs` syntax. If `hadoop fs` is used it operates as expected, but the command output includes a report stating that the command has been deprecated and to use `hdfs dfs` instead.

The basic syntax is `hdfs dfs [generic_options] [command_options] [path]`. The HDFS Shell includes extensive online help that describes its various options.

- Typing `hdfs dfs` with no options displays command usage information.
- Typing `hdfs dfs -help` displays online manual information.
- Typing `hdfs dfs -help <command>` displays detailed `<command>` help information.

Note

Many Hadoop files are extremely large. Using HDFS Shell commands on such large files might take significant time. For example, commands to upload, download, or view large files might execute for a long time.

Creating and Viewing Directories

The HDFS Shell includes the `-mkdir` and `-ls` commands for creating and listing the contents of directories. In all cases, a user would require permissions to create or view a directory or the commands will fail. A few commands and options are illustrated here.

- Make a directory in a specific path:
`hdfs dfs -mkdir /user/steve/dir1`
- Make a series of directories in a specific path:
`hdfs dfs -mkdir -p /user/steve/dir3/dir4`
Option `-p` would create `dir3`, if necessary, before creating `dir4`
- List the contents of a directory:
`hdfs dfs -ls /user/steve`

- List the contents of a directory recursively:
`hdfs dfs -ls -R /user/steve`

Home and Current Working Directory

Administrators familiar with Linux and Windows file systems will notice some similarities and differences when working with HDFS.

One of the similarities is the concept of a user's home directory. For example, assume that the user steve has a home directory of `/user/steve`. If this is true then the following commands are equivalent, if run as the user steve:

```
hdfs dfs -mkdir /user/steve/dir1
hdfs dfs -mkdir dir1
```

One of the primary differences is that HDFS does not have the concept of a current working directory. Because of this, it also has no "change directory" command. For example, assume that steve wanted to create a directory `dir2` in the path `/user/steve/dir1/`. The first command below would work as expected while the second command would not.

```
hdfs dfs -mkdir /user/steve/dir1/dir2
hdfs dfs -mkdir dir2
```

The second command would create `dir2` in `/user/steve` because there is no change directory command to make `/user/steve/dir1` the current working directory.

Creating Files

Assuming that a user has the necessary HDFS permissions, files can be created in HDFS.

- `-put`
Copies files from the local file system to HDFS. If the user steve runs `hdfs dfs -put /home/steve/fileA fileA`, it copies `fileA` from the local file system to the steve's home directory in the HDFS file system.
- `-appendToFile`
copies one or more source files on the local file system and appends them to a single file in HDFS. The HDFS file will be created if it does not exist. The command `hdfs dfs -appendToFile /home/steve/fileB fileA` copies `fileB` to HDFS and appends it to `fileA` in the user's home directory.

Creating a file larger than the default HDFS block size causes HDFS to automatically spread the file across multiple DataNodes. The default data block size is 128 MB but can be changed. Each data block is also replicated. The default replication factor is three. This means that a 10 GB file would consume 120 data blocks spread across multiple DataNodes. (40 blocks times a replication factor of 3 = 120 blocks) DataNodes, data block placement, and replication are described in detail in another lesson.

Viewing File Content

Some files May be binary and not usefully viewable.

- View the contents of a text file:
`hdfs dfs -cat fileA`
- Hadoop files are commonly very large and it might not be practical to view an entire large file.

- View only the last 1 KB of a file:
`hdfs dfs -tail fileB`

Copying and Moving Files

The HDFS Shell also allows a user to copy and move files.

- **-put**
Copies a file from the local file system to HDFS.
The command `hdfs dfs -put /home/steve/fileA /user/steve/fileA` copies fileA from the local file system to HDFS.
- The **-get** command works in reverse, copying a file from HDFS to the local file system. The command `hdfs dfs -get /user/steve/fileB /home/steve/fileB` copies fileB from HDFS to the local file system.
- The **-mv** command moves files inside HDFS. The `hdfs dfs -mv /user/steve/fileC /user/steve/dir1/fileC` moves fileC from steve's home directory to the dir1 subdirectory.
- When would the command `hdfs dfs -mv fileC dir1/fileC` be equivalent to the previous command?
- Answer: Assuming the current user is steve, the commands are equivalent because the second command uses valid path names relative to the user steve's home directory.
- The **-mv** command can also move multiple files at once, as long as the last argument is a directory name. The command `hdfs dfs -mv fileD fileE dir1` moves fileD and fileE to the dir1 directory.
- The **-getmerge** command copies and merges two or more HDFS files into a single file on the local file system. The command `hdfs dfs -getmerge fileF fileG /home/steve/combineFG` copies fileF and fileG from HDFS and merges them into single combinedFG file on the local file system.

Deleting Files and Directories

If you have the proper permissions, you may also remove files and directories.

- Remove one or more files from a home directory in HDFS:
`hdfs dfs -rm fileA`
`hdfs dfs -rm fileB fileC`
- Remove an empty directory (dir2) in HDFS:
`hdfs dfs -rmdir /user/steve/dir1/dir2`
- Recursively remove an entire directory (dir1) hierarchy:
`hdfs dfs -rm -R /user/steve/dir1`
Directory and its hierarchy are removed.

HDFS Trash Feature

Ambari installs HDP with the HDFS trash feature enabled. The trash feature helps when files or directories are accidentally deleted by a user. The trash feature only protects files and directories that have been deleted when using the HDFS Shell or the Ambari Files View. It does not protect HDFS delete actions taken using the Java API, WebHDFS, the HDFS NFS Gateway, or HUE.

With trash enabled, files or directories that have been deleted are temporarily moved to the `.Trash` directory in a user's home directory. The entire path of the file or directory is recreated beneath `.Trash/Current`. For example, deleting `/user/steve/dir1/fileA` would result in a trash path of `/user/steve/.Trash/Current/user/steve/dir1/fileA`. A file or directory can be recovered by moving it from `.Trash`.

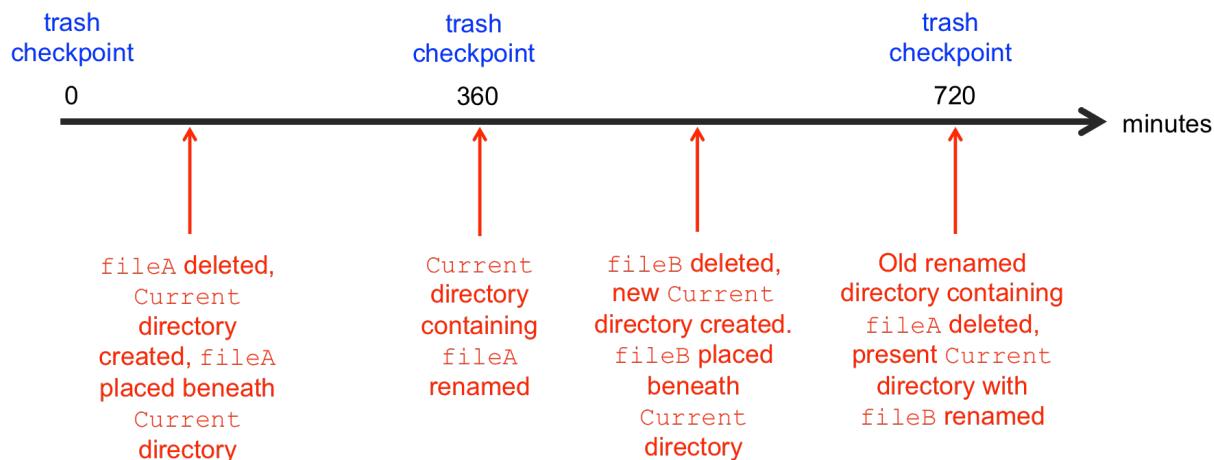
Trash is configured by two properties, `fs.trash.checkpoint.interval` in `core-default.xml` and `fs.trash.interval` in `core-site.xml`.

The `fs.trash.checkpoint.interval` property determines how often the NameNode should checkpoint the `.Trash` directory. The default value of 0 means that checkpoint operations should occur at the frequency determined by the `fs.trash.interval` property in `core-site.xml`.

The `fs.trash.interval` property determines how often checkpoints in the `.Trash` directory should be removed. A value of 0 disables trash. The HDP default value is 360 minutes.

The HDFS Shell `-rm` command includes a `-skipTrash` option that instructs the Shell not to move deleted files and directories to `.Trash`.

Trash Operation



Example: `/user/steve/.Trash/Current` renamed to `/user/steve/.Trash/150518175000`

Trash Operation

The `fs.trash.checkpoint.interval` determines the number of minutes between trash checkpoints. If zero, the value is set to the value of `fs.trash.interval`. Zero is the HDP default. The number for `fs.trash.checkpoint.interval` should be smaller than or equal to `fs.trash.interval`.

Every time the checkpointer runs, it renames the `.Trash/Current` directory to a new numeric name. For example, `.Trash/Current` could be renamed to `.Trash/150518175000`. When new files or directories are deleted, HDFS creates a new `.Trash/Current` directory to hold them.

How long the older and now renamed checkpoint directory—with its deleted files and directories—is retained is determined by the `fs.trash.interval` property in `core-site.xml`. It determines the number of minutes after which the checkpoint directory gets deleted. If zero, the trash feature is disabled. The HDP default is 360 minutes. It is important to note that it is not the individual files and directories that are older than the `fs.trash.interval` that are deleted, but it is the checkpoint directory that is older than the `fs.trash.interval` that is deleted.

The `fs.trash.interval` may be configured both on the server and the client. If trash is disabled on the server side then the client side configuration is checked. If trash is enabled on the server side then the value configured on the server is used and the client configuration value is ignored.

Overriding HDFS Default Properties

As a review, the HDFS Shell syntax is `hdfs dfs [generic_options] [command_options] [path]`. A common generic option is `-D <property>=<value>`. This option is used to override default HDFS properties. HDFS properties are commonly set in `hdfs-default.xml` or `hdfs-site.xml`.

One example is `-D dfs.blocksize=<N>`, which overrides the default 128 MB HDFS data block size when a file is written to HDFS. For example, the command `hdfs dfs -D dfs.blocksize=1048576 -put /home/steve/fileA fileA` writes `fileA` to a user's HDFS home directory with a data block size of one megabyte. Block sizes that are not at least one megabyte are invalid. Any value over one megabyte must be a multiple of 512 bytes.

Another example is `-D dfs.replication=<N>`, which overrides the default replication factor of three. For example, the command `hdfs dfs -D dfs.replication=5 -put /home/steve/fileB fileB` writes `fileB` to a user's HDFS home directory, creating five instances of each data block. There must be at least five DataNodes in the cluster because a single DataNode is not allowed to contain two of the same data block.

Changing Ownership

The HDFS Shell includes commands to change a file's or directory's owner and group membership.

- Change a file or directory's owner:
`hdfs dfs -chown danielle /data/weblogs/fileA`
The command requires HDFS superuser privileges.
- Change a file or directory's group membership:
`hdfs dfs -chgrp hdfs fileB`
The user running `-chgrp` must be a member of the group.
- Change owner and group membership simultaneously:
`hdfs dfs -chown hcat:hdfs /data/weblogs/fileC`
The command requires HDFS superuser privileges.
- Change owner and group membership recursively for a directory:
`hdfs dfs -chown -R hcat:hdfs /data/weblogs/dir1`

*Changing File and Directory Permissions***Generating Octal Values****Some Examples...**

Owner	Group	Others	Octal Permissions
rwx	rwx	rwx	777
rwx	r - x	---	750
rw -	rw -	r --	664
rw -	r --	---	640

- The syntax is the same for changing permissions on a file or directory.
- The difference is the effect of the execute permission:
 - Valid for a directory
 - Ignored for a file
- These examples use octal numbers:


```
hdfs dfs -chmod 755 dir1
hdfs dfs -chmod 644 fileA
hdfs dfs -chmod -R 755 dir1
```

Changing File and Directory Permissions

Users may change the permissions assigned to their files and directories. The HDFS superuser may change the permissions assigned to any HDFS file or directory. The command to change permissions is `-chmod`.

The same command syntax is used to change permissions on a file or directory. The difference is how execute permissions are interpreted on files and directories. Execute permissions are valid on a directory but ignored on a file. HDFS does not support the concept of executable files.

The `-chmod` command can use octal or symbolic notation to change permissions. Octal notation is illustrated here. Once understood, the octal offers a more compact syntax. Octal numbers include the numbers 0-7. The illustration shows how octal numbers are used to represent a combination of read, write, and execute permissions.

Read permission is assigned a value of four. Write permission is assigned a value of two. Execute permission is assigned a value of one. If these octal values are added together they total a maximum value of seven. To express it another way, the octal value of seven is used to represent the combination of read, write, and execute permissions. As a further example, the octal value of six represents the combination of only read and write permissions.

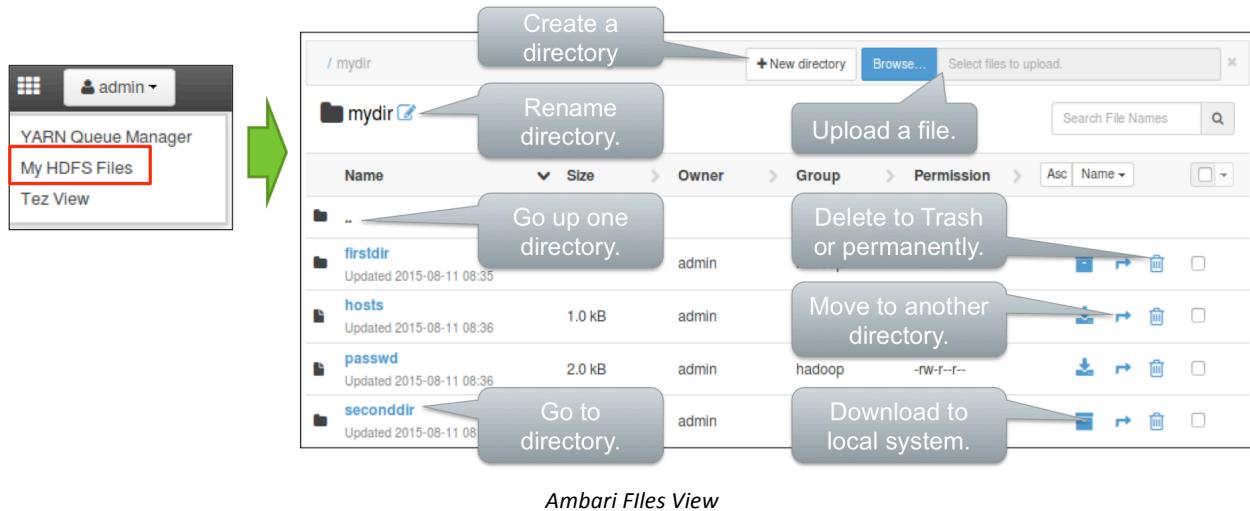
Each file and directory has a combination of owner, group, and other permissions. The owner's assigned read, write, and execute permissions can be represented by an octal value. Octal values can also represent the combination of permissions assigned to the group and others.

The illustration shows how three octal numbers can be used to represent the combination of read, write, and execute permissions assigned to the owner, group, and others for a specific file or directory.

Two examples of the command syntax are:

```
hdfs dfs -chmod 755 dir1
hdfs dfs -chmod 644 fileA.
```

Ambari Files View



Ambari Files View

The Ambari Files View is an Ambari Web UI plug-in providing a graphical user interface to HDFS files and directories.

The Files View can create a directory, rename a directory, browse to and list a directory, download a zipped directory, and delete a directory. It can also upload a file, list files, open files, download files, and delete a file.

When deleting a directory or file, the user has the choice of permanently deleting or not. If the user chooses to permanently delete, the file or directory is immediately removed from HDFS storage. If the user chooses not to permanently delete, the file or directory is moved to their .Trash directory, assuming that HDFS trash has been enabled. The file or directory would not be removed until the normal trash checkpoint interval.

Configuring HDFS for Files View

The 'Add Property' dialog box is shown. It has two main sections: 'Type' and 'Properties'. The 'Type' section contains a dropdown menu set to 'core-site.xml' and two small icons. The 'Properties' section contains a text input field with the following key-value pairs separated by newlines:


```
hadoop.proxyuser.root.groups=*
hadoop.proxyuser.root.hosts=*
```

 At the bottom right are 'Cancel' and 'Add' buttons.

Add Property Dialog

In order to use the Ambari Files View, you need to setup an HDFS proxy user for the Ambari daemon account.

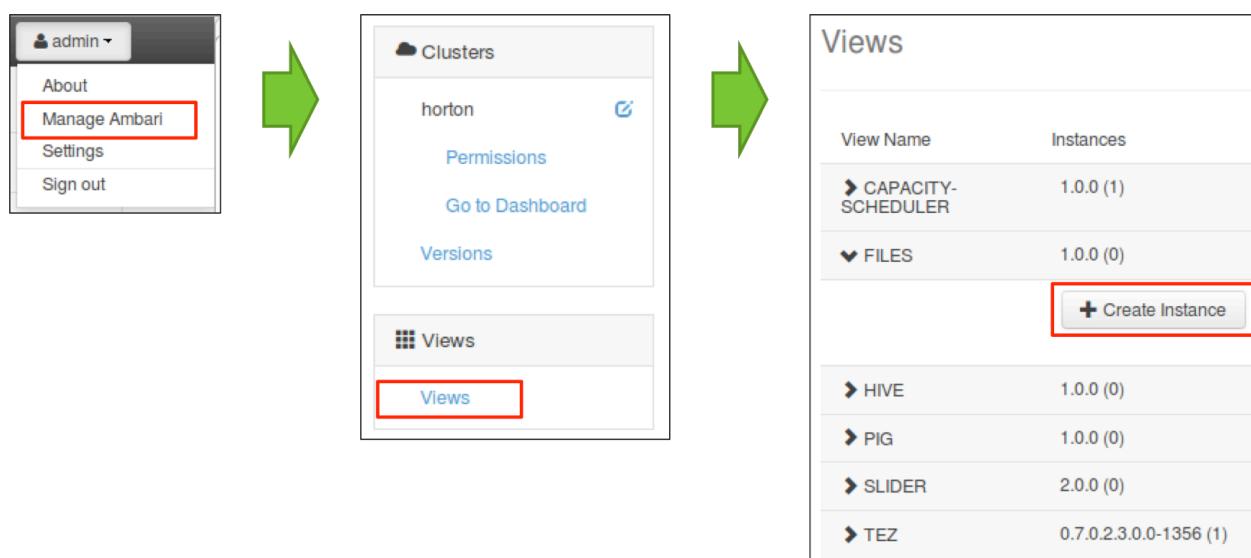
For example, if the AmbariServer daemon is running as root, you configure a proxy user for root in `core-site.xml` in the Ambari Web UI by selecting:

Services > HDFS > Configs > Advanced > Custom coresite > Add Property

Add the properties and values `hadoop.proxyuser.root.groups=*` and `hadoop.proxyuser.root.hosts=*` to the file as shown above. Save the changes and restart any affected services as indicated in the Ambari Web UI.

This assumes that Ambari was installed and runs as the root user account. If this is not the case, change root to the appropriate user account.

Creating a View Instance



Creating a View Instance

As an Ambari administrator, you must create an instance of the File View in order for users to be able to use it.

To create an instance:

- 1) Select **Manage Ambari** from the **admin** menu button
- 2) In the window select **Views**
- 3) In the next window expand **FILES**
- 4) Then click **Create Instance**

Filling out the Create Instance Form

Using HDFS Storage

The Create View Instance Form

Fill out the Create Instance form.

Select a version. Only version 1.0.0 was available at the time of this writing.

Enter a unique Instance Name. It cannot contain spaces or some special characters like a period.

Enter a Display Name. The Display Name is what appears in the Ambari Web UI on the View drop-down menu. Enter a brief optional description.

Click **Save** (not shown) to save the configuration.

Adding Permissions to Use the View Instance

View Name	Instances
► CAPACITY-SCHEDULER	1.0.0 (1)
▼ FILES	1.0.0 (1)
My HDFS Files	1.0.0

+ Create Instance

Permissions

Permission	Grant permission to these users	Grant permission to these groups
Use	Add User	Add Group

Adding Permissions to Use the View Instance

After a View instance has been created it can be edited and updated. One of the more important reasons for editing an instance is to grant permissions to use the instance to other users and groups.

Note

You can also delete a View instance using this window.

Browse HDFS Using the NameNode UI

The screenshot shows a browser window with the URL `node1:50070/dfshealth.html#tab-overview`. The top navigation bar has tabs for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The Utilities dropdown is open, showing 'Browse the file system' and 'Logs'.

Browsing HDFS Using the NameNode UI

Another method to view HDFS directory contents is to use the HDFS NameNode's Web-based NameNode UI.

There are two ways to access the NameNode UI

- Through the Ambari Web UI. In the Ambari Web UI:
Select the **Services** page
Select the **HDFS** service
Select **NameNode UI** from the **Quick Links** menu
- Enter the URL `http://<NameNode_host>:8080` in a browser window.

When the NameNode UI opens, select **Browse the file system** from the **Utilities** menu.

View Directory Contents

The screenshot shows the 'Browse Directory' window. It has a text input field with '/user/root' and a 'Go!' button. A callout points to the input field with the text 'Type a path.' Another callout points to a table row with 'dir1' and 'dir3' with the text 'Click a directory to open it.'. A callout points to a table row with 'textfile.txt' with the text 'Click a file.'

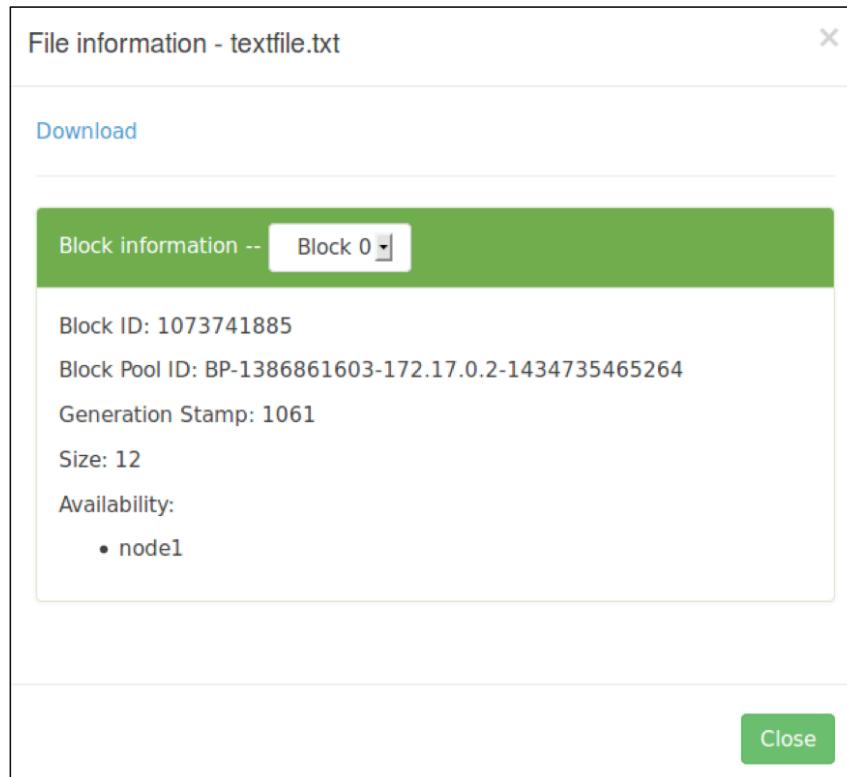
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x+	root	root	0 B	6/30/2015, 5:37:06 PM	0	0 B	dir1
drwxr-xr-x	root	root	0 B	6/30/2015, 5:10:30 PM	0	0 B	dir3
-rw-rw-r--+	root	root	1.48 KB	6/30/2015, 4:14:05 PM	3	128 MB	passwd
-rw-r--r--	root	root	12 B	7/8/2015, 1:54:35 PM	3	128 MB	textfile.txt

- User permissions still determine read privileges.

The Browse Directory Window

The Browse Directory window enables you to browse and list directory contents. You can move between directories by typing path names in the text box or by clicking the hypertext links in the Name column. Who you are when you launched the browser, along with your HDFS permissions, still determines what you are allowed to see.

File Information and Download

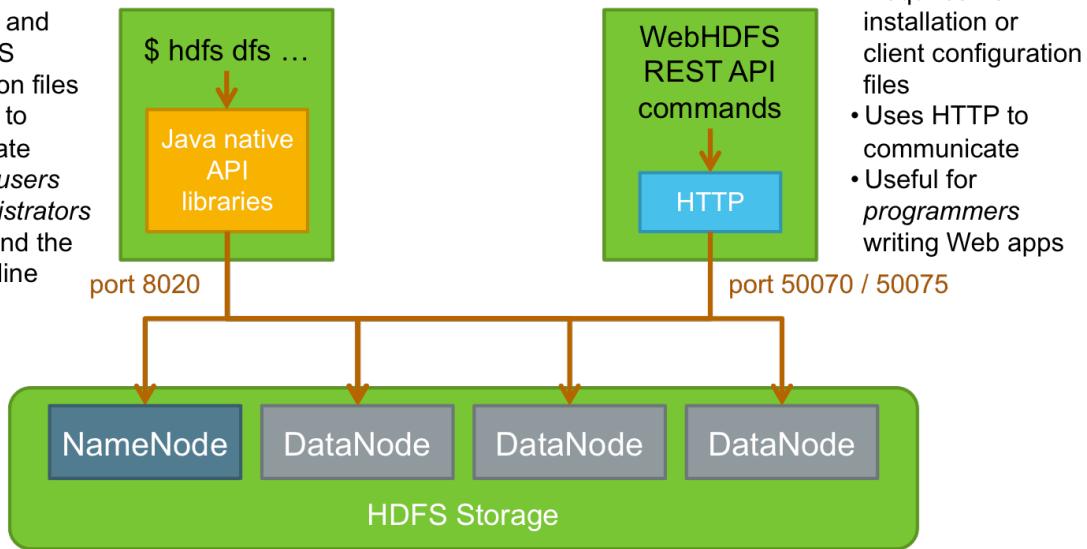


File Information in NameNode UI

Clicking on a file name in the NameNode UI opens a window with file information, as shown here. It also enables a user to download the file to the local system where the browser is running. Files could be extremely large and therefore take significant time and resources to download.

Using WebHDFS

- Requires installation and client HDFS configuration files
- Uses RPC to communicate
- Useful for *users* and *administrators* in scripts and the command line



Java Native API versus WebHDFS Access

Hadoop includes the Java native API libraries to support HDFS operations like creating, renaming, and deleting files, or creating, listing, or deleting directories. For example, the `hdfs dfs` command is just one of many commands that are implemented using the Java native API libraries. However, to use these commands you must install the Java native API libraries and also download and configure the HDFS configuration files, like `hdfs-site.xml` for example.

WebHDFS was created to enable HDFS access without having to install and configure the Java native API libraries. WebHDFS is a HTTP REST API. WebHDFS uses HTTP operations like GET, POST, PUT, and DELETE for file access and administration.

The HDFS Shell is useful for users or administrators. HDFS Shell commands can be used directly from a command-line prompt or used in shell scripts or programs. WebHDFS is primarily useful for programmers writing HDFS Web applications that will be used by users, administrators, or other programs.

A program using the Java native API or the WebHDFS REST API must have access to the NameNode and the DataNodes. Access to the NameNode is required because the NameNode maintains all of the file system namespace and metadata like file names, directory names, and the file system's hierarchical structure. The DataNodes maintain the actual file data blocks.

The Java native API libraries use RPC over port 8020 while the WebHDFS REST API uses port 50070 to connect to the NameNode and port 50075 to connect to a DataNode.

WebHDFS Features

WebHDFS includes a number of features:

- WebHDFS supports all HDFS file administration operations. This includes reading and writing files, viewing directories, granting permissions, configuring the replication factor, configuring quotas and access control lists, and so on.

- WebHDFS permits clients to access HDFS from multiple languages, other than just Java, and without having to install software. You can also use common tools like `curl` or `wget` to access HDFS.
- WebHDFS calls are faster than the `hdfs dfs` calls when the client is remote to the cluster.
- WebHDFS requires no additional servers because WebHDFS is built into the NameNode and DataNode.
- WebHDFS uses the full bandwidth of the Hadoop cluster for streaming data. File read and file operations are redirected to the appropriate DataNodes.
- WebHDFS is compatible with Kerberos authentication. It uses the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO), which extends Kerberos to Web applications.
- WebHDFS is completely Apache open source. Hortonworks contributed the code to Apache Hadoop as a first-class, built-in Hadoop component.

WebHDFS Enabled by Default

- To verify that WebHDFS is enabled, check either the `hdfs-site.xml` file or Ambari.

In `hdfs-site.xml`

```
<property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
</property>
```

In Ambari

The screenshot shows the Ambari UI for managing HDFS configurations. At the top, there are tabs for 'Settings' and 'Advanced'. Under 'Advanced', there are sections for 'NameNode', 'Secondary NameNode', 'DataNode', and 'General'. In the 'General' section, there is a configuration named 'WebHDFS enabled' with a value of 'true'. This configuration is highlighted with a red border around its row. There are also icons for edit, lock, and delete next to the configuration entry.

WebHDFS is Enabled by Default

WebHDFS is enabled by default when you install the Hortonworks Data Platform.

To verify that WebHDFS is enabled, you can:

- Check the `hdfs-site.xml` file
The property `dfs.webhdfs.enabled` must be set to true
- Use Ambari
Ensure that WebHDFS enabled is selected for HDFS at **Services > Config > Advanced > General**

Additional WebHDFS Properties

```
<property>
    <name>dfs.web.authentication.kerberos.principal</name>
    <value>HTTP/$<Fully_Qualified_Domain_Name>@$<Realm_Name>.COM</value>
</property>
<property>
    <name>dfs.web.authentication.kerberos.keytab</name>
    <value>/etc/security/spnego.service.keytab</value>
</property>
```

If Kerberos is enabled, WebHDFS requires the configuration of two additional hdfs-site.xml properties.

The property names are

dfs.web.authentication.kerberos.principal and
dfs.web.authentication.kerberos.keytab.

WebHDFS Operations

HTTP GET	HTTP PUT	HTTP POST	HTTP DELETE
OPEN	CREATE	APPEND	DELETE
GETFILESTATUS	MKDIRS		
LISTSTATUS	RENAME		
GETCONTENTSUMMARY	SETREPLICATION		
GETFILECHECKSUM	SETOWNER		
GETHOMEDIRECTORY	SETPERMISSION		
GETDELEGATIONTOKEN	SETTIMES		
	RENEWDELEGATIONTOKEN		
	CANCELDELEGATIONTOKEN		

WebHDFS Operations

WebHDFS includes a number of built-in operations. They are listed in the table. These operations, when combined with the proper URLs, enable HDFS file access and administration.

All WebHDFS operation URLs use the syntax:

`http://<NameNode>:50070/webhdfs/v1/<path>?op=operation_and_arguments>`

WebHDFS Examples

- The curl command can be used to test WebHDFS operations.
- Creating a directory named mydata:
`curl -i -X PUT "http://<NameNode>:50070/webhdfs/v1/web/mydata?op=MKDIRS&user.name=jason"`
- Listing a directory named mydata:
`curl -i "http://<NameNode>:50070/webhdfs/v1/web/mydata?op=LISTSTATUS&user.name=jason"`

Using HDFS Storage

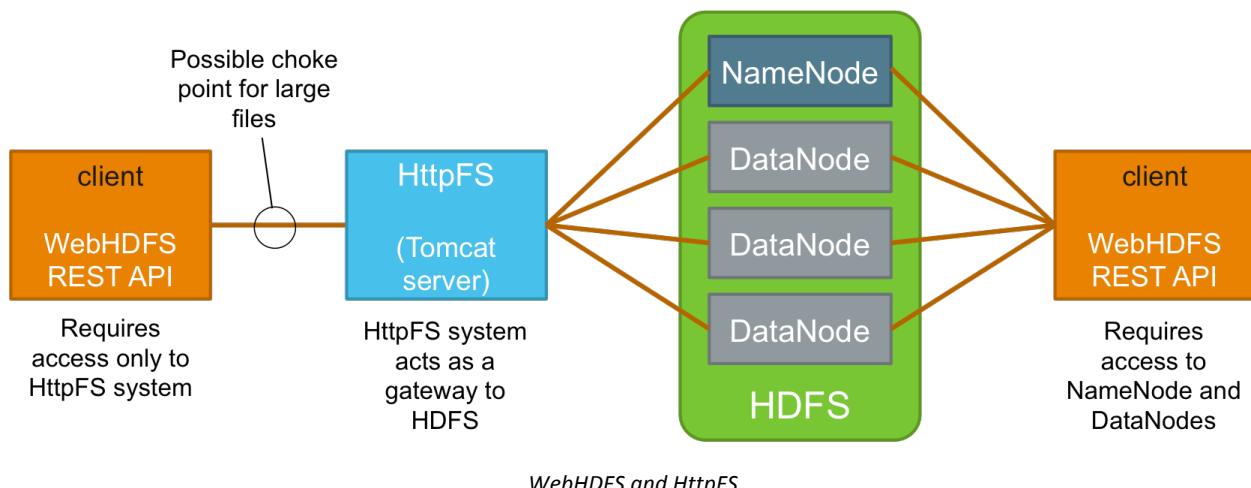
- **Reading a file named webdata:**
curl -i -L
"http://<NameNode>:50070/webhdfs/v1/web/mydata/webdata?op=OPEN&user.name=jason"
- **Writing a file is a two-step process.**
- **Create a file name on the NameNode.**
- **Write the file contents to a DataNode.**
WebHDFS ensures that files larger than an HDFS block are written across multiple DataNodes.
- **Create a file by creating a file name on the NameNode:**
curl -i -X PUT
"http://<NameNode>:50070/webhdfs/v1/web/mydata/largefile.json?op=CREATE".
The output from this command includes the URL used to write data to the file.
- **Write to the file by sending data to the DataNodes:**
curl -i -PUT -T largefile.json
"http://<DataNode>:50075/webhdfs/v1/web/mydata/largefile.json?op=CREATE&user.name=root&namenoderpcaddress=node1:8020&overwrite=false"
- **The curl command can perform a write operation using a single command that performs both steps:**
curl -I -X PUT largefile.json -L
"http://<NameNode>:50070/webhdfs/v1/web/mydata/largefile.json?op=CREATE&user.name=root"

WebHDFS Authentication

When security is **off** (Kerberos not enabled), the user that is authenticated is the user set in the `user.name=<name>` included in the URL. If `user.name` is not included in the URL, the server may either set the authenticated user to a default Web user, if there is one, or return an error response.

When security is **on** (Kerberos is enabled), authentication is performed by either Hadoop delegation token or Kerberos SPNEGO. The user encoded in the `delegation=<token>` argument is authenticated, or the user is authenticated by SPNEGO.

WebHDFS and HttpFS



HttpFS is a separate service from Hadoop NameNode. HttpFS is a server that provides a REST HTTP gateway supporting all HDFS read and write operations. HttpFS itself is Java Web application and runs using a preconfigured Tomcat server bundled with the HttpFS binary distribution. It is interoperable with the WebHDFS REST API.

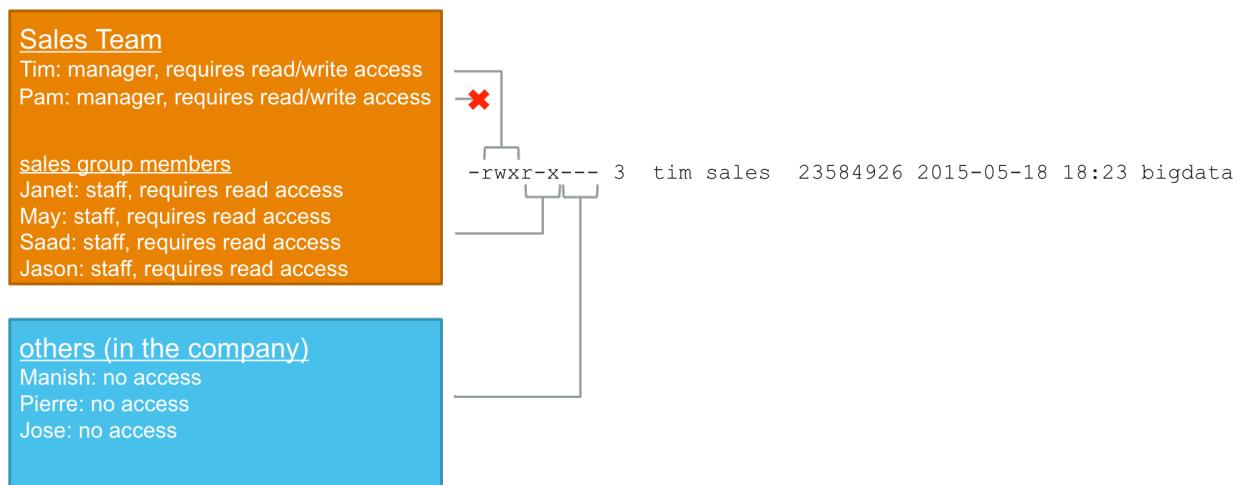
The main difference between WebHDFS access and HttpFS access is that with WebHDFS, the client system must have access to the NameNode and all DataNodes. With HttpFS, the client only needs access to the HttpFS system. HttpFS can be useful when access to a cluster is protected by a firewall. A firewall can be configured to allow HttpFS access to the cluster. This enables HttpFS to act as a gateway to the cluster.

In addition, HttpFS is a separate installation. After installation you must configure the `httpfs-site.xml`, `core-site.xml`, and `hdfs-site.xml` files. See the online HttpFS documentation for detailed installation and configuration instructions.

HttpFS can be used to transfer data between clusters running different versions of Hadoop, overcoming incompatible RPC version issues.

HttpFS can be used to access data in HDFS using HTTP utilities (such as `curl` and `wget`) and HTTP libraries from Perl and languages than Java.

Using HDFS Access Control Lists (ACLs)



Before the implementation of access control lists (ACLs), the HDFS permission model was equivalent to traditional UNIX permission model. In this model, permissions for each file or directory are managed by a set of three distinct user classes: owner, group, and other. There are three permissions for each user class: read, write, and execute. When a user attempts to access a file system object, HDFS enforces permissions according to the most specific user class applicable to that user.

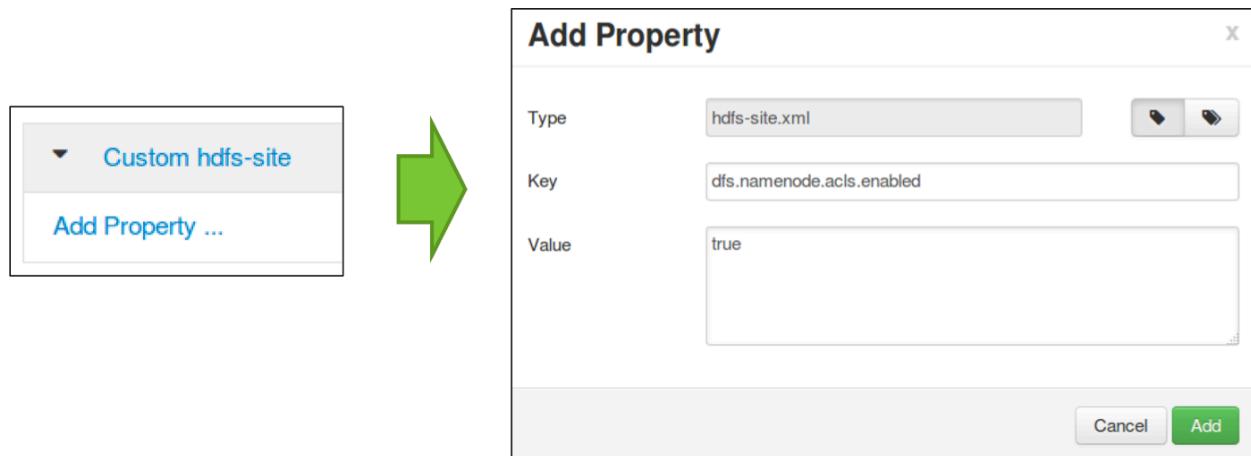
This model can sufficiently address a large number of security requirements, but not every requirement. ACLs extend the HDFS permission model to support more granular file access based on arbitrary combinations of users and groups.

In the example, Tim is the manager of the sales team. Tim requires read and write access to the file named bigdata. Members of the sales team only require read access to the file. Anyone outside of the sales team should not be able to read the file. This is easily accommodated by the traditional HDFS permissions. Tim is made the owner the file and the owner permissions are set to read and write. File execute permissions are not evaluated in HDFS. The file is assigned to the group sales, of which Janet, May, Saad, and Jason are members. Group permissions are configured for read access. Anyone else in the company falls into the other user class with permissions set to deny read access to the file.

This works until a second manager, Pam, joins the sales team. There is no way, using traditional HDFS permissions, to assign Pam read and write access to the file. This is where HDFS ACLs can help.

Hortonworks recommends the use of traditional permissions whenever possible, adding a few ACL entries when necessary. ACL entries, like regular permissions, are maintained on the NameNode and add overhead that consumes a small amount of NameNode CPU and memory resources.

Enabling HDFS ACLs



Ambari Web UI Add Property Window

HDFS ACLs are not enabled by default. When ACLs are disabled, the NameNode rejects all attempts to set an ACL. Only the `dfs.namenode.acls.enabled` property needs to be configured as `true` in the `hdfs-site.xml` file in order to enable ACLs.

To enable HDFS ACLs using the Ambari Web UI, select **Services > HDFS > Configs > Advanced >** and then click **Add Property** beneath **Custom hdfs-site**. An Add Property window opens. Add the property name as the key and true as the value and click **Add**. Save your change and restart the HDFS service as indicated by Ambari.

Managing ACLs

- Two HDFS Shell commands are used to manage ACLs:
 - `-dfs dfs -setacl` :Set, modify, or delete file and directory ACL entries
 - `-dfs dfs -getacl` :View file and directory ACL entries
 - `-dfs dfs -help setacl | getacl`: Get online help information
- Apache Ranger can also manage HDFS ACL entries.

- How do you know if a file or directory has ACL entries?

`-dfs dfs -ls` output indicates the presence of an ACL entry.

```
-rwxr-x---+ 3  tim sales  23584926 2015-05-18 18:23 bigdata
```

Using `getacl` on Files

```
$ hdfs dfs -getacl fileB
# file: passwd
# owner: root
# group: root
user::rw-
user:jason:rw-
group::r-
group:sales:r--
mask::rw-
other::r--
```

Using `getacl` on Files to View Permission and ACL Entries

The `getacl` command displays the HDFS permissions and any existing ACL entries. If no ACL entries exist, then only permissions are displayed.

In the example for `fileB`, the owner of the file is the user `root` and has `rw-` permissions. The file's assigned group is the group `root` and has only `r--` permissions. The specific user `jason` has `rw-` permissions while the specific group `sales` has `r--` permissions. Any other user or group is assigned `r--` permissions.

Using `setfacl` on Files

- Set (remove existing and replace) both permissions and ACL entries using a single command:
`$ hdfs dfs -setfacl --set user::rw-,group::r--,other::---,user:steve:rw-,user:jason:rw- fileA`
 Sets owner, group, and other permissions and adds ACL entries for steve and jason
- Modify an existing ACL by adding a new entry for the group eng:
`$ hdfs dfs -setfacl -m group:eng:rw- fileA`
- Remove the specific ACL entry for the user jason:
`$ hdfs dfs -setfacl -x user:jason fileA`
- Remove all ACL entries, leaving only the base owner, group, and other permissions:
`$ hdfs dfs -setfacl -b fileA`
 The user steve and the group eng are removed.

Using `setfacl` on Directories

The `setfacl` command can also be used on directories, but with a few differences. For example, the `-R` recursive option can be used to set, modify, or remove permissions and ACL entries from an entire directory hierarchy.

`hdfs dfs -setfacl -R ... dir1` recursively sets, modifies, or removes ACL entries.

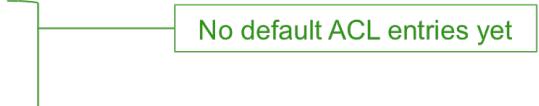
In addition there is also the separate concept of a default ACL. A default ACL can only be applied to a directory and not to a file. Default ACLs have no direct effect on permission checks for existing child files and directories, but instead define the permissions that new child files and directories will receive when they are created. Default ACLs are described in more detail later in this lesson.

Default Directory ACL Entries

Creating a Default Directory ACL

- Display permissions and ACL entries for a directory:

```
$ hdfs dfs -getfacl dir1
# file: dir1
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
```



- No default ACLs yet
- Add a default ACL entry:

```
$ hdfs dfs -setfacl -m default:user:jason:rw- dir1
```

- Creates default ACL entries for owner, group, and others too.

The `getfacl` command can also display the HDFS permissions and any existing ACL entries for directories, including any default ACL entries. If no ACL entries exist, then only standard HDFS permissions are displayed. The example displays the permissions for the directory `dir1`. It has no ACL entries.

A `-m` option can be used to add a default ACL entry to a directory. In the example for `dir1`, a default entry is added for the user `jason`. The user `jason` will have read and write permissions for any file or directory created beneath `dir1`.

In this example, the `getfacl` command was run after adding a default ACL entry for the user `jason`. Notice that in addition to a user entry for `jason`, default entries were also automatically created for the owner (user), group, and other user classes, along with a default mask. Any directory with default ACL entries must include default entries for the owner, group, and other user classes.

Listing Default Directory ACL Entries

- Display permissions and ACLs for a directory:

```
$ hdfs dfs -getfacl dir1
# file: dir1
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
default:user::rwx
default:user:jason:rwx-
default:group::r-x
default:mask::rwx
default:other::r-x
```

Only user:`jason` was specified so default ACLs were automatically created for the user, group, and other classes based on their current HDFS permissions.

Default mask generation is described later.

In this example, the `getfacl dfs -getfacl dir1` command was run after adding a default ACL entry for the user `jason`. Notice that in addition to a user entry for `jason`, default entries were also automatically created for the owner (user), group, and other user classes, along with a default mask. Any directory with default ACL entries must include default entries for the owner, group, and other user classes.

ACL masks are explained later.

Creating a File in a Directory with a Default ACL

- Create a file in a directory with a default ACL:

```
$ hdfs dfs -put /etc/passwd dir1
```

- View the file's permissions and ACLs:

```
$ hdfs dfs -getfacl dir1/passwd
# file: dir1/passwd
# owner: root
# group: root
user::rw-
user:jason:rw-
group::r-x
mask::r--
other::r--
```

#effective:r--
#effective:r--

Inherited from dir1's default ACL entries.

The mask::r-- entry is described later in this lesson.

When a file is created in a directory with default ACL entries, the file's permissions are inherited from the parent directory's default ACL entries.

For example, assume that the command `hdfs dfs -put /etc/passwd dir1` was run and that `dir1` has default ACL entries. The default ACL entries for `dir1` would be inherited as the permissions for the file `fileA` as shown in the illustration here.

The mask::r-- entry is described later in this lesson.

Creating a Directory in a Directory with a Default ACL

- Create a subdirectory of a directory with a default ACL:

- View the directory's permissions and ACLs:

```
$ hdfs dfs -getfacl dir1/dir2
# file: dir1/dir2
# owner: root
# group: root
user::rwx
user:jason:rw-
group::r-x
mask::r-x
other::r-x
default:user::rwx
default:user:jason:rw-
default:group::r-x
default:mask::rwx
default:other::r-x
```

Inherited from dir1's default ACL entries

Copied from dir1's default ACL entries

When a directory is created in a directory with default ACL entries, the child directory's permissions are inherited from the parent directory's default ACL entries. In addition, the parent directory's default ACL entries are copied to the child directory's default ACL entries.

For example, assume that the command `hdfs dfs -mkdir dir1/dir2` was run and that `dir1` has default ACL entries. The default ACL entries for `dir1` would be inherited as the permissions for the directory `dir2` as shown in the illustration here. In addition, the default ACL entries for the parent directory `dir1` would be copied and become the default ACL entries for the child directory `dir2`.

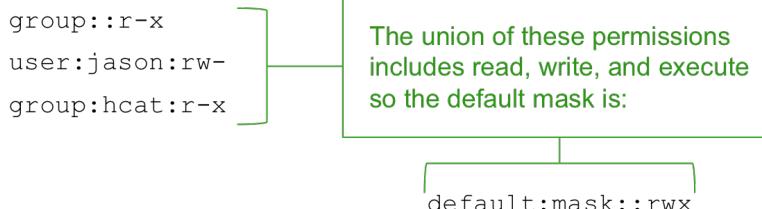
The `mask::r-x` and `default:mask::rwx` entries are described later in this lesson.

ACL Mask Types

There are two types of masks: *default* masks on directories and access masks on directories and files. In the output of a `-getfacl` command, a default mask is displayed as `default:mask:<perms>` while an access mask is displayed as `mask:<perms>`.

Default Mask

- The default mask helps to define the permissions and ACL entries inherited by child files and directories.
- A default mask can be explicitly defined by a user.
– `hdfs dfs -setfacl -m default:mask::r-- dir1`
- If not explicitly defined, a default mask is automatically generated.
– Based on the union of the permissions for the unnamed group, and any named users or groups.
– For example:



A default mask on a directory helps to define the permissions and ACLs inherited by child files and directories.

A default mask can be explicitly defined by a user. For example, the command `hdfs dfs -setfacl -m default:mask::r-- dir1` explicitly sets a default mask.

If a default mask is not specifically assigned by a user, HDFS generates a default mask for a directory based on the union of the permissions on all ACL entries that would be filtered by the mask. This includes the unnamed group, and any named users or named groups listed in the ACL.

For example, if the unnamed group had read permission, a named user had write permission, and a named group had execute permission, then the default mask would have read, write, and execute permissions.

Access Mask

The other type of mask is the access mask on a file or directory. The access mask is calculated using the default mask and the mode parameter. The process of generating an access mask is illustrated later in this lesson.

The access mask defines the effective permissions granted when accessing a file or directory. An access mask effects any named users, named groups, or the unnamed group. No unnamed group, named user, or named group may have greater effective permissions than are granted in the access mask. If the access mask has only read permission, then the unnamed group and any named users or groups are effectively granted only read permission, despite what their assigned permissions might be. For example, if the access mask is mask::r-- and there is an ACL of user:root:rw-, the effective permission for the root user is only r-- because the access mask grants at most only read permission.

The purpose of an access mask is to provide a user with a mechanism to quickly limit or restore the effective permissions for multiple users and groups using a single command. In the example here, the `hdfs dfs -setfacl-m mask::r-- fileA` command was used to change the access mask to no permissions. The effect would be to deny any permissions to the unnamed group, and any named users or named groups listed in ACL entries.

Changing the access mask does not change any assigned permissions, it only changes users' or groups' effective permissions. Assuming that all named users and groups were assigned at least read permissions, changing the mask to r-- would effectively result in read permissions for all the named users and groups. If some of these named users and groups also had write permissions, then changing the mask to rw- would effectively give these named users and groups read and write permissions.

Creating a File Affected by an Access Mask

- Create a file in a directory with a default ACL:
- View the file's permissions and ACLs:

```
$ hdfs dfs -getfacl dir1/passwd
# file: dir1/passwd
# owner: root
# group: root
user::rw-
user:jason:rw-    #effective:r--
group::r-x  #effective:r--
mask::r--
other::r--
```

The access mask r-- effectively limits the user jason and the unnamed group to read-only permissions.

When a file is created in a directory with default ACL entries, the file's permissions are affected by an access mask.

Assume that the command `hdfs dfs -put /etc/passwd dir1` was run and that `dir1` has default ACL entries. The default mask is used to help calculate an access mask, which in turn affects effective user and group permissions for the file. For example, in the illustration the access mask of r-- effectively limits the permissions of the user jason and the unnamed group.

Creating a Directory Affected by an Access Mask

- Create a subdirectory of a directory with a default ACL:
- View the directory's permissions and ACLs:

```
$ hdfs dfs -getfacl dir1/dir2
# file: dir1/dir2
# owner: root
# group: root
user::rwx
user:jason:rwx-          #effective:r--<-
group::r-x
mask::r-x                 access mask
other::r-x
default:user::rwx
default:user:jason:rwx-
default:group::r-x
default:mask::rwx
default:other::r-x
```

The access mask `r-x` effectively limits the user `jason` to read-only permissions but does not effectively limit the unnamed group.

When a directory is created in a directory with default ACL entries, the directory's permissions are affected by an access mask.

Assume that the command `hdfs dfs -mkdir dir1/dir2` was run and that `dir1` has default ACL entries. The default mask is used to help calculate an access mask, which in turn affects effective user and group permissions on the directory. For example, in the illustration the access mask of `r-x` effectively limits the permissions of the user `jason` but does not effectively limit the permissions of the unnamed group.

Calculating the Access Mask

Directories				Files			
default mask	<code>rwx</code>	<code>rwx</code>	<code>rwx</code>	default mask	<code>rwx</code>	<code>rwx</code>	<code>rwx</code>
755 mode parameter	<code>rwx</code>	<code>r-x</code>	<code>r-x</code>	644 mode parameter	<code>rw-</code>	<code>r--</code>	<code>r--</code>
calculated access mask	owner <code>rwx</code>	mask <code>r-x</code>	other <code>r-x</code>	calculated access mask	owner <code>rw-</code>	mask <code>r--</code>	other <code>r--</code>

The table illustrates the calculation of an access mask for a directory or a file. The directory's default mask is filtered by the mode parameter to calculate the access mask.

The mode parameter is calculated using the value of the `fs.permissions.umask-mode` property in the `hdfs-site.xml` file. The default property value is typically `022`. For directories, the value of `777` is filtered by the `umask-mode` value `022` to produce the mode parameter `755`. For files, the value `666` is filtered by the `umask-mode` value `022` to produce the mode parameter `644`.

In the table, the directory's default mask of `777` is filtered by the `755` mode parameter, which yields the access mask of `r-x`. It also yields the owner's `rwx` permissions and other's `r-x` permissions.

The file's default mask of `666` is filtered by the `644` mode parameter, which yields the access mask of `rw-`. It also yields the owner's `r--` permissions and other's `r--` permissions.

Knowledge Check Questions

- 1) Which file system would the command `hdfs dfs -ls file:///bin` access?
- 2) What permissions are necessary to delete a file in HDFS?
- 3) Which file contains the property that configures the HDFS superuser?
- 4) What is a potential issue with the `hdfs dfs -cat <file_name>` command?
- 5) The command `hdfs dfs -ls dir1` assumes `dir1` is in which directory?
- 6) When is the `<home_directory>/._Trash/Current` directory created?
- 7) Which permissions does the group have after running the command `hdfs dfs -chmod 750 dir1`?
- 8) Which permissions are required to make a file executable in HDFS?
- 9) True or false? User's with access to the NameNode UI are able to view the contents of any HDFS directory.
- 10) True or false? The Java native API and WebHDFS use RPC to connect to the NameNode.
- 11) True or false? WebHDFS is built into the NameNode and DataNode.
- 12) What is missing from this WebHDFS API prefix?
`http://<NameNode>:50070/webhdfs/`
- 13) Explain why writing a file with WebHDFS is a two-step process.
- 14) What is the main difference between HDFS client access through HttpFS and access through WebHDFS?
- 15) True or false? Hortonworks recommends the use of traditional permissions whenever possible, adding a few ACL entries when necessary.

- 16)True or false? HDFS ACLs are enabled by default.
- 17)Which command could be used to indicate the presence of an ACL?
- 18)Which command displays file and directory ACL entries?
- 19)List the two types of ACL masks.
- 20)What is the purpose of an access mask?

Knowledge Check Answers

1) Which file system would the command `hdfs dfs -ls file:///bin` access?
The local file system

2) What permissions are necessary to delete a file in HDFS?
Execute to access the directory and write to delete a file in the directory

3) Which file contains the property that configures the HDFS superuser?
hdfs-site.xml

4) What is a potential issue with the `hdfs dfs -cat <file_name>` command?
It displays the contents of a file, which might not be text, and might be far too large to practically display.

5) The command `hdfs dfs -ls dir1` assumes `dir1` is in which directory?
The user's home directory

6) When is the `<home_directory>/._Trash` directory created?
When a user deletes a file using the HDFS Shell and HDFS trash is enabled

7) Which permissions does the group have after running the command `hdfs dfs -chmod 750 dir1`?
Read and execute

8) Which permissions are required to make a file executable in HDFS?
Files are not executable in HDFS

9) True or false? User's with access to the NameNode UI are able to view the contents of any HDFS directory.
False

10) True or false? The Java native API and WebHDFS use RPC to connect to the NameNode.
False

11) True or false? WebHDFS is built into the NameNode and DataNode.
True

12) What is missing from this WebHDFS API prefix?
`http://<NameNode>:50070/webhdfs/`
v1

13) Explain why writing a file with WebHDFS is a two-step process.

First, the file name is written to the NameNode. As a result, the NameNode issues a temporary redirect that includes the WebHDFS URL necessary to write the contents of the file to the DataNodes. Next, the user or application must use the WebHDFS URL provided by the NameNode to write the file contents to the DataNodes. If the file is larger than a single HDFS data block, DataNode pipelining ensures that the file is written across multiple DataNodes for scalability and better reliability.

14) What is the main difference between HDFS client access through HttpFS and access through WebHDFS?

The main difference between WebHDFS access and HttpFS access is that with WebHDFS the client system must have access to the NameNode and all DataNodes. With HttpFS, the client only needs access to the HttpFS system. HttpFS can be useful when access to a cluster is protected by a firewall. A firewall can be configured to allow HttpFS access to the cluster. This enables HttpFS to act as a gateway to the cluster.

15) True or false? Hortonworks recommends the use of traditional permissions whenever possible, adding a few ACL entries when necessary.

True

16) True or false? HDFS ACLs are enabled by default.

False

17) Which command could be used to indicate the presence of an ACL?

hdfs dfs -ls

18) Which command displays file and directory ACL entries?

hdfs dfs -getfacl

19) List the two types of ACL masks.

Access masks and default masks

20) What is the purpose of an access mask?

An access mask defines the effective permissions of files and directories for named users listed in the ACL (user:<username>:rw-), named groups listed in the ACL (group:<groupname>:rw-), and the unnamed group permission

Summary

- Hadoop can use many file system types, the default commonly being HDFS.
- HDFS is a hierarchical, distributed, replicated, write-once read-many, append-only file system designed for sequential access by multiple readers.
- There are several methods to access HDFS including the HDFS Shell, WebHDFS, the HDFS NFS Gateway, a Java API, Ambari File View, and HUE.
- An HDFS file or directory has an owner and a group owner.
- HDFS files and directories have read, write, and execute permissions for the owner, a group, and everyone else.
- HDFS home directories are used in concert with permissions to control data access.
- The HDFS Shell is a command-line HDFS user interface.
- HDFS trash enables a user to recover accidentally deleted files and directories.
- The NameNode UI includes a browser to view directory contents.
- ACLs extend the HDFS permission model to support more granular file access based on arbitrary combinations of users and groups.
- HDFS ACLs are not enabled by default.
- The setfacl and getfacl HDFS Shell commands are used to manage ACLs.
- There are two ACL mask types:
- The *default mask* on a directory defines ACL entries inherited by child file and directories.
- The *access mask* defines the effective permissions of files and directories.

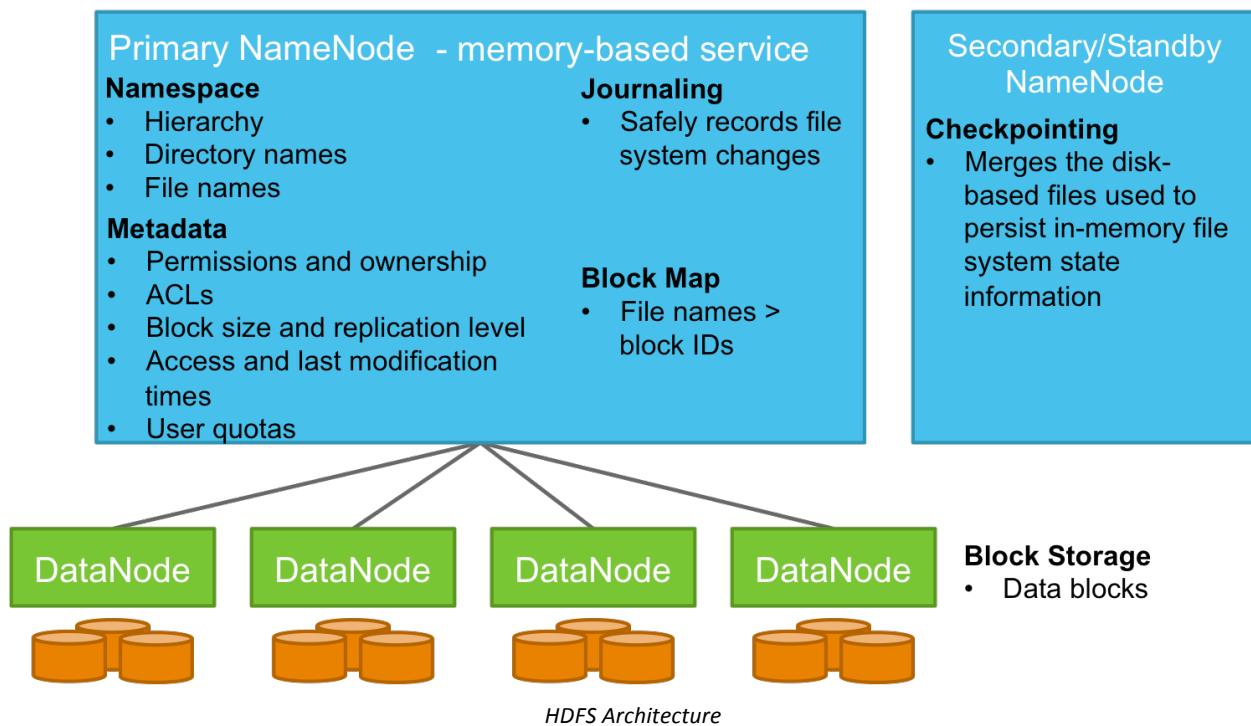
Managing HDFS Storage

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe HDFS architecture and operation
- ✓ Managing HDFS using the Ambari Web, NameNode and DataNode UIs
- ✓ Managing HDFS using Command-line Tools
- ✓ Enable and manage HDFS quotas

HDFS Architecture and Operation



The NameNode and DataNode are components of the HDFS service. The NameNode is an HDFS master component while a DataNode is an HDFS worker component. The NameNode and DataNode are implemented as daemons running inside a Java virtual machine.

The NameNode maintains critical HDFS information. To enhance HDFS performance, it maintains and serves this information from memory. The memory-based information includes namespace information, metadata information, journaling information, and a block map information. Because the NameNode maintains all file system state information in memory, it is critical to ensure that the NameNode has sufficient memory. Refer to the online documentation or the lesson [Installing the Hortonworks Data Platform](#) for more information about NameNode memory sizing.

- The namespace information includes the hierarchical structure of the file system, as well as the directory and file names.

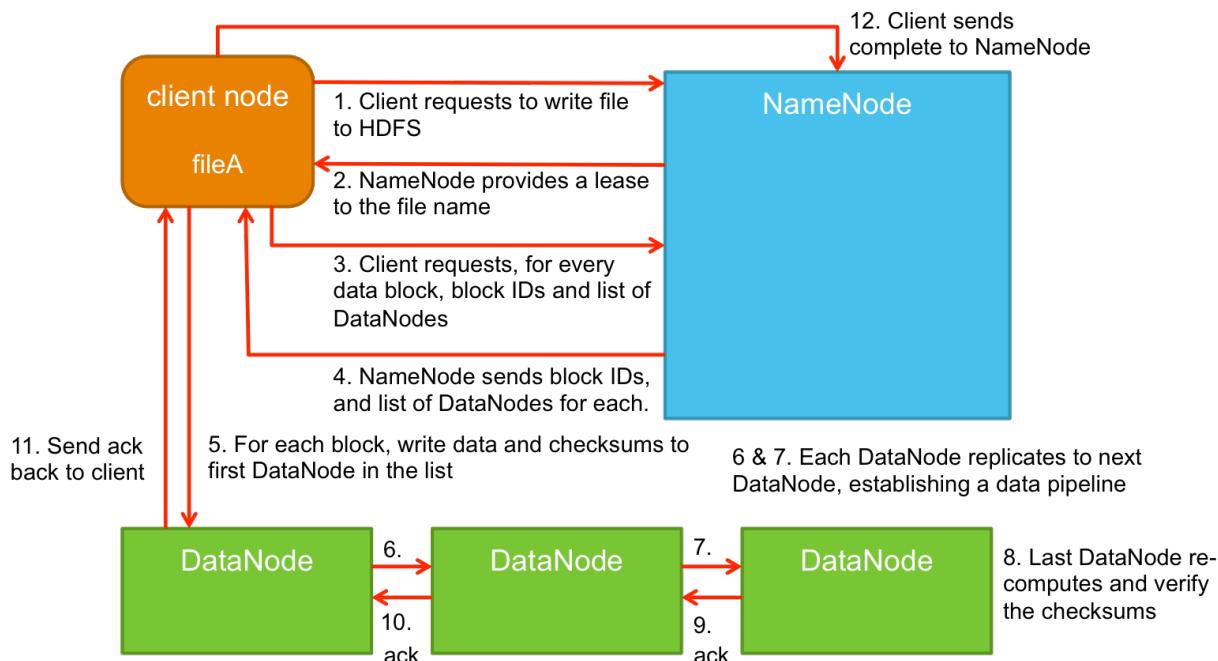
- The metadata information includes file and directory permissions, ownership, ACLs, the last modification and access times, the current replication level, and file block size. Directory-specific metadata also includes quota information. Quotas are described later in this lesson.
- The journaling function protects file system consistency when writing to the file system. File system changes are persisted to disk in the `fsimage_<N>` and `edits_<N>` files. Journaling and these files are described in more detail later in this lesson.
- The block map maps file names to block IDs. Each block ID references a block of data stored somewhere on one of the DataNodes. Block placement is described in more detail later in this lesson.

The DataNodes store HDFS file data. Notice the division of labor; the NameNode stores the file and directory names while the DataNodes store only the file data blocks. Files larger than the HDFS block size are spread across multiple DataNodes. Block size and placement are described in more detail later in this lesson.

DataNodes commonly have multiple disks to improve I/O throughput.

There might also be a Secondary or Standby NameNode. A Secondary or Standby NameNode offloads the checkpoint operation from the Primary NameNode. Checkpointing is described later in this lesson. Having a second NameNode also provides redundancy in case the Primary NameNode fails.

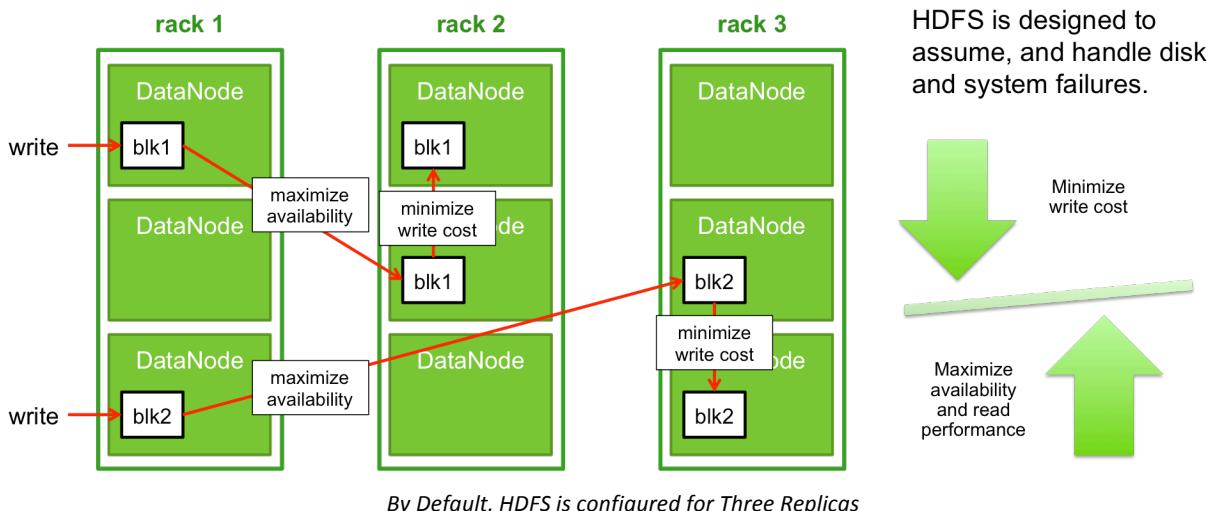
Writing to HDFS Storage



- 1) An HDFS client sends the NameNode a request to write a new file to HDFS storage.
- 2) The NameNode verifies the requested path and file name for availability. If available, the NameNode sends a lease to the client for the file name and path.
- 3) The client verifies the default HDFS data block size and requests block IDs and a list of DataNodes for each block. A program can request a different block size. Block size is described in more detail later in this lesson.

- 4) The NameNode sends the client the required block IDs, and for each one sends a list of DataNodes. The number of DataNodes in the list depends on the replication factor. A program can request a different number of replicas while writing a file. Replication is described in more detail later in this lesson.
- 5) The client writes a data block and its checksum information to the first DataNode in the list of DataNodes. Checksums are described in more detail later in this lesson.
- 6) The first DataNode replicates the block and its checksum information to the next DataNode in the list.
- 7) The second DataNode replicates the block and its checksum information to the last DataNode in the list.
- 8) The last DataNode re-calculates the checksum information to verify the data block. (If a checksum indicates a corrupt data block, the client is notified and the write is re-attempted.)
- 9) The last DataNode sends an acknowledgement back to the second DataNode.
- 10)The second DataNode sends the acknowledgement back to the first DataNode.
- 11)The first DataNode sends the acknowledgement (or checksum error) back to the client.
- 12)When all data blocks have been written to disk, the client informs the NameNode that the write is complete.

Replication and Block Placement



HDFS is designed to assume that disk failures will occur. As a result, HDFS is also designed to automatically and transparently handle disk failures. It does this by automatically replicating data across different disks and DataNodes.

A file can be divided into one or more data blocks. The default data block size of 128 megabytes is determined by the `dfs.blocksize` property in the `hdfs-site.xml` file. Any client is able to override the default block size, as long as they choose a multiple of one megabyte. Any file larger than the block size must occupy multiple data blocks. For example, a one gigabyte file would consume eight data blocks. Files smaller than the block size only consume their required space. For example, a one megabyte file would only consume one megabyte, plus the space for the checksum file. Historically, HDFS required all blocks within the same file to have the same block size. Starting with HDP 2.3, HDFS supports varying block sizes within the same file. This enhancement supports specific use cases like merging several small files together into a single large file. For example, this is done as part of YARN log aggregation.

The data block replica placement policy is a tradeoff between minimizing the write cost, and maximizing the read bandwidth, and data reliability and availability. For example, placing all replicas on a single DataNode in a single rack incurs the lowest write bandwidth penalty because the data block write pipeline only runs within a single DataNode, but would result in no real redundancy. Doing the opposite and placing each data block replica on a separate DataNode in a separate rack would maximize reliability and availability, but would incur higher write bandwidth delays. HDFS write policy is designed to use a middle ground between these two extremes.

When a first block of a new data is written to HDFS, performance is the primary concern. The first block, called a replica, is placed on the same DataNode as the HDFS client. If the HDFS client is external to the cluster, then a DataNode is chosen at random.

The second replica is written with higher availability as the primary concern. The second replica is placed on a DataNode in a separate rack. If there is only a single rack or no rack, another DataNode is chosen at random.

The third replica is written to minimize the write cost and is placed on another DataNode in the same rack as the second replica. The assumption is that there is more network bandwidth and less network delay between DataNodes residing in the same rack. If there is only a single rack or no rack then the third replica is placed on a randomly chosen third DataNode.

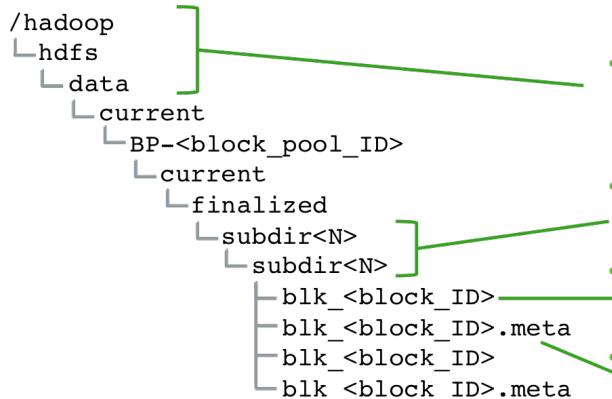
By default, HDFS is configured for three replicas. The default replication factor is determined by the `dfs.replication` property in `hdfs-site.xml`. When writing a file, a program can change the replication factor to another value. If a program chooses to have more than three replicas, the remaining replicas are placed on randomly chosen DataNodes. More replicas are useful for files that are commonly read by multiple programs. The more data block replicas, the more opportunity for simultaneous reads by multiple programs.

There are data block placement restrictions. Two replicas of the same data block cannot be co-located on the same DataNode. Also, no more than two replicas may reside in the same rack when the number of replicas is less than twice the number of racks.

It is important to note that HDFS is not able to ensure that data blocks are written across DataNodes in separate racks without an administrator manually configuring rack awareness. When configuring rack awareness, the administrator manually labels each DataNode with a rack name. Configuring rack awareness is described in a another lesson.

DataNode Disks

DataNode Directory Structure



- DataNodes store file data blocks on disk.
– `dfs.blocksize` in `hdfs-site.xml` determines default data block size (128 MB)
- `dfs.datanode.data.dir` in `hdfs-site.xml` determines the parent directory(ies). (Nodes have multiple disks.)
- Data blocks are scattered across subdirectories to improve performance.
- A file can be divided into data blocks, and each has a unique block ID number.
- The corresponding `.meta` file contains checksum data for its data block.
– One checksum per each-512 byte chunk

DataNode Disks

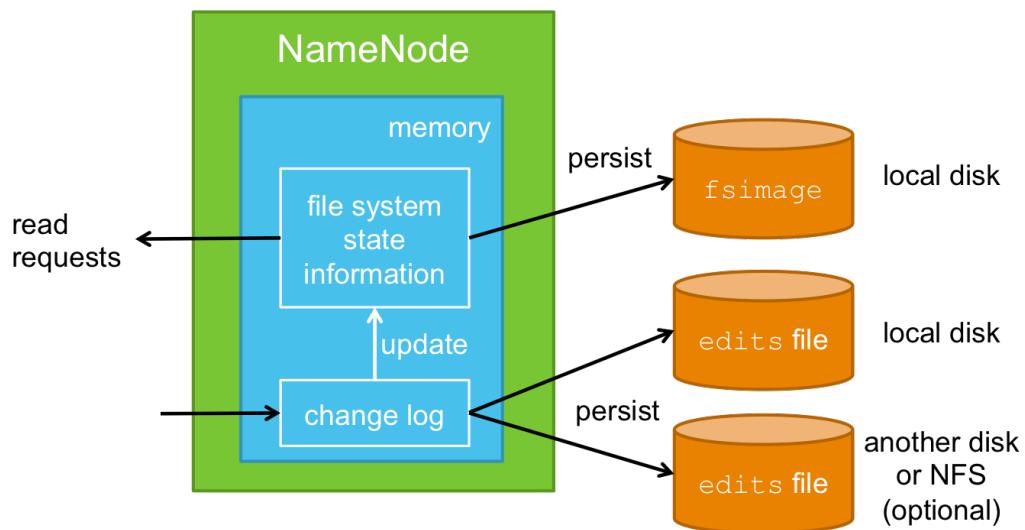
DataNodes store HDFS data blocks on disk. The HDFS data blocks appear as files in the DataNode's local file system. For example, if the DataNode is running Linux, the HDFS data blocks could appear as files in an ext4 file system. The file system hierarchy in the illustration shows a portion of the local file system on a DataNode. This portion of the local file system is used to hold HDFS data blocks. One or more local disks on one or more DataNodes works in concert to hold the HDFS data blocks.

The value in the `dfs.datanode.data.dir` property in the `hdfs-site.xml` file determines the parent directory used to store HDFS file data blocks. This property can contain a comma-separated list of parent directories, which enables a DataNode to use multiple physical disks for data block storage. For example, the property could list `/hadoop/hdfs/data1`, `/hadoop/hdfs/data2`, and so on, which map to multiple disks.

Data blocks are scattered across multiple subdirectories. A new sibling directory is created each time the number of blocks in a directory reaches 64. This shallow directory structure helps to maintain HDFS read performance over time.

Each data block is assigned a unique block ID number. This block ID is used by the NameNode to locate and read file data. Each data block appears as a file in the DataNode's local file system and the file name is based on the HDFS block ID number.

To protect data integrity, each data block is protected by one or more checksums. A checksum is calculated and stored on disk for each 512-byte chunk in a data block. This is controlled by the property `dfs.bytes-per-checksum` in `hdfs-site.xml`, and is typically not modified. The checksums are re-calculated and compared any time the data block is read.

Persisting File System Information on the NameNode*Persisting File System Information on the NameNode*

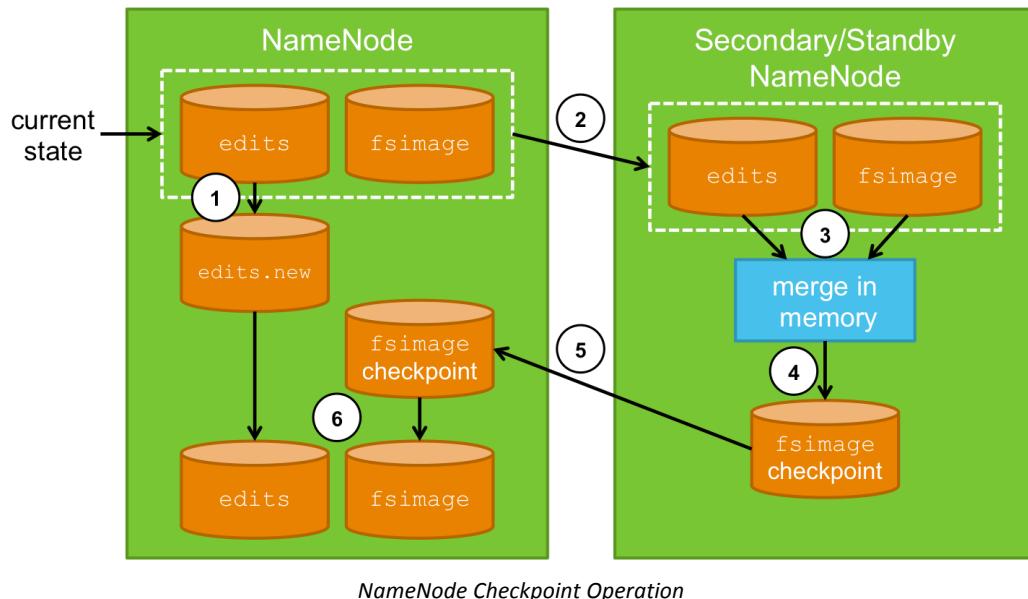
To enhance performance, the current state of the HDFS file system is maintained in NameNode memory. Requests from users or programs for file system information are served out of NameNode memory. The file system state information in memory must be updated whenever a client writes to, or otherwise updates, the file system.

While memory is fast, it is also volatile. For example, a hardware or power failure would result in the loss of HDFS file system state. To permit recovery, file system state information in NameNode memory is regularly persisted to disk.

When a client writes to HDFS, the change request that is sent to memory is first persisted to a disk-based edit log. The change must be persisted to the disk-based edit log before the file system state in memory is updated and also before a success code is returned to the client.

For additional protection, it is possible to persist multiple copies of the disk-based edit log. The directory path location of the edit log is determined by the `dfs.namenode.edits.dir` property in the `hdfs-site.xml` file. An administrator can configure a comma-separated list of directory paths, each one pointing to a separate copy of the edit log. For example, one directory path could be located on a local disk, while a second directory path could be located on a second local disk. The second path could also map to a directory on a remote NFS server.

The actual file system state information in memory is also regularly persisted to disk as an `fsimage` file. Persisting the file system state to an `fsimage` file is called checkpointing.

NameNode Checkpoint Operation

Checkpointing proceeds as follows:

- 1) The Secondary NameNode asks the Primary NameNode to roll its current `edits` file and create a new `edits` file. The Primary NameNode begins to use the new, empty `edits` file.
- 2) The Secondary NameNode retrieves the `fsimage` and `edits` files from the Primary NameNode.
- 3) The Secondary NameNode loads the `fsimage` file into memory and applies each change recorded in the `edits` file.
- 4) The Secondary NameNode creates a new, checkpointed `fsimage` file.
- 5) The Secondary NameNode sends the new, checkpointed `fsimage` file to the Primary NameNode.
- 6) The Primary NameNode saves the new, checkpointed `fsimage` file to disk. This file is used if the Primary NameNode is restarted. NameNode startup is described later in this lesson.

NameNodes must periodically perform a checkpoint operation or the `edits` file would continue to grow without bounds. A checkpoint operation merges the changes recorded in the current `edits` file with the information in the current `fsimage` file, and then replaces the `edits` and `fsimage` files with a new files. The new `edits` file will initially be empty.

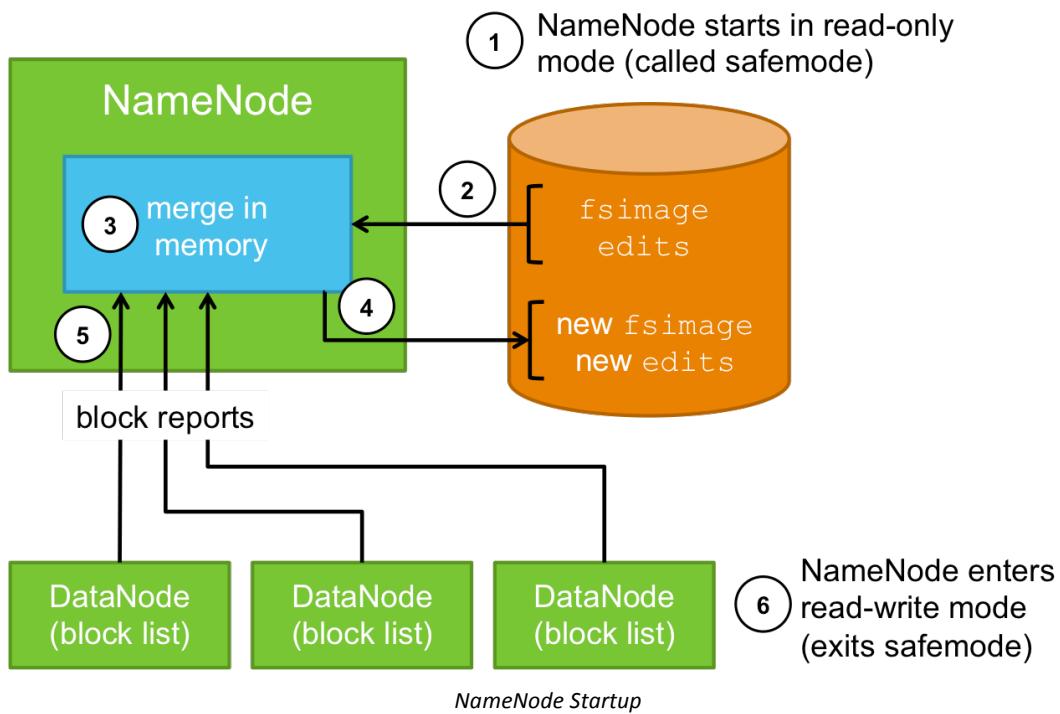
The location of the `fsimage` file is determined by the directory path listed in the `dfs.namenode.name.dir` property in the `hdfs-site.xml` file. It is possible to maintain multiple copies of the `fsimage` file by configuring a comma-separated list of directories.

Checkpoint operations can consume a large amount of CPU and memory resources. To maintain the performance of the Primary NameNode during checkpoints, checkpoint operations are offloaded to a Secondary NameNode. If NameNode High Availability has been configured, then a Standby NameNode replaces the Secondary NameNode and checkpointing is offloaded to the Standby NameNode instead.

Checkpoints occur every hour based on the value in the `dfs.namenode.checkpoint.period` property in the `hdfs-site.xml` file. Checkpoints can occur more frequently based on the number of transactions in the current `edits` file. If the number of transactions reaches the value in the `dfs.namenode.checkpoint.txns` property in `hdfs-site.xml`, then a checkpoint occurs immediately. The default is 1,000,000 transactions.

Checkpointing works nearly the same way if NameNode HA is configured. The difference is that the Secondary NameNode is replaced by a Standby NameNode and the Primary NameNode is referred to as the Active NameNode. NameNode HA is described and configured in another lesson.

NameNode Startup



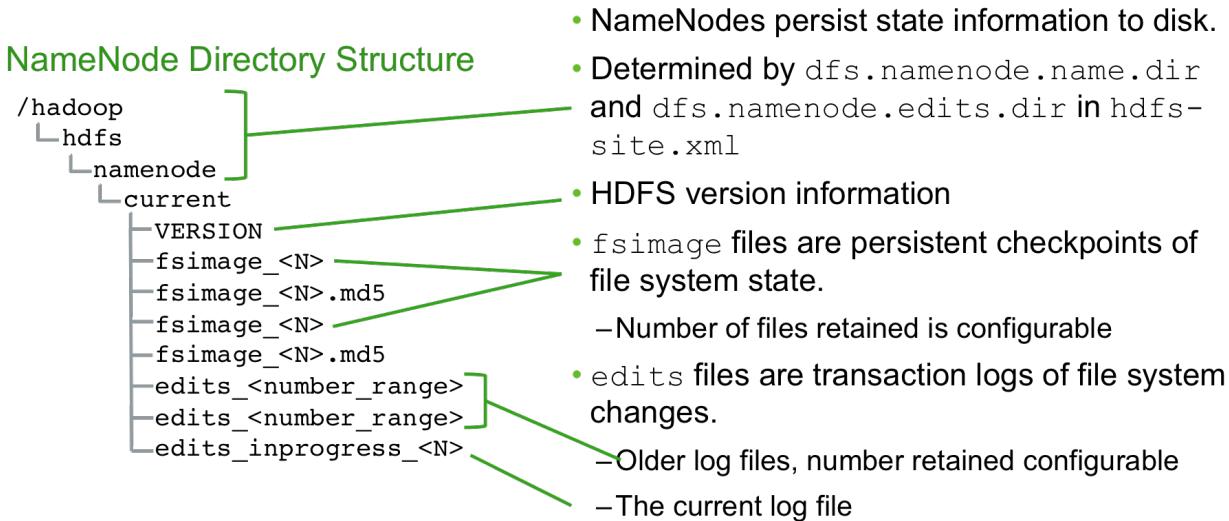
Startup proceeds as follows:

- 1) When a NameNode starts up it enters safe mode, which is a read-only mode.
- 2) The NameNode loads its latest `fsimage` and `edits` files into memory.
- 3) The NameNode performs a checkpoint that merges the information in the `fsimage` and `edits` files. This creates an up-to-date, in-memory image of the file system.
- 4) It writes an up-to-date `fsimage` file to disk and creates a new, empty `edits` file to track file system changes.

At this point the NameNode begins to listen for RPC or HTTP HDFS client requests. However, the NameNode is still running in read-only mode. The NameNode must be in read-only mode because the list of data blocks that contain HDFS file data are not persisted on the NameNode. Each DataNode must send its block list to the NameNode. These block lists are sent in a block report.

- 5) The DataNodes send their block lists to the NameNode. The NameNode aggregates this information to rebuild the block map in memory. The block map is used to locate and read file data. The purpose of safe mode is to provide the NameNode the time necessary to rebuild the block map.
- 6) NameNode exits safe mode, it begins its normal read-write operation. A NameNode may exit safe mode only when 99.9% of all data blocks are minimally replicated. Minimally replicated means at least one replica is available. This percentage is determined by the `dfs.namenode.safemode.threshold-pct` property in `hdfs-site.xml`.

NameNode Disk



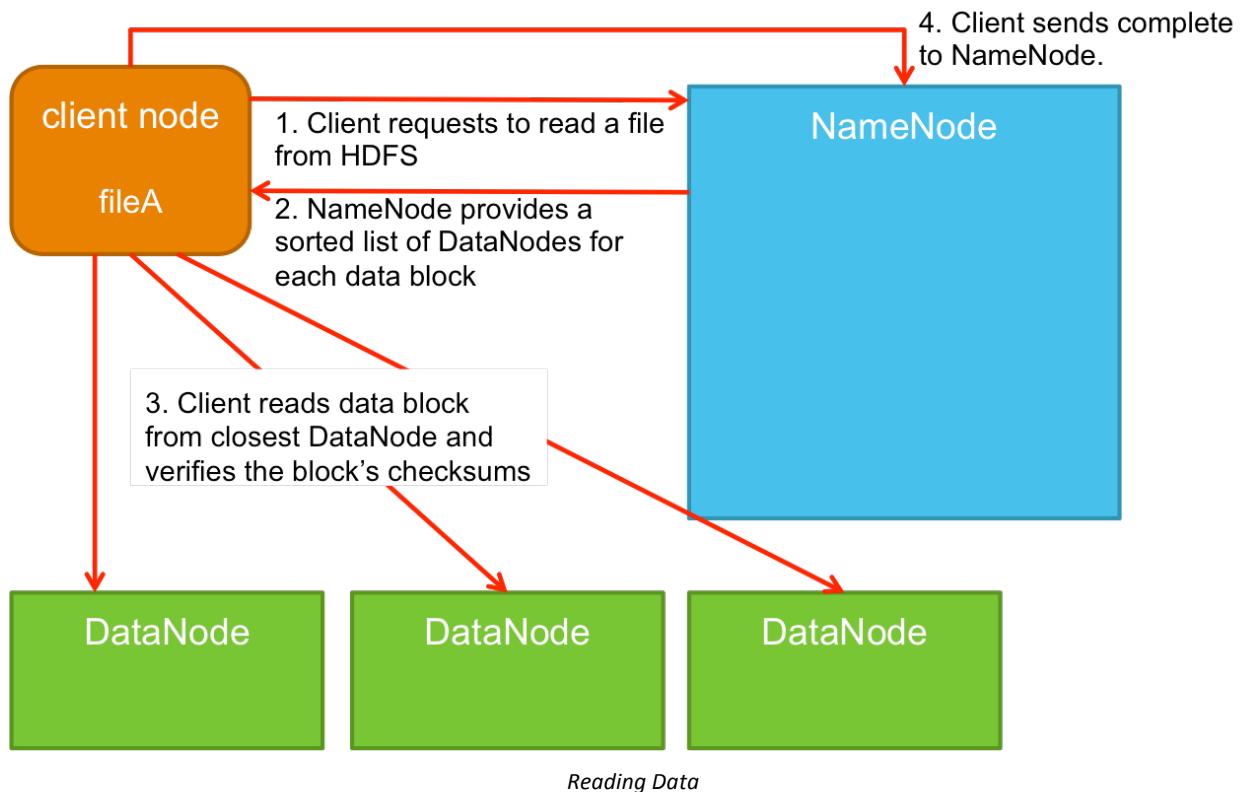
NameNode Disks

NameNodes persist HDFS storage state information to disk. The value recorded in the `dfs.namenode.name.dir` property in the `hdfs-site.xml` file determines the parent directory of the persisted information. This property can contain a comma-separated list of multiple directories. For example, one directory could be located on a local disk while a second directory could map to a directory mounted from an NFS server. This would provide redundancy for this critical information. Configuring NameNode High Availability (HA) is a popular method to achieve NameNode redundancy. NameNode HA is described in another lesson.

The number of past `edits` files to retain is controlled by the `dfs.namenode.num.extra.edits.retained` property in `hdfs-site.xml`.

The number of `fsimage` checkpoint files to retain is controlled by the `dfs.namenode.num.checkpoints.retained` property in `hdfs-site.xml`.

Reading Data



HDFS read performance is enhanced by never moving the actual file data through the NameNode machine. All data is read directly from one or more DataNodes, depending on the size of the file.

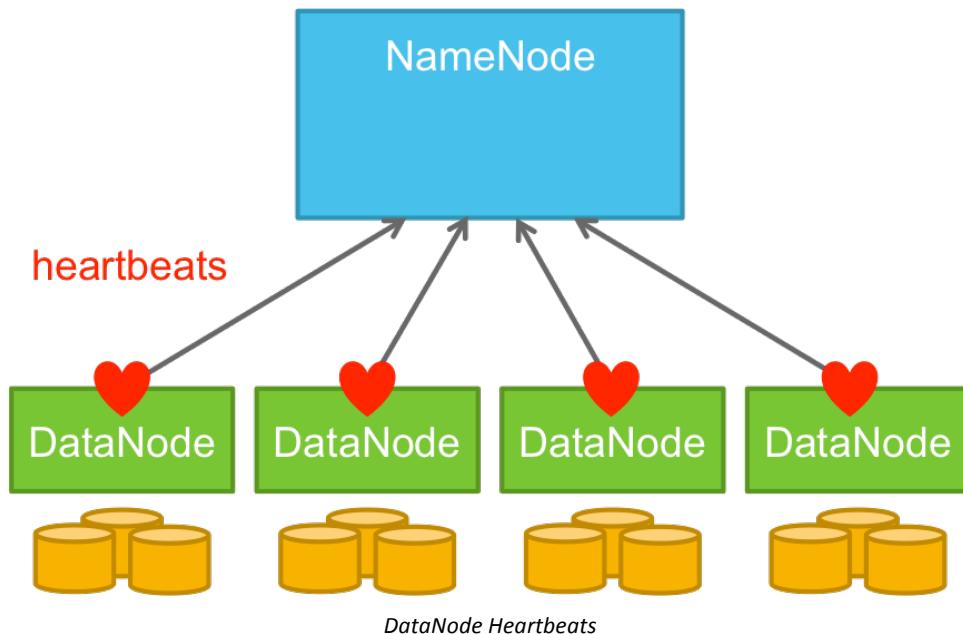
To begin a read operation, an HDFS client contacts the NameNode to determine the locations of the blocks for a file. Each data block commonly has multiple replicas. For each data block, the NameNode returns a sorted list of the DataNodes with replicas of the block. The list is sorted by network proximity to the client. This sorted list is particularly important if the HDFS client is part of the cluster. If the HDFS client is part of the cluster, for example a Tez or MapReduce task, then the client will read from the local DataNode, if the local DataNode contains a copy of the data block. All data blocks of an HDFS file are read in order as an HDFS client opens connections to DataNodes across the cluster.

If a client encounters an error while communicating with a DataNode, it will try the next closest DataNode for that data block. It will also remember a DataNode with failed read so that it will not try it again.

The HDFS client also reads and verifies the checksums associated with each data block. If a corrupted data block is found, it is reported to the NameNode before the client attempts to read a replica from the next closest DataNode. The NameNode will replicate a new data block from a good replica and then delete the corrupted data block.

HDFS policies determine how data blocks are re-replicated. When re-replicating a data block, if the number of existing replicas is one, then HDFS places the second one on a different rack. When the number of existing replicas is two, if the two replicas are on the same rack, HDFS places the third one on a different rack. If there is only a single rack, HDFS places the third one on a different DataNode on the same rack of first replica. When the number of available replicas is more than two, place the rest of the replicas randomly.

DataNode Availability



The NameNode listens for DataNode heartbeats to determine DataNode availability. All configuration settings related to DataNode availability are configurable in the `hdfs-site.xml` file.

The heartbeat interval is three seconds by default, as determined by the property `dfs.heartbeat.interval`. If a heartbeat has not been received by the NameNode for 30 seconds, the DataNode is marked as stale. The 30-second threshold is determined by the `dfs.namenode.stale.datanode.interval`. The minimum possible value is three times the heartbeat interval.

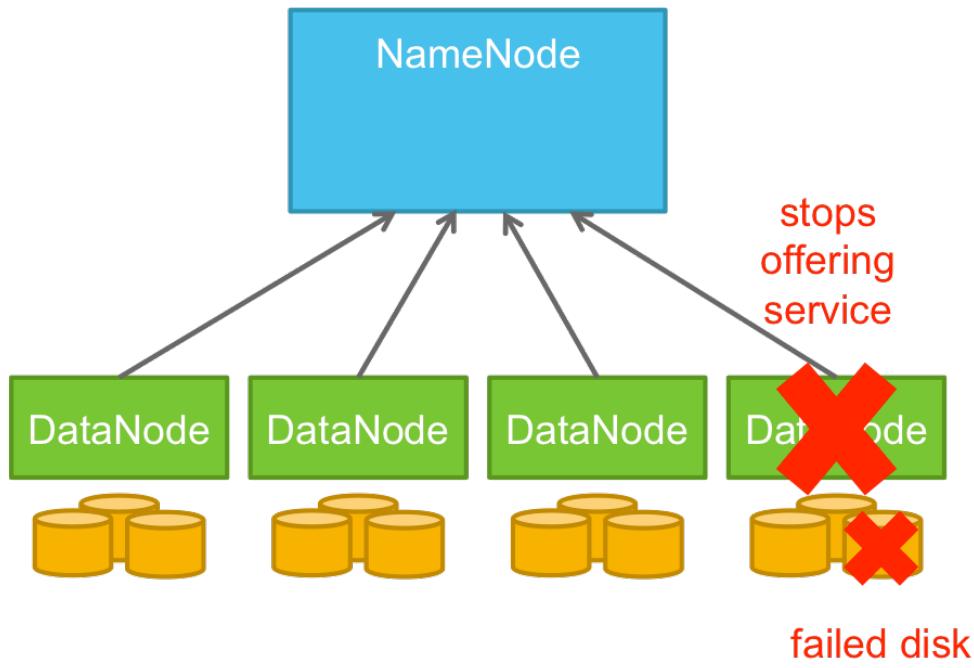
A stale DataNode is returned at the end of the list of DataNodes when the NameNode is trying to satisfy client read requests. This is determined by `dfs.namenode.avoid.read.stale.datanode`, when set to `true`. In HDP it is set to `true` by default.

A NameNode avoids writing to a stale DataNode if `dfs.namenode.avoid.write.stale.datanode` is set to `true`. In HDP it is set to `true` by default. Stale DataNodes are written to only if the number of stale DataNodes exceeds the ratio determined by `dfs.namenode.write.stale.datanode.ratio`. In HDP it is set to 1, which means that HDFS may write to a stale DataNode.

A NameNode declares a DataNode dead when it has not received a heartbeat from it for 10 minutes and 30 seconds. This period is determined by the formula $(2 \times \text{dfs.namenode.heartbeat.recheck-interval}) + (10 \times \text{dfs.heartbeat.interval})$.

A NameNode does not use data blocks from a dead DataNode to satisfy client requests. Also, a dead DataNode no longer offers its data blocks to the NameNode, which causes the replication factor of those data blocks to fall below their minimum value. The NameNode will immediately begin to re-replicate these data blocks to other DataNodes using the remaining data block replicas.

Failed DataNode Disks



Failed DataNode

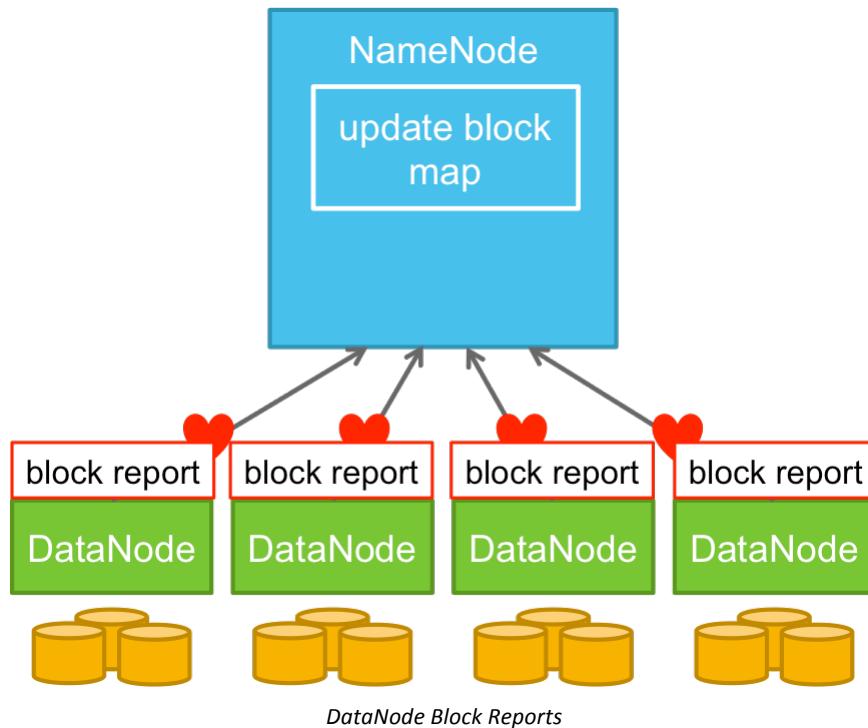
Each DataNode typically has multiple disk drives. The use of multiple disk drives is used to enhance HDFS performance and increase the amount of disk space that can be used for HDFS storage.

Performance is enhanced because the read-write heads on multiple disk drives can move independently of each other and simultaneously satisfy multiple client I/O requests. In this scenario, a single disk controller could become a bottleneck. Configuring multiple disk controllers per DataNode mitigates this by increasing overall storage channel I/O bandwidth.

Using multiple disk drives also increases the amount of storage space that a DataNode can offer to HDFS. While using a larger disk is possible, using a single, large-capacity disk would not offer the same I/O throughput as using multiple, lower-capacity disks.

The potential problem with using multiple disk drives is that it increases the opportunity for drive failure. For example, if a disk drive has a mean-time-between-failure (MTBF) rating of 200,000 hours, it is assumed that, on average, the disk will fail at around 200,000 hours. But 8 disks will collectively have an MTBF of 200,000 hours divided by 8. In a cluster with hundreds of disk drives, drive failures become a regular event.

By default, a DataNode will stop offering read-write service if one of its disk drives fail. However, the data blocks on the failed drive are still available on other DataNode so there is no strict reason why the DataNode should stop functioning as a DataNode. It still has other data blocks to offer. An administrator can make a DataNode more tolerant of drive failure by configuring the `dfs.datanode.failed.volumes.tolerated` property in `hdfs-site.xml`. By default, a DataNode tolerates zero failed drives, but this number can be increased to one or more drive failures.

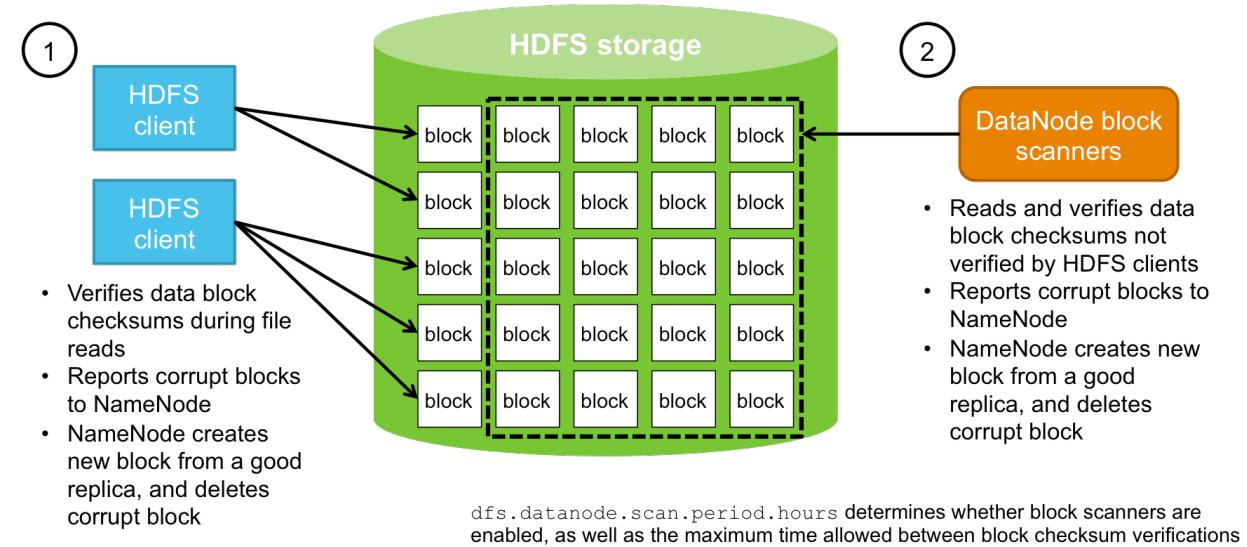
DataNode Block Reports

Each DataNode maintains a list of its data blocks. This block information is not persisted on the NameNode and therefore must be sent to the NameNode in a block report.

At DataNode startup, a block report is sent to the NameNode after a configurable delay. The delay is determined by the `dfs.blockreport.initialDelay` property, which in HDP is set to 120 seconds. The NameNode uses the block reports to update its block map. The block map is used to locate, read, write, and append HDFS files.

After initial startup, each DataNode periodically sends an updated block report to the NameNode. The period is determined by the `dfs.blockreport.intervalMsec` property, which in HDP is set to 21600000 milliseconds, or 6 hours. The block report is sent as part of the heartbeat.

If the number of data blocks on a DataNode is below the threshold set by the `dfs.blockreport.split.threshold` property, a single block report that includes every HDFS storage directory is sent to the NameNode. If the threshold is exceeded, then a separate block report is sent for each HDFS directory and the block report spans multiple heartbeats. In HDP, the threshold is set to 1,000,000 blocks.

Verifying Data Integrity*Verifying Data Integrity*

Over time disk media suffers a slow deterioration that can change a data bit, effectively corrupting a data file. This is sometimes referred to as data rot. The more data stored on disk, the higher the probability of experiencing a small amount of data rot.

HDFS typically stores an massive amount of data on disk and therefore is more susceptible to data rot than other file system types. HDFS was designed with this in mind and includes two techniques to check data for corruption.

When an HDFS client reads a data block it performs a checksum verification. If the checksums for a data block are okay, the client informs the DataNode, which records this information. This is an acceptable technique for files that are regularly read by clients. However, HDFS commonly has data that is rarely read and therefore would not have its checksums regularly checked by HDFS clients.

A second technique is available that can regularly verify the checksums of data blocks that have not been recently read by an HDFS client. Each DataNode runs a block scanner that reads through unread data blocks on a cyclical basis and verifies their checksums. The block scanner does not fix corrupted blocks.

The block scanner is configurable. The `dfs.datanode.scan.period.hours` property controls whether the block scanner is disabled or not. The value of 0 disables the block scanner. A positive number defines the time period in which all data blocks must be checked. A DataNode will not scan any individual block more than once in the specified time period. The block scanner adjusts its read rate to ensure it completes block scanning within the configured time period. If you have large volumes of typically unread data and you would like to configure block scanning, add the property `dfs.datanode.scan.period.hours` to the `hdfs-site.xml` file and configure it with a positive number. For example, using 560 would ensure that each unread block has its checksum verified at least every two weeks.

Whenever an HDFS client or the block scanner detects a corrupt block, it notifies the NameNode. The NameNode marks the replica as corrupt, but does not immediately schedule deletion of the replica. Instead, it replicates a good copy of the block from another DataNode. Once the good replica count reaches the block's replication factor, the corrupt replica is scheduled to be removed. The goal of this process is to preserve data as long as possible. So even if all replicas of a block are corrupt, the policy allows the user to retrieve its data from the corrupt replicas.

If all replicas of a block are corrupted, but in different places, there is a possibility of manually examining and fixing the data. There is no automatic utility to perform this type of data repair.

Managing HDFS Using UIs

Option	Description
Ambari Web UI	Browser-based, HDFS configuration and service management interface
NameNode UI	Browser-based interface for basic status monitoring and directory browsing
DataNode UI	Browser-based interface, most commonly used to get block scanner reports (a scanner report is shown later)
HDFS command-line tools	Various command-line tools to interact with the HDFS service and its files, directories, and metadata (described later)
Manual configuration	Manually editing configuration files (not compatible with Ambari administration)

HDFS Management Options

Ambari HDFS Service Management

Ambari HDFS Service Management

The Ambari Web UI is the easiest method to manage the HDFS service and its NameNode and DataNode components. To manage the HDFS service components, log in to the Ambari Web UI and click **Services**, then **HDFS**, and then view the actions available on the **Service Actions** menu button.

Start, **Stop**, and **Restart All** do exactly as they are labeled. **Start** starts the NameNode and DataNode components if they are not running. **Stop** stops the NameNode and DataNode components if they are running. **Restart All** stops and then starts the NameNode and DataNode components. Restarting these components is often necessary when one or more HDFS configuration properties have been changed. Following a configuration property change, the Ambari Web UI will notify the administrator if a restart is required.

If the configuration property only applies to the DataNodes, then **Restart DataNodes** can be used to restart only the DataNode components.

The **Move NameNode** and **Move SNameNode** enable an administrator to migrate these master service components to another host in the cluster. The ability to move these components enables an organization to more easily perform hardware maintenance, or move these components to a host with more physical resources as might be required as the cluster grows. The choices are unavailable in the screen capture because the cluster was a single-node cluster.

Enable NameNode HA enables an administrator to configure and run a Standby NameNode. A Standby NameNode can automatically assume control of the HDFS service should the Primary NameNode fail or otherwise become unavailable. NameNode HA is described and configured in another lesson. Again, this choice was unavailable in the screen capture because this was a single-node cluster.

Run Service Check enables an administrator to use Ambari to test the HDFS service components for proper functionality. This is the same test that is automatically run by Ambari at the end of the installation process to ensure installation was successful.

Turn On Maintenance Mode enables maintenance mode on any node running an HDFS service component. Maintenance Mode is described in another lesson.

Rebalance HDFS moves data blocks between the DataNodes, as necessary, to ensure that they are evenly distributed. This can help to reduce performance “hot spots” where certain DataNodes do more work than other DataNodes, negatively effecting overall HDFS performance. Rebalancing HDFS is described in more detail later in this lesson.

Download Client Configs downloads the HDFS configuration files to the host running the Ambari Web UI browser interface. These configuration files can be used to manually configure a machine that was manually installed with only the HDFS client software.

Ambari HDFS Configuration Management

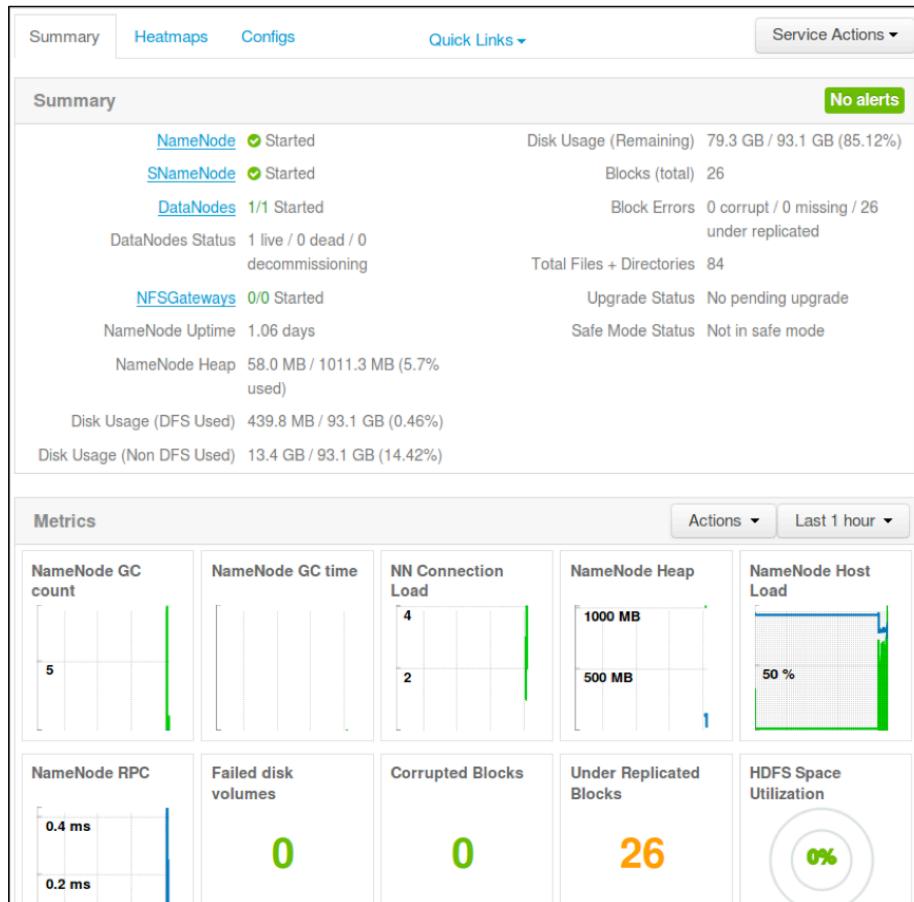
The screenshot shows the Ambari HDFS Configuration Management interface. The left sidebar lists services: HDFS, MapReduce2, YARN, Tez, Hive, Pig, ZooKeeper, and Ambari Metrics. The main area shows the 'Configs' tab for the 'HDFS Default (1)' group. It displays two versions: V2 (8 days ago, HDP-2.3) and V1 (19 days ago, HDP-2.3). A 'Save' button is visible. Below the configurations are tabs for 'NameNode' and 'DataNode'.

HDFS Configuration Management

The Ambari Web UI is the easiest method to manage the configuration properties of the NameNode and DataNode components. To manage HDFS component configuration, log in to the Ambari Web UI and click **Services**, then **HDFS**, and then click the **Configs** tab.

Managing HDFS Storage

Ambari HDFS Monitoring

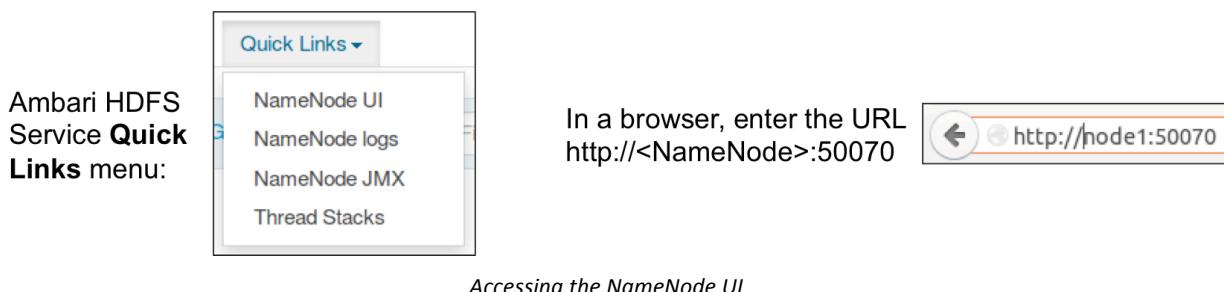


Ambari HDFS Monitoring

The Ambari Web UI is used to monitor the HDFS service and storage. For example, it monitors and reports:

- Service component state
- Service component history
- File system usage
- File system state
- Heatmaps

The NameNode UI



Accessing the NameNode UI

The NameNode includes a Web server with a Web interface. The NameNode UI is a browser-based interface for basic status monitoring and directory browsing.

There are two ways to access it.

- The first is to select **NameNode UI** from the HDFS services **Quick Links** menu.
- The other is to open a browser and enter the URL `http://<NameNode>:50070`.

The address and port number of the NameNode UI is determined by the `dfs.namenode.http-address` or `dfs.namenode.https-address` properties in the `hdfs-site.xml` file. The default address and port number is `0.0.0.0:50070`. A NameNode could be configured with multiple network interfaces and multiple network addresses. The IP address `0.0.0.0` configures the NameNode to listen for connection requests on any of its configured IP addresses.

NameNode UI Tabs

The screenshot shows the NameNode UI Overview tab. The top navigation bar includes tabs for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The Utilities tab has a dropdown menu with options 'Browse the file system' and 'Logs'. Below the tabs, the 'Overview' section displays system information:

Started:	Wed Jul 08 10:34:31 EDT 2015
Version:	2.7.1.2.3.0.0-2410, r3f5897393b53dfa5bd8ae6407444db5d48743556
Compiled:	2015-06-18T14:09Z by jenkins from (no branch)
Cluster ID:	CID-c7ce3751-fb7a-49a4-9556-92e5eec360a2
Block Pool ID:	BP-1386861603-172.17.0.2-1434735465264

Below this is a 'Summary' section.

NameNode UI Tabs

The NameNode UI features several tabs, each displaying different types of status, health, and usage information.

The **Overview** tab provides status and health information about the NameNode along with overall HDFS storage usage information. Similar information is also available in the Ambari Web UI. Similar HDFS usage information is also available in the output of the `hdfs dfsadmin -report` command. This command is described later in this lesson.

The **DataNodes** tab provides status and HDFS usage information for each individual DataNode. It also includes decommissioning status information. Decommissioning is described in another lesson.

The **DataNode Volume Failures** tab reports volumes (disks) that have failed on DataNodes. By default, a DataNode with a failed volume will stop offering its service. How to change this default behavior was described earlier in this lesson.

The **Snapshot** tab provides a list of HDFS directories that are currently configured to support HDFS snapshots, along with a current list of directories that have snapshots. HDFS snapshots are useful as part of a backup process. HDFS snapshots are described in another lesson.

The **Startup Progress** tab reports information about the last time the NameNode was restarted. It details which `fsimage` and `edits` files were used, the time the NameNode spent in each phase of the startup, and percentage progress of each step in NameNode startup. This information could be used to help troubleshoot NameNode startup failures.

The **Utilities** tab includes the choice to browse the HDFS file system or view HDFS-related log files. Browsing the file system was described in an earlier lesson. To view a log file, click **Logs** and then click the log file to view. The contents of the log file opens in the browser window.

DataNode UI

Block report for block pool: BP-1472918407-172.17.0.2-1431707688874		
Total Blocks	:	395
Verified in last hour	:	0
Verified in last day	:	9
Verified in last week	:	395
Verified in last four weeks	:	395
Verified in SCAN_PERIOD	:	395
Not yet verified	:	348
Verified since restart	:	9
Scans since restart	:	9
Scan errors since restart	:	0
Transient scan errors	:	0
Current scan rate limit Kbps	:	1024
Progress this period	:	64%
Time left in cur period	:	98.46%

DataNode UI

Each DataNode includes a Web server with a Web interface. A common use is to view a DataNode's block scanner report. To view a block scanner report, use the URL

`http://<DataNode>:50075/blockScannerReport`. If all numbers are zero, block scanning is not enabled or has not started yet.

The address and port number of the DataNode UI is determined by the `dfs.datanode.http.address` or `dfs.datanode.https.address` properties in the `hdfs-site.xml` file. The default address and port number is `0.0.0.0:50075`. A DataNode could be configured with multiple network interfaces and multiple network addresses. The IP address `0.0.0.0` configures the DataNode to listen for connection requests on any of its configured IP addresses.

Note

The final period in the `dfs.datanode.http.address` and `dfs.datanode.https.address` is correct. The syntax for these DataNode UI properties and that for the similar NameNode UI properties is slightly different. The NameNode UI properties employ a hyphen character rather than a final period.

Managing HDFS Manually and Using Command-Line Tools

Command Line Tools

Command	Description
<code>hdfs dfs</code>	HDFS Shell to manage files, directories, and their metadata
<code>hdfs fsck</code>	Checks and reports on file system inconsistencies (does not repair)
<code>hdfs dfsadmin</code>	Reports basic file system information and statistics and performs various file system administration tasks

HDFS Command-Line Tools

HDFS can be monitored and managed using command-line tools. This section of the lesson introduces various options for the `hdfs dfs`, `hdfs fsck`, and `hdfs dfsadmin` commands.

`hdfs dfs`

Managing files, directories and metadata

Determining Consumed Storage Space

- The HDFS Shell `du` command reports the number of bytes consumed by a file or directory.
- Syntax: `hdfs dfs -du [-s] [-h] [path]`
- Examples:

```
[root@node1 ~]# hdfs dfs -du
1520  dir1
0  dir3
1520  passwd
12  textfile.txt
[root@node1 ~]# hdfs dfs -du dir1
0  dir1/dir2
1520  dir1/passwd
[root@node1 ~]# hdfs dfs -du -s dir1
1520  dir1
[root@node1 ~]# hdfs dfs -du -h dir1
0  dir1/dir2
1.5 K  dir1/passwd
[root@node1 ~]#
```

Determining Consumed Storage Space

Sometimes a user or administrator needs to know how much space is consumed by a file or an entire directory. While the command `hdfs dfs -ls` displays file sizes, it does not display directory sizes. The HDFS Shell `du` command meets this need by reporting the number of bytes consumed by a file or a directory. It is only the sum of files sizes and does not include the size of all the replication blocks. The command syntax along with a few examples are shown here.

The first command does not include a path argument and reports file and directory space usage in the user's home directory.

The second command adds a `dir1` path argument and reports file and directory usage for just the `dir1` directory.

The third command adds the `-s` option. The use of the summary option restricts the output to a single-line summary of the space consumed by the directory.

The last command illustrates the `-h` option. The “human” readable option reports sizes in kilobytes, megabytes, gigabytes, and terabytes, rather than just bytes.

Monitoring File System Space

- The HDFS Shell `df` command reports the file system's total capacity, along with the current amount of free and used storage space.
- Syntax: `hdfs dfs -df [-h]`
- Examples:

```
[root@node1 ~]# hdfs dfs -df
Filesystem           Size      Used   Available  Use%
hdfs://node1:8020  100000174080  461144064  85128048640  0%
[root@node1 ~]# hdfs dfs -df -h
Filesystem           Size      Used   Available  Use%
hdfs://node1:8020    93.1 G   439.8 M    79.3 G   0%
[root@node1 ~]#
```

“Human” readable format,
K, M, G, and T bytes
instead of bytes

The `hdfs dfs-df` Command

Sometimes an administrator needs to know HDFS file system usage information. For example, they might need to know the total capacity of the file system. They might need to know how much of that total capacity is currently free or currently used. The HDFS Shell `df` command meets this need by reporting the file system's total capacity, along with the current amount of free and used storage space. It also reports the percentage of the total file system space that is currently used. The command syntax along with two examples are shown here.

The first command reports total capacity (Size), along with the current amount of that capacity that is used or still available.

The last command illustrates the `-h` option. The “human” readable option reports sizes in kilobytes, megabytes, gigabytes, and terabytes, rather than just bytes.

HDFS `fsck`

The `fsck` command checks file system consistency. Hortonworks recommends running the `fsck` command in specific circumstances. First, run `fsck` when there is any concern about possible file corruption. This might be the result of an HDFS or hardware malfunction. Second, run `fsck` before upgrading HDFS to a newer version.

Unlike UNIX or Linux systems, `fsck` in HDFS does not repair corrupt data blocks or files. It only reports corrupt data blocks and other inconsistencies. If all the replicas for a data block are corrupt, an administrator must manually repair the file, if possible. The corrupted data blocks are not automatically removed unless the `fsck` command is run with the `-delete` option. The only thing that is automatically repaired are data block replication issues. This is described later in this lesson.

All of the information required to run `fsck` is located on the NameNode. DataNodes are never contacted as a result of running `fsck`. The `fsck` command only reads the metadata and block information located in NameNode memory. As a result, `fsck` completes very quickly even on large clusters.

While any user may run `fsck`, the user must have access permissions to any files or directories being checked. The only user with complete access to everything in HDFS is the HDFS superuser, which is commonly the `hdfs` user account.

- Syntax:

```
-hdfs fsck [path] [options] [> <output_file>]
```

Options	Description
<code>-files</code>	Reports a list of file and directories checked
<code>-blocks</code>	Reports block ID numbers checked (requires <code>-files -blocks</code> syntax)
<code>-locations</code>	Reports a list of DataNodes locations for each block ID number (requires <code>-files -blocks -locations</code> syntax)
<code>-racks</code>	Prepends the rack name on each reported DataNode location (requires at least <code>-files -blocks -racks</code> syntax). Really only useful if HDFS rack awareness has been configured (described in another lesson).
<code>-move</code>	Moves files with corrupted data blocks to the <code>/lost+found</code> directory
<code>-delete</code>	Deletes files with corrupted data blocks
<code>-openforwrite</code>	List files open for write during <code>fsck</code> (open files are not checked)

Understanding `fsck` Output

The `fsck` command reports a lot of information and it is important to understand the meaning of the reported information. Failure to correctly understand the information could lead to the lack of proper action.

The `fsck` command reports the following information about data blocks:

- Minimally replicated blocks:** These are all the data blocks that have at least one good replica. This means that the data is not completely corrupted and the related file is still readable. The NameNode will automatically replicate the good replica until the file's minimum replication factor is met.
- Over-replicated blocks:** Each file has a replication factor. Any extra data blocks that exceed this replication factor are reported as over-replicated blocks. The NameNode will automatically delete any extra data blocks.
- Under-replicated blocks:** These are data blocks that do not meet the file's minimum replication factor. The NameNode will automatically replicate the data block until it meets the file's replication factor.

- **Mis-replicated blocks:** This refers to data blocks that are replicated, but not correctly according to the HDFS block placement policies. For example, a data block should never be replicated to the same DataNode. If it were, it would be reported as mis-replicated. The NameNode will move data blocks to correct this problem.
- **Corrupt blocks:** This is a serious problem that the NameNode cannot fix. This reports data blocks where every replica is corrupt based on checksum failures. These data blocks are not automatically deleted but cannot be automatically repaired either. A user with the necessary technical skill will have to open the file with the corrupt data block(s) and attempt to manually repair it.

Primary Output

• `hdfs fsck /user/root`

```
Status: HEALTHY
Total size: 4829660 B
Total dirs: 6
Total files: 5
Total symlinks: 0
Total blocks (validated): 7 (avg. block size 689951 B)
Minimally replicated blocks: 7 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 3.0
Corrupt blocks: 0 (0.0 %)
Missing replicas: 0 (0.0 %)
Number of data-nodes: 3
Number of racks: 1
FSCK ended at Tue May 19 14:25:40 EDT 2015 in 2 milliseconds
```

HEALTHY status

No corrupt blocks

fsck Primary Output

This is an example of `fsck` on a specific user directory which, for training purposes, has a very limited number of files, directories, and data blocks. Notice that the status is reported as **HEALTHY** and that there are no corrupt blocks.

The `-files` Option

```
/user/root <dir>
/user/root/.Trash <dir>
/user/root/ambari-metrics 1814809 bytes, 1 block(s):  OK
/user/root/big1 3011212 bytes, 3 block(s):  OK
/user/root/dir1 <dir>
/user/root/dir1/hosts 173 bytes, 1 block(s):  OK
/user/root/dir1/passwd 1733 bytes, 1 block(s):  OK
/user/root/dir2 <dir>
/user/root/dir2/dir3 <dir>
/user/root/passwd 1733 bytes, 1 block(s):  OK
/user/root/web_logs <dir>
```

Prepends a list of files and directories to the primary output.

hdfs fsck /user/root -files

The `-files` option prepends a list of the files and directories being examined to the main `fsck` output.

For each file listed, the file's size and number of data blocks is reported. The status of the file is also reported. In the example, the status is OK.

Directories only exist in NameNode memory and therefore do not have a size or consume any data blocks. Because no data blocks are examined, there is no `fsck` status to report for directories.

The `-blocks` Option

```
/user/root/big1 3011212 bytes, 3 block(s):  OK
0. BP-1472918407-172.17.0.2-1431707688874:blk_1073742266_1442 len=1048576 repl=3
1. BP-1472918407-172.17.0.2-1431707688874:blk_1073742267_1443 len=1048576 repl=3
2. BP-1472918407-172.17.0.2-1431707688874:blk_1073742268_1444 len=914060 repl=3
```

For each file listed by `-files`, append a block ID number, size, and the replication factor.

```
hdfs fsck /user/root -files -blocks
```

The `-blocks` option lists one or more block ID numbers below each file name listed by the `-files` option. Every replica of a data block shares the same block ID number. The number of block IDs depends on the size of the file and the block size used when the file was initially created.

In addition to block ID numbers, the `-blocks` option reports the block size and the number of replicas.

A block pool ID is automatically generated by the NameNode for each HDFS installation. The block pool ID is shared across all the DataNodes in an HDFS cluster.

The `-locations` Option

```
/user/root/big1 3011212 bytes, 3 block(s):  OK
0. BP-1472918407-172.17.0.2-1431707688874:blk_1073742266_1442 len=1048576 repl=3
[172.17.0.2:50010, 172.17.0.3:50010, 172.17.0.4:50010]
1. BP-1472918407-172.17.0.2-1431707688874:blk_1073742267_1443 len=1048576 repl=3
[172.17.0.2:50010, 172.17.0.3:50010, 172.17.0.4:50010]
2. BP-1472918407-172.17.0.2-1431707688874:blk_1073742268_1444 len=914060 repl=3
[172.17.0.2:50010, 172.17.0.4:50010, 172.17.0.3:50010]
```

For each data block, list the DataNodes that contain a replica.

```
hdfs fsck /user/root -files -blocks -locations
```

The NameNode keeps track of the locations of data block replicas. The `-locations` options caused `fsck` to report the DataNode location for each data block replica. Consider a scenario where there are a number of corrupt data blocks. Using `fsck` with the `-locations` option might help an administrator to determine that a disk controller or disk drive on a specific DataNode is failing. The failing storage hardware or the DataNode itself could be replaced.

The `-racks` Option

```
/user/root/big1 3011212 bytes, 3 block(s):  OK
0. BP-1472918407-172.17.0.2-1431707688874:blk_1073742266_1442 len=1048576 repl=3
  [/default-rack/172.17.0.2:50010, /default-rack/172.17.0.3:50010, /default-rack/
  172.17.0.4:50010]
1. BP-1472918407-172.17.0.2-1431707688874:blk_1073742267_1443 len=1048576 repl=3
  [/default-rack/172.17.0.2:50010, /default-rack/172.17.0.3:50010, /default-rack/
  172.17.0.4:50010]
2. BP-1472918407-172.17.0.2-1431707688874:blk_1073742268_1444 len=914060 repl=3
  [/default-rack/172.17.0.2:50010, /default-rack/172.17.0.4:50010, /default-rack/172.17.0.3:50010]
```

Prepend a rack name to each DataNode listed.

`hdfs fsck /user/root -files -blocks -locations -racks`

The `-racks` option prepends a rack name to each DataNode listed by the `-locations` option. The default rack name is `/default-rack`. The label `/default-rack` is used when there is only a single rack of systems used for the cluster, or when rack awareness has not been configured.

If rack awareness has been configured, then `/default-rack` is replaced by administrator-assigned rack names.

Rack awareness is described and configured in another lesson.

Problem Report Example

```
/user/root/dir1/passwd: Under replicated BP-1386861603-172.17.0.2-1434735465264
:blk_1073741883_1059. Target Replicas is 3 but found 1 live replica(s), 0 decommissioned replica(s) and 0 decommissioning replica(s).
```

fsck Problem Report

This is an example of an error or problem reported by `fsck`. This particular message was the result of configuring only a single-node cluster but leaving the default replication factor set to three. The NameNode is compelled to create three replicas of each data block yet it cannot do it without violating data block placement policy. The HDFS data block placement policy will not allow a single DataNode to contain more than one replica of any given data block.

dfsadmin

The `dfsadmin` tools are a collection of file system administration tools for performing a variety of tasks. For example, an administrator can use them to check HDFS health, status, and usage information. An administrator can also perform such tasks as forcing a checkpoint operation, transitioning a NameNode in and out of safe mode, or transitioning a DataNode in and out of the cluster. Later in this lesson you will see how to use the `dfsadmin` tools to manage HDFS quotas.

The basic syntax is shown here. It is `hdfs dfsadmin [options]`. There are currently over thirty options for the `dfsadmin` command. To display these options and get on-screen help, use the command `hdfs dfsadmin -help`.

Because `dfsadmin` is a set of administration tools, you must be the HDFS superuser in order to use the tools.

dfsadmin Examples

- To transition a NameNode into safemode:
`hdfs dfsadmin -safemode enter`
- To force a NameNode checkpoint operation that creates both a new `fsimage` and `edits` file:
`hdfs dfsadmin -saveNamespace`

- To create only a new edits file:
hdfs dfsadmin -rollEdits
- To exit NameNode safemode:
hdfs dfsadmin -safemode leave
- To download the latest fsimage file (useful for doing remote backups):
hdfs dfsadmin -fetchImage

Some of these commands are required when configuring NameNode HA.

Health, Status, and Usage Report

- hdfs dfsadmin -report can display status and usage information similar to the NameNode UI.

The summary section:

```
[hdfs@node1 ~]$ hdfs dfsadmin -report
Configured Capacity: 300000522240 (279.40 GB)
Present Capacity: 221331869696 (206.13 GB)
DFS Remaining: 219495591936 (204.42 GB)
DFS Used: 1836277760 (1.71 GB)
DFS Used%: 0.83%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
```

A section for each DataNode:

Live datanodes (3): Name: 172.17.0.4:50010 (node3) Hostname: node3 Decommission Status : Normal Configured Capacity: 100000174080 (93.13 GB) DFS Used: 612110336 (583.75 MB) Non DFS Used: 26222866432 (24.42 GB) DFS Remaining: 73165197312 (68.14 GB) DFS Used%: 0.61% DFS Remaining%: 73.17% Configured Cache Capacity: 0 (0 B) Cache Used: 0 (0 B) Cache Remaining: 0 (0 B) Cache Used%: 100.00% Cache Remaining%: 0.00% Xceivers: 2 Last contact: Tue May 19 19:51:27 EDT 2015
--



Running a Report with dfsadmin

Perhaps a more common `dfsadmin` operation is to generate a health, status, and usage report. The report is divided in sections.

The first section is a summary section providing general information about HDFS usage and state. The remaining sections, one for each DataNode, provide per-DataNode usage and state information. The Decommissioning Status has to do with transitioning a DataNode in or out of a cluster, and is described in another lesson.

Manual Configuration

HDFS can also be manually configured by editing its configuration files. Manually editing files can be time consuming and error prone. It also requires more knowledge and experience on the part of the administrator.

The most commonly edited file is `hdfs-site.xml`.

Others include `core-site.xml`, `hadoop-policy.xml`, `hdfs-log4j`, `ssl-client.xml`, `ssl-server.xml`.

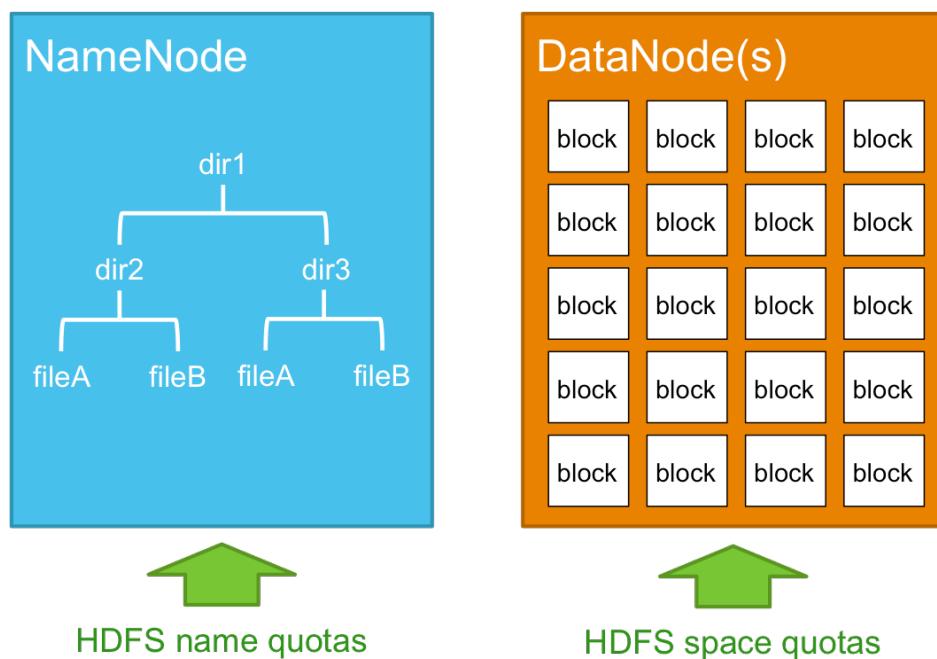
To manually change a configuration property, an administrator must locate and edit the correct configuration file. There are a number of possible configuration files although the `hdfs-site.xml` file is the most common one.

Because Hadoop and HDFS are cluster technologies, a change must commonly be made to multiple systems. One option is to manually edit the file on all the effected systems. A second option is to edit a single file and copy it to all the effected systems. Following this, all the effected service components must be restarted. This could include the NameNodes, one or more of the DataNodes, or both. Any other service that might be affected by a change must also be restarted.

Warning!

Manually editing configuration files is not compatible with Ambari administration. Manually modified configuration files are overwritten by the information in the Ambari database when the HDFS service is restarted.

HDFS Quotas



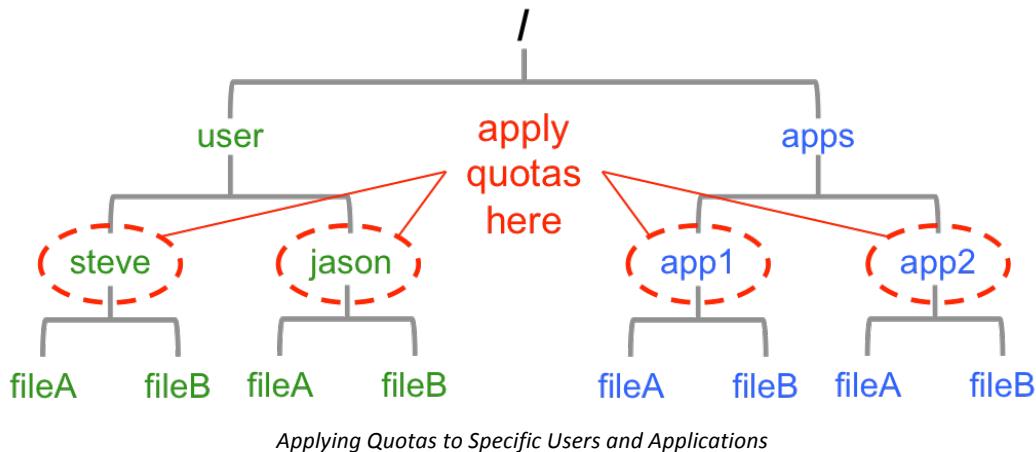
HDFS Name and Space Quotas

HDFS enables an administrator to establish two types of per-directory quotas. Name quotas limit the number of directory and file names allowed in a directory hierarchy. The parent directory name itself counts against the quota. Names consume file system inodes (index nodes), which are a finite resource. Name quotas prevent a user or application from consuming all the file system inodes.

Space quotas limit the number of bytes that can be consumed by a directory hierarchy. Each replica of a data block counts against a space quota. Space quotas only applied to data blocks on the DataNodes and not the metadata on the NameNode. Space quotas prevent a user or application from consuming all of the file system space.

Both name and space quotas are hard limits. Renaming a directory does not remove its quotas.

Limiting Specific Users and Applications



Even though quotas are configured per directory, it is still possible to apply quotas to specific users and applications. A user or application can be given a home directory with quota limits. If the user or application does not have write permissions in any other directory, their quota effectively limits them. Only the user or application should have write permissions in their home directory.

Creating a new file or directory name fails if it exceeds a name quota. During file writes, a block allocation fails if allocating an entire block would exceed a space quota. Based on the operation of name and space quotas, it is possible to create a new file name but not be able to write any data to it.

Configuring Quotas

You must be an HDFS superuser to administer quotas.

- **Setting a name quota on one or more directories:**
`hdfs dfsadmin -setQuota <n> <directory> [<directory>] ...`
Issue the command again to modify a name quota.
- **Removing a name quota on one or more directories:**
`hdfs dfsadmin -clrQuota <directory> [<directory>] ...`
- **Setting a space name quota on one or more directories:**
`hdfs dfsadmin -setSpaceQuota <n> <directory> [<directory>] ...`
Issue the command again to modify a space quota.
- **Removing a space quota on one or more directories:**
`hdfs dfsadmin -clrSpaceQuota <directory> [<directory>] ...`

An attempt to set a name or space quota will still succeed even if the directory would be in immediate violation of the new quota.

Viewing Quota Information

	space quota	remaining space available	number of bytes in directory (not replicated)	directory name
	QUOTA	REM_QUOTA	SPACE_QUOTA	CONTENT_SIZE PATHNAME
[root@node1 ~]# hdfs dfs -count -v -q /user/root	200	191	21474836480	15363052 /user/root
	name quota	remaining names available	current directory count	current file count

Viewing Quota Information

Any user may view current quota information using the HDFS Shell `count` command.

To view quota information: `hdfs dfs -count -v -q <directory_name>`

If a name quota has not been set, the QUOTA column displays none and the REM_QUOTA column displays inf, which means infinite.

It is possible for the REM_QUOTA and REM_SPACE_QUOTA columns to report negative numbers. This is commonly the result of setting a name or space quota below the directory's current name or space usage.

The CONTENT_SIZE column displays the total size of all the files in the directory's hierarchy. This size does not account for file replication. For example, in the illustration, if all files in the `/user/root` directory hierarchy used the default replication factor of three, then the actual space counted towards the space quota would be 3×15363052 .

Knowledge Check Questions

- 1) List examples of items that are tracked as part of the NameNode metadata.
 - 2) True or false? When writing a data block, an HDFS client must open a connection to each DataNode provided in the list from the NameNode.
 - 3) Regarding data block placement during write operations, what is the primary concern when deciding where to place the second replica?
 - 4) On a DataNode, what type of information is contained within the `blk_<block_ID>.meta` file?
 - 5) On a NameNode, what type of information is contained within an `fsimage` file?
 - 6) What is the purpose of safemode during NameNode startup?
 - 7) How does a NameNode determine DataNode availability?
 - 8) In the default HDP configuration, what happens to the DataNode service if only a single DataNode disk fails?
 - 9) True or false? The NameNode UI is a browser-based interface for status monitoring and directory browsing.
- 10) A common use for the DataNode UI is to view _____.
- 11) Which HDFS Shell command displays the number of bytes in a file or directory?
- 12) The `fsck` command reads information from the _____.
- 13) True or false? From the `fsck` report, only mis-replicated blocks require administrator action.
- 14) Why are manual configuration file edits not compatible with Ambari administration?
- 15) List the two types of HDFS quotas.

- 16) Quotas are configured per directory, so how are they applied per user?
- 17) What is the affect of running the command `hdfs dfsadmin -setQuota 100 /user/root`?
- 18) Which users may run the command to view quotas?
- 19) True or false? The number of bytes listed in the CONTENT_SIZE column in a quota report includes the bytes from replicated data blocks.

Knowledge Check Answers

- 1) List examples of items that are tracked as part of the NameNode metadata.

Permissions

Ownership

ACLs

Access and last modification times

Quotas

Block size

Replication level

- 2) True or false? When writing a data block, an HDFS client must open a connection to each DataNode provided in the list from the NameNode.

False, an HDFS client only writes to the first DataNode, which writes to the next DataNode, which in turn writes to the next DataNode, and so on...

- 3) Regarding data block placement during write operations, what is the primary concern when deciding where to place the second replica?

The primary concern is availability. HDFS chooses a DataNode on another rack, or if there is no other rack, at least another DataNode.

- 4) On a DataNode, what type of information is contained within the `blk_<block_ID>.meta` file?

Checksum information for its corresponding data block. One checksum for every 512 bytes of data.

- 5) On a NameNode, what type of information is contained within an `fsimage` file?

The fsimage file contains a static image of the state of the HDFS file system at a specific point in time.

- 6) What is the purpose of safemode during NameNode startup?

It is a read-only mode that provides the NameNode time to collect block reports from the DataNodes. The block reports are used to rebuild the in-memory block map used to locate and read file data.

- 7) How does a NameNode determine DataNode availability?

The NameNode listens for heartbeats from the DataNodes. Heartbeats should arrive every three seconds by default.

- 8) In the default HDP configuration, what happens to the DataNode service if only a single DataNode disk fails?

The DataNode stops offering its service.

- 9) True for false? The NameNode UI is a browser-based interface for status monitoring and directory browsing.

True

- 10) A common use for the DataNode UI is to view ***a block scanner report***.

11)Which HDFS Shell command displays the number of bytes in a file or directory?

The du command

12)The fsck command reads information from the ***NameNode***.

13)True or false? From the fsck report, only mis-replicated blocks require administrator action.

False. Only corrupt blocks require administrator action.

14)Why are manual configuration file edits not compatible with Ambari administration?

Because the Ambari database will be used to overwrite the manual configuration change when the service is restarted.

15)List the two types of HDFS quotas.

Name and space quotas

16)Quotas are configured per directory, so how are they applied per user?

Quotas are applied per user by applying quotas to a user's home directory.

17)What is the affect of running the command `hdfs dfsadmin -setQuota 100 /user/root?`

It limits the number of file and directory names that can be created beneath /user/root to 99

18)Which users may run the command to view quotas?

Any user may view current quota information using the HDFS Shell count command

19)True or false? The number of bytes listed in the CONTENT_SIZE column in a quota report includes the bytes from replicated data blocks.

False

Summary

- A NameNode maintains namespace, file and directory metadata, journaling information, and file block maps in memory.
- A DataNode maintains file data blocks and checksums and sends periodic block reports to the NameNode.
- Checksums are used to detect, replace, and remove corrupt data.
- Checkpointing periodically consolidates and persists NameNode information to disk. This information is used during NameNode startup.
- NameNodes listen for DataNode heartbeats in order to detect availability.
- The Ambari Web UI, the NameNode and DataNode UIs, and HDFS command-line tools can be used to manage HDFS.
- Manually editing configuration files is not compatible with Ambari management.
- Quotas limit user and application consumption of HDFS resources.
- Name quotas limit the number of files and directory names that can be created in a directory hierarchy.
- Space quotas limit the number of bytes that can be created in a directory hierarchy.
- Quotas are configured per directory by an HDFS superuser.
- Any user may view a quota report.

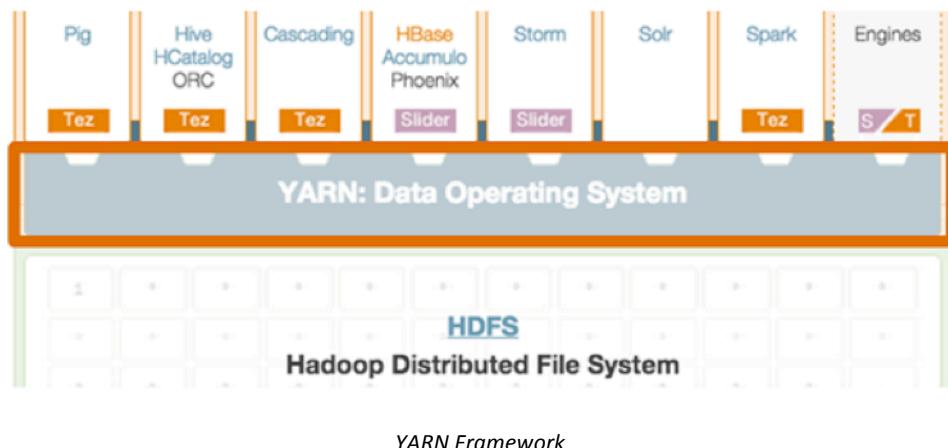
YARN Resource Management

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe YARN resource management
- ✓ Summarize YARN architecture and operation
- ✓ Identify and use YARN management options
- ✓ Summarize YARN response to component failure
- ✓ Describe YARN work-preserving restarts
- ✓ Describe YARN log aggregation

YARN Resource Management



YARN (unofficially “Yet Another Resource Negotiator”) is the computing framework for Hadoop. If you think about HDFS as the cluster file system for Hadoop, YARN would be the cluster operating system. It is the architectural center of Hadoop.

YARN’s Purpose

A computer operating system, such as Windows or Linux, manages access to resources, such as CPU, memory, and disk, for installed applications. In similar fashion, YARN provides a managed framework that allows for multiple types of applications – batch, interactive, online, streaming, and so on – to execute on data across your entire cluster. Just like a computer operating system manages both resource allocation (Which application gets access to CPU, memory, and disk now, and which one has to wait if contention exists?) and security (Does the current user have permission to perform the requested action?), YARN manages resource allocation for the various types of data processing workloads, prioritizes and schedules jobs, and enables authentication and **multitenancy**.

Multitenancy

Software multitenancy is achieved when a single instance of an application serves multiple groups of users, or “tenants.” Each tenant shares common access to an application, hardware, and underlying resources (including data), but with specific and potentially unique privileges granted by the application based on their identification. This is in contrast with **multi-instance architectures**, where each user gets a unique instance of an application, and the application then competes for resources on behalf of its tenant.

A typical example of a multitenant application architecture would be Software-as-a-Service (SaaS) cloud computing, where multiple users and even multiple companies are accessing the same instance of an application at the same time (for example, Salesforce CRM). A typical example of a multi-instance architecture would be applications running in virtualized or Infrastructure-as-a-Service (IaaS) environments (for example, applications running in KVM virtual machines).

NOTE

In prior versions of Hadoop, resource management was part of the MapReduce process. In that scenario, you had a single application handling both job scheduling and running data processing jobs at the same time. Starting with Hadoop 2.0, MapReduce is simply another data processing application running on top of the YARN framework.

YARN Architecture and Operation

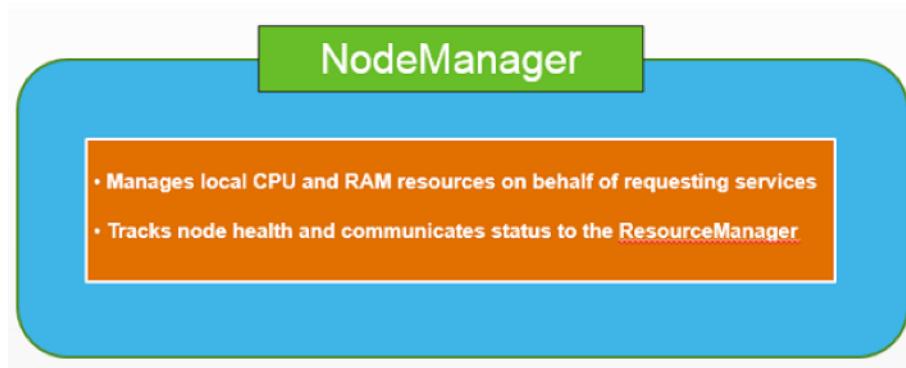


YARN Architecture Components

At a high level, the YARN framework is made up of two core components.

- The ResourceManager component runs on a master node and centrally manages resources globally for all YARN applications.
- The NodeManager component runs on each worker node in the cluster, and executes all tasks on its local resources as directed by the global ResourceManager component.

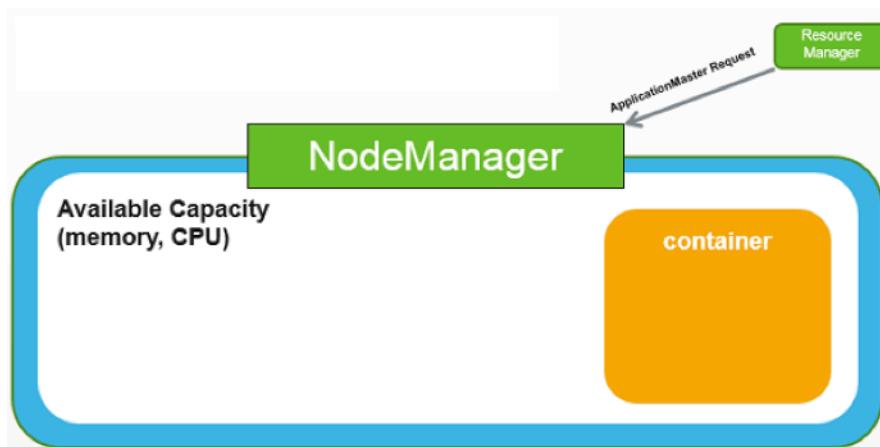
YARN NodeManager



YARN NodeManager Service

The YARN NodeManager is a daemon/service that runs on each worker node in the cluster. It manages local resources on behalf of the requesting services (such as the ResourceManager and ApplicationMasters). In addition, it tracks the health of the node and communicates its status with the ResourceManager.

Containers



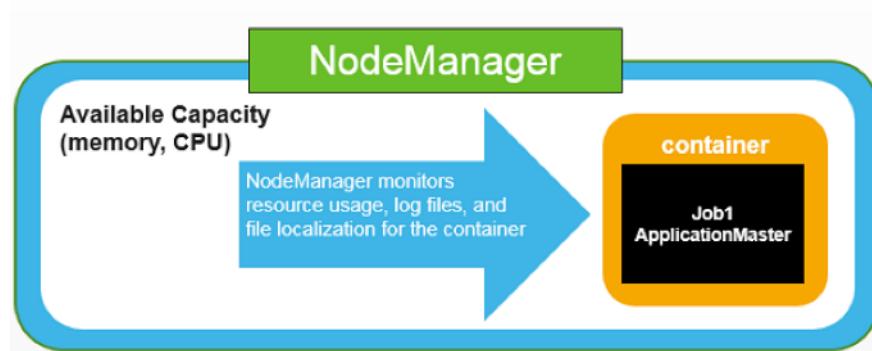
Container

When a ResourceManager makes an ApplicationMaster request (a request to launch an application and the jobs required to run it), the NodeManager begins by creating a container and allocating CPU and memory resources to it.

*Allocated Container*

A container is a unit of work within a YARN application that is allocated specific CPU and memory resources by the NodeManager on behalf of the ResourceManager. The container is the component that performs the work of the specific YARN application. A container is launched each time a new ApplicationMaster request is made by the ResourceManager. When a job is executed, the ApplicationMaster requests additional resources from the ResourceManager (via the NodeManager on which it is running). If additional resources can be allotted, the ResourceManager can then request additional containers to run that task from across the cluster.

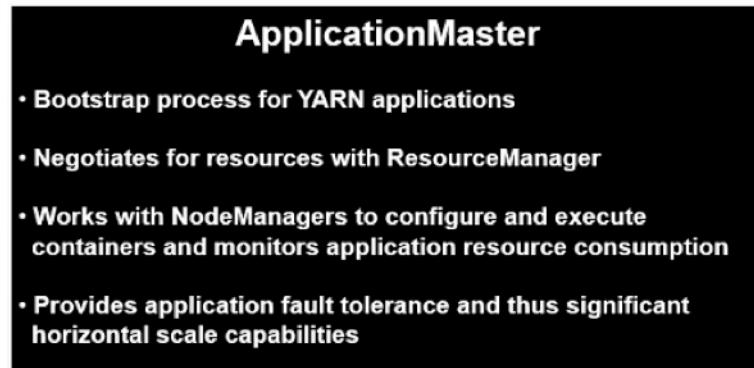
ApplicationMaster

*YARN ApplicationMaster*

The NodeManager manages logs generated by the container, monitors resource usage by the container, and manages files localization (downloading and organizing various file resources needed by containers, such as JAR files, code libraries, and other application-specific files).

Once the NodeManager spawns a container, the ApplicationMaster can be launched using the resources it controls.

Application Master



ApplicationMaster Process

The ApplicationMaster component serves as the bootstrap process that launches and manages everything for a YARN application once it gets past the application submission process and its own launch.

Each YARN application has its own type of ApplicationMaster. For example, Pig jobs require different ApplicationMaster code than Hive, and the Hive and Pig ApplicationMaster code would be very different than a Storm ApplicationMaster. Each ApplicationMaster, then, is an instance of that application's framework-specific library of code.

The responsibilities of the ApplicationMaster include:

- Negotiate appropriate resources (containers) from the ResourceManager
- Work with NodeManagers to execute and monitor containers and their resource consumption
- Determine the resource allocation per container
- Provide fault tolerance for applications

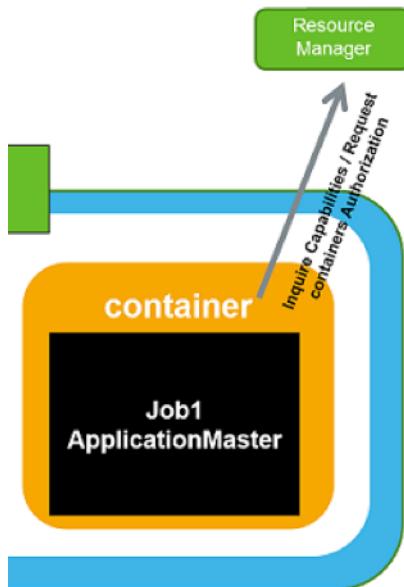
Many of the tasks traditionally performed by a cluster-level resource manager are actually pushed down to the application layer via the implementation of the ApplicationMaster. The global ResourceManager component is responsible for monitoring, tracking, and approving resource allocation requests, but once those requests are granted, the ApplicationMaster then handles all of the heavy lifting in terms of communicating with NodeManagers, tracking specific resource usage for the application, allocating and deallocating containers / resources as needed, and general progress monitoring for the application.

Application Fault Tolerance

The ApplicationMaster is also responsible for application fault tolerance, which critically means that the global ResourceManager is not responsible for this function either (making it a *pure scheduler*). Moving all of these functions out of a centralized resource management component and into the worker node layer allows for massive scalability to the tune of tens of thousands of nodes because the central ResourceManager is not nearly as much of a bottleneck as it otherwise would be (and was in previous versions of Hadoop where the scheduler was MapReduce).

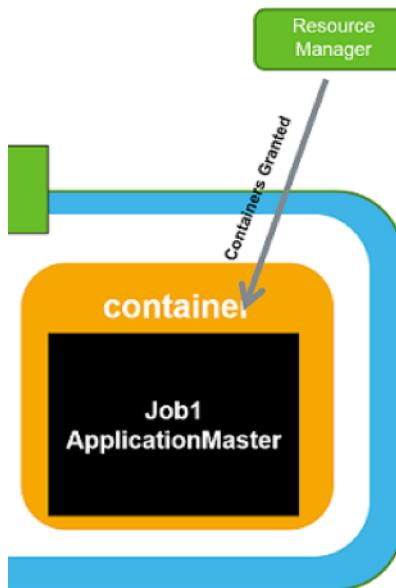
NOTE

From a security perspective, ApplicationMasters are essentially user-code that might or might not be validated services depending on what an organization allows in their Hadoop environment. Because of this, the YARN platform (ResourceManager and NodeManager components) has to protect itself at all costs from faulty or malicious ApplicationMasters and the resources granted to them. Even though the ApplicationMaster is responsible for a significant amount of monitoring and management, it is not a privileged service. All requests for resources/containers must be granted and executed by the ResourceManager and NodeManager components.

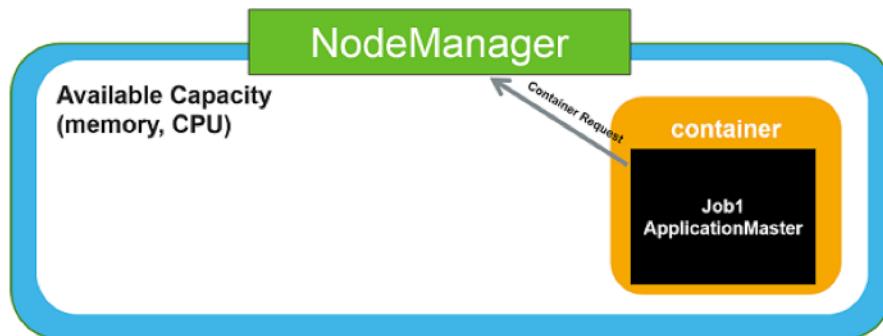
Job Scheduling

Job Scheduling – Inquire about Capabilities

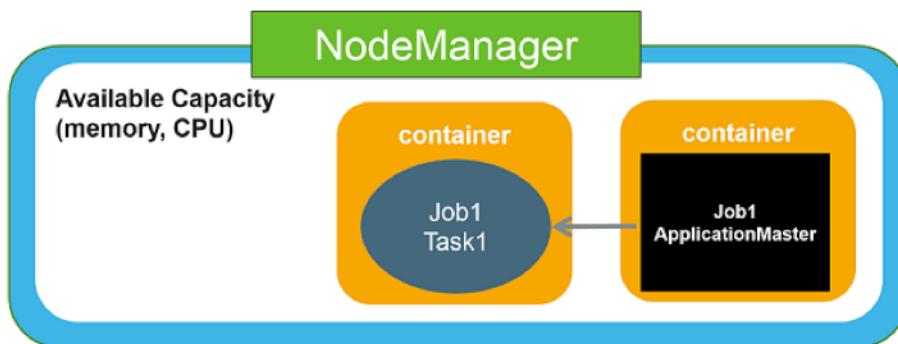
The ApplicationMaster starts by discovering cluster capabilities via an inquiry with the ResourceManager. It will then make a request for resources, and in response receive an authorization token granting it whatever resources it needs or is allowed.

*Job Scheduling – Containers Granted*

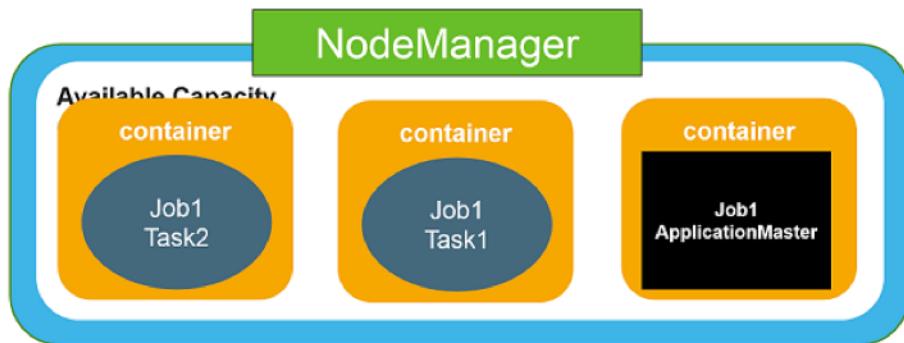
The ApplicationMaster is then responsible for communicating with the NodeManagers it will use to allocate these resources into containers.

*Job Scheduling – Communicate with NodeManager*

Once created, it will configure those containers to perform its required tasks.

*Job Scheduling – Configuring Containers*

This process is repeated until the ApplicationMaster has exhausted its allotted container resources or until all required jobs tasks have been assigned.



Job Scheduling – Repeat until Complete or all Allocated Resources are Used

NOTE

The images above represent a high-level overview of the communication process between the ApplicationMaster, ResourceManager, and NodeManager. Additional sub-requests and verification checks are performed in the process of completing the tasks shown that are not usually important from the perspective of the Hadoop administrator in terms of architecting and managing a cluster.

Role of the NodeManager

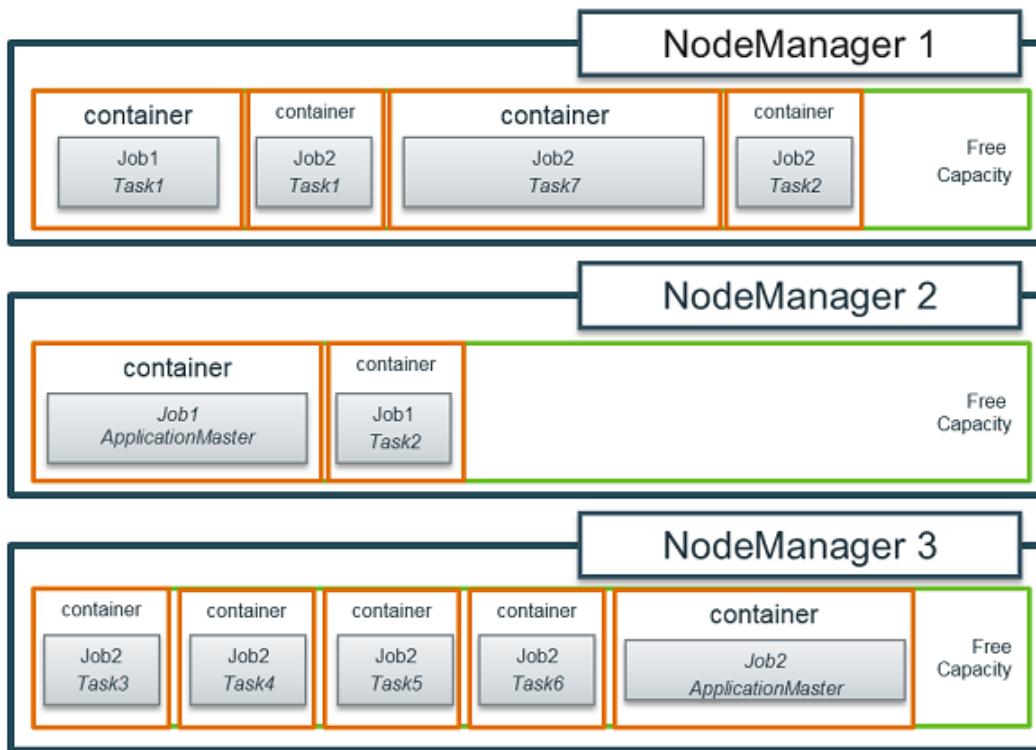
The NodeManager manages and monitors container resource usage, but does not have visibility into the application job tasks. The ApplicationMaster tracks and monitors the application resource usage and progress. Thus, if something is wrong with a job task, it is up to the ApplicationMaster to correct the issue. On the other hand, if the problem is at the container level, the NodeManager must correct the issue. The NodeManagers and ApplicationMasters across the cluster will be in frequent communication.

Note

While the example above shows the ApplicationMaster requesting the resources it needs from the NodeManager on which it is currently running, in reality the ApplicationMaster will attempt to schedule the resources on a worker node that contains the data it needs (thus moving the processing to the data, rather than requiring the cluster to move data to a worker node with available CPU and memory.)

Example

The following graphic illustrates how containers, ApplicationMasters, and job tasks might be spread across a three-node cluster.

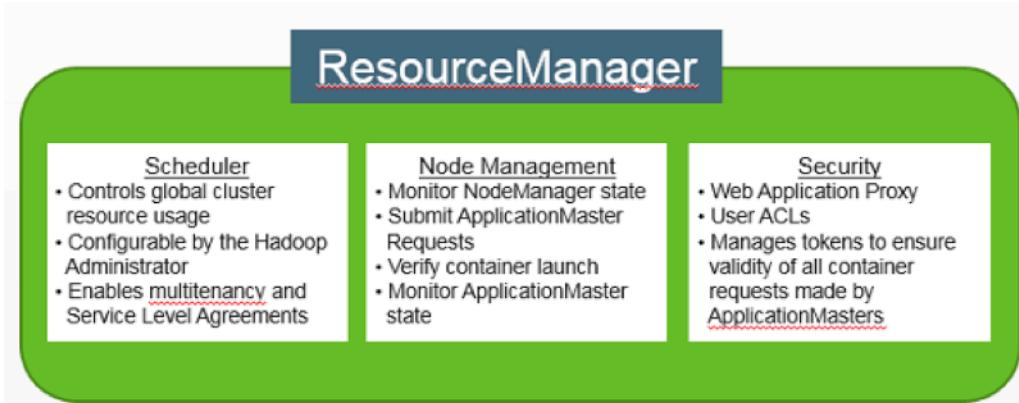


In this example, the Job1 ApplicationMaster was started on NodeManager 2. The first task for Job1 was started on NodeManager 1, and the second Job1 task was started on NodeManager 2. This completed all the tasks required for Job1.

The Job2 ApplicationMaster was launched on NodeManager 3. The first two Job2 tasks were launched on NodeManager 1, Job2 tasks 3 through 6 were launched on NodeManager 3, and the final Job2 task was launched back on NodeManager 1.

The main point to this is to illustrate that the ApplicationMaster can initiate the creation of containers on any appropriate NodeManager in the cluster. The default behavior is for all jobs to be collocated where data blocks already exist, even if more processing power is available on a node without those data blocks whenever possible.

YARN ResourceManager



YARN Resource Manager Services

The ResourceManager component is made up of a number of services that perform three main duties:

- Scheduling
- Node Management
- Security

Scheduling

The YARN Scheduler is a single component that controls resource usage according to parameters set by the Hadoop administrator. This provides efficiency by allowing different organizations to use a centrally pooled set of cluster resources (multitenancy) while at the same time controlling each tenant's access to those resources. This ensures that each organization can be guaranteed the minimum required resources needed in order to meet its Service Level Agreements. At the same time, this also allows organizations to access excess capacity not being used by the others, thus providing elasticity and lower overall cost of deployment. The scheduling mechanism used and specific settings are under the control of the Hadoop administrator.

Node Management

Node and ApplicationMaster management in the ResourceManager is accomplished via a number of services that perform a variety of tasks:

- Monitor NodeManagers for heartbeat (sent by NodeManager every second by default, expected within 10 minutes)
- Submit ApplicationMaster launch requests to appropriate NodeManagers
- Verify that resource container components were actually launched on appropriate NodeManagers (within 10 minutes) and attempts restart if required
- Monitor ApplicationMasters running in containers for heartbeat (expects one every 10 minutes) and attempts restart if required

NOTE:

Only the ApplicationMaster is monitored. Job task monitoring is the responsibility of the ApplicationMaster itself.

- Maintain a list of submitted ApplicationMasters across the cluster and their current state

Security

The ResourceManager serves as a Web application proxy and controls access to resources via ACLs (Access Control Lists). It manages resource/application security via token-based systems which verify that all container requests are valid. The ApplicationMaster must pass a verified containerToken to the NodeManager that contains information about the resources that should be allocated to that container. This checking mechanism prohibits a rogue ApplicationMaster from allocating more resources than it has been allotted by the ResourceManager.

YARN Component Summary

Below is a brief summary of all of the YARN components discussed in this lesson:

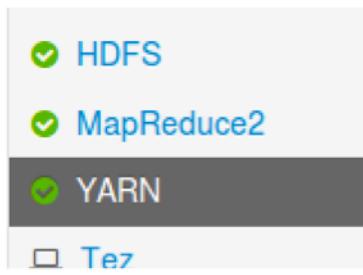
ResourceManager	NodeManager	Container	ApplicationMaster
Schedule global resources	Manage local memory and CPU allocation	Allocated RAM and CPU cores by NodeManager	YARN application bootstrap process
Enable multitenancy			Negotiate resources
Enable SLA enforcement			Provide application fault tolerance
Monitor and manage NodeManagers	Track and report on node health		Work with NodeManager for container restart
Monitor and manage ApplicationMasters	Manage file localization for containers	Run ApplicationMasters and job tasks	
Monitor containers globally	Monitor and manage local containers		Monitor job tasks and containers across cluster
Manage ACLs			
Manage Tokens			

YARN Management Options

There are four basic options an HDP administrator has for managing YARN are:

- Ambari
- The ResourceManager UI
- Command line management and manual editing / transferring of configuration files
- The YARN API

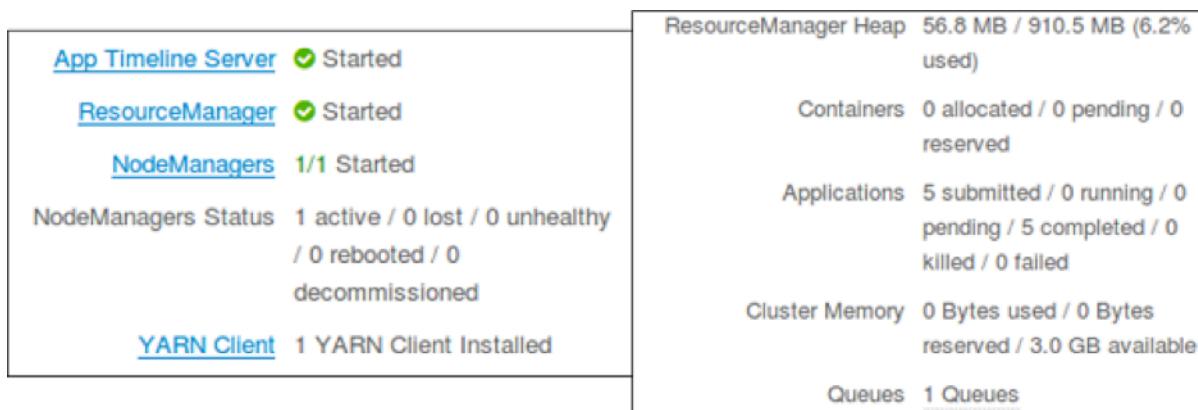
Ambari UI



Ambari Web UI Services Menu

When you go to the **Services** tab in the Ambari Web UI and select **YARN**, the resulting **Summary** tab displays a number of helpful menus and information.

Summary Information

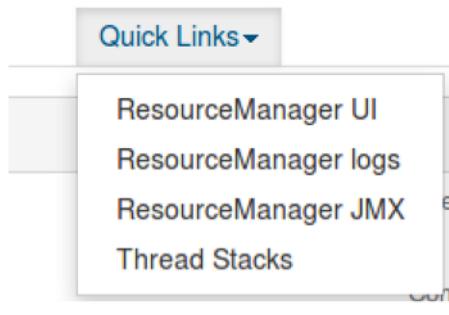


Ambari - YARN Service Summary Tab

For example, in the **Summary** section of the **Summary** tab you will be able to see the status of the ResourceManager as well as the number of NodeManagers and their collective status. To the right of this section, you can view information regarding:

- ResourceManager heap usage
- Number of containers allocated, pending, and reserved
- Number of applications submitted, running, pending, completed, killed, or failed
- Cluster memory used or reserved
- Number of queues

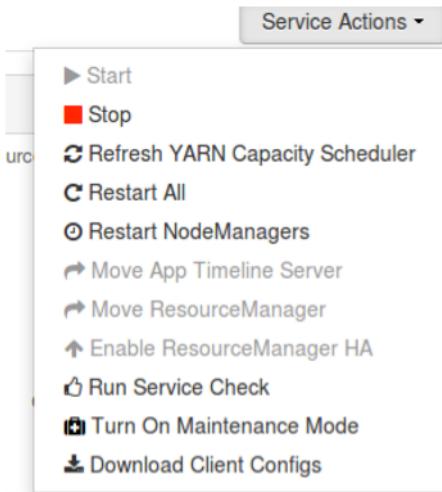
Quick Links



Ambari - YARN Quick Links

Under the **Quick Links** menu you will see links to the ResourceManager UI, ResourceManager logs, ResourceManager Java Machine Extension (JVM), and Thread Stacks.

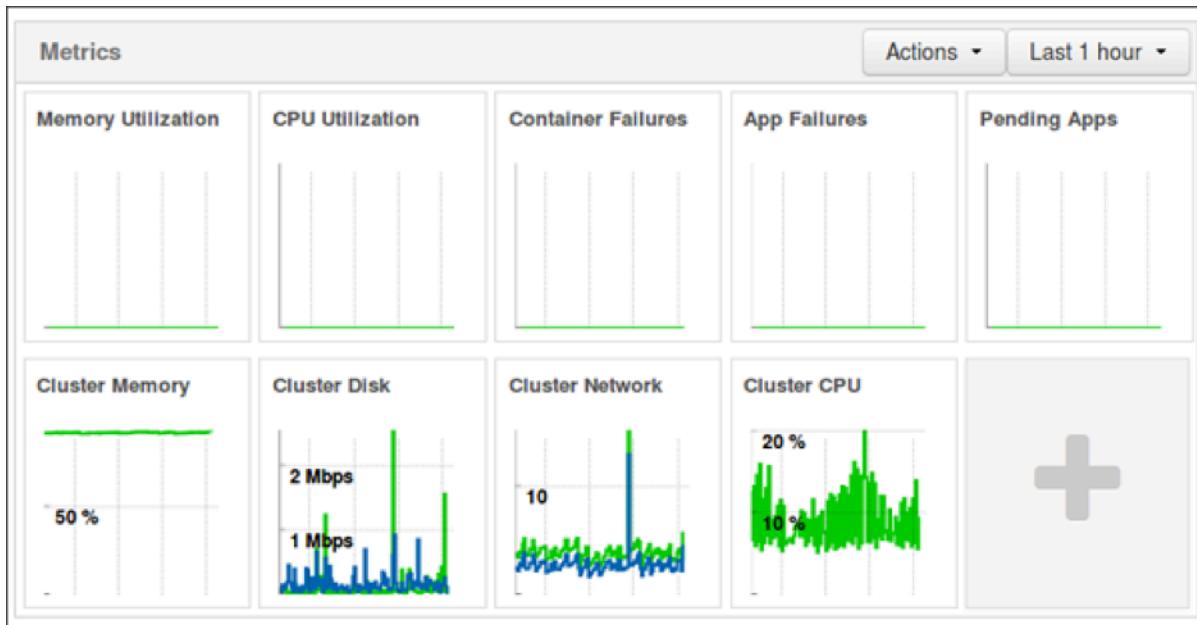
Service Actions



Ambari- YARN Service Actions Menu

Under the **Service Actions** menu button at the top-right of the screen, you have the option to stop and restart YARN as a whole, refresh the Capacity Scheduler, restart NodeManagers, and perform other tasks.

Summary Metrics



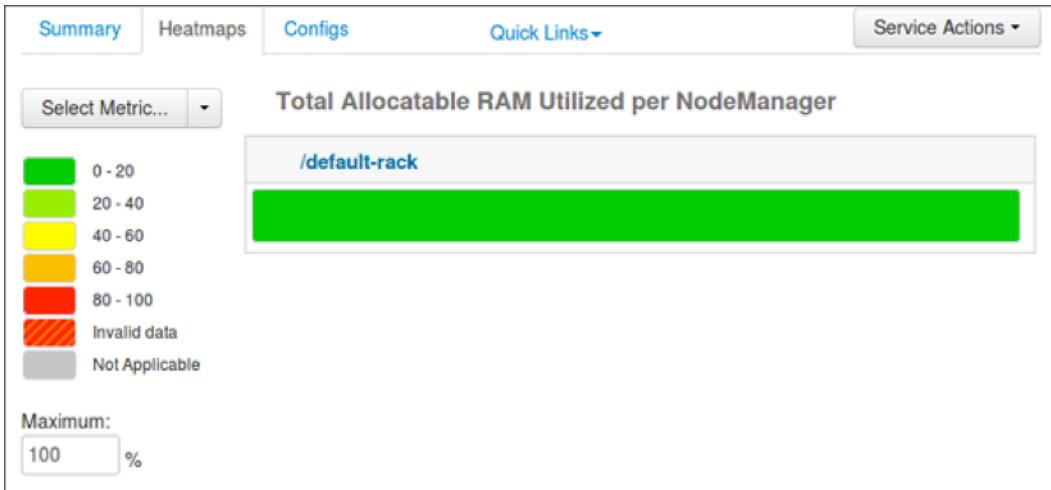
Ambari – YARN Service Metrics

At the bottom of the **YARN Services** tab you will find the **Metrics** section. The default widgets that display are:

- **Memory Utilization:** Percentage of total memory allocated to containers running in the cluster.
- **CPU Utilization:** Percentage of total virtual cores allocated to containers running in the cluster.
- **Container Failures:** Percentage of all launched containers failing in the cluster.
- **App Failures:** Percentage of all launched applications failing in the cluster.
- **Pending Apps:** Count of applications waiting for cluster resources to become available.
- **Cluster Memory:** Percentage of total NodeManager host memory in use.
- **Cluster Disk:** Sum of disk throughput for all NodeManager hosts.
- **Cluster Network:** Sum of network throughput for all NodeManager hosts.
- **Cluster CPU:** Percentage of CPU across all NodeManager hosts.

In addition, a Bad Local Disks widget can be added, which tracks the number of unhealthy local disks across all NodeManagers.

Heatmaps

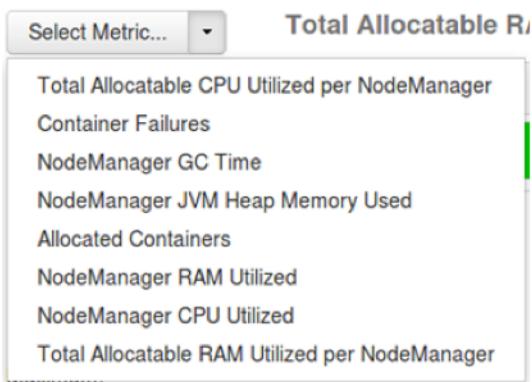


Ambari – YARN Heat Map

The YARN metrics available in the **Heatmaps** tab include:

- Total Allocatable CPU Utilized per NodeManager
- Container Failures
- NodeManager GC Time
- NodeManager JVM Heap Memory Used
- Allocated Containers
- NodeManager RAM Utilized
- NodeManager CPU Utilized
- Total Allocatable RAM Utilized per NodeManager

Selecting a Specific Metric to View



Ambari – Selecting a Specific Heat Map Metric to View

Configuration Management

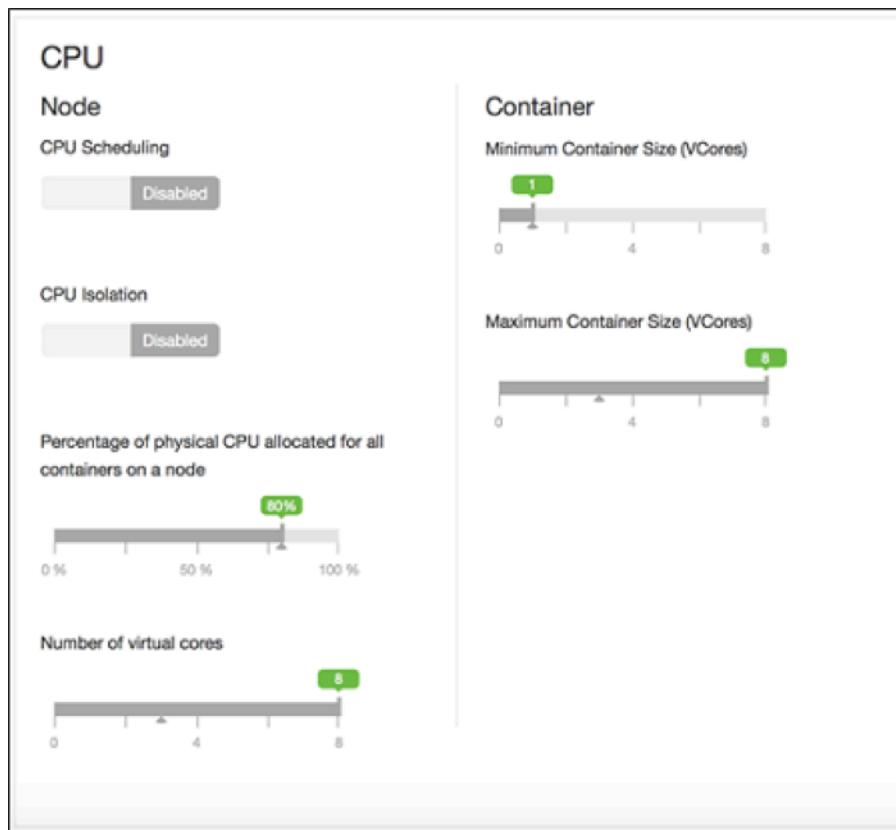
The screenshot shows the Ambari YARN Configuration Management interface. At the top, there are tabs for Summary, Heatmaps, Configs (which is selected), and Quick Links. Below the tabs, there's a group selector set to 'YARN Default (1)'. A 'Manage Config Groups' button is available. A search bar labeled 'Filter...' is present. The main area shows two configuration groups: V2 and V1, both created by admin 24 days ago using HDP-2.3. A message indicates 'admin authored on Fri, Jun 05, 2015 04:06'. There are 'Discard' and 'Save' buttons at the bottom. The interface is divided into sections: Memory (Node and Container settings with sliders for minimum and maximum memory sizes) and YARN Features (Node Labels and Pre-emption, both currently disabled).

Ambari – YARN Configuration Management

The **Configs** tab provides the ability to modify the default behavior of YARN and YARN applications. The default **Settings** sub-tab contains several customizable settings that can be configured via a GUI. These include:

- **Memory:** Controls the amount of node memory that can be allocated for containers, as well as the minimum and maximum size of those containers.
- **YARN Features:** Enable Node Labels to restrict YARN applications so that they can only run on nodes that have a specified label. Pre-emption allows the specification of higher-priority applications, which can reclaim resources from lower-priority applications in the event of resource contention.
- **CPU:** Enable CPU scheduling, which changes the default scheduler behavior to include both memory and CPU (rather than just memory) when making scheduling decisions. Enable CPU Isolation, which allows the isolation of CPU-heavy processes. When applicable (more on that in another lesson), this section also controls the percentage of physical CPU allocated for containers on a node, and the number of virtual cores allocated for containers. The Container settings set the minimum and maximum number of virtual cores (VCores) that can be allocated to any individual container.

YARN Resource Management



Ambari – CPU Scheduling

Advanced Configuration

The screenshot shows the Advanced Config Tab for the Resource Manager. It has two tabs: **Settings** and **Advanced**, with **Advanced** selected.

Resource Manager Settings:

- ResourceManager:** Set to `sandbox.hortonworks.com`.
- ResourceManager Java heap size:** Set to `250 MB`. Includes lock and copy icons.
- yarn.acl.enable:** Enabled (green checkmark). Includes lock and copy icons.
- yarn.admin.acl:** An empty text input field with lock and copy icons.
- yarn.log-aggregation-enable:** Enabled (green checkmark). Includes lock and copy icons.

Advanced Config Tab

The **Advanced** sub-tab of the **YARN Configs** tab provides an interface to highly tune the way YARN behaves, uses resources, and makes capacity scheduling decisions. Properties can be set for the following components:

Resource Manager: heap size, enable and define ACLs, and aggregate logs.

Node Manager: virtual-to-physical memory ratio, log location, local directory, remote application logs, auxiliary services, log retention, and heap size.

The screenshot shows the Ambari interface for configuring the Node Manager. The top navigation bar has 'Node Manager' selected. Below it, under the heading 'Node Manager', there is a list of configuration properties with their current values and edit icons (lock, plus, minus, and refresh).

Property	Value	Actions
yarn.nodemanager.vmem-pmem-ratio	10	Lock, Add, Subtract, Refresh
yarn.nodemanager.log-dirs	/hadoop/yarn/log	Lock, Add, Subtract, Refresh
yarn.nodemanager.local-dirs	/hadoop/yarn/local	Lock, Add, Subtract, Refresh
yarn.nodemanager.remote-app-log-dir	/app-logs	Lock, Add, Subtract, Refresh
yarn.nodemanager.remote-app-log-dir-suffix	logs	Lock, Add, Subtract, Refresh
yarn.nodemanager.aux-services	mapreduce_shuffle	Lock, Add, Subtract, Refresh
yarn.nodemanager.log.retain-second	604800	Lock, Add, Subtract, Refresh

Ambari – Advanced Config, Node Manager

Application Timeline Server: heap size, timeline services enabled, heartbeat intervals, timeline service Webapp settings, and state directory.

Application Timeline Server

App Timeline Server	sandbox.hortonworks.com
AppTimelineServer Java heap size	250 MB <input checked="" type="checkbox"/> <input type="button" value="C"/>
yarn.timeline-service.enabled	<input checked="" type="checkbox"/> <input type="button" value="C"/>
yarn.timeline-service.leveldb-timeline-store.path	/hadoop/yarn/timeline <input type="button" value="C"/>
yarn.timeline-service.leveldb-timeline-store.ttl-interval-ms	300000 <input type="button" value="C"/>
yarn.timeline-service.store-class	org.apache.hadoop.yarn.server.timeline.LevelDBTimelineStore <input type="button" value="C"/>
yarn.timeline-service.ttl-enable	<input checked="" type="checkbox"/> <input type="button" value="C"/>
yarn.timeline-service.ttl-ms	2678400000 <input type="button" value="C"/>

Ambari – Advanced Config, Timeline Server

General: YARN heap size.

General

YARN Java heap size	250 MB <input type="checkbox"/> <input type="button" value="C"/>
---------------------	--

Ambari – Advanced Config, Heap Size

- **Fault Tolerance:** enable service recovery, resource manager port setting, retry and max wait parameters, and HA enablement.

YARN Resource Management

Fault Tolerance

yarn.nodemanager. recovery.enabled	true	lock	reset	recycle
yarn.resourcemanager. recovery.enabled	true	lock	reset	recycle
yarn.resourcemanager. work-preserving- recovery.enabled	true	lock	reset	recycle
yarn.resourcemanager. zk-address	sandbox.hortonworks.com:2181	lock	reset	recycle
yarn.resourcemanager. connect.retry-interval.ms	30000	lock	reset	recycle
yarn.resourcemanager. connect.max-wait.ms	900000	lock	reset	recycle
yarn.resourcemanager. ha.enabled	false	lock	reset	recycle

Ambari – Advanced Config, Fault Tolerance

In addition, this section enables you to edit Capacity Scheduler settings (more on that in another lesson), Ranger settings, YARN environment settings, log settings, and YARN Web interface settings.

Custom ranger-yarn-audit

- Custom ranger-yarn-plugin-properties
- Custom ranger-yarn-policymgr-ssl
- Custom ranger-yarn-security
- Custom yarn-site

Advanced Config Settings

Many **Advanced** configuration settings will have default recommended settings, indicated by a blue recycle arrow. At any point, an administrator can click this arrow and the value will be reset back to a recommended value.

The screenshot shows the Ambari Settings interface with the 'Resource Manager' section selected. It includes fields for 'ResourceManager' (set to 'sandbox.hortonworks.com'), 'ResourceManager Java heap size' (set to '250 MB'), and several configuration parameters like 'yarn.acl.enable', 'yarn.admin.acl', and 'yarn.log-aggregation-enable'. A callout bubble points to the 'C' icon in the 'yarn.log-aggregation-enable' row, indicating it's a recommended setting.

Recommended Advanced Config Settings

ResourceManager UI

The screenshot shows the 'Quick Links' dropdown menu in the ResourceManager UI. It lists several options: 'ResourceManager UI', 'ResourceManager logs', 'ResourceManager JMX', and 'Thread Stacks'.

ResourceManager UI

The YARN ResourceManager UI can be accessed via the **Quick Links** menu on the YARN Services tab. It is also available by default by opening a Web browser to <http://<ResourceManagerHost>:8088>, although this is configurable in the **Advanced yarn-site** section under **Advanced** configuration tab.

Applications

The screenshot shows the 'Applications' section of the ResourceManager UI. It displays a list of application states: NEW, NEW SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, and KILLED.

ResourceManager Applications Information

YARN Resource Management

The ResourceManager UI allows a Hadoop administrator to access information on all YARN jobs in the cluster and filter the view by those types, including:

- New
- New Saving
- Submitted
- Accepted
- Running
- Finished
- Failed
- Killed jobs

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Progress	Tracking URL	Blacklisted Nodes
application_1435591840129_0005	ambariqa	word count	MAPREDUCE	default	Mon Jun 29 11:33:00 -0400 2015	Mon Jun 29 11:33:24 -0400 2015	FINISHED	SUCCEEDED	N/A	<div style="width: 100%;">History</div>	N/A	
application_1435591840129_0004	ambariqa	DistributedShell	YARN	default	Mon Jun 29 11:32:38 -0400 2015	Mon Jun 29 11:32:44 -0400 2015	FINISHED	SUCCEEDED	N/A	<div style="width: 100%;">History</div>	N/A	
application_1435591840129_0003	ambariqa	OrderedWordCount	TEZ	default	Mon Jun 29 11:32:20 -0400 2015	Mon Jun 29 11:32:36 -0400 2015	FINISHED	SUCCEEDED	N/A	<div style="width: 100%;">History</div>	N/A	
application_1435591840129_0002	ambariqa	PigLatinpigSmoke.sh	TEZ	default	Mon Jun 29 11:32:02 -0400 2015	Mon Jun 29 11:32:18 -0400 2015	FINISHED	SUCCEEDED	N/A	<div style="width: 100%;">History</div>	N/A	
application_1435591840129_0001	ambariqa	PigLatinpigSmoke.sh	MAPREDUCE	default	Mon Jun 29 11:31:27 -0400 2015	Mon Jun 29 11:31:48 -0400 2015	FINISHED	SUCCEEDED	N/A	<div style="width: 100%;">History</div>	N/A	

Ambari ResourceManager UI All Applications

It provides information about these jobs, including:

- Application ID
- User account that initiated the job
- the job Name
- Application Type
- Queue the job will run in
- job Start and Finish Time
- current job State
- Final Status
- Running Containers
- Progress

- Tracking UI
- Blacklisted Nodes

Clicking an application ID brings the administrator to another Web page, which provides additional information about that application.

The screenshot shows the Ambari ResourceManager UI for a specific application. The top navigation bar includes the Hadoop logo and the title "Application application_1435591840129_0004". The left sidebar has links for Cluster (About, Nodes, Node Labels, Applications), Scheduler, and Tools. The main content area is divided into two sections: "Application Overview" and "Application Metrics".

Application Overview:

- User: ambariqa
- Name: DistributedShell
- Application Type: YARN
- Application Tags:
- YarnApplicationState: FINISHED
- FinalStatus Reported by AM: SUCCEEDED
- Queue: default
- Started: Mon Jun 29 11:32:38 -0400 2015
- Elapsed: 6sec
- Tracking URL: History
- Log Aggregation Status: SUCCEEDED
- Diagnostics:

Application Metrics:

- Total Resource Preempted: <memory:0, vCores:0>
- Total Number of Non-AM Containers Preempted: 0
- Total Number of AM Containers Preempted: 0
- Resource Preempted from Current Attempt: <memory:0, vCores:0>
- Number of Non-AM Containers Preempted from Current Attempt: 0
- Aggregate Resource Allocation: 5635 MB-seconds, 7 vcore-seconds

At the bottom, there are search and navigation buttons: "Search", "First", "Previous", "1", "Next", "Last".

Ambari ResourceManager UI Specific Application Information

Some of this is available on the previous screen, but additional information includes:

- Total Resources and containers Preempted (if applicable)
- Aggregate Resource Allocation
- Drill-down capabilities into the Attempt ID, Node information, and logs for the specific application attempt

Additional Information

The screenshot shows the Ambari ResourceManager UI with the "Tools" link selected in the left sidebar. The sidebar contains the following links:

- Cluster
- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

ResourceManager Additional Links Menu

The ResourceManager UI contains a number of additional helpful pages available by clicking the appropriate link on the left-hand side of the page.

The **About** link provides information about the cluster as a whole, including resource information, version information, and Cluster ID.

The **Nodes** link provides information about individual nodes, including rack location, node state, node address and HTTP address, number of containers, memory and virtual CPU resources used and available, and the node software version.

The **Node Labels** link provides information about node labels.

The **Scheduler** link provides information about application queues.

The **Tools** sub-menu contains additional links that provide information about cluster configuration, logs, server stacks, server metrics, and ResourceManager errors and warnings.

Command Line

YARN can also be managed from a Command Line Interface (CLI). When managing YARN from a CLI, there are two basic command types: user and administration. The general usage format is `yarn COMMAND args...`

NOTE

All commands listed in this section would be preceded by `yarn`. For example, the `jar` command would actually be `yarn jar` from the command line. The `rmadmin` command would be `yarn rmadmin`.

User Commands

Command	Options and Arguments	Purpose
<code>jar</code>	<code><jar file></code>	Runs a jar file
<code>classpath</code>		Prints the class path needed to get the Hadoop jar file and required libraries
<code>version</code>		Prints the version of YARN
<code>application</code>	<code>-list</code> <code>[-appTypes <Types>]</code> <code>[-appStates <States>]</code>	<code>-list</code> Lists applications via the ResourceManager. Use additional arguments <code>-appTypes <Types></code> and <code>-appStates <States></code> to filter application list that is returned. <code><Types></code> include YARN, Tez, MapReduce, etc. <code><States></code> include SUBMITTED, KILLED, etc.
<code>node</code>	<code>-status <ApplicationID></code> <code>-kill <ApplicationID></code> <code>-list</code> <code>[states <States>]</code> <code>[-all]</code> <code>-status <NodeID></code>	Prints application status Kills an application Lists cluster nodes. Can be used with <code>-states <States></code> to filter nodes based on node state, or <code>-all</code> to list all nodes. Valid <code><States></code> example: RUNNING Prints a status report on the specified node

logs	<ul style="list-style-type: none"> -applicationId <applicationID> [-appOwner <AppOwner>] [-containerId <ContainerID>] [-nodeAddress <NodeAddress>] 	<p>NOTE: the “d” at the end of each ID is lower case, and the option is case sensitive!</p> <p>If application owner is not specified, assumed to be the current user.</p> <p>If either container ID or node address is specified, the other must also be specified.</p> <p><NodeAddress> format = nodename:port</p>
------	--	---

The `yarn jar` command executes a jar file. Additional syntax might include the `[mainClass]` as well as additional arguments.

The `yarn classpath` command prints the class path needed to get the Hadoop jar file and required libraries.

The `yarn version` command prints the current version of YARN running on the system.

The `yarn application` command has three primary options:

- `-list`: Lists applications via the ResourceManager. In addition, to additional sub-options are available when using this command: `-appTypes` and `-appStates`. The `-appTypes` option allows the user to filter the list by application type, including YARN, Tez, MapReduce, etc. The `-appStates` option allows the user to filter by application state. Valid states include ALL, NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, and KILLED.
- `-status`: prints the status of a given applicationID.
- `-kill`: kills an application, given its applicationID.

The `yarn node` command has two primary options:

- `-list`: provides information about nodes in the cluster. The `-list` option can be paired with either the `-all` option, or the `-states <States>` option. An example of a valid node state would be **RUNNING**.
- `-status`: provides a status report on a given node.

The `yarn logs` command dumps log information. The command requires the `-applicationId` option. (Note that the “d” at the end of the option is lower case, and the option is case sensitive.) It can also take three additional optional options:

- `-appOwner`: if an application owner ID is not provided, the current user ID will be used.
- `-containerId`: limits output to a specific container ID. Note – if this is specified, the node ID must also be specified for the command to work properly.
- `-nodeAddress`: specifies the node on which a specific container ID is running. If this is specified, the container ID must also be specified. The node address format should be nodename:port.

Administrator Commands

Command	Options and Arguments	Purpose
<code>resourcemanager</code>		Start the ResourceManager daemon

nodemanager		Start the NodeManager daemon
proxyserver		Start the web proxy server
daemonlog	-getlevel <host:port> <name>	Prints the log level of the named daemon running at <host:port>
	-setlevel <host:port> <name> <level>	Sets the log level of the named daemon running at <host:port> <level> options include DEBUG, INFO, NOTICE, WARN, etc.
rmadmin	-refreshQueues	Reloads all queues ACLs, states and scheduler properties
	-refreshNodes	Refresh host information at the ResourceManager
	-refreshUserToGroupsMappings	Refresh user-to-groups mappings
	-refreshSuperUserGroupsConfiguration	Refresh superuser proxy group mappings
	-refreshAdminAcls	Refresh administration ACLs
	-refreshServiceAcl	Refresh service-level authorization policy file
	-getGroups [username]	Return groups the specified user belongs to
	-help [COMMAND]	Displays help for the given command
	-transitionToActive <serviceID>	Transitions service into Active state
	-transitionToStandby <serviceID>	Transitions service into Standby state
	-getServiceState <serviceID>	Returns the state of the service
	-checkHealth <serviceID>	Requests a health check for the specified service

There are three administrative YARN commands that restart core services:

- `yarn resourcemanager`: starts (or restarts) the central ResourceManager daemon
- `yarn nodemanager`: starts (or restarts) the NodeManager daemon
- `yarn proxyserver`: starts (or restarts) the Web proxy server

The `yarn daemonlog` command takes one of two options: `-getlevel` or `-setlevel`. To either get or set the log level for a particular daemon, you specify the `host:port` you wish to work with as well as the name of the daemon you wish to either monitor or set. In addition, if you use the `-setlevel` option, you must also specify the logging level you want to change to: for example, DEBUG, INFO, NOTICE, WARN, etc.

The `yarn rmadmin` command runs the ResourceManager administration client. It has 12 options available:

- **-refreshQueues:** Reload the queues' acls, states and scheduler specific properties. ResourceManager will reload the `mapred-queues` configuration file.
- **-refreshNodes:** Refresh the hosts information at the ResourceManager.
- **-refreshUserToGroupsMappings:** Refresh user-to-groups mappings.
- **-refreshSuperUserGroupsConfiguration:** Refresh superuser proxy groups mappings.
- **-refreshAdminAcls:** Refresh ACLs for administration of ResourceManager.
- **-refreshServiceAcl:** Reload the service-level authorization policy file ResourceManager will reload the authorization policy file.
- **-getGroups [username]:** Get groups the specified user belongs to.
- **-help [COMMAND]:** Displays help for the given command or all commands if none is specified.
- **-transitionToActive <serviceID>:** Transitions the service into Active state.
- **-transitionToStandby <serviceID>:** Transitions the service into Standby state.
- **-getServiceState <serviceID>:** Returns the state of the service.
- **-checkHealth <serviceID>:** Requests that the service perform a health check. The `rmadmin` tool will exit with a non-zero exit code if the check fails.

Manual Configure File Editing

When managing a cluster using Ambari, manually editing of YARN configuration files accomplishes little. The files are overwritten by Ambari when the service is restarted, thus erasing any changes that are made. However, if you are not using Ambari, the settings that can be modified via the **YARN Services > Configs** tab are available in a collection of files located at `/etc/hadoop/conf`. The three primary files for basic YARN configuration are:

- `yarn-site.xml`: overrides default parameters contained in `yarn-default.xml`, which is embedded in the `hadoop-yarn-common-<version number>.jar` file. Contains most of the YARN-specific parameters for the ResourceManager, NodeManager, and Timeline Server components. Settings not listed here will default to their `yarn-default.xml` values.
- `yarn-env. (sh/cmd) - .sh` file for Linux installations, `.cmd` for Windows installations. Sets YARN environmental variables and Java heap size configuration settings.
- `capacity-scheduler.xml` – sets Capacity Scheduler parameters.

The `/etc/hadoop/conf` directory contains a large number of additional configuration files where additional properties can be read (and in non-Ambari environments, written) for components like log4j, Ranger, SSL, HDFS, and general Hadoop settings.

REST API

YARN also has a REST API available for managing and interacting with the cluster. Following is a sample list of simple YARN API GET calls that can be made using the `cURL` application:

ResourceManager

```
curl -X GET http://node1:8088/ws/v1/cluster/info
curl -X GET http://node1:8088/ws/v1/cluster/metrics
curl -X GET http://node1:8088/ws/v1/cluster/scheduler
curl -X GET http://node1:8088/ws/v1/cluster/apps
curl -X GET http://node1:8088/ws/v1/cluster/nodes
```

NodeManager

```
curl -X GET http://node1:8042/ws/v1/node/info
curl -X GET http://node1:8042/ws/v1/node/apps
curl -X GET http://node1:8042/ws/v1/node/containers
```

Note that when communicating with the ResourceManager port 8088 must be specified by default, and when communicating with a NodeManager port 8042 is the default. At the time of this writing, Timeline Server APIs had not been published, but when they are, the default port will be 8188.

These API calls are examples of the same ones used by the ResourceManager UI to display administrative information.

For example, the API call `GET http://node1:8088/ws/v1/cluster/info` provides information that is also visible by opening a Web browser to the URL `http://node1:8088/cluster/cluster`.

REFERENCE

There are dozens and dozens of API calls available that allow you to perform management of resources at the cluster level, node level, and application level, as well as perform a variety of queries into historical performance metrics. Refer to the following online resources for the complete list of available APIs:

<http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/WebServicesIntro.html>
<http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/ResourceManagerRest.html>
<http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/NodeManagerRest.html>

Check out <http://hortonworks.com/docs> for updated information and specific examples of how to manage applications and other cluster resources.

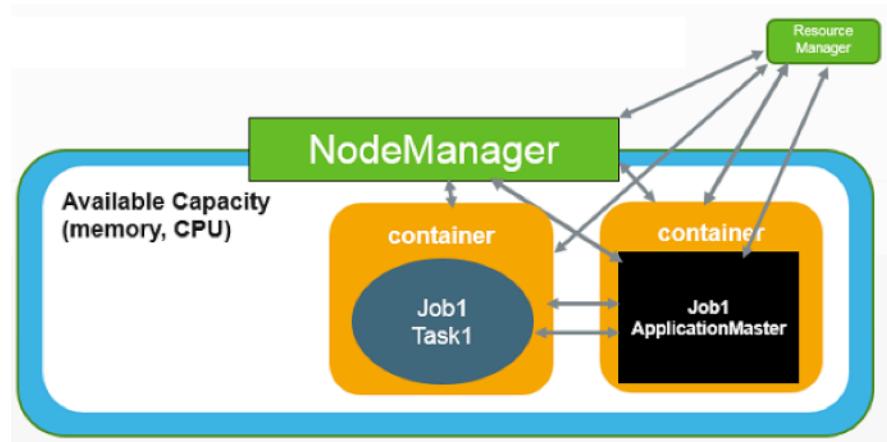
YARN Component Failure Management

Werner Vogels, CTO of Amazon.com, has been famously quoted as saying:

"Everything fails, all the time."

When designing Web, cloud, and Big Data solutions at massive scale, it becomes nearly impossible to avoid component failures from time to time. Sometimes the failures can be due to problems with hardware (for example, hard drive failure, power supply failure, etc.) Other times, the failure can occur at the platform level – for example, at any level in the YARN component hierarchy. Still other failures can occur with the applications that are running on the platform. In all cases, the primary goal when designing the system should be to minimize application or platform downtime rather than component failure. Assume that any component can fail at any time, and design the system such that when this happens, recovery is both automated and immediate, or at the very least easy and quick.

Monitoring



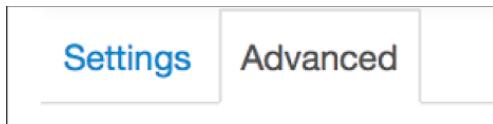
Failure Monitoring in YARN

With YARN, failures can occur at the following layers:

- Job/container
- ApplicationMaster
- NodeManager
- ResourceManager

The diagram above illustrates the failure monitoring that occurs in YARN. As you will note, the containers, NodeManagers, and ResourceManager are in constant communication, ensuring that each of the components they believe are there are still there. The components are designed to survive or respawn in the event of any failures, meaning that no single component (including the ResourceManager) is a point of failure for the entire cluster. Properly written YARN ApplicationMasters will also be in communication with the containers and job tasks that they spawn, and will be programmed to remediate the situation in the event of a failure.

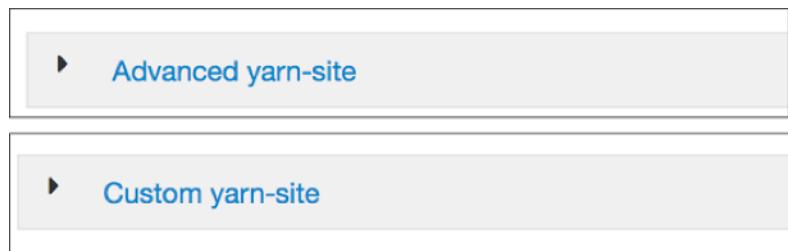
Configure Failure Monitoring



Ambari Services > YARN > Configs > Advanced

YARN defaults to a number of values when it comes to components monitoring each other. These components can be modified in the Ambari Web UI by visiting **Services > YARN > Configs > Advanced**.

Modifications



Modifications via Advanced yarn-site or Custom yarn-site Sections

Most modifications will either be made by going to the **Advanced yarn-site** section of the Ambari Web UI, or by adding them via the **Add Property ...** link under the **Custom yarn-site** section.

A few additional relevant properties are listed under the **Fault Tolerance** section.

Changes made here will propagate to `/etc/hadoop/conf/yarn-site.xml` when the YARN service is restarted.

It is important to note that in most cases, the default settings will function well at a very high scale. However, should a particular application or architecture decision dictate that failure detection and recovery times need to be increased or decreased, an administrator should know what can be changed and where to go to change it.

ResourceManager Check Properties

Key	Default Value	Description	Location
yarn.am.liveness-monitor.expiry-interval-ms	600000	Time to wait until an ApplicationMaster is considered dead	Custom yarn-site
yarn.resourcemanager.container.liveness-monitor.interval-ms	600000	Time to wait until an container is considered dead	Custom yarn-site
yarn.nm.liveness-monitor.interval-ms	600000	Time to wait until an NodeManager is considered dead	Custom yarn-site

The YARN ResourceManager has a number of default settings that control how frequently other components are monitored for failure by the ResourceManager and when they are considered to be dead, prompting the ResourceManager to remediate the issue. Some key examples include:

- `yarn.am.liveness-monitor.expiry-interval-ms = 600000`

(600,000 milliseconds = 10 minutes)

Defines how long the ResourceManager waits to hear from an ApplicationManager before it is considered dead.

- `yarn.resourcemanager.container.liveness-monitor.interval-ms = 600000`

(600,000 milliseconds = 10 minutes)

Defines how long the ResourceManager waits to hear from a container before it is considered dead.

- `yarn.nm.liveness-monitor.interval-ms = 600000`

(600,000 milliseconds = 10 minutes)

Defines how long the ResourceManager waits to hear from a NodeManager before it is considered dead.

All of the examples above can be edited under **Services > YARN > Configs > Advanced > Custom yarn-site (Add Property ...)**

NodeManager Check Properties

Key	Default Value	Description	Location
yarn.nodemanager.container-monitor.interval-ms	3000	How often NodeManager checks on containers	Advanced yarn-site
yarn.nodemanager.health-checker.interval-ms	600000	How often NodeManager runs the health checker script	Advanced yarn-site
yarn.nodemanager.disk-health-checker...	0.25 90	Parameters for disk health: min 25% healthy, max 90% utilization, min 1000 MB free space	Advanced yarn-site
(3 available)	1000		
yarn.resourcemanager.nodemangers.heartbeat-interval-ms	1000	How often NodeManager sends heartbeat to ResourceManager	Custom yarn-site

The YARN NodeManager has a number of default settings that control how frequently containers are monitored for failure, how frequently an overall system health check script is executed, parameters which the health check must fit within in order to continue to list itself as a healthy, available node, and how often the NodeManager sends a heartbeat to the ResourceManager. Some key examples include:

- `yarn.nodemanager.container-monitor.interval-ms = 3000`

(3,000 milliseconds = 3 seconds)

How often NodeManager checks on containers.

- `yarn.nodemanager.health-checker.interval-ms = 600000`

(600,000 milliseconds = 10 minutes)

How often NodeManager runs health script

- `yarn.nodemanager.disk-health-checker.min-healthy-disks = 0.25`

Sets the minimum percentage of healthy disks threshold to 25%.

- `yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage = 90`

Sets the maximum disk utilization per available disk threshold to be 90%.

- `yarn.nodemanager.disk-health-checker.min-free-space-per-disk-mb = 1000`

Sets the minimum amount of free disk space per disk threshold to 1,000 MB.

All of the examples above can be edited under **Services > YARN > Configs > Advanced > Advanced yarn-site**. In addition, you can configure how often the NodeManagers send a heartbeat to the ResourceManager with the following property:

- `yarn.resourcemanager.nodemangers.heartbeat-interval-ms = 1000`

(1,000 milliseconds = 1 second)

This can be edited under **Services > YARN > Configs > Advanced > Custom yarn-site (Add Property ...)**

YARN Component Failure Recovery

What happens when a YARN component detects failure in another component?

Container/Job Task Recovery

When a container or job task attempt has been determined to have failed, an exception can be propagated back to the ApplicationMaster that launched the job task. Unresponsive tasks can also be noticed and killed based on configuration of the ApplicationMaster itself. The NodeManager does not manage job tasks, but it does periodically check the state of the containers it is running. In the event a failure is detected, the ApplicationMaster and NodeManager will coordinate regarding a restart of the container. If necessary, the ApplicationMaster will also request new container from the ResourceManager (for example, if the original NodeManager is unable to restart the container for any reason).

ApplicationMaster Recovery

Up to two attempts to start a failed ApplicationMaster will be made by default. When an ApplicationMaster failure is detected, the ResourceManager will start a new instance running in a new container. The new ApplicationMaster can pick up where the old one left off, thus recovering the state of running and completed containers without the need to completely restart the job.

NodeManager Recovery

When a NodeManager is detected as having failed after 10 attempts to reestablish a connection, the ResourceManager blacklists it, removing it from the list of available nodes. Any ApplicationMasters and containers running on the failed NodeManager will be recovered by the ResourceManager and the replacement ApplicationMaster. In addition to the ResourceManager actions, an ApplicationMaster running on another NodeManager can detect that the NodeManager has failed and blacklist it.

ResourceManager Recovery

When a ResourceManager fails, what happens next depends on how it was configured. If work-preserving restarts have been configured, no launched application will need to be restarted as a result of the failure. If HA has been configured and enabled, the backup ResourceManager will become primary and no interruption of service will occur. No second attempt will be made to connect after a failed attempt. If HA has not been configured, then a Hadoop Administrator must manually restart the component. Assuming no HA has been configured, no new jobs or resource requests can be submitted while the ResourceManager is down. However, once a new ResourceManager has been initialized, the state of the previous ResourceManager instance can be recovered from ZooKeeper (again, assuming work-preserving restart has been configured). The new ResourceManager instance can then update the status of the rest of the running components and essentially start at the point where the old ResourceManager stopped. While the ResourceManager is down, ApplicationMasters and NodeManagers can still continue to process all jobs that are running prior to the ResourceManager failure, but any resource requests that had not been fulfilled prior to shut down must be resubmitted to the new ResourceManager to be processed.

YARN Work-Preserving Restarts

YARN supports the concept of a work-preserving restart, which in theory allows for the recovery of the state of a cluster or component in the event that component is restarted, without the need to restart dependent components. This feature is turned on by default in HDP 2.3. There are two types of work-preserving restarts: ResourceManager and NodeManager.

ResourceManager

A ResourceManager work-preserving restart allows for recovery from failure of the ResourceManager component for any reason (software or hardware failure) without the need to re-launch running applications. Once a ResourceManager is restarted or a new one initialized, the applications on the cluster can simply pick up where they left off prior to the interruption.

NodeManager

NodeManager work-preserving restart allows for recovery from software-based failure and restart of the NodeManager component without the need to re-launch running containers or job tasks on that node, assuming no other components on the server were affected by whatever caused the NodeManager to restart. In the event of a complete server crash (rather than just the NodeManager component) all components would need to be restarted.

Default Configuration

Key	Default Value
yarn.resourcemanager.recovery.enabled	true
yarn.resourcemanager.work-preserving-recovery.enabled	true
yarn.resourcemanager.store.class	org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore
yarn.resourcemanager.am.max-attempts	2
yarn.resourcemanager.zk-address	<host>:2181
yarn.resourcemanager.zk-state-store.parent-path	/rmstore
yarn.resourcemanager.zk-num-retries	1000
yarn.resourcemanager.zk-retry-interval-ms	1000
yarn.resourcemanager.zk-timeout-ms	10000
yarn.resourcemanager.zk-acl	world:anyone:rwcda
yarn.nodemanager.recovery.enabled	true
yarn.nodemanager.recovery.dir	/var/log/hadoop-yarn/nodemanager/recovery-state-dir
yarn.nodemanager.address	0.0.0.0:45454

Work-preserving restarts are configured automatically during installation, as indicated in the table above. Three components are configured: the ResourceManager, ZooKeeper State Store (which maintains the current state of the ResourceManager), and NodeManagers, whose current state is stored on their local file system by default.

Another value that can be tuned but is not usually necessary:

- `yarn.resourcemanager.work-preserving-recovery.scheduling-wait-ms = 10000`

(10,000 milliseconds = 10 seconds)

Determines how long the ResourceManager waits before allocating new containers on work-preserving recovery to give the NodeManagers time to “settle” before resuming application work.

YARN Log Aggregation

YARN log aggregation is enabled by default in HDP 2.3. A NodeManager generates and maintains logs for the activities it is managing or participating in and stores them on its local file system. Once an activity is complete, the log file is then aggregated to a central location in HDFS for longer-term storage and analysis. This avoids the need to truncate log files on the local file system to conserve disk space, and also allows for the ability to view all log files from a central web UI – by default, the Job History Server.

Default Configuration

Key	Default Value
yarn.log-aggregation-enable	true
yarn.log-aggregation-retain-seconds	2592000 (30 days)
yarn.log.server.url	http://<HistoryServer>:19888/jobhistory/logs
yarn.nodemanager.log-aggregation.compression-type	gz
yarn.nodemanager.log-aggregation.debug-enabled	false
yarn.nodemanager.log-aggregation.num-log-files-per-app	30
yarn.nodemanager.log-aggregation.roll-monitoring-interval-seconds	-1
yarn.nodemanager.remote-app-log-dir	/app-logs
yarn.nodemanager.remote-app-log-dir-suffix	logs

The values in the table above represent the default YARN log aggregation settings in HDP 2.3. YARN log aggregation is enabled by a checkbox under the Resource Manager section of **Services > YARN > Configs > Advanced** in the Ambari Web UI. All other settings are configurable under the **Advanced yarn-site** section.

NOTE

-1 = never.

Knowledge Check Questions

- 1) What two daemons / services make up the YARN framework?
- 2) What construct is granted CPU and memory resources?
- 3) What daemon / service is responsible for application fault tolerance?
- 4) What daemon / service controls access to global cluster resources?
- 5) What daemon / service makes resource requests after an application has launched?
- 6) What daemon / service manages file localization for applications?
- 7) True or False? By default, the various tasks for a single application / job will run on the same node if sufficient memory and CPU resources are available on that node.
- 8) The YARN component that enables multitenancy and adherence to Service Level Agreements is called the _____.
- 9) List the YARN management options



- 10)What does this icon mean under **Services > YARN > Configs > Advanced**?
- 11)Where can you quickly view the amount of utilized memory for the NodeManagers in your cluster on a per-machine basis?
- 12)Which Web interface provides detailed information on running applications?
- 13)What CLI command will display a list of every node in the cluster regardless of its status?
- 14)Assume you do not believe the information from the previous command is accurate. Which CLI command will update the node information at the ResourceManager?
- 15)True or False? When a cluster is managed by Ambari, manual changes to `yarn-site.xml` will go into effect after the service is restarted.
- 16)What are the ports used to access ResourceManager and NodeManager REST APIs?
- 17)YARN is designed to be able to recover from failure of which components?
- 18)What component of the YARN stack is not directly monitored for failure by the ResourceManager?
- 19)Name the sections under **Services > YARN > Configs > Advanced** where YARN monitoring behavior can be modified.
- 20)By default, how long does the ResourceManager wait until concluding that a NodeManager has failed?
- 21)By default, how frequently does the NodeManager send a heartbeat to the ResourceManager if it is functioning properly?
- 22)Which YARN component is responsible for restarting a failed ApplicationMaster?
- 23)Which HDP feature allows applications to continue rather than fail and relaunch after a ResourceManager failure?
- 24)Which HDP feature centralizes YARN logging?

Knowledge Check Answers

1) What two daemons / services make up the YARN framework?

ResourceManager and NodeManager

2) What construct is granted CPU and memory resources?

Containers

3) What daemon / service is responsible for application fault tolerance?

ApplicationMaster

4) What daemon / service controls access to global cluster resources?

ResourceManager

5) What daemon / service makes resource requests after an application has launched?

ApplicationMaster

6) What daemon / service manages file localization for applications?

NodeManager

7) True or False? By default, the various tasks for a single application / job will run on the same node if sufficient memory and CPU resources are available on that node.

False. The tasks will be directed to whatever nodes contain the data by default, and will thus often be split across multiple nodes.

8) The YARN component that enables multitenancy and adherence to Service Level Agreements is called the _____.

ResourceManager

9) List the YARN management options.

Ambari Web UI, ResourceManager UI, Command line and manual configuration, and YARN API

10)What does this icon mean under Services > YARN > Configs > Advanced?

Reset that property to the recommended default



11)Where can you quickly view the amount of utilized memory for the NodeManagers in your cluster on a per-machine basis?

Services > YARN > Heatmaps

12)Which web interface provides detailed information on running applications?

ResourceManager UI

13)What CLI command will display a list of every node in the cluster regardless of its status?

`yarn nodes -list -all`

14)Assume you do not believe the information from the previous command is accurate. Which CLI command will update the node information at the ResourceManager?

`yarn rmadmin -refreshNodes`

15)True or False? When a cluster is managed by Ambari, manual changes to

`yarn-site.xml` will go into effect after the service is restarted.

False. Manual changes will be overwritten by Ambari when the service restarts.

16)What are the ports used to access ResourceManager and NodeManager REST APIs?

ResourceManager = 8088. NodeManager = 8042.

17) YARN is designed to be able to recover from failure of which components?

Any of them. ResourceManager, NodeManager, container, ApplicationMaster, and if ApplicationMaster is designed correctly, job tasks as well.

18) What component of the YARN stack is not directly monitored for failure by the ResourceManager?

Job tasks – these are monitored only by the ApplicationMaster.

19) Name the sections under **Services > YARN > Configs > Advanced** where YARN monitoring behavior can be modified.

Fault Tolerance (?), Advanced yarn-site, and Custom yarn-site.

20) By default, how long does the ResourceManager wait until concluding that a NodeManager has failed?

10 minutes

21) By default, how frequently does the NodeManager send a heartbeat to the ResourceManager if it is functioning properly?

Every one second

22) Which YARN component is responsible for restarting a failed ApplicationMaster?

ResourceManager

23) Which HDP feature allows applications to continue rather than fail and relaunch after a ResourceManager failure?

Work-preserving restart

24) Which HDP feature centralizes YARN logging?

YARN log aggregation

Summary

- YARN is a collection of components that work together to schedule resources across a cluster for various types of workloads and enable multitenancy
 - ResourceManager, NodeManager, container, ApplicationMaster, and job tasks
- YARN can be managed by the Ambari Web UI, ResourceManager UI, command-line, and REST API calls
 - In non-Ambari environments, direct edits can be made to configuration files as well
- YARN is designed to be able to recover from failure of any component with the minimum amount of downtime and lost application progress
- Work-preserving restart capabilities are configured by default at the ResourceManager and NodeManager
- YARN logs are centrally aggregated by default and visible via the Job History Server

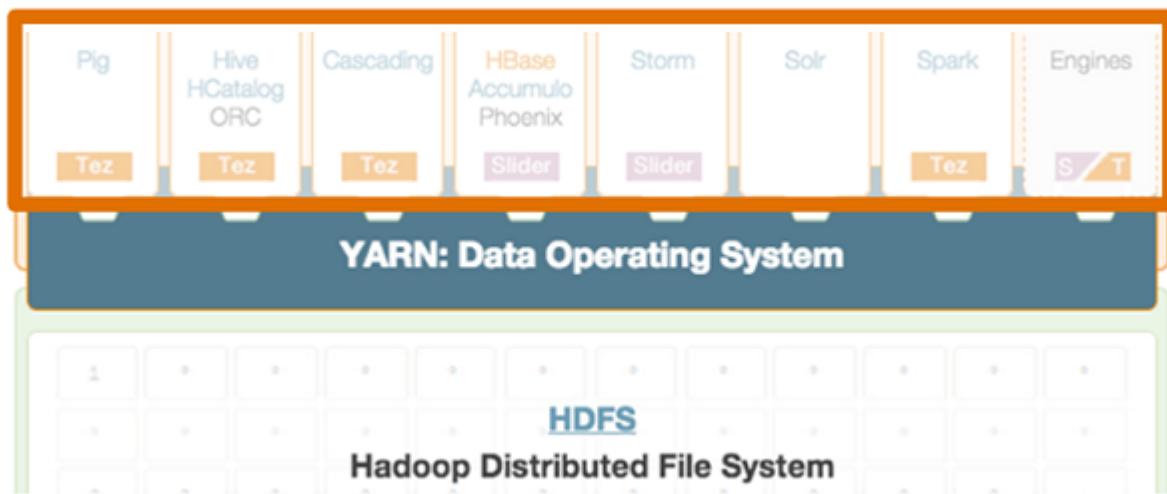
YARN Applications

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Describe the basics of running simple YARN applications
 - MapReduce and Tez
 - Apache Pig
 - Apache Hive

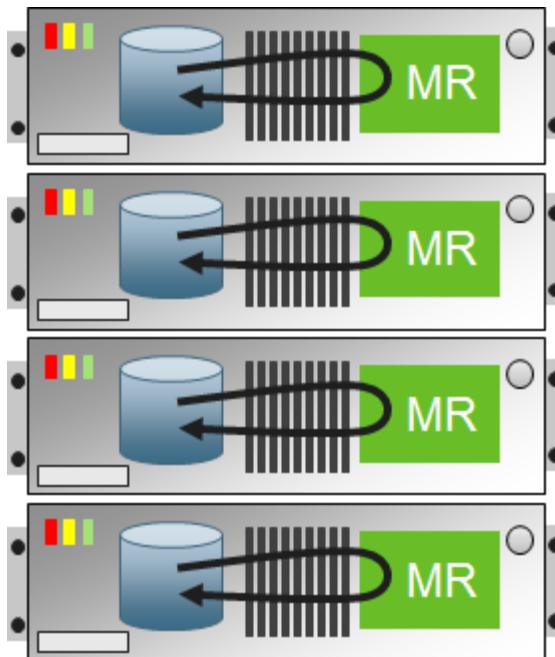
YARN Applications



YARN is a framework for managing resources used by Hadoop applications. The applications themselves are comprised of additional frameworks, which at a minimum include the logic component and one of the Hadoop data processing frameworks such as MapReduce2 or Tez.

The image above is actually out of date in some respects at the time of this printing. For example, HDP now supports Spark running natively on YARN as opposed to requiring an intermediary like Tez to do so. The general principle of how applications interact with YARN, however, remains the same.

Traditional MapReduce Applications



MapReduce, now known as MapReduce2 as of HDP 2.0, is a solution for scaling data processing. MapReduce2 is not a program. It is a framework to write distributed data processing programs. Programs written using the MapReduce framework have successfully scaled across thousands of machines.

MapReduce2 works by distributing the program code across the machines in a cluster. The code on each machine, shown as MR in the illustration, works on just the piece of data that is typically (but not always) stored on the local machine. Results from each machine are aggregated at the end.

Historically MapReduce was a single component that managed all data processing and resource scheduling in Hadoop. Today, the resource management is handled by YARN, so MapReduce2 is simply a data processing framework. Even that function is being supplanted, however, by more efficient data processing frameworks such as Tez. Tez is now the default framework as of HDP 2.3.

MapReduce2 applications were (and still are) written in a variety of programming and scripting languages. Examples include Java, Python, and Perl. MapReduce2 could also be utilized by HDP users using the mapred command.

Apache Pig and Apache Hive

HDP offers platforms to ease application development by providing Hadoop users with interpreted languages that enable them to leverage existing knowledge of SQL and similar scripting languages. Two popular options include Apache Pig and Hive.

Apache Pig

Apache Pig is a high-level platform for transforming or analyzing large datasets. Pig includes a scripted, procedural-based language that excels at building data pipelines to aggregate and add structure to data. Pig also provides data analysts with tools to analyze data.

Pig was developed at Yahoo! years ago to make it easier for data analysts to perform complex data transformations using a simple scripting language. Data analysts can use Pig without having to learn the complexities of writing custom Tez or MapReduce2 programs. Pig's infrastructure layer includes a compiler that automatically converts a Pig script to a sequence of MapReduce2 jobs or a single Tez job.

Pig and Hive are complementary tools that are often used together. Pig is often used to pre-structure data for Hive, a process known as ETL (Extract, Transform, Load). Hive's query language is then great for asking a question of your data.

Ambri Pig View and Grunt Shell

Name	Last Executed	Last Results	Actions
No pig scripts have been created. To get started, click New Script.			

Ambari offers the ability to create an instance of the Pig View, which allows HDP users to interact with Pig via the Ambari Web UI interface.

Pig can also be interacted with from the command line via the Grunt shell, which is accessed by typing `pig` at the command line.

Both the Pig View and the Grunt Shell can be used to initiate Pig scripts or to run individual commands.

Apache Hive

Hive is a data warehouse infrastructure built on top of Hadoop. It was designed to enable users with database experience to analyze data using familiar SQL-like statements. Hive includes a SQL-like language called Hive Query Language, or HQL. Hive and HQL enable an enterprise to utilize existing skillsets to quickly derive value from a Hadoop deployment.

Hadoop was built to collect, store, and analyze massive amounts of data. As such, the Hadoop distributed file system, called HDFS, is a reservoir of data from multiple sources. The data is often a mix of unstructured, semi-structured, and structured data. Hive provides a mechanism to project structure onto HDFS data and then query it using HQL. However, there is a limit to what Hive can do. Sometimes it is necessary to use another tool, like Apache Pig, to pre-format the unstructured data before processing it using Hive.

If you are familiar with databases, then you understand that unstructured data has no schema associated with it. If you are not familiar with database schemas, they define the columns of a database along with the type of data in each column. Data types include such things as a string, an integer, a floating point number, or a date.

A Hive installation includes a metastore database. Several database types are supported by Hive including an embedded database used for development or testing, or an external database like MySQL used for production deployments. To project structure on HDFS data, HQL includes statements to create a table with user-defined schema information. The table schema is stored in the metastore database.

The user-defined schema is associated with the data stored in one or more HDFS files when you use HQL statements to load the files into a table. The format of the data on HDFS remains unchanged but it appears as structured data when using HQL commands to submit queries.

Ambri Hive View

Ambari offers the ability to create an instance of the Hive View, which allows HDP users to interact with Hive via the Ambari Web UI interface.

Hive can also be interacted with from the command line using Hive CLI or the newer Beeline CLI. For simple demonstration purposes, course labs will use Hive CLI, which is accessed by typing `hive` at the command line.

Both the Hive View and the CLI tools can be used to initiate Hive scripts or to run individual commands.

Summary

- MapReduce was the original data processing framework.
- Tez is now the default data processing framework.
 - Less use of disk storage (in-memory processing capabilities),
 - Does not require a map phase before a reduce,
 - Containers can be “pre-warmed” and reused,
 - Tez jobs can interact and downstream jobs can accept data and begin processing before the upstream job is completed
- Apache Pig is a scripting language often used for ETL operations.
- Apache Hive is a SQL-like data warehouse infrastructure built on Hadoop.

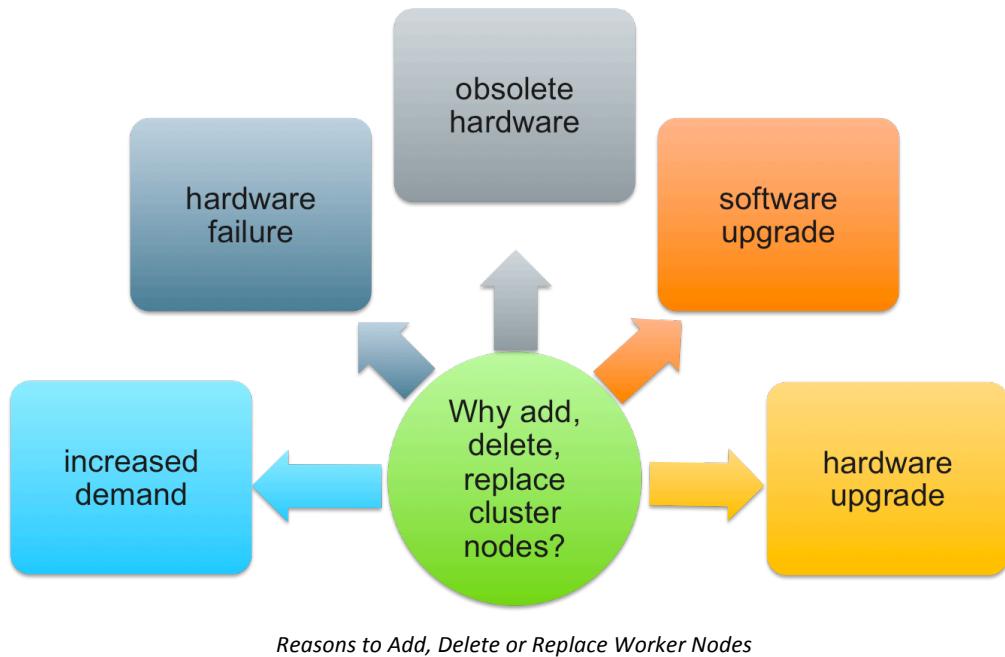
Adding, Deleting, and Replacing Worker Nodes

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Identify reasons to add, replace, and delete worker nodes
- ✓ Add a worker node
- ✓ Configure and run the HDFS Balancer
- ✓ Delete a worker node
- ✓ Move a master component

Working with Cluster Nodes



There are many reasons to add, delete, or replace a worker node. Several reasons are illustrated here.

- A cluster might need to grow over time to meet increased resource demand. A cluster can grow by adding more nodes or by adding additional hardware resources to existing nodes.
- Hardware can and will fail. Failed hardware needs to be repaired or replaced. In either case, the node must be removed from the cluster. A new or repaired node must be added back in to the cluster.
- Hardware becomes obsolete over time and must be replaced by newer hardware. The obsolete hardware will need to be removed and replaced by the newer hardware.
- Software must be periodically upgraded for a variety of reasons. Reasons include improved reliability, stronger security, and additional functionality. To perform a software upgrade, a node must be removed from the cluster, upgraded, and then added back to the cluster.

- Hardware can be upgraded to add additional CPU, memory, storage, and network resources. To perform a hardware upgrade, a node must be removed from the cluster, upgraded, and then added back in to the cluster.

Adding, deleting, or replacing a node will most often involve the worker nodes due to their larger number and the fact that the worker nodes provide most of the cluster's computational and storage resources.

Recommendations When Adding or Replacing Nodes

Use these recommendations when adding or replacing worker nodes:

- First, do not skip proper hardware burn-in testing. Detecting hardware problems before adding a node to a cluster helps to avoid the additional inconvenience and downtime associated with removing a failing or failed node from a cluster.
- Hardware models from a manufacturer improve over time. Motherboards support more and faster CPUs, more memory is possible, and network and disk controllers change. The result is that newer hardware purchased is often different from previous hardware purchases, even if they are the same model.
- As the system hardware resources change the cluster configuration settings must typically also change. As new sets of worker nodes are added to a cluster, the result is that the existing systems might require one set of configuration settings while the newer systems might require a different set. Ambari configuration groups was designed to address this scenario.
- Because of configuration issues, the recommendation is to purchase groups of identical new systems and then use Ambari configuration groups. Ambari configuration groups distribute different sets of configuration files to different sets of systems as directed by the administrator. Using Ambari configuration groups was described in another lesson.
- Optionally it is also possible to use YARN node labels to ensure that an application runs on only certain nodes. For example, a Spark application uses memory extensively. If newer nodes have significantly more and faster memory, then YARN node labels could be used to ensure that the Spark application runs only on the newer nodes. YARN node labels were described in another lesson.

Adding a Worker Node Using the Ambari Web UI

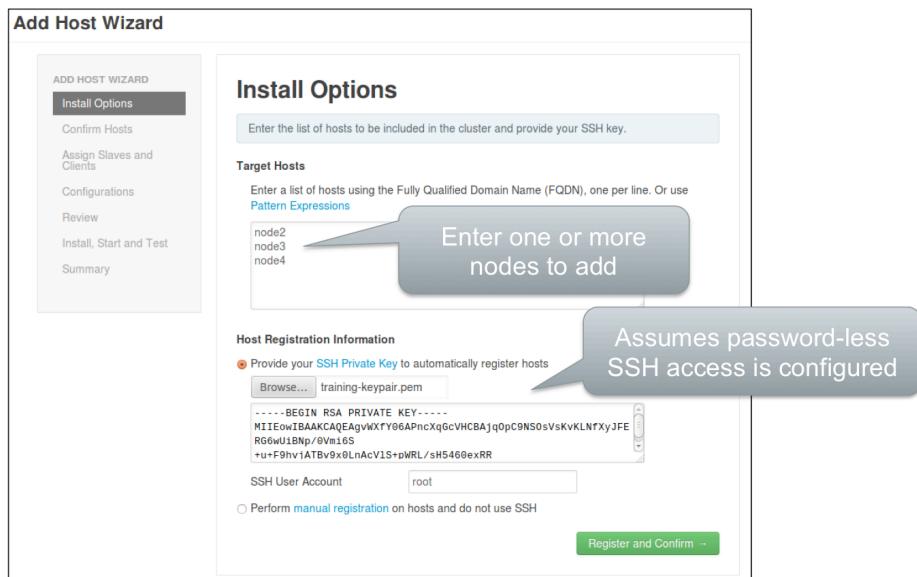
	IP Address	Rack	Cores	RAM	Disk Usage	Load Avg	Versions	Components
172.17.0.2	/default-r...	4 (4)	7.79GB	HDP-2.3.0.0-241022	Components			

Add New Hosts - Ambari Web UI

The Ambari Web UI can be used to add new nodes to a cluster. The process is wizard driven. To start the process, log in to the Ambari Web UI and click the **Hosts** page, then the **Actions** menu button, and then select **Add New Hosts**.

Select Nodes for Ambari Agents

- Enter the hostnames of the nodes to add.
 - Production clusters typically must use fully-qualified domain names.
- Choice of the **Host Registration Information** button depends on whether password-less SSH has been configured.



Installing an Ambari Agent

Ambari agents must be installed on each node and then registered with the Ambari Server before the Ambari Server can install HDP on the nodes. Type the fully-qualified domain name (FQDN) of each node in the **Target Hosts** text box. If you choose to use short host names instead, the installer will open a warning window when you click **Register and Configure**. Use of FQDNs is highly recommended to ensure proper cluster operation in all situations.

The Host Registration Information section includes two radio buttons. Which radio button is selected depends on whether or not you have pre-configured the Ambari Server machine with password-less SSH access to the cluster nodes.

- If password-less SSH has been pre-configured, select the first radio button and click **Browse**. Use the Browse window to locate the pre-configured SSH RSA private key file that enables the Ambari Server to log in to each node. The Hortonworks online manuals at <http://docs.hortonworks.com> include instructions for configuring password-less SSH log in.
- If password-less SSH log in has not been pre-configured, select the second radio button. This opens a window warning you that you must manually install and configure Ambari agent software on each node. Once installed and properly configured, an agent will automatically register with the Ambari Server. Manually installing and registering Ambari agents is described in the Hortonworks online documentation at <http://docs.hortonworks.com>.

After the choices have been made, click **Register and Confirm**.

Install and Register Ambari Agents

<input type="checkbox"/>	Host	Progress	Status	Action
<input type="checkbox"/>	node2	<div style="width: 100%; background-color: #2e7131;"></div>	Success	<input type="button" value="Remove"/>
<input type="checkbox"/>	node3	<div style="width: 100%; background-color: #2e7131;"></div>	Success	<input type="button" value="Remove"/>
<input type="checkbox"/>	node4	<div style="width: 100%; background-color: #2e7131;"></div>	Success	<input type="button" value="Remove"/>

Some warnings were encountered while performing checks against the 3 registered hosts above [Click here to see the warnings.](#)

Confirm Hosts Window

The Confirm Hosts window enables you to monitor the progress of the Ambari agent installation and registration process. Each node with a successful agent registration displays Success and a green progress bar.

If you decide not to include one or more nodes in the cluster, you may click **Remove** next to that host before proceeding.

Ambari performs configuration checks on each node once an agent has been installed.

If Ambari detects any potential configuration problems they can be viewed by clicking the link [Click here to see the warnings.](#) A window opens with detailed information about potential issues. Some warnings are serious and must be resolved before proceeding while some warnings can be potentially ignored. In each case, a warning must be evaluated for its impact on cluster operation.

When ready to proceed with the installation process, click **Next**.

Assign Slaves and Clients

Add Host Wizard

ADD HOST WIZARD

- [Install Options](#)
- [Confirm Hosts](#)
- Assign Slaves and Clients**
- [Configurations](#)
- [Review](#)
- [Install, Start and Test](#)
- [Summary](#)

Assign Slaves and Clients

Assign slave and client components to hosts you want to run them on. Hosts that are assigned master components are shown with *. "Client" will install HDFS Client, MapReduce2 Client, YARN Client, Tez Client, HCat Client, Hive Client, Pig and ZooKeeper Client.

Host	all none	all none	all none	all none
node2	<input checked="" type="checkbox"/> DataNode	<input type="checkbox"/> NFSGateway	<input checked="" type="checkbox"/> NodeManager	<input checked="" type="checkbox"/> Client
node3	<input checked="" type="checkbox"/> DataNode	<input type="checkbox"/> NFSGateway	<input checked="" type="checkbox"/> NodeManager	<input checked="" type="checkbox"/> Client
node4	<input checked="" type="checkbox"/> DataNode	<input type="checkbox"/> NFSGateway	<input checked="" type="checkbox"/> NodeManager	<input checked="" type="checkbox"/> Client

Show: 25 1 - 3 of 3 ⌂ ⌃ ⌁ ⌂ ⌃ ⌁

[← Back](#) [Next →](#)

Add Host Wizard

Use the Assign Slaves and Clients window to choose where to run service worker components and Hadoop client software.

The **DataNode** is the HDFS service worker component.

An HDFS **NFS Gateway** machine can be used as a method to access HDFS.

The **NodeManager** is the YARN service worker component.

Client represents the Hadoop client software. Client software is used by user and application clients to access cluster services and resources. For example, client software includes the HDFS Shell that is used to access HDFS storage.

Make the required selections and click **Next**.

Choose Configuration Groups

The screenshot shows the 'Add Host Wizard' interface. On the left, a sidebar lists steps: 'Install Options', 'Confirm Hosts', 'Assign Slaves and Clients', 'Configurations' (which is selected and highlighted in dark grey), 'Review', 'Install, Start and Test', and 'Summary'. The main panel is titled 'Configurations' and contains the following text: 'Select the configuration groups to which the added hosts will belong to.' Below this is a table with two columns: 'Service' and 'Configuration Group'. The services listed are HDFS, YARN, MapReduce2, Tez, Hive, and Pia. Each service has a dropdown menu next to it showing the current configuration group: 'HDFS Default', 'YARN Default', 'MapReduce2 Default', 'Tez Default', 'Hive Default', and 'Pia Default'. At the bottom of the panel are 'Back' and 'Next' buttons.

Service	Configuration Group
HDFS	HDFS Default
YARN	YARN Default
MapReduce2	MapReduce2 Default
Tez	Tez Default
Hive	Hive Default
Pia	Pia Default

Choosing Configuration Groups

The configuration settings that should be applied to a new DataNode might vary based on the size and amount of the DataNode's compute, storage, and network resources. For this reason, you are offered the choice of choosing a specific configuration group for each service. Configuration groups are described in another lesson.

Review and Deploy

The screenshot shows the 'Add Host Wizard' interface. On the left, a sidebar lists steps: 'Install Options', 'Confirm Hosts', 'Assign Slaves and Clients', 'Configurations', 'Review' (which is highlighted in dark grey), 'Install, Start and Test', and 'Summary'. The main panel is titled 'Review' and contains the message 'Please review the configuration before installation'. Below this, it shows cluster details: 'Admin Name : admin', 'Cluster Name : horton', and 'Total Hosts : 4 (3 new)'. A section titled 'Repositories:' lists download URLs for various HDP components across different platforms:

- redhat6 (HDP-2.3):
<http://s3.amazonaws.com/dev.hortonworks.com/HDP/centos6/2.x/BUILDS/2.3.0.0-2410>
- redhat6 (HDP-UTILS-1.1.0.20):
<http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/centos6>
- redhat7 (HDP-2.3):
<http://s3.amazonaws.com/dev.hortonworks.com/HDP/centos7/2.x/BUILDS/2.3.0.0-2410>
- redhat7 (HDP-UTILS-1.1.0.20):
<http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/centos6>
- suse11 (HDP-2.3):
<http://s3.amazonaws.com/dev.hortonworks.com/HDP/suse11sp3/2.x/BUILDS/2.3.0.0-2410>
- suse11 (HDP-UTILS-1.1.0.20):
<http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/suse11sp3>

At the bottom of the panel are three buttons: 'Back' (with a left arrow icon), 'Print' (blue button), and 'Deploy' (green button).

Review and Deploy

The Review window enables you to print the configuration and deploy the nodes to the cluster.

Click **Deploy** when ready to install the software.

Install, Start, and Test

Add Host Wizard

Install, Start and Test

Please wait while the selected services are installed and started.

4 % overall

Show: All (3) In Progress (3) Warning (0) Success (0) Fail (0)		
Host	Status	Message
node2	4%	Installing DataNode
node3	4%	Installing DataNode
node4	4%	Installing DataNode

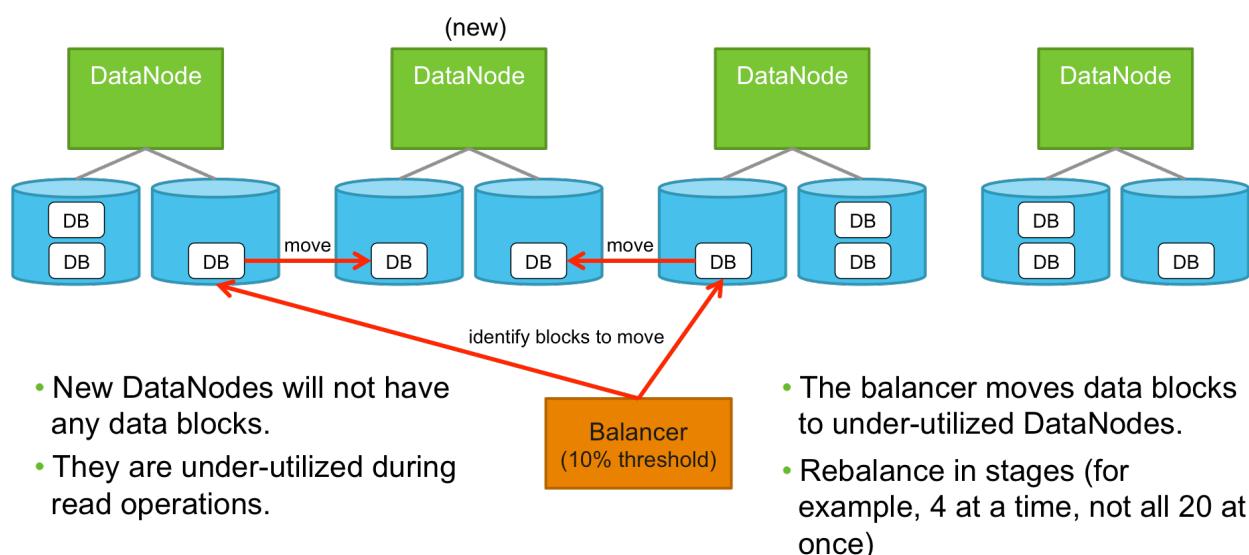
3 of 3 hosts showing - Show All Show: 25 1 - 3 of 3 Next →

Install, Start and Test

The Install, Start and Test window displays the installation progress of each node.

If any node fails to install properly, the error displayed in the Message column is a hypertext link. Click the link to get more information about the error. Once you resolve the issue, you can use Ambari to install the node again.

The HDFS Balancer

*HDFS Balancer*

When a new DataNode is added to a cluster, its will not have data blocks on its disk drives. This DataNode will be severely under-utilized during HDFS read operations. The primary method to move some of the existing data blocks to the new DataNode is to run the HDFS balancer utility.

The HDFS balancer utility moves data blocks from over-utilized to under-utilized DataNodes. The balancer is programmed to run with a default threshold of ten percent. This means that the balancer will move data blocks if the disk usage on any DataNode is different for the overall HDFS usage by plus or minus than ten percent. The default threshold can be overridden when the administrator runs the balancer.

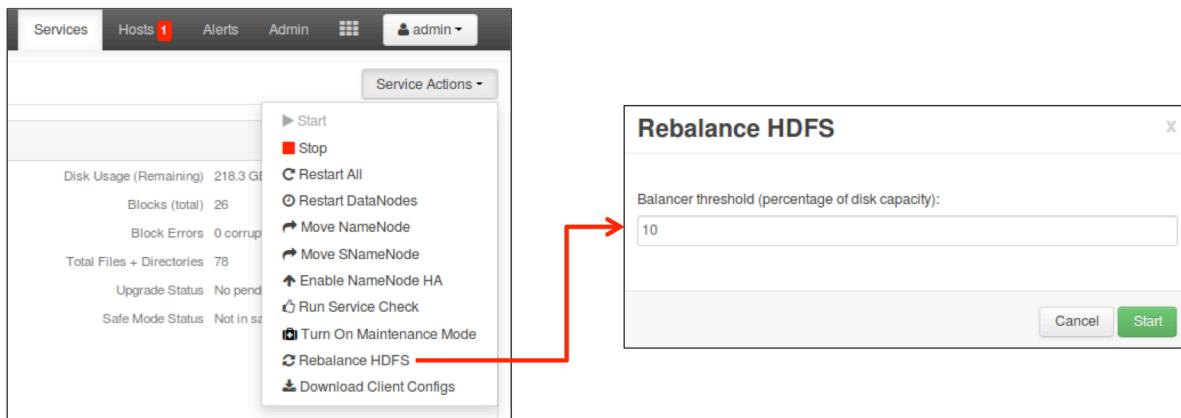
Rebalancing HDFS storage could move a significant amount of data across the network. For this reason, the amount of network bandwidth consumed by the balancer on each DataNode is configurable. The default value is set by the property:

`dfs.datanode.balance.bandwidthPerSec` in the `hdfs-site.xml` file. The default is set to 6,250,000 bytes per second.

Also to preserve network bandwidth for running processes, Hortonworks recommends HDFS rebalancing in groups. For example if adding 20 new nodes, consider rebalancing 4-5 nodes at a time rather than balancing all 20 at once.

Running the Balancer Using Ambari

- Ambari Web UI > Services > HDFS > Service Actions > Rebalance HDFS



Running the Balancer in Ambari Web UI

The balancer can be run by an administrator or operator from the Ambari Web UI or from a command-line prompt. When launching the balancer using Ambari, the Web UI interface prompts the user for the balancer threshold.

The default is ten percent but another value can be chosen.

Running the Balancer Using CLI

- From the command line as the HDFS superuser. (`su - hdfs`)
- Using the default threshold:
`hdfs balancer`

- The balancer will exit when done.
- **Changing the threshold to 5 percent:**
hdfs balancer -threshold 5
- **Display other options:**
hdfs balancer -help

The balancer can also be run from a command-line prompt. It must be run with HDFS superuser privileges.

- To run the balancer using the default settings, type the command `hdfs balancer`.
The balancer will run until complete.
- To run the balancer with a threshold other than the default, type the command `hdfs balancer -threshold 5`.
The balancer will run with a threshold of five percent rather than ten percent.

The balancer does have other, less common options. To view these options, type the command `hdfs balancer -help`.

Monitoring DataNode Balance

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
node1:50010 (172.17.0.2:50010)	1	In Service	93.13 GB	439.97 MB	19.96 GB	72.74 GB	26	439.97 MB (0.46%)	0	2.7.1.2.3.0.0-2525
node2:50010 (172.17.0.3:50010)	2	In Service	93.13 GB	439.99 MB	19.96 GB	72.74 GB	26	439.99 MB (0.46%)	0	2.7.1.2.3.0.0-2525
node3:50010 (172.17.0.4:50010)	1	In Service	93.13 GB	440 MB	19.96 GB	72.74 GB	26	440 MB (0.46%)	0	2.7.1.2.3.0.0-2525

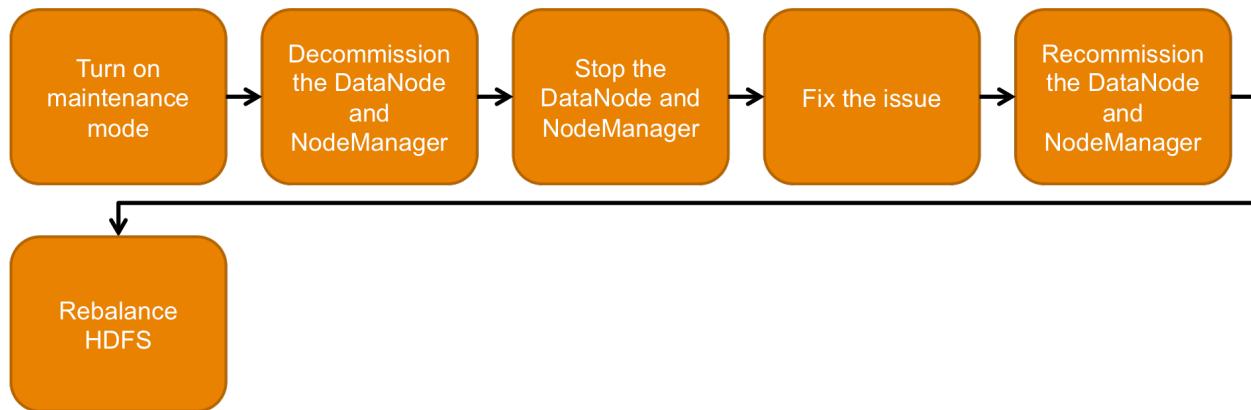
NameNode UI Datanodes Tab

An administrator must be able to view DataNode balance in order to determine whether they should run the balancer. The NameNode UI's Datanodes tab is used to view balance information. The relevant columns are highlighted here.

Similar information is available by becoming the HDFS superuser and generating an HDFS report using `hdfs dfsadmin -report`.

You can also examine the HDFS Heatmaps in the Ambari Web UI. These can visually indicate under or over-utilized DataNodes.

Decommissioning and Re-commissioning a Worker Node



Decommissioning and Recommissioning a Worker Node

Decommissioning and recommissioning is a multi-step process. Worker nodes normally run both a DataNode and a NodeManager, and both are typically commissioned or decommissioned together.

DataNode - NodeManager Termination Differences

With the replication level set to three, HDFS is resilient to individual DataNode failures. However, there is a high chance of data loss when you terminate multiple DataNodes without decommissioning them first. Decommissioning multiple DataNodes should be accomplished on a schedule that permits the replication of the data of blocks that reside on DataNodes being taken out of service. For additional data safety, consider decommissioning on a single DataNode at a time.

Decommissioning a NodeManager is different. If a NodeManager is shut down, the ResourceManager will reschedule the tasks on other NodeManagers in the cluster. However, decommissioning a NodeManager might be required in situations where you want a NodeManager to stop accepting new tasks, or when the tasks take time to execute but you still want to be agile in your cluster management.

Selecting a Worker Node

Name	Cores	RAM	Disk Usage	Load Avg	Versions	Components
node3	4 (4)	7.79GB		1.02	HDP-2.3.0.0-2525	11 Components
node4	4 (4)	7.79GB		1.02	HDP-2.3.0.0-2525	11 Components

The Hosts Tab

Use the **Hosts** page to select the worker node to decommission. In a large cluster, the list of worker nodes could be very long. Use the **Name** text box or the **Filter** button to select a filter to reduce the number of nodes displayed.

Once the node to be decommissioned has been found in the list, click the node's name.

Turning on Maintenance Mode

Host Actions - Turn on Maintenance Mode

Although it is optional, Hortonworks recommends turning on maintenance mode. Maintenance Mode affects a service, component, or host object in the following two ways:

- Maintenance mode suppresses alerts, warnings, and status change indicators generated for the object
- Maintenance mode exempts an object from host-level or service-level bulk operations

You typically turn on Maintenance Mode when performing hardware or software maintenance, changing configuration settings, troubleshooting, decommissioning, or removing cluster nodes.

Decommissioning

Decommissioning is a process that supports removing a worker service component from a cluster. You must decommission the worker running on a host before removing the component or host from service in order to avoid potential loss of data or processing disruption

Decommissioning a DataNode

Decommissioning a DataNode

Decommissioning a DataNode safely replicates the HDFS data to other DataNodes in the cluster.

Ambari updates the `/etc/hadoop/conf/dfs.exclude` file with the hostname of the DataNode when an administrator decommissions a DataNode. This file is defined by the `dfs.hosts.exclude` property in the `hdfs-site.xml` file.

Decommissioning a NodeManager

Adding, Deleting, and Replacing Worker Nodes

The screenshot shows the Ambari Web UI for node3. In the Components section, the DataNode (HDFS) status is set to Decommissioned. The NodeManager status is set to Started. A context menu is open over the NodeManager component, with 'Decommission' highlighted. The Host Metrics section contains two graphs: CPU Usage and Disk Usage. The CPU Usage graph shows a single bar reaching 20% with a value of 3. The Disk Usage graph shows two bars: one at 186.2 GB and another at 372.5 GB, both with a value of 2.7 GB.

Decommissioning and NodeManager

Decommissioning a NodeManager stops a NodeManager from accepting new job requests from the ResourceManager.

Ambari updates the `/etc/hadoop/conf/yarn.exclude` file with the hostname of the NodeManager when an administrator decommissions a NodeManager. This file is defined by the `yarn.resourcemanager.nodes.exclude-path` property in the `yarn-site.xml` file.

Monitoring Decommissioning

- Ambari Web UI

The screenshot shows the Ambari Web UI for node3. In the Components section, the DataNode (HDFS) status is set to Decommissioned. The Metrics Monitor and NodeManager statuses are set to Started. The rest of the interface is identical to the previous screenshot.

- NameNode UI > Datanodes tab (<http://<NameNode>:50070>)

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
node1:50010 (172.17.0.2:50010)	0	In Service	93.13 GB	439.97 MB	19.97 GB	72.73 GB	26	439.97 MB (0.46%)	0	2.7.1.2.3.0.0-2525
node2:50010 (172.17.0.3:50010)	1	In Service	93.13 GB	439.99 MB	19.97 GB	72.73 GB	26	439.99 MB (0.46%)	0	2.7.1.2.3.0.0-2525
node3:50010 (172.17.0.4:50010)	0	Decommission In Progress	93.13 GB	439.99 MB	19.97 GB	72.73 GB	26	440 MB (0.46%)	0	2.7.1.2.3.0.0-2525

Decommissioned when finished

Monitoring Decommissioning

DataNode status can be live or dead. A live DataNode can be decommissioning or decommissioned. A dead DataNode can also be decommissioned. These states are reported using different terms in different management interfaces.

The primary ways to view DataNode status are to use:

- Ambari Web UI's **Hosts** page
- NameNode UI's **DataNodes** tab

Performing Maintenance

Stopping All Components

Once all cluster components are shut down, you can perform scheduled maintenance activities.

The screenshot shows the Ambari UI for node3. The 'Host Actions' dropdown menu is open, with 'Stop All Components' highlighted. Other options include Start All Components, Restart All Components, Turn On Maintenance Mode, Delete Host, Set Rack, and Download Client Configs.

Stopping All Components

Starting All Components

Once you have fixed the hardware, upgraded the hardware or software, or addressed whatever the issue was, you can restart the node's service components.

The screenshot shows the Ambari UI for node3. The 'Host Actions' dropdown menu is open, with 'Start All Components' highlighted. Other options include Stop All Components, Restart All Components, Turn On Maintenance Mode, Delete Host, Set Rack, and Download Client Configs.

Starting All Components

Re-Commissioning

Re-commissioning is a process that enables a worker service component in a cluster.

Re-commissioning a NodeManager

Recommissioning the NodeManager enables the ResourceManager to assign jobs to the worker node.

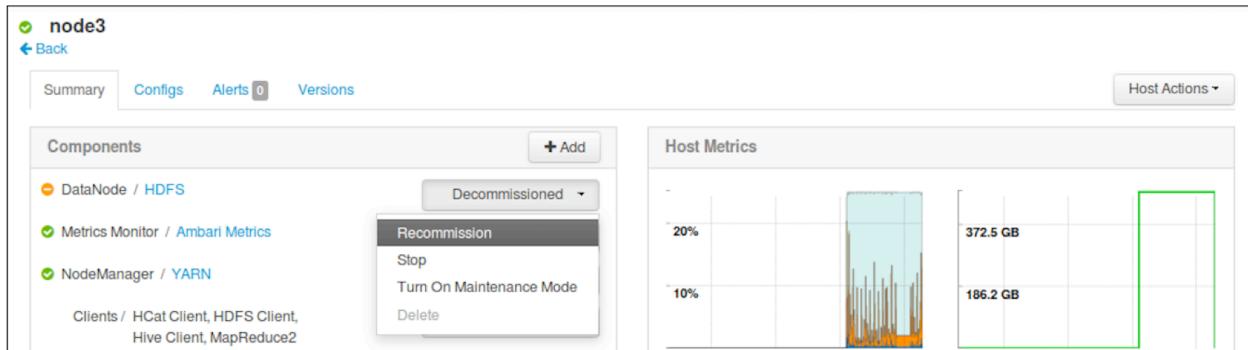
The screenshot shows the Ambari UI for node3. The 'Host Actions' dropdown menu is open, with 'Recommission' highlighted under the NodeManager section. Other options include Start, Turn On Maintenance Mode, and Delete. The 'Components' table shows the NodeManager status as Decommissioned. The 'Host Metrics' section displays CPU Usage, Disk Usage, and Memory Usage graphs.

Re-commissioning a Node Manager

When an administrator recommissions a NodeManager, Ambari updates the /etc/hadoop/conf/yarn.exclude file to remove the hostname of the NodeManager. This file is defined by the yarn.resourcemanager.nodes.exclude-path property in the yarn-site.xml file.

Re-commissioning a DataNode

Recommissioning the DataNode enables the NameNode to access storage on the worker node.



Re-commissioning a DataNode

When an administrator recommissions a DataNode, Ambari updates the /etc/hadoop/conf/dfs.exclude file to remove the hostname of the DataNode. This file is defined by the dfs.hosts.exclude property in the hdfs-site.xml file.

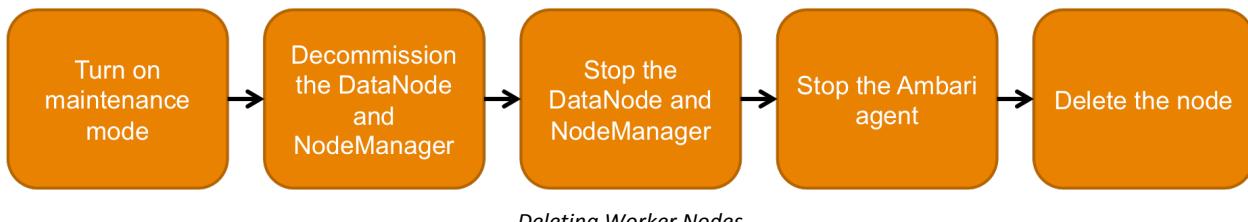
Rebalancing HDFS

Once a DataNode has been recommissioned, run the balancer to ensure that it is not under-utilized during HDFS read operations. The balancer was described earlier in this lesson.

The balancer can be launched from the Ambari Web UI or from a command-line prompt.

Use the NameNode UI or an `hdfs dfsadmin -report` to verify DataNode balance.

Deleting Worker Nodes



Deleting Worker Nodes

Deleting a worker node is a multi-step process even though Ambari automates much of the actual work. The general steps are illustrated in the diagram above.

The Ambari Web UI includes different ways to accomplish this task using different drop-down menus in various UI locations.

Selecting a Node to Delete

Adding, Deleting, and Replacing Worker Nodes

The screenshot shows the Ambari interface for managing hosts. The 'Hosts' tab is selected, indicated by a red box. The main area displays a table of worker nodes with columns for Name, IP Address, Rack, Cores, RAM, Usage, Load Avg, Versions, and Components. A callout bubble points to the 'Name' column header and the 'Any' search input field, suggesting to use the name text box to filter nodes. Another callout bubble points to the 'Filter' button at the top right of the table. A third callout bubble points to the 'node1' row, which has a red warning icon and a red number '3' in a box, with the text 'Click the node to remove'.

Selecting a Worker Node to Delete

Select the worker node to delete. In a large cluster, the list of worker nodes could be very long. Use the **Filter** button or the **Name** text box to select a filter to reduce the number of nodes displayed.

Once the node to be deleted has been found in the list, click the node name.

Turning On Maintenance Mode

The screenshot shows the Ambari interface for managing a specific host. The host is named 'node4'. The 'Host Actions' dropdown menu is open, showing options like Start All Components, Stop All Components, Restart All Components, Turn On Maintenance Mode (which is highlighted), Delete Host, Set Rack, and Download Client Configs.

Turning on Maintenance Mode

Although it is optional, Hortonworks recommends turning on maintenance mode. Maintenance Mode affects a service, component, or host object in the following two ways:

- Maintenance mode suppresses alerts, warnings, and status change indicators generated for the object
- Maintenance mode exempts an object from host-level or service-level bulk operations

You typically turn on Maintenance Mode when performing hardware or software maintenance, changing configuration settings, troubleshooting, decommissioning, or removing cluster nodes.

Decommissioning

Decommissioning is a process that supports removing a worker service component from a cluster. You must decommission the worker running on a host before removing the component or host from service in order to avoid potential loss of data or disruption on processing.

Decommissioning the DataNode

The screenshot shows the Ambari UI for a host named 'node4'. In the 'Components' section, the 'DataNode / HDFS' component is listed as 'Started'. A dropdown menu is open over this component, showing options: 'Decommission', 'Restart', 'Stop', 'Turn On Maintenance Mode', and 'Delete'. The 'Decommission' option is highlighted. In the 'Host Metrics' section, all metrics are shown as 'No Data Available'.

Decommissioning the DataNode

Decommissioning a DataNode safely replicates the HDFS data to other DataNodes in the cluster.

Decommissioning the NodeManager

The screenshot shows the Ambari UI for a host named 'node4'. In the 'Components' section, the 'DataNode / HDFS' component is listed as 'Decommissioned'. Other components like 'Metrics Monitor / Ambari Metrics' and 'NodeManager / YARN' are listed as 'Started'. A dropdown menu is open over the 'DataNode / HDFS' component, showing options: 'Decommission', 'Restart', 'Stop', 'Turn On Maintenance Mode', and 'Delete'. The 'Decommission' option is highlighted. In the 'Host Metrics' section, all metrics are shown as 'No Data Available'.

Decommissioning the NodeManager

Decommissioning a NodeManager stops a NodeManager from accepting new job requests from the ResourceManager.

Stopping All Components

The screenshot shows the Ambari UI for node4. The 'Host Actions' dropdown menu is open, with 'Stop All Components' highlighted in red. Other options include 'Start All Components', 'Restart All Components', 'Turn Off Maintenance Mode', 'Delete Host', 'Set Rack', and 'Download Client Configs'.

Stopping the DataNode, the NodeManager and Ambari Metrics

Stop any components running on the node. If you do not, Ambari displays warning messages when you attempt to delete the node.

Stopping the Ambari Agent

```
[root@node4 scripts]# ambari-agent stop
Verifying Python version compatibility...
Using python /usr/bin/python2.6
Found ambari-agent PID: 17222
Stopping ambari-agent
Removing PID file at /var/run/ambari-agent/ambari-agent.pid
ambari-agent successfully stopped
[root@node4 scripts]#
```

ambari-agent stop Command

Stop the Ambari agent running on the node. If you do not, Ambari displays warning messages when you attempt to delete the node. Use the command `ambari-agent stop` to shut down the Ambari agent.

Deleting the Node

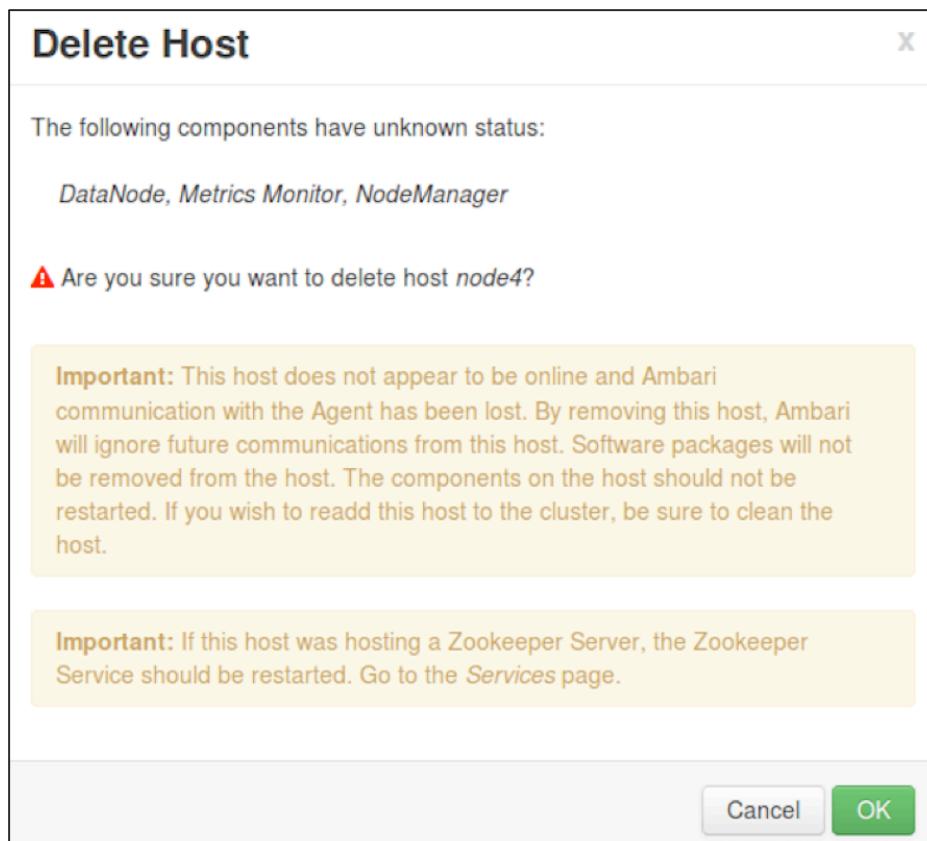
The screenshot shows the Ambari UI for node4. The 'Host Actions' dropdown menu is open, with 'Delete Host' highlighted in red. Other options include 'Start All Components', 'Stop All Components', 'Restart All Components', 'Turn Off Maintenance Mode', 'Set Rack', and 'Download Client Configs'.

Deleting a Worker Node

The final step is to actually delete the node. This removes the node from the Ambari database. Ambari will no longer expect to manage this host.

Confirm the Deletion

When you delete a host, Ambari displays this important information.



Confirming the Worker Node Deletion

Read the information and click **OK** to confirm the deletion.

Manual Decommissioning and Recommissioning

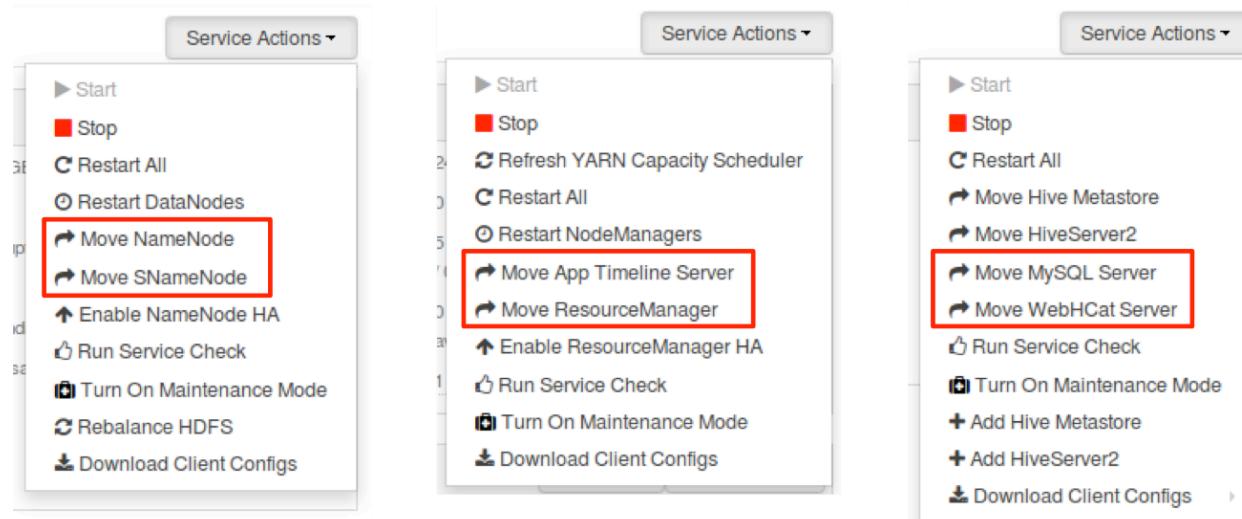
DataNodes and NodeManagers can be manually decommissioned and recommissioned using command-line commands.

- **To decommission nodes manually:**
Add the worker node's hostname to the `dfs.exclude` and `yarn.exclude` files.
Then run the `hdfs dfsadmin -refreshNodes` and `yarn rmadmin -refreshNodes` commands.
- **To recommission nodes manually:**
Remove the worker node's hostname from the `dfs.exclude` and `yarn.exclude` files.
Then run the `hdfs dfsadmin -refreshNodes` and `yarn rmadmin -refreshNodes` commands again.

IMPORTANT

Manually editing configuration files is not compatible with Ambari administration. Manually modified configuration files are overwritten by the information in the Ambari database when the HDFS or YARN services are restarted. Manual decommissioning and recommissioning is not correctly recognized by Ambari.

Moving a Master Component



Examples of Moving Master Components

The Ambari Web UI can be used to move many of the master service components. The ability to move a master service component creates an opportunity to perform hardware or software maintenance on existing nodes with limited cluster downtime.

To enquire whether Ambari can move a specific master component:

- Browse to the **Services** page
- Click the **Service Actions** menu button

Knowledge Check Questions

- 1) List three reasons to add or remove cluster nodes.
- 2) Why are Ambari configuration groups beneficial when growing a cluster?
- 3) What does the HDFS balancer do?
- 4) What is the advantage of turning on maintenance mode when decommissioning or deleting worker nodes?
- 5) Which interfaces can be used to check HDFS balance?
- 6) What is the potential problem with manually decommissioning and recommissioning worker nodes?
- 7) True or false? Ambari Web UI can move some of the master server components.

Knowledge Check Answers

- 1) List three reasons to add or remove cluster nodes.

***Increased cluster demand
failing hardware
obsolete hardware
software upgrades
hardware upgrades***

- 2) Why are Ambari configuration groups beneficial when growing a cluster?

Because Ambari configuration groups are able to accommodate the fact that different sets of cluster nodes might require differing configuration settings based on their different hardware configurations.

- 3) What does the HDFS balancer do?

The HDFS balancer moves data blocks from over-utilized DataNodes to under-utilized DataNodes.

- 4) What is the advantage of turning on maintenance mode when decommissioning or deleting worker nodes?

Maintenance mode suppresses alerts, warnings, and status change indicators generated for the object and exempts an object from host-level or service-level bulk operations.

- 5) Which interfaces can be used to check HDFS balance?

The NameNode UI and the command-line hdfs dfsadmin -report utility.

- 6) What is the potential problem with manually decommissioning and recommissioning worker nodes?

It is not compatible with Ambari administration. Ambari does not correctly recognize the status change.

- 7) True or false? Ambari Web UI can move some of the master server components.

True

Summary

- The Ambari Web UI includes a wizard that is used to add new nodes to a cluster.
- Use Ambari configuration groups and YARN node labels to accommodate worker node configuration differences.
- Decommissioning safely replicates the HDFS data to other DataNodes in the cluster.
- Decommissioning stops a NodeManager from accepting new job requests.
- The HDFS balancer utility moves data blocks from over-utilized to under-utilized DataNodes, and should be run after recommissioning or adding a new DataNode.
- The NameNode UI and the `hdfs dfsadmin -report` utility are used to determine storage utilization balance among DataNodes.
- Deleting a node does not remove its cluster software.
- The Ambari Web UI can be used to move many of the master service components.

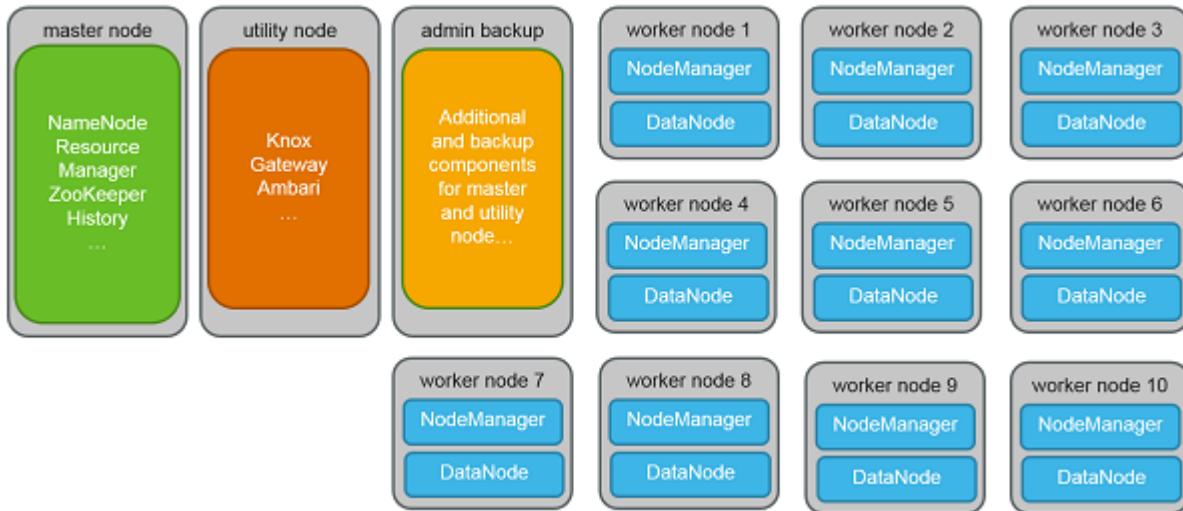
The YARN Capacity Scheduler

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Summarize the purpose and operation of the YARN capacity scheduler
- ✓ Configure YARN capacity scheduler queues
- ✓ Control access to YARN queues
- ✓ Monitor YARN queues
- ✓ Start and stop YARN queues

Capacity Scheduler Operation

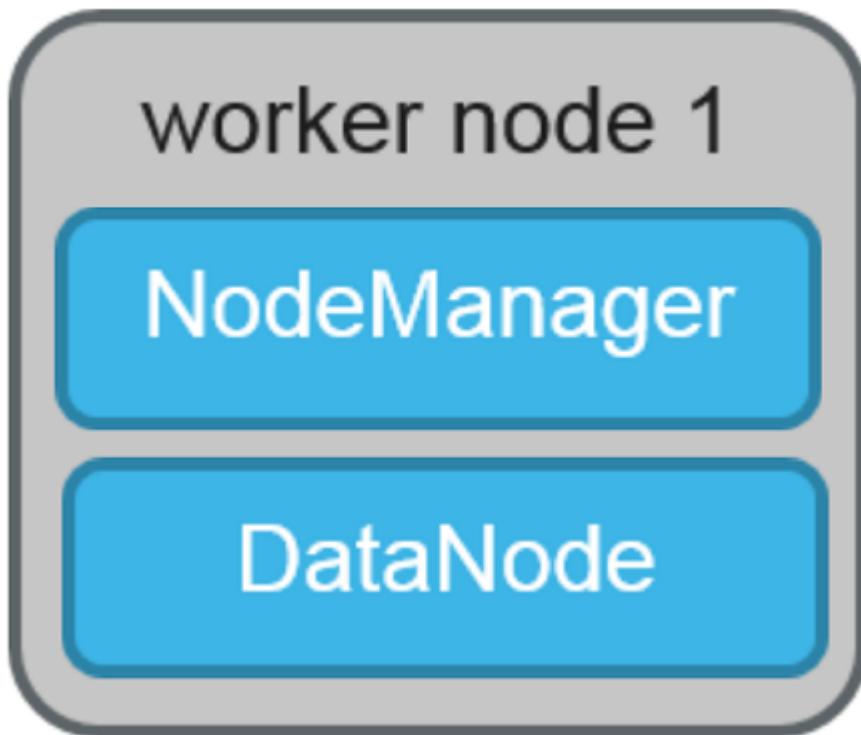


The YARN capacity scheduler manages allocation of resources to applications.

Scenario

To understand the importance of the YARN capacity scheduler, it helps to create a scenario. The scenario examines capacity scheduling in the context of an HDP cluster that contains 10 worker nodes, which have capacity that can be provided to YARN applications.

Worker Node Profile



For simplicity sake, assume that each worker node is identical and each one has the same amount of resources available for YARN applications. In the scenario, this will mean that each worker node has 128 GB of RAM available for YARN applications to use. CPU resources are not accounted for by default.

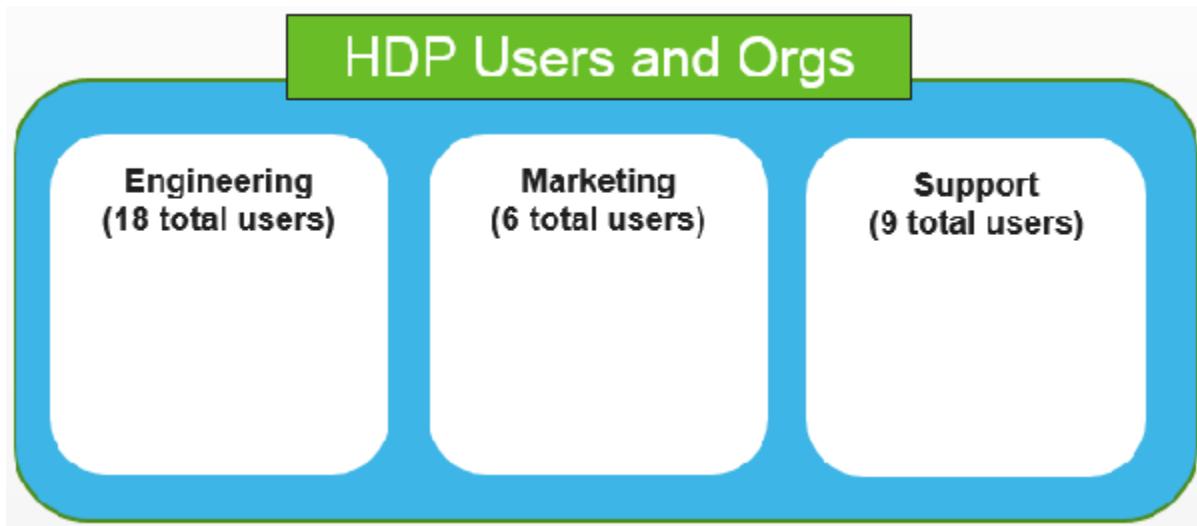
CPU scheduling is only available on Linux-based HDP clusters with cgroups enabled, so will be covered in a different course in more detail.

Total Resources Available

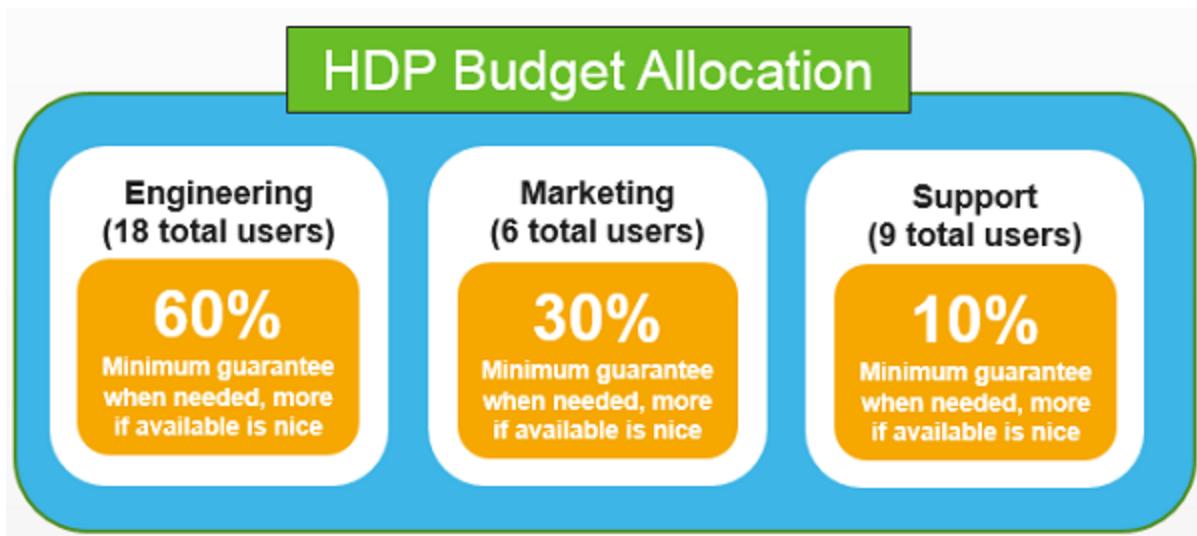
Aggregate pool of
resources

1,280 GB RAM

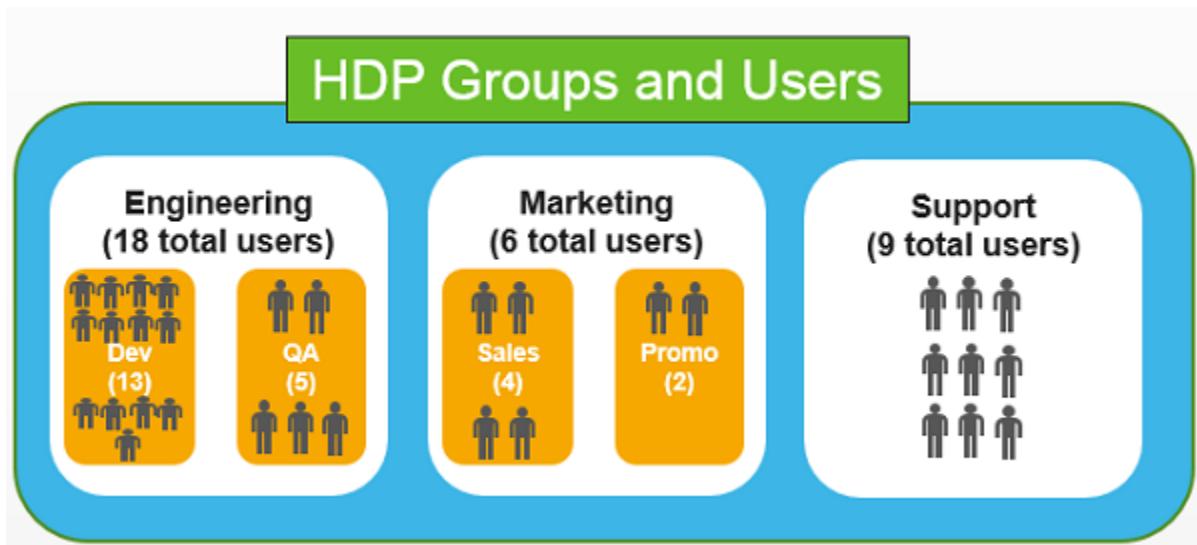
If there are 10 worker nodes with 128 GB of RAM each, that provides 1,280 GB of RAM that can be allocated to containers for running YARN applications. From a global cluster perspective, this is simply seen as a big pool of resources that *in most cases* can be divided up across all applications without regard to where the available resources might be at any given time.

Organization Structure

In this scenario, this HDP cluster is being set up for use by three different organizations within the company: Engineering, Marketing, and Support. Engineering plans to have 18 active users in the cluster, Marketing will have 6, and Support will have 9.

Service Level Agreements

When it comes to SLAs (Service Level Agreements), the cluster build and maintenance is being financed 60% by Engineering, 30% by Marketing, and 10% by Support. Thus, at a minimum, each group expects that percentage of available cluster resources to be granted to it upon demand. However, the company has agreed that all unused resources should be made available to other organizations until they are needed by that organization.

Groups and Users

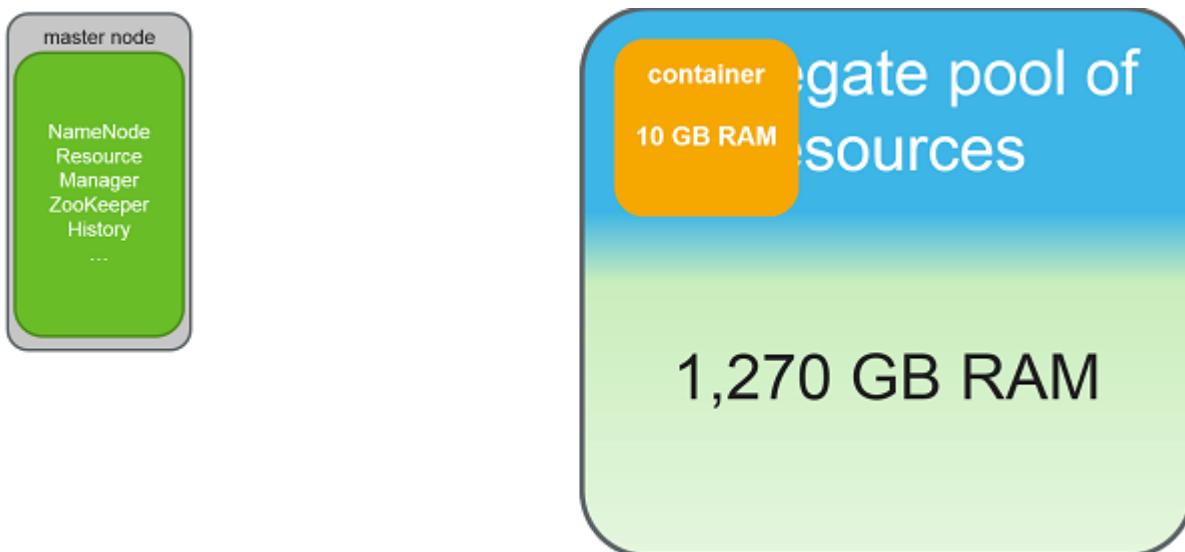
The Engineering organization is divided into two groups of users – Development with 13 users, and QA with 5. The QA group has a smaller number of users, but the resources granted to the Engineering group should be split evenly among the two. The Marketing organization also has two groups – Sales with 4 users, and Promotions with 2. At this time, the Marketing organization does not see the need to divide resource allocations between the two groups, but they reserve the right to request differently in the future. The Support organization will grant 9 users access to its HDP cluster resources, however none of those 9 individuals belong to a single defined group. They are spread across multiple functions, and so the users will each need to be managed manually rather than by group identification.

Resource Allocation

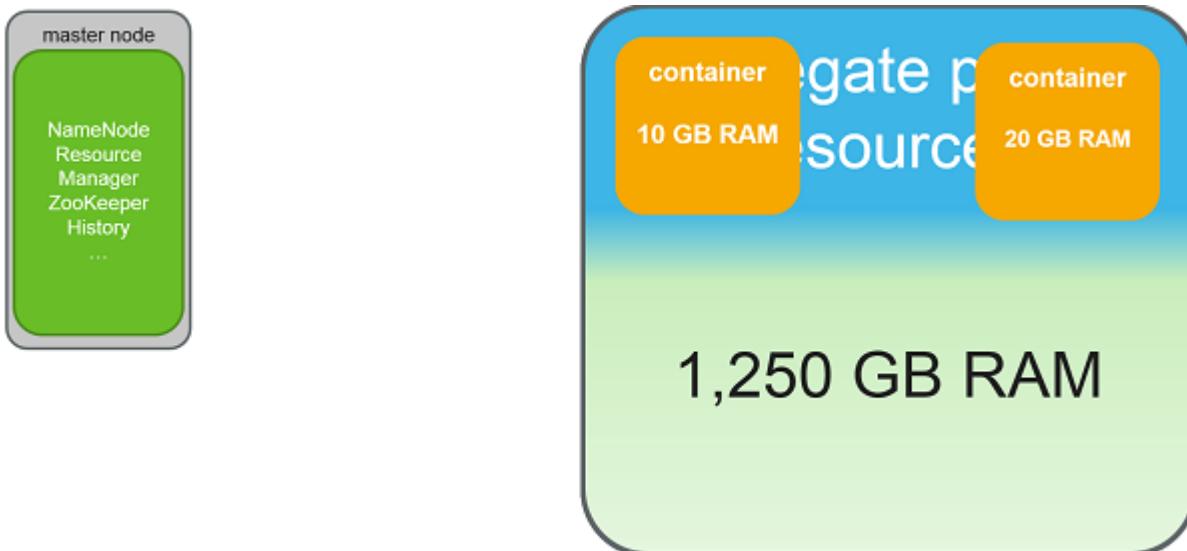
Resources are granted to applications via containers. By default, a container is just a memory reservation.

Default Behavior

In this example, a request for a 10 GB container has been issued. Once that container has been provisioned, the resources available for additional containers is dropped appropriately – in this example, from 1,280 GB of RAM down to 1,270 GB.



If another 20 GB container is provisioned, the allocatable RAM drops to 1,250 GB.

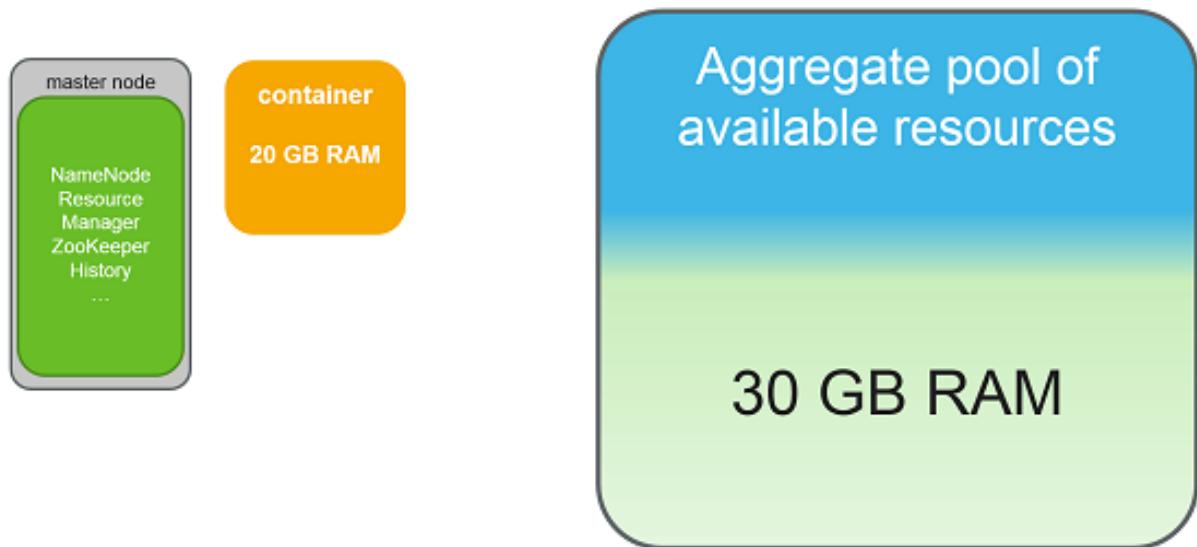


New containers can be allocated until all available memory has been claimed by container provisioning.

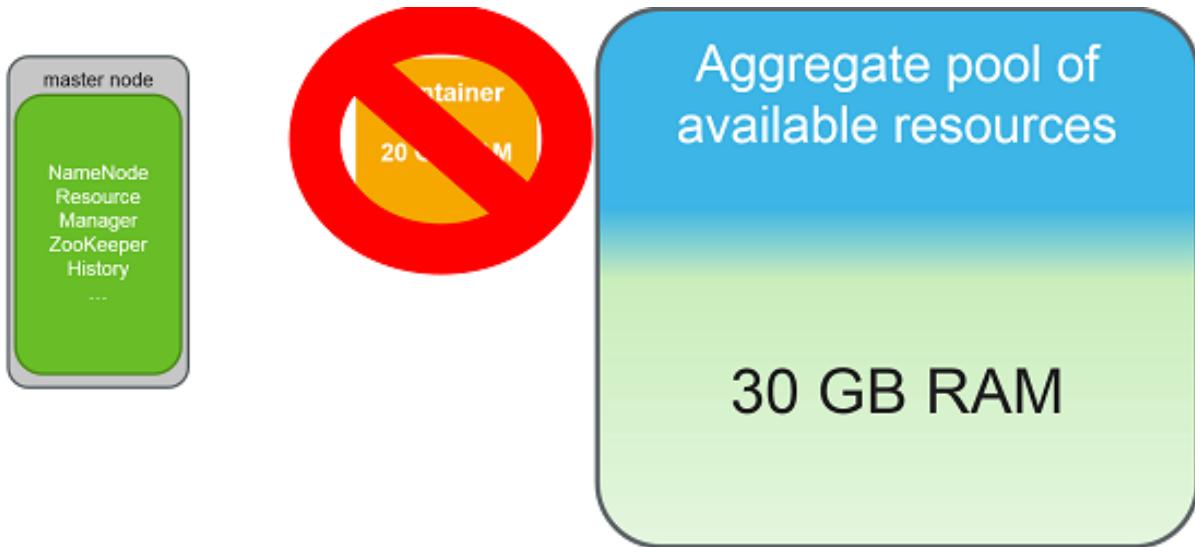
Capacity Scheduler Memory Configuration

Capacity scheduler behavior can be modified under **Services > YARN > Configs > Settings**. The node settings control how much memory is to be allocated to YARN application containers per server. The container settings control the maximum and minimum container sizes that will be granted to YARN applications.

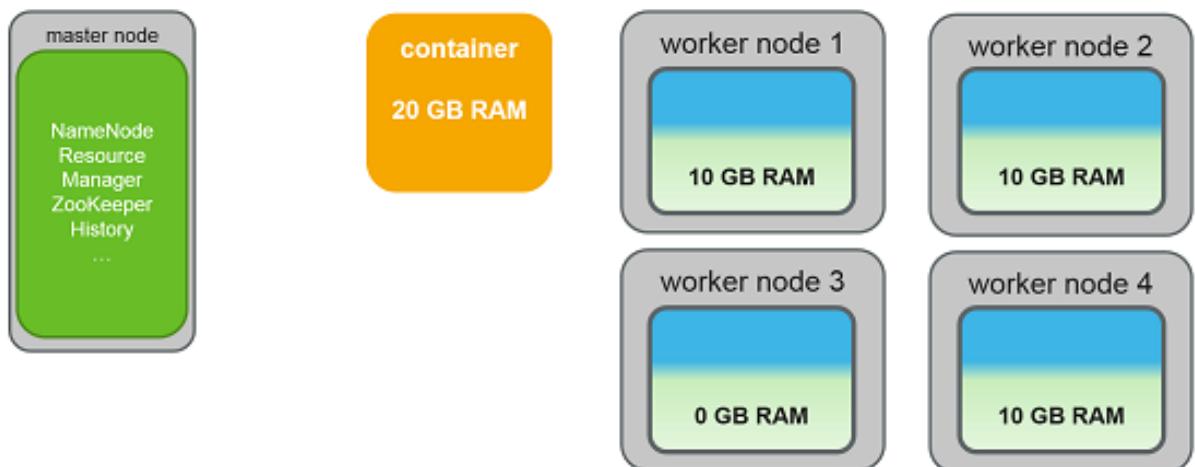
Application Reservations



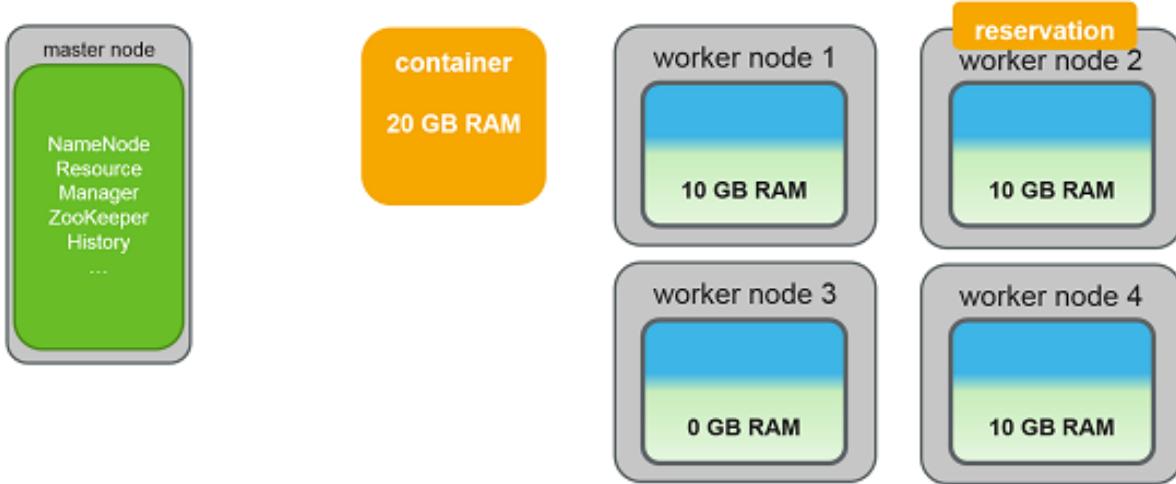
In some cases you may run into a situation in which the cluster shows enough free resources are available in order to launch a container, but the container gets stuck in a wait state.



This can happen when aggregate resources are available to meet the container request needs, but there is no single node in the cluster that has enough unreserved resources, so the container cannot be scheduled.

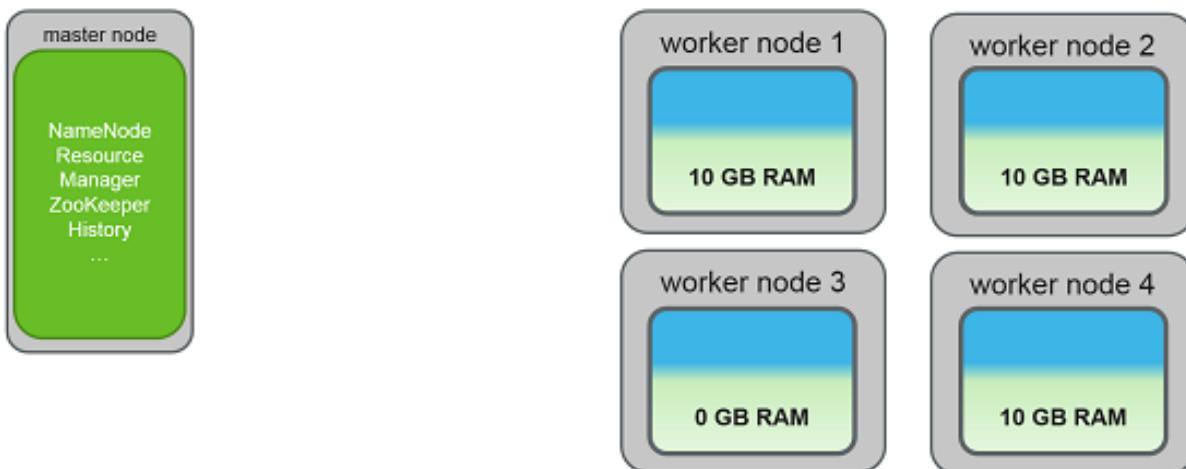


When this happens, the YARN capacity scheduler picks one of the nodes in the cluster and places an application reservation on it.



A node with an application reservation will not be allowed to provision any containers until the oversize container has been provisioned – either on that node, or perhaps on another node in the cluster that happened to have the space available first.

This ensures that oversize containers are not caught in a perpetual pending state as smaller, easier to grant container requests are granted across the cluster.



The YARN capacity scheduler will only place a single application reservation on a single node at one time. Thus, if you have 20 nodes in your cluster, no more than 20 application reservations can be made.

To view reservation status, open a Web browser to <http://<ResourceManager>:8088/cluster/scheduler> or under **Services > YARN** go to **Quick Links > ResourceManager UI** and click the **Scheduler** link.

The YARN Capacity Scheduler

The screenshot shows the Hadoop YARN Capacity Scheduler web interface. At the top, it displays "NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING Applications". The left sidebar includes links for Cluster (About, Nodes, Node Labels, Applications), New (NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, FAILED, KILLED), Scheduler, and Tools. The main content area has sections for Cluster Metrics (with zero values for various metrics like Apps Submitted, Apps Pending, etc.) and Scheduler Metrics (Scheduler Type: Capacity Scheduler, Scheduling Resource Type: [MEMORY], Minimum Allocation: <memory:512, vCores:1>, Maximum Allocation: <memory:4296, vCores:3>). Below these are Application Queues and Aggregate scheduler counts tables.

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Progress	Tracking UI	Blacklisted Nodes
No data available in table												

Aggregate scheduler counts			
Total Container Allocations(count)	Total Container Releases(count)	Total Fulfilled Reservations(count)	Total Container Preemptions(count)
0	0	0	0
Last scheduler run			
Time	Allocations(count - resources)	Reservations(count - resources)	Releases(count - resources)
Sun Jul 19 21:16:16 -0500 2015	0 - <memory:0, vCores:0>	0 - <memory:0, vCores:0>	0 - <memory:0, vCores:0>
Last Preemption			
Time	Container Id	Node Id	Queue
N/A		N/A	
Last Reservation			
Time	Container Id	Node Id	Queue
N/A		N/A	
Last Allocation			
Time	Container Id	Node Id	Queue
N/A		N/A	
Last Release			
Time	Container Id	Node Id	Queue
N/A		N/A	

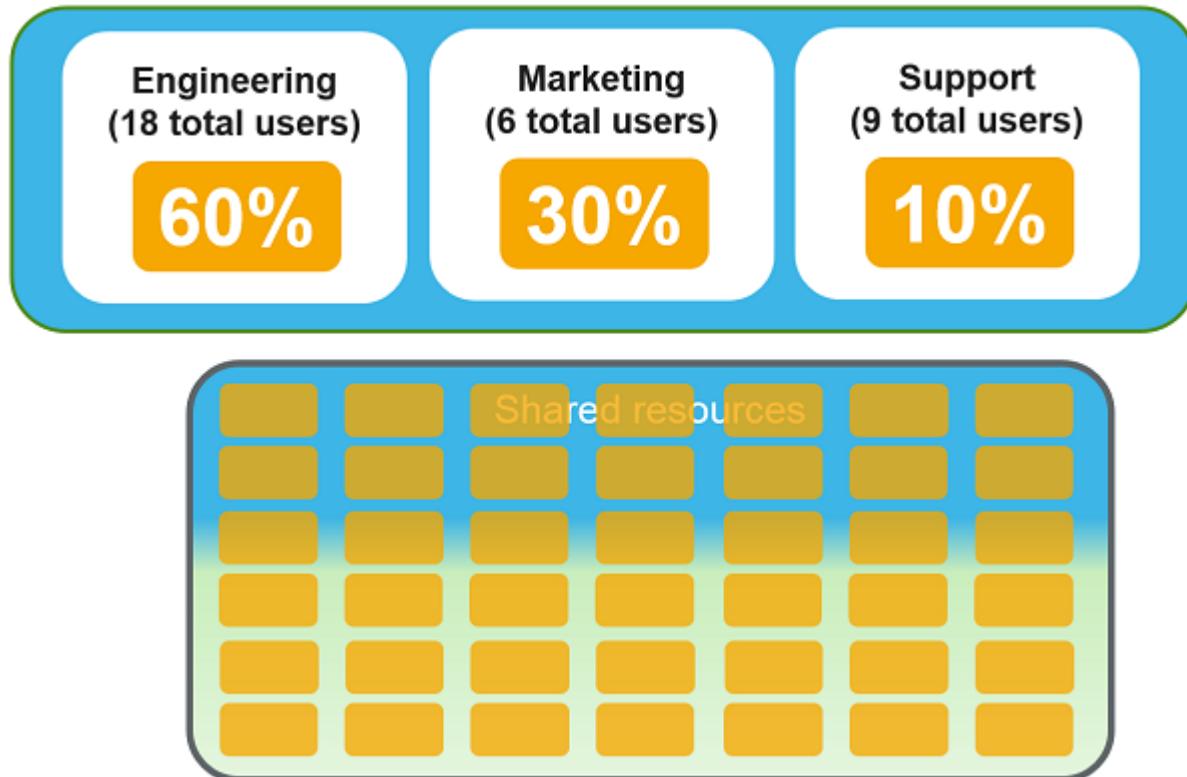
Scroll to the bottom of the resulting screen to get information on total historical reservation activity and most updated reservation, reserved container allocation, and reservation release information.

Aggregate scheduler counts			
Total Container Allocations(count)	Total Container Releases(count)	Total Fulfilled Reservations(count)	Total Container Preemptions(count)
0	0	0	0
Last scheduler run			
Time	Allocations(count - resources)	Reservations(count - resources)	Releases(count - resources)
Sun Jul 19 21:16:16 -0500 2015	0 - <memory:0, vCores:0>	0 - <memory:0, vCores:0>	0 - <memory:0, vCores:0>
Last Preemption			
Time	Container Id	Node Id	Queue
N/A		N/A	
Last Reservation			
Time	Container Id	Node Id	Queue
N/A		N/A	
Last Allocation			
Time	Container Id	Node Id	Queue
N/A		N/A	
Last Release			
Time	Container Id	Node Id	Queue
N/A		N/A	

Queues

Queues are the fundamental unit of resource scheduling.

Potential SLA Problem



YARN applications are scheduled on a First In, First Out (FIFO) basis. This means that without prior configuration, a single group can use cluster resources at will as long as they get their requests in first. In this example, Support applications have taken up 100% of cluster resources, even though they are only paying for 10% of the cluster costs. If a user in Engineering or Marketing wants to run an application, they must wait for resources to free up. Thus, the SLAs for Engineering and Marketing are not being met.

Examining Queues

By default, all jobs are assigned to queues. All cluster resources available to YARN applications are part of a generic “root” queue, and by default a single queue – named “default” is assigned 100% of the resources available through root. If SLAs or other business requirements demand it, the resources can be divided and controlled using queue limits, user and group limits, and a variety of other mechanisms to specifically control applications and users in the cluster.

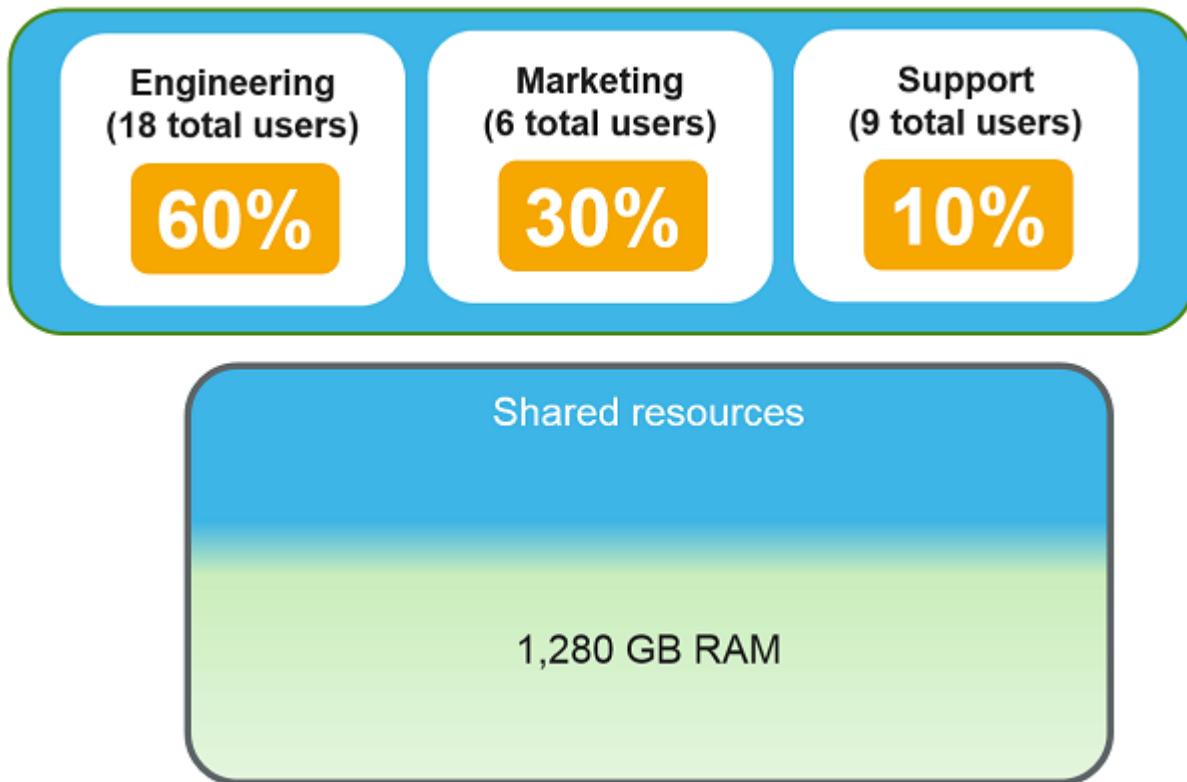
The YARN Capacity Scheduler

View Queue Status

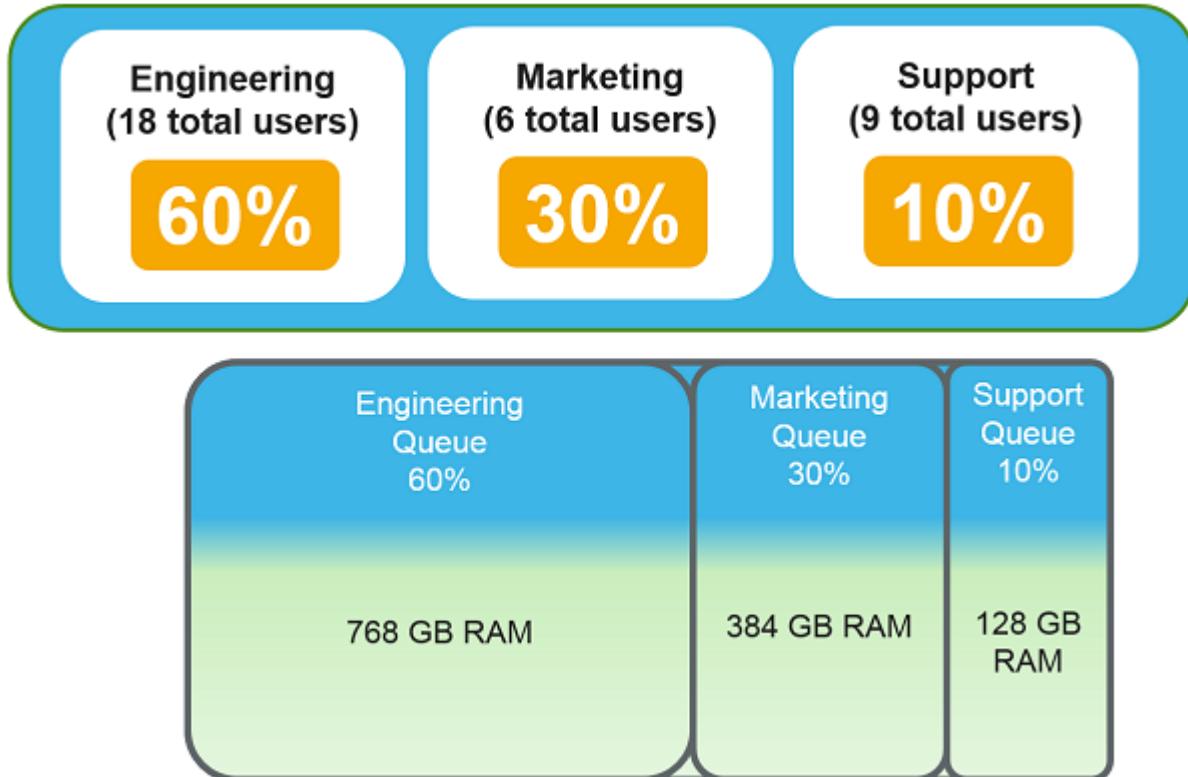
The screenshot shows the Hadoop YARN Capacity Scheduler UI. At the top, it displays "NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING Applications". The left sidebar includes links for Cluster, Applications (Active, Nodes, Node Labels), Scheduler (New, Saving, Submitted, Accepted, Running, Pending, Failed, Killed), and Tools. The main area has sections for Cluster Metrics (with zero values) and Scheduler Metrics (Scheduler Type: Capacity Scheduler, Scheduling Resource Type: [MEMORY], Minimum Allocation: <memory 512, vCores 1>, Maximum Allocation: <memory 4096, vCores 3>). It also shows Application Queues with a legend for Capacity (green bar), Used (orange bar), and Used (over capacity) (yellow bar). A table below lists application details like ID, User, Name, Application Type, Queue, StartTime, FinishTime, State, FinalStatus, Running Containers, Progress, Tracking UI, and Blacklisted Nodes. The table shows "No data available in table". Below this is an "Aggregate scheduler counts" section with tables for Total Container Allocations, Total Container Releases, Total Fulfilled Reservations, and Total Container Preemptions. The last scheduler run table shows a single entry for Sun Jul 19 21:18:16 -0500 2015.

To view current YARN applications, open a Web browser to <http://<ResourceManager>:8088/cluster/scheduler> or under **Services > YARN** go to **Quick Links > ResourceManager UI** and click on the **Scheduler** link. This presents a dashboard for YARN applications and resource usage, status, containers, queues, and other relevant information regarding application performance in the cluster.

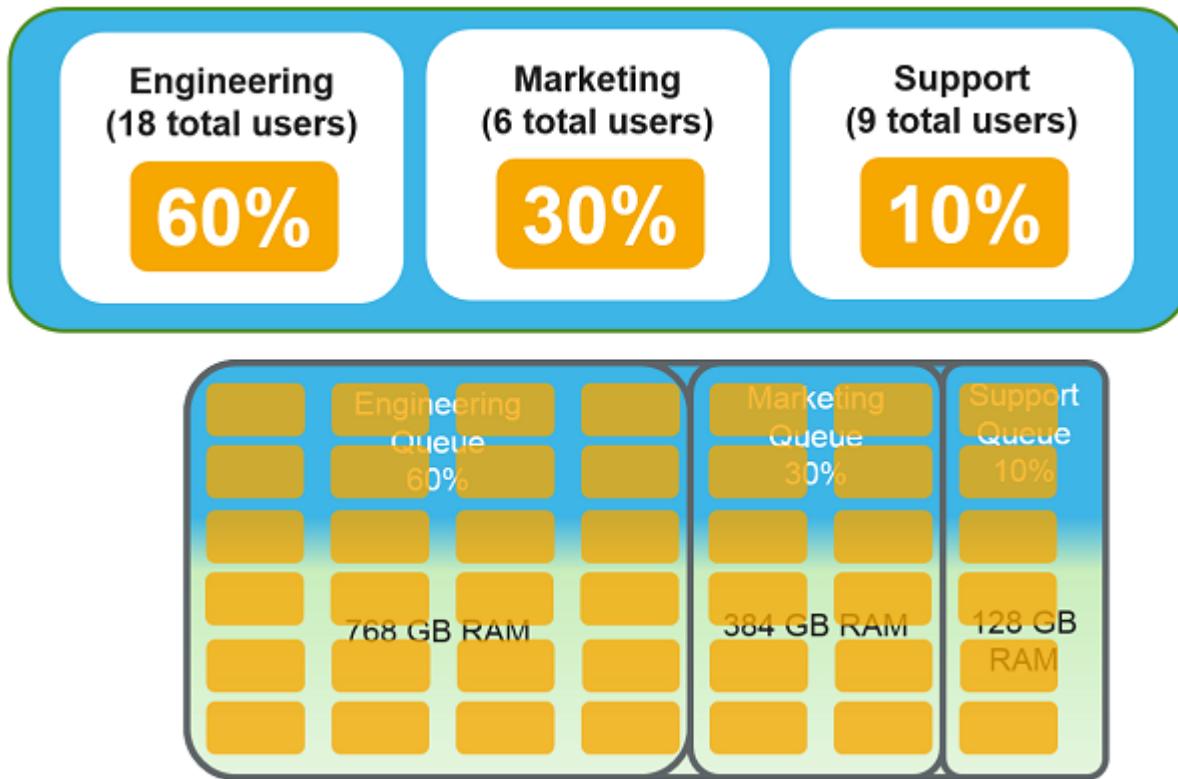
Limits and Elasticity



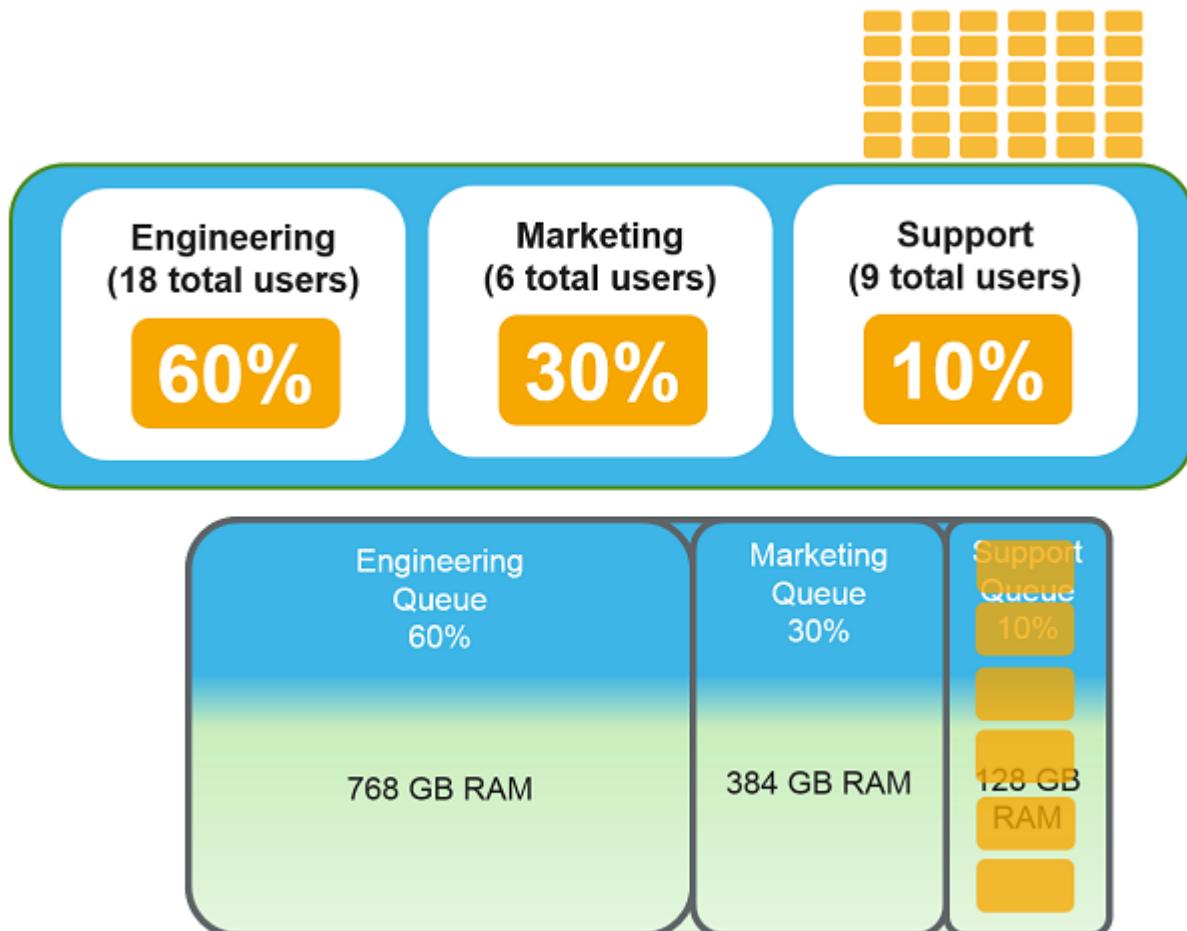
Queues can be configured so that they protect SLAs while enabling full cluster usage when contention is not an issue.



For example, assume the Support group is the only user on the cluster at the moment. Queues can be configured so that Support can still use up to 100% of the cluster if no one else needs it at the moment, even though they are only guaranteed 10% of the cluster resources.

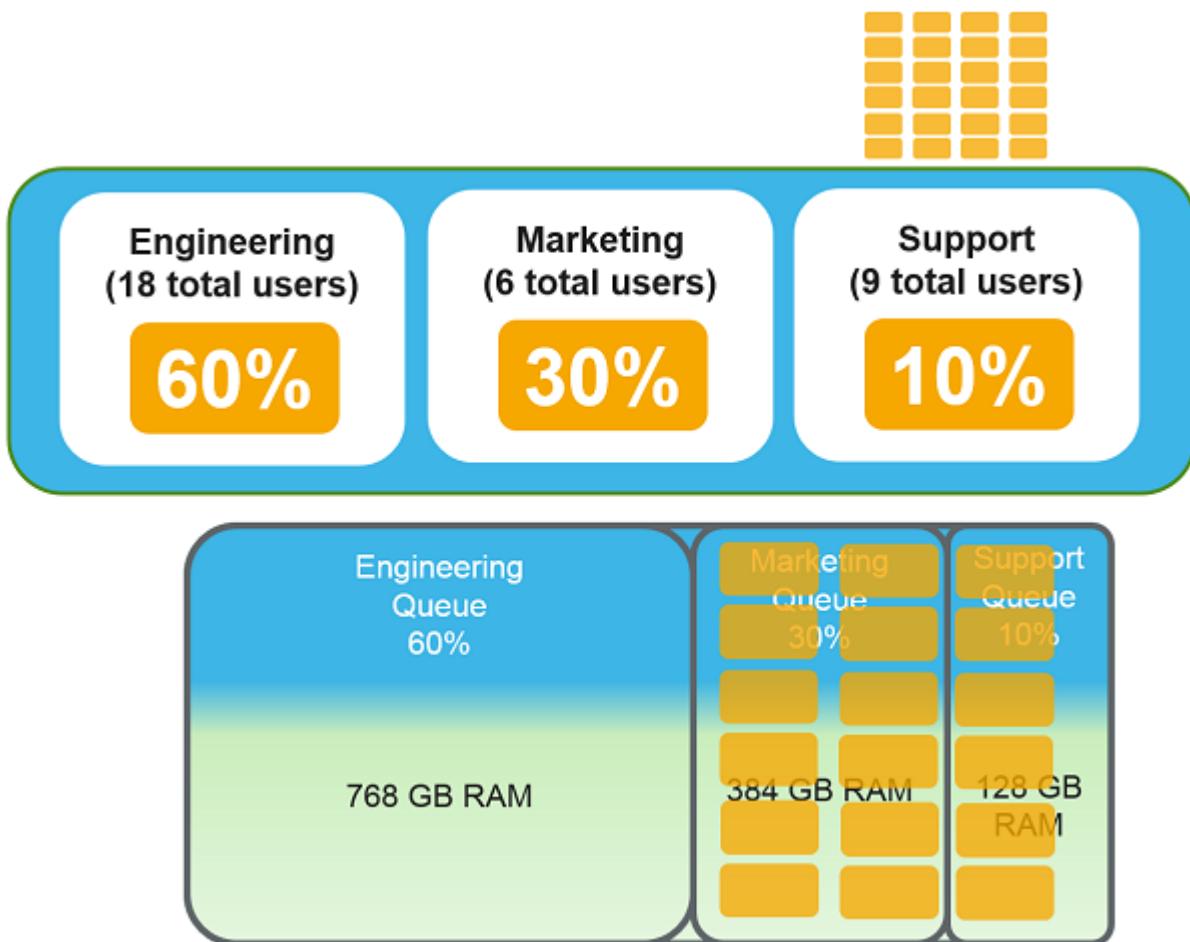


Queues can also be configured to strictly limit usage only to what the SLA grants, regardless of available cluster resources. In the illustration below, Support only gets 10% of cluster resources, regardless of whether or not other resources are being used.



Queues can furthermore be configured to allow only a certain amount of resource access beyond the base SLA guarantee. The illustration below represents this type of middle position - Support can “burst” beyond the 10% minimum, but only up to 40% of total resources. They can still get more than their guarantee, but less than total resources currently available.

The YARN Capacity Scheduler



The YARN Capacity Scheduler

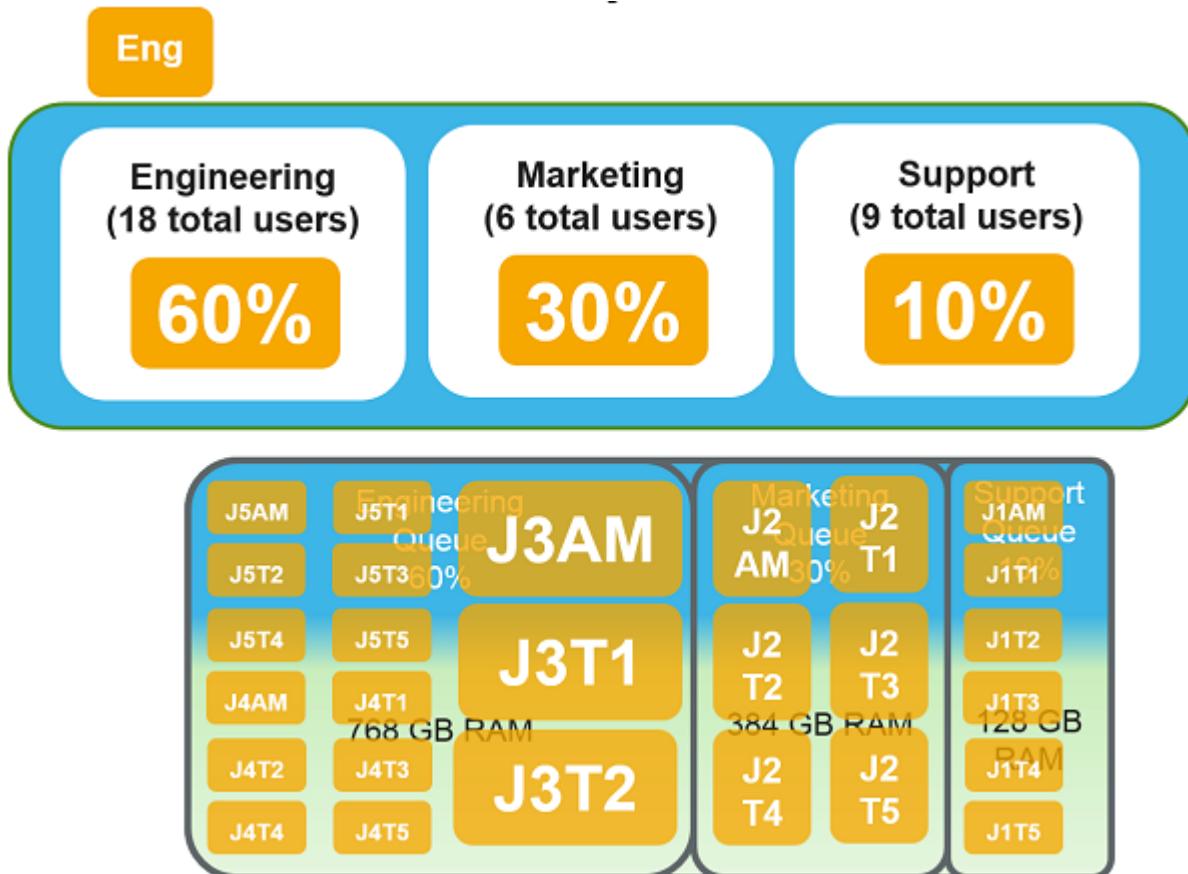
YARN Queue Manager (Ambari View)

The screenshot shows the Ambari YARN Queue Manager interface. At the top, there's a navigation bar with 'Ambari' and 'hortonworks' followed by 'Ops' and 'Alerts'. On the right, there are links for 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user icon labeled 'admin'. Below the navigation is a list of queues: 'root (100%)' and 'default (100%)', both marked with a green checkmark. The 'default' queue is selected. To the left, a sidebar titled 'Scheduler' contains settings like 'Maximum Applications' (10000), 'Maximum AM Resource' (20 %), 'Node Locality Delay' (40), 'Calculator' (org.apache.hadoop.yarn), 'Queue Mappings' (empty), and 'Queue Mappings Override' (disabled). In the center, under the 'default' queue, there are sections for 'Capacity' (Level Total 100%), 'Access Control and Status' (State: Running, Administer Queue: Anyone, Submit Applications: Anyone), and 'Resources' (User Limit Factor 1, Minimum User Limit 100 %, Maximum Applications: Inherited, Maximum AM Resource: Inher %, Ordering policy dropdown). At the bottom left, there's a 'Versions' section showing 'v2' (an hour ago) and 'v1' (Current, version1).

The YARN Queue Manager controls queues and their settings. It can be accessed by clicking the menu icon to the left of the **admin** menu button and selecting **YARN Queue Manager**. It is here that queues can be added or modified to meet business needs.

Queue Characteristics

Administrators need to fully understand the characteristics and behaviors of YARN queues.

SLAs and Elasticity

If you set up queues so that they can go beyond their guaranteed minimums, by default you are still at risk of not meeting SLAs. For example, in this illustration the Support queue has taken up 100% of the resources on the cluster. When a job assigned to the Engineering queue appears, it must wait for one of the other jobs to finish before it will be granted resources in the cluster. As a result, SLAs might not be met.

NOTE

In the illustration above:

J = Job

AM = ApplicationMaster

T = Task.

Thus J3T2 = Job 3, Task 2. J1AM = Job 1, ApplicationMaster.

Preemption

Container

Minimum Container Size (Memory)



0 MB 1024 MB 2304 MB

250MB

Maximum Container Size (Memory)



0 MB 1024 MB 2304 MB

2250MB

YARN Features

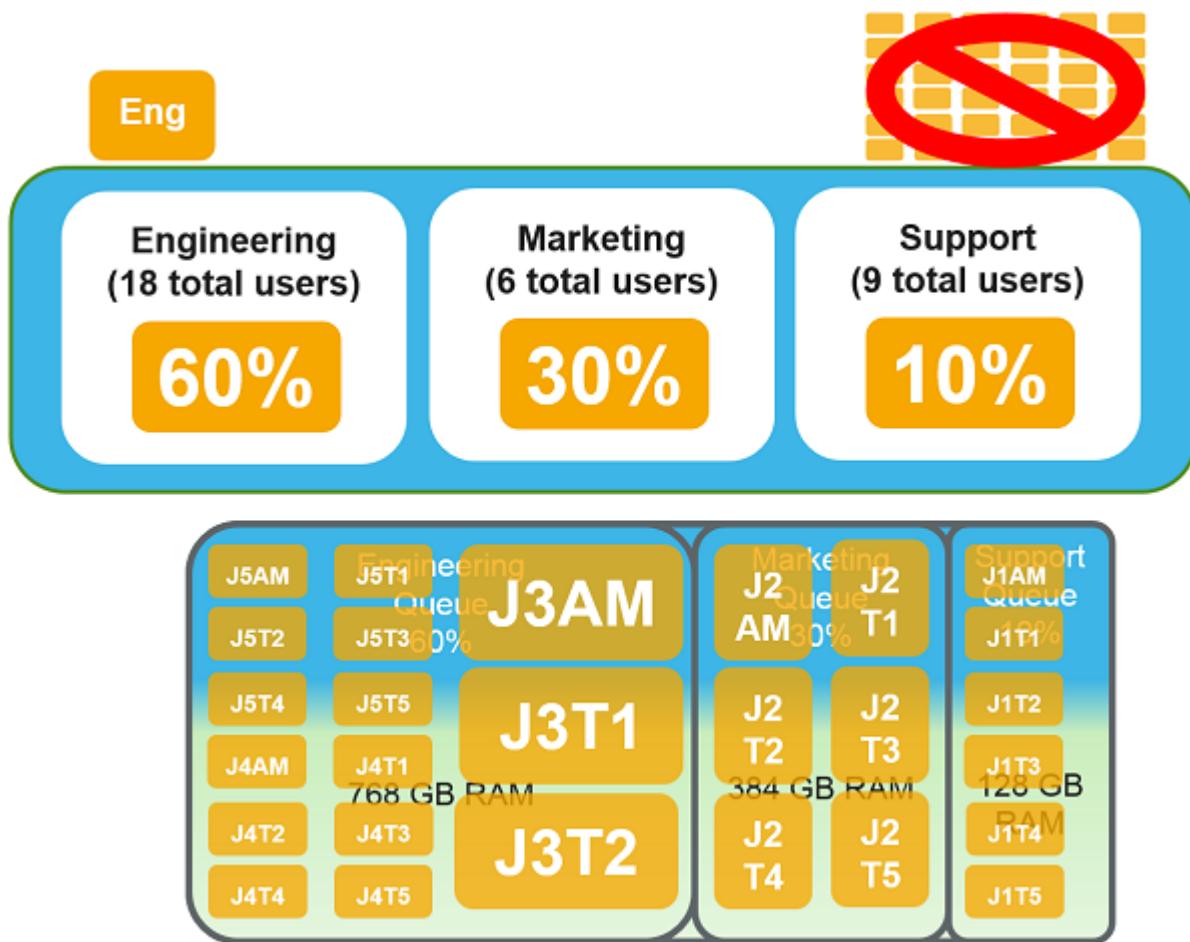
Node Labels

Disabled

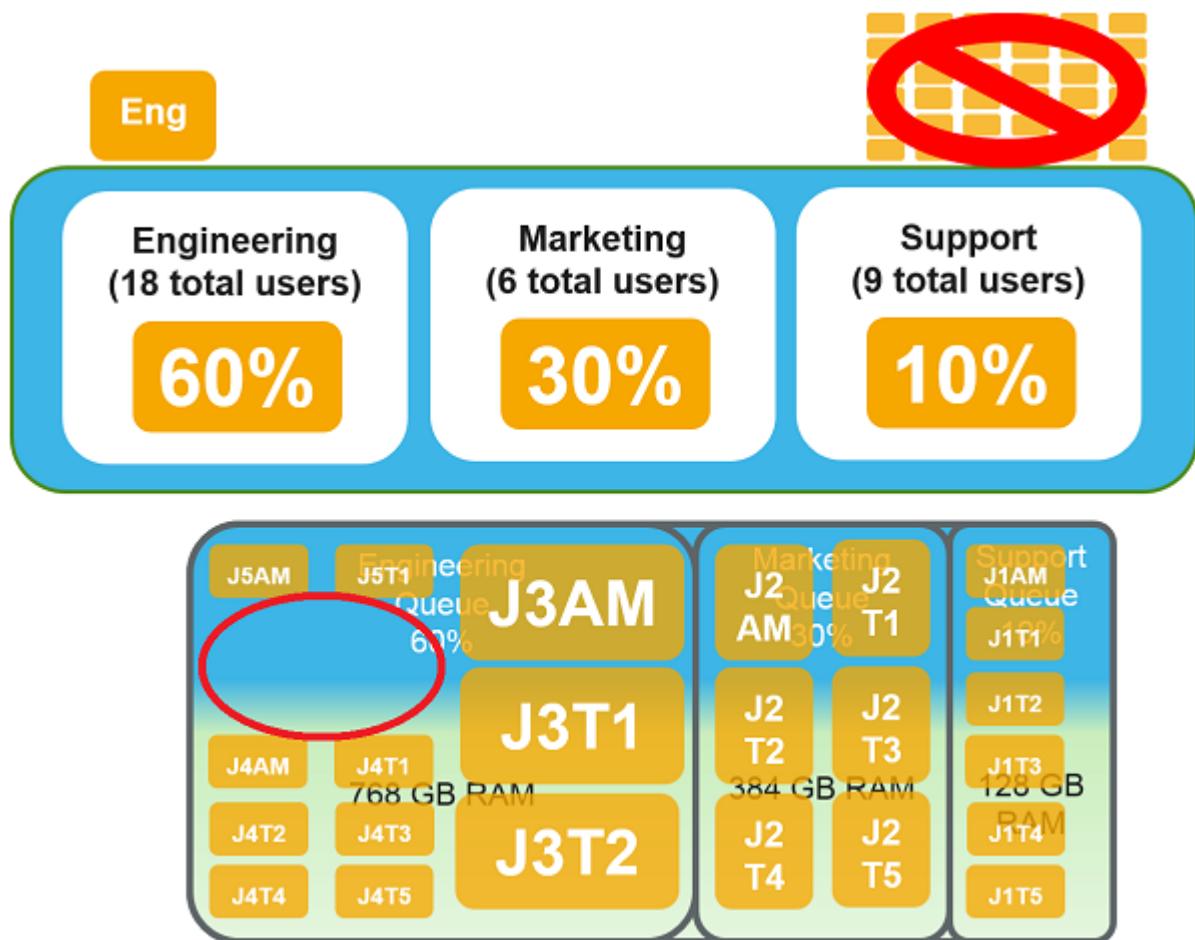
Pre-emption

Disabled

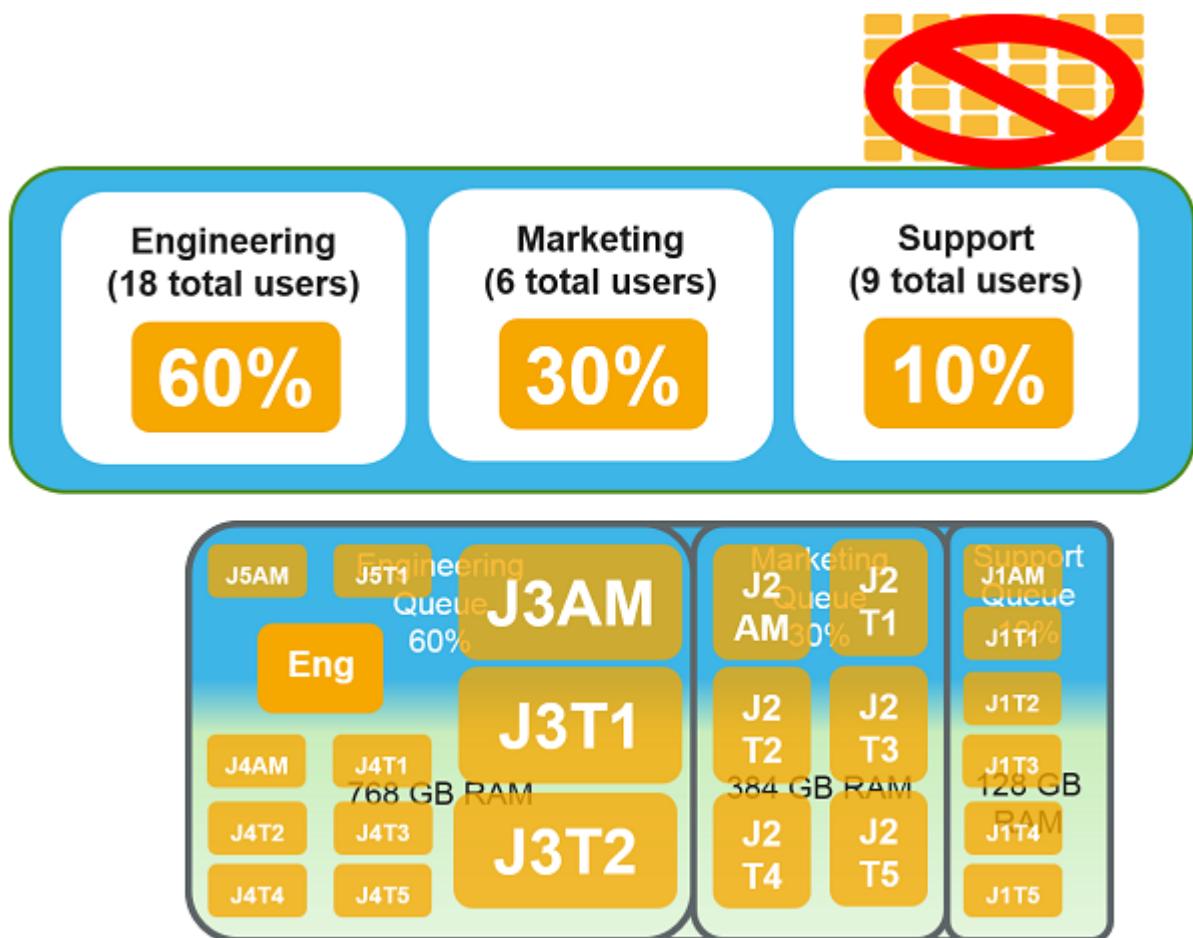
Preemption enables YARN to reclaim resources, forcibly if necessary, that should be allocated to another application that has a higher priority. You enable preemption under **Services > YARN > Configs > Settings**. Once enabled, when an application that is guaranteed resources on the cluster prepares to launch, YARN will interrupt any scheduled jobs from the queue that has exceeded its resource guarantees.

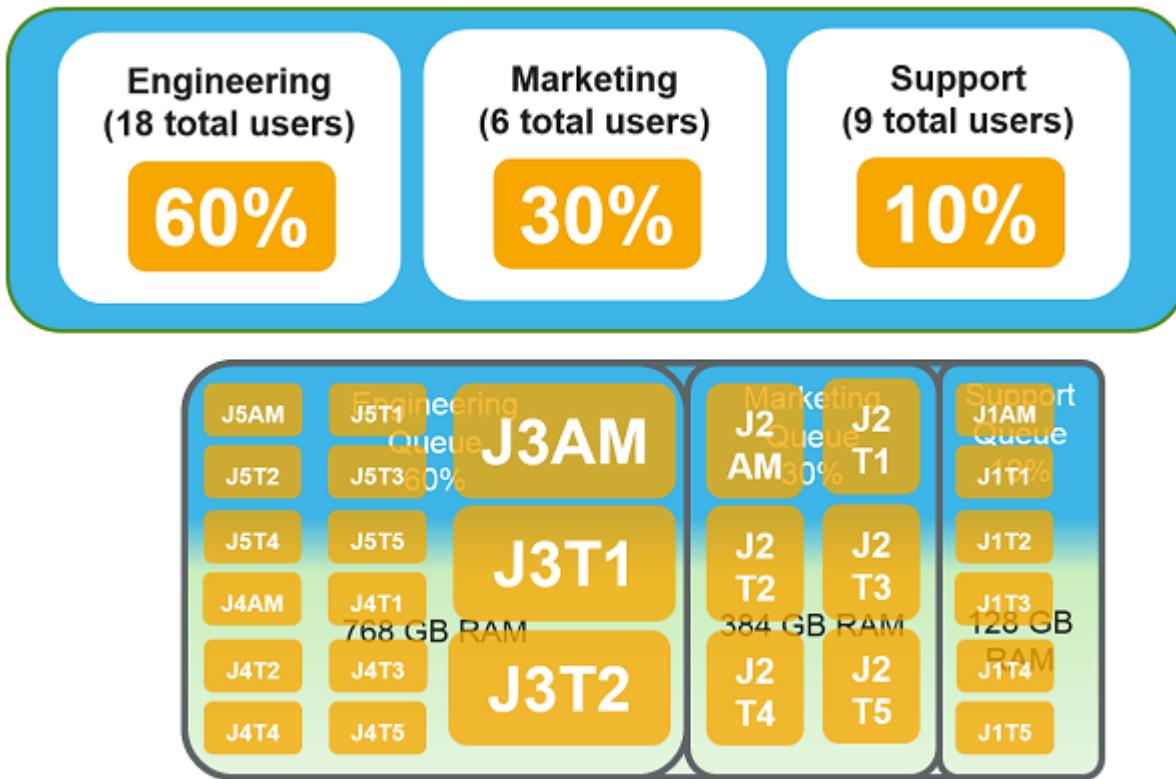


It then alerts applications on a Last In, First Out (LIFO) basis that their containers are getting ready to be removed from them and gives them a few seconds to shut down tasks gracefully. After this, if the required resources have not yet been freed up, YARN will forcibly kill the containers necessary (again, on a LIFO basis) in order to allocate the resources needed to meet the minimum queue guarantees for the new application.

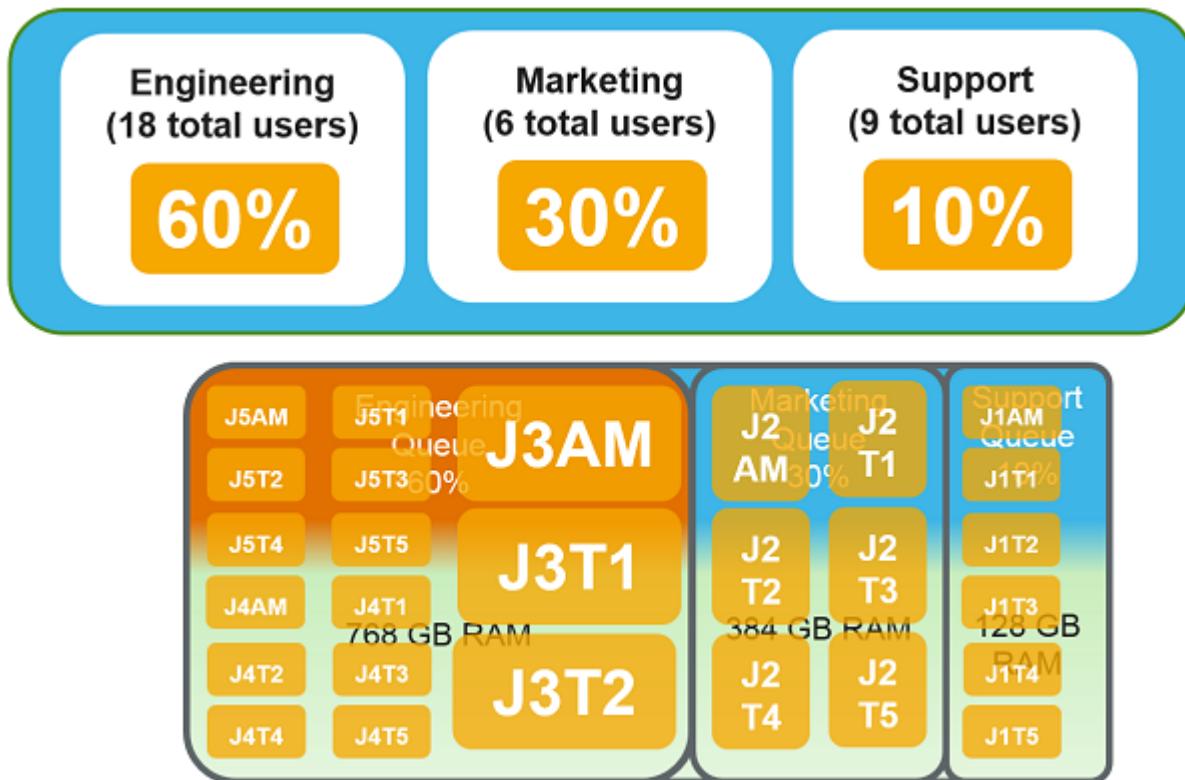


Then the new application will be launched.

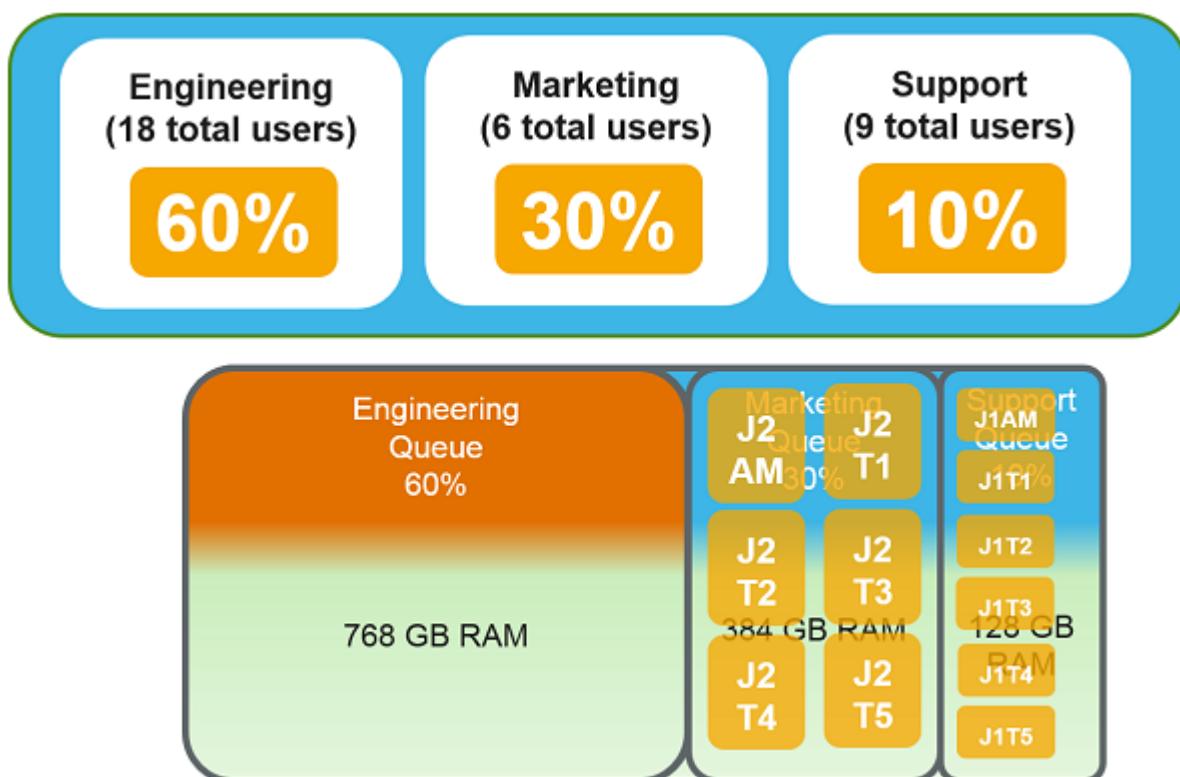


Queue States

Queues exist in one of two states: running and stopped. A stopped queue cannot have new applications assigned to it. If a running queue with applications deployed is changed from started to stopped, the applications currently running will be allowed to finish. In this example, the decision is made to temporarily disable – but not permanently delete – the Engineering queue. The queue is changed from running to stopped status.



The applications running in that queue will finish, but no new applications can be assigned to this queue's resources.



The YARN Capacity Scheduler

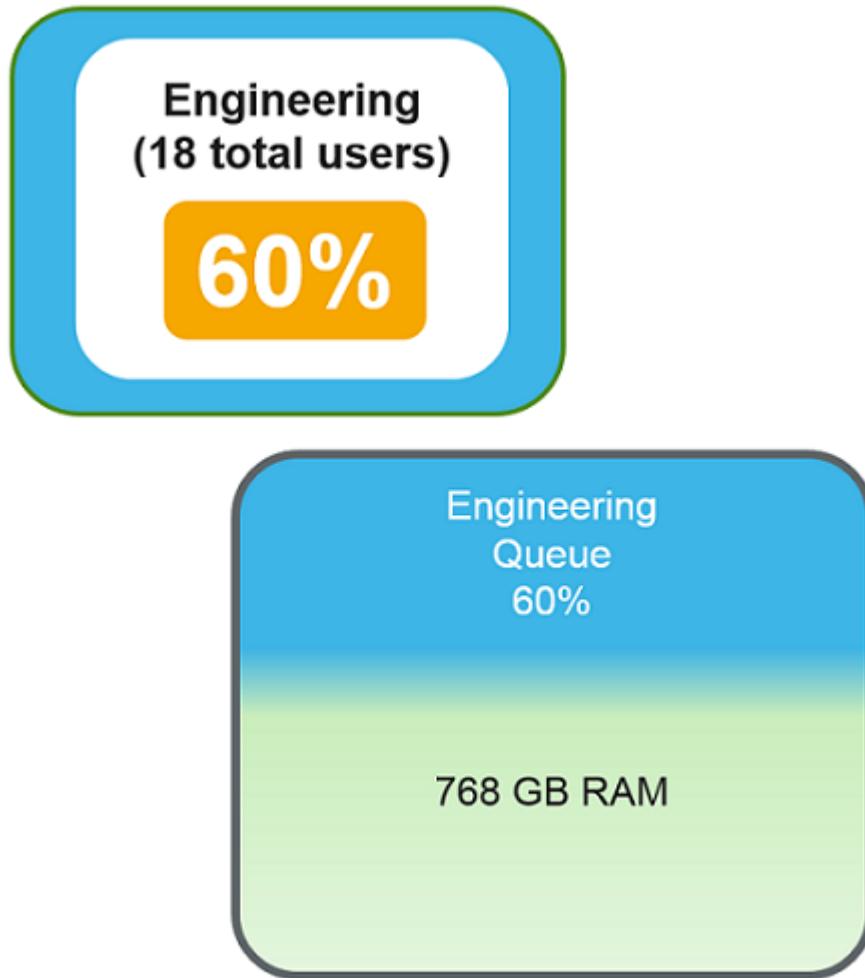
This enables a non-intrusive “draining” of the queue without affecting in-place applications.

The resources that are now not being used by the Engineering queue are now available to users of other queues if their configuration allows it. Resources are granted on a FIFO, first come, first served basis.

Queues can be stopped by going to the YARN Queue Manager view.

The screenshot shows the Ambari interface for managing YARN queues. On the left, there's a sidebar with 'Add Queue' and 'Actions' buttons. Below that is a list of queues: 'root (100%)' (green), 'default (0%)' (blue, selected), 'Engineering (60%)' (green), 'Marketing (30%)' (green), and 'Support (10%)' (green). The main panel is titled 'default' and shows 'root.default' under 'Capacity'. It has two sliders: 'Capacity' set to 0% and 'Max Capacity' set to 66%. A checkbox for 'Enable node labels' is present. Below this is a 'Scheduler' section with various configuration fields like 'Maximum Applications' (10000), 'Maximum AM Resource' (20%), 'Node Locality Delay' (40), 'Calculator' (org.apache.hadoop.yarn), 'Queue Mappings' (empty), and 'Queue Mappings Override' (checkbox). To the right are sections for 'Access Control and Status' (State: Running/Stopped, Administer Queue: Anyone/Custom, Submit Applications: Anyone/Custom) and 'Resources' (User Limit Factor: 1, Minimum User Limit: 100%, Maximum Applications: Inherited, Maximum AM Resource: Inher, Ordering policy dropdown).

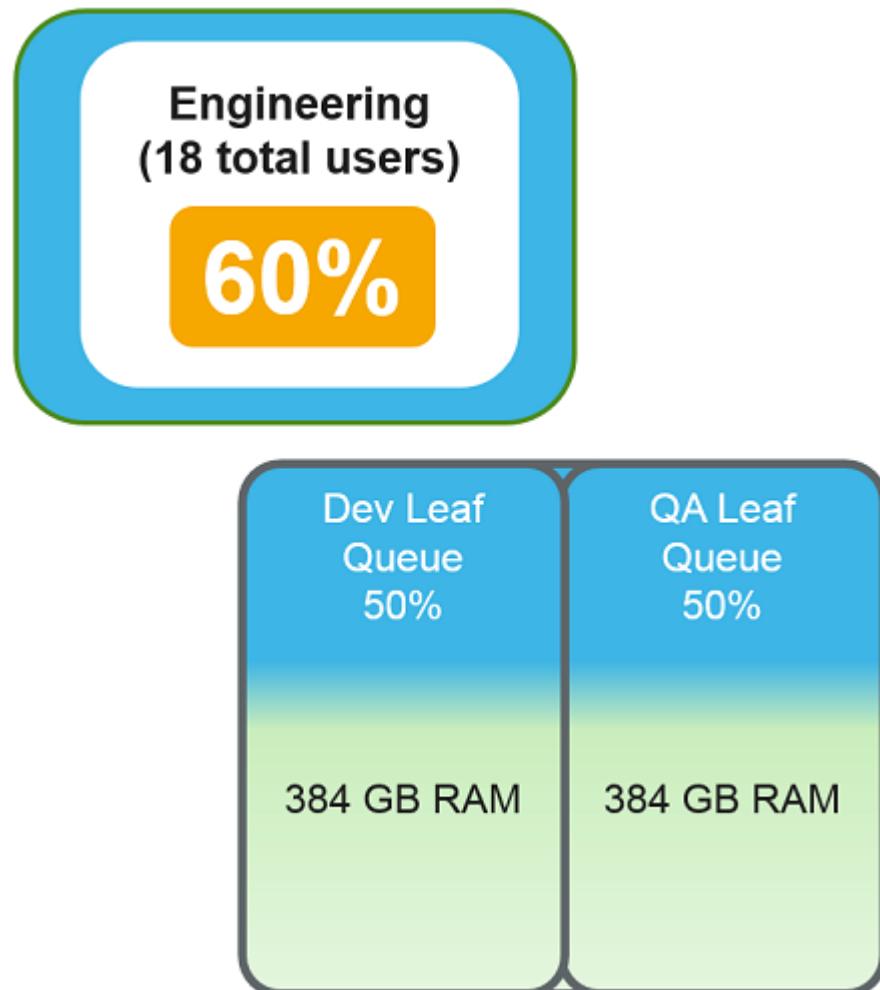
In this example, we have reconfigured all root resources to our Engineering, Marketing, and Support queues and we want to disable the default queue. Because the default queue cannot be deleted (nor can the root queue), the next best thing is to simply change it to a Stopped state. Thus, even though in the screen capture the default queue still has a maximum capacity of 66%, the stopped state will prohibit any applications from being assigned to this queue.

Parent and Leaf Queues

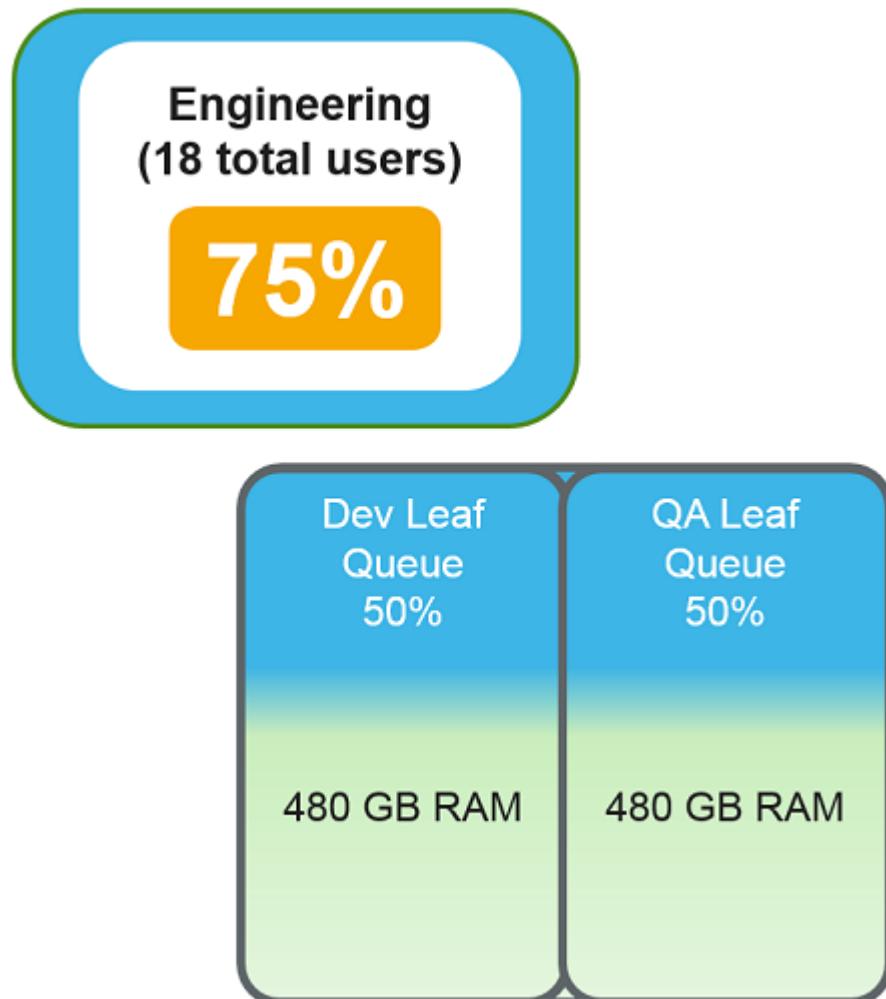
There are two types of queues: parent queues and leaf queues. A parent queue is simply a queue that has one or more configured leaf queues. A leaf queue is a child of a parent queue. Just like parent queues, a leaf queue is assigned a certain percentage of the parent queue's resources. Unlike a parent queue, leaf queues can be assigned job tasks. Once a queue has been configured with one or more leaf queues, that queue will no longer accept application requests.

Every queue is a leaf of the overall “root” queue, which represents all resources available in the cluster. The root queue has at least one queue configured at all times – the queue named “default” – and as such, no resources can be assigned to the root queue.

In this example, we will take a look at the Engineering queue – which has been configured as a leaf of the root queue. Engineering queue gets 60% of the total resources guaranteed to it in the cluster. When a leaf queue is configured for Engineering, that 60% of the overall cluster resources must be divided between the configured leaf queues. For example – if you plan to divide the Engineering resources up evenly between Dev and QA, you would configure two appropriately named leaf queues and give each of them 50% of the Engineering queue's resources. (Which means they each get 30% of the overall cluster resources, because they are getting 50% of 60% of the cluster.)



By default, all changes to a parent queue will flow down to its leaf queues. For example, if you change the total resource percentage available to the Engineering queue from 60% to 75% of the cluster, the leaf queues would automatically each get their proportional share of the newly available resources.



To create a leaf queue, in the YARN Queue Manager view, first select the queue that you want to make a parent. Then click the **Add Queue** button on the top-left. Enter a name for the leaf queue and click the **check mark** to create it. By default, the leaf queue will be assigned whatever resources are remaining, so if this is the first leaf queue configured, it will get 100% of the parent queue. If you intend to add another leaf queue, you can use the sliders to drop the capacity percentage down, then highlight the parent queue again and repeat the process.

The YARN Capacity Scheduler

The screenshot shows the Ambari interface for managing the YARN Capacity Scheduler. On the left, a sidebar lists queues: root (100%), default (0%), Engineering (60%), Dev (50%), QA (50%), Marketing (30%), and Support (10%). The QA queue is selected. The main panel displays the configuration for the QA queue under the root.Engineering.QA path. The 'Capacity' section shows a Level Total of 100% and a capacity of 50%. The 'Access Control and Status' section indicates the queue is running and anyone can administer or submit applications. The 'Resources' section includes settings for User Limit Factor (1), Minimum User Limit (100%), Maximum Applications (Inherited), Maximum AM Resource (Inher.), and Ordering policy (fifo). The 'Scheduler' section contains parameters like Maximum Applications (10000), Maximum AM Resource (20 %), Node Locality Delay (40), and Calculator (org.apache.hadoop.yarn).

Queue Refresh

This screenshot shows the same Ambari interface as above, but with the 'Actions' dropdown menu open. The 'Save and Refresh Queues' option is highlighted, indicating it has been selected. The rest of the interface remains the same, showing the configuration for the QA queue.

The YARN Capacity Scheduler

Changes to queue configuration are not effective until the queue has been refreshed. Depending on what type of change is made, queues might be able to be refreshed without restarting YARN, or they might require that the ResourceManager be restarted before they can go into effect. For example, adding leaf queues can be done without restarting the ResourceManager, but deleting or disabling a queue requires a ResourceManager restart. In the Yarn Queue Manager view, click the orange **Actions** button to see which option is available based on the changes you want to make.

You can also elect to make and save configuration changes without immediately refreshing the queues. This might be useful when making a destructive change (deleting a queue, for example) but wanting to wait until the cluster is less heavily utilized before implementing it.

If you choose to implement your changes, you will be asked to leave a note detailing what you did. Afterwards, the changes will be made.

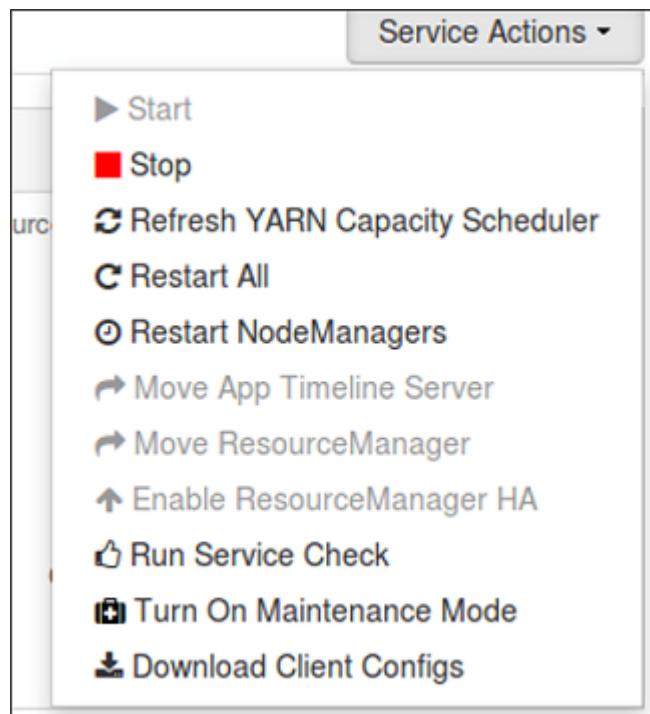
The screenshot shows the Ambari interface for managing YARN queues. On the left, a tree view lists queues: root (100%), default (0%), Engineering (60%), Dev (50%), QA (50%), Marketing (30%), and Support (10%). The QA queue is selected. A modal dialog titled 'Notes' contains the text 'Subdivided Engineering queue into Dev and QA'. Below the notes, there are 'Close' and 'Save changes' buttons. To the right of the notes, there's a 'Resources' section with various configuration fields like User Limit Factor (1), Minimum User Limit (100 %), Maximum Applications (Inherited), Maximum AM Resource (Inher %), and Ordering policy (fifo). At the bottom of the resources section, there are buttons for 'Administer Queue' (Anyone, Custom) and 'Submit Applications' (Anyone, Custom).

You can confirm those changes by going to the ResourceManager UI and clicking the **Scheduler** link. In this example, we can now see that Engineering has two leaf queues configured.

The screenshot shows the ResourceManager UI's Application Queues page. The legend indicates the status of each queue: Capacity (green), Used (blue), Used (over capacity) (orange), and Max Capacity (grey). The tree view shows the following hierarchy: Queue: root, Queue: Engineering, Queue: Engineering.Dev, Queue: Engineering.QA, Queue: Marketing, Queue: Support, and Queue: default. The 'Used (over capacity)' status is highlighted for the Engineering, Engineering.Dev, and Engineering.QA queues.

You can also refresh capacity scheduler queues by going to **Services > YARN** and clicking the **Service Actions** menu button and choosing **Refresh Capacity Scheduler**.

The YARN Capacity Scheduler



A third option for an HDP administrator is from the command line use the `yarn rmadmin -refreshQueues` command.

Application Limits

The screenshot shows the Ambari Queue Configuration page for the 'default' queue under the 'YARN' service. The page includes the following sections:

- Actions:** + Add Queue, Actions dropdown.
- Queue List:** root (100%), default (100%) (selected).
- Scheduler:** Maximum Applications (10000), Maximum AM Resource (20 %), Node Locality Delay (40), Calculator (org.apache.hadoop.yarn), Queue Mappings, Queue Mappings Override (Disabled).
- Capacity:** Capacity (100 %) and Max Capacity (100 %) sliders, Level Total (100%).
- Access Control and Status:** State (Running), Administer Queue (Anyone), Submit Applications (Anyone).
- Resources:** User Limit Factor (1), Minimum User Limit (100 %), Maximum Applications (Inherited), Maximum AM Resource (Inher %), Ordering policy.
- Versions:** v2 (an hour ago), v1 (version1).

Application limits can be set on a per-queue basis in order to control the behavior of YARN applications. To avoid overloading cluster resources – caused either by malicious users, or by accident – you can place a static, configurable limit on the total number of concurrently active (both running and pending) applications running at the same time. The maximum-applications configuration property is used to set this limit, with a default value of 10,000.

The limit for running applications in any specific queue is a fraction of this total limit, proportional to its capacity. This is a hard limit, which means that once this limit is reached for a queue, any new applications to that queue will be rejected, and clients will have to wait and retry later. This limit can be explicitly overridden on a per-queue basis.

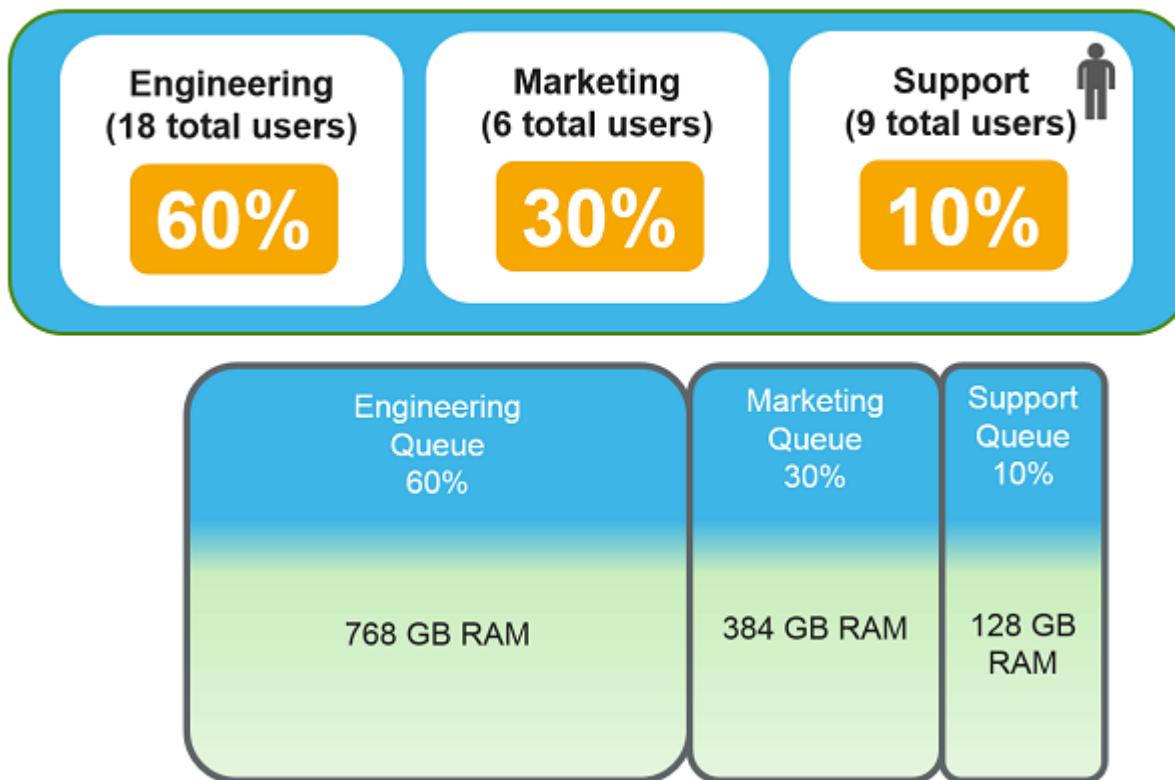
There is another resource limit that can be used to set a maximum percentage of cluster resources allocated specifically to ApplicationMasters. The Maximum AM Resources property has a default value of 20%, and exists to avoid cross-application deadlocks where significant resources in the cluster are occupied entirely by the Containers running ApplicationMasters, but sufficient resources do not exist for the ApplicationMasters to perform any job tasks.

To configure application limits, go to the YARN Queue Manager view in Ambari Web UI and find Scheduler section on the left side of the window. Here you can set the value for the Maximum Applications number and Maximum AM (ApplicationMaster) Resource usage percentage.

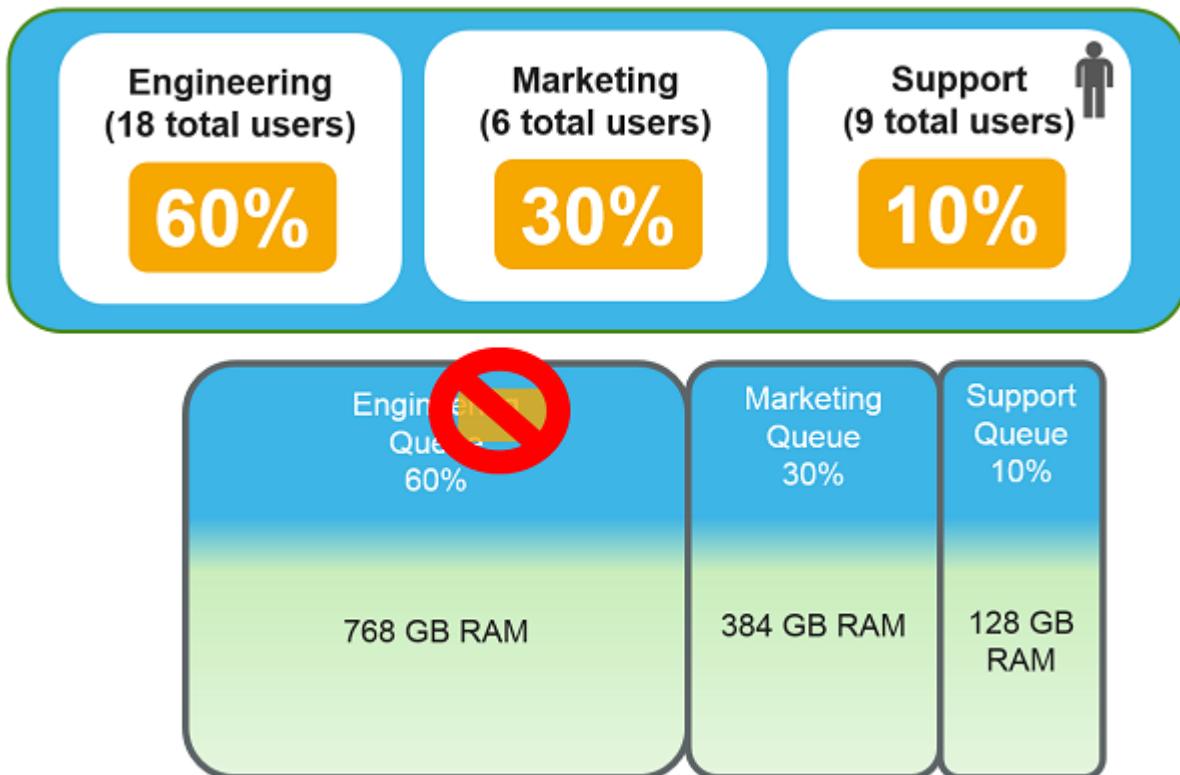
Queue Access Control

By default any user can specify any queue when launching an application. This can be controlled and managed by the HDP administrator.

Queue ACLs



ACLs allow the HDP administrator to specify queues that can only be used by specific users or groups. As with all other settings, a leaf queue's ACLs are inherited from parent unless otherwise specified.



Queue ACLs are set by modifying the **Submit Applications** setting under **Access Control and Status** in the YARN Queue Manager view.

Select the **Custom** button – which by default blocks access to all users and groups.

The YARN Capacity Scheduler

The screenshot shows the Ambari interface for managing the YARN Capacity Scheduler. On the left, a tree view lists queues: root (100%), default (0%), Engineering (60%), Dev (50%), QA (50%), Marketing (30%), and Support (10%). The 'Dev' queue is selected. On the right, a detailed configuration window for 'root.Engineering.Dev' is open. The 'Capacity' section shows 'Level Total' at 100% and 'Dev' capacity set to 50%. The 'Access Control and Status' section shows 'State' as 'Running', 'Administrator Queue' as 'Anyone', and 'Submit Applications' as 'Anyone'. The 'Resources' section includes fields for 'User Limit Factor', 'Minimum User Limit', 'Maximum Applications', 'Maximum AM Resource', and 'Ordering policy'. The 'Groups' field contains 'dev'.

Then type the list of users and groups in the appropriate area below in a comma-separated list.

This screenshot is similar to the previous one but shows a user interaction. In the 'Groups' field of the 'Access Control and Status' section, the text 'dev' has been typed into the input field. The rest of the configuration remains the same as in the first screenshot.

When finished, click the orange **Actions** button and then click **Save and Refresh Queues**.

The YARN Capacity Scheduler

The screenshot shows the Ambari interface for managing the YARN Capacity Scheduler. On the left, there's a sidebar with actions like 'Save and Restart ResourceManager', 'Save and Refresh Queues', 'Save Only', and 'Download config'. Below that is a list of queues: Dev (50%), QA (50%), Marketing (30%), and Support (10%). The Support queue is currently selected. To the right of the sidebar is a large panel for the 'Scheduler' configuration, which includes fields for Maximum Applications, Maximum AM Resource, Node Locality Delay, Calculator, Queue Mappings, and Queue Mappings Override. Overlaid on this is a modal window titled 'Support' with tabs for 'Isot Support' and 'Peer Support'. The 'Isot Support' tab is active, showing the capacity configuration. It has a 'Capacity' section where 'Level Total' is set to 100% and the 'Support' queue is configured with a capacity of 10%. There are also sections for 'Access Control and Status' (with State set to Running, Administer Queue set to Anyone, and Submit Applications set to Anyone) and 'Resources' (with User Limit Factor set to 1, Minimum User Limit set to 100%, and Maximum Applications set to Inherited). A note in the Resources section says 'leave blank to deny access for everyone'.

Default Queue Mapping

Default queue mapping removes the need for a user to specify a queue when launching an application. When a user launches an application without specifying a queue, YARN will automatically assign the application to the appropriate queue. By default, this cannot be overridden – thus if a user tries to specify a queue other than the one to which they are mapped, the application will still be directed to the administrator-configured default queue. In some cases, default queue mapping may need to be overridden, in which case the value to allow for this must be set.

Setting and Priority

Scheduler		State	Running	Stop
Maximum Applications	10000	Administer Queue	Anyone	Custom
Maximum AM Resource	20 %	Submit Applications	Anyone	Custom
Node Locality Delay	40			
Calculator	org.apache.hadoop.yarn.			
Queue Mappings	u:support01:Support,u:support02:Support,u:support13:Support,g:promo:	<input type="button" value="Edit"/>		
Queue Mappings Override	<input type="checkbox"/> Disabled			

Default queue mapping settings are configured in the YARN Queue Manager view under the Scheduler section. The one key thing to remember about them is that the first mapping that is read will apply for any given user. For example, assume a user has both a user and a group mapping. If the user mapping appears first in the list, that will be the default queue for that user. However, if a group that this user belongs to has a queue mapping that appears before the user-specific queue mapping, it will be the group mapping, not the user mapping, that takes precedence.

Syntax

The syntax for queue mapping is as follows:

- User mappings format: u:<username>:<queuename>
- Group mappings format: g:<groupname>:<queuename>

These should be formatted as a comma-separated list, in order of priority (in case of conflicts).

Example:

```
u:support01:Support,u:support02:Support,u:support13:Support,g:promo:Marketing,g:sales:Marketing,g:dev:Dev,g:qa:QA
```

NOTE

Take note that leaf queue names (Dev and QA in the example above) do not contain their parent queue names. All queue names must be unique, even if they are leaf queues.

If queues have been configured to match user and group names, a shortcut default mapping can be configured:

- `u:%user:%user` assigns all users to a queue that matches their user name
- `u:%user:%primary_group` assigns all users to a queue that matches their group name

ACLs vs. Default Queue Mappings

ACL settings must allow application mapping according to default queue mapping settings – otherwise, applications will fail to launch.

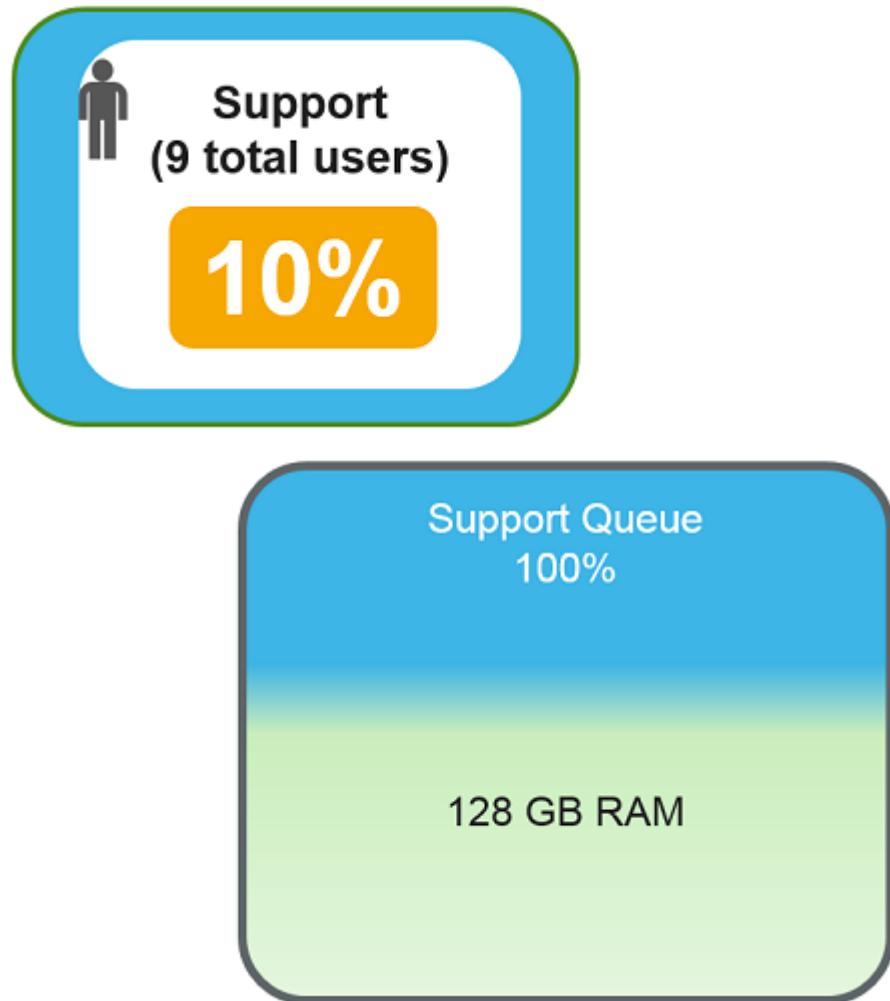
Example:

User support13 is mapped to the Support queue by default, however ACLs are set so that the user is blocked from accessing the Support queue. If default queue mapping override is not enabled, the user cannot launch any applications in the cluster. If override is enabled, then the user will have to specify a different queue when launching an application.

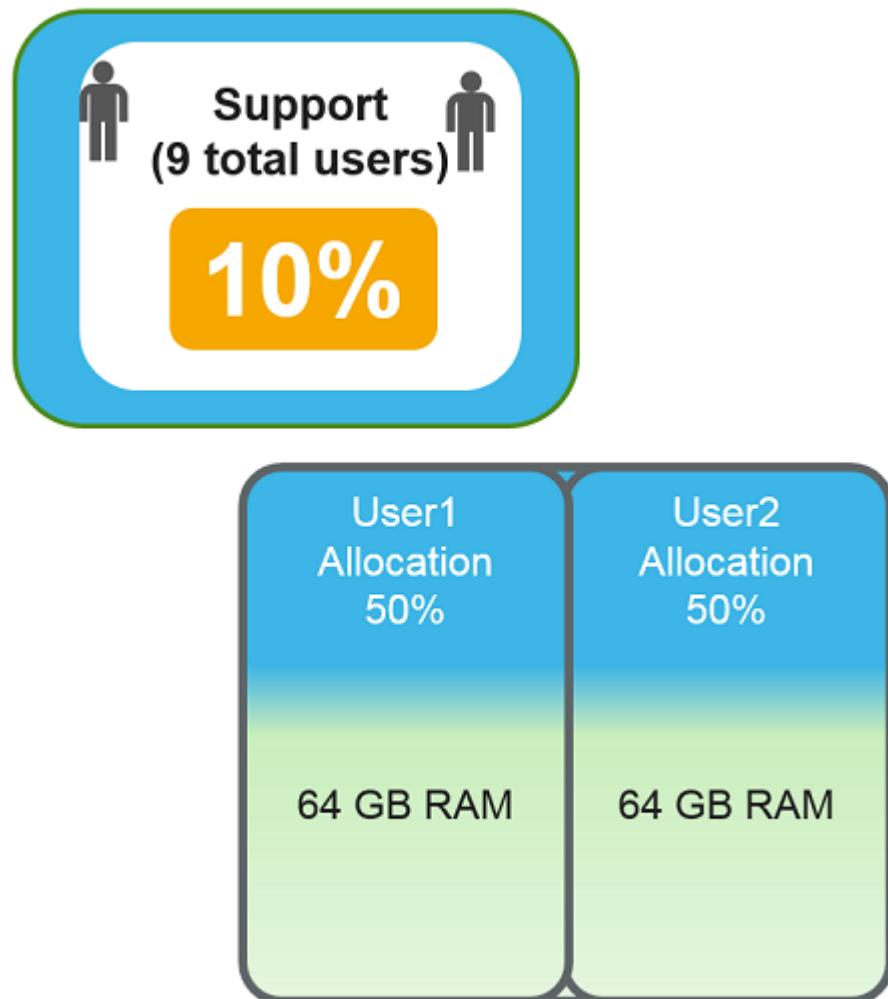
User Limits

In addition to ACLs and Default Queue Mapping, user limits can be set that will affect cluster behavior.

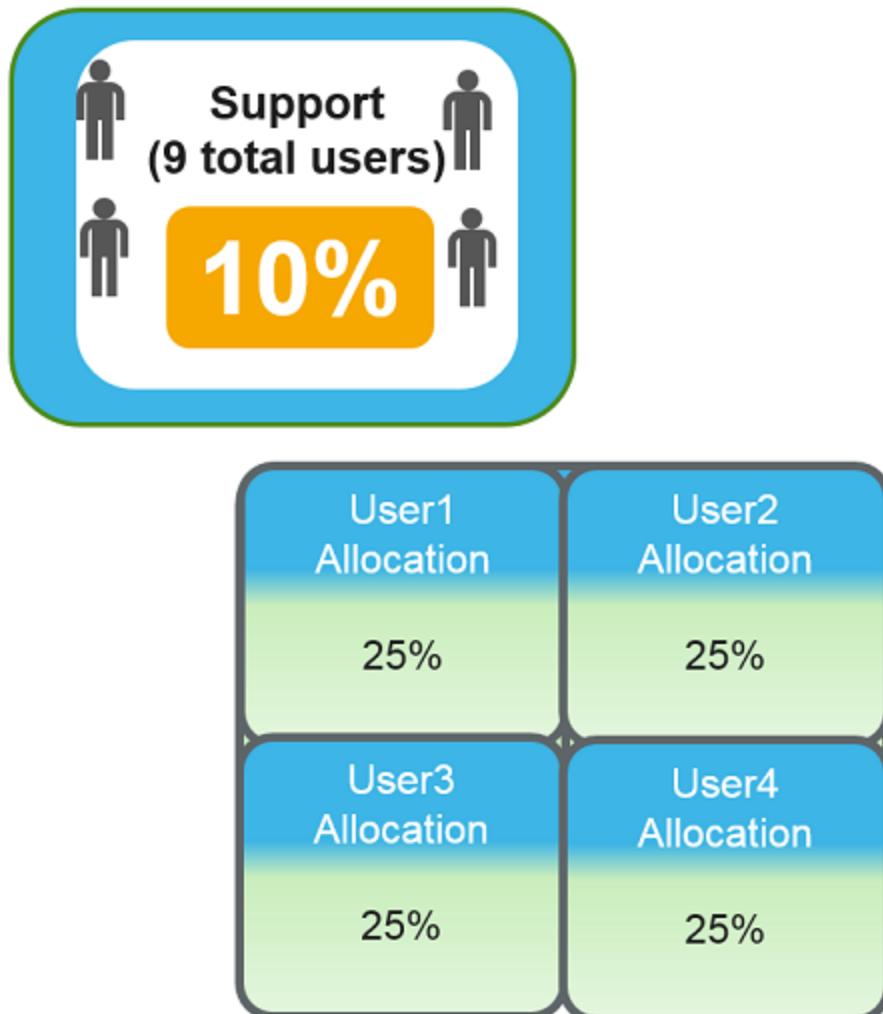
Minimum User Limits



By default, the minimum user limit restricts the number of users that can access queue resources at one time. If only a single user submits jobs to a queue, by default that user is guaranteed 100% of the queue capacity. However, depending on Minimum User Limit settings, as additional users join, the queue can be configured so that resources are divided up evenly among the users. A setting of 50% for this value allows two users to run applications, each entitled to 50% of queue capacity.

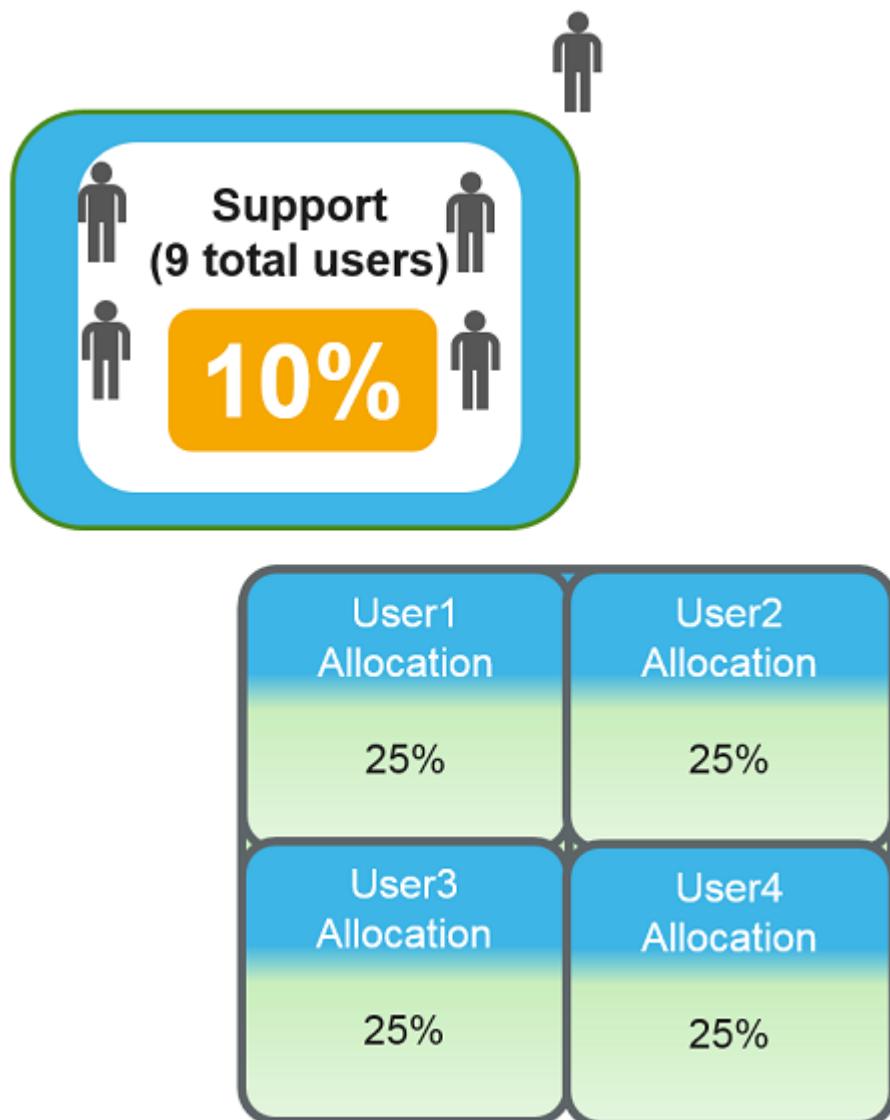


A value of 25% allows four users to split the capacity with a 25% share each.



A value of 10% allows 10 users 10% each, a value of 5% allows 20 users 5% each, and so on. The minimum user limit places a lower-bound guarantee on the percent of queue capacity available to any given user.

Example: The minimum user limit is set to 25%, thus a maximum of four users can access the queue at any given time. If a fifth user tries to access the queue, their applications will have to wait until one of the previous users are finished, regardless of whether or not the queue itself still has resources available, because to allow a fifth user would mean YARN could no longer guarantee the availability of 25% of the capacity to the first four users.



Minimum user limits are only strictly enforced when a queue has the same setting for capacity and maximum capacity (meaning no elasticity). The minimum user limit reserves the % of the capacity setting. By default, in a queue where Capacity and Maximum Capacity are the same value, only one user can run an application at one time. However, if the maximum capacity setting exceeds the guaranteed capacity setting, then a second user can launch applications and use whatever extra elastic resources are available.

User Limit Factors

The user limit factor value enables control over what percentage of cluster resources a single user can access. By default, an individual user is limited to the minimum queue guarantee.

Example:

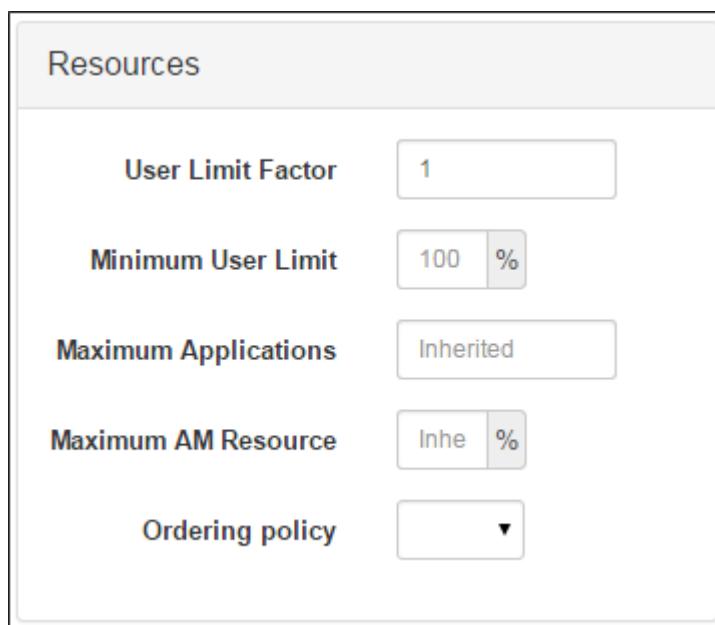
- Support queue has a 10% minimum, but can access up to 100% of cluster resources if available. A single user in that queue could use up to 10% of cluster resources.

User limit factor setting configures a maximum percentage of cluster resources available to any user based on their queue's allocated resource guarantee.

- **Example 1:**
User limit factor of 1 (the default setting) means the maximum resources a Support queue user can access is 100% of the minimum queue allocation, or 10% of the cluster.
- **Example 2:**
User limit factor of 0.5 means the maximum individual user allocation is 50% of the minimum queue allocation, or 5% of the cluster.
- **Example 3:**
User limit factor of 2 means the maximum individual user allocation is 200% of the minimum queue allocation, or 20% of the cluster.

NOTE

Under no circumstance can a user gain access to more than the maximum percentage of cluster resources as configured in the queue settings.

Configure User Limit and User Limit Factors

User limit and user limit factor are configured under the Resources section of the YARN Queue Manager view.

Queue Administrators

A normal user can submit applications, as well as view information about and control those own applications. Queue administrators can be configured who have the ability to submit applications just like any other user, view information about all applications in a queue, and kill any application in the queue. This power can be granted without granting any additional administrative privileges.

Configure Queue Administrators

Access Control and Status

State Running Stopped

Administer Queue Anyone Custom ↻

Users	dev01	
Groups	Comma-separated list of gro	

leave blank to deny access for everyone

Submit Applications Anyone Custom

Users	Comma-separated list of user	
Groups	dev	

leave blank to deny access for everyone

Queue administrators are configured in the Access Control and Status section of the YARN Queue Manager View, under Administer Queue. To configure an admin, click the **Custom** button, then specify the user or group of users that is to have administrative access to applications in this queue.

In the example above, anyone in the dev group would be allowed to control and view their own applications, but user dev01 could control and view any application in that queue

Knowledge Check Questions

- 1) The fundamental unit in YARN resource scheduling is the _____.
- 2) Queues (can/cannot) use resources beyond the Capacity setting.
- 3) If you want to allow a queue to interrupt applications from other queues if it has not yet met its minimum guarantee, you need to enable _____.
- 4) Unused cluster resources are granted to applications on a (FIFO/LIFO) basis.
- 5) When an application is preempted, YARN kills containers on a (FIFO/LIFO) basis.
- 6) The Sales queue has 40% of cluster resources reserved. You want to configure a leaf queue to have 10% of total cluster resources reserved. When you create the leaf queue, to what should you set its Capacity setting?
- 7) Stopping a queue (does/does not) automatically kill running applications in that queue after a configured amount of time has passed.
- 8) What mechanism ensures large containers will eventually have a node on which they can run even if resources are fragmented across nodes?
- 9) QueueA has been configured to allow Jane access too submit applications. Her default queue mapping is set to QueueB. Is there any way for Jane to launch an application on the cluster?
- 10) Bill is assigned to QueueB, which has a minimum guarantee of 20% of cluster resources, and a maximum of 50%. The user limit factor for QueueB = 3. What is the effective limit on the percentage of cluster resources Bill can access?
- 11) QueueA has a minimum user limit value of 20%. How many users can run applications at one time in this queue?
- 12) An administrator leaves the minimum user limit of QueueA at 100%. How many users can run applications at one time in this queue?
- 13) Sayad is configured as a queue administrator for the Engineering queue. The HDP administrator configured two leaf queues – Engineering.Dev and Engineering.QA. Does Sayad also have queue administrator privileges at the leaf queue level?

Knowledge Check Answers

- 1) The fundamental unit in YARN resource scheduling is the _____.
Queue
- 2) Queues (can/cannot) use resources beyond the Capacity setting.
Can. Capacity Maximum sets the maximum, but capacity sets the guaranteed minimum.
- 3) If you want to allow a queue to interrupt applications from other queues if it has not yet met its minimum guarantee, you need to enable _____.
Preemption
- 4) Unused cluster resources are granted to applications on a (FIFO/LIFO) basis.
FIFO
- 5) When an application is preempted, YARN kills containers on a (FIFO/LIFO) basis.
LIFO
- 6) The Sales queue has 40% of cluster resources reserved. You want to configure a leaf queue to have 10% of total cluster resources reserved. When you create the leaf queue, to what should you set its Capacity setting?
25%. (10% of cluster resources = 25% of the Sales queue's total 40%)
- 7) Stopping a queue (does/does not) automatically kill running applications in that queue after a configured amount of time has passed.
Does not. Stopping a queue prevents new applications from launching, but does not affect running applications.
- 8) What mechanism ensures large containers will eventually have a node on which they can run even if resources are fragmented across nodes?
Reservations
- 9) QueueA has been configured to allow Jane access to submit applications. Her default queue mapping is set to QueueB. Is there any way for Jane to launch an application on the cluster?
Yes, if queue mapping override is disabled, Jane can specify to use QueueA when she launches an application. Otherwise, no.
- 10) Bill is assigned to QueueB, which has a minimum guarantee of 20% of cluster resources, and a maximum of 50%. The user limit factor for QueueB = 3. What is the effective limit on the percentage of cluster resources Bill can access?
50%. The user limit factor sets an upper bound of 3x the minimum, which would be 60%, however Bill cannot access more resources than the queue itself will allow. Thus, 50% is the upper limit, no matter what the user limit factor is configured to be.

11) QueueA has a minimum user limit value of 20%. How many users can run applications at one time in this queue?

Five, unless elasticity has been configured.

12) An administrator leaves the minimum user limit of QueueA at 100%. How many users can run applications at one time in this queue?

One, unless elasticity has been configured.

13) Sayad is configured as a queue administrator for the Engineering queue. The HDP administrator configured two leaf queues – Engineering.Dev and Engineering.QA. Does Sayad also have queue administrator privileges at the leaf queue level?

Yes – all settings of leaf queues are inherited from the parent queue. Sayad would be a queue administrator for both the Dev and QA leaf queues.

Summary

- HDP cluster resources are aggregated into a unit of work called a queue
- By default, only memory resources are considered when launching applications
- Queues can be finely controlled including
 - Dividing up resources among leaf queues and controlling elasticity and preemption settings
 - Controlling container, application, and user limits and reservations
 - Configuring ACLs and default queue mappings, with or without override enabled
 - Stopping queues for administrative purposes
 - Configuring queue administrators

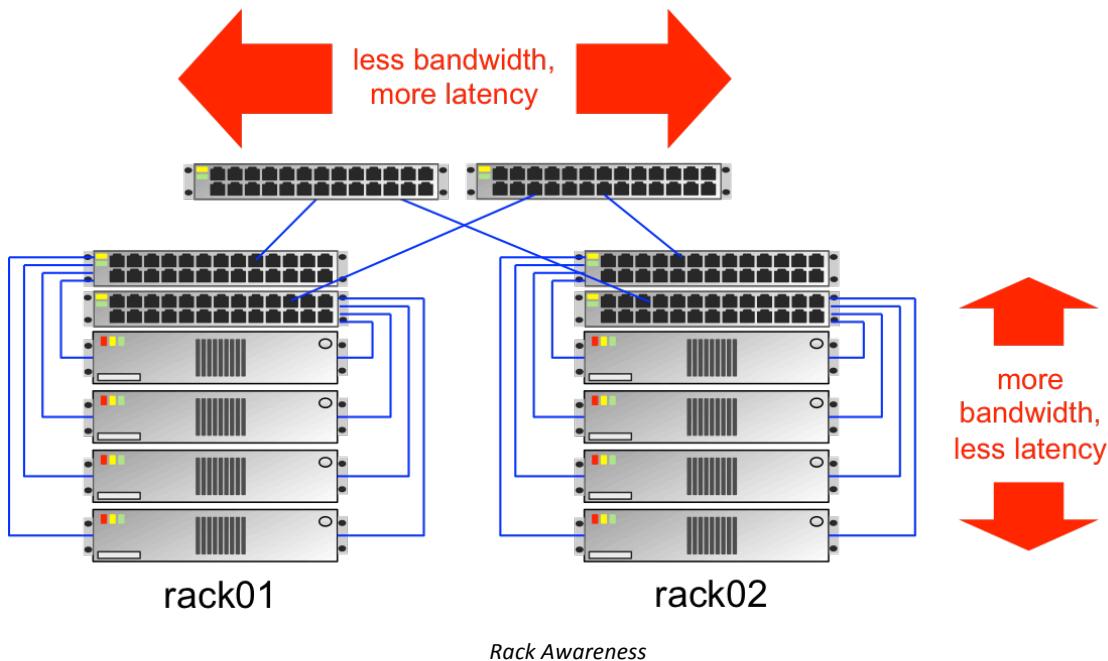
Configuring Rack Awareness

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Summarize the purpose and benefits of rack awareness
- ✓ Configure rack awareness

Why Rack Awareness



By default, HDFS and YARN are not rack aware. Rack awareness is defined as the knowledge of which nodes are installed on which racks in the data center.

Rack awareness must be manually configured by an administrator. Rack awareness is designed based on the assumption that there is more network bandwidth available and less network latency between the nodes within a rack than there is between nodes in different racks.

Benefits of Rack Awareness

The goals of rack awareness are greater HDFS data block availability and increased cluster performance.

In small clusters where there is a single rack, or no racks, a single or multiple switches might connect all nodes. Without rack awareness configured, HDFS and YARN are unaware of the network and rack topology and cannot make data and container placement decisions based on the network or rack topology. In regard to HDFS writes, the first data block replica will be written to one DataNode, while the remaining replicas will be written to two other DataNodes selected at random.

For larger cluster installations that span multiple racks, it is important to configure rack awareness to ensure that replicas of data blocks are placed on multiple racks. This ensures that the loss of a single rack switch or rack power does not render all data block replicas unavailable.

If HDFS is rack aware, it writes the first replica to a DataNode on one rack, while the second and third data blocks are written to DataNodes co-located in another rack. Replication is expedited because the second and third replicas are written to DataNodes within the same rack where network bandwidth is the highest and network latency is the lowest. HDFS operations that are rack aware include file creation and appends, the HDFS balancer, and the decommissioning of a DataNode.

Normally YARN attempts to co-locate containers on the same node as the required data block. This reduces network data transfers and enables the best application performance. If co-location is not possible, rack awareness enables YARN to create the container on another DataNode within the same rack as a DataNode with the required data block. This enables the data block information to read within a rack where network bandwidth is the highest and network latency is the lowest.

Configuring Rack Awareness

- Rack awareness is achieved in Ambari by:
 - An executable Python script: /etc/hadoop/conf/topology_script.py
 - A mapping file: /etc/hadoop/conf/topology_mapping.data
- The mapping file is updated by Ambari.

The screenshot shows the Ambari interface for managing a host named 'node1'. The 'Host Actions' dropdown menu is open, and the 'Set Rack' option is highlighted with a red box. Other options in the menu include 'Start All Components', 'Stop All Components', 'Restart All Components', 'Turn On Maintenance Mode', 'Delete Host', and 'Download Client Configs'.

Configuring Rack Awareness in Ambari

Configuring rack awareness was a manual process in HDP versions prior to version 2.3. An administrator had to use Ambari or an editor to update the `core-site.xml` file and add the `net.topology.script.file.name` property. This property is configured with the full path name to an executable script used to configure rack awareness. An administrator also had to write the rack awareness script.

In simple, hierarchical network topologies the rack awareness script could parse the IP addresses of the DataNodes in order to assign their rack names. For example in the address 10.x.y.z, the x component could be used to denote different data centers, the y component could be used to denote different racks in a data center, and the z component could be used to denote a specific DataNode within a rack. This assumes a very simple network topology where the cluster administrator has complete control over IP address assignment.

In more complex network topologies, or in environments where a cluster administrator does not have control over IP address assignments, it is typical to use a mapping file in conjunction with the script. The script reads the mapping file, which has been manually updated to map the IP addresses and host names of the DataNodes to their rack names.

Starting with HDP 2.3, configuring rack awareness is accomplished using Ambari, and a pre-configured script and mapping file. The `core-site.xml` file already specifies the Python script `/etc/hadoop/conf/topology_script.py`. This script references the mapping file `/etc/hadoop/conf/topology_mapping.data`, which is updated either manually by editing the file or by using Ambari. The NameNode(s) execute the script which reads the mapping file and assigns rack names to the DataNodes.

To assign a rack name to a node using the Ambari Web UI, browse to the **Hosts** page and select a node. With the node selected, select **Set Rack** from the **Host Actions** menu button.

The Set Rack Window



Ambari Set Rack Window

Rack names are hierarchical and resemble Linux path names. Rack names can be arbitrarily chosen by an administrator. They can include data center information, or not. What they must include are rack names that differentiate one rack from another in a data center.

The Set Rack window enables an administrator to easily set a rack location for a specific DataNode. The rack name may contain alphanumeric characters, periods, hyphens, or forward slashes. The rack name must begin with a forward slash. The rack name will be prepended to the DataNode name.

Example

For example, if the rack name is `/rack01`, and the DataNode name is `node3.west.com`, the full name of the node for rack awareness would be `/rack01/node3.west.com`. Rack name examples include:

- `/rack3/node1`
- `/datacenter/rack_09/host4.sales.com`
- `/london/rack06/node07.wet.internal`.

If rack awareness has not been configured, the default rack name for all DataNodes is `/default-rack`.

WARNING!

Prior to Ambari 2.1.2, manual edits to the `/etc/hadoop/conf/topology_mapping.data` file were overwritten by the information in the Ambari database with the HDFS service was restarted. Starting with Ambari 2.1.2, the file can be manually edited.

Checking for Rack Awareness

```
[root@node1 ~]# su - hdfs
[hdfs@node1 ~]$ hdfs fsck -racks
Connecting to namenode via http://node1:50070/fsck?ugi=hdfs&racks=1&path=%2F
FSCK started by hdfs (auth:SIMPLE) from /172.17.0.2 for path / at Tue Jul 21 13:01:43 EDT 2015
.....Status: HEALTHY
Total size: 457577414 B
Total dirs: 51
Total files: 28
Total symlinks: 0
Total blocks (validated): 26 (avg. block size 17599131 B)
Minimally replicated blocks: 26 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 3.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 3
Number of racks: 3
FSCK ended at Tue Jul 21 13:01:43 EDT 2015 in 9 milliseconds
```

The filesystem under path '/' is **HEALTHY**

[hdfs@node1 ~]\$ █

hdfs fsck -racks

The `hdfs fsck -racks` utility displays the number of racks of which the NameNode is aware. This can provide an administrator with an indication of whether or not rack awareness has been configured.

If the cluster is comprised of multiple DataNodes on multiple racks and rack awareness has been configured, the number of racks reported should be greater than one. By default, all DataNodes belong to the /default-rack when rack awareness is not configured, and this is reported by the `fsck` utility as one rack.

The `hdfs fsck <path> -files -blocks -locations -racks` command also prints rack names. For each file, the `fsck` command prints the file's block ID numbers and the locations of the blocks. The location includes the rack name and DataNode name or IP address.

Viewing Rack Names

```
[root@node1 ~]# su - hdfs
[hdfs@node1 ~]$ hdfs dfsadmin -report
Configured Capacity: 300000522240 (279.40 GB)
Present Capacity: 233279918080 (217.26 GB)
DFS Remaining: 231895834624 (215.97 GB)
DFS Used: 1384083456 (1.29 GB)
DFS Used%: 0.59%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-----
Live datanodes (3):

Name: 172.17.0.3:50010 (node2)
Hostname: node2
Rack: /rack02
Decommission Status : Normal
Configured Capacity: 100000174080 (93.13 GB)
DFS Used: 461365248 (439.99 MB)
Non DFS Used: 22240165888 (20.71 GB)
```

Verifying Rack Names

The `hdfs dfsadmin -report` utility displays the rack name for each DataNode. If rack awareness is not configured, the rack-name entry is omitted for each DataNode in the cluster.

For large clusters with many DataNodes, the output of this command could be lengthy. This might make it difficult to view the report on the screen. For this reason, the command output could be redirected to a file and then the file could be read using an editor.

Knowledge Check Questions

- 1) True or false? Rack awareness must be manually configured.
- 2) Rack awareness assumes that there is _____ bandwidth and _____ latency between nodes co-located within the same rack.
- 3) The two primary benefits of rack awareness are greater HDFS availability and _____.
- 4) By default in HDP 2.3, does rack awareness use a pre-configured script, or a pre-configured script and a mapping file?
- 5) What does a rack name of /default-rack indicate?
- 6) Which command displays the number of racks in a cluster?
- 7) Which command displays rack names in a cluster?

Knowledge Check Answers

- 1) True or false? Rack awareness must be manually configured.

True

- 2) Rack awareness assumes that there is **more** bandwidth and **less** latency between nodes co-located within the same rack.

- 3) The two primary benefits of rack awareness are greater HDFS availability and **increased cluster performance**.

- 4) By default in HDP 2.3, does rack awareness use a pre-configured script, or a pre-configured script and a mapping file?

A pre-configured scripts and a mapping file

- 5) What does a rack name of /default-rack indicate?

That rack awareness has not been configured

- 6) Which command displays the number of racks in a cluster?

hdfs fsck -racks

- 7) Which command displays rack names in a cluster?

hdfs dfsadmin -report or possibly hdfs fsck <path> -files -blocks -locations

Summary

- Rack awareness is when the NameNode and ResourceManager have knowledge of which nodes are installed on which racks in the data center.
- Rack awareness must be manually configured (commonly using Ambari).
- Rack awareness results in greater HDFS availability and increased cluster performance.
- Rack names are hierarchical, and the default rack name is /default-rack.
- The `hdfs fsck -racks` utility displays the number of racks of which the NameNode and ResourceManager are aware.
- The `hdfs dfsadmin -report` utility displays the rack name for each DataNode.
- The `hdfs fsck` with a `-locations` option prints a rack name along with a DataNode name.

Configuring HDFS and YARN High Availability (HA)

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Summarize the purpose and operation of NameNode HA
- ✓ Add additional ZooKeeper servers
- ✓ Configure NameNode HA using Ambari
- ✓ Summarize the purpose and operation of ResourceManager HA
- ✓ Configure ResourceManager HA using Ambari

NameNode HA

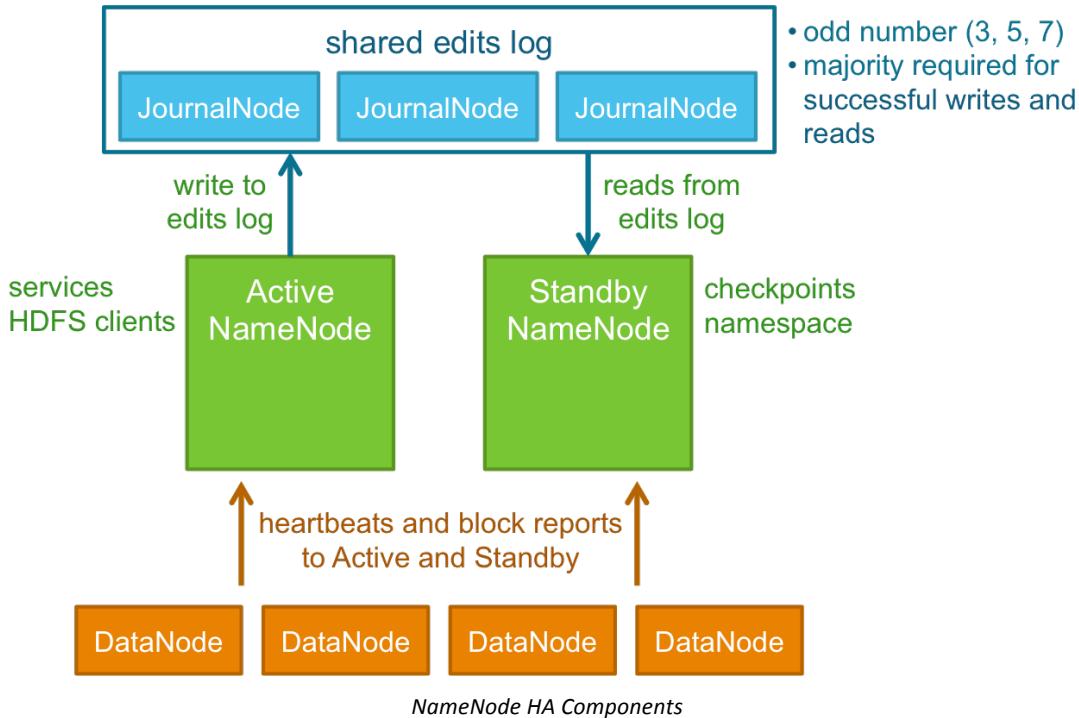
In Hadoop prior to version 2.0, the NameNode was a single point of failure. The entire cluster would become unavailable if the NameNode failed or became unreachable. Even maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.

The HDFS NameNode High Availability (HA) feature eliminates the NameNode as a single point of failure. It enables a cluster to run redundant NameNodes in an Active/Standby configuration.

NameNode HA enables fast failover to the Standby NameNode in response to a failure, or a graceful administrator-initiated failover for planned maintenance.

There are two ways of configuring NameNode HA. Using Ambari is the easiest way. Manually editing the configuration files and starting or restarting the necessary daemons is also possible. However, manual configuration of NameNode HA is not compatible with Ambari administration. Any manual edits to the `hdfs-site.xml` file would be overwritten by information in the Ambari database when the HDFS service is restarted.

NameNode HA Components



In a NameNode HA configuration there is an Active and a Standby NameNode. It is highly recommended that these NameNode machines have identical hardware configurations. The Active NameNode services HDFS client requests. The Standby NameNode performs namespace checkpoint operations. (In a NameNode HA configuration there is no need for a Secondary NameNode.)

To provide fast failover, it is necessary that the Standby NameNode have up-to-date data block location information. To maintain accurate information about the data block locations, DataNodes are configured with the locations of both NameNodes and send block reports and heartbeats to each.

In order for the Standby NameNode to keep its state synchronized with the Active NameNode, both NameNodes communicate with a group of separate daemons called JournalNodes. When the Active NameNode performs any namespace modification, it must durably log a modification record to a majority of the JournalNodes. The Standby NameNode continuously watches the JournalNodes for changes to the edits log. Once a Standby NameNode observes the edits, it applies these edits to its own in-memory namespace.

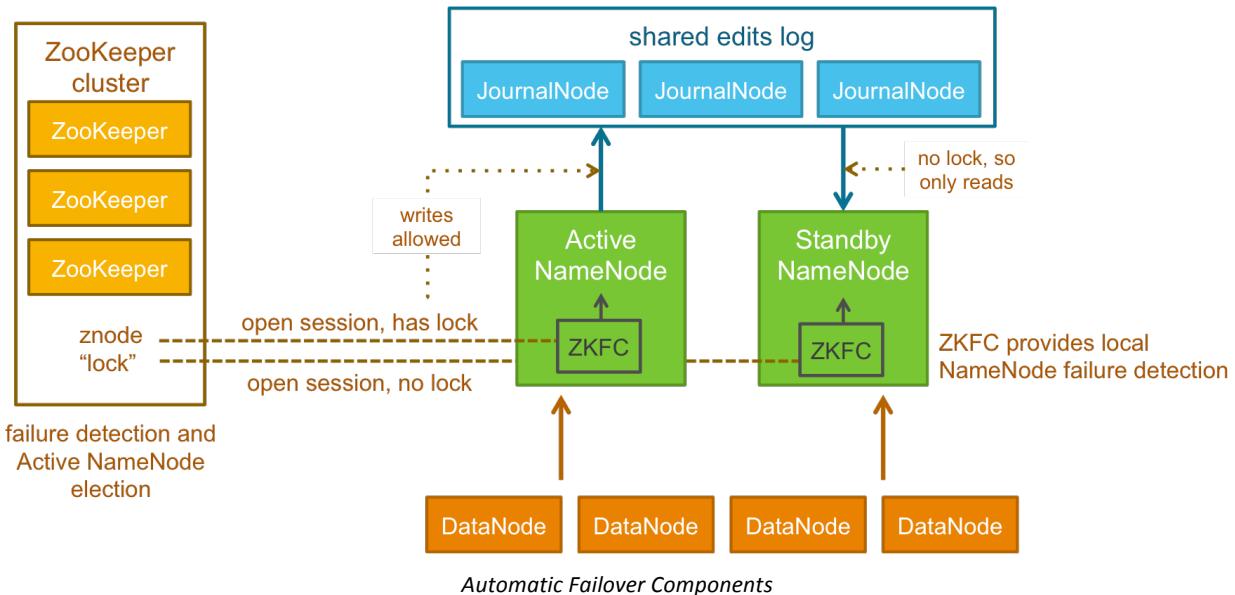
In a failover event, the Standby NameNode ensures that it has read all of the edits from the JournalNodes before promoting itself to the Active state. This mechanism ensures that the namespace state is fully synchronized before a failover completes.

It is vital for the correct operation of NameNode HA cluster that only one of the NameNodes should be Active at a time. Having two Active NameNodes would cause the namespace state on the two machines to quickly diverge and cause potential data loss. This situation is called a split-brain scenario.

To prevent the split-brain scenario, the JournalNodes allow only one NameNode to be a writer at a time. During failover, the NameNode, that is chosen to become Active, takes over the role of writing to the JournalNodes. This process prevents the other NameNode from continuing in the Active state and thus lets the new Active NameNode proceed with the failover safely.

Because JournalNodes operate on a majority, or quorum basis, an administrator must deploy an odd number of them. Three JournalNodes are common but five or seven can be configured if greater tolerance to failure is required. JournalNode daemons are relatively lightweight and can be co-located with other master or client service daemons.

Automatic Failover Components



Automatic failover requires two components: a ZooKeeper cluster (ensemble) and a ZooKeeper FailoverController (ZKFC). ZooKeeper is a highly available service for maintaining small amounts of coordination data, notifying clients of changes in that data, and monitoring clients for failures. A ZooKeeper cluster is typically comprised of three or five ZooKeeper servers. To enhance performance if the ZooKeeper servers are co-located on the Active or Standby NameNodes, Hortonworks recommends configuring the ZooKeeper servers such that the ZooKeeper data and the HDFS metadata are stored on separate disk drives. The ZKFC is a daemon that runs on each NameNode.

The ZKFC is a ZooKeeper client that monitors and manages the state of the NameNode daemon. The ZKFC pings its local NameNode on a periodic basis. So long as the NameNode responds, the ZKFC considers the NameNode healthy. If the NameNode has crashed, frozen, or otherwise entered an unhealthy state, the ZKFC will mark it as unhealthy.

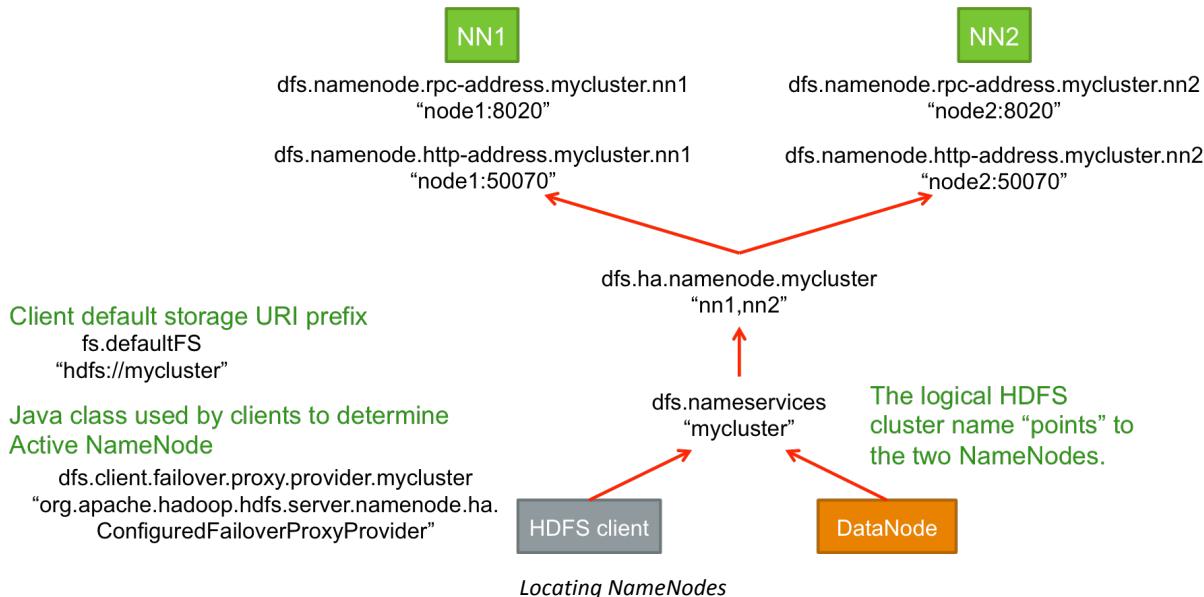
When the Active NameNode is healthy, its ZKFC holds a session open in the ZooKeeper cluster. The session maintains a “lock” on a ZooKeeper ephemeral znode. During NameNode failure, the session expires and the lock znode will be automatically deleted.

If the local NameNode is healthy, and its ZKFC detects that no other NameNode currently holds the lock znode, the ZKFC will try to acquire the lock. If it succeeds, then it is responsible for initiating a failover to make its local NameNode active. During the failover process, the previously Active NameNode is fenced as necessary, and then the new NameNode transitions to an active state.

For maintaining system correctness, it is important to have only one NameNode in the Active state at any given time. Only one NameNode will ever be allowed to write to the JournalNodes, so there is no potential for corrupting the HDFS file system metadata in a split-brain scenario. However when a failover occurs, it is still possible that the previous Active NameNode could serve read requests from clients, which might be out of date until that NameNode shuts down after trying to write to the JournalNodes.

For this reason, it is recommended to configure one or more fencing methods. For example, a fencing method could use SSH to log in to a NameNode machine and shut down the NameNode daemon. Fencing methods are configured as a comma-separated list in the `dfs.ha.fencing.methods` property in the `hdfs-site.xml` file. Any fencing methods used during a failover are attempted in order until one indicates that fencing has succeeded. The following two methods are packaged with Hadoop: shell and sshfence. Refer to the online documentation for more information about fencing methods and configuration.

Locating NameNodes



The HDFS clients and the DataNodes must be able to locate both NameNodes. Once NameNode HA has been configured, property settings in the `hdfs-site.xml` file provide the necessary connection information.

The HDFS service must be configured with a new nameservice ID. The nameservice ID is a logical name for the HDFS cluster. In the example here, the logical cluster name is configured by the `dfs.nameservices` property as the name `mycluster`.

The nameservice ID must be associated with the two NameNodes. To achieve this, the `dfs.ha.namenode.mycluster` property contains a comma-separated list with the strings `nn1` and `nn2`. These strings are used in other properties to identify how to connect to the actual NameNodes. These properties are illustrated here.

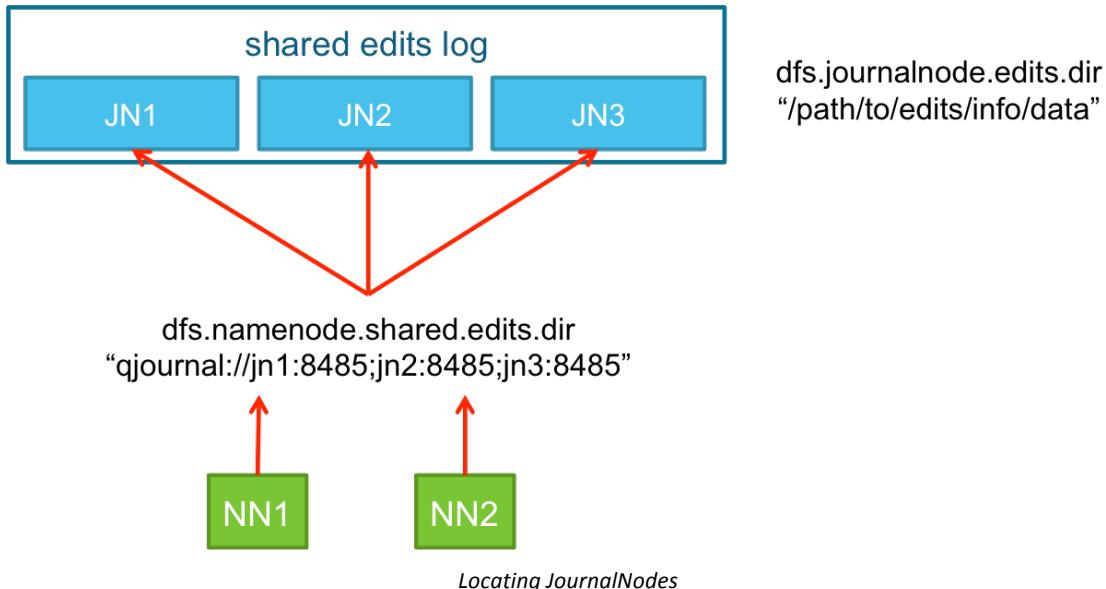
The actual NameNodes to connect to are defined in four different properties, and each property contains either the string `nn1` or `nn2`. These properties define the actual host names of the NameNodes along with the port numbers to use to connect using either RPC or HTTP.

The DataNodes regularly heartbeat and send data block reports to both NameNodes, regardless of which one is active. However, an HDFS client should only contact the Active NameNode. The `dfs.client.failover.proxy.provider.mycluster` property determines the Java class used by the client to determine which NameNode is currently the Active NameNode.

Note

While these are the primary NameNode HA configuration parameters, there are others. For a complete list, refer to the online documentation.

Locating JournalNodes



The Active NameNode writes edits log information to the JournalNodes. The Standby NameNode reads edits log information from the JournalNodes. In either case, the NameNodes must know how to connect to all of the configured JournalNodes. The `dfs.namenode.shared.edits.dir` property contains a semicolon-separated list of JournalNode host names and port numbers.

Recoverability Versus Availability

There are major differences between a Secondary NameNode and a Standby NameNode. One of the biggest differences is that a Standby NameNode enables high availability while a Secondary NameNode enables only recoverability.

In the event of NameNode failure or unreachability, the NameNode HA automatic failover process takes only seconds. Because the NameNodes runs in virtual lockstep, there is likely no metadata loss during failover.

It is different with a Secondary NameNode. In the event of a Primary NameNode failure, only manual recovery is possible. Manual recovery could take hours and it is likely that some metadata will be lost. The information shown here provides an overview of the steps in the recovery process.

ZooKeeper Prerequisites

Ambari NameNode Configuration Requires at Least three ZooKeeper Servers

Ambari will not allow an administrator to configure NameNode HA unless there are at least three ZooKeeper servers. If necessary, use Ambari Web UI > **Services** > **ZooKeeper** > **Service Actions** > **Add ZooKeeper Server** to add more ZooKeeper servers.

A dialog window opens (not shown here) that enables an administrator to select the node for the ZooKeeper server. Once the node has been selected the installation process starts.

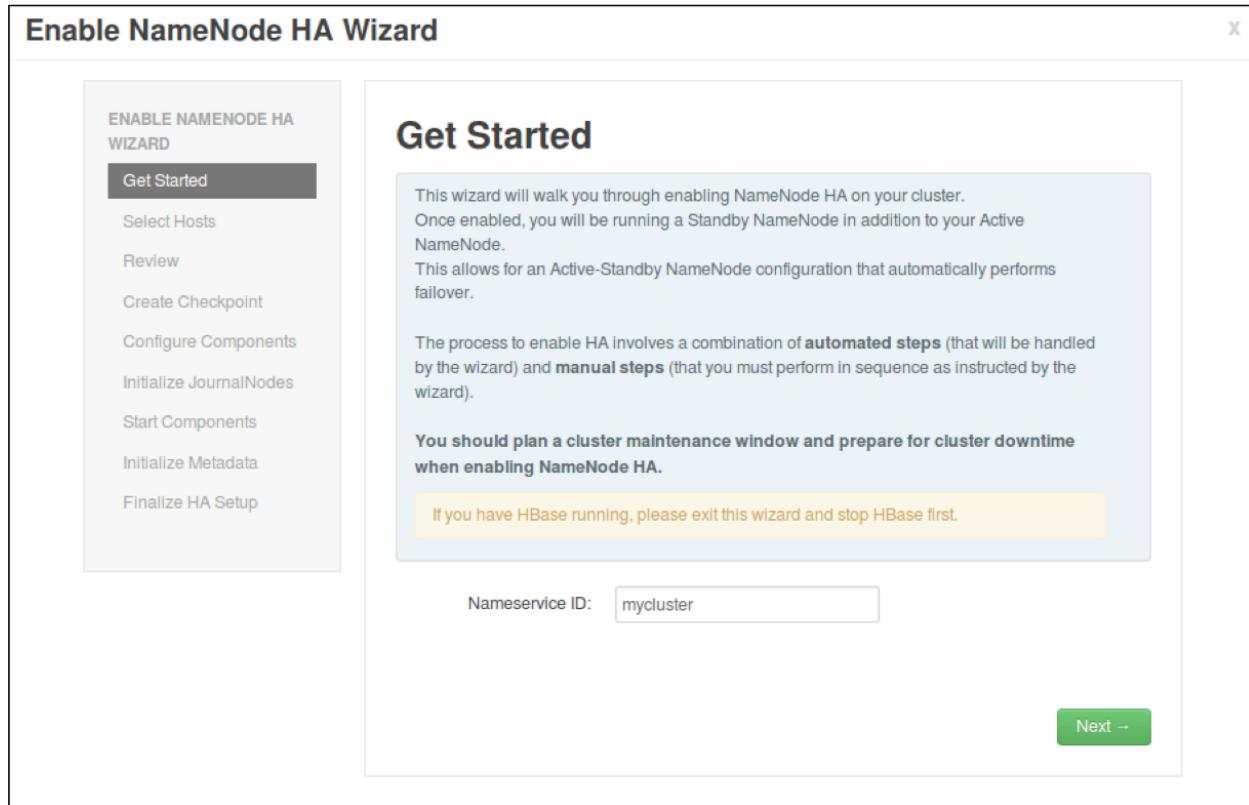
If additional ZooKeeper servers had to be installed, restart any affected services indicated in the Ambari Web UI.

Configuring NameNode HA Using Ambari

Enabling NameNode HA

To enable NameNode HA, in Ambari click **Services** > **HDFS** > **Service Actions** > **Enable NameNode HA**. This opens a configuration wizard. Plan for a few minutes of cluster down time. The amount of time depends on cluster size.

Configure the Nameservice ID

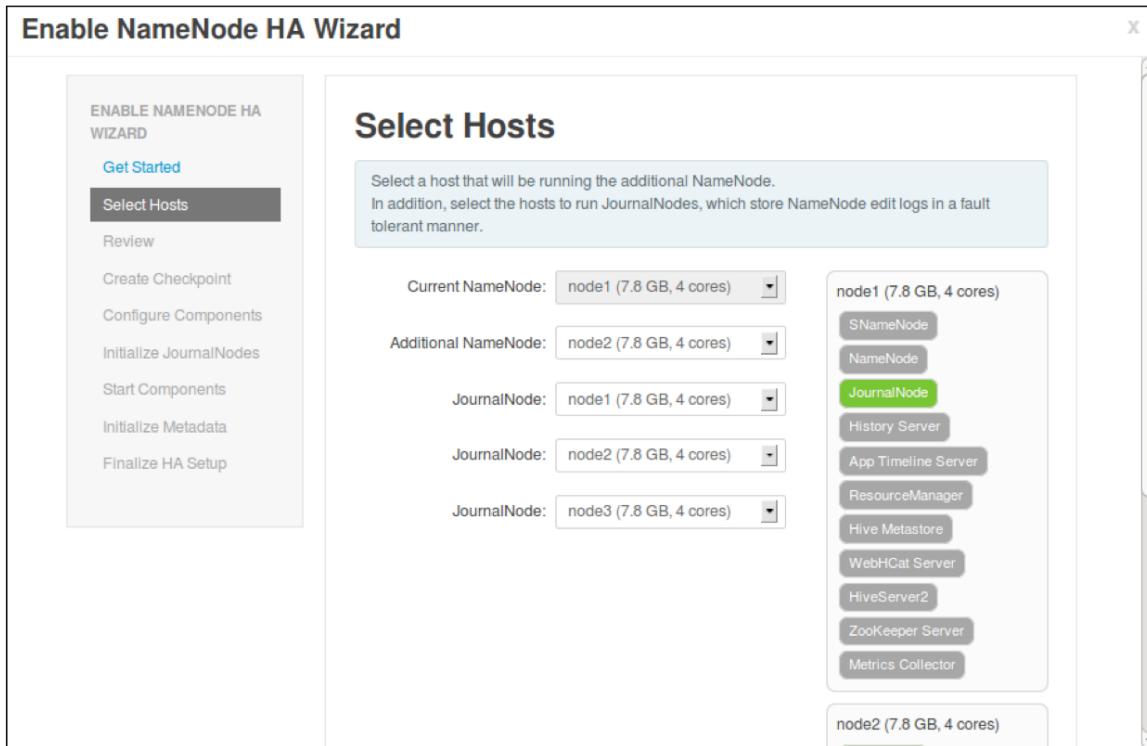


The NameNode HA Wizard

In the Getting Started window, type the Nameservice ID. The Nameservice ID is the logical name of the HDFS cluster.

Then click **Next**.

Select Host Locations

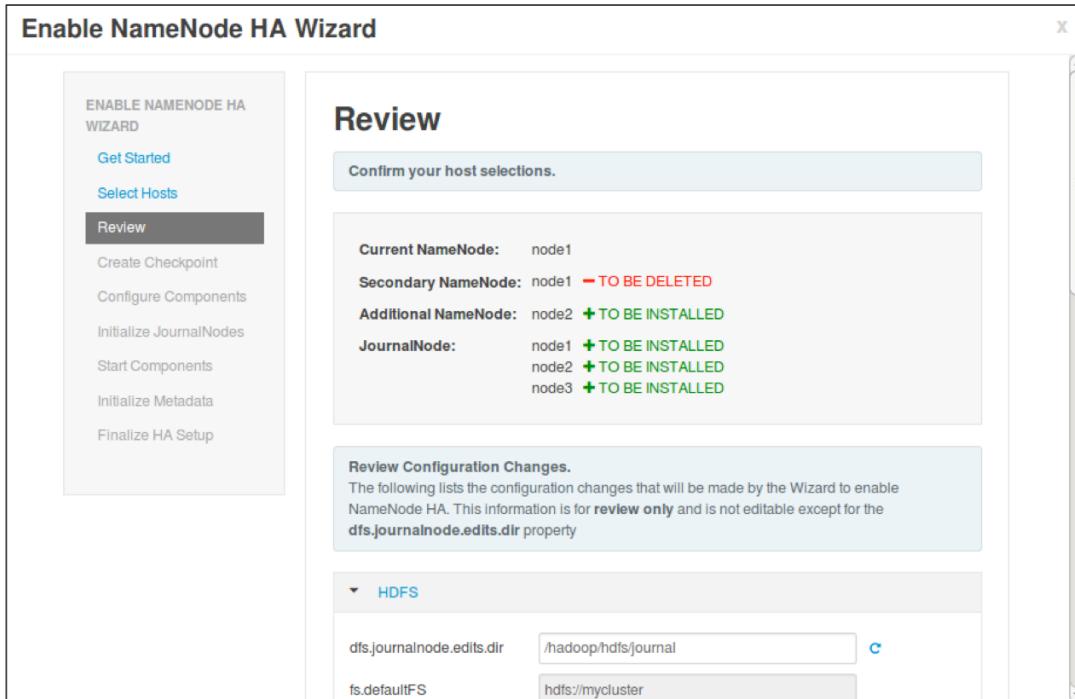


Select Hosts Window

The Select Hosts window enables an administrator to choose where the service daemons will run. The additional NameNode should run on an identical, but separate, host from the existing NameNode. The JournalNodes are lightweight and can run with other daemons on existing master or client machines.

Click **Next** to continue.

Review the Configuration



NameNode Wizard Review Window

Review and confirm the selections. While not entirely shown here, the properties and their settings that will be added to the `hdfs-site.xml` file are listed. Review and confirm those as well.

After reviewing and confirming your selections, click **Next**.

Required Manual Steps for NameNode

1. Login to the NameNode host **node1**.
 2. Put the NameNode in Safe Mode (read-only mode):
`sudo su hdfs -l -c 'hdfs dfsadmin -safemode enter'`
 3. Once in Safe Mode, create a Checkpoint:
`sudo su hdfs -l -c 'hdfs dfsadmin -savenameSpace'`
 4. You will be able to proceed once Ambari detects that the NameNode is in Safe Mode and the Checkpoint has been created successfully.

If the **Next** button is enabled before you run the "Step 4: Create a Checkpoint" command, it means there is a recent Checkpoint already and you may proceed without running the "Step 4: Create a Checkpoint" command.

Checkpoint not created yet **Next →**

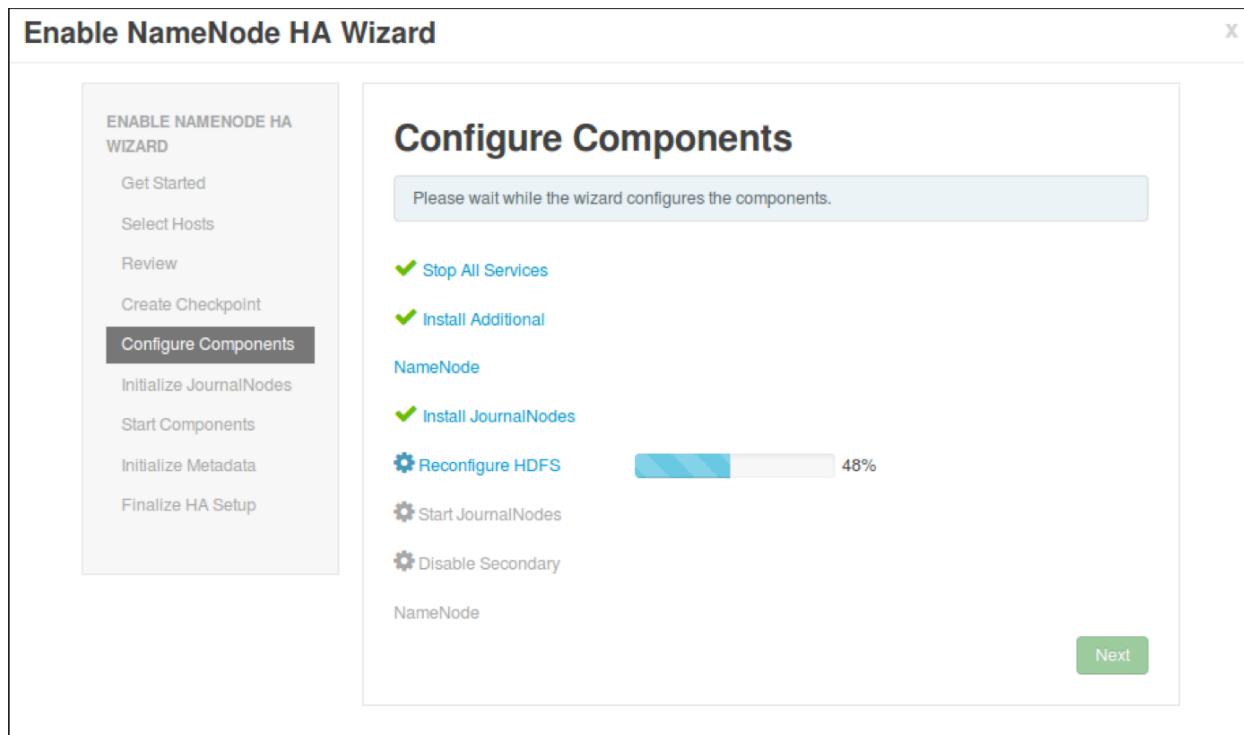
NameNode HA Manual Steps

Configuring NameNode HA requires running a few command-line utilities at various points in the process. The first manual steps are here and are required to checkpoint the existing NameNode. The checkpoint ensures that the `edits` and `fsimage` files are up-to-date before being transferred to the new Standby NameNode.

The **Next** button is not available until Ambari detects a successful checkpoint.

When the button becomes available, click **Next**.

Progress Window - Configuring Components



NameNode Progress Window

Monitor the progress window.

Then click **Next** to continue.

Required Manual Steps for JournalNodes

Enable NameNode HA Wizard

Manual Steps Required: Initialize JournalNodes

1. Login to the NameNode host **node1**.
2. Initialize the JournalNodes by running:
`sudo su hdfs -l -c 'hdfs namenode -initializeSharedEdits'`
3. You will be able to proceed once Ambari detects that the JournalNodes have been initialized successfully.

JournalNodes not initialized yet **Next →**

The **Next** button is unavailable until the JournalNodes are initialized.

JournalNodes Manual Steps

The JournalNodes must be manually initialized. Run the command displayed in the window.

Click **Next** once the button becomes available.

Progress Window - Starting Components

Enable NameNode HA Wizard

Start Components

Please wait while the wizard starts the components.

✓ Start ZooKeeper Servers	100%
⚙️ Start NameNode	9%

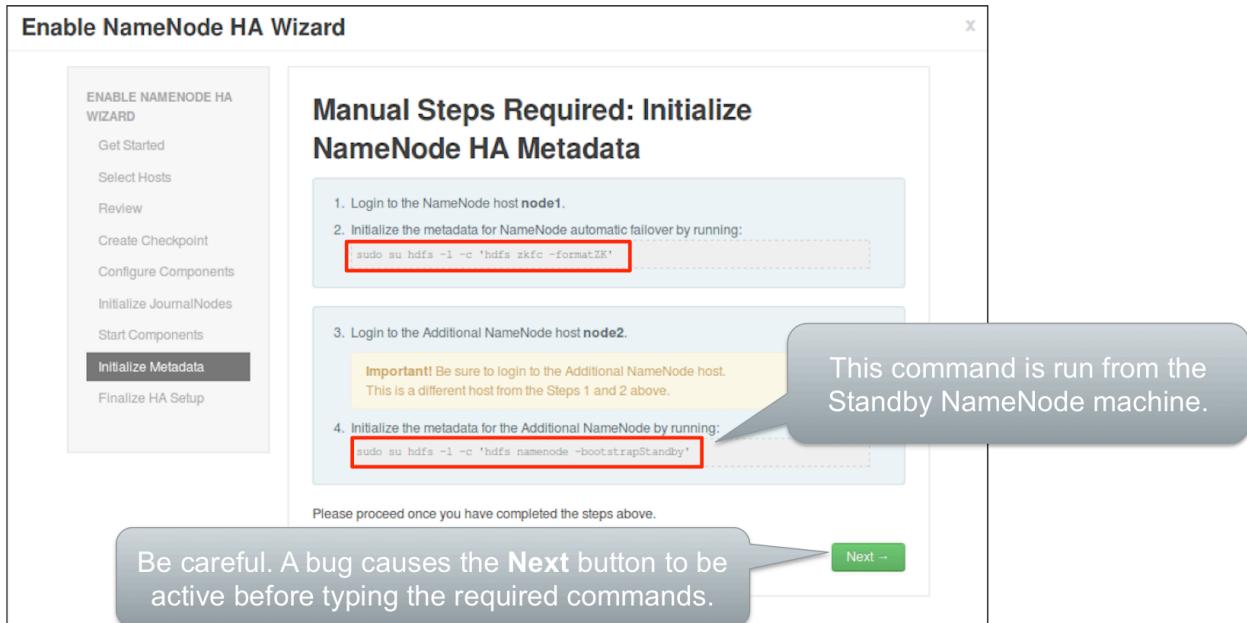
Next

Starting Components Progress Window

Monitor the progress window.

Click **Next** to continue.

Required Manual Steps for NameNode HA Metadata



This command is run from the Standby NameNode machine.

Initializing NameNode HA Metadata

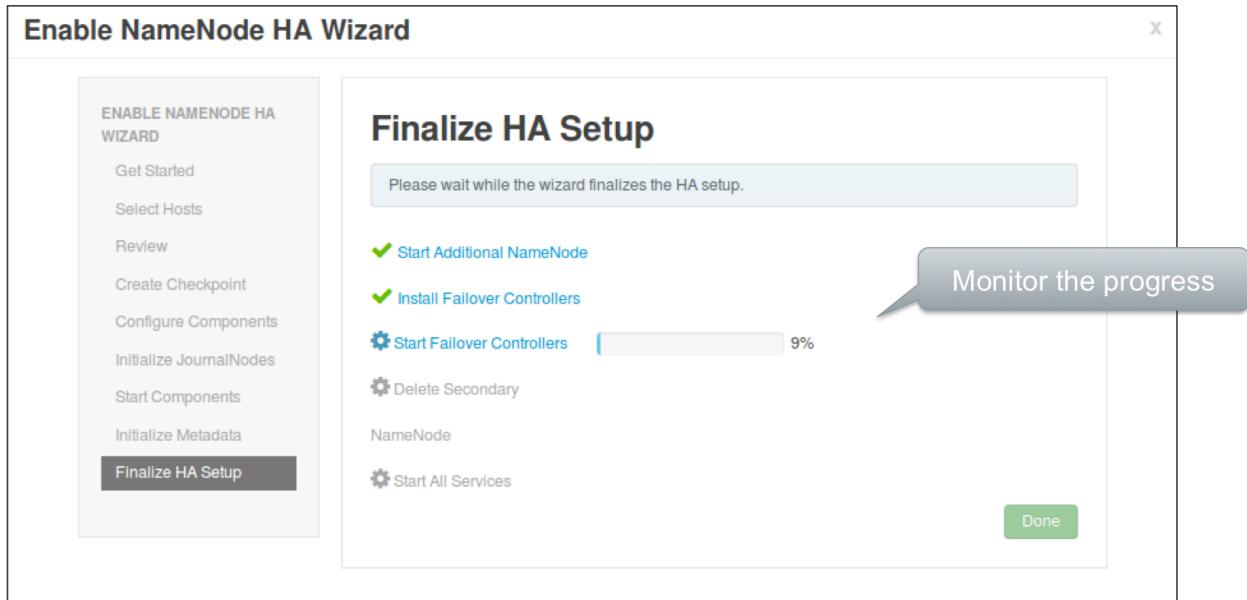
The final set of manual commands are necessary to configure the ZooKeeper monitoring service as well as the bootstrap the new NameNode.

WARNING

The second command must be run from the host running the Standby NameNode.

Click **Next** after running these commands.

Progress Window- Finalizing Setup



Finalizing NameNode HA Setup

Monitor the progress window.

Click **Next** to continue.

Verifying the Configuration

The screenshot shows the Ambari UI for a cluster named 'horton'. The left sidebar lists services: HDFS (green), MapReduce2, YARN, Tez, Hive, Pig, ZooKeeper, and Ambari Metrics. The main panel has tabs for Summary, Heatmaps, and Configs, with 'Summary' selected. Under 'Summary', it shows the status of various components: Active NameNode (Started), ZKFailoverController (Started), Standby NameNode (Started), ZKFailoverController (Started), DataNodes (3/3 Started), and JournalNodes (3/3 JournalNodes Live). The 'JournalNodes' section is highlighted with a red box. To the right, there are metrics: Disk Usage (Remaining) 215.3 GB / 279.4 GB (77.07%), Blocks (total) 26, Block Errors 0 corrupt / 0 missing / 0 under replicated, Total Files + Directories 82, Upgrade Status No pending upgrade, and Safe Mode Status Not in safe mode. A green box at the top right says 'No alerts'.

Verifying the Configuration

At the end of the process there should be an Active NameNode, a Standby NameNode, a ZKFailoverController for each NameNode, and three JournalNodes.

From the command line, you can get the service state by running `hdfs haadmin -getServiceState`.

Viewing Status Using the NameNode UI

The image shows two screenshots of the NameNode UI. Both screenshots have a red box around the browser's address bar, which displays 'node2:50070/dfshealth.html#tab-overview' and 'node1:50070/dfshealth.html#tab-overview' respectively. Below the address bar, there is a navigation bar with tabs: Hadoop, Overview, Datanodes, and Datanode Volum. The 'Overview' tab is selected. The main content area is titled 'Overview 'node2:8020' (active)' and 'Overview 'node1:8020' (standby)'. In both sections, there is a table with three rows: 'Namespace:' (containing 'mycluster'), 'Namenode ID:' (containing 'nn2' for node2 and 'nn1' for node1), and 'Started:' (containing the start time 'Wed Jul 22 19:54:12 EDT 2015'). Red boxes highlight the 'Namespace:' and 'Namenode ID:' fields in both tables.

The NameNode UI

The NameNode UI can be used to view the active or standby state of a NameNode. Use a browser to connect to the NameNode UI on each NameNode. The Web pages will display the current active or standby status along with the new Nameservice ID and NameNode ID.

Initiating a Failover

The screenshot shows two Ambari service summary pages. The left page displays the 'Summary' tab for the HDFS service, listing components like Active NameNode, Standby NameNode, ZKFailoverController, DataNodes, JournalNodes, and NFSGateways, all in Started status. The right page shows a detailed view for the 'Active NameNode' component under 'node1', with a dropdown menu open showing options: Restart, Stop (which is highlighted with a red box), Move, Turn On Maintenance Mode, Rebalance HDFS, and Started.

- To manually initiate failover:

```
-hdfs haadmin -failover
```

Administrator-Initiated Failover

To initiate a failover using Ambari, locate the Active NameNode and stop it. When you stop the Active NameNode, NameNode HA will fail the HDFS service over to the Standby NameNode.

To find the Active NameNode in Ambari, use **Services > HDFS** and look on the **Summary** tab. On the **Summary** tab, click **Active NameNode** to browse to the host running the Active NameNode daemon. Find the Active NameNode service in the service list and select **Stop** from its drop-down menu.

To manually initiate a failover, run the command `hdfs haadmin -failover`.

ResourceManager HA

In Hadoop prior to version 2.4, the ResourceManager was a single point of failure. The entire cluster would become unavailable if the ResourceManager failed or became unreachable. Even maintenance events such as software or hardware upgrades on the ResourceManager machine would result in periods of cluster downtime.

The YARN ResourceManager High Availability (HA) feature eliminates the ResourceManager as a single point of failure. It enables a cluster to run one or more ResourceManagers in an Active/Standby configuration.

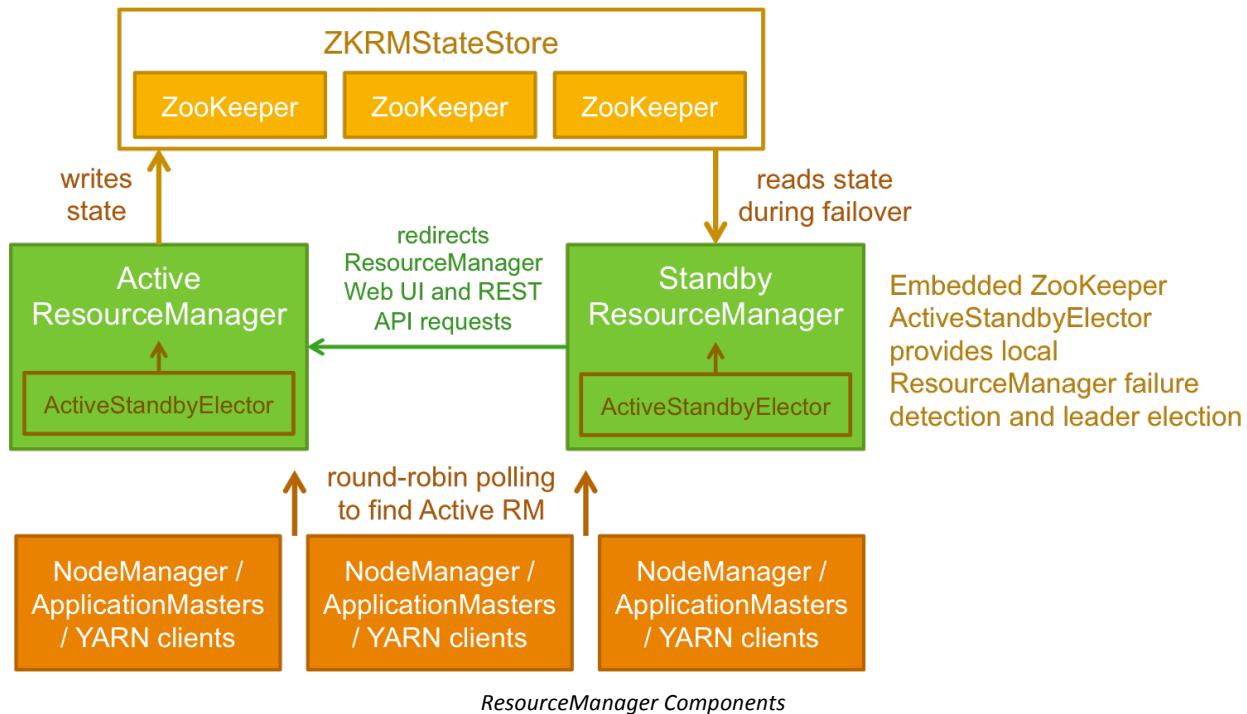
ResourceManager HA enables fast failover to the Standby ResourceManager in response to a failure, or a graceful administrator-initiated failover for planned maintenance.

There are two ways of configuring ResourceManager HA. Using Ambari is the easiest way.

WARNING!

Manually editing the configuration files and starting or restarting the necessary daemons is also possible. However, manual configuration of ResourceManager HA is not compatible with Ambari administration. Any manual edits to the `yarn-site.xml` file would be overwritten by information in the Ambari database when the YARN service is restarted.

ResourceManager Components



There is only a single Active ResourceManager at any time in the Hadoop cluster. All other ResourceManagers are in Standby mode. The `yarn-site.xml` file is configured with a list of all the ResourceManagers.

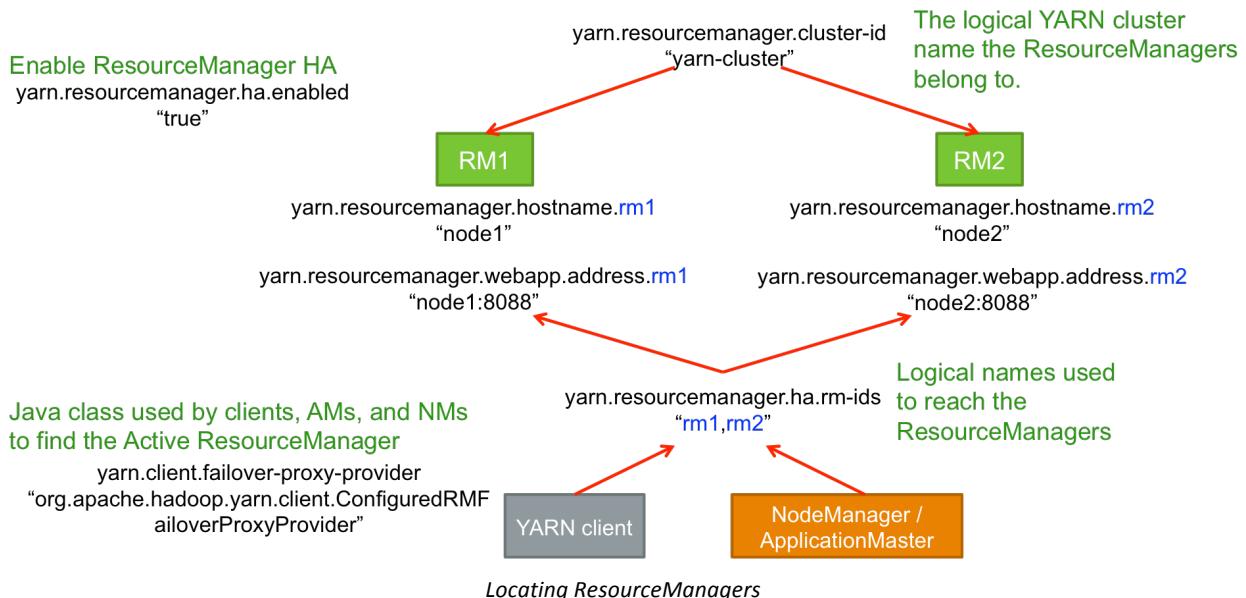
Using the `yarn-site.xml` configuration information, YARN clients, ApplicationMasters, and NodeManagers try connecting to the ResourceManagers in a round-robin fashion until they find the Active one. If the Active goes down, they resume the round-robin polling until they find the new Active.

The ResourceManagers have an embedded Zookeeper-based ActiveStandbyElector used to determine which ResourceManager should be Active. When the Active fails or becomes unresponsive, another ResourceManager is automatically elected to be the Active.

The ResourceManager being promoted to an active state loads the current state from the ZooKeeper state store and continues to operate from where the previous Active left off. A new attempt is spawned for each managed application previously submitted to the ResourceManager. Applications can checkpoint periodically to avoid losing any work during failure events.

Assuming that a Standby ResourceManager is up and running, the Standby automatically redirects all Web requests to the Active. This includes the ResourceManager Web UI and YARN REST API requests.

Locating ResourceManagers



The YARN clients, NodeManagers, and ApplicationMasters must be able to locate all ResourceManagers. Once ResourceManager HA has been configured, property settings in the `yarn-site.xml` file provide the necessary connection information.

The YARN service must be configured with a new cluster ID. The cluster ID is a logical name for the YARN cluster. In the example here, the logical cluster name is configured by the `yarn.resourcemanager.cluster-id` property as the name `yarn-cluster`. The cluster ID is used during the election process to ensure that a ResourceManager does not try to take over another YARN cluster.

The YARN clients, NodeManagers, and ApplicationMasters must be able to connect to the ResourceManagers. To achieve this, the `yarn.resourcemanager.ha.rm-ids` property contains a comma-separated list with the strings `rm1` and `rm2`. These strings are used in other property names to identify how to connect to the actual ResourceManagers. These properties are illustrated here.

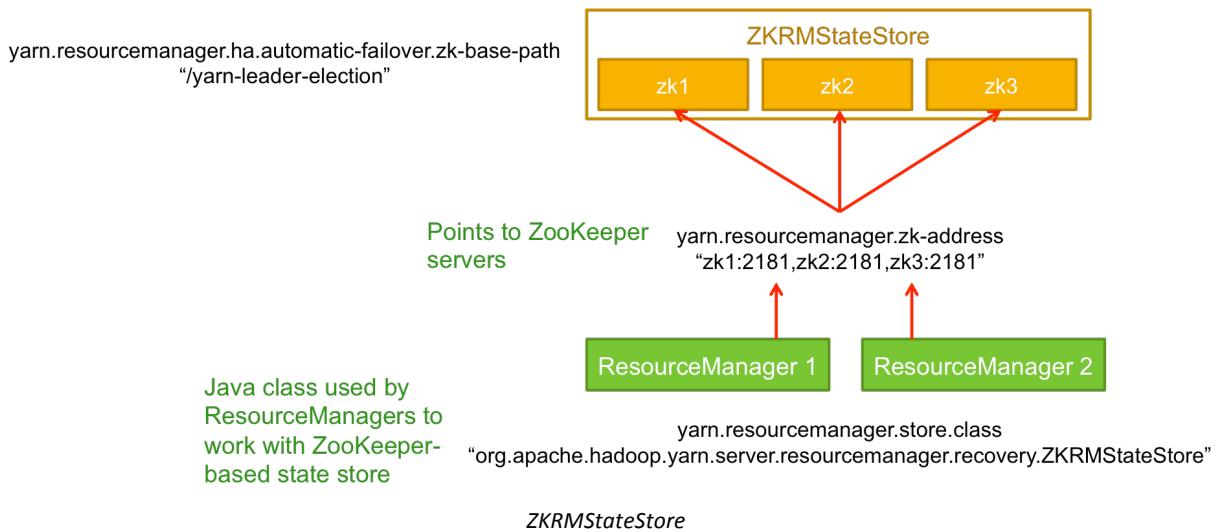
The actual ResourceManagers to connect to are defined in four different properties, and each property contains either the string `rm1` or `rm2`. These properties define the actual hostnames of the ResourceManagers along with the Web port number to use when connecting via HTTP.

The YARN clients, NodeManagers, and ApplicationMasters should only work with the Active ResourceManager. The `yarn.client.failover-proxy-provider` property determines the Java class used by these entities to determine which ResourceManager is currently the Active ResourceManager.

Note

While these are the primary ResourceManager HA configuration parameters, there are others. For a complete list, refer to the online documentation.

Working with the State Store



The Active ResourceManager continually writes job state information to the ZooKeeper-based state store named ZKRMStateStore. The Standby ResourceManager must be able to read this information during a failover. In either case, the ResourceManagers must know how to connect to all of the configured ZooKeeper servers.

The `yarn.resourcemanager.zk-address` property contains a comma-separated list of ZooKeeper hostnames and port numbers. It is used by the ResourceManagers to connect to the ZooKeeper-based state store.

The `yarn.resourcemanager.store.class` property determines the Java class that is used by the ResourceManagers to connect with and use the ZooKeeper-based ZKRMStateStore.

Configuring ResourceManager HA

The screenshot shows the Ambari web interface with the navigation bar at the top. The 'Services' tab is selected. On the left, the YARN service is highlighted in the tree view. The main panel displays the 'Summary' tab for the YARN service. The 'Service Actions' dropdown menu is open, showing various options including 'Enable ResourceManager HA'. This indicates that the user is in the process of enabling ResourceManager HA.

Enabling ResourceManager HA

To enable ResourceManager HA, in Ambari click **Services > YARN > Service Actions > Enable ResourceManager HA**. This opens a configuration wizard.

Enabling ResourceManager HA

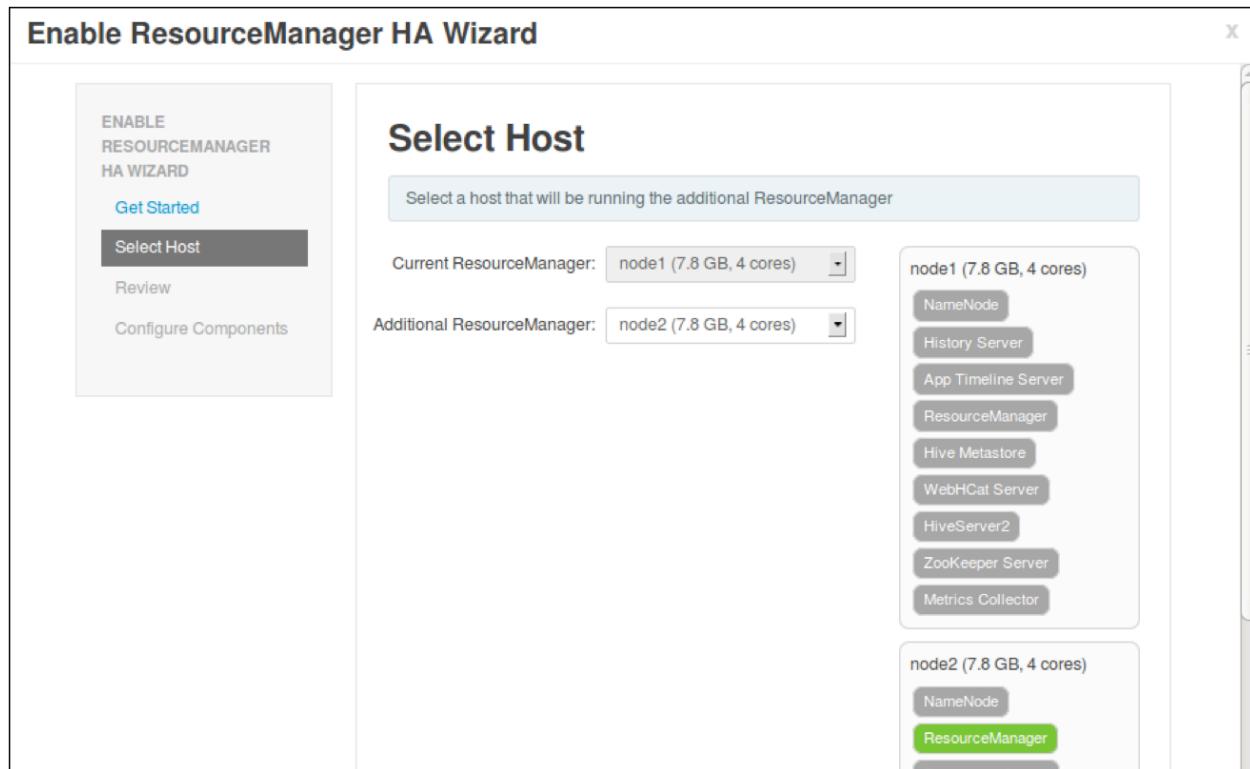
The screenshot shows the 'Enable ResourceManager HA Wizard' window. The left sidebar lists steps: 'Get Started' (selected), 'Select Host', 'Review', and 'Configure Components'. The main panel has a title 'Get Started' with a descriptive text box: 'This wizard will walk you through enabling ResourceManager HA on your cluster. Once enabled, you will be running a Standby ResourceManager in addition to your Active ResourceManager. This allows for an Active-Standby ResourceManager configuration that automatically performs failover.' Below this is a note: 'You should plan a cluster maintenance window and prepare for cluster downtime when enabling ResourceManager HA.' A large callout bubble points to this note with the text 'There will be minimal interruption.' A green 'Next →' button is visible on the right.

ResourceManager HA Wizard

Read the information and ensure that you have a maintenance window to complete the process. It only takes a few minutes to enable ResourceManager HA so there will be minimal downtime.

Click **Next** when ready to continue.

Select Host Location



Selecting ResourceManager Host Location

The Select Host window enables an administrator to choose where the service daemon will run. The additional ResourceManager should run on an identical, but separate, host from the existing ResourceManager.

After selecting a host, click **Next** to continue.

Review the Configuration

Review

Confirm your host selections.

Current ResourceManager: node1
Additional ResourceManager: node2 + TO BE INSTALLED

Review Configuration Changes.
The following lists the configuration changes that will be made by the Wizard to enable ResourceManager HA. This information is for **review only** and is not editable.

YARN

yarn.resourcemanager.ha.enabled	<input checked="" type="checkbox"/>
yarn.resourcemanager.ha.rm-ids	rm1.rm2

Reviewing the ResourceManager Configuration

Review and confirm the selections. While not entirely shown here, the properties and their settings that will be added to the `yarn-site.xml` file are listed. Review and confirm those as well.

After reviewing and confirming your selections, click **Next**.

Progress Window - Configuring Components

Configure Components

Please wait while ResourceManager HA is being deployed.

- ✓ Stop Required Services
- ✓ Install Additional ResourceManager
- ✓ Reconfigure YARN

Start All Services 8%

Complete

Monitoring Configuration Progress

Monitor the progress window.

Click **Next** to continue.

Verify the Configuration

The screenshot shows the Ambari dashboard for the 'horton' cluster. The YARN service is selected in the sidebar. The main summary page displays various cluster metrics. Two ResourceManager components are listed: 'Standby ResourceManager' and 'Active ResourceManager', both marked as 'Started'. A red box highlights the 'Active ResourceManager' entry. Other visible metrics include NodeManagers status (3 active), YARN Clients (3 installed), and ResourceManager Uptime (291.79 secs).

Verifying ResourceManager Configuration

At the end of the process there should be an Active ResourceManager and a Standby ResourceManager.

From the command line, you can get the service state by running `yarn rmadmin -getServiceState`.

Initiating a Failover

The left screenshot shows the Ambari dashboard with the YARN service selected. The 'Active ResourceManager' is highlighted with a red box. The right screenshot shows the detailed view for node 'node2', specifically the 'Components' tab. The 'Active ResourceManager / YARN' component is listed and its 'Stop' button is highlighted with a red box.

- WARNING!** With automatic failover enabled, manual failover may cause split brain or other incorrect states:

`- yarn rmadmin -forcemanual -transitionToStandby rml`

Administrator Initiated Failover

To initiate a failover using Ambari, locate the Active ResourceManager and stop it. When you stop the Active ResourceManager, ResourceManager HA will fail the YARN service over to the Standby ResourceManager.

To find the Active ResourceManager in Ambari, use **Services > YARN** and look on the **Summary** tab. On the **Summary** tab, click **Active ResourceManager** to browse to the host running the Active ResourceManager daemon. Find the Active ResourceManager service in the service list and select **Stop** from its drop-down menu.

Automatic failover is enabled by default and you should not use the manual transition command as it might cause a split-brain condition or other incorrect state. If you are very sure you know what you are doing, you can force a manual failover by specifying the `-forcemanual` option.

Knowledge Check Questions

- 1) NameNode HA enable fast failover and permits _____ failover for maintenance.
- 2) What NameNode HA function do the JournalNodes provide?
- 3) In NameNode HA, a DataNode sends its block reports to which NameNode?
- 4) True or false? Because there is no Secondary NameNode in NameNode HA, the Active NameNode must perform its own checkpoint operations.
- 5) Because a Standby NameNode does not own the lock in ZooKeeper, it is not allowed to _____ to shared edits log.
- 6) The DataNodes and HDFS clients use a _____ HDFS cluster name to access the two NameNodes.
- 7) A Standby NameNode enables availability while a Secondary NameNode only enables _____ .
- 8) True or false? ResourceManager HA support two or more ResourceManagers.
- 9) NodeManagers, ApplicationMasters, and YARN clients use _____ polling to find the Active ResourceManager.
- 10) Current job information and state is stored in a _____ -based state store.
- 11) What is the name of the embedded ZooKeeper client that provides failure detection and leader election?
- 12) True or false? Ambari automatically configures automatic failover.
- 13) True or false? Initiating manual failover in an automatic failover configuration might cause split-brain or other incorrect states.

Knowledge Check Answers

- 1) NameNode HA enable fast failover and permits ***Administrator initiated*** failover for maintenance.
- 2) What NameNode HA function do the JournalNodes provide?
They provide shared edits log files to the two NameNodes.
- 3) In NameNode HA, a DataNode sends its block reports to which NameNode?
To both NameNodes
- 4) True or false? Because there is no Secondary NameNode in NameNode HA, the Active NameNode must perform its own checkpoint operations.
False. The Standby NameNode performs the checkpoint operations.
- 5) Because a Standby NameNode does not own the lock in ZooKeeper, it is not allowed to ***_write_*** to shared edits log.
- 6) The DataNodes and HDFS clients use a ***_logical_*** HDFS cluster name to access the two NameNodes.
- 7) A Standby NameNode enables availability while a Secondary NameNode only enables ***_recoverability_***.
- 8) True or false? ResourceManager HA support two or more ResourceManagers.
True
- 9) NodeManagers, ApplicationMasters, and YARN clients use ***_round-robin_*** polling to find the Active ResourceManager.
- 10) Current job information and state is stored in a ***_ZooKeeper_***-based state store.
- 11) What is the name of the embedded ZooKeeper client that provides failure detection and leader election?
ActiveStandbyElector
- 12) True or false? Ambari automatically configures automatic failover.
True
- 13) True or false? Initiating manual failover in an automatic failover configuration might cause split-brain or other incorrect states.
True

Summary

- The purpose of NameNode HA is to eliminate the NameNode as a single point of failure.
- The purpose of ResourceManager HA is to eliminate the ResourceManager as a single point of failure.
- NameNode and Resource HA feature automatic failover from an Active node to a Standby node.
- Both NameNode and ResourceManager HA require at least three ZooKeeper servers.
- JournalNodes are used to provide a shared edits log from the redundant NameNodes.
- ZooKeeper is used to provide failover coordination for both NameNode an ResourceManager HA .

Monitoring a Cluster

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Summarize the purpose and operation of Ambari Metrics
- ✓ Summarize the features and benefits of the Ambari Dashboard
- ✓ Summarize the purpose and operation of Ambari Alerts
- ✓ Add and Configure an Ambari alerts

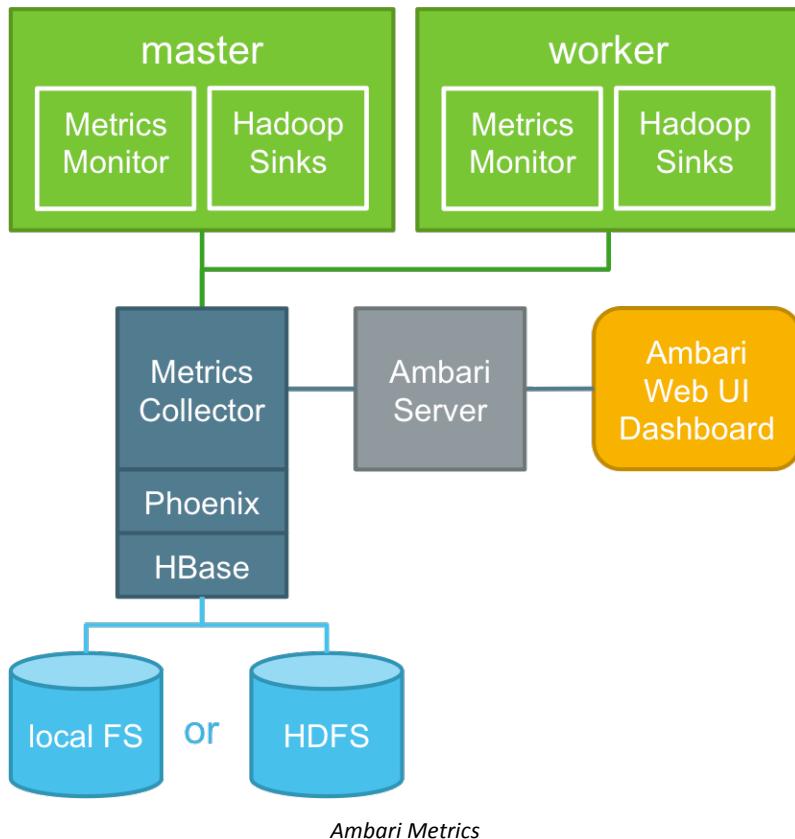
Ambari Metrics

Ambari Metrics System (AMS) is a built-in metrics collection system for Ambari. It collects host, cluster, service, and service component information. Starting with Ambari 2.0 and HDP 2.3, Ambari Metrics has replaced Ganglia as the cluster monitoring system.

Ambari Metrics is comprised of three primary components:

- Metrics Monitor
- Metrics Hadoop Sinks
- Metrics Collector

Ambri Metrics Components



Ambri Metrics is made up of three primary components.

- **Metrics Monitor**
Runs on each master and worker node in the cluster. It collects host health status and metric information and forwards it to the Metrics Collector.
- **Metrics Hadoop Sink**
Runs on each master and worker node in the cluster. It plugs into various service components and collects metric information and forwards it to the Metrics Collector.
- **Metrics Collector**
Runs on a single system and receives information from the Metrics Monitors and Hadoop Sinks. The Collector leverages Phoenix and an HBase daemon to store metric and health data. In embedded mode the HBase daemon uses the local file system. In distributed mode the HBase daemon uses HDFS. Embedded mode is the default installation.

The Metrics Collector sends information to the Ambari Server. Ambari Metrics information is displayed in the Ambari Web UI.

Switching to Distributed Mode

There are three steps required to switch from embedded mode to distributed mode.

- 1) In the Ambari Web UI, browse to **Services > Ambari Metrics > Configs > General** and change **Metrics Service operation mode** from `embedded` to `distributed`. This updates the `ams-site.xml` file.

Monitoring a Cluster

Metrics Service operation mode distributed

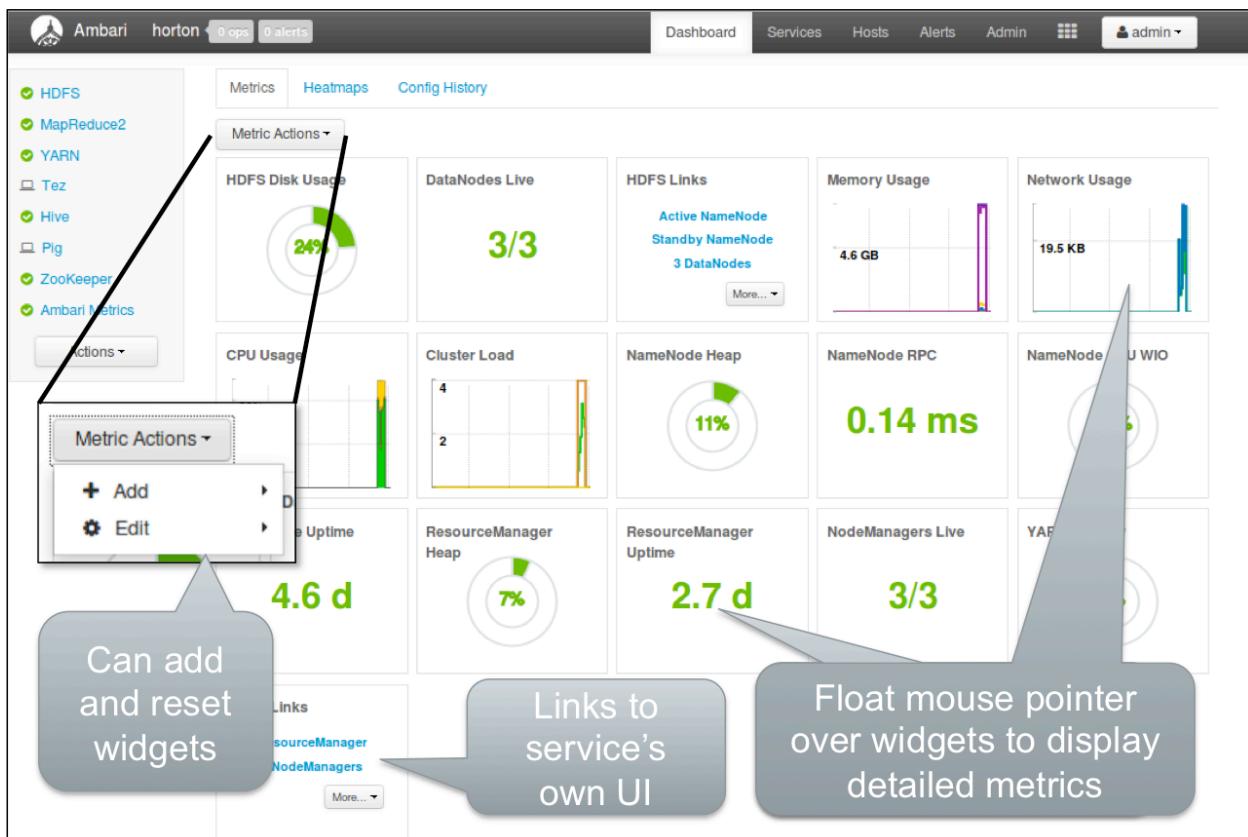
- 2) Also in the Ambari Web UI, browse to **Services > Ambari Metrics > Configs > Advanced ams-hbase-site** and change **hbase.rootdir** from `file:///var/lib/ambari-metrics-collector/hbase` to `hdfs://<namenode>:8020/amsdirectory`. Replace `<namenode>` with the actual HDFS NameNode host name and `amsdirectory` with the HDFS directory that will be used to store the Ambari Metrics data. The updates the `ams-hbase-site.xml` file.

hbase.rootdir `hdfs://namenode_hostname:8020/amsdirectory`

- 3) Restart the Ambari Metrics service using the Ambari Web UI.
Services > Ambari Metrics > Summary > Service Actions > Restart All

The Ambari Metrics configuration files are installed by Ambari in the directories `/etc/ambari-metrics-collector/conf` and `/etc/ambari-metrics-monitor/conf`.

Ambari Dashboard



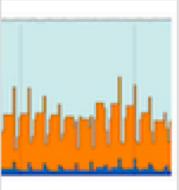
The Ambari Dashboard

The Ambari Web UI displays the **Dashboard** page as its home page. Administrators use the Dashboard to view the operating status of their cluster. Each Ambari Metrics widget displays status information for a cluster. By default, the Dashboard displays cluster-wide metrics as well as HDFS, YARN, HBase, Storm, and Ambari Metrics service metrics.

Status information appears as simple pie charts, bar charts, complex charts showing usage and load, and single values representing operating parameters such as uptime and average RPC queue wait times. Some widgets feature sets of links to additional service UIs like the HDFS NameNode UI or the YARN ResourceManager UI.

The number and types of widgets displayed depends on the installed services. An administrator can add and remove individual widgets, and rearrange the Dashboard by dragging and dropping each widget to a new location in the Dashboard. Floating the mouse pointer over a widget displays an X icon that can be used to delete the widget from the Dashboard. Select **Metric Actions > Add** to add a widget back to the Dashboard. Select **Metric Actions > Edit** to reset all widgets to their defaults.

Types of Widgets

	Gauge A view to display metrics that can be expressed in percentage.		Number A view to display metrics that can be expressed as a single number with optional unit field.
	Graph A view to display metrics that can be expressed in line graph or area graph over a time range.		Template A view to display metric value along with a templated text.

Four Types of Ambari Widgets

Monitoring a Cluster

Per-Service Widgets

The screenshot shows the Ambari interface for monitoring a YARN cluster. On the left, a sidebar lists services: HDFS, MapReduce2, YARN (selected), Tez, Hive, Pig, ZooKeeper, and Ambari Metrics. A red box highlights the 'Metrics' section of the main content area. This section displays various performance metrics in a grid format, including Memory Utilization, CPU Utilization, Container Failures, App Failures, Cluster Memory (50%), Cluster Disk (100 Mbps), Cluster Network (40), and Cluster CPU (20%). To the right of the metrics is a dropdown menu for time periods: Last 1 hour, Last 2 hours, Last 4 hours, Last 12 hours, Last 24 hours, Last 1 week, Last 1 month, and Last 1 year. A large gray callout bubble on the left side of the metrics section says 'Customizable service dashboard'.

Per-Service Widgets

Additional widgets are available for various services on their **Services** page. For example, HDFS, YARN, HBase, Storm, and Ambari Metrics all have widgets on their **Services** page. The example here is for YARN.

Metrics displayed in the widgets can be filtered by time. Metrics can be displayed for time periods ranging anywhere from the last hour up to the last year.

The **Actions** menu button enables an administrator to use a wizard to add custom widgets to a service dashboard. The wizard includes the ability to set warning and critical alarms in the widgets. Alarms are described later in this lesson.

Ambari Alerts

Ambari Alerts was added starting with Ambari 2.0. It replaced Nagios as the cluster alerting system. Alerts are useful for identifying and troubleshooting problems.

Ambari Alerts monitors:

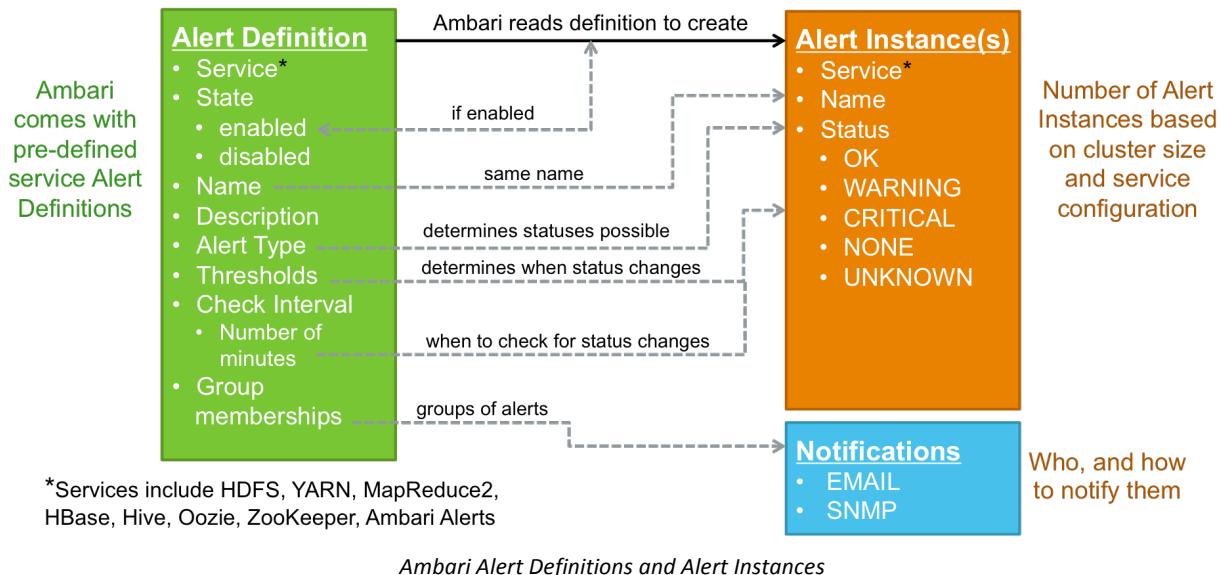
- Cluster hosts
- Cluster
- Cluster services
- Cluster service components

Ambari Alerts can trigger:

- Visual Ambari Web UI alarms
- Email notifications
- SNMP traps

Email and SNMP notifications can be customized to notify different users for different alarms at different severity levels.

Alert Definitions and Alert Instances



Ambari predefines a set of Alert Definitions designed to monitor the cluster and hosts. Ambari uses an Alert Definition to create Alert Instances that perform the actual service component or host checks. A single Alert Definition might be used to create one or hundreds of Alert Instances.

For example, if there were 60 DataNodes, then the Datanode process Alert Definition would be used to create 60 DataNode process Alert Instances. The number of Alert Instances also depends on the services installed in the cluster. For example, if HBase is not installed there would not be any HBase Alert Instances based on the HBase Alert Definitions. The status of such an alert would be UNKNOWN.

Each Alert Definition includes several attributes:

- The service name being monitored. Services include HDFS, YARN, MapReduce2, HBase, Hive, Oozie, ZooKeeper, and Ambari Alerts.
- The state of the Alert Definition. It can be enabled or disabled. When a cluster is created or modified, Ambari reads the Alert Definitions and creates Alert Instances for the specific components to watch. If an Alert Definition is enabled, Ambari creates Alert Instances for each appropriate component or host. If an Alert Definition is disabled, all its associated Alert Instances are disabled.
- The pre-defined name of the Alert Definition. The name of each Alert Instance is the same as its Alert Definition.
- A brief description of the Alert Definition.
- There are five Alert Definition types that include PORT, METRIC, AGGREGATE, WEB, and SCRIPT. The characteristics of each are described later in this lesson.

- One or more thresholds. Alert Definition thresholds determine the status of an Alert Instance. For example, two different thresholds might determine when the status of an Alert Instance might transition from OK to WARNING, or from WARNING to CRITICAL.
- The check interval defines how often, in minutes, Ambari checks the status of an alert.
- Alert Definition groups enable an administrator to create groups of alerts and configure notifications for each group. This determines how, and to whom, alerts are sent. Notifications are sent by either email or SNMP when an Alert Instance transitions from one status to another. There are no configured notifications, by default.

Alert Types

Type	Description	Possible Statuses	Configurable Thresholds	Threshold Units
PORT	Monitors a network port for response. Example: DataNode Process	OK, WARNING, CRITICAL	yes	seconds
METRIC	Monitors a quantifiable measure. Example: NameNode Host CPU Utilization	OK, WARNING, CRITICAL	yes	variable
AGGREGATE	Aggregate of the statuses of another set of alerts. Example: Percent DataNodes With Available Space	OK, WARNING, CRITICAL	yes	percentage
WEB	Monitors a Web UI for response. Example: NameNode Web UI	OK, WARNING, CRITICAL	no	n/a
SCRIPT	Uses a custom script to monitor/check a measurable unit. Example: NodeManager Health Summary	OK, CRITICAL	no	n/a

Ambari Alert Types

The table lists and describes the characteristics of the five alert types. Only PORT, METRIC, and AGGREGATE alerts have configurable WARNING and CRITICAL thresholds. They are configurable in the Ambari Web UI when editing an Alert Definition.

PORT alerts transition from a status of OK to either WARNING or CRITICAL based on the number of seconds it takes to receive a response from a port. The response time is checked at the configured check interval.

METRIC alerts transition from a status of OK to either WARNING or CRITICAL based on a change in a measurement unit. Measurement units can be such things as a percentage, time, the number of HDFS directories, the number of DataNodes, and so on. Whatever the measurement unit is, it is checked at the configured check interval.

AGGREGATE alerts transition from a status of OK to either WARNING or CRITICAL based on a percentage measured during the most recent check at the configured check interval.

WEB and **SCRIPT** alerts do not have configurable thresholds in the Ambari Web UI.

For **WEB** alerts, the alert attempts to access a URL and the alert status is determined based on the HTTP response. The status is OK if the HTTP response is less than 400. The status is WARNING if the response is 400 or above. The status is CRITICAL when the alert cannot connect to the URL. Because the status depends on a non-configurable HTTP response, the thresholds cannot be changed. However, the text message displayed in the Response column can be modified.

For SCRIPT alerts, the alert executes a script and the script determines status of OK or CRITICAL. The thresholds and response text message are built-into the script so neither can be modified in the Ambari Web UI. This is why the descriptions of SCRIPT alerts include information about their thresholds, because this information is not visible otherwise.

Viewing Alert Definitions

Alert Definition Name	Status	Service	Last Status Changed	State
Metrics Collector - ZooKeeper Server Process	CRIT	ZooKeeper	5 hours ago	Enabled
Metrics Collector - HBase Master Process	CRIT	HDFS	5 hours ago	Enabled
Metrics Monitor Status	OK (3)	YARN	5 days ago	Enabled
NodeManager Health	OK (3)	YARN	5 days ago	Enabled
NodeManager Web UI	OK (3)	ZooKeeper	5 days ago	Enabled
ZooKeeper Server Process	OK (3)	HDFS	7 days ago	Enabled
JournalNode Process	OK (3)	HDFS	7 days ago	Enabled
DataNode Process	OK (3)	HDFS	7 days ago	Enabled
DataNode Web UI	OK (3)	HDFS	7 days ago	Enabled
DataNode Storage	OK (3)	HDFS	7 days ago	Enabled

Viewing Alert Definitions

Starting with Ambari 2.0, there is a new **Alerts** page in the Ambari Web UI. It displays all the Alert Definitions.

Each Alert Definition can have one or more Alert Instances. Which Alert Definitions result in actual Alert Instances depends on the installed services. How many Alert Instances are created depends on the cluster's size and service configuration. Click an **Alert Definition** name to view or edit the Alert Definition.

If any Alert Instance has transitioned to a WARNING or CRITICAL status, the Status column will display WARN or CRIT for that Alert Definition.

The State column enables an administrator to disable or enable all Alert Instances associated with a particular Alert Definition. Click the current state to toggle between enabled and disabled.

Editing an Alert Definition

The screenshot shows the Ambari interface for managing cluster alerts. The top navigation bar includes links for Dashboard, Services, Hosts (with a count of 1), Alerts, Admin, and a user icon for admin. The main content area is titled 'Metrics Collector - ZooKeeper Server Process'. It displays a configuration section with a 'Description' field containing a placeholder message about a host-level alert for the Metrics Collector's ZooKeeper server process. Below this is an 'Interval' field set to '1 Minute'. The 'Thresholds' section contains three rows: 'OK' (green) with the template 'TCP OK - {0:.3f}s response on port {1}', 'WARNING' (orange) with the template 'TCP OK - {0:.3f}s response on port {1}' and a value of '1.5 Seconds', and 'CRITICAL' (red) with the template 'Connection failed: {0} to {1}:{2}' and a value of '5 Seconds'. To the right of the thresholds, there is a summary of alert details: State (Enabled), Service (Ambari Metrics), Component (Metrics Collector), Type (PORT), Groups (AMBARI_METRICS Default), Last Change (Wed, Jul 29, 2015 08:49), and Changed (empty). A large callout bubble points to the 'Edit' button with the text 'Click to modify the configuration'. Another callout bubble points to the 'CRITICAL' threshold row with the text 'Response' text can be modified too. A third callout bubble points to the bottom right with the text Click Save button (not shown).

Editing an Alert Definition

While viewing an Alerts Definition, click **Edit** to modify its configuration. You can modify the check Interval, the WARNING and CRITICAL thresholds, and enable or disable all Alert Instances. You can also modify the text message to be displayed in the Response column.

In the example, the alert will check the response time from the ZooKeeper Service port every minute. If the response time is over 1.5 seconds the status changes to WARNING. If the response time is over 5 seconds the status changes to CRITICAL.

The text message template that appears next to the OK, WARNING, and CRITICAL severity levels generates the message in the Response column when viewing Alert Instances. The variable in the text templates is based on Python string format. Viewing Alert Instances is described and illustrated later in this lesson.

Click the **Save** button (not shown) to save any modifications.

Monitoring a Cluster

Viewing All Alert Instances for an Alert Definition

The screenshot shows the Ambari interface for monitoring a cluster. The top navigation bar includes links for Dashboard, Services, Hosts (with a red notification), Alerts, Admin, and a user dropdown. The main content area is titled "DataNode Process". On the right side, there is a summary box with the status "OK (3)". Below it, detailed information is provided: State (Enabled), Service (HDFS), Component (DataNode), Type (PORT), Groups (HDFS Default, DataNodeAlerts), and Last Changed (Wed, Jul 22, 2015 19:56). The "Configuration" section contains a description: "This host-level alert is triggered if the individual DataNode processes cannot be established to be up and listening on the network.", an interval of 1 minute, and thresholds for OK, WARNING, and CRITICAL levels. The "Instances" section lists three hosts: node1, node2, and node3, all showing an OK status for 7 days with a response message of "TCP OK - 0.000s response on port 50010". A red box highlights this table. Two callout bubbles provide additional context: one points to the "Response" column with the text "Response message", and another points to the "Instances" table with the text "Instances of the alert are shown below the Alert Definition".

Service / Host	Status	24-Hour	Response
HDFS / node1	OK for 7 days	0	TCP OK - 0.000s response on port 50010
HDFS / node2	OK for 7 days	0	TCP OK - 0.000s response on port 50010
HDFS / node3	OK for 7 days	0	TCP OK - 0.000s response on port 50010

Viewing Alert Instances for an Alert Definition

Displaying an Alert Definition also displays all of its Alert Instances. The service name, the host the Instance is running on, and the status of each Instance is also displayed along with the current Response message.

Monitoring a Cluster

Viewing All Alert Instances for a Host

The screenshot shows the Ambari Web UI interface. At the top, there's a navigation bar with tabs for Dashboard, Services, Hosts (highlighted with a red box), Alerts, Admin, and a user icon for admin. Below the navigation bar, a breadcrumb trail shows 'horton' and 'node1'. A callout bubble points to the 'Alerts' tab, which is highlighted with a red box and has a count of 2. The main content area displays a table of alert instances for host 'node1'. The columns include Service, Alert Definition Name, Status, and Response. Some rows show critical errors like 'Metrics Collector - ZooKeeper Server Process' and 'Metrics Collector - HBase Master Process' failing for 5 hours. Other rows show healthy states for Ambari Agent Disk Usage and Heartbeat. The table includes pagination at the bottom.

Service	Alert Definition Name	Status	Response
All	Any	All	Any
Ambari Metrics	Metrics Collector - ZooKeeper Server Process	CRIT	for 5 hours Connection failed: [Errno 111] Connection refused to node1:...
Ambari Metrics	Metrics Collector - HBase Master Process	CRIT	for 5 hours Connection failed: [Errno 111] Connection refused to node1:...
Ambari	Ambari Agent Disk Usage	OK	for 13 days Capacity Used: [18.42%, 18.6 GB], Capacity Total: [101.1 G...
Ambari	Ambari Agent Heartbeat	OK	for 5 hours node1 is healthy
Ambari Metrics	Metrics Collector Process	OK	for 17 hours TCP OK - 0.000s response on port 6188
Ambari Metrics	Metrics Collector - HBase Master CPU Utilization	UNKNW	for 5 hours [Alert][ams_metrics_collector_hbase_master_cpu] Unable to ...
Ambari Metrics	Metrics Monitor Status	OK	for 5 days Ambari Monitor is running on node1
HDFS	NameNode Last Checkpoint	OK	for 7 days Last Checkpoint: [4 hours, 26 minutes, 1290 transactions]
HDFS	DataNode Process	OK	for 7 days TCP OK - 0.000s response on port 50010
HDFS	NameNode Web UI	OK	for 7 days HTTP 200 response in 0.000s

Viewing All Alert Instances for a Host

An administrator can also view all the Alert Instances running on a specific host by opening the **Hosts** page in the Ambari Web UI. On the **Hosts** page, click a specific host and select its **Alerts** tab. The **Alerts** tab lists all the Alert Instances running on that host.

Viewing Alerts in an Alert Group

The screenshot shows the Ambari Web UI interface. At the top, there's a navigation bar with tabs for Dashboard, Services, Hosts (highlighted with a red box), Alerts, Admin, and a user icon for admin. Below the navigation bar, a breadcrumb trail shows 'horton'. A callout bubble points to a dropdown menu labeled 'Groups: MAPREDUCE2 Default (4)' with a red box around it. Another callout bubble points to a list of alert definitions for the 'MapReduce2 Alert Definitions' group, with a red arrow pointing from the dropdown to the list. The main content area displays a table of alert definitions. The columns include Alert Definition Name, Status, Service, Last Status Changed, and State. All alert definitions listed are for the 'MapReduce2' service and are marked as 'Enabled'. The table includes pagination at the bottom.

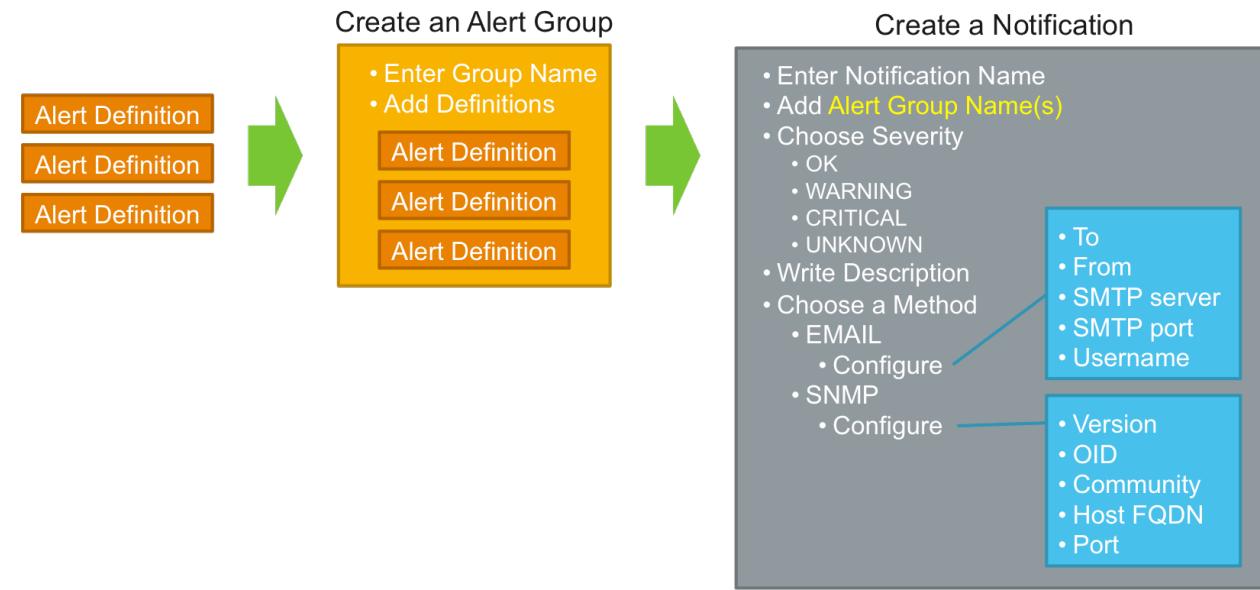
Alert Definition Name	Status	Service	Last Status Changed	State
History Server Process	All	MapReduce2	5 days ago	Enabled
History Server Web UI	All	MapReduce2	5 days ago	Enabled
History Server RPC Latency	OK	MapReduce2	5 days ago	Enabled
History Server CPU Utilization	OK	MapReduce2	5 days ago	Enabled

Viewing Alerts in an Alert Group

The **Groups** drop-down menu is used to filter the list of Alert Definitions displayed on the **Alerts** page. An administrator can select **All** to display all Alert Definitions or they can select a specific Alert Group and display only its Alert Definitions. An administrator can also filter the display using the **Alert Definition Name** text box.

Default and custom groups appear on the **Groups** drop-down menu. There are default service groups for HDFS, YARN, MapReduce2, HBase, Hive, Oozie, ZooKeeper, and Ambari Alerts.

Alerts, Alert Groups, and Notifications



Alerts, Alert Groups and Notifications

Notifications are configured in order to determine to whom an alert is sent, and how it is sent. Alerts can be sent using SMTP or SNMP. By default, there are no configured notifications.

An administrator can create custom groups of alerts and configure notifications. An alert group can be added as a member to one or more notifications. Ambari comes with default alert groups for Hadoop services like HDFS, YARN, ZooKeeper, and others. These default alert groups cannot be modified or deleted, but they can be duplicated. A duplicate can be modified.

Notifications determine which users will be notified of an alert by email or SNMP. Because an alert group can belong to multiple notifications, when an alert is triggered some recipients might receive a notification via SNMP while others might receive the notification via email. Specific email and SNMP configuration information is configured using the Ambari Web UI.

A notification's configuration also determines the alert status that will trigger a notification. For example, a transition from OK to WARNING could be sent to one group of users via SNMP while a transition to CRITICAL could be sent to another group of users via email.

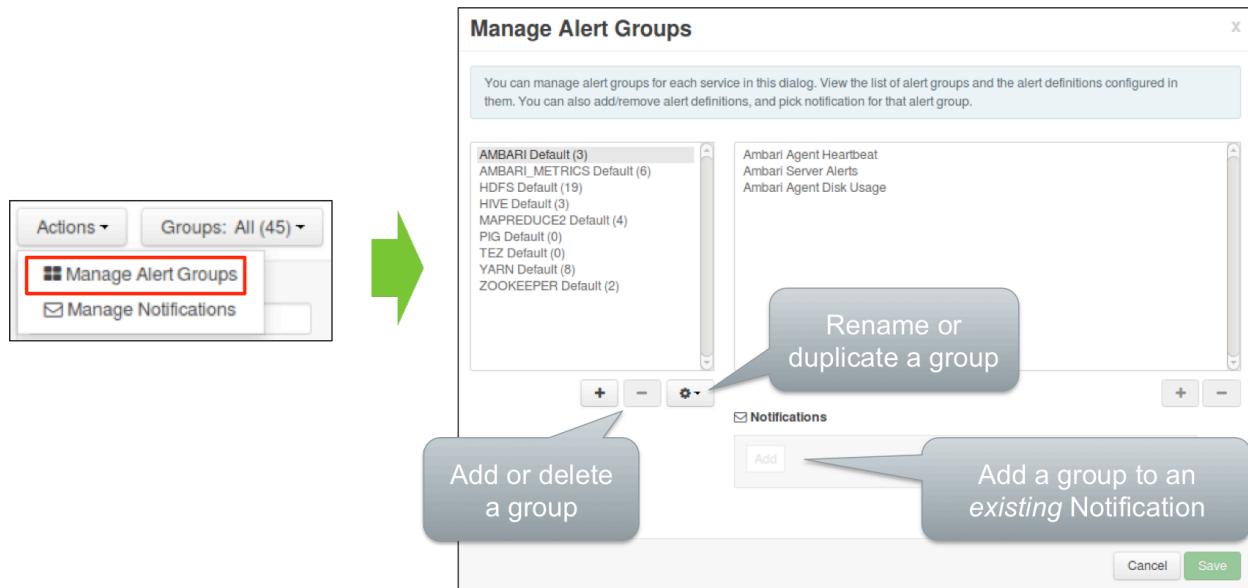
Adding and Configuring an Ambari Alert

The screenshot shows the Ambari Web UI interface. At the top, there is a header bar with the Ambari logo, the cluster name "horton", and a status indicator showing "0 ops" and "2 alerts". The navigation bar includes links for Dashboard, Services, Hosts (with a count of 1), Alerts, Admin, and a user dropdown for "admin". Below the header, a search bar has "Groups: All (45)" selected. Underneath the search bar are two buttons: "Actions" and "Manage Alert Groups". To the right of the search bar are filters for Status (All), Service (All), Last Status Changed (Any), and State (All). Below these filters, a specific alert entry is shown for "Metrics Collector - ZooKeeper Server Process". The alert details are: Status is CRIT, Service is Ambari Metrics, Last Status Changed was 5 hours ago, and the State is Enabled. The alert is highlighted with a red border.

Managing Alert Groups and Notifications

An administrator uses the **Alerts > Actions** menu button to create, modify, or delete Alert Groups and Notifications. The default alert groups cannot be deleted or modified.

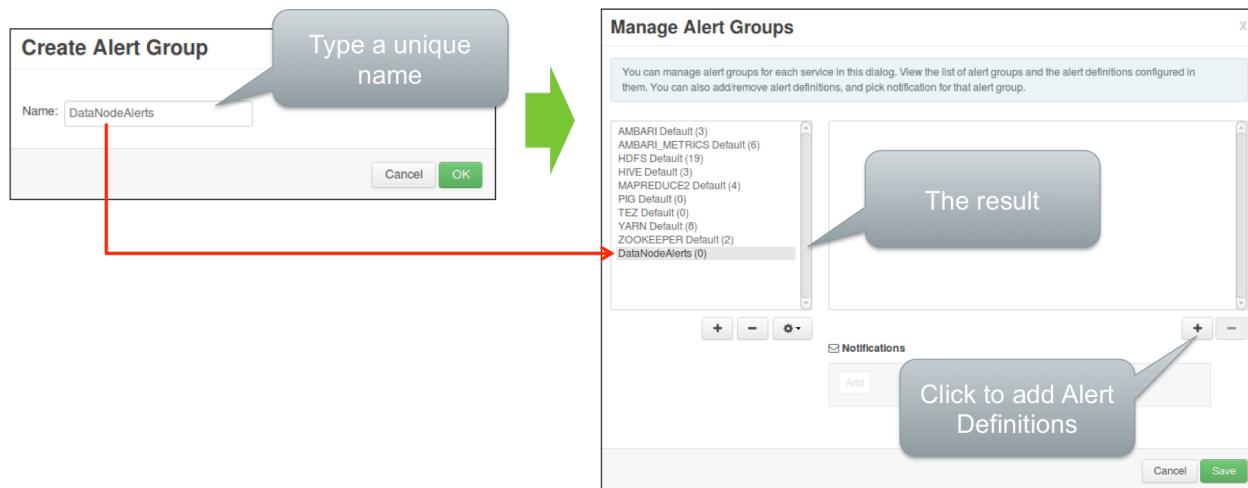
Creating a New Alert Group



Creating an Alert Group

Use **Manage Alert Groups** to create a new Alert Group. The plus (+) and minus (-) buttons create or delete a group. The **gear** button renames or duplicates a group. The **New** button adds a group to an existing Notification. You have to know the name of the existing Notification, or at least the first few characters of its name, in order to add it.

Naming an Alert Group



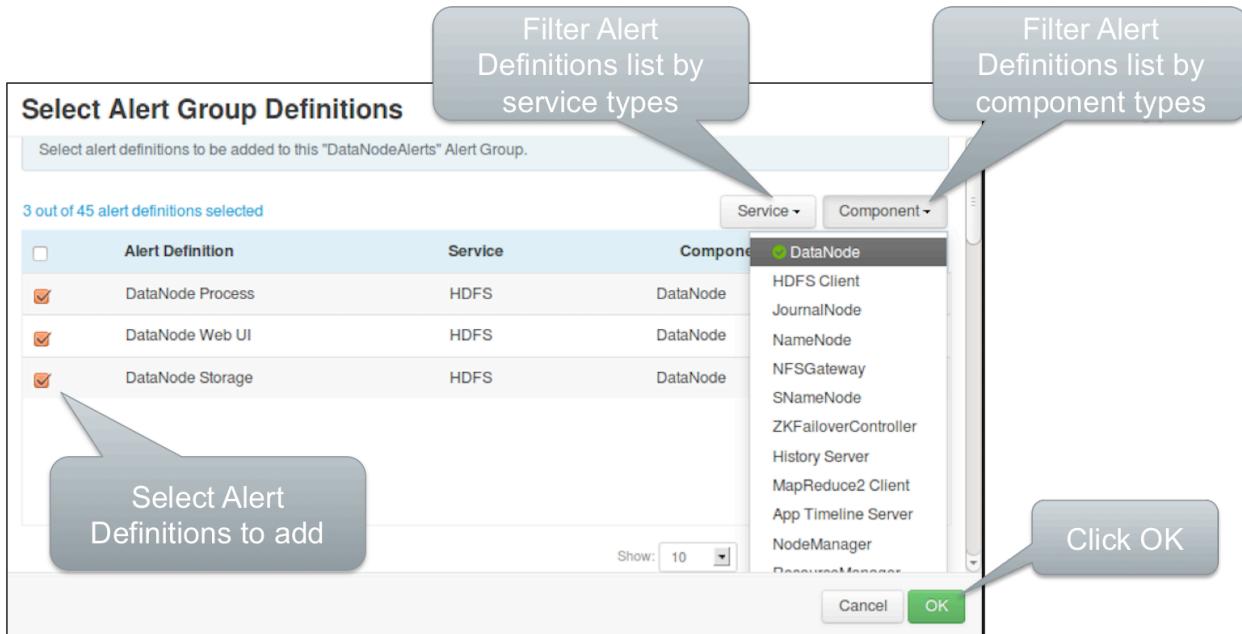
Naming an Alert Group

In the **Create Alert Group** dialog box, type a unique name for the new Alert Group and click **OK**. The new Alert Group will appear in the **Manage Alert Groups** window.

Click the plus (+) button to add new Alert Definitions to the group.

In this example, the new Alert Group named DataNodeAlerts will contain only alerts associated with HDFS DataNodes.

Adding Alert Definitions to an Alert Group



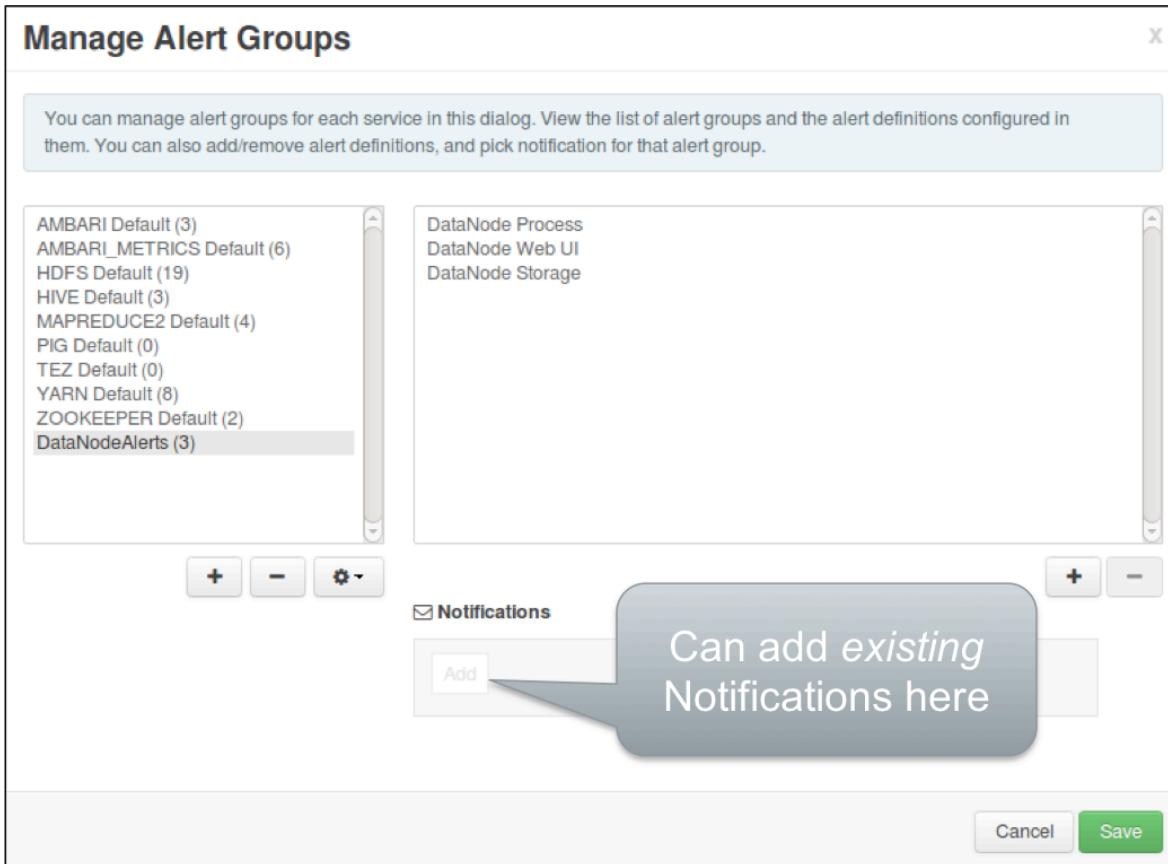
Adding Alert Definitions to an Alert Group

Use the Select Alert Group Definitions window to add Alert Definitions to an Alert Group. There are many Alert Definitions to choose from so use the **Service** or **Component** menus to filter the list of displayed Alert Definitions. The **Service** menu enables you to filter the list of alerts by specific service types. The **Component** menu enables you to filter the list of alerts by specific service component types.

To add an Alert Definition to an Alert Group, click the **Alert Definition's check box**. When the desired Alert Definitions have been selected, click **OK**.

In this example, only alerts associated with HDFS DataNodes are selected for inclusion in the DataNodesAlerts group.

Viewing and Saving an Alert Group

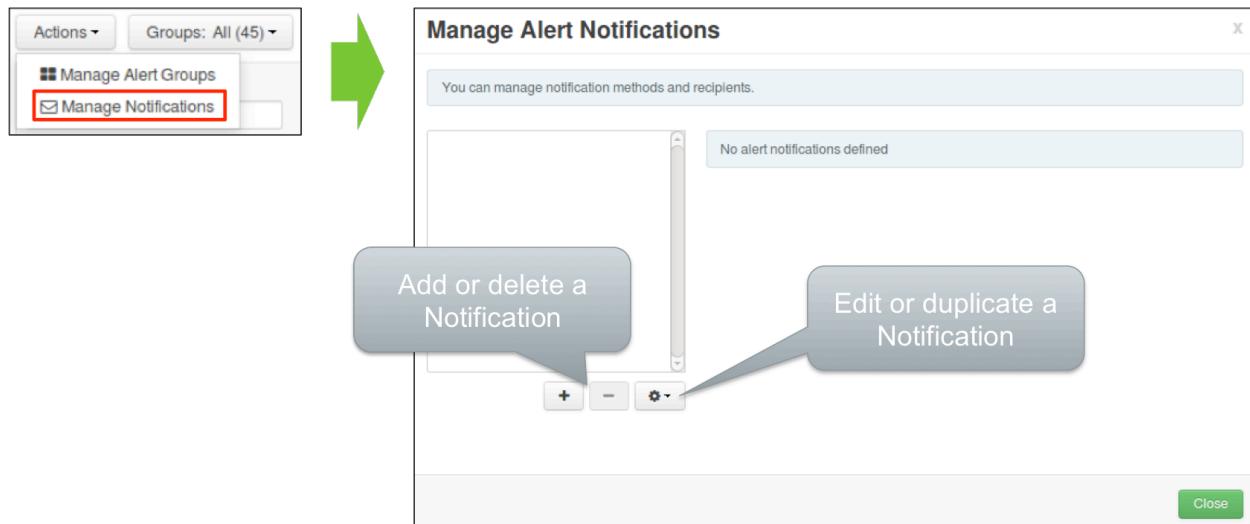


Viewing and Saving an Alert Group

View and confirm the configuration of the new Alert Group. If there is an existing Notification that will work for this group, add it here. To add a Notification click **New** and type its name.

When finished with this window, click **Save**.

Creating a New Notification



Creating a New Notification

Use Manage Notifications to create a new Notification. The plus (+) and minus (-) buttons create or delete a Notification. The gear button edits or duplicates a Notification.

Configure Notification Details

The screenshot shows the 'Create Alert Notification' dialog. It has several input fields and dropdowns. Callout bubbles point to various fields: 'Name the Notification' points to the 'Name' field; 'Add Alert Group(s)' points to the 'Groups' section where 'Custom' is selected; 'Choose severity level(s)' points to the 'Severity' dropdown; 'Type a description' points to the 'Description' text area; 'Choose a method' points to the 'Method' dropdown set to 'EMAIL'; 'Configure method details' points to the 'Email To', 'SMTP Server', 'SMTP Port', and 'Email From' fields; and 'Username and Password not shown' points to a note below. To the right, a detailed view of the 'Method' section is shown for the 'EMAIL' option, including fields for 'Version' (SNMPv1), 'OID', 'Community', 'Hosts' (with validation note 'Hosts must be a valid Fully Qualified Domain Name (FQDN)'), and 'Port'. A 'Save' button is at the bottom right.

Notification Details

Type a unique name for the Notification. The name is used to identify the Notification.

Add one or more Alert Groups to the Notification. You can use the **Shift key** in concert with the mouse pointer to select more than one group. Then select one or more severity levels. All alarms in the selected groups at the selected severity levels will be sent to users using the selected EMAIL or SNMP method.

The method can be either **EMAIL** or **SNMP**. The window is context sensitive and will change depending upon which method you select. Selecting **EMAIL** displays SMTP and email delivery configuration settings. Selecting **SNMP** displays SNMP configuration settings. Both are shown in the screen captures.

A notification is sent only once when an alert status has changed. The alert status can change only when the measured metric is checked at the alert's configured check interval. This policy is designed to avoid sending dozens or even hundreds of notifications.

For **SNMP**, a trap is sent for each alert status change. For example, if two alerts transition to CRITICAL then two traps are sent.

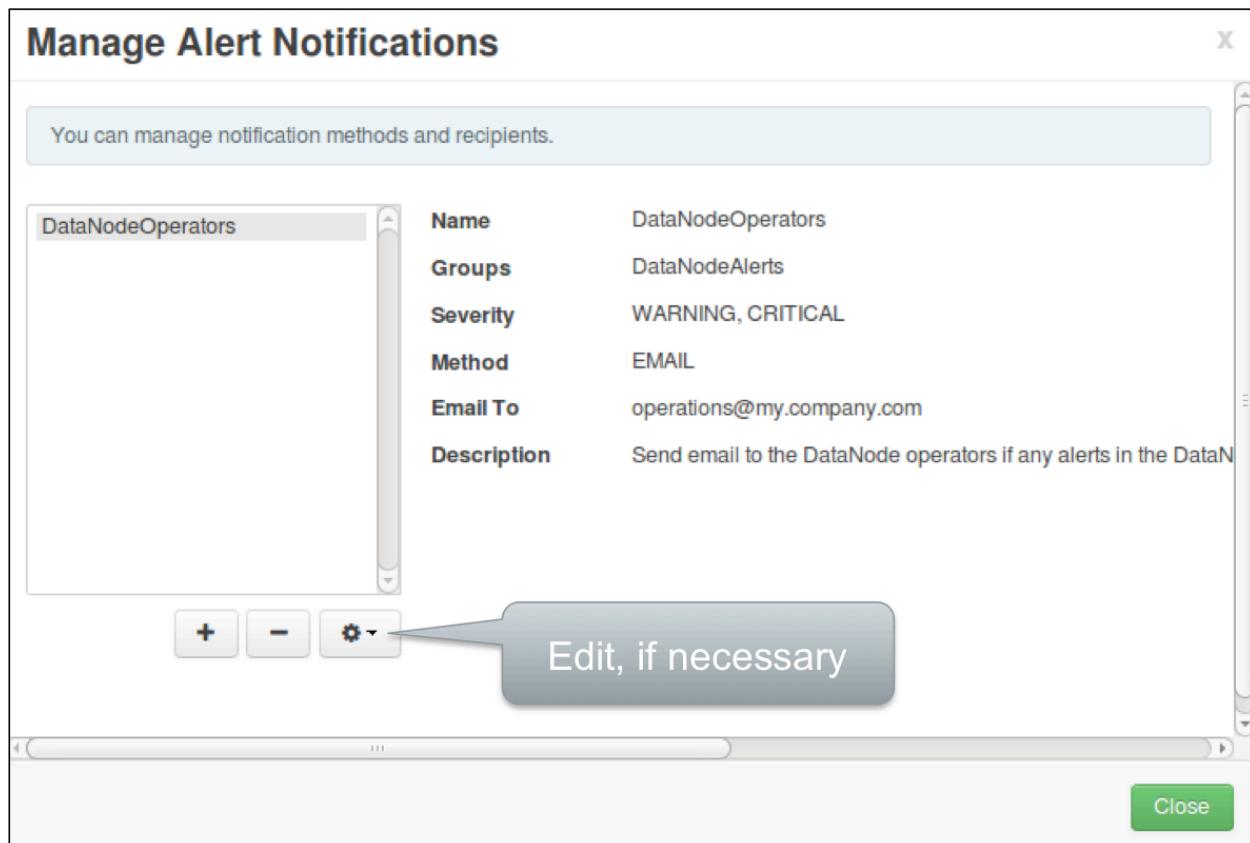
For **email**, an email digest is sent that includes all status changes. For example, if two alerts transition to CRITICAL then a single email is sent that reports that X alert is CRITICAL and Y alert is CRITICAL.

Email and SNMP Configuration

Email Settings	Description
Email To	A comma-separated list of one or more email addresses to send the alert email.
SMTP Server	The FQDN or IP address of the SMTP server to use to relay the alert email.
SMTP Port	The SMTP port on the SMTP Server.
Email From	A single email address to use in the "From:" field in the alert email header.
Use Authentication	Check if your SMTP server requires authentication in order to relay messages. Also provide the username and password credentials for the SMTP server.
SNMP Settings	Description
OID	1.3.6.1.6.3.1.1.5.4 (recommended if a custom OID is not used)
Community	public (not used in SNMP version 3)
Hosts	A comma-separated list of one or more SNMP host FQDNs.
Port	The SNMP port where snmptrapd daemon is listening

Email and SNMP Configuration

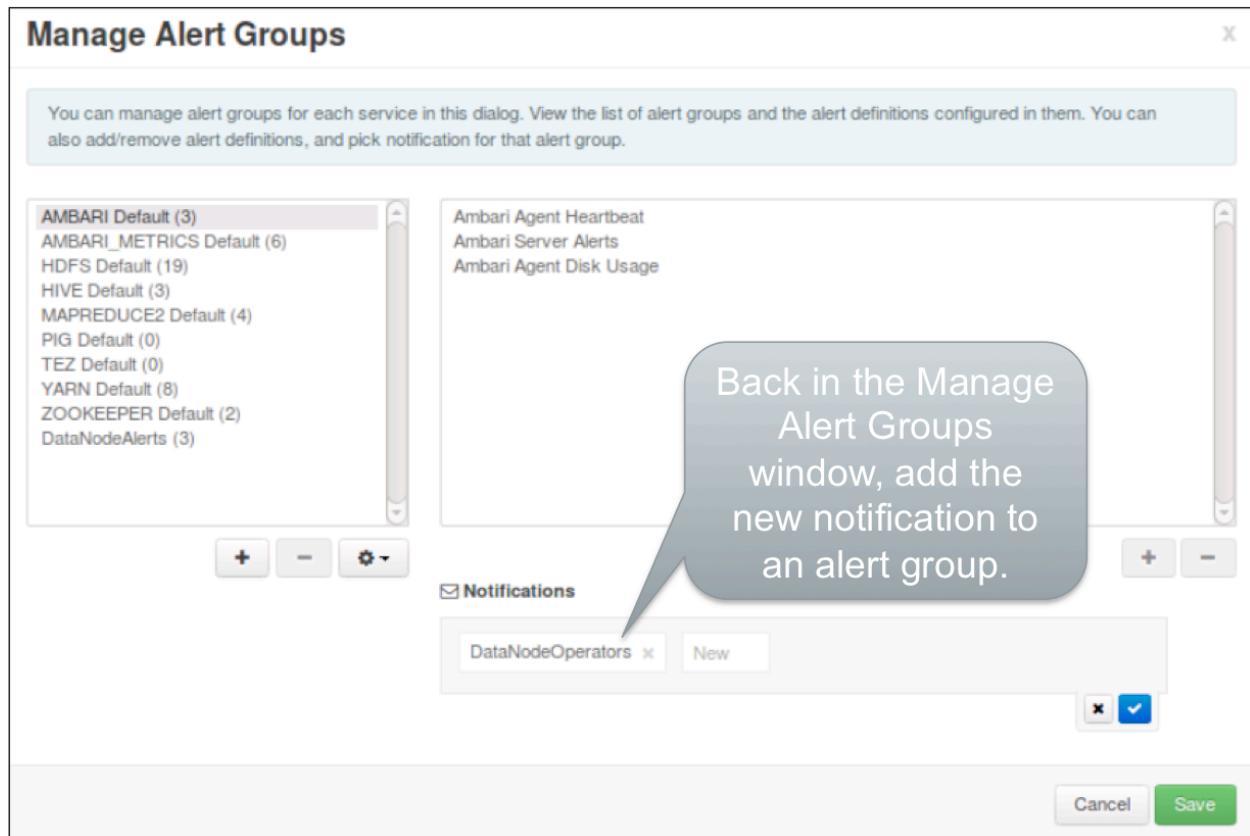
View and Confirm the Notification



View and Confirm the Results

The final window displays the name and configuration of the new Notification. View and confirm the result.

Add a Notification to an Alert Group



Add a Notification to an Alert Group

Use **Alerts > Actions > Manage Alert Groups** to add the new notification to an alert group.

Knowledge Check Questions

- 1) An Ambari Dashboard gauge widget displays a metric expressed as a _____ .
- 2) The Ambari Metrics Collector can run in two different modes, they are _____ mode and _____ mode.
- 3) The Metrics Collector receives host data from the Metrics _____ .
- 4) Disabling an Alert Definition has what affect?
- 5) Ambari Alerts can trigger visual alarms, SNMP traps, and _____ .
- 6) Which two alert types do not have configurable thresholds?
- 7) True or false? Each default Alert Group has a pre-configured Notification.
- 8) True or false? An Alert Group can belong to multiple Notifications.
- 9) True or false? Email can be sent to one person if an alert crosses the WARNING threshold and to another person if it crosses the CRITICAL threshold.

Knowledge Check Answers

- 1) An Ambari Dashboard gauge widget displays a metric expressed as a **percentage**.
- 2) The Ambari Metrics Collector can run in two different modes, they are **embedded** mode and **distributed** mode.
- 3) The Metrics Collector receives host data from the Metrics **Monitor**.
- 4) Disabling an Alert Definition has what affect?
It disables any Alert Instance associated with the Alert Definition
- 5) Ambari Alerts can trigger visual alarms, SNMP traps, and **Email**.
- 6) Which two alert types do not have configurable thresholds?
Web and Script
- 7) True or false? Each default Alert Group has a pre-configured Notification.
False
- 8) True or false? An Alert Group can belong to multiple Notifications.
True
- 9) True or false? Email can be sent to one person if an alert crosses the WARNING threshold and to another person if it crosses the CRITICAL threshold.
True

Summary

- Ambari Metrics System (AMS) is a system for collecting, aggregating, and serving Hadoop and system metrics.
- Ambari Metrics Monitor runs on each node and send system-level data to Metrics Collector that runs on a single node.
- Metrics Hadoop Sinks run on each node and send Hadoop component metrics to Metrics Collector that runs on a single node.
- Ambari Alerts are useful for identifying and resolving problems.
- Ambari Alerts can trigger visual Ambari Web UI alarms, email notifications, and SNMP traps.
- The number of Alert Instances created varies by cluster size and service configuration.
- Notifications determine who and how alerts are received.

Protecting a Cluster with Backups

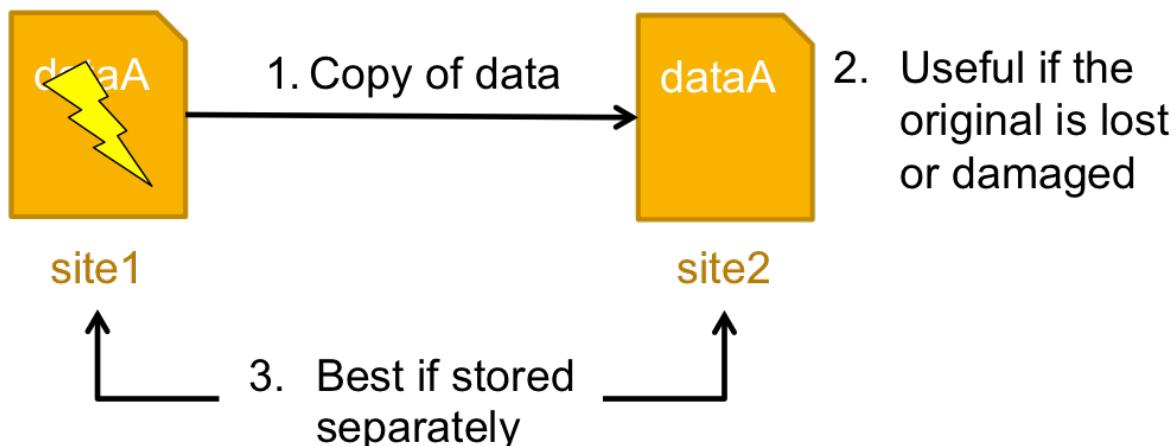
Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Summarize Hadoop backup considerations
- ✓ Enable and manage HDFS snapshots
- ✓ Copy data using DistCp
- ✓ Use Snapshots and DistCp together

Hadoop Backups

A backup is:



Importance of Backups

A backup is a redundant copy of data made at a specific point in time. Having a current backup is useful to restore data that has been lost due to deletion or damage. It is best to store the copy of the data in a separate location. Using a separate set of hosts is good, using a geographically separated data center is better.

HDFS replication already maintains multiple copies of data on separate hosts so why are backups still needed? Disasters, large and small, still occur. Systems fail due to hardware and software problems. A hardware or software problem that affects multiple systems could render data unreachable, or worse, unrecoverable.

The same is true for both natural and man-made disasters. Fires, floods, wind, earthquakes, war, human error, or criminal or terrorist acts can all destroy a data center or make its data unreachable. The results of such disasters could be catastrophic for an organization without a backup in an alternate location.

What to Back Up



When planning for backups, backing up application data seems obvious. However, perhaps not all application data requires a backup. You must consider how critical the data is, and whether or not it can be retrieved using some other method. For example, a backup of raw ingest data could be used along with extract-transform-load programs to regenerate any missing but required processed data. Perhaps the opposite might be true; an organization might not care about the raw data but only requires restored access to the processed data.

When considering HDFS backups, consider files, directories, Hive tables, and HBase tables.

Hadoop also relies on a considerable amount of metadata. For example, all data in HDFS is lost if the HDFS metadata on the NameNode is lost. For this reason, it is vital to have current backup copies of the `fsimage` and `edits` files. This is particularly true if NameNode HA has not been configured.

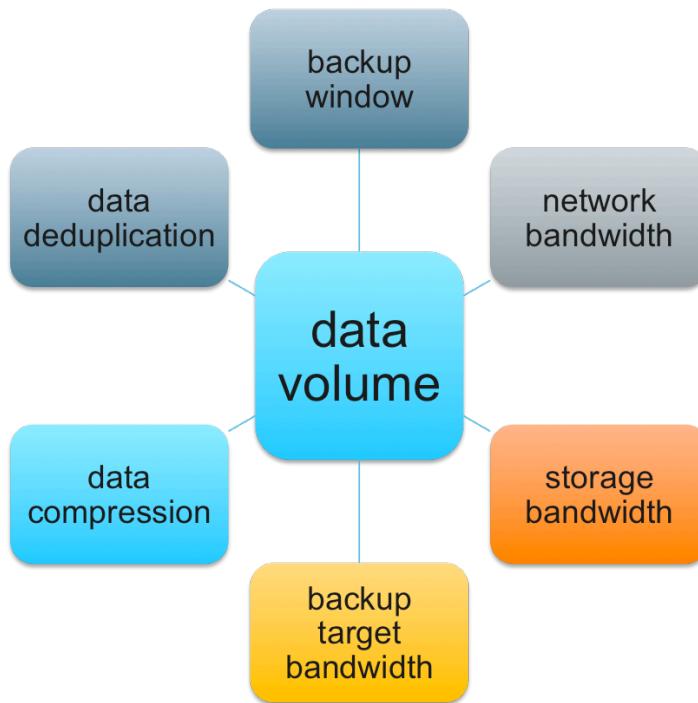
In addition, the Hadoop ecosystem relies on several databases. For example, these include Ambari, Hive, Oozie, and Ranger. These should be backed up using the database vendors' recommended backup procedures. The use of database HA features also provides additional protection.

Zookeeper also maintains application data in its znode hierarchies. Znode information is stored on the ZooKeeper servers beneath the directory specified by the `dataDir` property in the `zoo.cfg` file. By default, Ambari installs the `zoo.cfg` file in the `/etc/zookeeper/conf` directory.

Finally, you should consider a backup of the master, slave node, and Ambari Server /etc subdirectories. These hold the cluster and cluster application configuration files as well as the Ambari configuration files. The Ambari Server configuration files are found in the `/etc/ambari-server/conf` directory.

An alternative to backing up and restoring these configuration files manually would be to use Ambari Blueprints or Cloudbreak to initially install your cluster. These tools use customizable blueprint files to automatically install and configure a cluster. The blueprint can install multiple clusters that share the same configuration. This capability can be useful for disaster recovery.

Backup Considerations



Backup Considerations

The primary consideration for backup is the amount of data to back up. Hadoop clusters typically store anywhere from terabytes of data to petabytes of data. However, not necessarily all of this data needs to be backed up. An organization must determine what data is mission critical and what data is exhaust data and does not require back up. This determination can dramatically affect the backup design by reducing the overall amount of data to back up.

One of the primary concerns is the time required to back up large amounts of data. As mentioned above, one method of dealing with this is to reduce the amount of data to back up. The backup window can also be enlarged by being able to back up data while HDFS is online and serving clients. This is made possible by using HDFS snapshots, which are described later in this lesson.

Bandwidth is another primary consideration. Data must be transferred from disks and across a network in order to back it up. Disk and network bandwidth must be sufficient to handle the volume of data in the time allotted, while simultaneously accommodating daily cluster activities.

It is common for backup utilities or backup devices to compress data during backup. However, HDFS data is typically already compressed and it is unlikely that it can be compressed again. If you are using such a device, this affects the time and space required to perform a backup.

Another common operation during backup is deduplication. With deduplication, backing up a file the first time backs up the entire file. However, backing up the same file a second time backs up only any changed data. This saves backup time and space. However, deduplication is only useful if the same file is backed up multiple times. An organization has to decide whether multiple versions of a file will be backed up. This consideration is amplified by the fact that HDFS data is immutable. HDFS files can be appended but existing data cannot be updated.

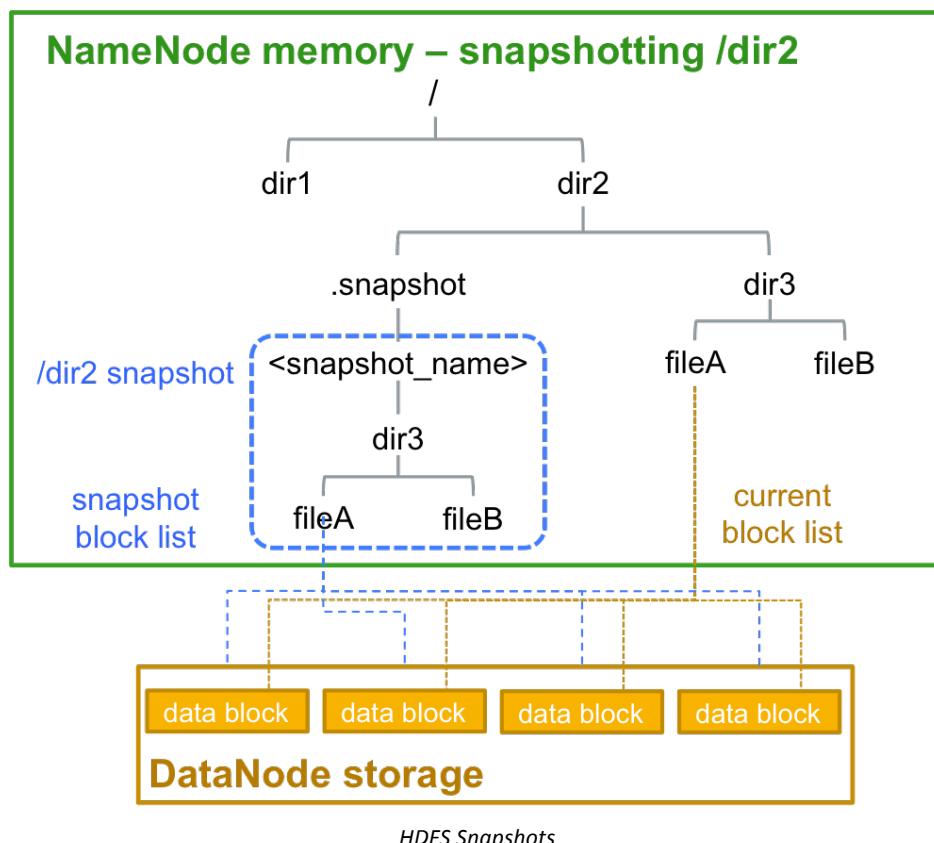
Using HDFS Snapshots

A snapshot is a point-in-time, read-only image of the entire file system or a sub tree of the file system.

HDFS snapshots are useful for:

- **Protection against user error:** With snapshots, if a user accidentally deletes a file, the file can be restored from the latest snapshot that contains the file.
- **Backup:** Files can be backed up using the snapshot image while the file system continues to serve HDFS clients.
- **Test and development:** Files in an HDFS snapshot can be used to test new programs without affecting the HDFS file system that is concurrently supporting HDFS clients.
- **Disaster recovery:** Snapshots can be replicated to a remote recovery site for disaster recovery.

HDFS Snapshot Operation



A snapshot is created in NameNode memory, which makes snapshot creation a very fast operation. Only the NameNode is aware of a snapshot. No data blocks on the DataNodes are copied. Snapshot information is persisted to NameNode storage during HDFS checkpoint operations.

DataNodes are not aware of snapshots or the fact that only snapshots of the original file might own some of the data blocks. For example, additional data could have been appended to a file after a snapshot has been taken. In this case, the block list of the current file would include more blocks than the snapshot of the file. A file could also be deleted after a snapshot has been taken. In this case, there would be no block list for a “current” file but there would be a block list for the snapshot of the file. The file could be restored by copying the snapshot out of the .snapshot directory.

Snapshots Administrator Tasks

Snapshots can be taken on any directory once the directory has been enabled for snapshots. A directory enabled for snapshots is considered snapshottable. A snapshottable directory is able to accommodate 65,536 simultaneous snapshots. Each snapshot consumes additional NameNode memory.

To enable snapshots requires HDFS superuser privileges. The syntax to enable snapshots on a directory is `hdfs dfsadmin -allowSnapshot <directory_path>`.

There is no limit on the number of snapshottable directories. Administrators may set any directory to be snapshottable. If there are snapshots in a snapshottable directory, the directory can be neither deleted nor renamed before all the snapshots are deleted.

Nested snapshottable directories are currently not allowed. In other words, a directory cannot be set to snapshottable if one of its ancestors/descendants is a snapshottable directory.

Before disallowing snapshots, all snapshots on the directory must be disabled. The syntax to disable snapshots on a directory is `hdfs dfsadmin -disallowSnapshot <directory_path>`.

Snapshots User Tasks

Before a user can take a snapshot of a directory, the directory must be snapshottable. The `hdfs lsSnapshottableDir` command lists any snapshottable directories.

All snapshots are given a name. By default a snapshot name is based on the current time. The format of the default snapshot name is `yyyymmdd-hhmmss.SSS`. Optionally, a user can specify a snapshot name of their choice.

Existing snapshots can also be renamed. Rename a snapshot using the command syntax `hdfs dfs -renameSnapshot <directory_path> <old_name> <new_name>`.

Create a snapshot using the command syntax `hdfs dfs -createSnapshot <directory_path> [<snapshot_name>]`.

View snapshots by listing the contents of the `.snapshot` directory. Each subdirectory is the name of a separate snapshot.

Each snapshot consumes memory space on the NameNode. For this reason, snapshots that are no longer needed should be removed. Delete a snapshot using the command syntax `hdfs dfs -deleteSnapshot <directory_path> <snapshot_name>`. The `<snapshot_name>` is not an optional argument, even if the snapshot has only a default name based on time it was created.

Viewing Snapshots in NameNode UI

The screenshot shows the NameNode UI interface with the 'Snapshot' tab selected. The main section is titled 'Snapshot Summary'. It displays two rows of snapshot information for 'Snapshottable directories':

Path	Snapshot Number	Snapshot Quota	Modification Time	Permission	Owner	Group
/user/root	1	65536	8/14/2015, 2:37:32 PM	rwxr-xr-x	root	root
/user/steve	0	65536	7/24/2015, 2:55:48 PM	rwxr-xr-x	steve	steve

Below this, it shows one row of 'Snapshotted directories':

Snapshot ID	Snapshot Directory	Modification Time
s20150814-143732.076	/user/root/.snapshot/s20150814-143732.076	8/14/2015, 2:37:32 PM

NameNode UI

The NameNode UI displays snapshot information on its **Snapshot** tab. The **Snapshot** tab lists all directories enabled for snapshots along with a list of all current snapshots.

Using DistCp

Hadoop DistCp (distributed copy) can be used to copy data between Hadoop clusters or within a Hadoop cluster. DistCp can copy just files from a directory or it can copy an entire directory hierarchy. It can also copy multiple source directories to a single target directory.

DistCp:

- Uses MapReduce to implement its I/O load distribution, error handling, and reporting.
- Has built-in support for multiple file system types. It can work with HDFS, Amazon S3, Cassandra, and others. DistCp also supports copying between different HDFS versions.
- Can generate a significant workload on the cluster if a large volume of data is being transferred.
- Has many command options. Use `hadoop distcp -help` to get online command help information.

Copying with DistCp

Copying between clusters:

```
hadoop distcp hdfs://<namenode1>:8020/<source>
hdfs://<namenode2>:8020/<destination>
```

Where <source> is a directory or file path and <destination> is a directory path

Source directories are recursively copied.

Copying from multiple sources (files or directories):

```
hadoop distcp hdfs://<namenode1>:8020/<source1>
hdfs://<namenode1>:8020/<source2> hdfs://<namenode2>:8020/<destination>
```

Where <destination> must be a directory path

Specifying multiples sources in a file:

```
hadoop distcp -f hdfs://<namenode1>:8020/<source_list>
hdfs://<namenode2>:8020/<destination>
```

Where <source_list> contains:

```
hdfs://<namenode1>:8020/<sourceA>
hdfs://<namenode1>:8020/<sourceB>
```

If the source is a directory path, the entire directory and all its contents are recursively copied to the destination. The source can also be an individual file. By default, files already existing at the destination are not copied. A count of not copied files is reported at the end of each job and is accurate if the job finishes without error.

Files can be copied from multiple sources into a single destination directory. Multiple sources can be specified as command-line arguments or be specified by reading them from a file. The sources can be directories, files, or a combination of both.

For example, to copy multiple directories to a single directory, use the syntax `hadoop distcp hdfs://<namenode1>:8020/<source1> hdfs://<namenode1>:8020/<source2> hdfs://<namenode2>:8020/<destination>`. To specify multiple sources in file, use the syntax `hadoop distcp -f hdfs://<namenode1>:8020/<source_list> hdfs://<namenode2>:8020/<destination>`. The <source_list> file would contain a line for each source. For example, each line could contain one of the following entries; `hdfs://<namenode1>:8020/<sourceA> or hdfs://<namenode1>:8020/<sourceB>`.

When copying from multiple sources, DistCp will abort the job if two sources collide. For example, if `fileA` exists in the first source and is copied to the destination directory, then a `fileA` in the second source would collide at the destination directory and the job would abort.

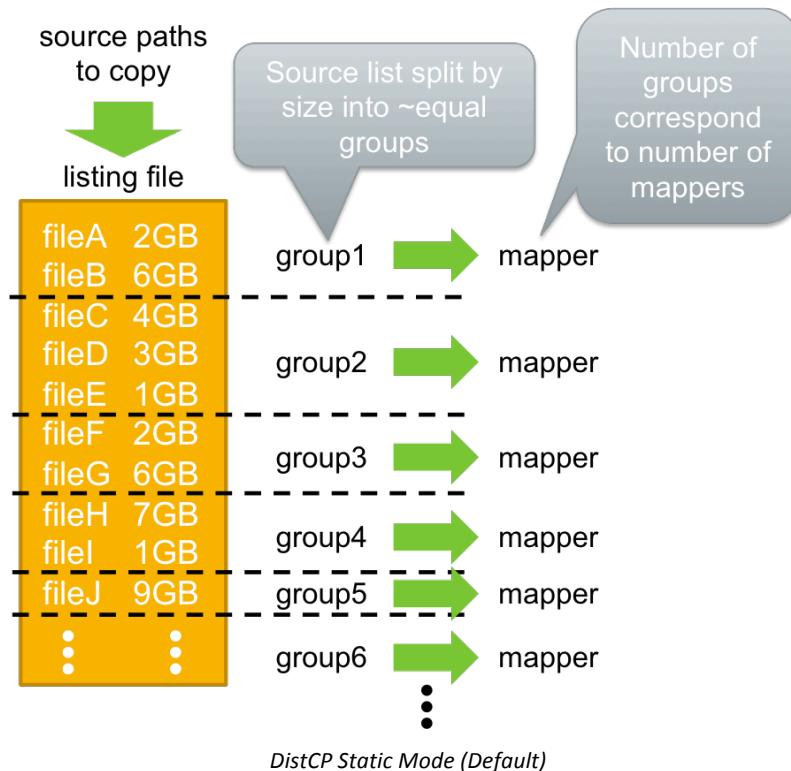
Updating and Overwriting

The `DistCp -update` option is used to copy files from the source that do not already exist at the target, or that exist but have different contents. The `DistCp -overwrite` option overwrites target files even if they exist at the source and have the same contents.

The `-update` and `-overwrite` options warrant further description because their handling of source paths varies from the default in a subtle manner. By default, `DistCp` copies a source directory and its contents to the destination directory. However, the `-update` and `-overwrite` options cause only the contents of the source directory to be copied to the destination directory.

The following two examples illustrate the difference. Consider a default scenario where `/sourcedir/fileA` is copied to `/destdir`. The result would be `/destdir/sourcedir/fileA`. If the `-update` or `-overwrite` options are used the result is different. If the source `/sourcedir/fileA` is copied to `/destdir`, the result would be `/destdir/fileA`. The file is copied but not the source directory it was in.

Static Mode



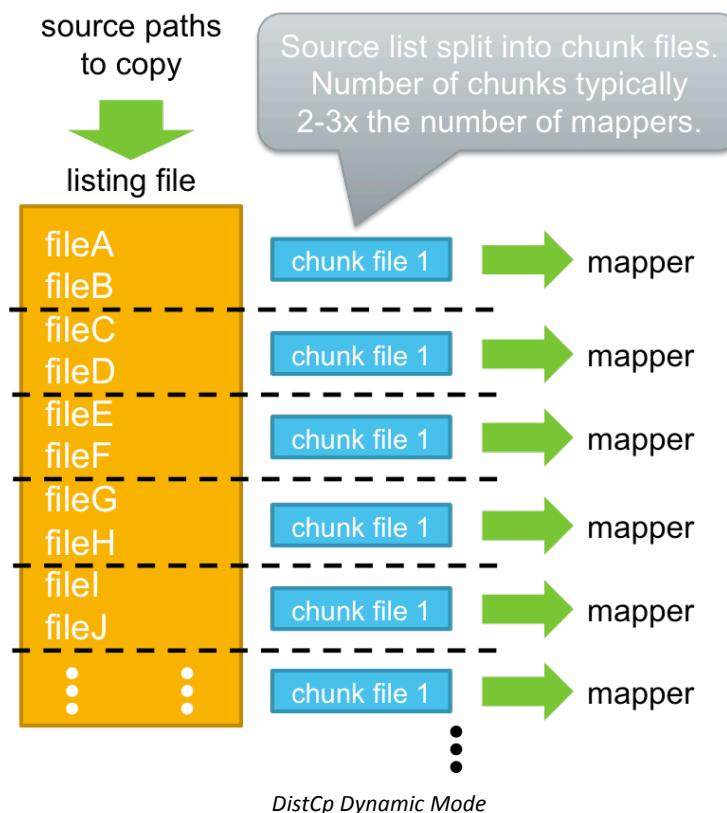
The goal of static mode is to balance the load evenly across mappers and make each mapper copy approximately the same number of bytes. To achieve this, all of the source file paths are calculated from the `distcp` command-line arguments and aggregated in to a temporary listing file. Then the listing file is split into groups of file paths, such that the sum of file sizes in each group are nearly equal to every other group. The splitting is not always perfect.

The number of groups will correspond to the number of mappers. The number of mappers used to copy data defaults to 20 so a listing file with at least 20 file paths could use all 20 mappers. The `distcp` command includes a `-m <n>` option that can be used to specify a different number of mappers. Each group of files is processed by a different mapper.

The smallest level of granularity for DistCp is a file. Each file is processed by only a single mapper. For example, if there are only four files to copy then a maximum of four mappers are used.

With static mode, mappers that are faster finish early and are not assigned any more groups to process. Mappers that are slower still must process all of the files assigned to them. This is the default mode although it can be explicitly specified by adding the `-strategy uniformsize` option.

Dynamic Mode



The goal of dynamic mode is to improve copy performance. To achieve this, all of the source file paths are calculated from the `distcp` command-line arguments and aggregated in a temporary listing file. Then the listing file is split into chunk files such that each chunk has about the same number of source file paths. The splitting is not always perfect. The number of chunk files created by DistCp is typically about two to three times the number of mappers. This means that a mapper might finish one chunk file and move onto another.

The file paths listed in each chunk file are copied by a single mapper. The number of mappers used to copy data defaults to 20 so a listing file with at least 20 file paths could use all 20 mappers. The `distcp` command includes a `-m <n>` option that can be used to specify a different number of mappers.

With dynamic mode, mappers that are faster finish early and can be dynamically assigned another chunk file to process. Mappers that are slower still must process all of the files in the chunk assigned to them. This efficient operation typically makes dynamic mode faster than static mode.

Dynamic mode is specified by using the `-strategy dynamic` option.

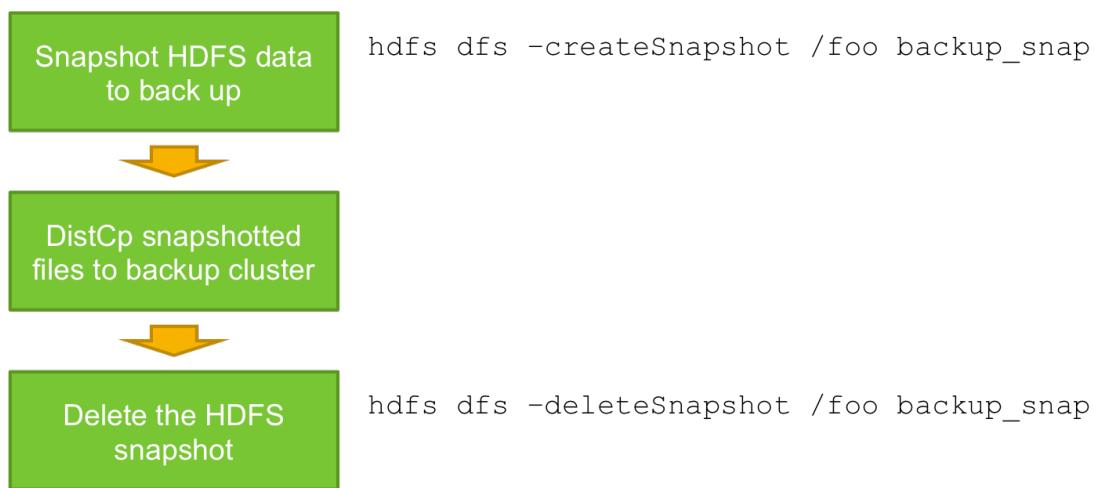
DistCp Recommendations

The number of mappers to use when copying data will affect performance. The default maximum number of mappers is 20. The `-m <n>` option can be used to specify a specific number of mappers. The question is, how many should be used?

The number will depend on factors like the sizes of the source and destination clusters, the size of the copy data, and the available storage and network bandwidth. The key is to vary the number of mappers and monitor the results over time until an effective number of mappers is found for a specific amount of data transfer.

If DistCp reports that it is out of memory before it completes a copy job, a likely problem would be the amount of Java heap space. Java heap space can be increased using the command `export HADOOP_CLIENT_OPTS="-Xms64m -Xms1024m"`. In this command the Java heap space would be set to 1024 megabytes.

Using Snapshots and DistCp Together



Using Snapshots and DistCp Together

HDFS snapshots can be combined with DistCp to create the basis for an online backup solution. Because a snapshot is a read-only, point-in-time copy of the data, it can be used to back up files while HDFS is still actively serving application clients.

Backups can even be automated using tools like Apache Falcon and Apache Oozie. Oozie is a workflow engine that can perform tasks. DistCp is one of the tasks that is built-into Oozie, which means Oozie can be configured to perform a backup. However, the Oozie job would have to be manually launched using a command-line command or by adding it to a scheduler like the Linux cron utility. This is where Falcon becomes useful.

Falcon can automate jobs using Oozie as its workflow engine. Falcon can trigger jobs to run based on the current time and on the availability of data. Falcon also can retry failed jobs or delete old data based on user-defined retention policies. All of these features are useful for backup purposes.

Restoring from a Backup

Restoring from a DistCp backup on another cluster is a matter of reversing the backup process.

On the backup cluster, find the snapshot files that you want to restore. On the production cluster, move or remove any files that should be replaced. Of course if these files are already missing then this step is not necessary. Lastly, run the `distcp` command without the `-update` or `-overwrite` options to restore only the missing files.

Knowledge Check Questions

- 1) If HDFS data is already redundant across multiple DataNodes, why do you need to back it up?
- 2) How are HDFS snapshots useful for disaster recovery?
- 3) True or false? Snapshots are fast because a minimal number of data blocks are copied when a snapshot is initially created.
- 4) Who may run the command `hdfs dfsadmin -allowSnapshot <directory_path>`?
- 5) The `DistCp -update` option ensures that only files that meet two criteria are copied. What are those criteria?
- 6) Which `DistCp` mode might typically be faster, static or dynamic?
- 7) Which two Apache frameworks can be used to automate HDFS backups?

Knowledge Check Answers

- 1) If HDFS data is already redundant across multiple DataNodes, why do you need to back it up?
Because disasters happen that can affect more than one system or an entire data center.
- 2) How are HDFS snapshots useful for disaster recovery?
Because snapshots can be replicated to another cluster that could take over responsibility for a failed cluster.
- 3) True or false? Snapshots are fast because a minimal number of data blocks are copied when a snapshot is initially created.
False. Snapshots are created in NameNode memory.
- 4) Who may run the command `hdfs dfsadmin -allowSnapshot <directory_path>`?
Only an HDFS superuser.
- 5) The DistCp `-update` option ensures that only files that meet two criteria are copied. What are those criteria?
The file does not exist at the destination, or the file at the destination has different contents.
- 6) Which DistCp mode might typically be faster, static or dynamic?
Dynamic.
- 7) Which two Apache frameworks can be used to automate HDFS backups?
Apache Falcon and Apache Oozie.

Summary

- It is still important to back up Hadoop because of system failures, and man-made or natural disasters.
- Consider HDFS data, metadata, and Hadoop configuration files for backup.
- An HDFS snapshot is a read-only, point-in-time image of data.
- Snapshot creation is fast because a snapshot is created in NameNode memory; no data blocks are copied.
- Users can snapshot a directory, as long as an HDFS administrator has enabled snapshots on the directory.
- DistCp copies HDFS data between or within Hadoop clusters.
- DistCp leverages MapReduce mappers to distribute I/O and speed the data transfer.
- HDFS snapshots can be combined with DistCp to back up HDFS data to backup clusters.

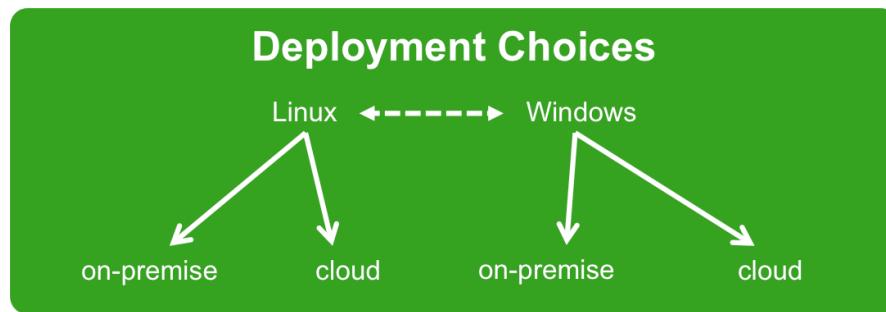
Installing the Hortonworks Data Platform

Lesson Objectives

After completing this lesson, students should be able to:

- ✓ Identify Hadoop cluster deployment options
- ✓ Plan for Hadoop cluster deployment
- ✓ Describe the Ambari installation process
- ✓ Perform an interactive HDP installation using Apache Ambari

Hadoop Deployment Options



Hadoop Deployment Options

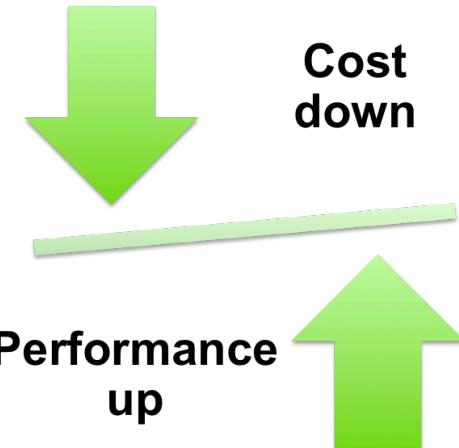
There are a lot of deployment choices with Hadoop.

First, consider whether you want an onsite deployment using your own hardware. If that is not desirable or possible, then you have the choice of a cloud-based deployment. For example, Cloudbreak is able to install a cluster in several popular cloud environments including Microsoft Azure, Amazon Web Services, Google Cloud Platform, OpenStack, and Docker-based environments.

You also have choices when it comes to the operating system platform. You can deploy on Microsoft Windows or several Linux platforms.

Several vendors, including Teradata, sell Hadoop appliances which further simplify deployment.

Planning a Hadoop Cluster Deployment



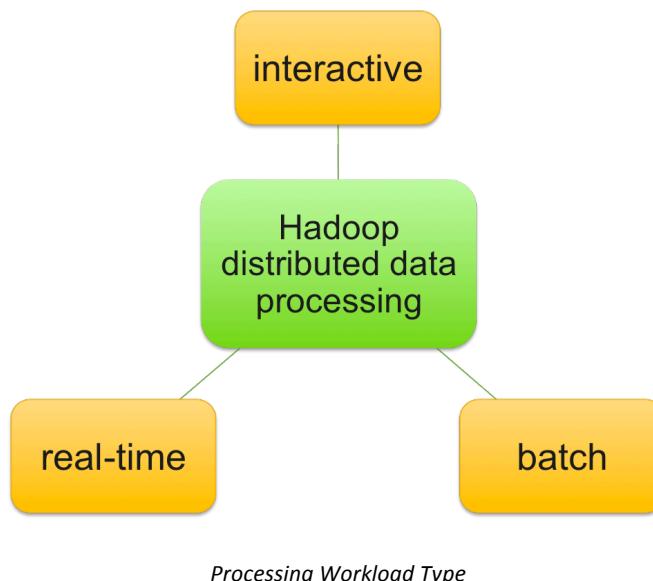
Cost versus Performance

Planning for a new cluster is not trivial. The core design challenge is cost versus performance goals. Sufficient hardware is required to meet performance goals but too much hardware needlessly increases costs. Hardware sizing help is available. Hortonworks provides some basic help in an online Hadoop Cluster Configuration Guide available at <http://hortonworks.com/cluster-sizing-guide/>. Customized help is available by engaging the Hortonworks Professional Services team. Hortonworks Pre-Sales Engineers can offer some guidance based on their in-the-field experience.

Be sure to consider:

- Workload Type
- Storage
- Hardware
- Operating Systems
- Software
- Databases

Consider Processing Workload Type



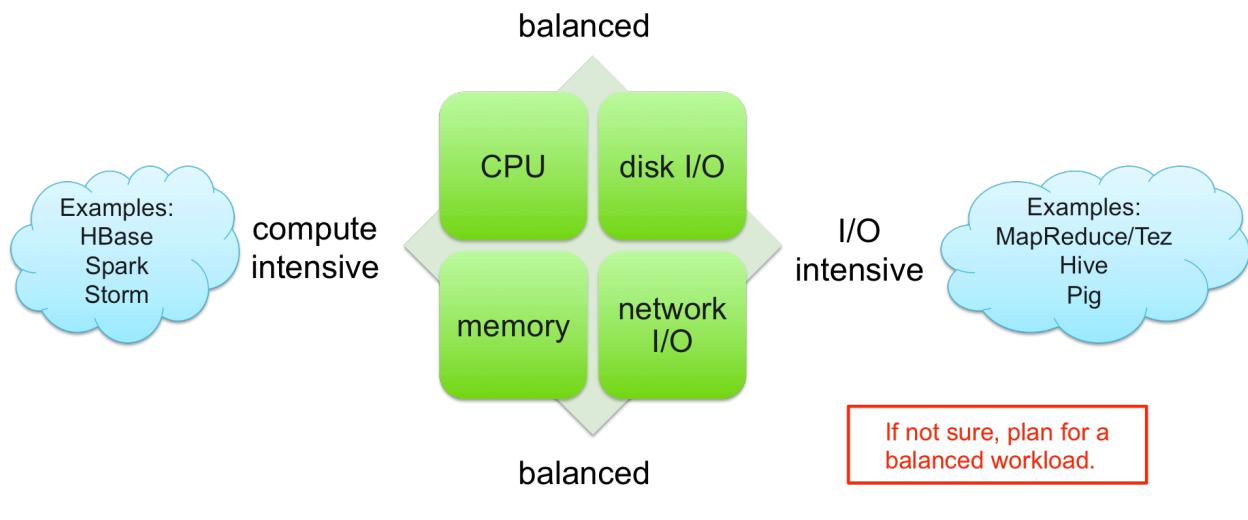
Hadoop clusters commonly support three different types of distributed processing workloads: interactive, batch, and real-time. Each of these has unique characteristics and places a different type of load on the cluster resources. Understanding these workload types can be helpful when attempting to size hardware for a cluster.

Interactive processing involves the processing of data with the continual exchange of information between the cluster and a user. It is commonly used for the analysis of existing historical data—data older than 15 minutes—or for data entry. It places sporadic loads on the cluster resources that are often difficult to predict in advance.

Batch processing is commonly used to analyze historical data. Historical data is primarily stored on disk and is moved to memory for processing. Batch jobs are run sporadically or periodically and run from a few seconds to multiple hours. They place sporadic to periodic loads on the cluster resources. It is much easier to predict the resource loads for those batch jobs that are repeated at regular time intervals.

Real-time processing is commonly used to ingest a continuous stream of data, process it, and output it to storage. Real-time data is typically initially ingested into memory and then moved to disk after processing. Real-time jobs are always running unless manually stopped. They are primarily used by automated systems to prevent certain outcomes or to optimize operations. Real-time processing can consume both computational and I/O resources depending on the application and job.

Consider Processing Workload Patterns

*Cluster Workload Patterns*

Computational power, network I/O bandwidth, disk I/O bandwidth, and disk space are the most important parameters to consider for accurate hardware sizing. Cluster workloads often fall into one of three categories: compute intensive, I/O intensive, or balanced.

A compute intensive workload is typically bound by CPU or memory constraints and is characterized by the need for a large number of CPUs and large amounts of memory. Examples of compute intensive workloads commonly include HBase, Spark, or Storm jobs.

An I/O intensive workload is often bound by disk I/O bandwidth or perhaps network I/O bandwidth. Examples of I/O intensive workloads commonly include MapReduce jobs. Because historically Hive and Pig jobs have been converted to MapReduce jobs, Hive and Pig jobs have also been I/O intensive. With the current HDP release, Hive and Pig jobs are now converted to Tez jobs by default. Tez is more disk I/O efficient than MapReduce, which reduces but does not eliminate the overall disk I/O bandwidth requirements.

A balanced workload is characterized by an even blend of both computational and I/O bandwidth needs. A cluster running a mix of batch, interactive, and real-time data processing jobs often has a balanced workload. If you are unsure what your workload will be, plan for a balanced workload.

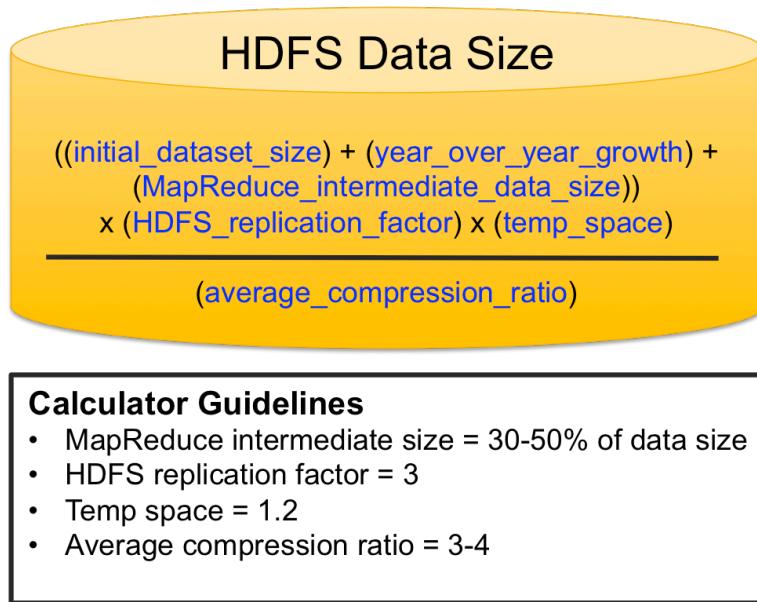
Planning for Workloads

Machine Type	Workload	Storage	CPU Cores	Memory (GB)	Network
Worker Nodes	Balanced	Twelve 2-3 TB disks	8	128-256	1 Gb minimum onboard with 2x10 Gb added
	Compute	Twelve 1-2 TB disks	10	128-256	1 Gb minimum onboard with 2x10 Gb added
	Storage-heavy (backup data)	Twelve 4+ TB disks	8	128-256	1 Gb minimum onboard with 2x10 Gb added
HDFS NameNode	Balanced	Four or more 2-3 TB disks in RAID 10	8	128-256	1 Gb minimum onboard with 2x10 Gb added
YARN ResourceManager	Balanced	Four or more 2-3 TB disks in RAID 10	8	128-256	1 Gb minimum onboard with 2x10 Gb added

Hardware Sizing Guidelines for a Balanced Workload

Hortonworks recommends working with a sales solution engineer or consultant when planning your hardware specifications.

Storage Calculator



Storage Calculator

The storage calculator is designed to provide a rough estimate of the amount of HDFS storage space required when designing a cluster. It accounts for not only the initial dataset size and any expected year-over-year growth, but also accounts for data that is temporarily created as a result of data processing.

Hardware Sizing

Achieving optimal results from a Hadoop implementation begins with choosing the correct-sized hardware. The effort involved in the planning stages can pay off dramatically in terms of the improved performance and decreased total cost of ownership associated with the installed cluster.

Consider Deploying a Small Pilot Cluster

Because it is difficult to predict hardware resource usage without benchmarking and testing, Hortonworks recommends installing a small pilot cluster for the testing and benchmarking of applications. Such a pilot cluster can yield valuable and reliable information used to determine hardware sizing requirements.

The best results are obtained by benchmarking your applications in the pilot cluster. If that is not possible, then run benchmark applications in the pilot cluster and use the results to extrapolate from that how your applications will perform. Terasort, DFSIO, and HiBench are commonly used but there are other benchmark utilities.

Even if a cluster is not initially optimally sized, the modular nature of Hadoop components make future cluster modification easier. A cluster can be resized by adding or removing nodes.

Hardware Guidelines for Master Nodes

Master nodes run master service components. As a result, availability is a primary concern. Availability is enhanced through redundancy. Where possible, all hardware components should be configured for redundancy.

Use RAID 10 storage for both the operating system and all data disks. Configure dual, bonded Ethernet NICs. Consider dual power supplies and cooling fans. Also use ECC-protected memory.

Some Hadoop master service components support high availability configurations across multiple hosts. You may even consider virtualizing the master servers to gain the benefits of live virtual machine migration or virtual machine high availability solutions like VMware HA.

While not all master service components have the same CPU, memory, storage, or network hardware requirements, consider using the same hardware specifications for all master nodes. This enables an administrator to more easily migrate master service components to any master node as a result of maintenance requirements or following a system failure.

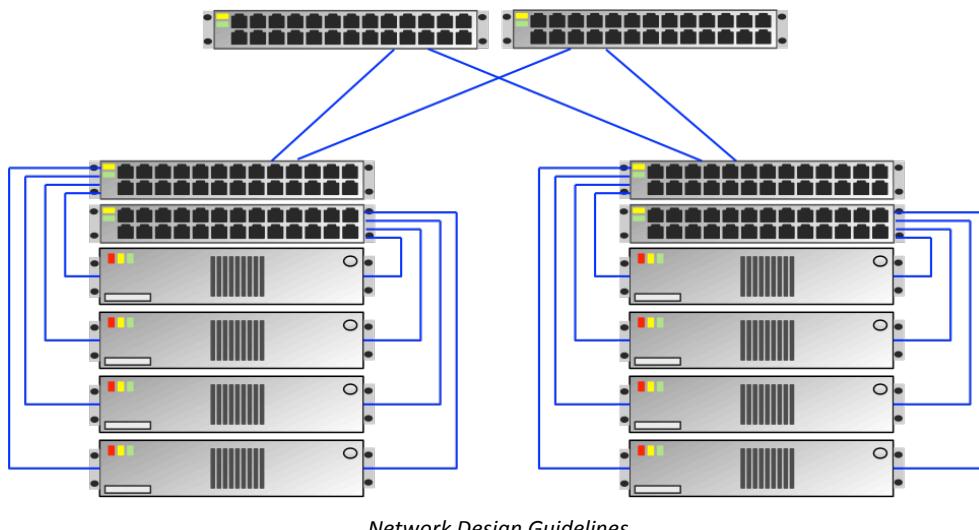
Hardware Guidelines for Worker Nodes

Worker nodes perform data processing so throughput is more important than availability. Worker nodes are already redundant within the cluster so hardware redundancy in the individual nodes is not as necessary. If a worker node fails, Hadoop automatically takes action to protect data and restart any failed processing jobs. For example, by default, HDFS maintains three copies of all data blocks and ensures that not all copies reside on the same worker node. If a worker node fails, HDFS automatically makes new copies of all data blocks that resided on the failed machine.

Because throughput is so important for performance, plan for parallel computing and data paths. This includes using dual-CPU socket servers, using multiple disk drives (as many as 8-12) and disk controllers, using fast drives with a fast disk interconnect, and using multiple, bonded Ethernet NICs.

To simplify cluster configuration, try to use the same hardware specification for all worker nodes. Many Hadoop configuration properties are tied to the number of CPUs, the amount of memory, or the number of disks on the worker node. If different groups of worker nodes have different specifications, then these groups of worker nodes must use separate configuration files with different configuration property settings. The good news is that the Ambari Configuration Groups feature accommodates this, but there is still some additional administrative effort involved.

Network Design Guidelines



Network availability and sufficient network bandwidth are critical for cluster operation.

To help avoid cluster failure, avoid single points of network failure. This means using dual, bonded network ports, dual top-of-the-rack switches, and dual core switches.

Network bandwidth is the most challenging parameter to estimate because Hadoop workloads greatly vary from cluster to cluster and even within the same cluster at different times. It has been typical to see dual 1 Gb Ethernet ports on the worker nodes, but you might need more. Using 10 Gb Ethernet ports helps to ensure that your network bandwidth will be sufficient well in the future, but is more expensive to purchase. In any case, to help ensure that your cluster receives all available network bandwidth you should dedicate the networks switches to the cluster.

You should also consider the effect of a worker node failure. HDFS maintains three copies of all data blocks and ensures that not all copies reside on the same worker node. If a worker node fails, HDFS automatically makes additional copies of all data blocks that resided on the failed machine. This can result in significant additional network traffic as many of these data blocks will have to be copied across the network. For example, if a worker node with 10 terabytes of data fails, the cluster will produce approximately 10 terabytes of network traffic to recover.

Ambari and Metrics Collector Hardware Guidelines

Number of Cluster Nodes	Memory Size Guideline (MB)	Disk Space (GB)
1	1024	10
10	1024	20
50	2048	50
100	4096	100
300	4096	100
500	8096	200
1000	12288	200
2000	16384	500

Memory and Disk Requirements Based on Cluster Size

The Ambari Server host does not require a large amount of memory. The minimum required is 1 gigabyte. However, the standalone Ambari Metrics Collector was added starting with Ambari 2.0.

The memory and storage requirements for the Metrics Collector are based on the number of cluster nodes. The table illustrates the memory and storage space requirements for various cluster sizes. The Ambari Server and Ambari Metrics Collector can be co-located on the same machine.

Hardware Testing

When purchasing a large number of systems, there is always the possibility that some hardware will fail early—often within a few hours of operation. If the cluster is already installed then the hardware failure will disrupt cluster operation. To help avoid this, hardware should always be thoroughly tested before being placed into service. Testing the hardware sufficiently takes many hours to complete so have your hardware provider do it, if possible.

If you have to perform the testing, then use a vendor-supplied diagnostic test utility where available. Another option is to install an operating system and use operating system utilities.

A few examples of Linux commands include `fio`, `dd`, and `hdparm`. The `fio` is the flexible I/O tester, `dd` is a disk-to-disk copy utility, and `hdparm` gets or sets SATA/IDE device parameters. Depending on the command and options used, these utilities can test various hardware subsystems. Use the operating system documentation to learn how to run these or other utilities.

Supported Operating Systems

The following 64-bit operating systems are tested and supported by Hortonworks:

- CentOS 6, 7
- Red Hat Enterprise Linux (RHEL) 6, 7
- Oracle Linux 6, 7
- SUSE Linux Enterprise Server (SLES) 11, SP3
- Debian 6, 7 (support in HDP 2.3.2 maintenance release)
- Ubuntu Precise 12.04, 14.04 (support in HDP 2.3.2 maintenance release)
- Windows Server 2008, 2012

Required Software Packages

The following software packages are required for a Hadoop cluster:

- `yum` (CentOS or RHEL)
- `zypper` (SLES)
- `php_curl` (SLES)
- `apt-get` (Ubuntu)
- `reposync`
- `rpm` (CentOS, RHEL, or SLES)
- `scp`
- `curl`
- `wget`
- `unzip`
- `chkconfig`
- `tar`
- Java software, one of the following:
 - Oracle JDK 1.8
 - Oracle JDK 1.7 u51 or higher
 - OpenJDK 1.8
 - OpenJDK 1.7 u51 or higher

OS Pre-Configuration

Required

There are several required configuration changes that must be completed before installing HDP.

- Configure NTP on all cluster nodes to ensure synchronized time.

- Configure all cluster nodes for forward and reverse DNS lookups.
- Configure the system that will run Ambari for password-less SSH access to cluster nodes. If this is not possible for security or other reasons, manually install and register the Ambari agents before HDP installation.
- Open HDP-specific network ports or disable the firewall.
- Disable IPv6 on all cluster nodes
- For the duration of the installation process disable Security Enhanced Linux (SELinux). It can be re-enabled following installation.

Instructions and examples of making these changes are provided by the Hortonworks online documentation at <http://docs.hortonworks.com>.

Recommended

There are a number of ways to tune operating systems to enhance cluster performance.

- Linux file systems record the last access time for all files but there is a small performance cost associated with this. To disable last access time recording, use the noatime option when mounting a file system. Use the instructions in your vendor documentation to add the noatime mount option.
- The ext3 and ext4 file systems normally reserve five percent of their disk space for the exclusive use of the root user. This can lead to an excessive waste of space with multiple terabyte-sized storage. You may disable or lower this reservation when creating a file system, or afterwards by tuning the file system. Use the instructions in your vendor documentation to change the root-reserved space.
- Linux kernels have a feature named transparent huge pages that is not recommended for Hadoop workloads. Use the instructions in your operating system documentation to disable it.
- Ethernet jumbo frames increase an Ethernet packet's maximum payload from 1500 bytes to approximately 9000 bytes. This payload increase increases network performance. Use the instructions in your vendor documentation to enable jumbo frames.
- BIOS-based power management commonly has the ability to increase or decrease CPU clock speeds under certain conditions. Because Hadoop operates as a cluster of machines, having some machines running slower clock speeds can have an adverse affect on total cluster processing throughput. To avoid this, use the instructions in your vendor documentation to disable BIOS-based power management.
- Linux systems also place limits on the total number of files that a process may have open at the same time. It also places limits on the total number of processes that a user may run at the same time. These limits could interfere with cluster operation. Use the instructions in your vendor documentation to increase these limits, as necessary.

For a complete list of OS preconfiguration, refer to the online documentation at <http://docs.hortonworks.com>.

Supported Databases

Database	Ambari	Hive	Oozie	Ranger
Derby			default	
MySQL 5.6	✓	default	✓	✓
Oracle 11g r2	✓	✓	✓	✓
PostgreSQL 8.x, 9.3+	default	✓	✓	
SQL Server 2008 R2+	✓	✓	✓	

Supported Databases

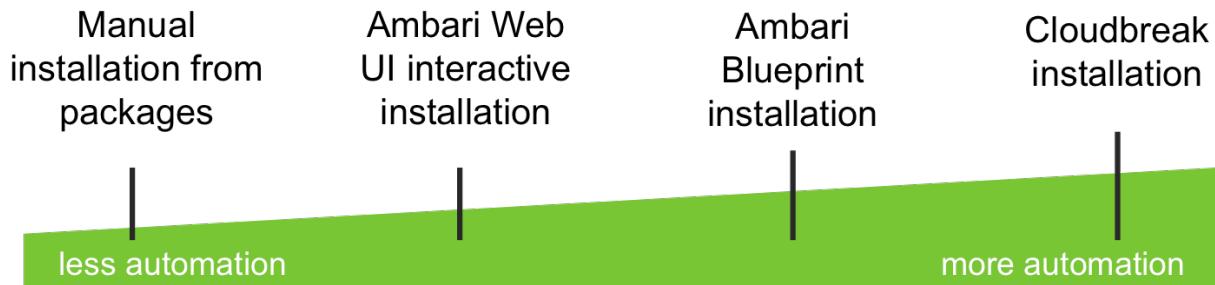
Some Apache frameworks require a database to maintain configuration information and metadata. The table lists the default database for each framework along with any other databases that are supported by the framework. Default databases are automatically installed when installing HDP using Ambari. If you have an existing database and you would like to connect to it during HDP installation, you must perform specific installation configuration steps. These steps are listed in the online HDP manual installation documentation.

To ease administration, consider choosing a single database type for all frameworks. Database administrators should implement high availability and regularly back up the databases. Heavy use of Falcon plus Oozie, or Ambari, might require dedicated instances.

Note: Derby, the default Oozie database, is not recommended for production use.

For more information about installing and configuring databases for individual software platforms, refer to the online document *Non-Ambari Cluster Installation* at <http://docs.hortonworks.com>.

The Ambari Installation Process



HDP Installation Methods

There are four primary, documented HDP installation methods.

Manual installation from software packages is the oldest and most manual method. This method is potentially the most time consuming and error prone. It also requires more technical expertise than the other methods. However, because every step is manual, you have very fine-grained control over the installation process and the final configuration of the cluster.

The Ambari Web UI interactive installation employs a wizard to guide the user through the installation process. While the user still has to make installation decisions, much of the more mundane work is automatically performed by Ambari. This is the method that will be described in this lesson.

An **Ambari Blueprint** installation is an unattended installation. The operator performing the installation does not have to make any configuration decisions. All of the work is done prior to the installation when the blueprint—an installation configuration template—is created. Ambari Blueprints is described in another course.

Cloudbreak is the most recent installation method. It uses Ambari Blueprints, customizable scripts, and a graphic user interface to perform cloud-based installations. It has different built-in blueprints intended to support different workload types. Custom blueprints can be added.

It can deploy clusters in Microsoft Azure, Amazon Web Services, Google Cloud Platform, and Openstack. It can also deploy clusters into platforms that support Docker containers. It also can use user-defined policies to autoscale—automatically expand and contract—existing clusters.

Ambari Interactive Installation Overview

- 1) Pre-installation steps – prepare the base operating systems for HDP
- 2) Installation steps using the Ambari Web UI:
 - a) Install the Ambari software.
 - b) Start the Ambari Server.
 - c) Log in to the Ambari Web UI.
 - d) Use the Ambari installation wizard to:
 - i) Install Ambari agents on the nodes
 - ii) Register Agents with the Ambari Server
 - iii) Define the Hadoop service components to install to each node
 - iv) Use Ambari agents to install service components on the nodes
 - v) Validate the installation

This is an overview of the interactive installation process before getting into all of the specific details.

The first step is to perform the pre-installation steps. Most of these steps have already been described and include such things configuring NTP, setting up password-less SSH access from the Ambari Server to the cluster nodes, disabling SELinux, and so on. The exact steps will vary based on variables like the cluster size, hardware type and configuration, and operating system type.

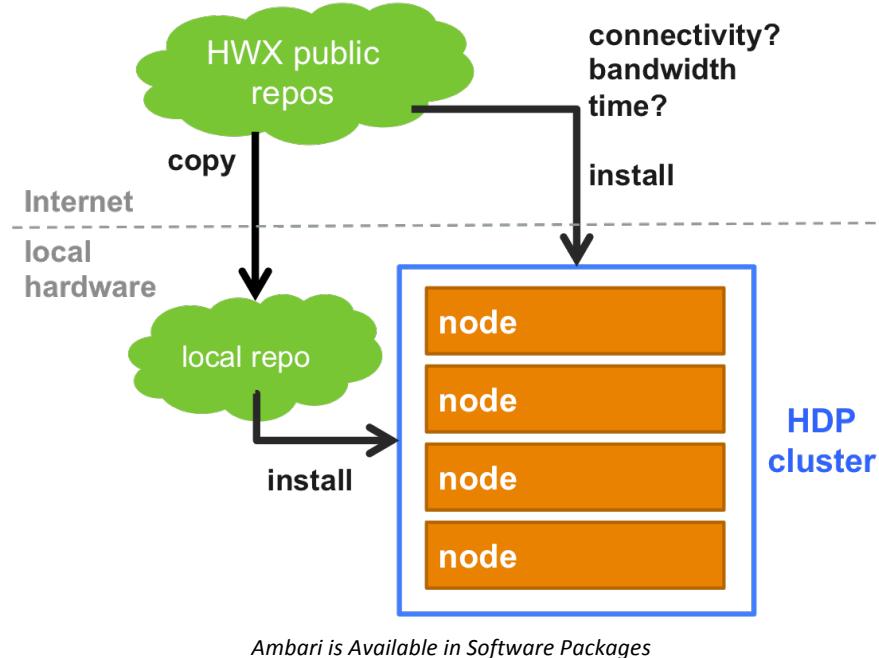
Once the hardware and base operating systems have been prepared, it is time to install the Ambari Server software. Ambari software download and installation only takes a few minutes.

After Ambari has been installed, you must start the Ambari Server.

With the Ambari Server running, open a browser on your desktop and use it to log in to the Ambari Server. The Ambari Server detects that no cluster is available and automatically presents the choice to launch the interactive installation wizard.

To install a new cluster, the Ambari Server must first use password-less SSH to install Ambari agents on the cluster nodes. The agent on each node then registers with the Ambari Server. Once agent registration is complete, Ambari can use the agents to install HDP cluster software to the appropriate nodes. The user defines which software components are installed to which nodes. When installation is complete, Ambari validates the operation of each cluster service. Ambari and its documentation refer to this as “smoke tests”.

Software Repositories



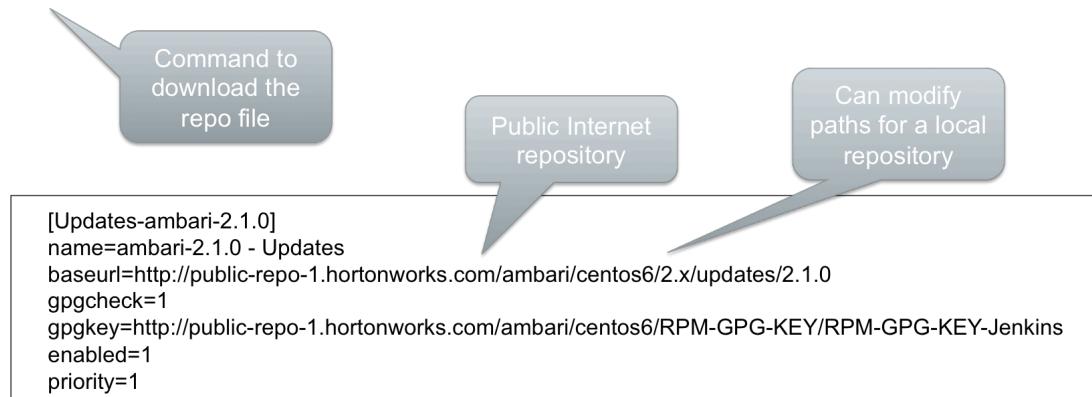
To install Ambari, or other HDP software, requires access to the software packages. These software packages are maintained by Hortonworks in public software repositories. For convenience, you can access these public repositories and directly install a cluster from them. However, there are at least two potential considerations with this approach.

First, it assumes that all cluster nodes and the Ambari Server have access to these repositories. This is often not the case due to an organization's security policies and protections. Second, installing directly from these remote public repositories can take significant time. How much time depends on the number of nodes to install and how much network bandwidth is available.

To mitigate these problems, it is possible to copy the software from the public repositories to a local machine. The local machine is configured as a software repository and the Ambari Server and cluster nodes are configured to go to the local repository rather than the public repository.

Example CentOS Ambari Repo File

```
wget -nv http://public-repo-1.hortonworks.com/ambari/centos6/2.x/updates/2.1.0/ambari.repo -O /etc/yum.repos.d/ambari.repo
```



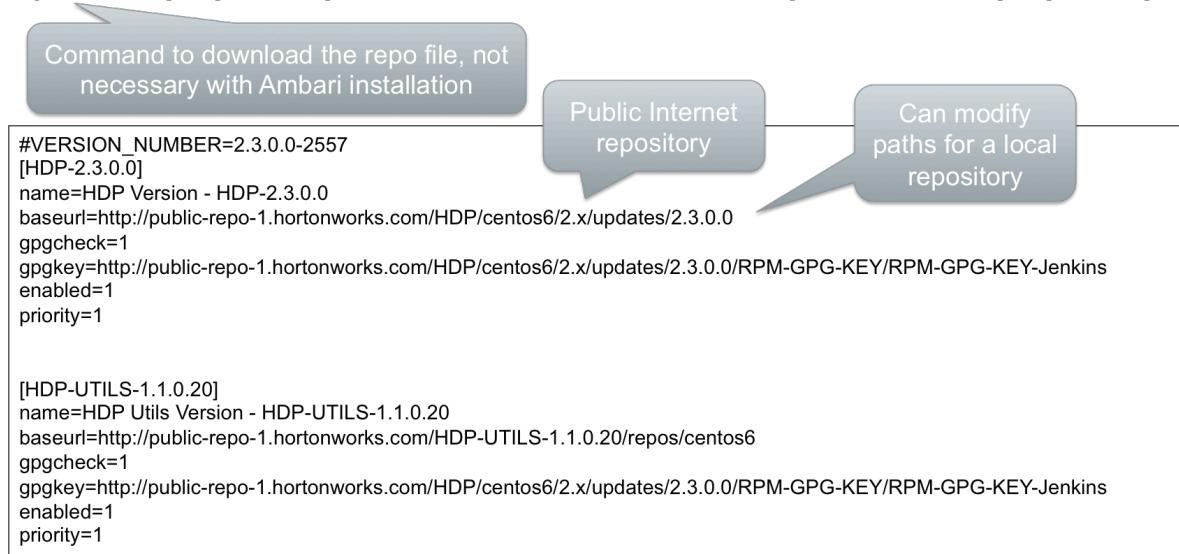
CentOS Ambari Repo File

This is an example of a CentOS repo file that configures a system with the location to access, download, and install the Ambari software packages. Similar repo files are available for all supported operating systems and versions. Refer to Hortonworks documentation at <http://docs.hortonworks.com> for information about where to go to download repo files.

If you have created a local repository by following the instructions in the Hortonworks documentation, you may modify the baseurl and gpgkey URLs to point to your local repository.

Example CentOS HDP Repo File

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.3.0.0/hdp.repo -O hdp.repo
```

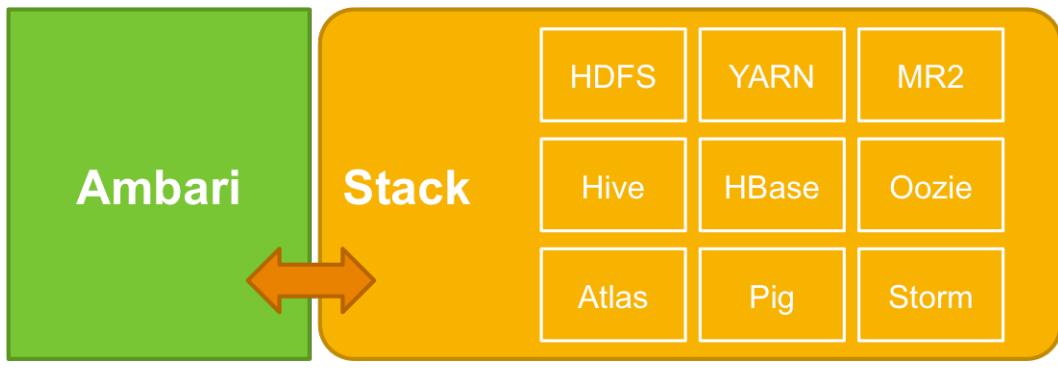


CentOS HDP Repo File

This is an example of a CentOS repo file that configures a system with the location to access, download, and install the HDP and HDP-UTILS software packages. Similar repo files are available for all supported operating systems and versions. Refer to Hortonworks documentation at <http://docs.hortonworks.com> for information about where to go to download repo files.

If you have created a local repository by following the instructions in the Hortonworks documentation, you may modify the baseurl and gpgkey URLs to point to your local repository.

Ambari Stacks



Ambari Stacks

Ambari uses the concept of a Stack to install software. A Stack and its associated services are defined in a Stack definition. A Stack defines a set of services and where to obtain the software packages for those services. A Stack can have one or more versions, and each version can be active or inactive. For example, the HDP Stack used by Ambari can include versions 2.2 and 2.3, but only one of those versions is active on a cluster node at any one time. The concept of active and inactive versions is tied to the upgrade process. Upgrading is described in another course.

A service defines a set of components that make up the service. For example, the NameNode and DataNode are components of the HDFS service. The ResourceManager and NodeManager are components of the YARN service.

Components have a defined lifecycle. For example, they can be installed, started, stopped, and restarted.

By leveraging the Stack definition, Ambari has a consistent and defined interface to install, manage, and monitor a set of services. Ambari is also easily extensible by adding new Stack versions with new services.

Installing Ambari

Once an appropriately modified repo file is in place, you can download, install, and start the Ambari Server.

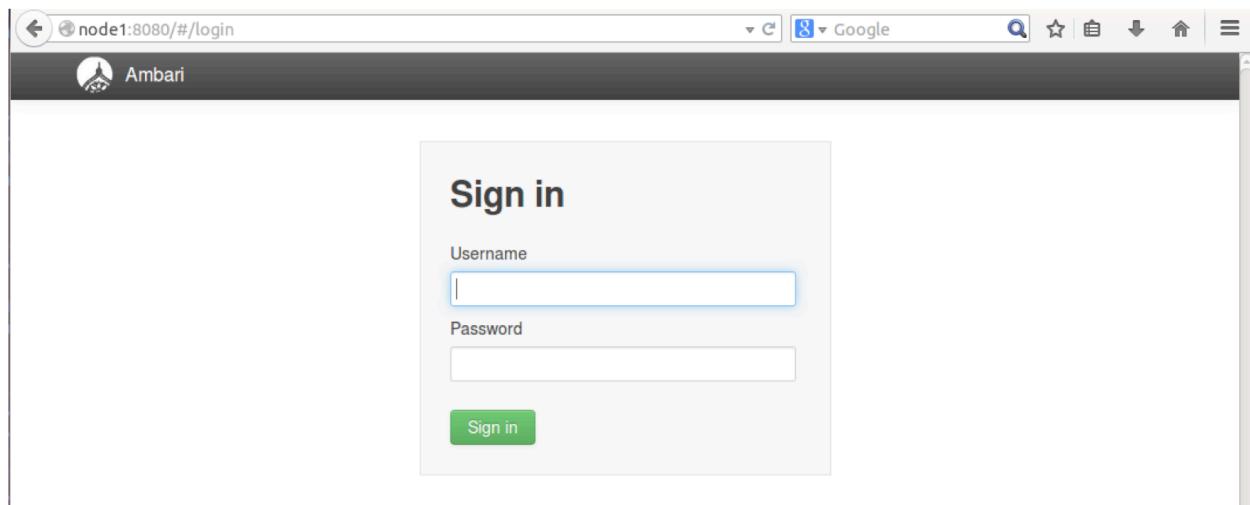
Note

The command examples below are from a CentOS system.

- 1) Download and install the Ambari Server software
 - a) For example, the `yum -y install ambari-server` command references the information in the repo file and uses it to download and install the Ambari software. The Ambari software package is over 300 MB so this process normally takes a few minutes when connected to the Hortonworks public repository.
- 2) Next, the Ambari Server must be configured and its database must be initialized by running the `setup` command.

- a) This is a one-time operation. The `ambari-server setup -s` option silently installs Ambari using all of the default configuration settings. Normally the `ambari-server setup` command interactively prompts for information as it runs. This allows the administrator to make choices about how Ambari is configured. The silent option instructs setup to accept all the default choices during the initialization.
- 3) After initialization is complete, start the Ambari Server using the command `ambari-server start`.
- a) If you would like the Ambari Server to start every time the system is rebooted, add the `start` command to the operating system's start-up scripts.
- 4) The Ambari Server includes a Web server component. To log in to the Ambari Server open a browser and enter the URL `http://<Ambari_server_hostname>:8080`. The default user name and password are admin and admin.
- 5) Hortonworks documentation includes information about how to configure the Ambari Server for secure operation using a security certificate. If Ambari is configured for secure operation, the URL uses https rather than http.

Ambari Web UI Log In

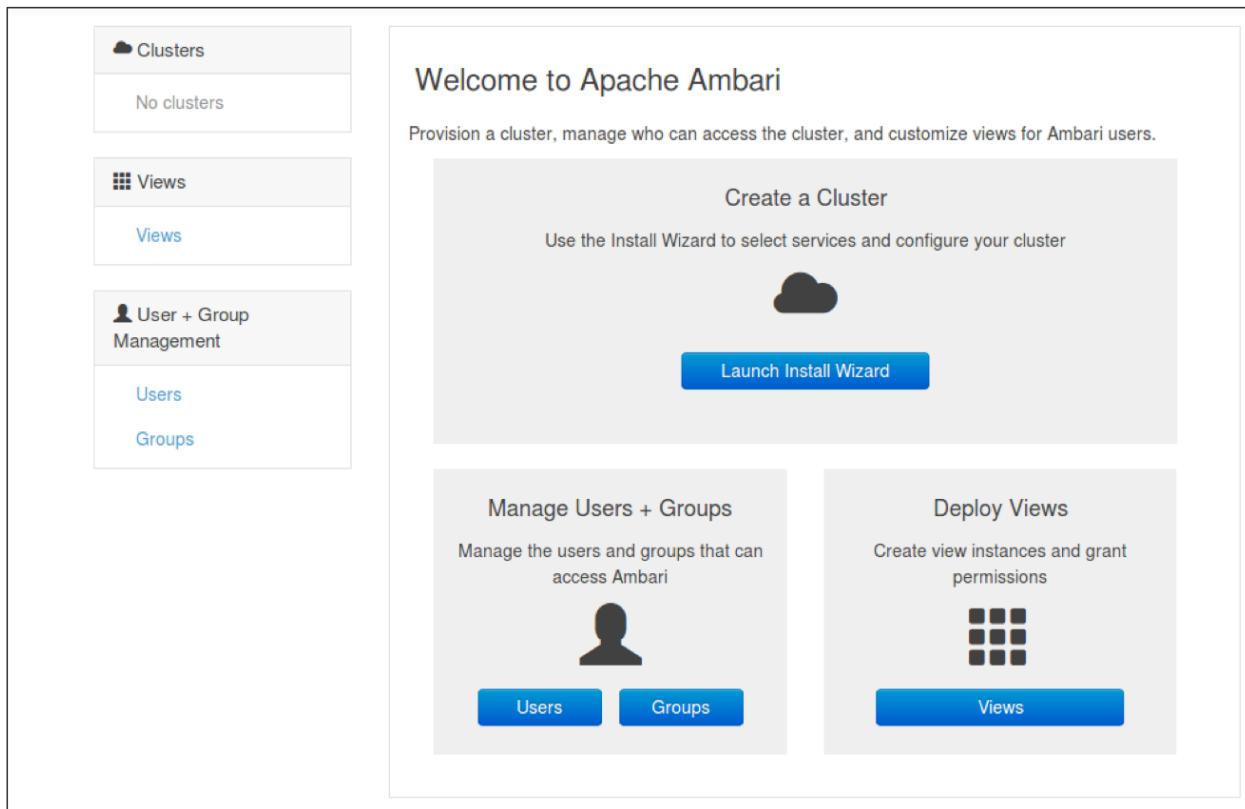


Ambari Web UI Login Screen

The default user name and password are admin and admin.

Installing the Hortonworks Data Platform

Welcome Screen

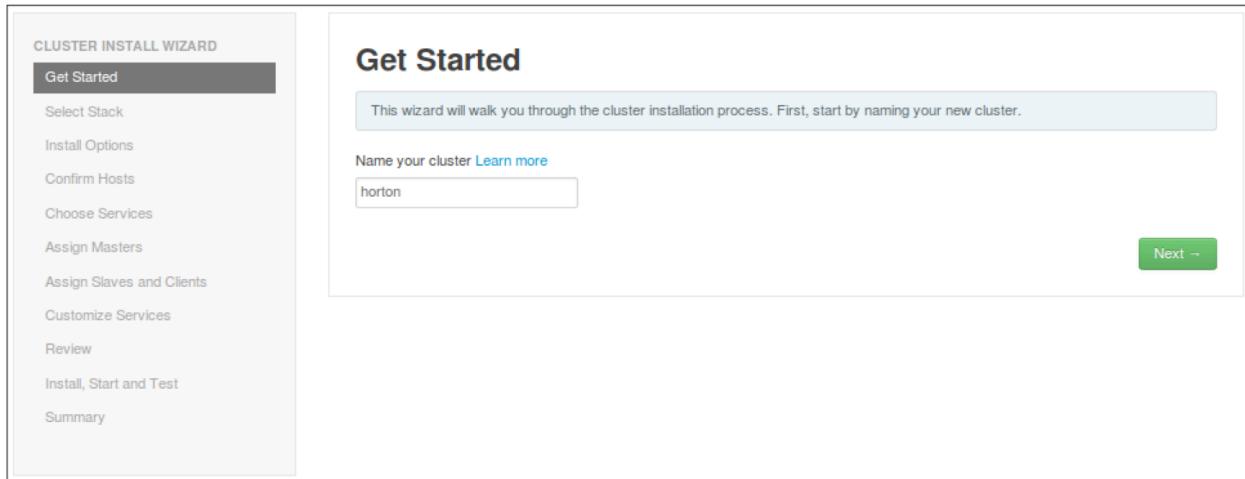


Ambari Web UI Welcome

The first time you log in to the Ambari Web UI the Welcome to Ambari window opens. In the upper-left corner Ambari notifies the user that there is no installed cluster.

Click **Launch Install Wizard** to start the installation process.

Naming Your Cluster



Ambari Web UI Get Started

Ambari must have a name for the cluster it manages.

Type that name in the Get Started window and click **Next**.

A cluster can be renamed after an installation.

Selecting the Stack Version

The screenshot shows the 'Select Stack' step of the Ambari Cluster Install Wizard. On the left, a sidebar lists steps: Get Started (selected), Select Stack (highlighted in grey), Install Options, Confirm Hosts, Choose Services, Assign Masters, Assign Slaves and Clients, Customize Services, Review, Install, Start and Test, and Summary. The main panel title is 'Select Stack' with the sub-instruction 'Please select the service stack that you want to use to install your Hadoop cluster.' Below is a 'Stacks' section with radio buttons for HDP 2.3 (selected), HDP 2.2, HDP 2.1, and HDP 2.0. A 'Back' button is at the bottom left, and a 'Next →' button is at the bottom right.

Ambari Web UI Select Stack

Select the HDP stack version to download. In this example, HDP 2.3 has been selected. Post installation, newer Stacks can be downloaded by Ambari and used to upgrade a cluster. Rolling upgrades are described in another course.

Customize Repositories

OS	Name	Base URL
redhat6	HDP-2.3	http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.3.0.0
	HDP-UTILS-1.1.0.20	http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/centos
redhat7	HDP-2.3	http://public-repo-1.hortonworks.com/HDP/centos7/2.x/updates/2.3.0.0
	HDP-UTILS-1.1.0.20	http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/centos
suse11	HDP-2.3	http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/updates/2.3.0.0
	HDP-UTILS-1.1.0.20	http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/suse11

Skip Repository Base URL validation (Advanced) [?](#)

Ambari Web UI Advanced Repository Options

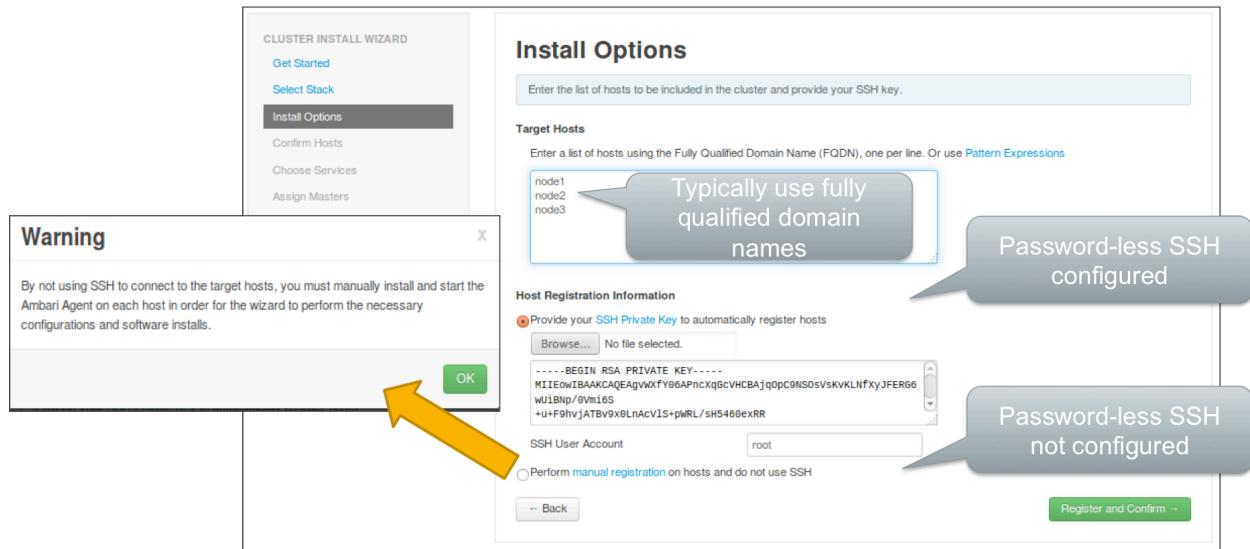
The Select Stack window includes an Advanced Repository Options section, which is shown here. The URLs listed here instruct Ambari where to go to access the HDP software packages for different operating systems. By default, the URLs point to the public, Internet-accessible Hortonworks repositories.

Use the check boxes to choose the operating systems that match the operating systems running on your cluster nodes. Using a consistent operating system across cluster nodes is highly recommended to simplify cluster administration.

The URLs to the software packages can be updated. For example, if you have created a local HDP software repository then you may update the URLs to point to the local repository.

Repository URLs can be updated following installation. This is useful if you move or modify a local repository and later need to use Ambari to reinstall existing cluster nodes or expand the cluster using new nodes.

Choose Nodes for Ambari Agents



Ambari Web UI Choosing Nodes for Ambari Agents

Ambari agents must be installed on each cluster node and registered with the Ambari Server before the Ambari Server can install HDP on the cluster. Type the fully-qualified domain name (FQDN) of each host in the cluster in the Target Hosts text box. If you choose to use short host names instead, the installer will open a warning window when you click Register and Configure. Use of FQDNs is highly recommended to ensure proper cluster operation.

Ambari can install additional cluster nodes after installation is complete. Expanding a cluster by adding new nodes is a common operation. Adding new nodes is described in another lesson.

The Host Registration Information section includes two radio buttons. Which radio button to select depends on whether or not you have pre-configured the Ambari Server machine with password-less SSH access to the cluster nodes. If password-less SSH has been pre-configured, select the first radio button and click **Browse**. Use the Browse window to locate the pre-configured SSH RSA private key file that enables the Ambari Server to log in to each cluster node. The Hortonworks online documentation include instructions for configuring password-less SSH log in.

If password-less SSH log in has not been pre-configured, select the second radio button. This opens a window warning you that you must manually install and configure Ambari agent software on each cluster node. Once installed and properly configured, an agent will automatically register with the Ambari Server.

After the choices have been made, click **Register and Confirm**.

Install and Register Ambari Agents

Ambari Web UI Confirm Hosts

The Confirm Hosts window enables you to monitor the progress of the Ambari agent installation and registration process. Each node with a successful agent registration displays Success and a green progress bar. If you decide not to include one or more nodes in the cluster, you may click Remove next to that host before proceeding.

Ambari performs configuration checks on each node once an agent has been installed. If Ambari detects any potential configuration problems they can be viewed by clicking the link **Click here to see the warnings**. A window opens with detailed information about potential issues. Some warnings are serious and must be resolved before proceeding while some warnings can be potentially ignored. In each case, a warning must be evaluated for its impact on cluster operation.

When ready to proceed with the installation process, click **Next**.

Choose Services to Install

Ambari Web UI Choose Services

Use the check boxes in the Choose Services window to select the services to install in the cluster. Any service not selected during installation can be installed later.

Click **Next** when ready to continue.

Assign Masters

Ambari Web UI Assign Masters

Once Ambari has been configured with a list of cluster nodes and the services to install, Ambari provides an initial layout of service master components. Use the Assign Masters window either to confirm or move the master components.

This is when information about workloads and resource capacity gained from pre-planning, or pre-testing with a pilot cluster, is very helpful. The resource capacity of each host must be considered along with service redundancy. For example, the primary and secondary HDFS NameNodes should run on two separate machines. You should run a ZooKeeper server across three to five separate machines.

Use the drop-down menu next to each component to move the master component to another cluster node, as necessary.

Click **Next** to proceed.

Installing the Hortonworks Data Platform

Assign Slaves and Clients

The screenshot shows the 'Assign Slaves and Clients' step of the Cluster Install Wizard. On the left, a sidebar lists steps: Get Started, Select Stack, Install Options, Confirm Hosts, Choose Services, Assign Masters, Assign Slaves and Clients (which is selected and highlighted in grey), Customize Services, Review, Install, Start and Test, and Summary. The main panel title is 'Assign Slaves and Clients'. It contains a brief description: 'Assign slave and client components to hosts you want to run them on. Hosts that are assigned master components are shown with *.' Below this is a table where hosts (node1*, node2*, node3*) are mapped to various services. The table has columns for Host, DataNode (checkboxes for all or none), NFSGateway (checkboxes for all or none), NodeManager (checkboxes for all or none), and Client (checkboxes for all or none). All checkboxes for DataNode, NodeManager, and Client are checked for all three hosts. At the bottom right of the table are buttons for 'Show' (set to 25), '1 - 3 of 3', and navigation arrows.

Ambari Web UI Assign Slaves and Clients

Use the Assign Slaves and Clients window to choose where to run service worker components and Hadoop client software.

The DataNode is the HDFS service worker component. An HDFS NFS Gateway machine can be used as a method to access HDFS. The HDFS NFS Gateway is described in another lesson. The NodeManager is the YARN service worker component. Client represents the Hadoop client software. Client software is used by user and application clients to access cluster services and resources. For example, client software is used to access HDFS storage. Client software is commonly installed on cluster nodes. Client software is also commonly installed on utility machines used as gateway machines to access cluster resources.

Make the required selections and click **Next**.

Customize Services

The screenshot shows the 'Customize Services' step of the Cluster Install Wizard. The sidebar is identical to the previous screenshot. The main panel title is 'Customize Services'. A message says: 'We have come up with recommended configurations for the services you selected. Customize them as you see fit.' Below this is a tabs menu: HDFS, MapReduce2, YARN, Tez, Hive (selected), Pig, ZooKeeper, Ambari Metrics, Slider, and Misc. Underneath is a group dropdown set to 'Hive Default (3)', a 'Manage Config Groups' button, and a 'Filter...' button. There are two tabs: 'Settings' (selected) and 'Advanced'. A section titled 'Hive Metastore' is expanded, showing 'Hive Metastore hosts' set to 'node1' and 'Database Type' set to 'MySQL'.

Ambari Web UI Customize Services

Hadoop services must typically be customized for each specific cluster installation. Ambari will do some limited customization based on the choices made during installation. For example, configuration properties that include the host name of the HDFS NameNode will be updated by Ambari based on your choice of a NameNode host in the earlier Assign Masters window.

In addition to the customizations made by Ambari, the Customize Services window enables a user to customize a myriad of configuration settings. Again, this is when information about workloads and resource capacity gained from pre-planning, or pre-testing with a pilot cluster, is very helpful.

Depending on the services selected in the earlier Choose Services window, Ambari might display red alert icons. In the example here, Ambari is alerting the user that the Hive service is missing a required configuration setting. In this case, scrolling down in the window reveals that the Hive service requires a database password for the metadata database it installs during installation.

Configuration Warnings

The screenshot shows a 'Configurations' dialog box. At the top, a message says: 'Some service configurations are not configured properly. We recommend you review and change the highlighted configuration values. Are you sure you want to proceed without correcting configurations?' Below this, a table lists configuration details for the HIVE service. One row is highlighted in yellow, indicating a warning: 'hive.auto.convert.join.noconditionaltask.size' with value '357913592'. The 'Description' column for this row states: 'Value is less than the recommended default of 357913941. If hive.auto.convert.join.noconditionaltask is off, this parameter does not take affect. However, if it is on, and the sum of size for n-1 of the tables/partitions for a n-way join is smaller than this size, the join is directly converted to a mapjoin (there is no conditional task).'. At the bottom of the dialog, a note says: 'Not enough physical RAM on the host node1. At least 8580 MB is recommended based on components assigned.' On the right side of the dialog are 'Cancel' and 'Proceed Anyway' buttons.

Service	Property	Value	Description
HIVE	hive.auto.convert.join.noconditionaltask.size	357913592	Value is less than the recommended default of 357913941. If hive.auto.convert.join.noconditionaltask is off, this parameter does not take affect. However, if it is on, and the sum of size for n-1 of the tables/partitions for a n-way join is smaller than this size, the join is directly converted to a mapjoin (there is no conditional task).

Ambari Web UI Configurations

Before proceeding with the installation, Ambari will warn of any configuration issues. You may resolve the issue before proceeding or you may choose to proceed anyway. Use caution when choosing to proceed anyway.

Review and Deploy

Review

Please review the configuration before installation

Admin Name : admin
Cluster Name : horton
Total Hosts : 3 (3 new)

Repositories:

- redhat6 (HDP-2.3):
<http://s3.amazonaws.com/dev.hortonworks.com/HDP/centos6/2.x/BUILDS/2.3.0.0-2410>
- redhat6 (HDP-UTILS-1.1.0.20):
<http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.20/repos/centos6>

Services:

HDFS

- DataNode : 3 hosts
- NameNode : node1
- NFSGateway : 0 host
- SNameNode : node2

YARN + MapReduce2

- App Timeline Server : node1
- NodeManager : 3 hosts
- ResourceManager : node1

Tez

Clients : 2 hosts

Back **Print** **Deploy →**

Ambari Web UI Review

The Review window allows you to print the configuration and deploy a cluster.

Click Deploy when ready to install the software.

Install, Start, and Test

Install, Start and Test

Please wait while the selected services are installed and started.

Host	Status	Message
node1	4%	Installing App Timeline Server

4 % overall

Show: All (1) | In Progress (1) | Warning (0) | Success (0) | Fail (0)

-- truncated --

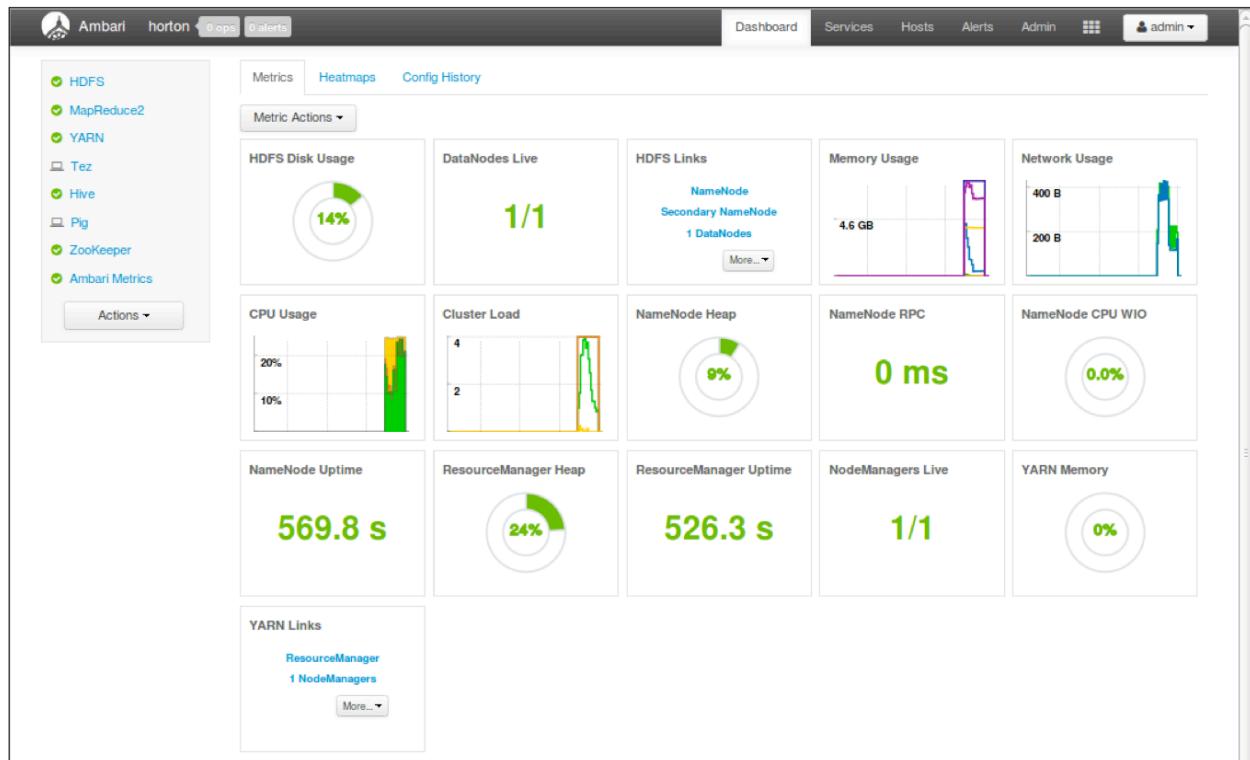
Review
Install, Start and Test (highlighted)
Summary

Next →

Ambari Web UI Install, Start, and Test

The Install, Start, and Test window displays the progress of each cluster node. This window was truncated for space reasons. If any node fails to install properly, the error displayed in the Message column is a hypertext link. Click the link to get more detailed error information. If you can resolve the issue, you can use Ambari to attempt to install the node again.

The Ambari Dashboard



Ambari Dashboard

When installation completes Ambari automatically opens to the Ambari Dashboard window.

Knowledge Check Questions

- 1) True or false? HDP supports both on-premise and cloud deployments.
- 2) Which of the three Hadoop deployment modes is commonly used for production environments?
- 3) What is an advantage of starting with a small pilot cluster?
- 4) What is a key consideration when planning for master nodes?
- 5) What is a key consideration when planning for worker nodes?
- 6) What are key considerations when planning the network?
- 7) Ambari Metrics Collector memory and storage requirements vary based on ____?
- 8) True or false? You cannot install HDP unless password-less SSH access is possible between Ambari and the cluster nodes.
- 9) True or false? Firewalls must be disabled during HDP installation.
- 10) Name the four installation options listed in this lesson.
- 11) From the perspective of installation, why is it important for an Ambari agent to be installed and registered on each cluster node?
- 12) What should be considered when choosing to install HDP directly from the public software repositories?
- 13) After installation, will an Ambari Server automatically restart after each reboot?
- 14) What is the default port number used to log in to the Ambari Server Web UI?
- 15) A Stack defines a set of services, and a service defines a set of _____.
- 16) Components can be _____, _____, _____, and _____.

Knowledge Check Answers

- 1) True or false? HDP supports both on-premise and cloud deployments.

True

- 2) Which of the three Hadoop deployment modes is commonly used for production environments?

Distributed Mode

- 3) What is an advantage of starting with a small pilot cluster?

A pilot cluster may be used for benchmark testing to help determine the proper size of hardware resources

- 4) What is a key consideration when planning for master nodes?

Availability

- 5) What is a key consideration when planning for worker nodes?

Throughput

- 6) What are key considerations when planning the network?

Avoiding single points of failure and ensuring adequate bandwidth

- 7) Ambari Metrics Collector memory and storage requirements vary based on Number of nodes in a cluster?

- 8) True or false? You cannot install HDP unless password-less SSH access is possible between Ambari and the cluster nodes.

False - Installation is still possible but Ambari agents must be manually installed and registered with the Ambari Server

- 9) True or false? Firewalls must be disabled during HDP installation.

False - Firewall may be disabled or they may be configured to allow HDP network traffic through specific ports.

- 10) Name the four installation options listed in this lesson.

Manual installation from packages, Ambari Web UI interactive installation, Ambari Blueprints installation, Cloudbreak installation

- 11) From the perspective of installation, why is it important for an Ambari agent to be installed and registered on each cluster node?

Because it is the agent that performs the HDP software installation and configuration under the control of the Ambari Server

12)What should be considered when choosing to install HDP directly from the public software repositories?

Do all nodes have connectivity? How many nodes are there and is there enough network bandwidth to complete the installation in a reasonable time?

13)After installation, will an Ambari Server automatically restart after each reboot?

No. You would have to add the start command to a start-up script.

14)What is the default port number used to log in to the Ambari Server Web UI?

Port 8080

15)A Stack defines a set of services, and a service defines a set of ***Components***.

16)Components can be ***Installed***, ***started***, ***stopped***, and ***restarted***.

Summary

- HDP may be installed on-premises or in the cloud on either Linux or Windows hosts.
- Distributed mode deployment is recommended for production environments.
- Consider using a small pilot cluster to help size your production cluster machines.
- Availability is key for master nodes, throughput is key for worker nodes.
- HDP includes four main installation methods.
- You may use either the public or your own local repositories to install HDP.
- A Stack defined a set of services, and a service defines a set of master and worker components.
- Ambari installs a Stack on the cluster nodes after installing and registering an agent on each node.

Classes Available Worldwide Through Our Partners



Study Options Worldwide

In combination with our partner providers, classes are often available in numerous locations across the world.



Private On-site Training

Hortonworks training in-house covers all of our basic coursework, and provides a more intimate setting for 6 or more students.

[Contact us for more details](#)



Learn from the company focused solely on Hadoop.



What Makes Us Different?

1. Our courses are designed by the **leaders and committers** of Hadoop
2. We provide an **immersive** experience in **real-world** scenarios
3. We prepare you to **be an expert** with highly valued, **fresh skills**
4. Our courses are available **near you**, or accessible **online**

Hortonworks University courses are designed by the leaders and committers of Apache Hadoop. We provide immersive, real-world experience in scenario-based training. Courses offer unmatched depth and expertise available in both the classroom or online from anywhere in the world. We prepare you to be an expert with highly valued skills and for Certification.