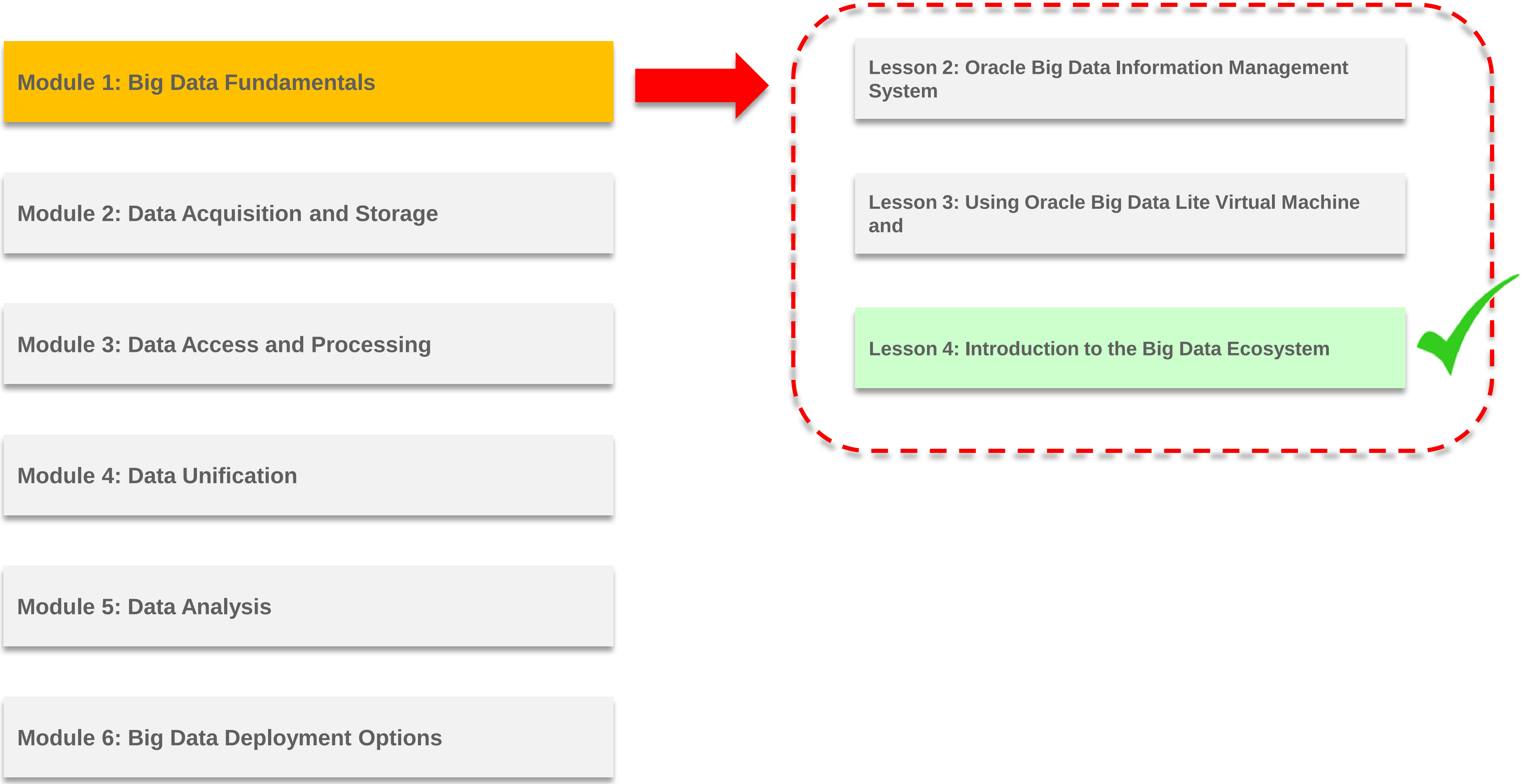
Introduction to the Big Data Ecosystem

Course Road Map



Module 1: Big Data Fundamentals

Module 2: Data Acquisition and Storage

Module 3: Data Access and Processing

Module 4: Data Unification

Module 5: Data Analysis

Module 6: Big Data Deployment Options

Lesson 2: Oracle Big Data Information Management System

Lesson 3: Using Oracle Big Data Lite Virtual Machine and

Lesson 4: Introduction to the Big Data Ecosystem ✓

Objectives

After completing this lesson, you should be able to:

- Define Hadoop and the *Hadoop Ecosystem*
- List the Hadoop core components
- Choose a Hadoop Distribution
- List some of the other related projects in the Hadoop Ecosystem

Computer Clusters

- A computer rack (commonly called a rack) is a metal frame used to hold various hardware devices such as servers, hard disk drives, and other electronic equipment.
- A computer cluster is a single logical unit consisting of multiple computers (or racks) that are linked through a fast local area network (LAN).
- The components of a cluster, nodes (computers used as a servers), run their own instance of an operating system.
- A node typically includes CPU, memory, and disk(s) storage.

Distributed Computing

- Distributed computing is a technique that allows individual computers to be networked together.
- A distributed file system is a client/server application that allows clients to access and process data stored on the server as if it were stored on their own computer.
- File systems that manage the storage across a network of machines are called distributed file systems.

Apache Hadoop

- Apache Hadoop:
  - Is an open-source software framework for ***Distributed Storage*** and ***Distributed Processing*** of big data on clusters of commodity hardware
  - Is a batch and interactive data-processing system for enormous amounts of data
- Open source available:
  - From the Apache Hadoop Foundation
  - As distributions, such as:
    - Cloudera's Distribution Including Apache Hadoop (CDH)
    - Hortonworks (HDP)
    - MapR

Types of Analyses That Use Hadoop

- *Market analysis*
- *Product recommendations*
- Demand forecasting
- *Fraud detection*
- Text mining
- *Index building*

- Graph creation and analysis
- Pattern recognition
- Collaborative filtering
- Prediction models
- Sentiment analysis
- Risk assessment

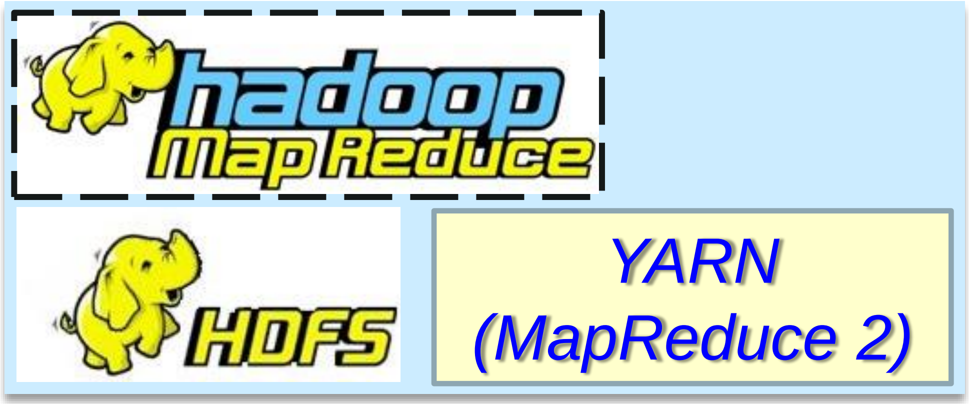
Types of Data Generated

- Financial transactions
- Sensors data
- Server logs
- Analytics
- Email and text messages
- Social media

Apache Hadoop Core Components

A Hadoop cluster has:

- Distributed data using Apache Hadoop Distributed File System (HDFS)
- Distributed processing using one of the following:
  - Yet Another Resource Negotiator (YARN) MR2, an extensible framework job scheduling and cluster resource management
  - MapReduce Framework (MR1)



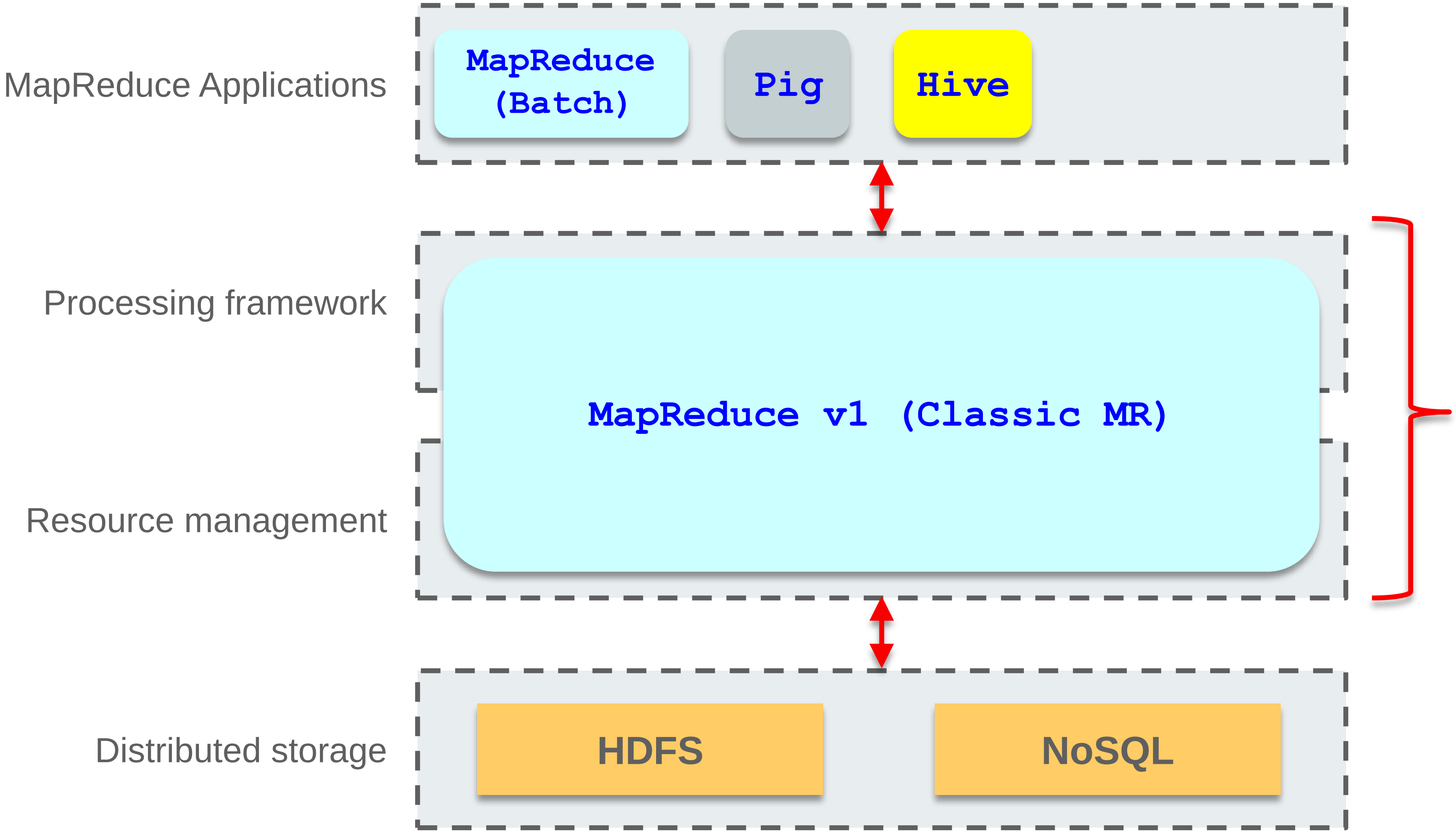
Apache Hadoop Core Components: HDFS

- Master-slave architecture
- Based on Google's File System (GFS) paper
- Stores and distributes data across the nodes in the cluster as data is loaded
- Redundant (reliability)
- Fault tolerant (high availability)
- Scalable (out instead of up)

Apache Hadoop Core Components: MapReduce Framework (MRv1)

- Is a programming model or framework for distributed computing
- Schedules and monitors tasks, and re-executes failed tasks
- Uses master-slave architecture
- Integrates with HDFS to provide the exact same benefits for distributed parallel data processing on the cluster
- ***Sends computations where the data is stored on local disks (data locality)***
- Hides complex "housekeeping" and distributed computing complexity tasks from the developer
- Supports only MapReduce applications

Running Applications Before Hadoop 2.x with MapReduce 1 (MR 1)



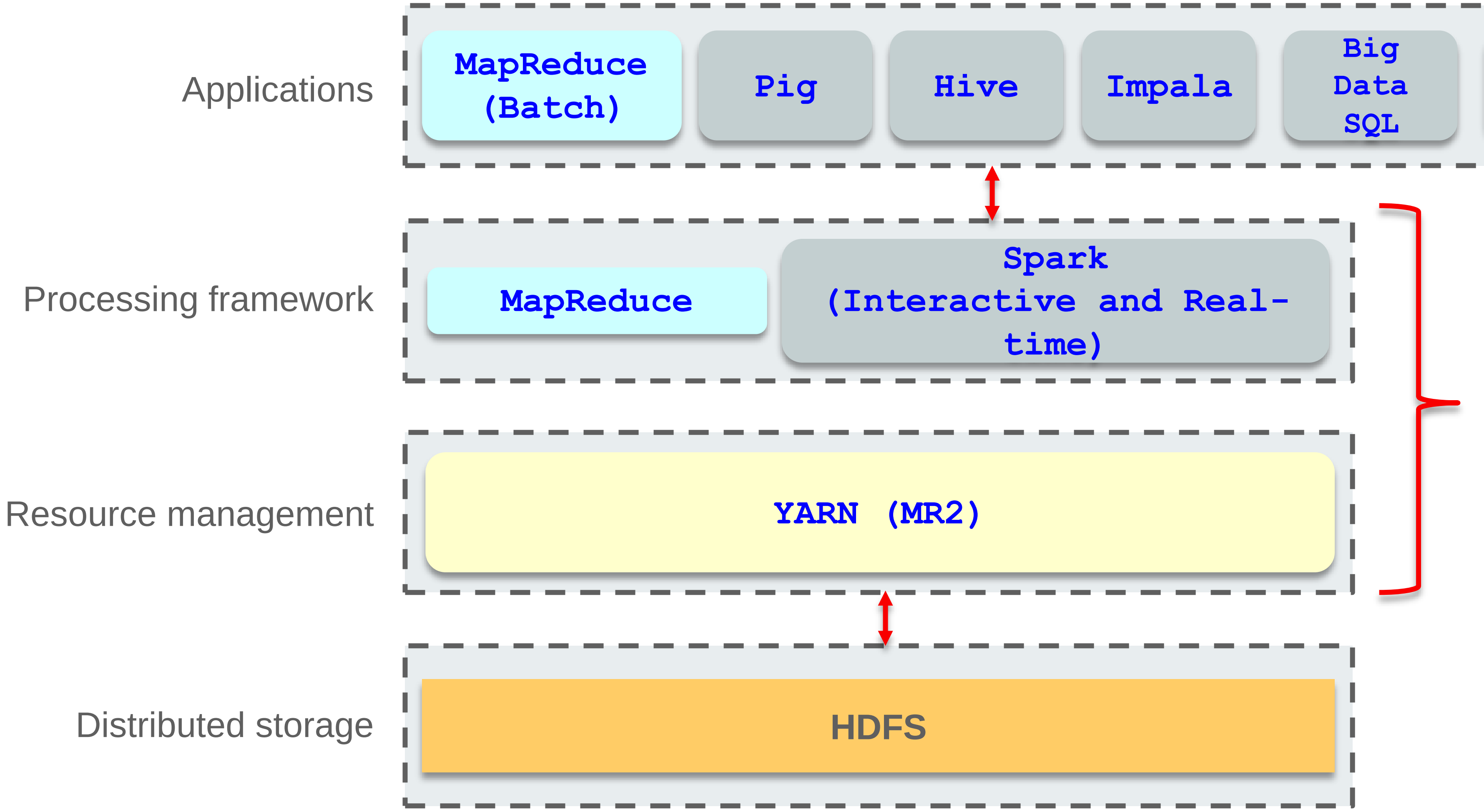
- Hadoop Cluster Resource Management *and*
- ***Batch*** Data Processing
- A single processing engine

Apache Hadoop Core Components: YARN (MR2)

- Is a subproject of Hadoop that separates resource management and processing components
- Is a resource-management framework for Hadoop that is independent of execution engines
- Provides a more efficient and flexible workload scheduling as well as a resource management facility, both of which ultimately enable Hadoop to run more than just MapReduce jobs such as Impala, Spark, and so on

YARN

Running Applications Starting with Hadoop 2.x With YARN (MR 2)



Applications

MapReduce (Batch)

Pig

Hive

Impala

Big Data SQL

Search

Processing framework

MapReduce

Spark (Interactive and Real-time)

Resource management

YARN (MR2)

Distributed storage

HDFS

- Hadoop Cluster Resource Management and
- Batch, ***<u>Interactive</u>***, and ***<u>Real-Time</u>*** Data Processing
- Multiple data processing engines

Apache Hadoop Ecosystem

YARN
(MapReduce 2)

Hadoop Core Components:

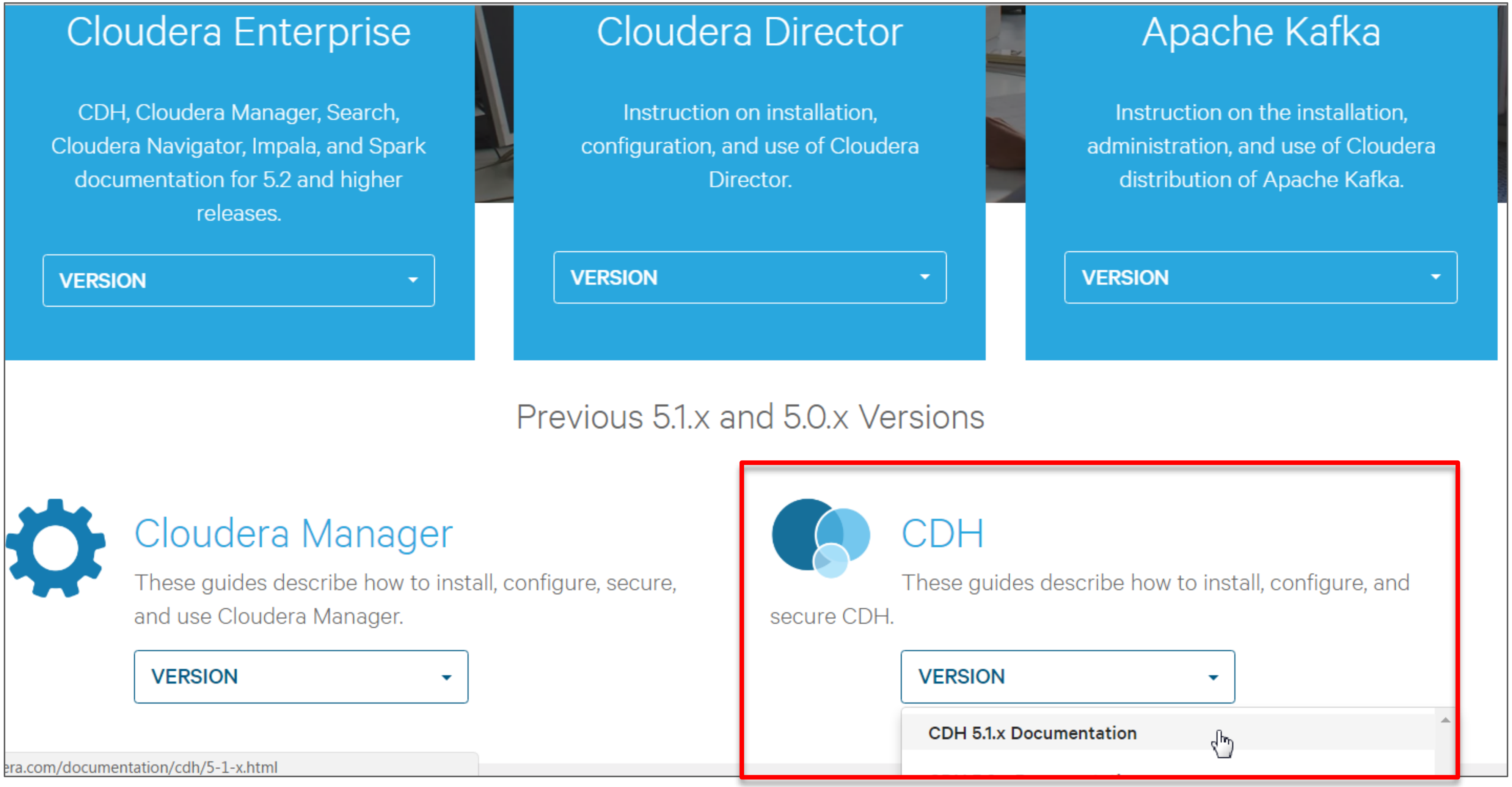
- HDFS (Storage)
- YARN (Processing and Resource Management, MR2)
- MapReduce (Distributed processing, MR1)

OOZiE
Spark
HIVE
ZOO
HUE
Apache Solr
FLUME
WHIRR
sqoop
APACHE HBASE
mahout

Hadoop Ecosystem:
A ***partial*** list of
related projects (extend core
Hadoop or make it easier to
use)

Additional Resources: Cloudera Distribution

http://www.cloudera.com/documentation.html



Cloudera Enterprise

CDH, Cloudera Manager, Search, Cloudera Navigator, Impala, and Spark documentation for 5.2 and higher releases.

VERSION

Cloudera Director

Instruction on installation, configuration, and use of Cloudera Director.

VERSION

Apache Kafka

Instruction on the installation, administration, and use of Cloudera distribution of Apache Kafka.

VERSION

Previous 5.1.x and 5.0.x Versions

Cloudera Manager

These guides describe how to install, configure, secure, and use Cloudera Manager.

VERSION

CDH

These guides describe how to install, configure, and secure CDH.

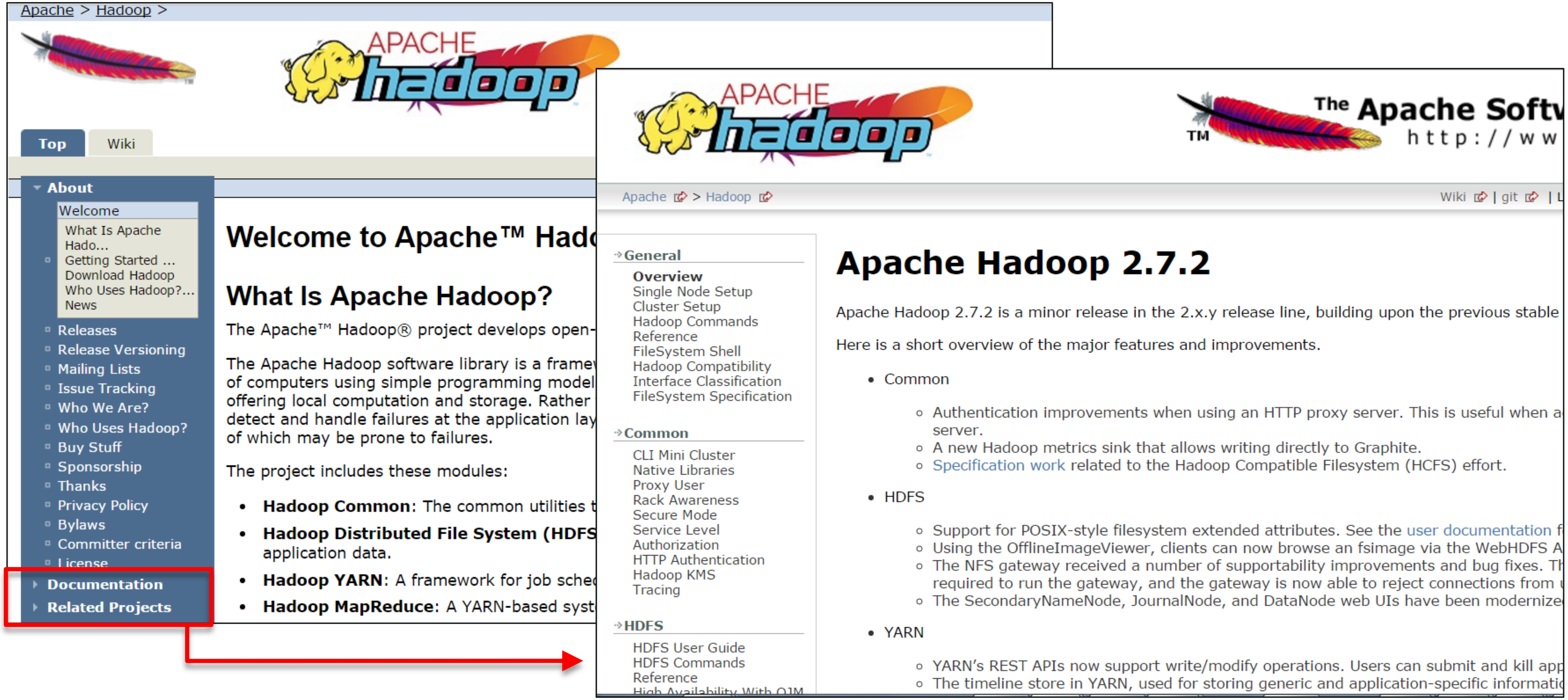
VERSION

CDH 5.1.x Documentation

era.com/documentation/cdh/5-1-x.html

Additional Resources: Apache Hadoop

http://hadoop.apache.org/



Apache > Hadoop >

Top | Wiki

- About
  - Welcome
    - What Is Apache Hado...
    - Getting Started ...
    - Download Hadoop
    - Who Uses Hadoop?...
    - News
  - Releases
  - Release Versioning
  - Mailing Lists
  - Issue Tracking
  - Who We Are?
  - Who Uses Hadoop?
  - Buy Stuff
  - Sponsorship
  - Thanks
  - Privacy Policy
  - Bylaws
  - Committer criteria
  - License
- Documentation
- Related Projects

Welcome to Apache™ Hado

What Is Apache Hadoop?

The Apache™ Hadoop® project develops open-

The Apache Hadoop software library is a frame
of computers using simple programming model
offering local computation and storage. Rather
detect and handle failures at the application lay
of which may be prone to failures.

The project includes these modules:

- **Hadoop Common**: The common utilities t
- **Hadoop Distributed File System (HDFS**
application data.
- **Hadoop YARN**: A framework for job sche
- **Hadoop MapReduce**: A YARN-based syst

The Apache Softw
http://ww

Apache > Hadoop | Wiki | git | L

General
- Overview
- Single Node Setup
- Cluster Setup
- Hadoop Commands Reference
- FileSystem Shell
- Hadoop Compatibility
- Interface Classification
- FileSystem Specification

Common
- CLI Mini Cluster
- Native Libraries
- Proxy User
- Rack Awareness
- Secure Mode
- Service Level Authorization
- HTTP Authentication
- Hadoop KMS
- Tracing

HDFS
- HDFS User Guide
- HDFS Commands Reference
- High Availability With QJM

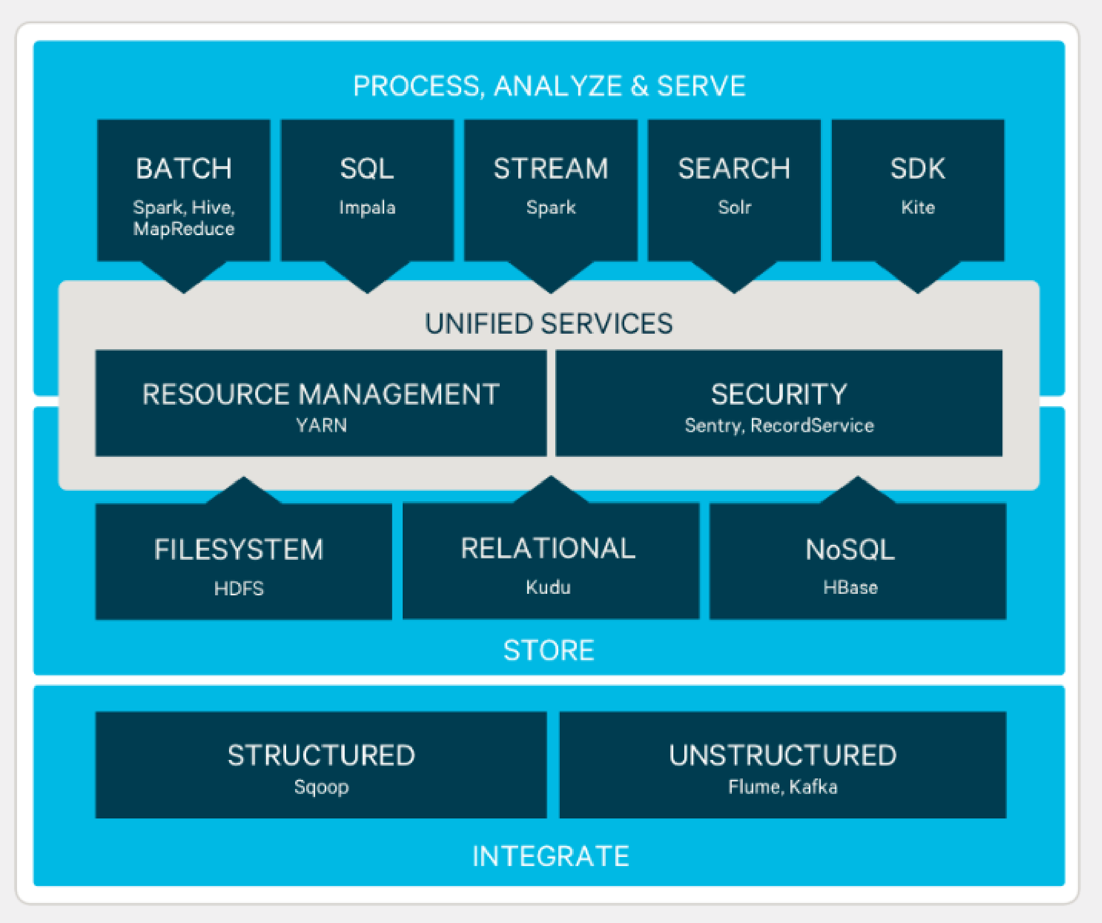
Apache Hadoop 2.7.2

Apache Hadoop 2.7.2 is a minor release in the 2.x.y release line, building upon the previous stable

Here is a short overview of the major features and improvements.

- Common
  - Authentication improvements when using an HTTP proxy server. This is useful when a
server.
  - A new Hadoop metrics sink that allows writing directly to Graphite.
  - Specification work related to the Hadoop Compatible Filesystem (HCFS) effort.
- HDFS
  - Support for POSIX-style filesystem extended attributes. See the user documentation f
  - Using the OfflineImageViewer, clients can now browse an fsimage via the WebHDFS A
  - The NFS gateway received a number of supportability improvements and bug fixes. Th
required to run the gateway, and the gateway is now able to reject connections from u
  - The SecondaryNameNode, JournalNode, and DataNode web UIs have been modernize
- YARN
  - YARN's REST APIs now support write/modify operations. Users can submit and kill app
  - The timeline store in YARN, used for storing generic and application-specific informati

CDH Architecture



Source: http://www.cloudera.com/products/apache-hadoop.html

CDH Components

Component	Description
Apache Hadoop	• A framework for executing applications on a large cluster of servers. It is built for massively parallel processing across a large number of nodes (servers). • Consists of the following core components: Hadoop Distributed File System (HDFS) and MapReduce
Hue **(Hadoop User Experience)**	• Is an open-source tool • Easy to use web front end for viewing files, running queries, performing searches, scheduling jobs, and more • Contains several applications to access a Hadoop cluster through a web front end
Apache Oozie	• Enables developers to create, edit, and submit workflows by using the Oozie dashboard • After considering the dependencies between jobs, the Oozie server submits those jobs to the server in the proper sequence.
Apache Spark	• It is an open source parallel data processing framework. • It complements Apache Hadoop. • It makes it easy to develop fast, unified Big Data applications combining batch, streaming, and interactive analytics on all your data.

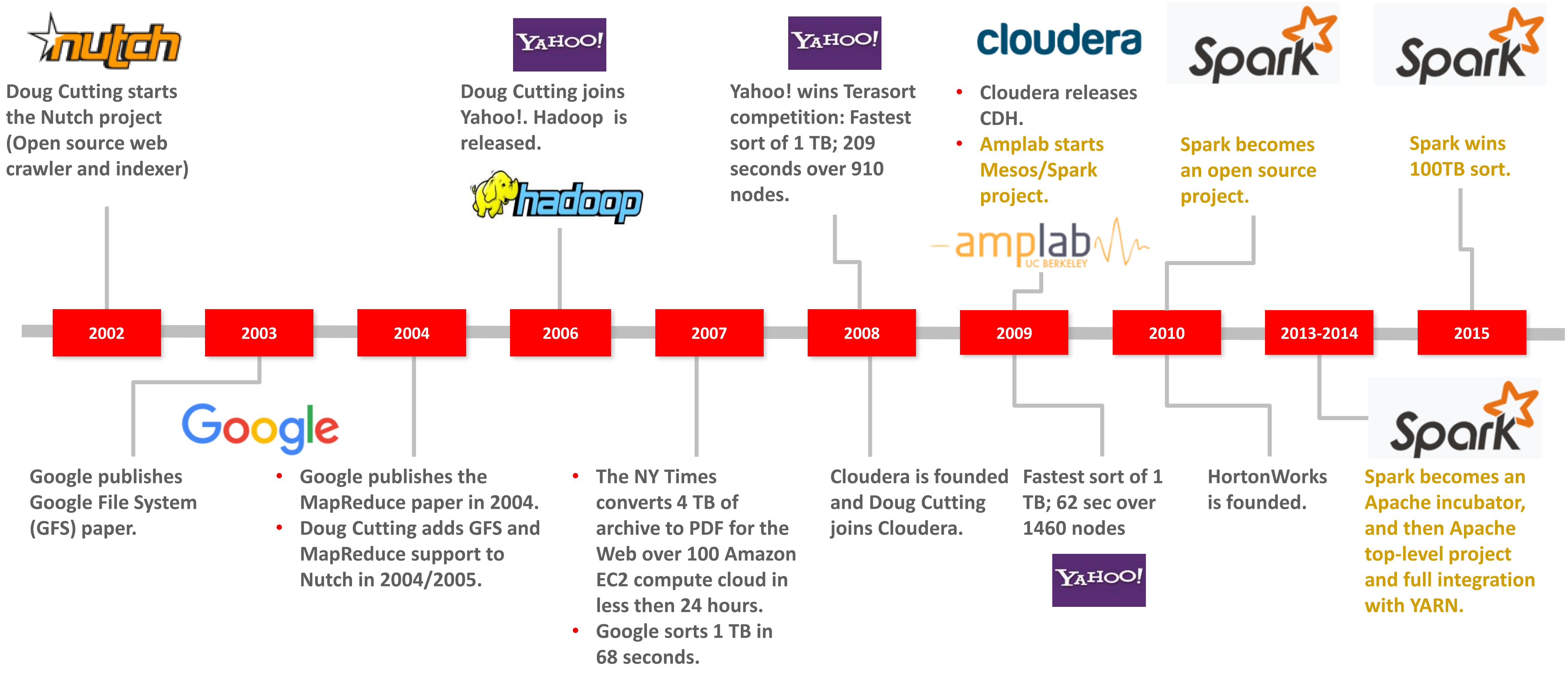
CDH Architecture

Component	Description
Apache Solr (Cloudera Search)	• Cloudera Search is one of Cloudera's near-real-time access products and is powered by Solr. It enables nontechnical users to search and explore data stored in or ingested into Hadoop, Oracle NoSQL Database, and HBase. • Users do not need SQL or programming skills to use Cloudera Search because it provides a simple, full-text interface for searching.
Apache Hive	• Hive Metastore provides a metadata layer that describes the data stored in HDFS. • It provides a SQL layer to data on HDFS. It can run SQL queries on HDFS data. • It uses Map/Reduce for execution and HDFS for storage.
Apache Pig	• It is an analysis platform that provides a data flow language called Pig Latin. • It is an alternative abstraction on top of MapReduce.
Cloudera Impala	• The Impala server is a distributed, massively parallel processing (MPP) database engine. • It consists of different daemon processes that run on specific hosts within your CDH cluster. • The core Impala component is a daemon process that runs on each node of the cluster.

CDH Components

Component	Description
Apache Flume	• A distributed, reliable, available service for efficiently moving large amounts of data as it is generated • Ideal for collecting logs from diverse systems and inserting them in HDFS
Apache Sqoop	• Imports tables from an RDBMS into HDFS • Imports data from RDBMS into HDFS as delimited text files or sequence files • Generates a class file that can encapsulate a row of the imported data
Apache Hbase	• Is a NoSQL data store • Provides scalable inserts, efficient handling of sparse data, and a constrained data access model
Apache ZooKeeper	• ZooKeeper is a centralized service for maintaining configuration information, naming, distributed synchronization, and group services. • HBase cannot be active without ZooKeeper.
Apache Mahout	• Scalable machine-learning and data-mining algorithms
Apache Whirr	• Apache Whirr is a set of libraries for running cloud services. It provides: – A cloud-neutral way to run services – A common service APISmart defaults for services

Hadoop Major Timelines at a Glance



2002

Doug Cutting starts the Nutch project (Open source web crawler and indexer)

2003

Google publishes Google File System (GFS) paper.

2004

- Google publishes the MapReduce paper in 2004.
- Doug Cutting adds GFS and MapReduce support to Nutch in 2004/2005.

2006

Doug Cutting joins Yahoo!. Hadoop is released.

2007

- The NY Times converts 4 TB of archive to PDF for the Web over 100 Amazon EC2 compute cloud in less then 24 hours.
- Google sorts 1 TB in 68 seconds.

2008

Yahoo! wins Terasort competition: Fastest sort of 1 TB; 209 seconds over 910 nodes.

Cloudera is founded and Doug Cutting joins Cloudera.

2009

- Cloudera releases CDH.
- Amplab starts Mesos/Spark project.

Fastest sort of 1 TB; 62 sec over 1460 nodes

2010

Spark becomes an open source project.

HortonWorks is founded.

2013-2014

Spark becomes an Apache incubator, and then Apache top-level project and full integration with YARN.

2015

Spark wins 100TB sort.

Where to Go for More Information

Component	Website
Cloudera Manager	http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html
Apache Hadoop	http://hadoop.apache.org/
Apache Hadoop – Cloudera	**http://www.cloudera.com/products/apache-hadoop.html**
fuse-dfs	http://fuse.sourceforge.net/
Cloudera Hue	http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/4.2.0/Hue-2-User-Guide/hue2.html
Apache Oozie	http://oozie.apache.org/
Apache Hive	https://hive.apache.org/
Apache Pig	http://pig.apache.org
Apache Flume	http://flume.apache.org/
Apache Sqoop	http://sqoop.apache.org/
Apache HBase	http://hbase.apache.org/
Apache ZooKeeper	http://zookeeper.apache.org
Apache Mahout	http://mahout.apache.org
Apache Whirr	https://whirr.apache.org/

Summary

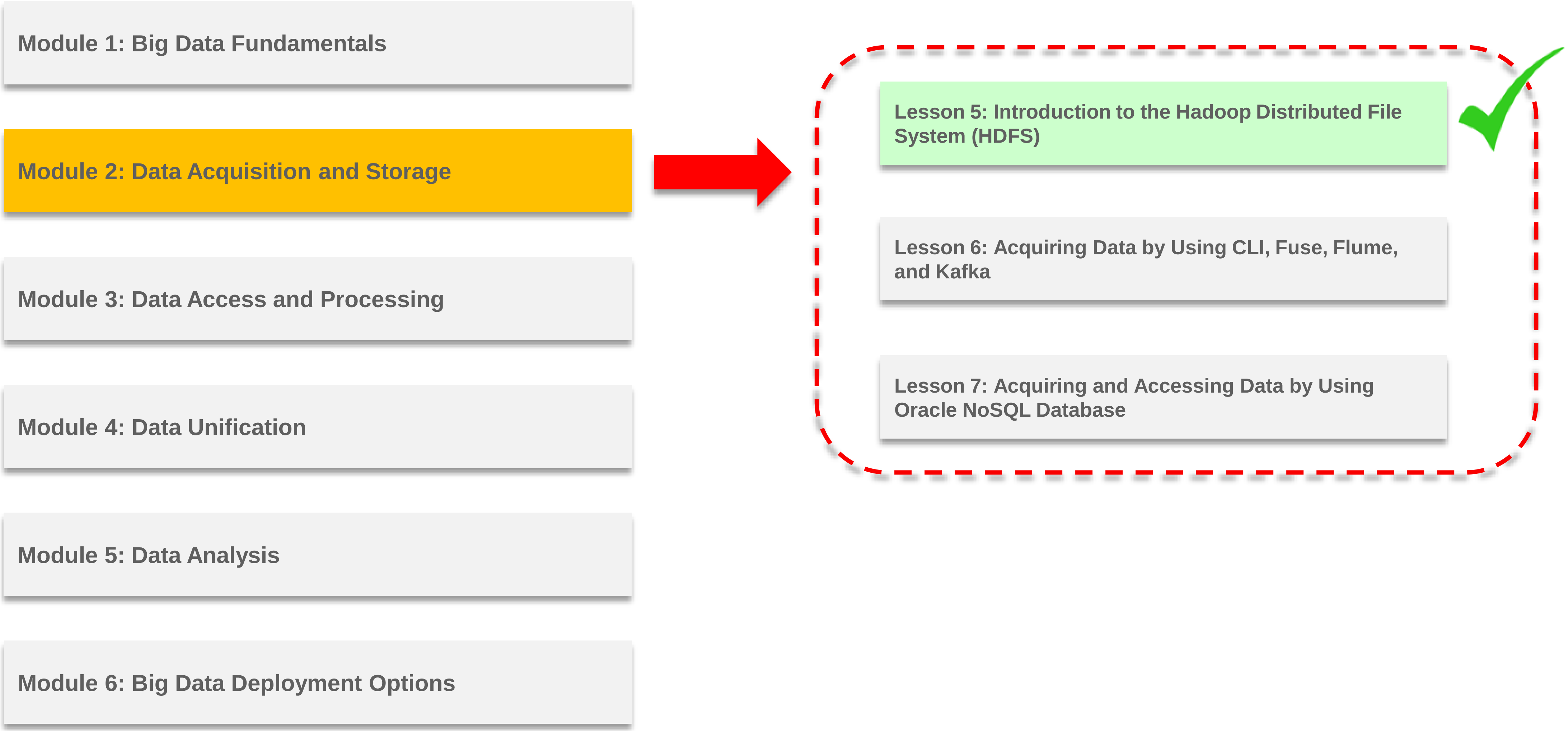
In this lesson, you should have learned how to:

- Define Hadoop and the *Hadoop Ecosystem*
- Describe the Hadoop core components
- Choose a Hadoop Distribution
- List some of the other related projects in the Hadoop Ecosystem

4

Introduction to the Hadoop Distributed File System (HDFS)

Course Road Map



Module 1: Big Data Fundamentals

Module 2: Data Acquisition and Storage

Module 3: Data Access and Processing

Module 4: Data Unification

Module 5: Data Analysis

Module 6: Big Data Deployment Options

Lesson 5: Introduction to the Hadoop Distributed File System (HDFS) ✓

Lesson 6: Acquiring Data by Using CLI, Fuse, Flume, and Kafka

Lesson 7: Acquiring and Accessing Data by Using Oracle NoSQL Database

Objectives

After completing this lesson, you should be able to:

- Describe the architectural components of HDFS
- Interact with data stored in HDFS by using various methods

Agenda

- Understand the architectural components of HDFS
- Interact with data stored in HDFS
  - Hue
  - Hadoop client file system shell command-line interface (CLI)
  - WebHDFS
  - HttpFS

HDFS Design Principles and Characteristics

Master-slave architecture

Fault-tolerant (HA)

Redundant

Supports MapReduce & Spark

Scalable

Commodity Hardware

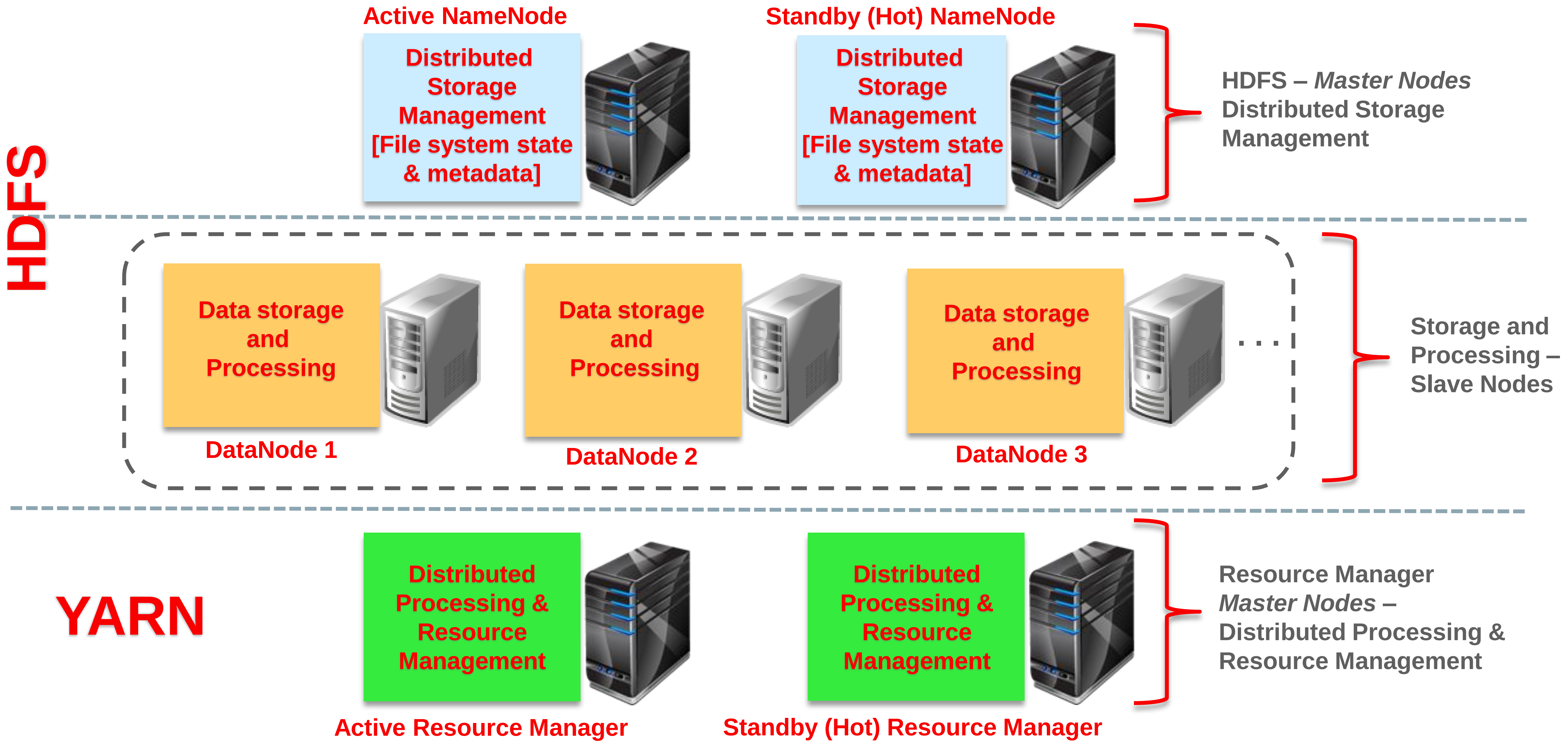
HDFS Key Definitions

Term	Description
Hadoop	A batch (MapReduce), interactive, or real-time (Spark and MR2) processing infrastructure that stores and distributes files and distributes work across a group of servers (nodes)
Hadoop Cluster	A collection of racks containing master and slave nodes
Blocks	HDFS breaks down a data file into blocks or “chunks” and stores the data blocks on different slave DataNodes in the Hadoop cluster.
Replication Factor	HDFS makes three copies of data blocks and stores them on different data nodes/racks in the Hadoop cluster.
NameNode (NN)	A service (daemon) that maintains a directory of all files in HDFS and tracks where data is stored in the HDFS cluster. It basically manages the file system’s metadata ***(does not contain actual data).***
DataNode (DN)	This is where the data is stored (HDFS) and processed (MapReduce). This is a slave node. HDFS stores the blocks or “chunks of data for a set of files on the data nodes.

HDFS Deployments: High Availability (HA) and Non-HA

- Non-HA Deployment (Prior to Hadoop 2.0):
  - Uses the NameNode/Secondary NameNode architecture
  - The Secondary NameNode is not a failover for the NameNode.
  - The NameNode was the **Single Point of Failure (SPOF)** of the cluster prior to Hadoop 2.0 and CDH 4.0.
- HA Deployment (Hadoop 2.0 and later):
  - HDFS HA addresses the SPOF by running two redundant NameNodes in the same cluster to provide a **Fast Failover** if needed:
    - **Active NameNode**: Responsible for all client operations in the cluster
    - **Standby NameNode**: Acts as a **“hot” backup** to the Active NameNode, maintaining enough system information to provide a fast failover if necessary
  - **HA allows a fast failover** to a new NameNode in case a machine crashes.
- In this course, you will focus on the HDFS HA deployment option only.

Sample Hadoop High Availability (HA) Cluster



HDFS

Active NameNode

Distributed Storage Management [File system state & metadata]

Standby (Hot) NameNode

Distributed Storage Management [File system state & metadata]

HDFS – *Master Nodes* Distributed Storage Management

Data storage and Processing

DataNode 1

Data storage and Processing

DataNode 2

Data storage and Processing

DataNode 3

. . .

Storage and Processing – Slave Nodes

YARN

Distributed Processing & Resource Management

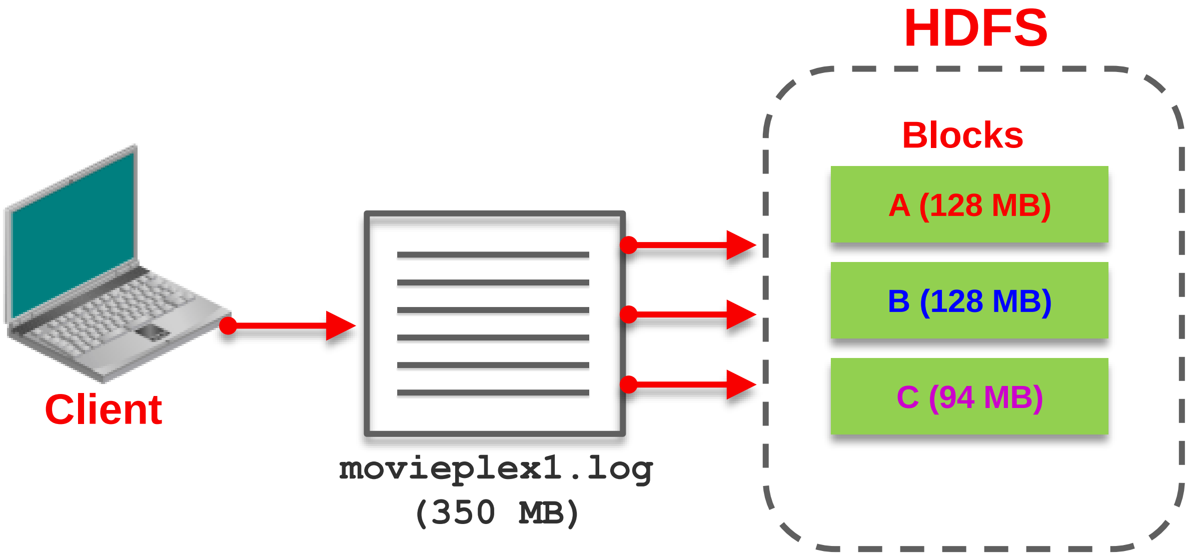
Active Resource Manager

Distributed Processing & Resource Management

Standby (Hot) Resource Manager

Resource Manager *Master Nodes* – Distributed Processing & Resource Management

HDFS Files and Blocks

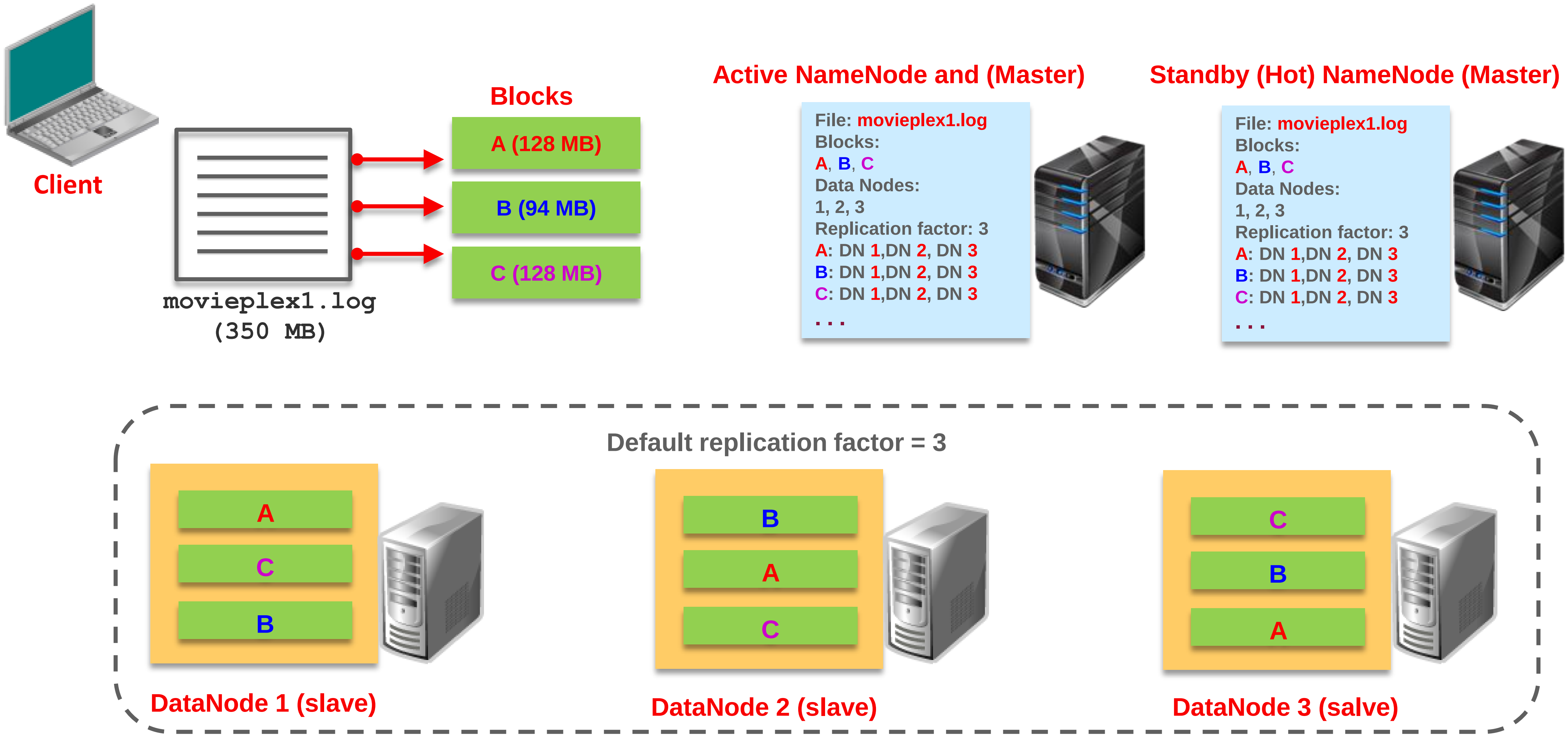


- Files in HDFS consist of blocks.
- HDFS blocks default to 128 MB in size (configurable).
- *Files are "chunked" into blocks as they are ingested (using Flume or Kafka) into HDFS.*

Assuming a default block size of 128 MB, HDFS ingests the `movieplex1.log` file into (3) blocks:

- **A (128 MB)**
- **B (128 MB)**
- **C (94 MB)**

Blocks are Replicated in the Cluster Upon Ingestion into HDFS



File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

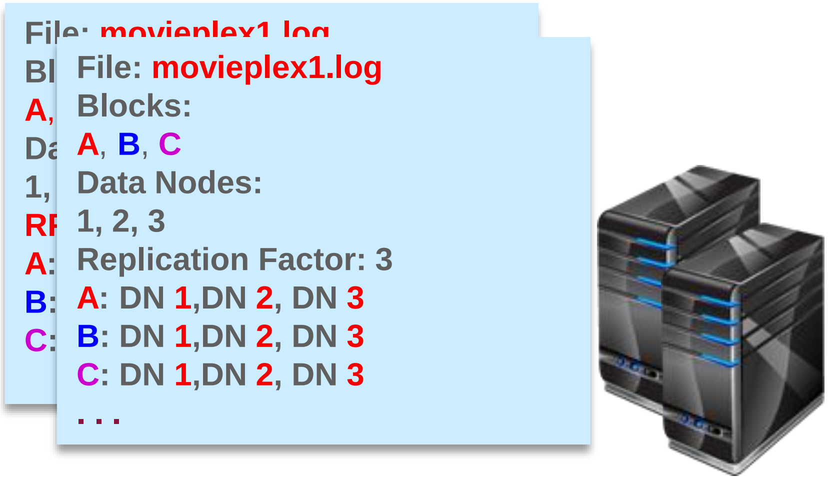
File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

Active and Standby NameNodes Daemons

NameNode stores file system metadata such as:

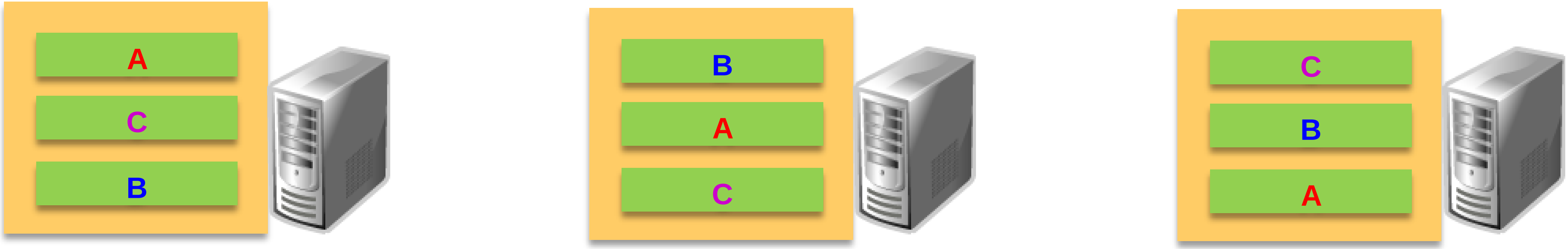
- File information (name, updates, replication factor, etc.)
- File blocks information and locations
- Access rights to the file
- Number of files in the cluster
- Number of DataNodes in the cluster

Active NameNode and Standby (Hot) NameNode (Masters)



File: movieplex1.log
Bl
A,
Da
1,
RF
A:
B:
C:

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication Factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .



DataNode 1 (slave)

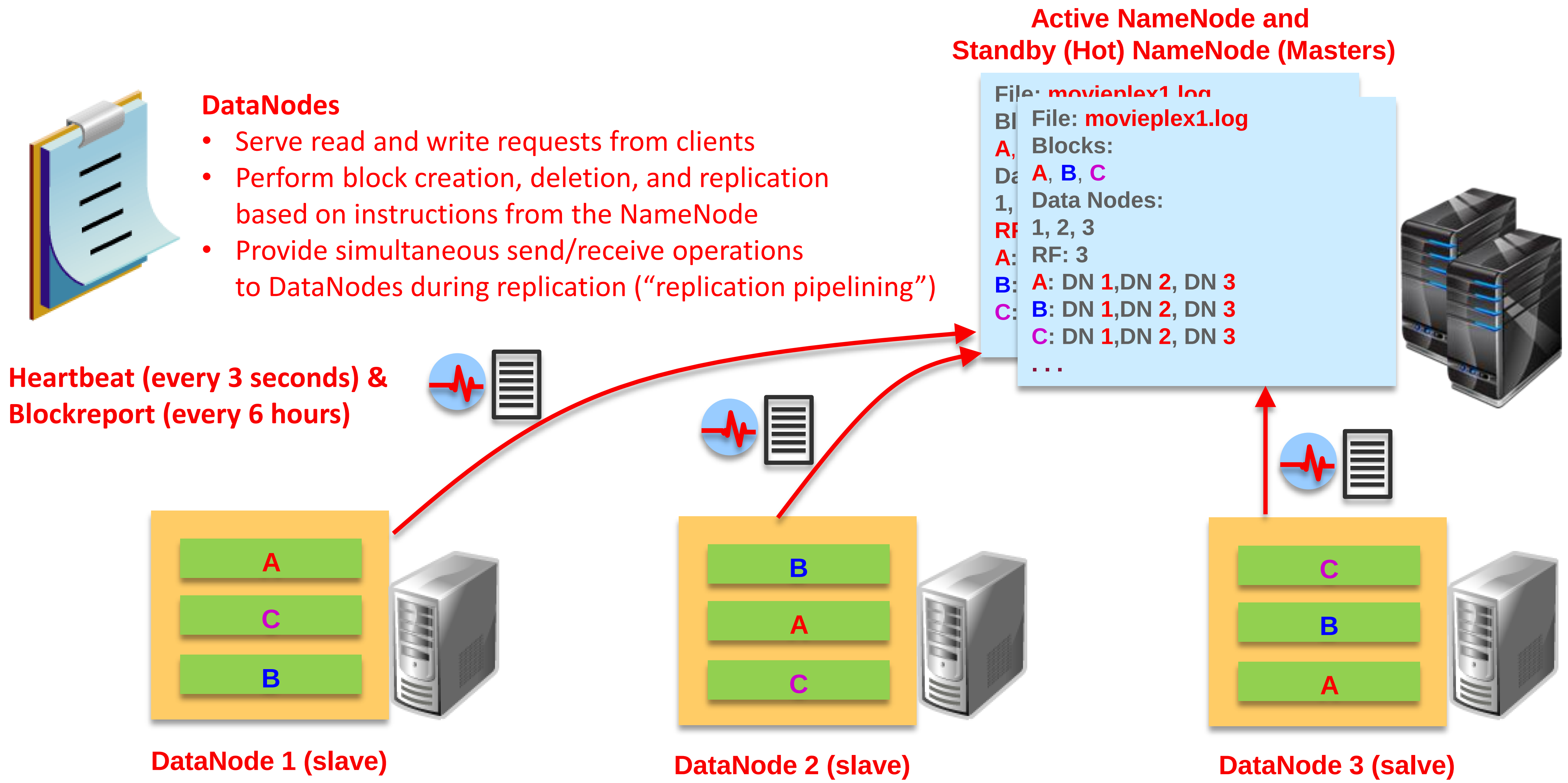
DataNode 2 (slave)

DataNode 3 (salve)

DataNodes Daemons

DataNodes

- Serve read and write requests from clients
- Perform block creation, deletion, and replication based on instructions from the NameNode
- Provide simultaneous send/receive operations to DataNodes during replication (“replication pipelining”)



Active NameNode and Standby (Hot) NameNode (Masters)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
RF: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

Heartbeat (every 3 seconds) & Blockreport (every 6 hours)

DataNode 1 (slave)

DataNode 2 (slave)

DataNode 3 (salve)

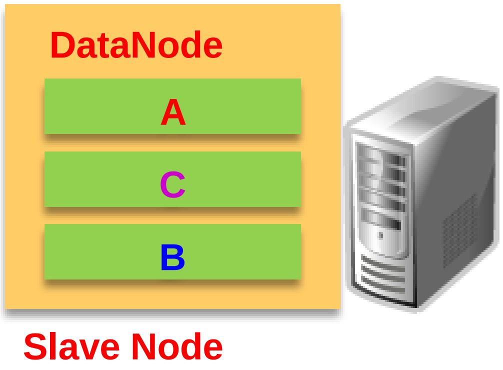
Functions of the NameNode

- Acts as the repository for all HDFS metadata
- Maintains the file system namespace
- Executes the directives for opening, closing, and renaming files and directories
- Stores the HDFS state in an image file (`fsimage`)
- Stores file system modifications in an edit log file (`edits`)
- On startup, merges the `fsimage` and `edits` files, and then empties `edits`
- Places replicas of blocks on multiple racks for fault tolerance
- Records the number of replicas (replication factor) of a file specified by an application

Functions of DataNodes

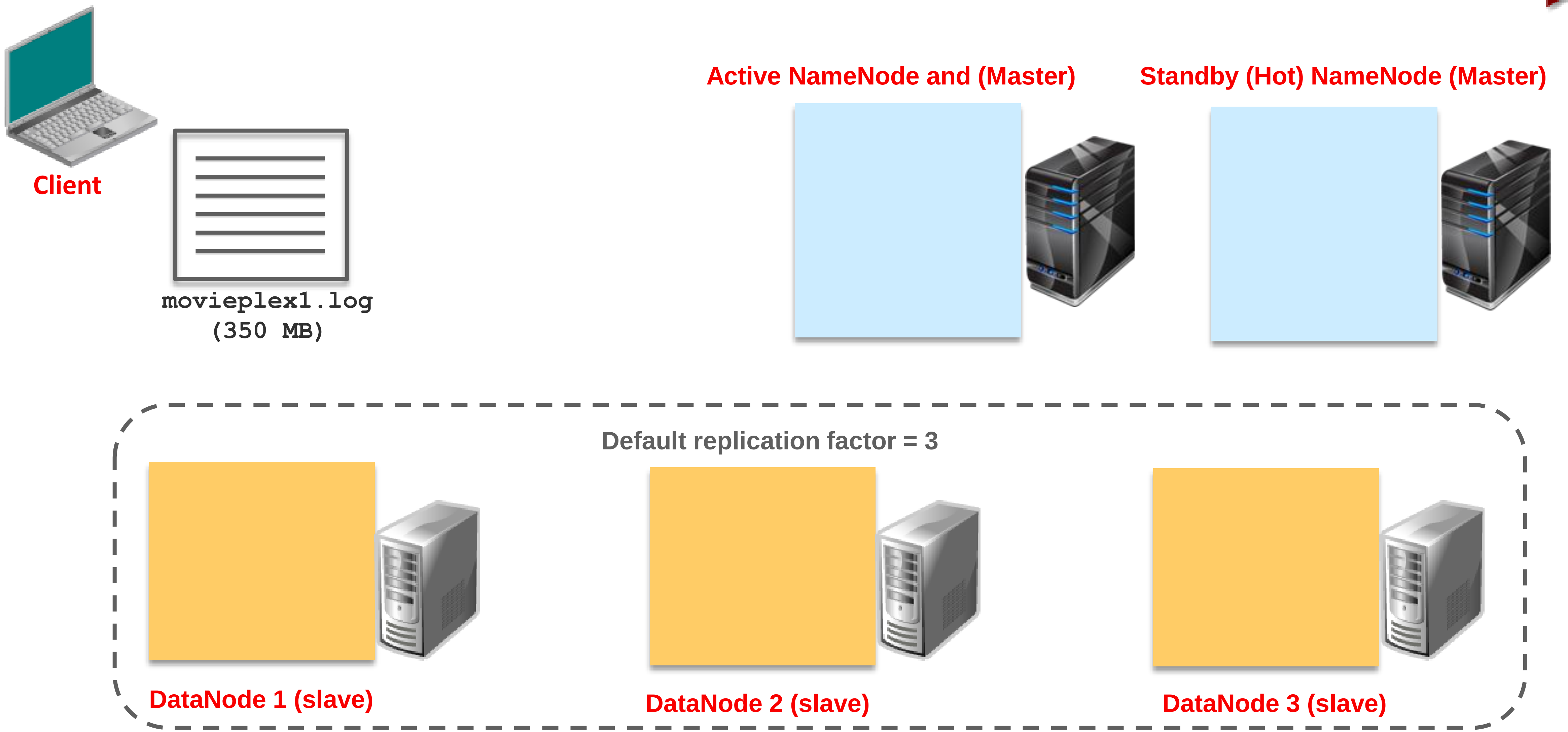
DataNodes perform the following functions:

- Serving read and write requests from the file system clients
- Performing block creation, deletion, and replication based on instructions from the NameNode
- Providing simultaneous send/receive operations to DataNodes during replication (“replication pipelining”)



- Heartbeat (I am alive!) every 3 seconds
- Blockreport (what data I am storing!) every 6 hours to NN

Writing a File to HDFS: Example



Client

movieplex1.log
(350 MB)

Active NameNode and (Master)

Standby (Hot) NameNode (Master)

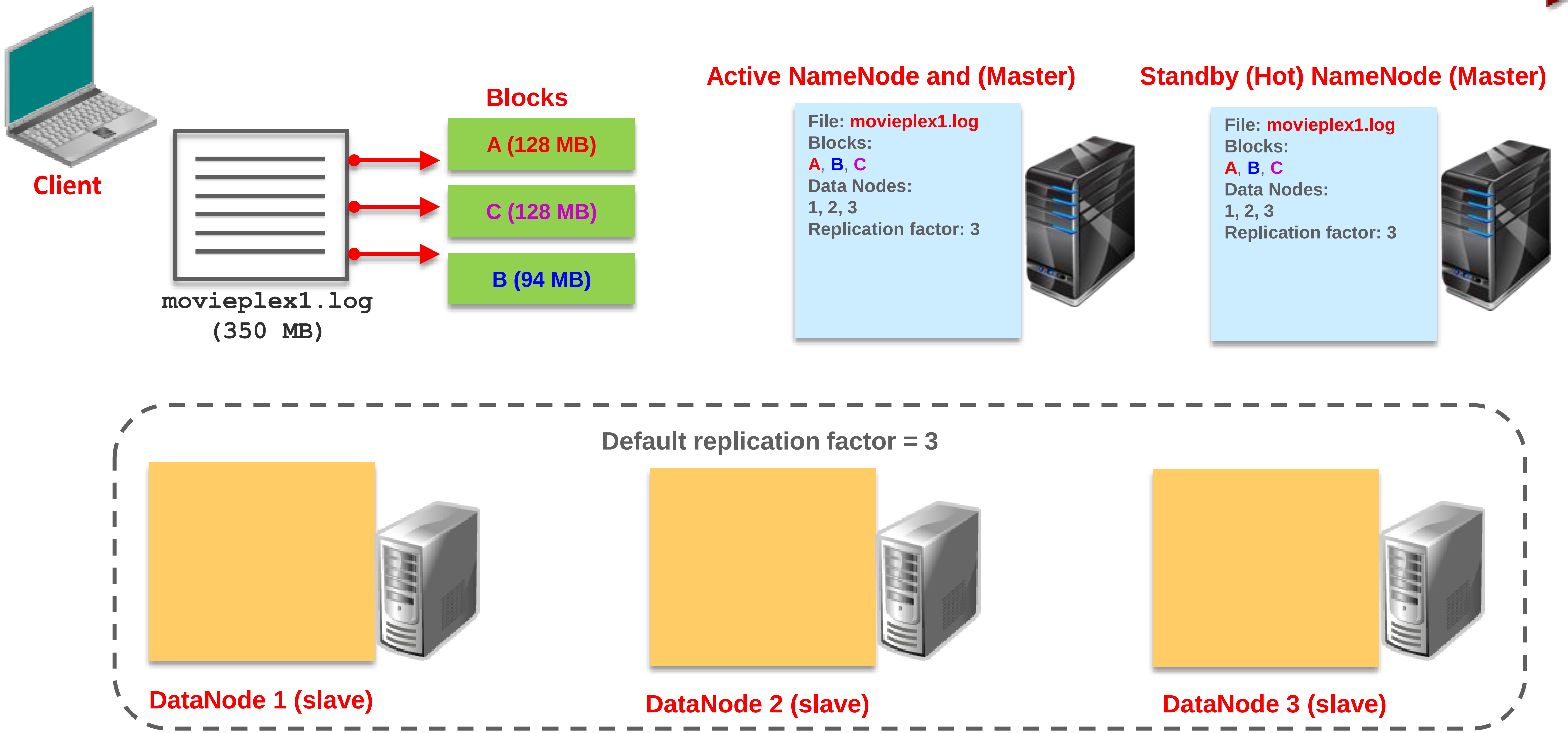
Default replication factor = 3

DataNode 1 (slave)

DataNode 2 (slave)

DataNode 3 (slave)

Writing a File to HDFS: File is “Chunked” into Blocks – Example



Client

movieplex1.log
(350 MB)

Blocks

A (128 MB)

C (128 MB)

B (94 MB)

Active NameNode and (Master)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3

Standby (Hot) NameNode (Master)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3

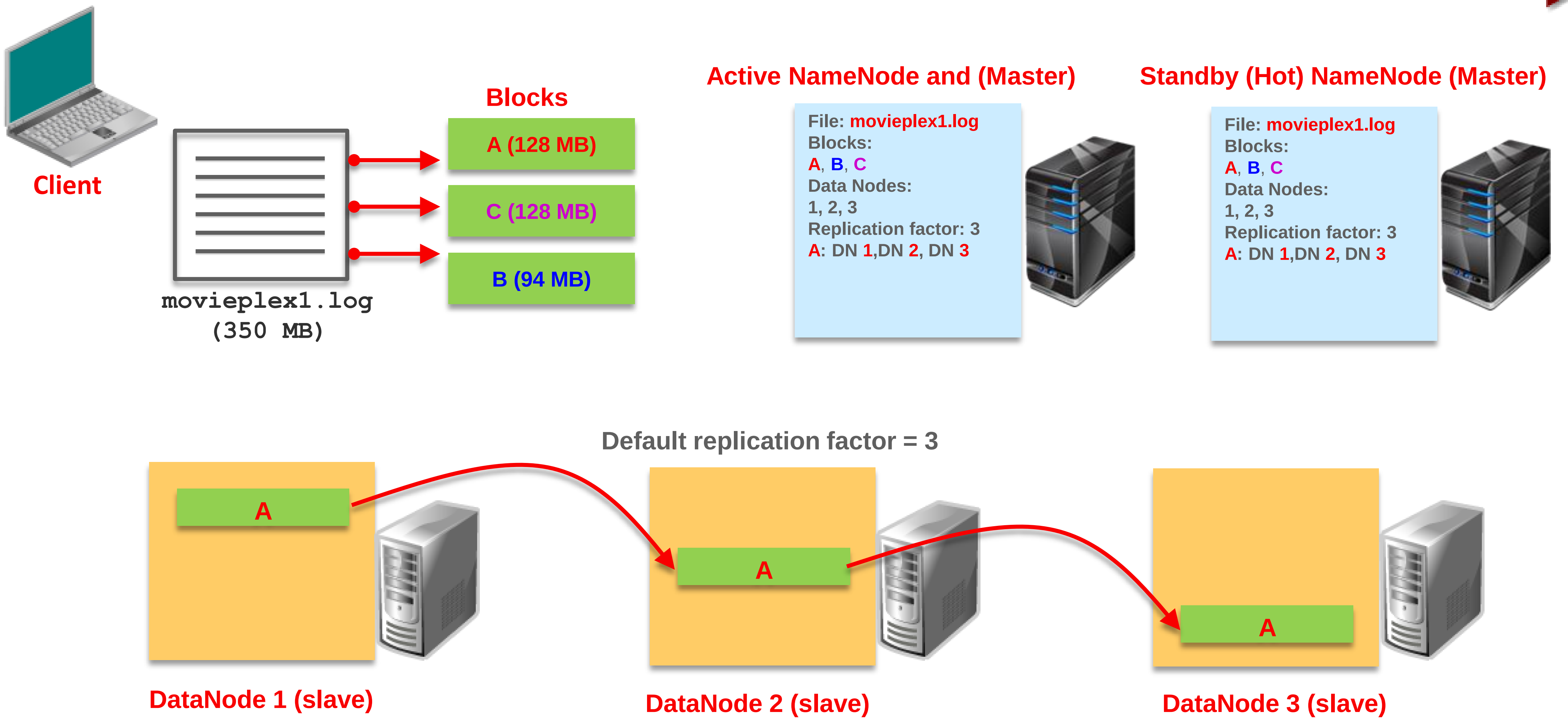
Default replication factor = 3

DataNode 1 (slave)

DataNode 2 (slave)

DataNode 3 (slave)

Writing a File to HDFS: Pipeline Created, Block A – Example



Client

movieplex1.log
(350 MB)

Blocks

A (128 MB)

C (128 MB)

B (94 MB)

Active NameNode and (Master)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3

Standby (Hot) NameNode (Master)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3

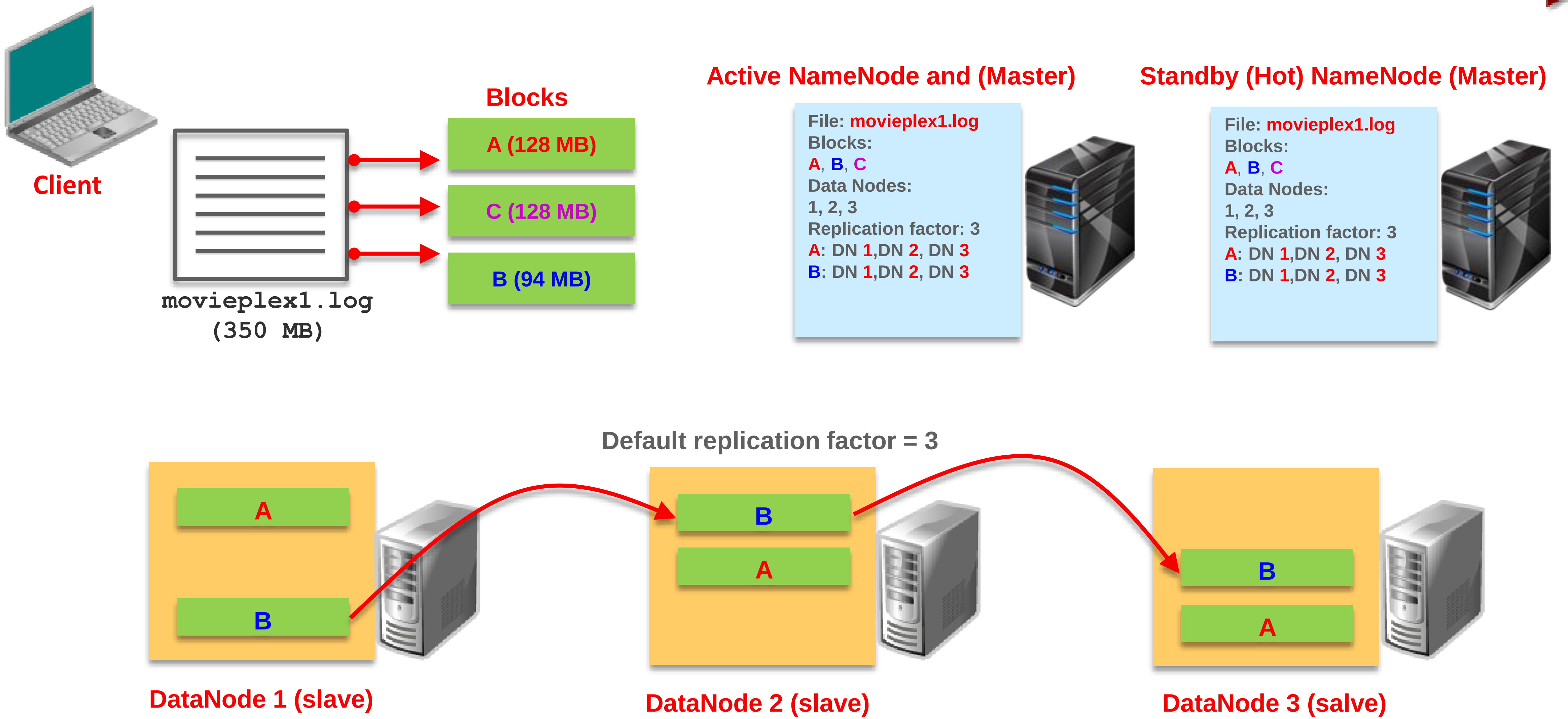
Default replication factor = 3

DataNode 1 (slave)

DataNode 2 (slave)

DataNode 3 (slave)

Writing a File to HDFS: Pipeline Created, Block B – Example



Client

movieplex1.log
(350 MB)

Blocks

A (128 MB)

C (128 MB)

B (94 MB)

Active NameNode and (Master)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3

Standby (Hot) NameNode (Master)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3

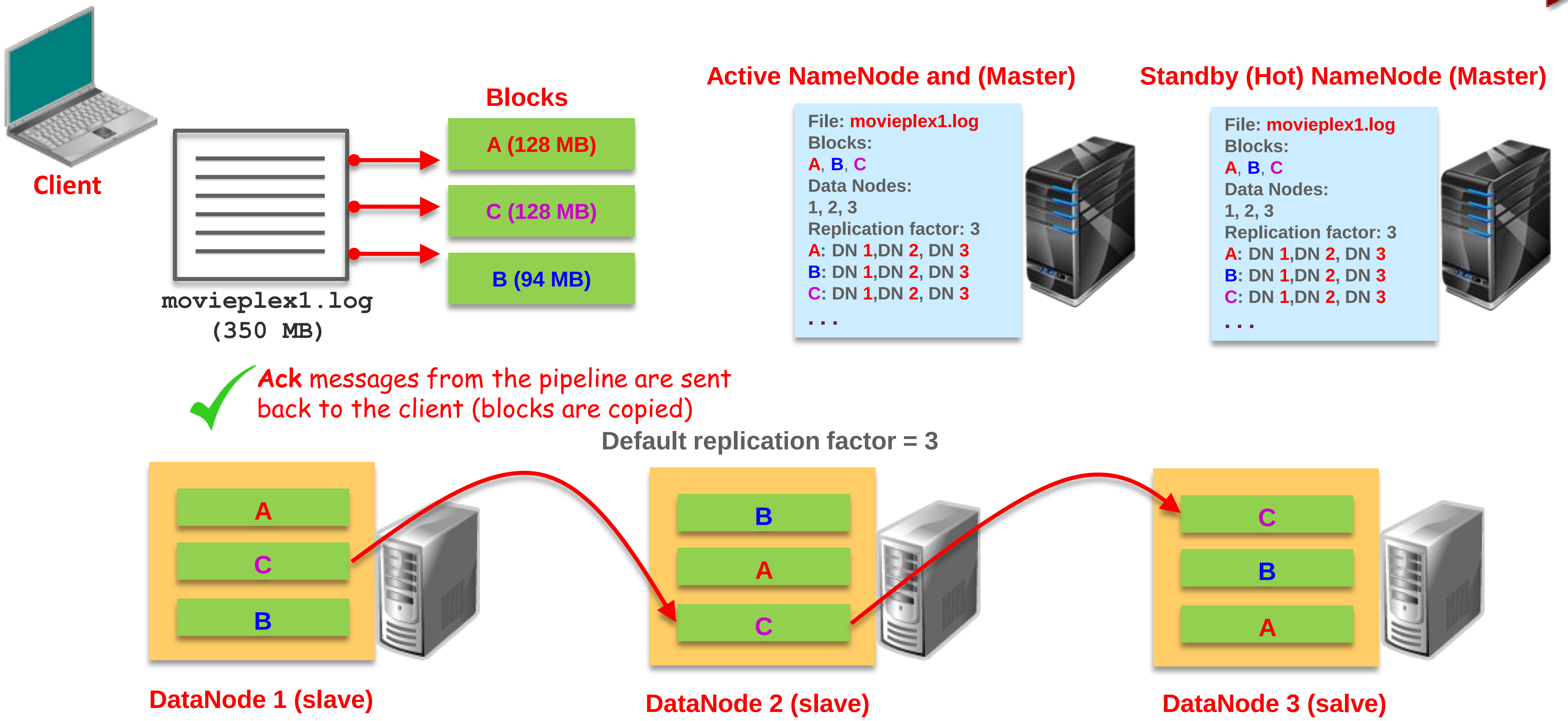
Default replication factor = 3

DataNode 1 (slave)

DataNode 2 (slave)

DataNode 3 (salve)

Writing a File to HDFS: Pipeline Created, Block C – Example



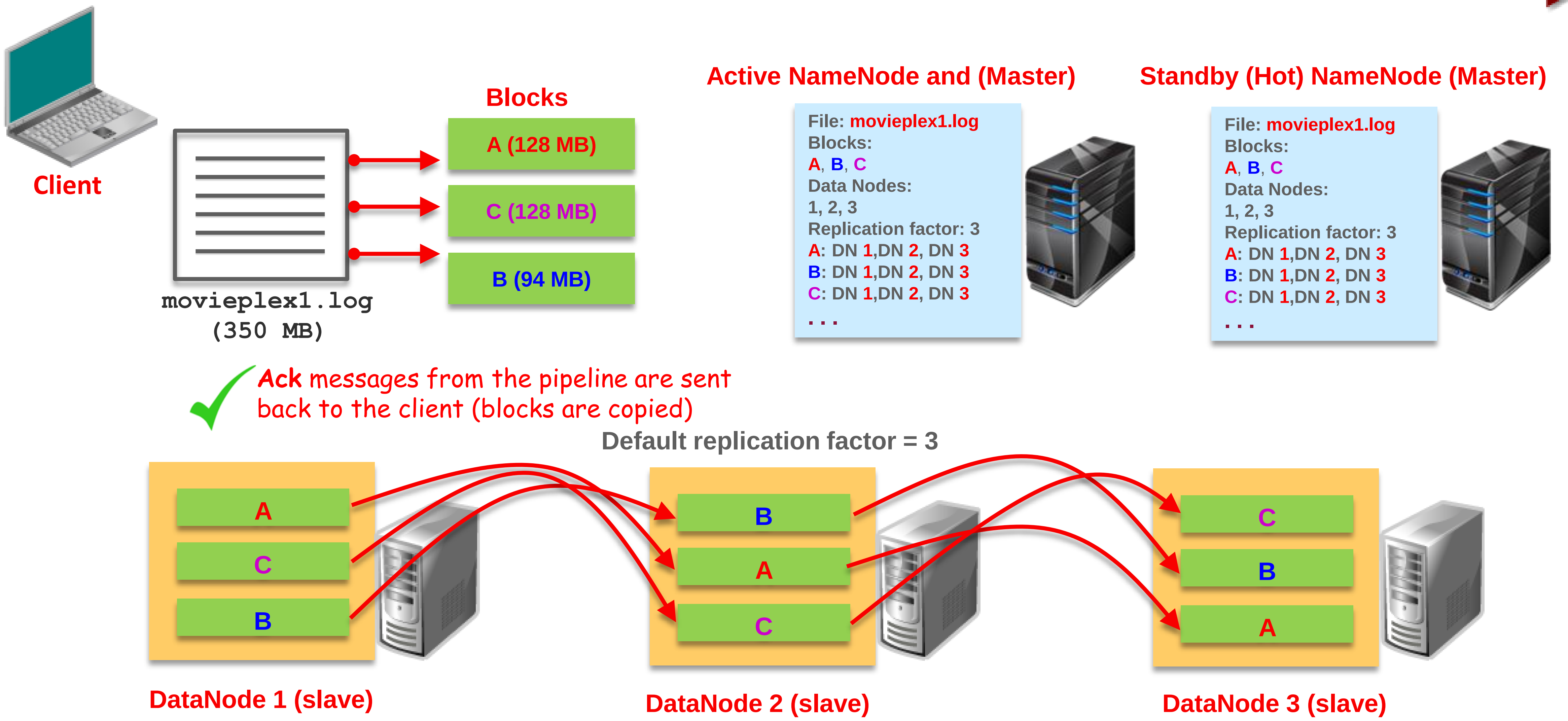
File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

✓ Ack messages from the pipeline are sent back to the client (blocks are copied)

Default replication factor = 3

Writing a File to HDFS: Example



Client

moviepleх1.log
(350 MB)

Blocks

A (128 MB)

C (128 MB)

B (94 MB)

Active NameNode and (Master)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

Standby (Hot) NameNode (Master)

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
Replication factor: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

✓ Ack messages from the pipeline are sent back to the client (blocks are copied)

Default replication factor = 3

DataNode 1 (slave)

DataNode 2 (slave)

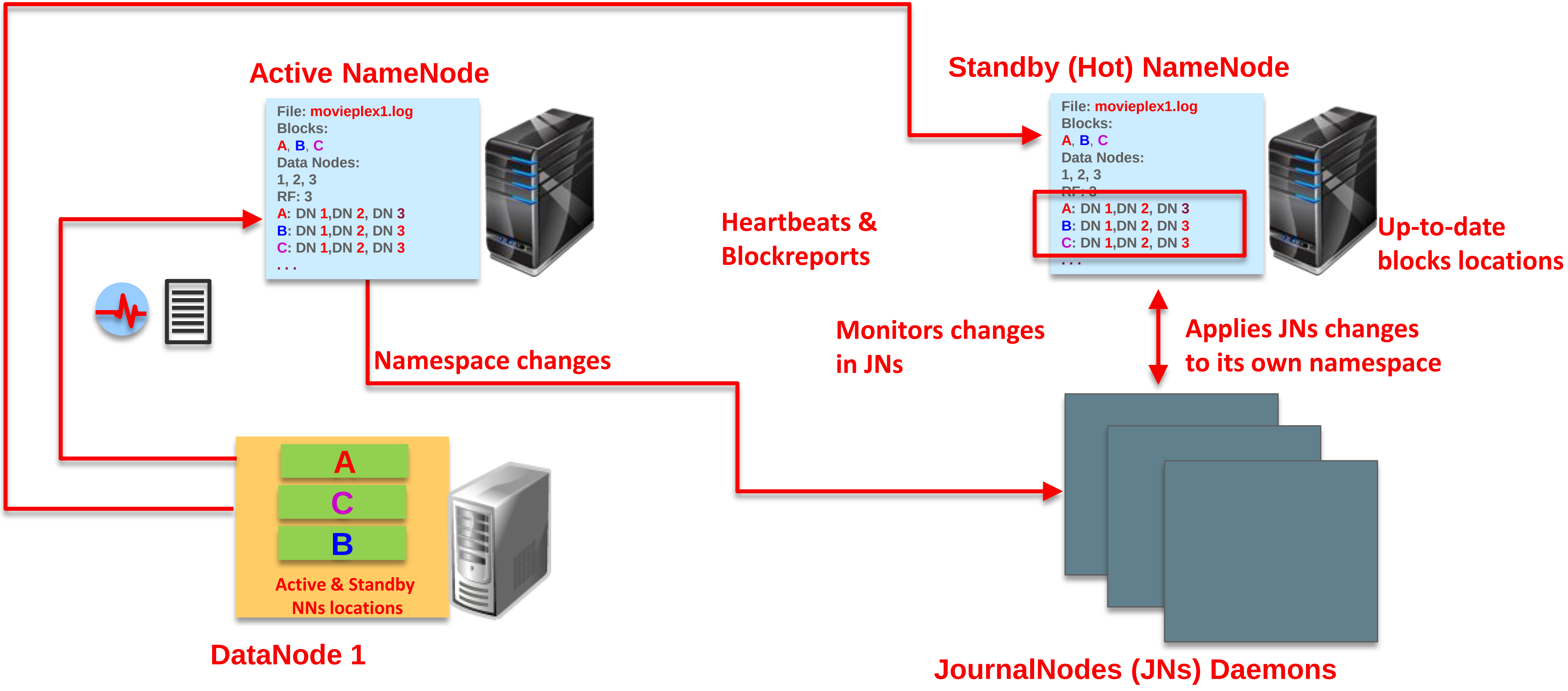
DataNode 3 (slave)

HDFS High Availability (HA) Using the Quorum Journal Manager (QJM)

- Prior to Hadoop 2.0.0, the NameNode was a single point of failure (SPOF) in an HDFS cluster.
- Each cluster had a single NameNode.
- The cluster is unavailable when the NameNode machine crashes or during software and hardware maintenance.
- HDFS HA addresses this problem by:
  - Running two redundant NameNodes in the same cluster:

    An **Active** NameNode and a **Hot Standby** NameNode
- HA provides fast failover to a new NameNode when the NameNode machine crashes or during regular software and hardware maintenance.
- Oracle Big Data Appliance (BDA) uses the HA implementation.

HDFS High Availability (HA) Using the Quorum Journal Manager (QJM) Feature



Active NameNode

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
RF: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

Standby (Hot) NameNode

File: movieplex1.log
Blocks:
A, B, C
Data Nodes:
1, 2, 3
RF: 3
A: DN 1,DN 2, DN 3
B: DN 1,DN 2, DN 3
C: DN 1,DN 2, DN 3
. . .

Up-to-date blocks locations

Heartbeats & Blockreports

Namespace changes

Monitors changes in JNs

Applies JNs changes to its own namespace

A
C
B

Active & Standby NNs locations

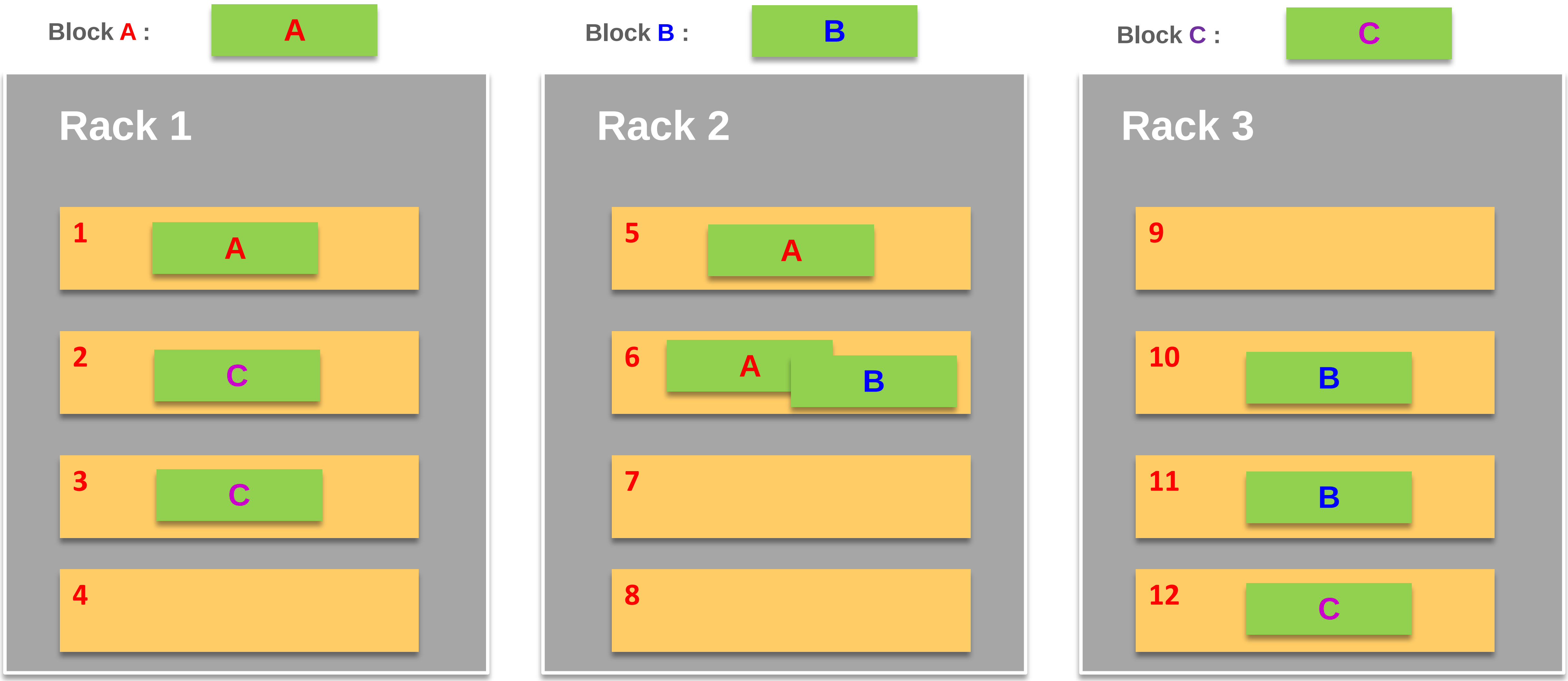
DataNode 1

JournalNodes (JNs) Daemons

Enabling HDFS HA

- Using Cloudera Manager:
  - Enable HA and Automatic Failover
- Using the command-line interface to configure automatic failover. Automatic failover adds the following components to an HDFS deployment:
  - A ZooKeeper quorum, which provides:
    - Failure detection
    - Active NameNode election
  - `ZKFailoverController` process (`ZKFC`), which provides:
    - Health monitoring
    - ZooKeeper session management
    - ZooKeeper-based election

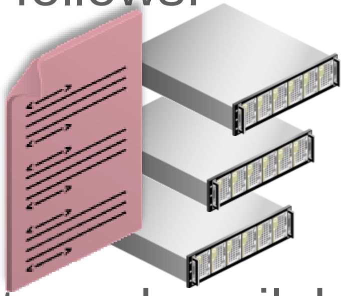
Data Replication Rack-Awareness in HDFS



Data Replication Process

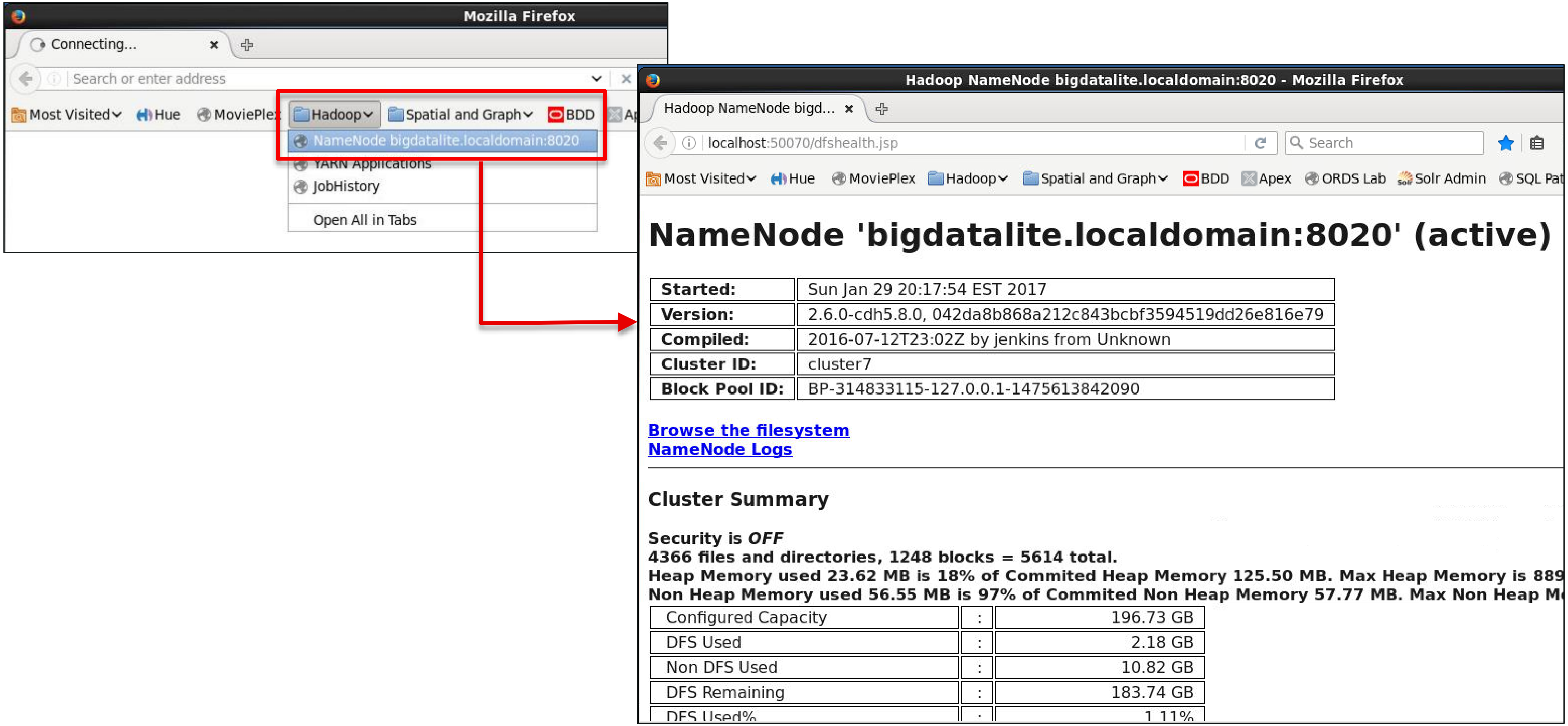
The number of file replicas that will be maintained by HDFS (the “replication factor”) is stored in the NameNode.

- If the factor is (3), the ***<u>HDFS Placement Policy</u>*** directs replication as follows:
  - One copy on one node in a local rack
  - One copy on a different remote rack
  - One copy on a different node in the same remote rack



- This policy improves the write performance and ensures data reliability and availability.
- *If the reader process requires data, HDFS makes sure that it pulls the nearest replica for the task, thereby reducing the read latency (data locality)*.

Accessing HDFS



NameNode 'bigdatalite.localdomain:8020' (active)

Started:	Sun Jan 29 20:17:54 EST 2017
Version:	2.6.0-cdh5.8.0, 042da8b868a212c843bcbf3594519dd26e816e79
Compiled:	2016-07-12T23:02Z by jenkins from Unknown
Cluster ID:	cluster7
Block Pool ID:	BP-314833115-127.0.0.1-1475613842090

Browse the filesystem
NameNode Logs

Cluster Summary

Security is *OFF*
4366 files and directories, 1248 blocks = 5614 total.
Heap Memory used 23.62 MB is 18% of Commited Heap Memory 125.50 MB. Max Heap Memory is 889
Non Heap Memory used 56.55 MB is 97% of Commited Non Heap Memory 57.77 MB. Max Non Heap M

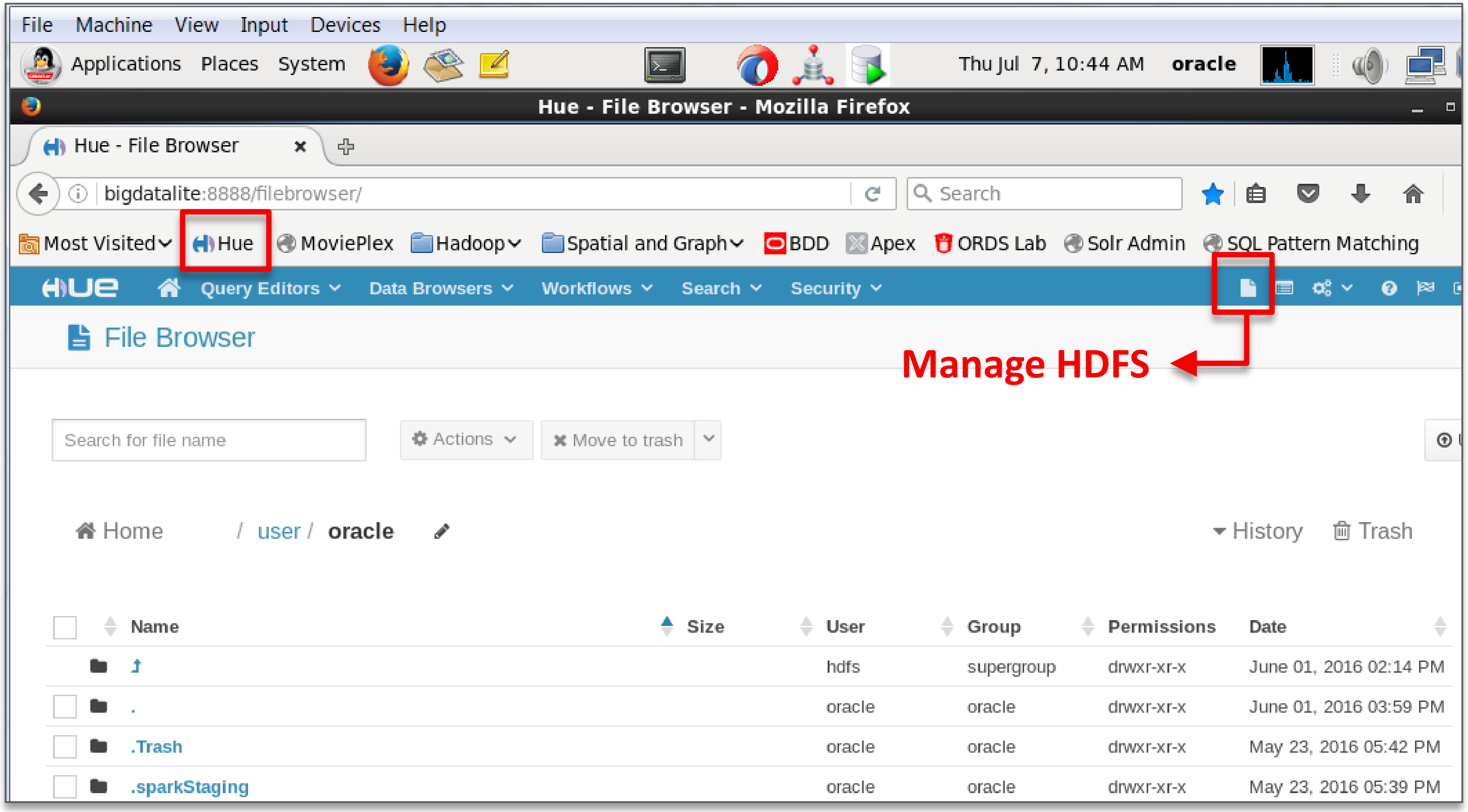
Configured Capacity	:	196.73 GB
DFS Used	:	2.18 GB
Non DFS Used	:	10.82 GB
DFS Remaining	:	183.74 GB
DFS Used%	:	1.11%

Agenda

- Understand the architectural components of HDFS
- Interact with data stored in HDFS
  - Hue
  - Hadoop client
  - WebHDFS
  - HttpFS

Using Cloudera Hue to Interact with HDFS

`http://bda1node03.example.com:8888`



Manage HDFS

bigdatalite:8888/filebrowser/

	Name	Size	User	Group	Permissions	Date
	↥		hdfs	supergroup	drwxr-xr-x	June 01, 2016 02:14 PM
□	.		oracle	oracle	drwxr-xr-x	June 01, 2016 03:59 PM
□	.Trash		oracle	oracle	drwxr-xr-x	May 23, 2016 05:42 PM
□	.sparkStaging		oracle	oracle	drwxr-xr-x	May 23, 2016 05:39 PM

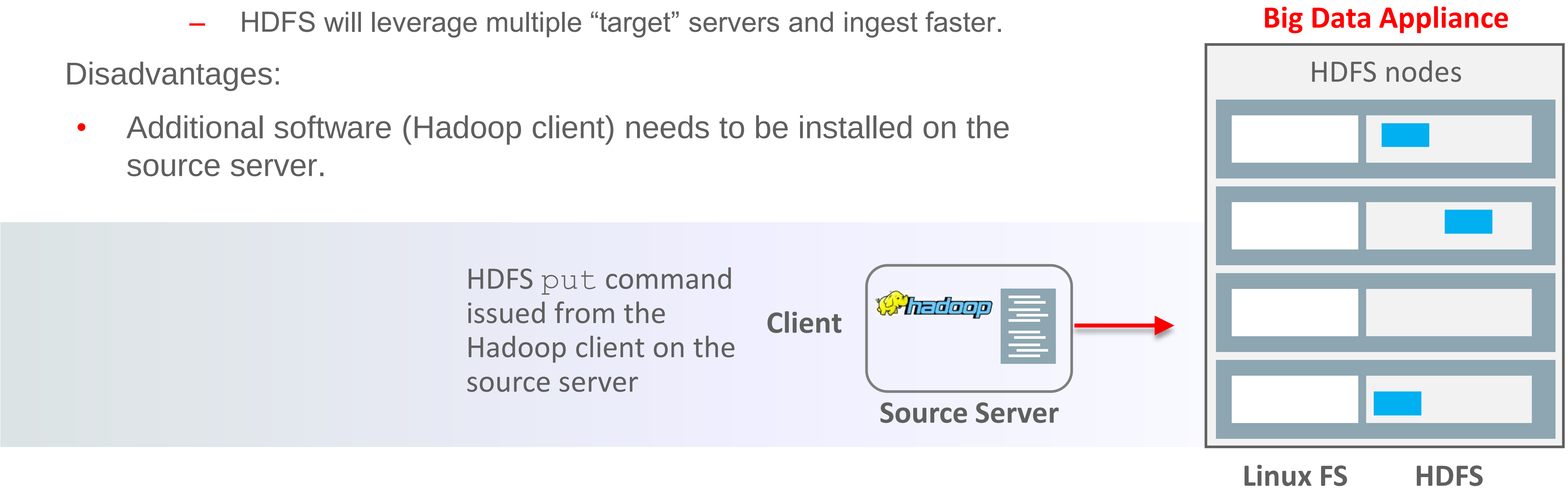
Using Hadoop Client to Batch Load Data

Advantages:

- Enables direct HDFS writes without intermediate file staging on Linux FS
- Easy to scale:
  - Initiate concurrent puts for multiple files.
  - HDFS will leverage multiple “target” servers and ingest faster.

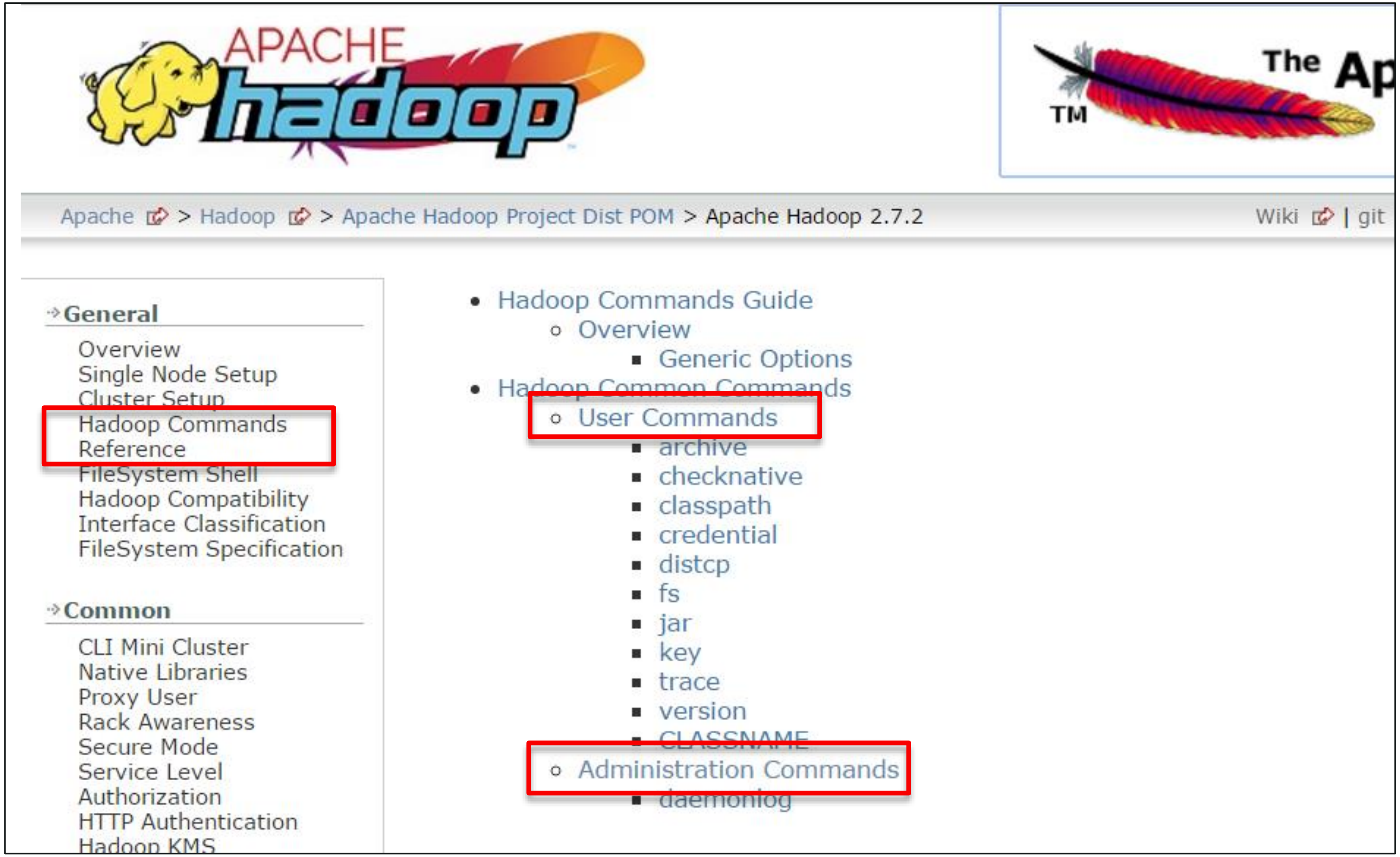
Disadvantages:

- Additional software (Hadoop client) needs to be installed on the source server.



HDFS put command issued from the Hadoop client on the source server
Client
hadoop
Source Server
Big Data Appliance
HDFS nodes
Linux FS
HDFS

HDFS Commands



Apache > Hadoop > Apache Hadoop Project Dist POM > Apache Hadoop 2.7.2

Wiki | git

General

Overview
Single Node Setup
Cluster Setup
Hadoop Commands Reference
FileSystem Shell
Hadoop Compatibility
Interface Classification
FileSystem Specification

Common

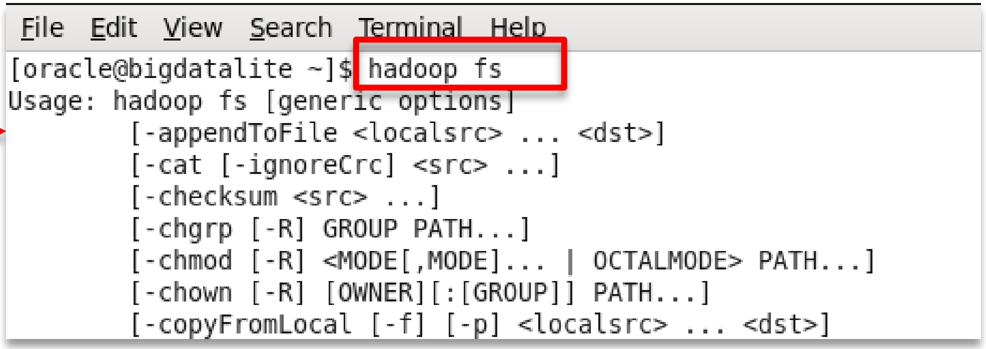
CLI Mini Cluster
Native Libraries
Proxy User
Rack Awareness
Secure Mode
Service Level Authorization
HTTP Authentication
Hadoop KMS

- Hadoop Commands Guide
  - Overview
    - Generic Options
- Hadoop Common Commands
  - User Commands
    - archive
    - checknative
    - classpath
    - credential
    - distcp
    - fs
    - jar
    - key
    - trace
    - version
    - CLASSNAME
  - Administration Commands
    - daemonlog

HDFS File System (FS) Shell Interface

- HDFS supports a traditional hierarchical file organization.
- You can use the **FS shell** command-line interface to interact with the data in HDFS.
- The syntax of this command set is similar to that of other shells.
  - You can create, remove, rename, and move directories/files.
- You can invoke FS shell as follows:

```
hadoop fs <args>
```



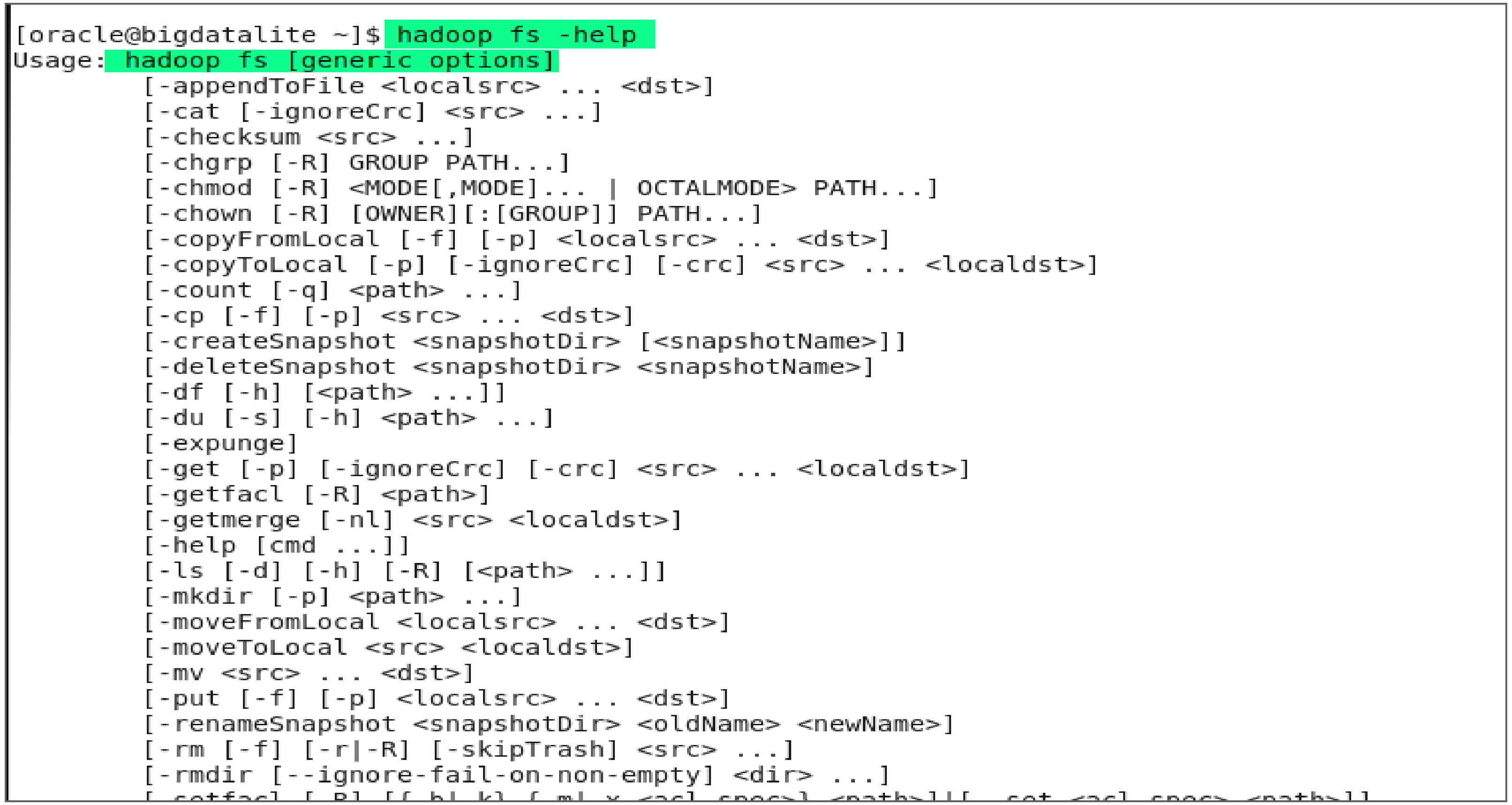
```
[oracle@bigdatalite ~]$ hadoop fs
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
```

- The general command-line syntax is as follows:

```
hadoop command [genericOptions] [commandOptions]
```

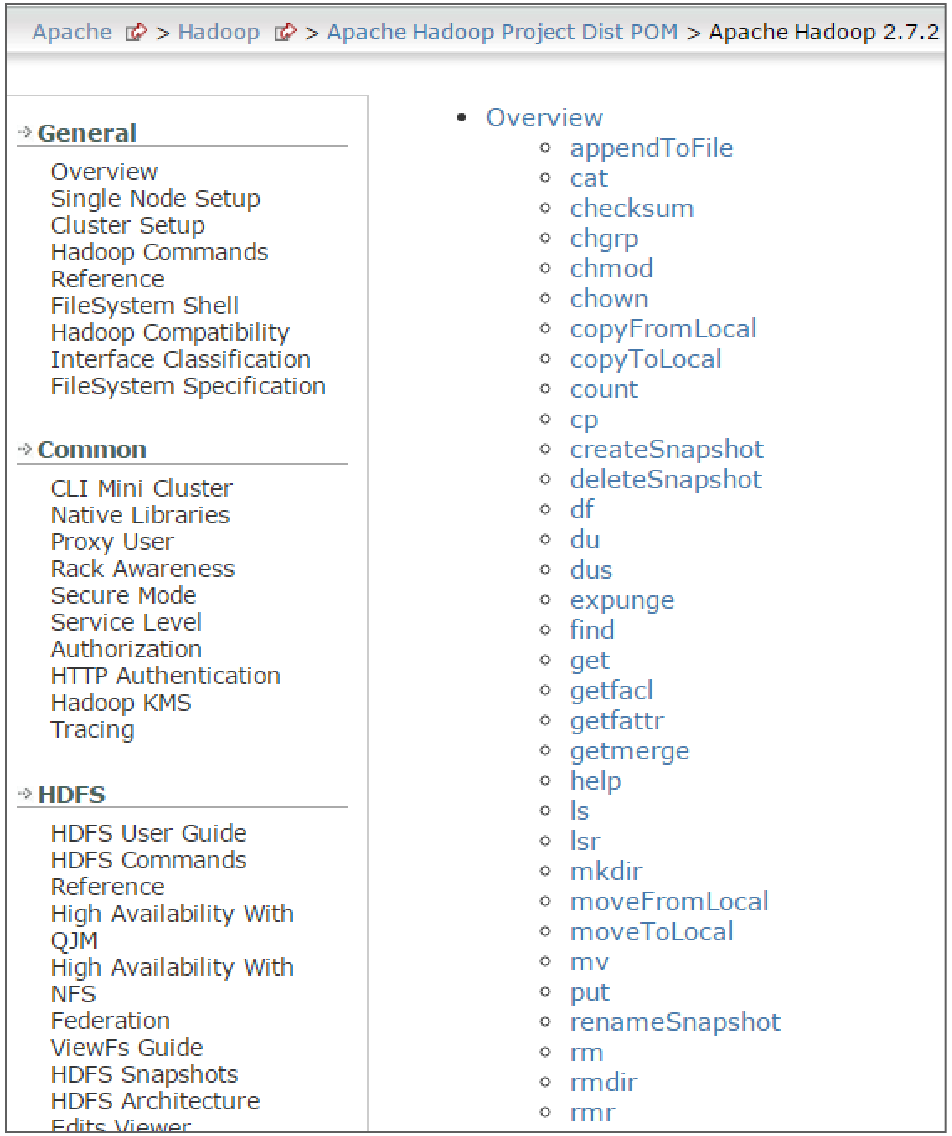
HDFS FS (File System) Shell Interface

```
hadoop fs -help
```



```
[oracle@bigdatalite ~]$ hadoop fs -help
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] <path> ...]
        [-cp [-f] [-p] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
        [-moveToLocal <src> <localdst>]
        [-mv <src> ... <dst>]
        [-put [-f] [-p] <localsrc> ... <dst>]
        [-renameSnapshot <snapshotDir> <oldName> <newName>]
        [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
        [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
        [-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]
```

FS Shell Commands



Apache > Hadoop > Apache Hadoop Project Dist POM > Apache Hadoop 2.7.2

General

- Overview
- Single Node Setup
- Cluster Setup
- Hadoop Commands Reference
- FileSystem Shell
- Hadoop Compatibility
- Interface Classification
- FileSystem Specification

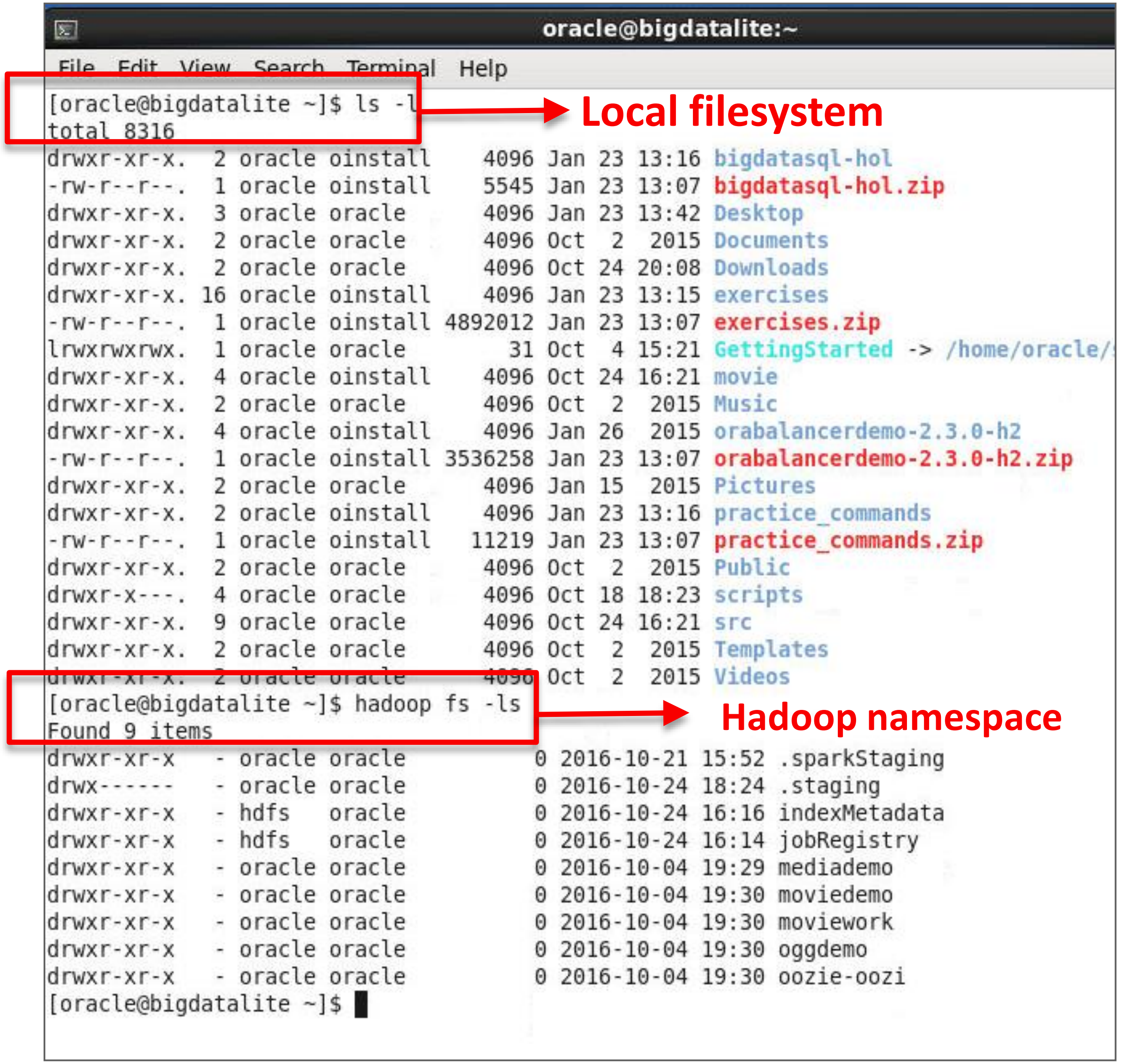
Common

- CLI Mini Cluster
- Native Libraries
- Proxy User
- Rack Awareness
- Secure Mode
- Service Level Authorization
- HTTP Authentication
- Hadoop KMS
- Tracing

HDFS

- HDFS User Guide
- HDFS Commands Reference
- High Availability With QJM
- High Availability With NFS
- Federation
- ViewFs Guide
- HDFS Snapshots
- HDFS Architecture
- Edits Viewer

- Overview
    - appendToFile
    - cat
    - checksum
    - chgrp
    - chmod
    - chown
    - copyFromLocal
    - copyToLocal
    - count
    - cp
    - createSnapshot
    - deleteSnapshot
    - df
    - du
    - dus
    - expunge
    - find
    - get
    - getfacl
    - getfattr
    - getmerge
    - help
    - ls
    - lsr
    - mkdir
    - moveFromLocal
    - moveToLocal
    - mv
    - put
    - renameSnapshot
    - rm
    - rmdir
    - rmr



```
[oracle@bigdatalite ~]$ ls -l
total 8316
drwxr-xr-x.  2 oracle oinstall    4096 Jan 23 13:16 bigdatasql-hol
-rw-r--r--.  1 oracle oinstall    5545 Jan 23 13:07 bigdatasql-hol.zip
drwxr-xr-x.  3 oracle oracle      4096 Jan 23 13:42 Desktop
drwxr-xr-x.  2 oracle oracle      4096 Oct  2  2015 Documents
drwxr-xr-x.  2 oracle oracle      4096 Oct 24 20:08 Downloads
drwxr-xr-x. 16 oracle oinstall    4096 Jan 23 13:15 exercises
-rw-r--r--.  1 oracle oinstall 4892012 Jan 23 13:07 exercises.zip
lrwxrwxrwx.  1 oracle oracle        31 Oct  4 15:21 GettingStarted -> /home/oracle/
drwxr-xr-x.  4 oracle oinstall    4096 Oct 24 16:21 movie
drwxr-xr-x.  2 oracle oracle      4096 Oct  2  2015 Music
drwxr-xr-x.  4 oracle oinstall    4096 Jan 26  2015 orabalancerdemo-2.3.0-h2
-rw-r--r--.  1 oracle oinstall 3536258 Jan 23 13:07 orabalancerdemo-2.3.0-h2.zip
drwxr-xr-x.  2 oracle oracle      4096 Jan 15  2015 Pictures
drwxr-xr-x.  2 oracle oinstall    4096 Jan 23 13:16 practice_commands
-rw-r--r--.  1 oracle oinstall   11219 Jan 23 13:07 practice_commands.zip
drwxr-xr-x.  2 oracle oracle      4096 Oct  2  2015 Public
drwxr-x---.  4 oracle oracle      4096 Oct 18 18:23 scripts
drwxr-xr-x.  9 oracle oracle      4096 Oct 24 16:21 src
drwxr-xr-x.  2 oracle oracle      4096 Oct  2  2015 Templates
drwxr-xr-x.  2 oracle oracle      4096 Oct  2  2015 Videos
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 9 items
drwxr-xr-x   - oracle oracle          0 2016-10-21 15:52 .sparkStaging
drwx------   - oracle oracle          0 2016-10-24 18:24 .staging
drwxr-xr-x   - hdfs   oracle          0 2016-10-24 16:16 indexMetadata
drwxr-xr-x   - hdfs   oracle          0 2016-10-24 16:14 jobRegistry
drwxr-xr-x   - oracle oracle          0 2016-10-04 19:29 mediademo
drwxr-xr-x   - oracle oracle          0 2016-10-04 19:30 moviedemo
drwxr-xr-x   - oracle oracle          0 2016-10-04 19:30 moviework
drwxr-xr-x   - oracle oracle          0 2016-10-04 19:30 oggdemo
drwxr-xr-x   - oracle oracle          0 2016-10-04 19:30 oozie-oozi
[oracle@bigdatalite ~]$ █
```

Local filesystem

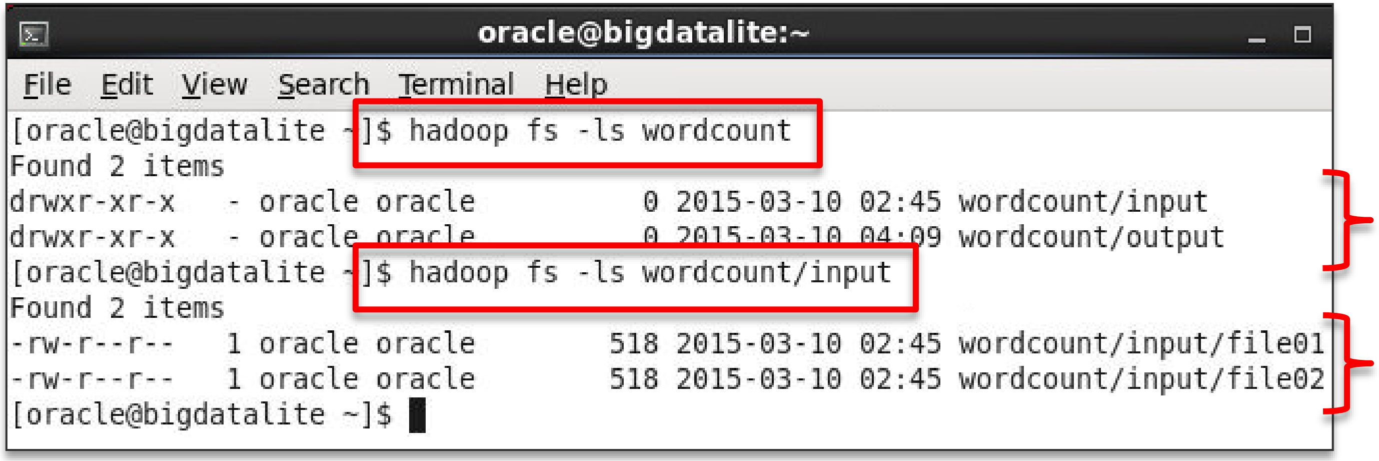
Hadoop namespace

Sample FS Shell Commands

Command	Description
`ls`	Lists attributes of files and directories
`cat`	Copies source paths to `stdout`
`cp`	Copy files from source to destination in HDFS
`mv`	Moves files from source to destination. Moving files across file systems is not permitted.
`rm`	Deletes files specified. The `-r` option deletes the directory and its contents.
`put`	Copies files from the local file system to HDFS
`get`	Copies files from HDFS to the local file system
`mkdir`	Creates one or more HDFS directories
`rmdir`	Deletes a directory
`jar`	Runs a jar file. Users can bundle their MapReduce code in a JAR file and execute it using this command.
`version`	Prints the Hadoop version
`help`	Return usage output (available commands to use)

`ls` Command

```
hadoop fs -ls
```



```
[oracle@bigdatalite ~]$ hadoop fs -ls wordcount
Found 2 items
drwxr-xr-x   - oracle oracle          0 2015-03-10 02:45 wordcount/input
drwxr-xr-x   - oracle oracle          0 2015-03-10 04:09 wordcount/output
[oracle@bigdatalite ~]$ hadoop fs -ls wordcount/input
Found 2 items
-rw-r--r--   1 oracle oracle        518 2015-03-10 02:45 wordcount/input/file01
-rw-r--r--   1 oracle oracle        518 2015-03-10 02:45 wordcount/input/file02
[oracle@bigdatalite ~]$ 
```

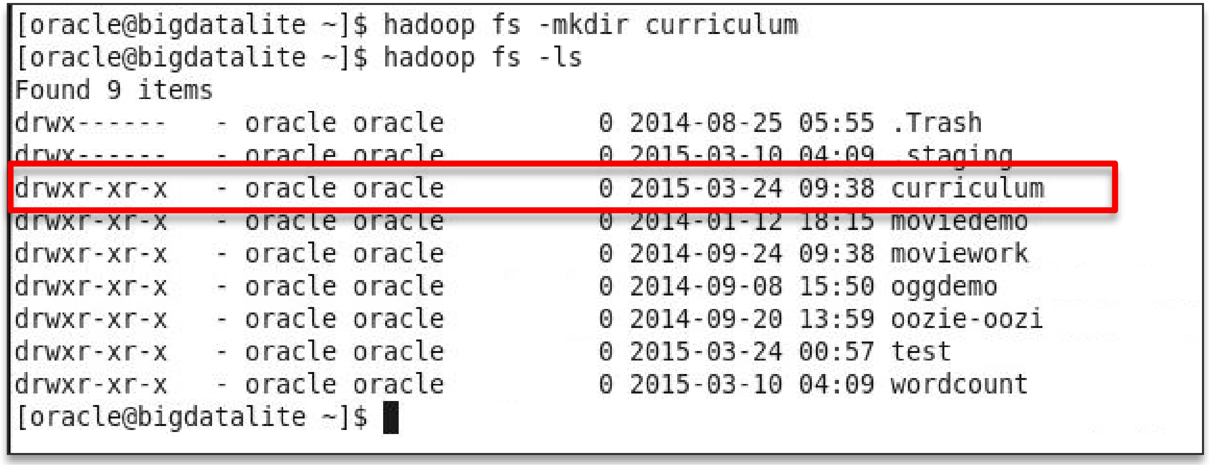
directories

files

- For a file, it returns `stat` on the file with the following format:
  - `permissions number_of_replicas userid groupid filesize modification_date modification_time filename`
- For a directory, it returns a list of its direct children as in UNIX. A directory is listed as:
  - `permissions userid groupid modification_date modification_time dirname`

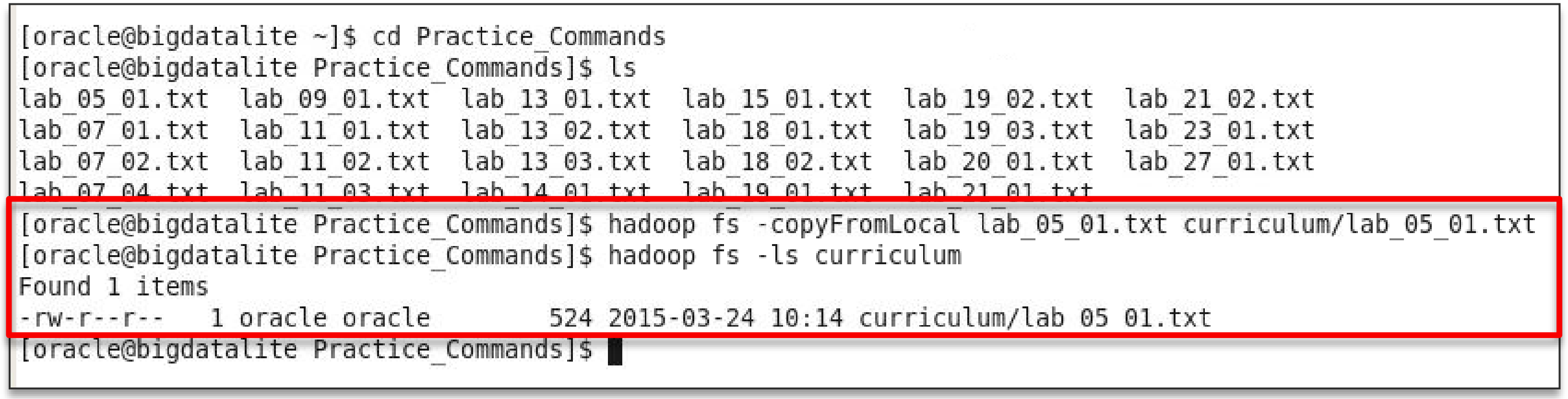
mkdir and copyFromLocal Commands

Create an HDFS directory named curriculum by using the mkdir command:



```
[oracle@bigdatalite ~]$ hadoop fs -mkdir curriculum
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 9 items
drwx------   - oracle oracle          0 2014-08-25 05:55 .Trash
drwx------   - oracle oracle          0 2015-03-10 04:09 .staging
drwxr-xr-x   - oracle oracle          0 2015-03-24 09:38 curriculum
drwxr-xr-x   - oracle oracle          0 2014-01-12 18:15 moviedemo
drwxr-xr-x   - oracle oracle          0 2014-09-24 09:38 moviework
drwxr-xr-x   - oracle oracle          0 2014-09-08 15:50 oggdemo
drwxr-xr-x   - oracle oracle          0 2014-09-20 13:59 oozie-oozi
drwxr-xr-x   - oracle oracle          0 2015-03-24 00:57 test
drwxr-xr-x   - oracle oracle          0 2015-03-10 04:09 wordcount
[oracle@bigdatalite ~]$ 
```

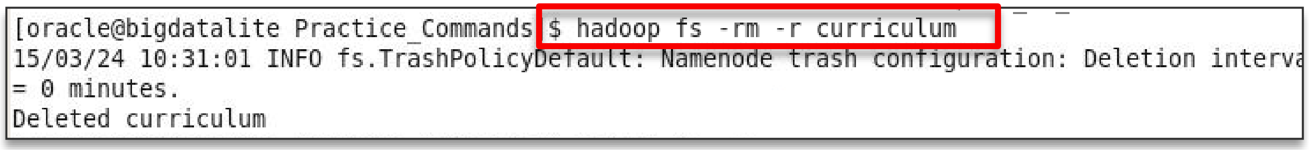
Copy lab_05_01.txt from the local file system to the curriculum HDFS directory by using the copyFromLocal command:



```
[oracle@bigdatalite ~]$ cd Practice_Commands
[oracle@bigdatalite Practice_Commands]$ ls
lab_05_01.txt  lab_09_01.txt  lab_13_01.txt  lab_15_01.txt  lab_19_02.txt  lab_21_02.txt
lab_07_01.txt  lab_11_01.txt  lab_13_02.txt  lab_18_01.txt  lab_19_03.txt  lab_23_01.txt
lab_07_02.txt  lab_11_02.txt  lab_13_03.txt  lab_18_02.txt  lab_20_01.txt  lab_27_01.txt
lab_07_04.txt  lab_11_03.txt  lab_14_01.txt  lab_19_01.txt  lab_21_01.txt
[oracle@bigdatalite Practice_Commands]$ hadoop fs -copyFromLocal lab_05_01.txt curriculum/lab_05_01.txt
[oracle@bigdatalite Practice_Commands]$ hadoop fs -ls curriculum
Found 1 items
-rw-r--r--   1 oracle oracle        524 2015-03-24 10:14 curriculum/lab_05_01.txt
[oracle@bigdatalite Practice_Commands]$ 
```

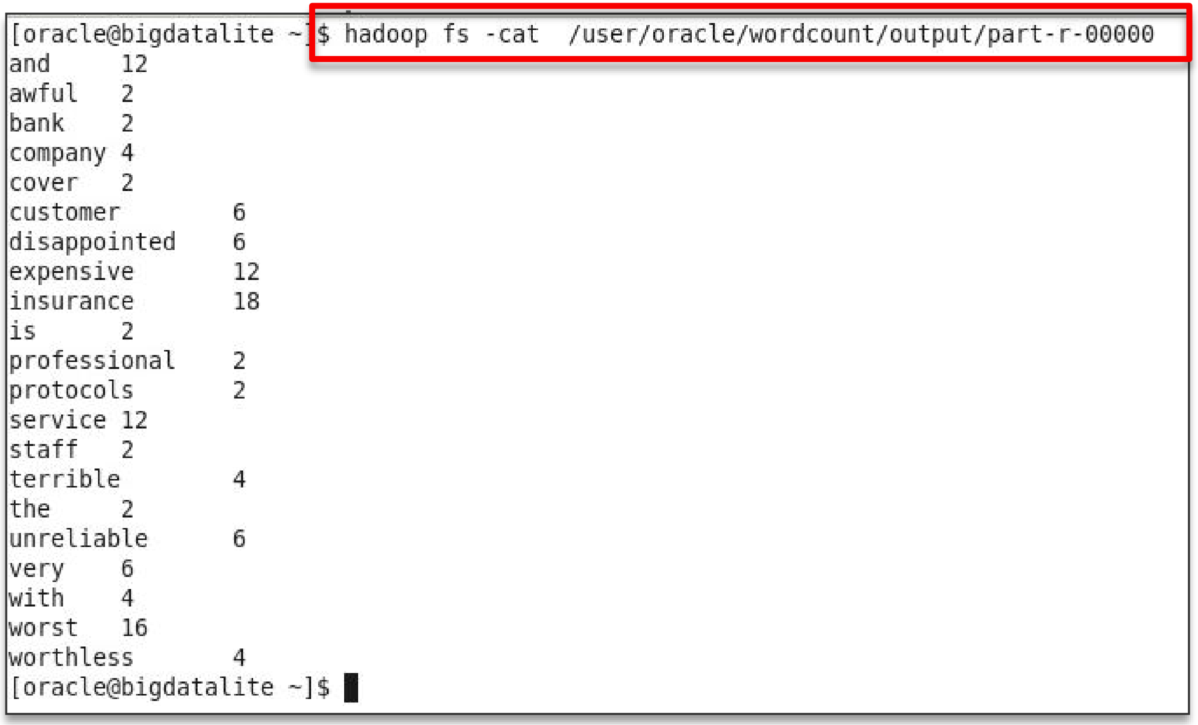
rm and cat Commands

Delete the curriculum HDFS directory by using the rm command.
Use the -r option to delete the directory and any content under it



```
[oracle@bigdatalite Practice_Commands]$ hadoop fs -rm -r curriculum
15/03/24 10:31:01 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interva
= 0 minutes.
Deleted curriculum
```

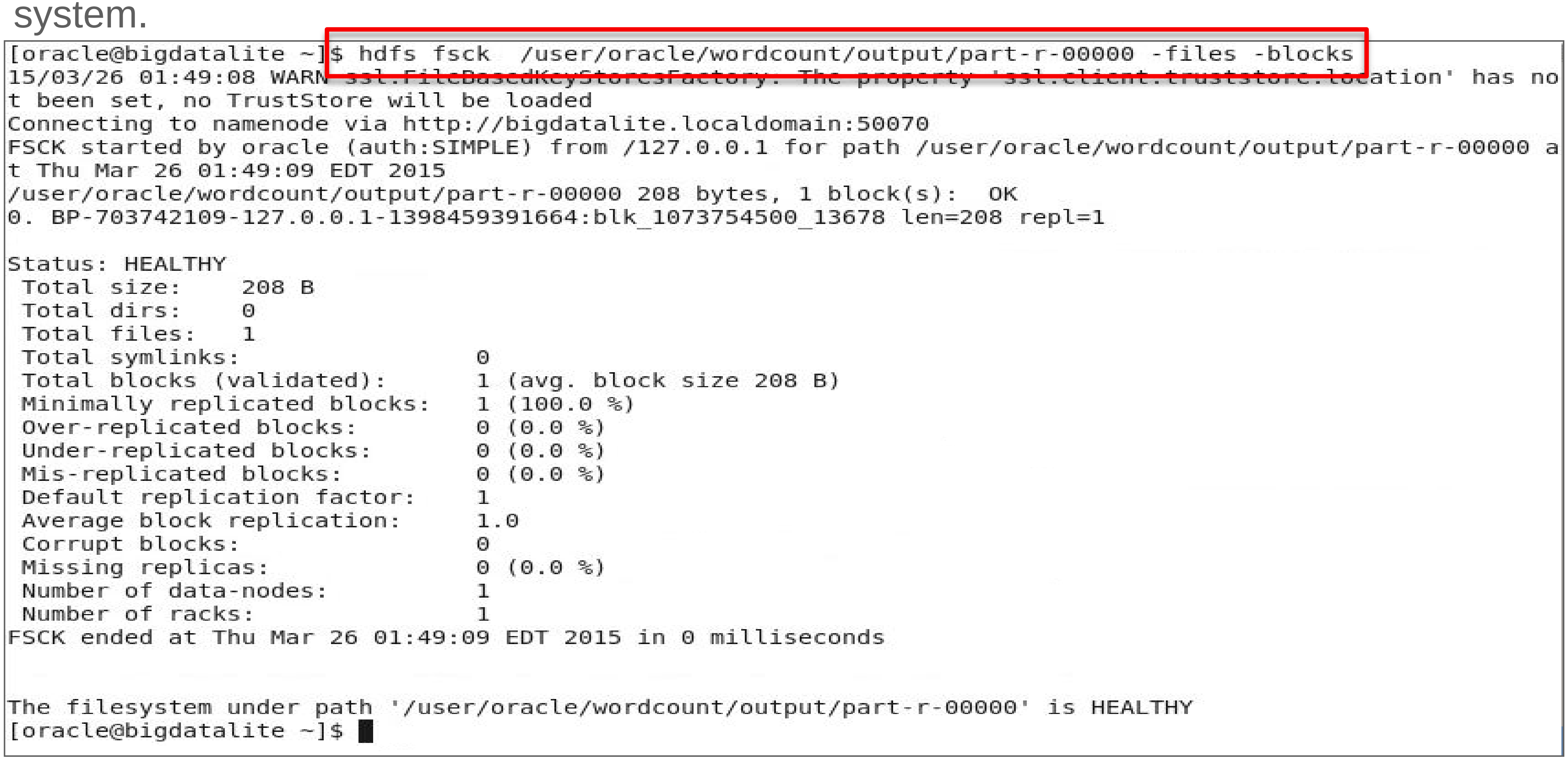
Display the contents of the part-r-00000 HDFS file by using the



```
[oracle@bigdatalite ~]$ hadoop fs -cat  /user/oracle/wordcount/output/part-r-00000
and     12
awful   2
bank    2
company 4
cover   2
customer        6
disappointed    6
expensive       12
insurance       18
is      2
professional    2
protocols       2
service 12
staff   2
terrible        4
the     2
unreliable      6
very    6
with    4
worst   16
worthless       4
[oracle@bigdatalite ~]$ 
```

Using the `hdfs fsck` Command: Example

Use the `hdfs fsck` file system checking utility to perform health checks on the file system.



```
[oracle@bigdatalite ~]$ hdfs fsck  /user/oracle/wordcount/output/part-r-00000 -files -blocks
15/03/26 01:49:08 WARN ssl.FileBasedKeyStoresFactory: The property 'ssl.client.truststore.location' has no
t been set, no TrustStore will be loaded
Connecting to namenode via http://bigdatalite.localdomain:50070
FSCK started by oracle (auth:SIMPLE) from /127.0.0.1 for path /user/oracle/wordcount/output/part-r-00000 a
t Thu Mar 26 01:49:09 EDT 2015
/user/oracle/wordcount/output/part-r-00000 208 bytes, 1 block(s):  OK
0. BP-703742109-127.0.0.1-1398459391664:blk_1073754500_13678 len=208 repl=1

Status: HEALTHY
 Total size:    208 B
 Total dirs:    0
 Total files:   1
 Total symlinks:                0
 Total blocks (validated):      1 (avg. block size 208 B)
 Minimally replicated blocks:   1 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    1
 Average block replication:     1.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          1
 Number of racks:               1
FSCK ended at Thu Mar 26 01:49:09 EDT 2015 in 0 milliseconds


The filesystem under path '/user/oracle/wordcount/output/part-r-00000' is HEALTHY
[oracle@bigdatalite ~]$
```

Agenda

- Understand the architectural components of HDFS
- Interact with data stored in HDFS
  - Hue
  - Hadoop client
  - WebHDFS
  - HttpFS

Loading Data with WebHDFS or HttpFS

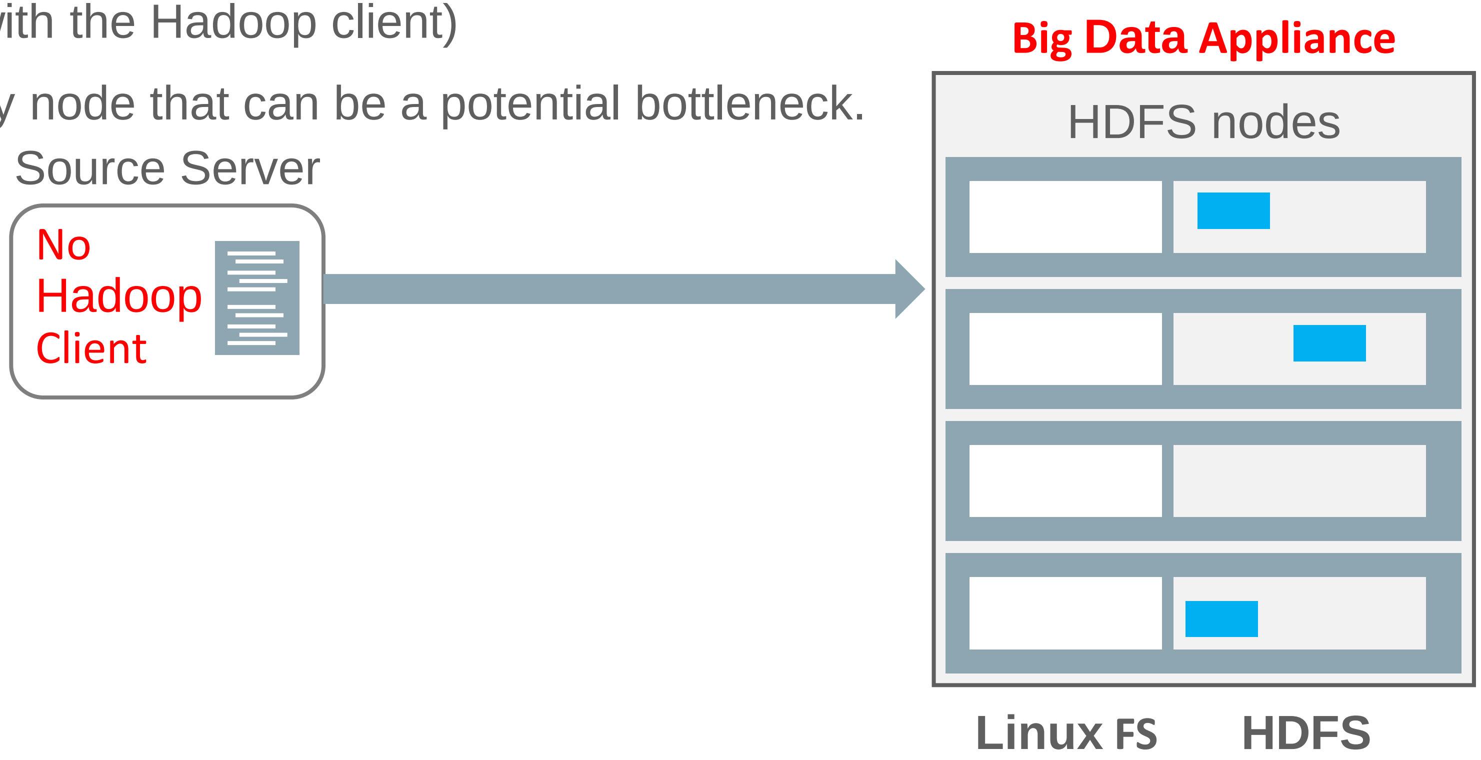
Advantages:

- WebHDFS performance comparable with the Hadoop client
- No additional software required on the client side

Disadvantages:

- Complex syntax (comparable with the Hadoop client)
- HttpFS utilizes a single gateway node that can be a potential bottleneck.

Initiate data loading on the client side with curl command.



Source Server

No Hadoop Client

Big Data Appliance

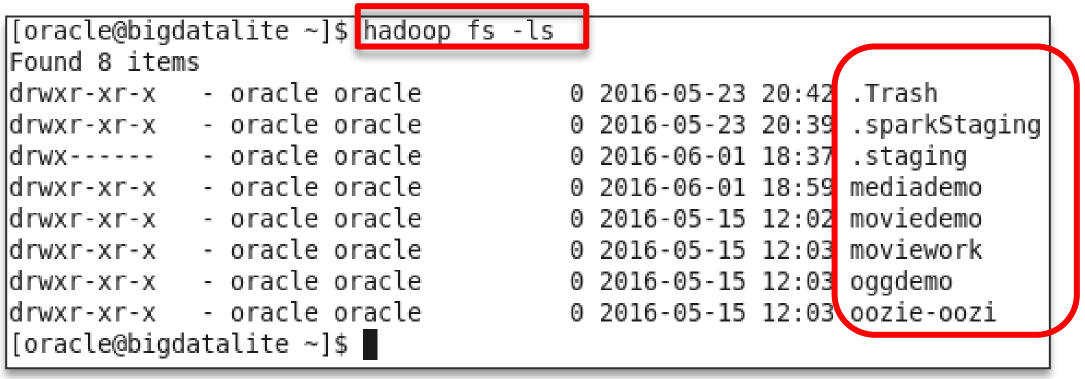
HDFS nodes

Linux FS

HDFS

hadoop fs -ls and LISTSTATUS

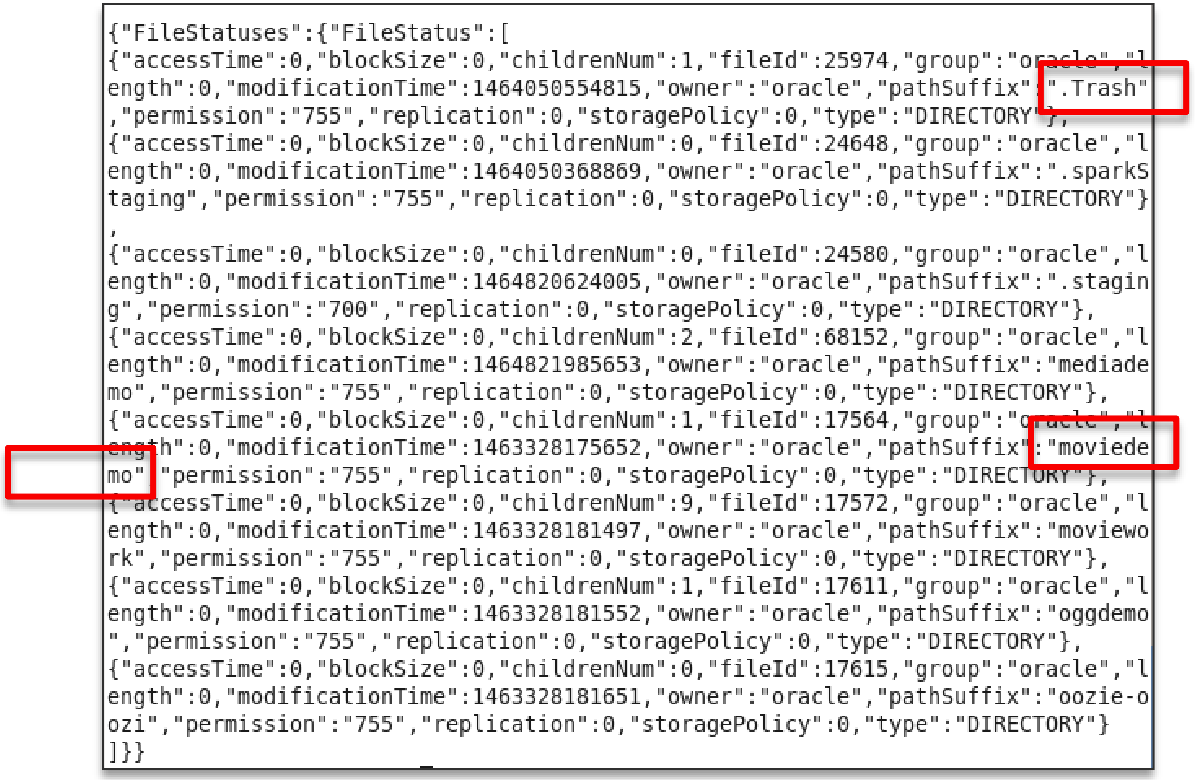
```
hadoop fs -ls
```



```
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 8 items
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:42 .Trash
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:39 .sparkStaging
drwx------   - oracle oracle          0 2016-06-01 18:37 .staging
drwxr-xr-x   - oracle oracle          0 2016-06-01 18:59 mediademo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:02 moviedemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 moviework
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oggdemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oozie-oozi
[oracle@bigdatalite ~]$ █
```

```
curl -i
    "http://bigdatalite.localdomain:50070/webhdfs/v1/
    user/oracle?op=LISTSTATUS"
```

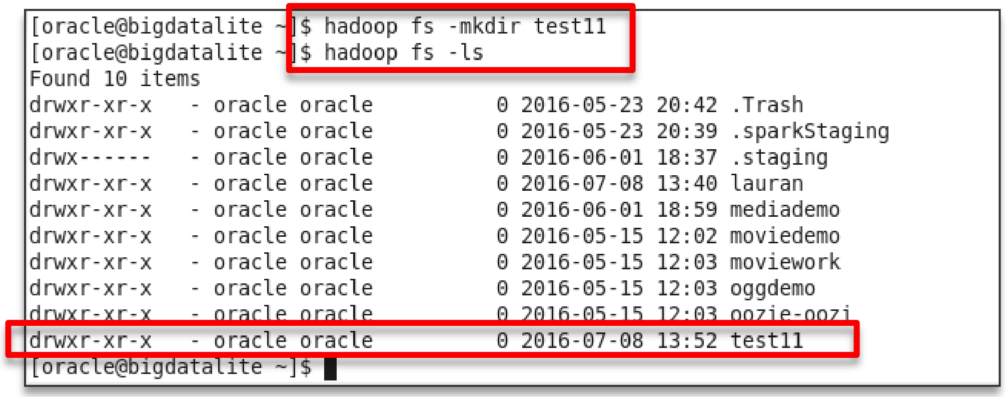
LISTSTATUS displays the same content of the hadoop fs -ls commend but in JSON format.



```
{"FileStatuses":{"FileStatus":[
{"accessTime":0,"blockSize":0,"childrenNum":1,"fileId":25974,"group":"oracle","l
ength":0,"modificationTime":1464050554815,"owner":"oracle","pathSuffix":".Trash"
,"permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":0,"fileId":24648,"group":"oracle","l
ength":0,"modificationTime":1464050368869,"owner":"oracle","pathSuffix":".sparkS
taging","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"}
,
{"accessTime":0,"blockSize":0,"childrenNum":0,"fileId":24580,"group":"oracle","l
ength":0,"modificationTime":1464820624005,"owner":"oracle","pathSuffix":".stagin
g","permission":"700","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":2,"fileId":68152,"group":"oracle","l
ength":0,"modificationTime":1464821985653,"owner":"oracle","pathSuffix":"mediade
mo","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":1,"fileId":17564,"group":"oracle","l
ength":0,"modificationTime":1463328175652,"owner":"oracle","pathSuffix":"moviede
mo","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":9,"fileId":17572,"group":"oracle","l
ength":0,"modificationTime":1463328181497,"owner":"oracle","pathSuffix":"moviewo
rk","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":1,"fileId":17611,"group":"oracle","l
ength":0,"modificationTime":1463328181552,"owner":"oracle","pathSuffix":"oggdemo
","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":0,"fileId":17615,"group":"oracle","l
ength":0,"modificationTime":1463328181651,"owner":"oracle","pathSuffix":"oozie-o
ozi","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"}
]}}
```

Uploading a Local File to an HDFS Directory with `hadoop fs`

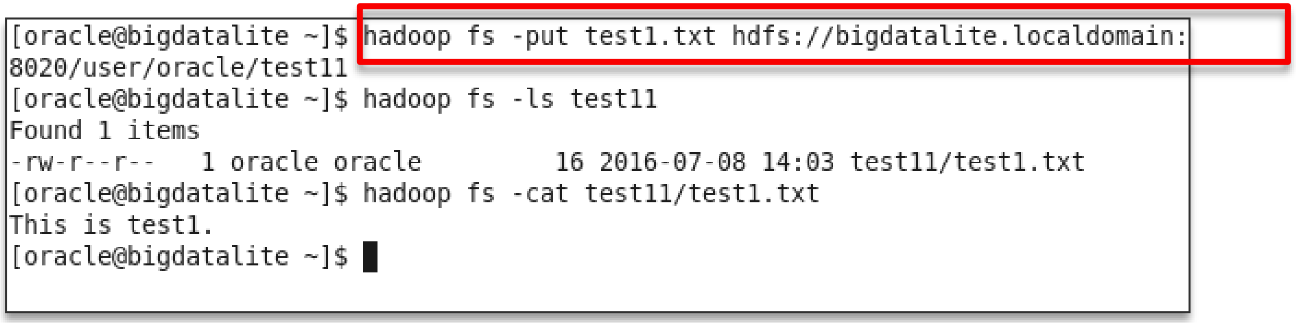
Create an HDFS directory named `test11` using hadoop fs CLI:



```
[oracle@bigdatalite ~]$ hadoop fs -mkdir test11
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 10 items
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:42 .Trash
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:39 .sparkStaging
drwx------   - oracle oracle          0 2016-06-01 18:37 .staging
drwxr-xr-x   - oracle oracle          0 2016-07-08 13:40 lauran
drwxr-xr-x   - oracle oracle          0 2016-06-01 18:59 mediademo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:02 moviedemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 moviework
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oggdemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oozie-oozi
drwxr-xr-x   - oracle oracle          0 2016-07-08 13:52 test11
[oracle@bigdatalite ~]$ 
```

Copying the local `test1.txt` file to HDFS directory `test11` using `hadoop fs` CLI:

```
hadoop fs -put test1.txt
    hdfs://bigdatalite.localdomain:8020/user/oracle/test11
```



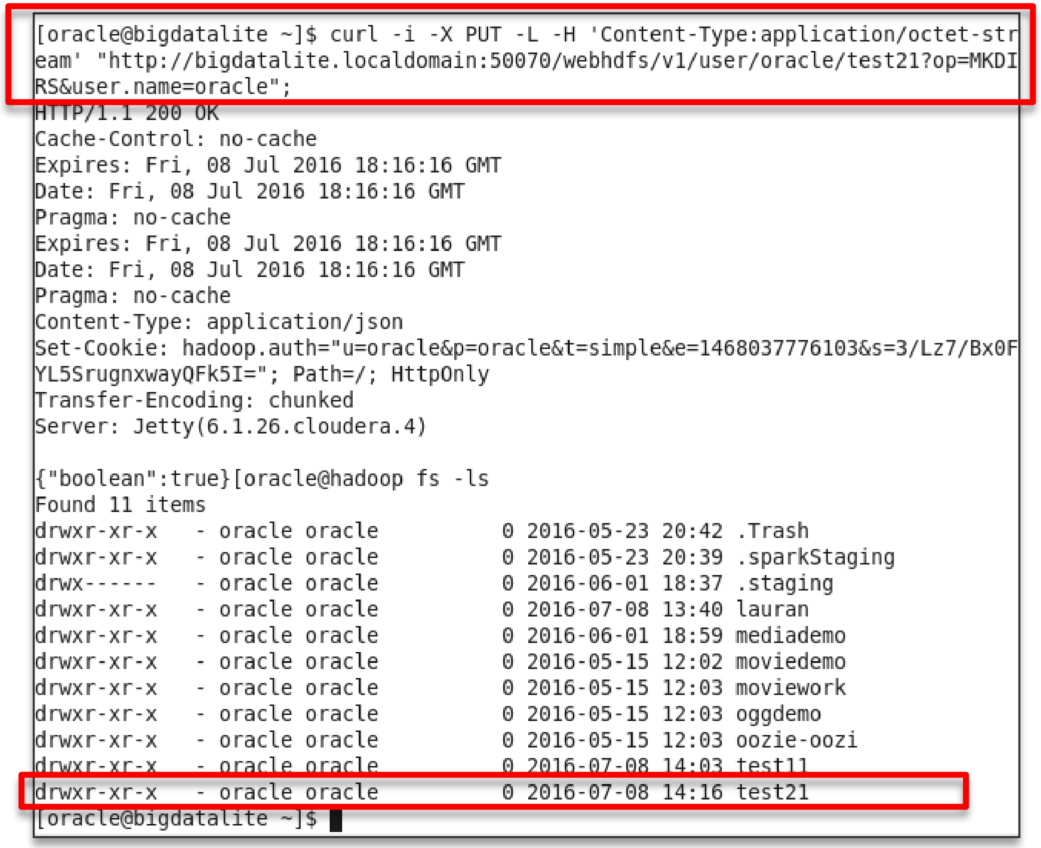
```
[oracle@bigdatalite ~]$ hadoop fs -put test1.txt hdfs://bigdatalite.localdomain:
8020/user/oracle/test11
[oracle@bigdatalite ~]$ hadoop fs -ls test11
Found 1 items
-rw-r--r--   1 oracle oracle         16 2016-07-08 14:03 test11/test1.txt
[oracle@bigdatalite ~]$ hadoop fs -cat test11/test1.txt
This is test1.
[oracle@bigdatalite ~]$ 
```

Confirm file upload
and
view its content

Creating an HDFS Directory with WebHDFS

Creating an HDFS directory named `test21` by using WebHDFS:

```
curl -i -X PUT -L -H 'Content-Type:application/octet-stream'
"http://bigdatalite.localdomain:50070/webhdfs/v1/user/oracle/test21?op=
MKDIRS&user.name=oracle";
```

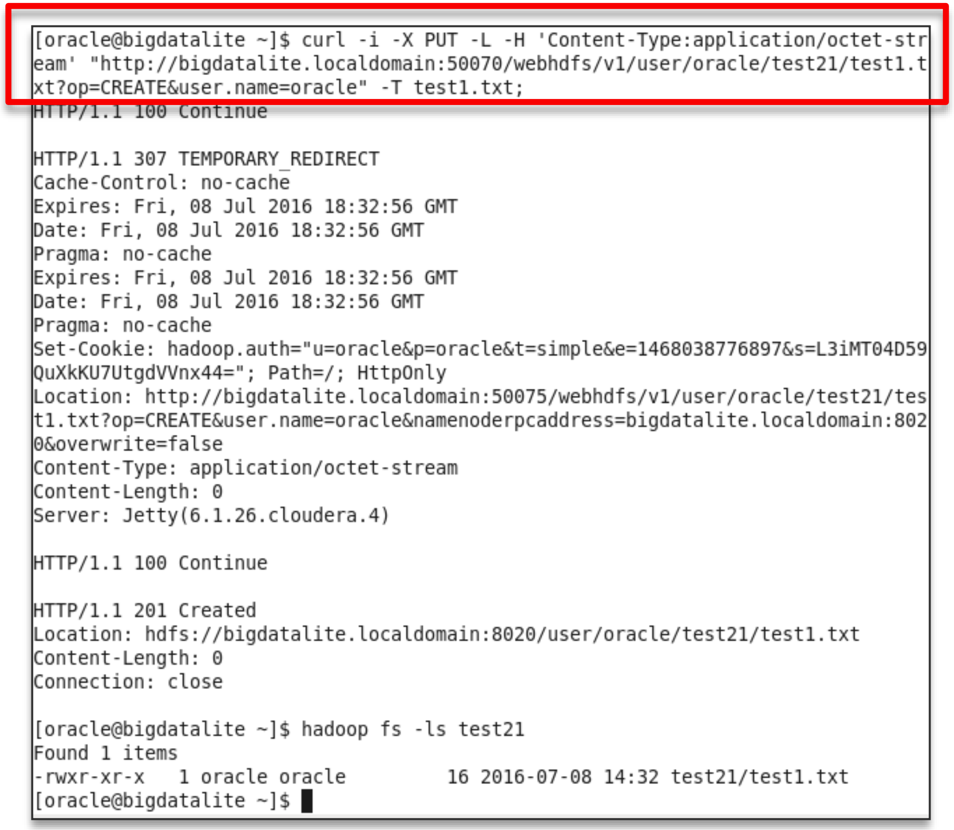


```
[oracle@bigdatalite ~]$ curl -i -X PUT -L -H 'Content-Type:application/octet-str
eam' "http://bigdatalite.localdomain:50070/webhdfs/v1/user/oracle/test21?op=MKDI
RS&user.name=oracle";
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Fri, 08 Jul 2016 18:16:16 GMT
Date: Fri, 08 Jul 2016 18:16:16 GMT
Pragma: no-cache
Expires: Fri, 08 Jul 2016 18:16:16 GMT
Date: Fri, 08 Jul 2016 18:16:16 GMT
Pragma: no-cache
Content-Type: application/json
Set-Cookie: hadoop.auth="u=oracle&p=oracle&t=simple&e=1468037776103&s=3/Lz7/Bx0F
YL5SrugnxwayQFk5I="; Path=/; HttpOnly
Transfer-Encoding: chunked
Server: Jetty(6.1.26.cloudera.4)

{"boolean":true}[oracle@hadoop fs -ls
Found 11 items
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:42 .Trash
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:39 .sparkStaging
drwx------   - oracle oracle          0 2016-06-01 18:37 .staging
drwxr-xr-x   - oracle oracle          0 2016-07-08 13:40 lauran
drwxr-xr-x   - oracle oracle          0 2016-06-01 18:59 mediademo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:02 moviedemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 moviework
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oggdemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oozie-oozi
drwxr-xr-x   - oracle oracle          0 2016-07-08 14:03 test11
drwxr-xr-x   - oracle oracle          0 2016-07-08 14:16 test21
[oracle@bigdatalite ~]$ 
```

Uploading a Local File to HDFS with WebHDFS

Creating an HDFS directory named `test21` by using WebHDFS:

```
curl -i -X PUT -L -H 'Content-Type:application/octet-stream'
"http://bigdatalite.localdomain:50070/webhdfs/v1/user/oracle/test21/tes
t1.txt?op=CREATE&user.name=oracle" -T test1.txt;
```

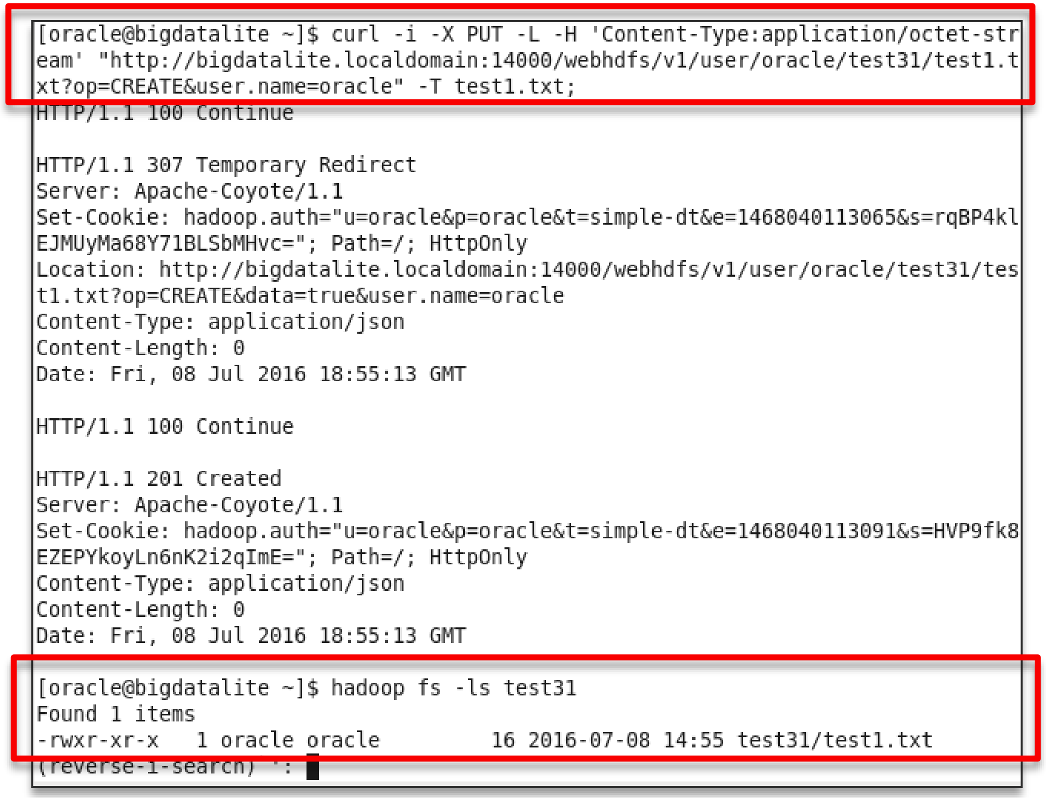


```
[oracle@bigdatalite ~]$ curl -i -X PUT -L -H 'Content-Type:application/octet-str
eam' "http://bigdatalite.localdomain:50070/webhdfs/v1/user/oracle/test21/test1.t
xt?op=CREATE&user.name=oracle" -T test1.txt;
HTTP/1.1 100 Continue

HTTP/1.1 307 TEMPORARY_REDIRECT
Cache-Control: no-cache
Expires: Fri, 08 Jul 2016 18:32:56 GMT
Date: Fri, 08 Jul 2016 18:32:56 GMT
Pragma: no-cache
Expires: Fri, 08 Jul 2016 18:32:56 GMT
Date: Fri, 08 Jul 2016 18:32:56 GMT
Pragma: no-cache
Set-Cookie: hadoop.auth="u=oracle&p=oracle&t=simple&e=1468038776897&s=L3iMT04D59
QuXkKU7UtgdVVnx44="; Path=/; HttpOnly
Location: http://bigdatalite.localdomain:50075/webhdfs/v1/user/oracle/test21/tes
t1.txt?op=CREATE&user.name=oracle&namenoderpcaddress=bigdatalite.localdomain:802
0&overwrite=false
Content-Type: application/octet-stream
Content-Length: 0
Server: Jetty(6.1.26.cloudera.4)

HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Location: hdfs://bigdatalite.localdomain:8020/user/oracle/test21/test1.txt
Content-Length: 0
Connection: close

[oracle@bigdatalite ~]$ hadoop fs -ls test21
Found 1 items
-rwxr-xr-x   1 oracle oracle         16 2016-07-08 14:32 test21/test1.txt
[oracle@bigdatalite ~]$ █
```

Creating an HDFS Directory and Loading Data by Using HttpFS

Creating an HDFS directory named `test31` by using HttpFS and uploading `test1.txt` to /`test31` HDFS directory

```
curl -i -X PUT -L -H 'Content-Type:application/octet-stream'
"http://bigdatalite.localdomain:14000/webhdfs/v1/user/oracle/test31/tes
t1.txt?op=CREATE&user.name=oracle" -T test1.txt;
```



```
[oracle@bigdatalite ~]$ curl -i -X PUT -L -H 'Content-Type:application/octet-str
eam' "http://bigdatalite.localdomain:14000/webhdfs/v1/user/oracle/test31/test1.t
xt?op=CREATE&user.name=oracle" -T test1.txt;
HTTP/1.1 100 Continue

HTTP/1.1 307 Temporary Redirect
Server: Apache-Coyote/1.1
Set-Cookie: hadoop.auth="u=oracle&p=oracle&t=simple-dt&e=1468040113065&s=rqBP4kl
EJMUyMa68Y71BLSbMHvc="; Path=/; HttpOnly
Location: http://bigdatalite.localdomain:14000/webhdfs/v1/user/oracle/test31/tes
t1.txt?op=CREATE&data=true&user.name=oracle
Content-Type: application/json
Content-Length: 0
Date: Fri, 08 Jul 2016 18:55:13 GMT

HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Set-Cookie: hadoop.auth="u=oracle&p=oracle&t=simple-dt&e=1468040113091&s=HVP9fk8
EZEPYkoyLn6nK2i2qImE="; Path=/; HttpOnly
Content-Type: application/json
Content-Length: 0
Date: Fri, 08 Jul 2016 18:55:13 GMT

[oracle@bigdatalite ~]$ hadoop fs -ls test31
Found 1 items
-rwxr-xr-x   1 oracle oracle         16 2016-07-08 14:55 test31/test1.txt
(reverse-i-search)`':
```

HttpFS uses default port 14000

Summary

In this lesson, you should have learned how to:

- Describe the architectural components of HDFS
- Use the **FS shell** command-line interface (CLI) to interact with data stored in HDFS

Practice 5: Overview

In this practice, you load a JSON log file into HDFS. This log file was used to track activity in an online movie application.