Using HDFS

|  |  |
| --- | --- |
| **Data Files** | ~/materials/data/shakespeare.tar.gz  ~/materials/data/access\_log.gz |

**In this exercise you will begin to get acquainted with the Hadoop. You will manipulate files in HDFS, the Hadoop Distributed File System.**

**Hadoop**

Hadoop is already installed, configured, and running on your virtual machine.

Most of your interaction with the system will be through a command-line wrapper called hadoop. If you run this program with no arguments, it prints a help message.

To try this, run the following command in a terminal window:

$ hadoop

The hadoop command is subdivided into several subsystems. For example, there is a subsystem for working with files in HDFS and another for launching and managing MapReduce processing jobs.

**Exploring HDFS**

The subsystem associated with HDFS in the Hadoop wrapper program is called FsShell. This subsystem can be invoked with the command hdfs dfs.

Open a terminal window (if one is not already open) by double-clicking the Terminal icon on the desktop.

In the terminal window, enter:

$ hdfs dfs

You see a help message describing all the commands associated with the

FsShell subsystem.

Enter:

$ hdfs dfs -ls

This shows you the contents of the directory in HDFS.

Try viewing the contents of the /user directory by running:

$ hdfs dfs -ls /user

You will see your home directory in the directory listing.

List the contents of your home directory by running:

$ hdfs dfs -ls -R

There are no files yet, so the command silently exits. This is different than if you

ran hdfs dfs -ls /foo, which refers to a directory that doesn’t exist and which

would display an error message.

NOTE: that the directory structure in HDFS has nothing to do with the directory

structure of the local filesystem; they are completely separate namespaces.

**Uploading Files**

Besides browsing the existing filesystem, another important thing you can do with FsShell is to upload new data into HDFS.

Change directories to the local filesystem directory containing the sample data we will be using in the course.

$ cd ~/materials/data

If you perform a regular Linux ls command in this directory, you will see a few files,

including two named shakespeare.tar.gz and shakespeare-

stream.tar.gz. Both of these contain the complete works of Shakespeare in text

format, but with different formats and organizations. For now we will work

with shakespeare.tar.gz.

Unzip shakespeare.tar.gz by running:

$ tar -zxvf shakespeare.tar.gz

This creates a directory named shakespeare/ containing several files on your local

filesystem.

Insert this directory into HDFS:

$ hdfs dfs -put shakespeare /user/[username]/shakespeare

This copies the local shakespeare directory and its contents into a remote, HDFS

directory named ~/shakespeare.

List the contents of your HDFS home directory now:

$ hdfs dfs -ls

You should see an entry for the shakespeare directory.

Now try the same -ls command but now add -R like this:

$ hdfs dfs –ls -R

You should see the same results, plus everything inside the previous list.

# 

# Relative Paths

If you pass any relative (non-absolute) paths to FsShell commands (or use

relative paths in MapReduce programs), they are considered relative to your

home directory.

We also have a Web server log file, which we will put into HDFS for use in future exercises. This file is currently compressed using GZip. Rather than extract the file to the local disk and then upload it, we will extract and upload in one step. First, create a directory in HDFS in which to store it:

$ hdfs dfs -mkdir weblog

Now, extract and upload the file. The -c option to gunzip uncompresses to standard output, and the dash (-) in the hdfs dfs -put command takes whatever is being sent to its standard input and places that data in HDFS.

$ cp access\_log.gz access\_log\_secondary.gz

$ gunzip access\_log.gz

$ hdfs dfs -put access\_log weblog/access\_log

Run the hdfs dfs -ls command to verify that the log file is in your HDFS home directory.

The access log file is quite large – around 500 MB. Create a smaller version of this file, consisting only of its first 5000 lines, and store the smaller version in HDFS. You can use the smaller version for testing in subsequent exercises.

$ hdfs dfs -mkdir testlog

# $ gunzip access\_log\_secondary.gz | head -n 5000

$ hdfs dfs -put access\_log\_secondary testlog/test\_access\_log

**Viewing and Manipulating Files**

Now let’s view some of the data you just copied into HDFS.

1. Enter:

$ hdfs dfs -ls shakespeare

This lists the contents of the ~/shakespeare HDFS directory, which consists of

the files comedies, glossary, histories, poems, and tragedies.

2. The glossary file included in the compressed file you began with is not strictly a work of Shakespeare, so let’s remove it:

$ hdfs dfs -rm shakespeare/glossary

NOTE: You could leave this file in place if you so wished. If you did, then it would be included

in subsequent computations across the works of Shakespeare, and would skew your results

slightly. As with many real-‐world big data problems, you make trade‐offs between the labor to

purify your input data and the precision of your results.

3. Enter:

$ hdfs dfs -cat shakespeare/histories | tail -n 50

This prints the last 50 lines of *Henry IV, Part 1* to your terminal. This command is

handy for viewing the output of MapReduce programs. Very often, an individual output

file of a MapReduce program is very large, making it inconvenient to view the entire file

in the terminal. For this reason, it’s often a good idea to pipe the output of the dfs -

cat command into head, tail, more, or less.

To download a file to work with on the local filesystem use the dfs -get command. This command takes two arguments: an HDFS path and a local path. It copies the HDFS contents into the local filesystem:

$ hdfs dfs -get shakespeare/poems ~/shakepoems.txt

$ less ~/shakepoems.txt

**Other Commands**

There are several other operations available with the hdfs dfs command to perform most common filesystem manipulations: mv, cp, mkdir, etc.

1. Enter:

$ hdfs dfs

This displays a brief usage report of the commands available within FsShell. Try playing around with a few of these commands if you like.

**END**