

# Exercise 6

*Get started with Hadoop*

## Prior Knowledge

Unix Command Line Shell

Simple Python

## Learning Objectives

Understand how to start and stop HDFS, and transfer files into and out of the Hadoop filesystem.

Understand how to create a Python mapper/reducer and execute this on a single-node setup of Hadoop.

Stop and start YARN.

## Software Requirements

(see separate document for installation of these)

- Apache Hadoop 2.7.1
- Python 2.7.x
- Nano text editor or other text editor

## Part A: Hadoop File System (HDFS)

1. Make sure you are running the Ubuntu VM, and start a fresh terminal window.
2. We need to clean up the Hadoop filesystem and re-format.  
First, make sure the hadoop fs is stopped:

```
stop-dfs.sh
```

3. Now let's empty the HDFS storage directory.

```
rm -rf /app/hadoop/tmp/dfs/
```

4. Now let's format the HDFS filesystem:  
`hadoop namenode -format`

You should see a lot of output ending something similar to this:

```
15/10/22 09:15:20 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
15/10/22 09:15:20 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
15/10/22 09:15:20 INFO util.GSet: Computing capacity for map NameNodeRetryCache
15/10/22 09:15:20 INFO util.GSet: VM type = 64-bit
15/10/22 09:15:20 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB
15/10/22 09:15:20 INFO util.GSet: capacity = 2^15 = 32768 entries
15/10/22 09:15:20 INFO namenode.FSImage: Allocated new BlockPoolId: BP-420615264-127.0.1.1-1445501720083
15/10/22 09:15:20 INFO common.Storage: Storage directory /usr/local/hadoop_store/hdfs/namenode has been successfully formatted.
15/10/22 09:15:20 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
15/10/22 09:15:20 INFO util.ExitUtil: Exiting with status 0
15/10/22 09:15:20 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at oxclo/127.0.1.1
*****/
```

It may prompt you

```
Re-format filesystem in Storage Directory
/app/hadoop/tmp/dfs/name ? (Y or N)
```

Choose y

5. Now let's start the Hadoop filesystem. Type:

```
start-dfs.sh
```

You should see output like:

```
15/10/22 09:18:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-oxclo.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-oxclo.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-oxclo.out
15/10/22 09:19:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

6. Now let's make a directory:

```
hadoop fs -mkdir -p /user/oxclo/wind
```

7. And check it worked:

```
hadoop fs -ls -R /
```

### You should see:

```
15/10/22 09:24:48 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
drwxr-xr-x - oxclo supergroup          0 2015-11-11 16:36 /user
drwxr-xr-x - oxclo supergroup          0 2015-11-11 16:36 /user/oxclo
drwxr-xr-x - oxclo supergroup          0 2015-11-11 16:36 /user/oxclo/wind
```

8. Now let's copy some datafiles from the local filesystem into the HDFS:

```
hadoop fs -put ~/datafiles/wind/2015/* /user/oxclo/wind
```

If you repeat the command to list the files on the hadoop filesystem:

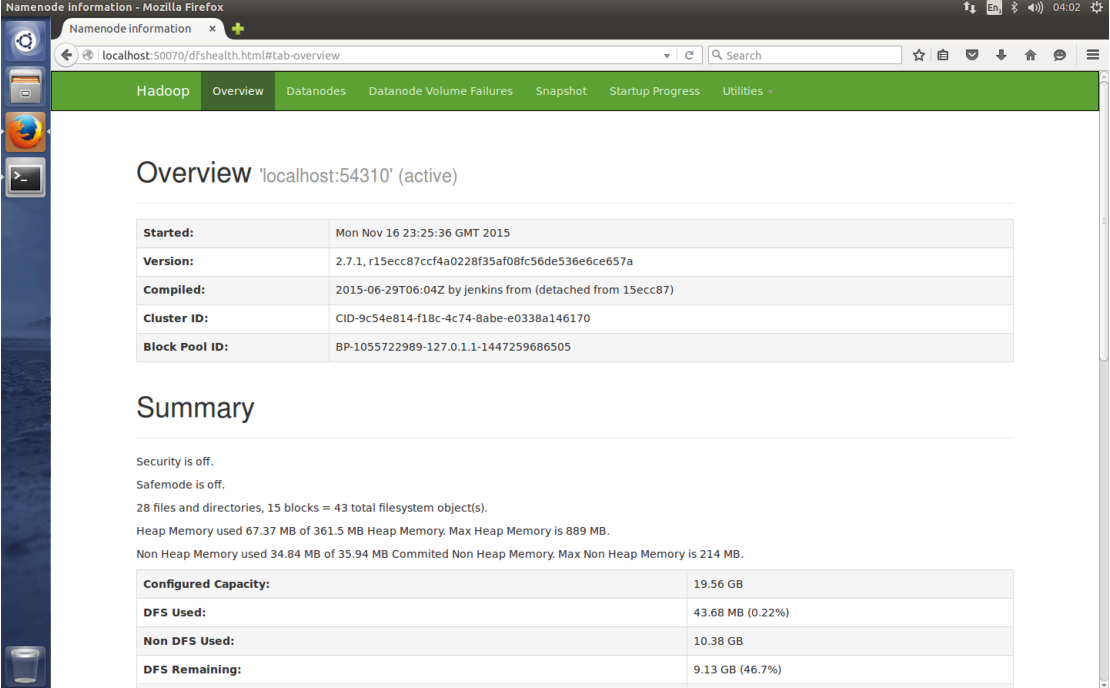
```
hadoop fs -ls -R /
```

you should see the new files in place:

```
15/10/22 09:45:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
drwxr-xr-x - oxclo supergroup 0 2015-11-11 16:36 /user
drwxr-xr-x - oxclo supergroup 0 2015-11-11 16:36 /user/oxclo
drwxr-xr-x - oxclo supergroup 0 2015-11-11 16:37 /user/oxclo/wind
-rw-r--r-- 1 oxclo supergroup 2716705 2015-11-11 16:37 /user/oxclo/wind/SF04.csv
-rw-r--r-- 1 oxclo supergroup 7896961 2015-11-11 16:37 /user/oxclo/wind/SF15.csv
-rw-r--r-- 1 oxclo supergroup 7250586 2015-11-11 16:37 /user/oxclo/wind/SF17.csv
-rw-r--r-- 1 oxclo supergroup 5538010 2015-11-11 16:37 /user/oxclo/wind/SF18.csv
-rw-r--r-- 1 oxclo supergroup 5818496 2015-11-11 16:37 /user/oxclo/wind/SF36.csv
-rw-r--r-- 1 oxclo supergroup 4952077 2015-11-11 16:37 /user/oxclo/wind/SF37.csv
```

9. You can browse the HDFS Web UI by going to <http://localhost:50070>

You will see something like:

1. 

**Overview 'localhost:54310' (active)**

<b>Started:</b>	Mon Nov 16 23:25:36 GMT 2015
<b>Version:</b>	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
<b>Compiled:</b>	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
<b>Cluster ID:</b>	CID-9c54e814-f18c-4c74-8abe-e0338a146170
<b>Block Pool ID:</b>	BP-1055722989-127.0.1.1-1447259686505

**Summary**

Security is off.  
Safemode is off.  
28 files and directories, 15 blocks = 43 total filesystem object(s).  
Heap Memory used 67.37 MB of 361.5 MB Heap Memory. Max Heap Memory is 889 MB.  
Non Heap Memory used 34.84 MB of 35.94 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

<b>Configured Capacity:</b>	19.56 GB
<b>DFS Used:</b>	43.68 MB (0.22%)
<b>Non DFS Used:</b>	10.38 GB
<b>DFS Remaining:</b>	9.13 GB (46.7%)

Take a look at the various tabs.

10. Congratulations you have successfully completed part A.

## Part B. Creating a mapper and reducer, and running map reduce

11. We have a lot of environmental and wind data from San Francisco, which we can analyse for various information.

12. First, let's look at the format of the input data.

Type

```
hadoop fs -cat /user/oxclo/wind/SF37.csv | head
```

You should see:

```
15/10/22 10:33:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
Station_ID,Station_Name,Location_Label,Interval_Minutes,Interval_End_Time,Wind_Velocity_
Mtr_Sec,Wind_Direction_Variance_Deg,Wind_Direction_Deg,Ambient_Temperature_Deg_C,Global_
Horizontal_Irradiance
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:15,3.59,0,263.1,8.61,0
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:20,3.342,0,268.3,8.58,0
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:25,3.175,0,271.3,8.45,0
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:30,2.586,0,279.7,8.45,0
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:35,3.137,0,273.8,8.44,0
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:40,2.781,0,277.9,8.54,0
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:45,3.177,0,269.9,8.66,0
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:50,2.802,0,277.5,8.72,0
SF37,"Chinatown Medical Center","Chinatown Medical Center",5,2015-01-5?
06:55,3.135,0,273.1,8.65,0
cat: Unable to write to output stream.
```

13. There is a header line and then a set of data, all in comma separated value (CSV) format.
14. Let's start by calculating maximum wind velocity for each district over the given period (the current year).
15. To do this, we can simply extract the Station\_ID and the Wind\_Velocity\_Mtr\_sec from each line using the mapper, and then we can summarize this information in the Reducer.
16. There is a skeleton mapper written for you at the URL:  
<http://freo.me/oxclo-mapper>
17. Create a directory for your code:  

```
mkdir ~/wind-analysis  
cd ~/wind-analysis
```
18. Edit this mapper to output the right key/value pairs.
  - a. You can use any Unix text editor. If you know how, then nano is a good terminal based editor.

I have installed Sublime Text (subl on the command line) which is a good Python editor.

You can use subl like this:  

```
subl wind-mapper.py
```

Some people like PyCharm and you can install that if you prefer

- b. The usual way of passing the key/value pairs from the mapper to the reducer is to use Tab delimited lines. A quick and easy way of printing those out in Python is to create an array of results and then use the join function to link them up with tabs:

```
result = [ k, v ]  
print ('\t'.join(result))
```

- c. Don't forget that Python is tab/space sensitive!
- d. You may need to cast velocity as a string using `str(velocity)`

19. Before we run this using Hadoop, we can simply test our script using the command-line.

- a. We first need to make the `mapper.py` executable:  
`chmod +x ./wind-mapper.py`
- b. We will use the local copy of one of the files
- c. Type:  
`cat ~/datafiles/wind/2015/SF37.csv | ./wind-mapper.py`
- d. You should see many data lines like this printed:

```
SF37 1.79
SF37 1.83
SF37 1.316
SF37 1.721
SF37 1.321
SF37 1.438
SF37 1.673
SF37 1.417
...
```

20. Now we need to create a reducer:

- a. The first task is to find the maximum wind speed recorded for each station.
- b. Once again start with the skeleton that is provided in:  
<http://freo.me/oxclo-reducer>
- c. Create a file locally and edit it to work. Call it `wind-reducer.py`  
Don't forget to  
`chmod +x ./wind-reducer.py`
- d. I recommend using a python dictionary  
(<https://docs.python.org/2/tutorial/datastructures.html#dictionaries>).  
  
to collect the maximum value by station id.
- e. Hint: you need to convert the number to a string before printing it  
with the join function, you can do this by using `str(max)`

21. Check that your code works locally before we try it with Hadoop  
`cat ~/datafiles/wind/2015/SF37.csv \`  
`| ./wind-mapper.py | ./wind-reducer.py`

22. You should see:  
SF37 7.079  
printed out.

23. Now we are ready to run this as a map-reduce job using Hadoop. Firstly, this will run different files on (potentially) different systems and processes. Secondly, it will access the data from HDFS instead of the local file system.

24. Before we can run the map-reduce job, we need to start up YARN, the job scheduler.

- a. Type:  
`start-yarn.sh`
- b. You should see:

```
starting yarn daemons
starting resourcemanager, logging to
/usr/local/hadoop/logs/yarn-hduser-resourcemanager-oxclo.out
localhost: starting nodemanager, logging to
/usr/local/hadoop/logs/yarn-hduser-nodemanager-oxclo.out
```

25. Now we can initiate the job. To run the python job, we utilize a generic capability of Hadoop to run code that uses standard input/output. Of course we've written those Python programs to do exactly that. This capability is called Hadoop Streaming

- a. Type (on one line)

```
yarn jar
/usr/local/hadoop/share/hadoop/tools/lib/hadoop-
streaming-2.7.3.jar -input /user/oxclo/wind/ -output
/user/oxclo/output -mapper ./wind-mapper.py -reducer
./wind-reducer.py
```



b. You should see a **lot** of log output, ending something *similar* to:

```
15/10/23 14:35:38 INFO mapred.Task: Task attempt_local1643623661_0001_r_000000_0 is
allowed to commit now
15/10/23 14:35:38 INFO output.FileOutputCommitter: Saved output of task
'attempt_local1643623661_0001_r_000000_0' to
hdfs://localhost:54310/usr/hduser/output/_temporary/0/task_local1643623661_0001_r_0000
00
15/10/23 14:35:38 INFO mapred.LocalJobRunner: Records R/w=235183/1 > reduce
15/10/23 14:35:38 INFO mapred.Task: Task 'attempt_local1643623661_0001_r_000000_0'
done.
15/10/23 14:35:38 INFO mapred.LocalJobRunner: Finishing task:
attempt_local1643623661_0001_r_000000_0
15/10/23 14:35:38 INFO mapred.LocalJobRunner: reduce task executor complete.
15/10/23 14:35:39 INFO mapreduce.Job: map 100% reduce 100%
15/10/23 14:35:39 INFO mapreduce.Job: Job job_local1643623661_0001 completed
successfully
15/10/23 14:35:39 INFO mapreduce.Job: Counters: 35
File System Counters
  FILE: Number of bytes read=6670124
  FILE: Number of bytes written=17106042
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=170316404
  HDFS: Number of bytes written=44
  HDFS: Number of read operations=78
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=9
Map-Reduce Framework
  Map input records=392695
  Map output records=235183
  Map output bytes=2485256
  Map output materialized bytes=2955658
  Input split bytes=600
  Combine input records=0
  Combine output records=0
  Reduce input groups=5
  Reduce shuffle bytes=2955658
  Reduce input records=235183
  Reduce output records=4
  Spilled Records=470366
  Shuffled Maps =6
  Failed Shuffles=0
  Merged Map outputs=6
  GC time elapsed (ms)=300
  Total committed heap usage (bytes)=3470262272
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=34172835
File Output Format Counters
  Bytes Written=44
15/10/23 14:35:39 INFO streaming.StreamJob: Output directory: /usr/hduser/output
```

c. Now you can check if there is any output:

```
hadoop fs -cat /user/oxclo/output/part-00000
```

d. You should see:

```
15/10/23 14:38:44 WARN util.NativeCodeLoader: Unable to load
native-hadoop library for your platform... using builtin-java
classes where applicable
SF04 34.12
SF36 11.05
SF15 7.92
SF17 5.767
SF18 10.57
SF37 7.079
```

26. If you do not see the same output, then check your code and re-run. If you re-run you will need to specify a new output directory (e.g. output-2).

27. Now modify your code to calculate the average wind velocity at each station.

Hint 1: copy your reducer.py code to reducer-average.py to start with.

Hint 2: you shouldn't need to change the mapper

Hint 3: you will need to specify a new output directory

Run your code using hadoop and compare your answers to these:

```
SF04 2.30098174812
SF36 2.46417253091
SF15 1.82141456775
SF17 0.518350025349
SF18 2.22022343917
SF37 2.2604035055
```

28. Congratulations! You have completed this exercise.

29. If you want to see a sample of code that calculates both the maxes and the averages, it is here:

<https://github.com/pzfreo/ox-clo/tree/master/code/wind-analysis/complete>

### 30. Extension:

Determine which direction (N,NE,E,SE,S,SW,W,NW) has had the strongest average winds this year across all stations.