Data Science

hw08 . Hadoop

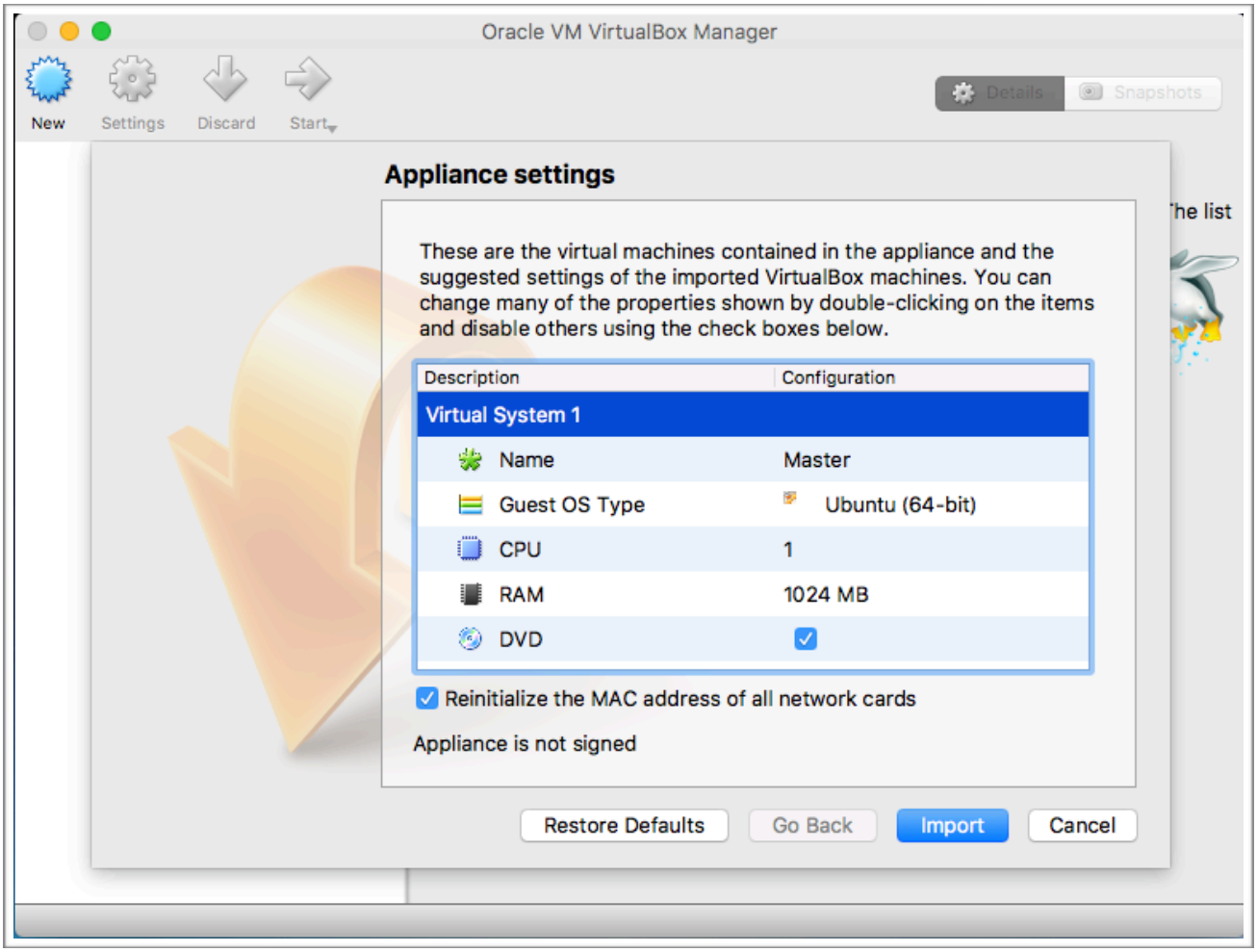
2017년 5월 17일

00 반
충남대학교 컴퓨터공학과
201202154
조윤재

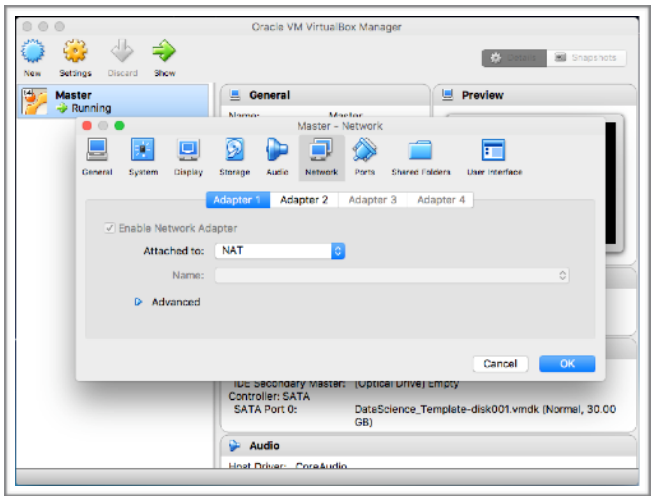
❖ Contents.

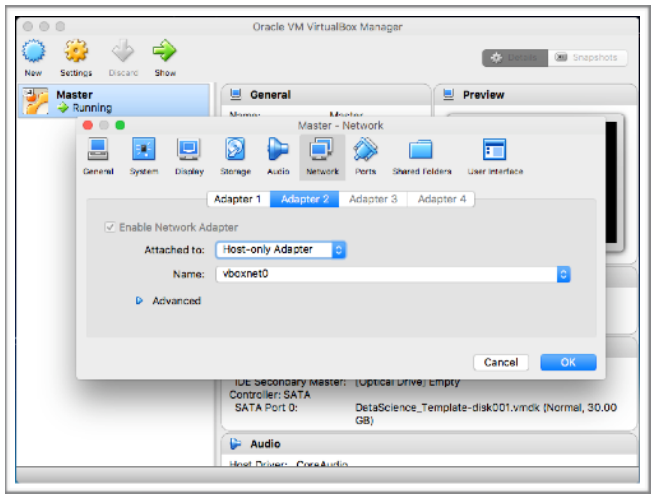
1. Hadoop Fully-Distribution Setting

1) 가장 먼저 Master로 사용될 가상머신을 RAM을 1GB로 할당하여 만들어 줍니다. MAC주소 초기화를 반드시 설정해주어야 합니다.

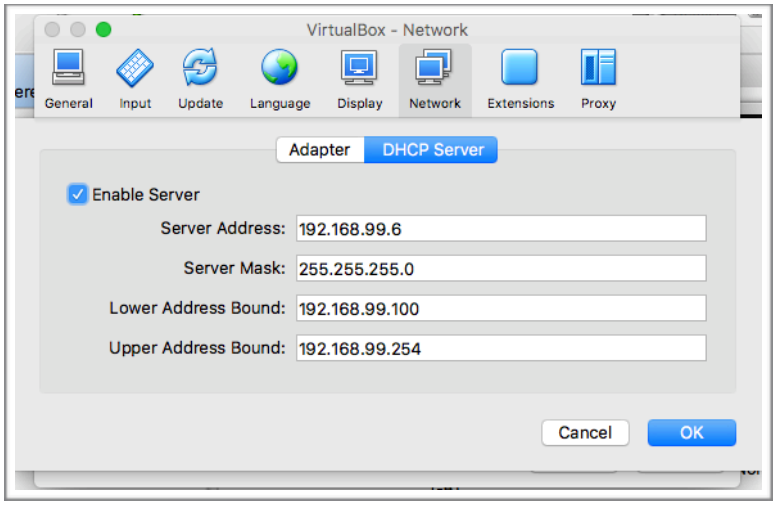


2) 만들어진 가상머신의 Network 환경설정을 해줍니다. 인터넷 연결을 위해 첫번째는 NAT, 두번째는 Host간 연결을 위해 Host-only Adapter로 설정합니다.





3) VirtualBox Preference의 Host-only Adapter에서 ip설정 범위를 확인 할 수 있습니다. Master는 이 범위에서 192.168.99.105 에 할당된 것을 확인 할 수 있습니다.




```
datascience@DataScience ~ $ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:fa:de:3c
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::c1e1:5fc5:8ec2:4a41/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:73 errors:0 dropped:0 overruns:0 frame:0
          TX packets:132 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17783 (17.7 KB)  TX bytes:12996 (12.9 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:dc:97:ec
          inet addr:192.168.99.105  Bcast:192.168.99.255  Mask:255.255.255.0
          inet6 addr: fe80::1cc0:7cfa:e801:6880/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:836 (836.0 B)  TX bytes:6609 (6.6 KB)
```

4) sudo vi /etc/hostname 명령어를 통해, hostname을 master로 바꿔 줍니다. 나중에 slave node들을 만들 때에도 마찬가지 방법으로 이름을 정해줍니다.

5) ssh와 rsync가 설치가 안되어있다면 sudo apt-get install ssh rsync 명령어를 통해 설치해줍니다.

```
datascience@DataScience ~ $ ssh
usage: ssh [-1246AaCfGgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
           [-D [bind_address:]port] [-E log_file] [-e escape_char]
           [-F configfile] [-I pkcs11] [-i identity_file] [-L address]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] [user@]hostname [command]
datascience@DataScience ~ $ rsync
rsync  version 3.1.1  protocol version 31
Copyright (C) 1996-2014 by Andrew Tridgell, Wayne Davison, and others.
Web site: http://rsync.samba.org/
Capabilities:
    64-bit files, 64-bit inums, 64-bit timestamps, 64-bit long ints,
    socketpairs, hardlinks, symlinks, IPv6, batchfiles, inplace,
    append, ACLs, xattrs, iconv, symtimes, prealloc
```

6) Node들간 ssh 통신을 위해 인증키 설정을 해줍니다.

```
datascience@DataScience ~ $ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/home/datascience/.ssh'.
Your identification has been saved in /home/datascience/.ssh/id_rsa.
Your public key has been saved in /home/datascience/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:6bYZSl7ZljD98X83byaClnj/Vv6iObK3PHzkY0hmAGM datascience@DataScience
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|        E        |
|       . o       |
|         o.      |
|        S ...    |
|       . = o+o. .|
|      . *.+B.+.o |
|     o +.=*.B.Oo*|
|      o oo.=*@.*X|
+----[SHA256]-----+
```

```
datascience@DataScience ~ $ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
datascience@DataScience ~ $ chmod 0600 ~/.ssh/authorized_keys
datascience@DataScience ~ $
```

ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys

7) sudo vi /etc/hosts 명령어를 통해, 각 ip 에 대해 Node의 이름을 적어 줍니다. master가 101이기 때문에 새로운 Node를 생성 할 때 Mac주소를 초기화 시켜 준다면 1씩 증가하게 됩니다.

```
127.0.0.1   localhost
192.168.99.105 master
192.168.99.106 slave1
192.168.99.107 slave2
192.168.99.108 slave3

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

</etc/hosts>

8) 적절한 JDK 버전을 설치해줍니다.

```
datascience@DataScience ~ $ java -version
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
```

sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java7-installer

9) Hadoop을 설치해줍니다.

```
datascience@DataScience ~ $ ls
anaconda3  Documents  hadoop-2.7.3         Music     Public  Templates
Desktop    Downloads  hadoop-2.7.3.tar.gz  Pictures  R       Videos
```

sudo wget http://apache.mirror.cdnetworks.com/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz
tar -xvf hadoop-2.7.3.tar.gz

10) PATH 설정을 해주고 정상적으로 설치가 되었는지 확인합니다.

```
128 export JAVA_HOME=/usr/lib/jvm/java-7-oracle
129 export HADOOP_HOME=/home/datascience/hadoop-2.7.3
130 export HADOOP_CONFIG_HOME=$HADOOP_HOME/etc/hadoop
131 export PATH=$PATH:$HADOOP_HOME/bin
132 export PATH=$PATH:$HADOOP_HOME/sbin
```

```
datascience@DataScience ~ $ vi .bashrc
datascience@DataScience ~ $ source .bashrc
datascience@DataScience ~ $ hadoop
Usage: hadoop [--config confdir] [COMMAND | CLASSNAME]
  CLASSNAME            run the class named CLASSNAME
 or
  where COMMAND is one of:
  fs                   run a generic filesystem user client
  version              print the version
  jar <jar>            run a jar file
                       note: please use "yarn jar" to launch
                             YARN applications, not this command.
  checknative [-a|-h]  check native hadoop and compression libraries availabilit
y
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archi
ve
```

11) ${HADOOP_HOME}/etc/hadoop/core-site.xml 을 수정해서, master로 사용되는 node를 설정합니다.

```
 19 <configuration>
 20     <property>
 21         <name>fs.defaultFS</name>
 22         <value>hdfs://master</value>
 23     </property>
 24 </configuration>
~
~
~
"core-site.xml" 24L, 876C                                 1,1           All
```

12) ${HADOOP_HOME}/etc/hadoop/mapred-site.xml.template 을 복사하여 mapred-site.xml를 만들고 수정해줍니다.

```
 19 <configuration>
 20     <property>
 21         <name>mapreduce.framework.name</name>
 22         <value>yarn</value>
 23     </property>
 24 </configuration>
~
~
~
"mapred-site.xml" 24L, 862C                               1,1           All
```

13) ${HADOOP_HOME}/etc/hadoop/hdfs-site.xml 을 수정하여 datanode, namenode에 대해 설정해줍니다.

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>4</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/home/datascience/hadoop-2.7.3/dfs/name</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/home/datascience/hadoop-2.7.3/dfs/data</value>
    </property>
    <property>
        <name>dfs.hosts</name>
        <value>/home/datascience/hadoop-2.7.3/include</value>
    </property>
</configuration>
~
~
~
~
"hdfs-site.xml" 36L, 1265C                                    1,1           All
```

14) ${HADOOP_HOME}/etc/hadoop/yarn-site.xml 을 수정하여 ResourceManager를 slave1로 설정해줍니다.

```
<configuration>
<!-- Site specific YARN configuration properties -->
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>slave1</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
~
"yarn-site.xml" 26L, 924C                                     1,1           All
```

15) hdfs와 yarn을 시작하는데 있어서 Java의 dependency가 있는데, 이 PATH를 따로 설정해주어야 합니다. 따라서 ${HADOOP_HOME}/etc/hadoop/hadoop-env.sh 와 yarn-env.sh에서 JAVA_HOME을 다음과 같이 다시 설정해줍니다.

JAVA_HOME=/usr/lib/jvm/java-7-oracle

16) ${HADOOP_HOME}/etc/hadoop/slaves 를 수정하여 모든 Node의 이름을 써줍니다.

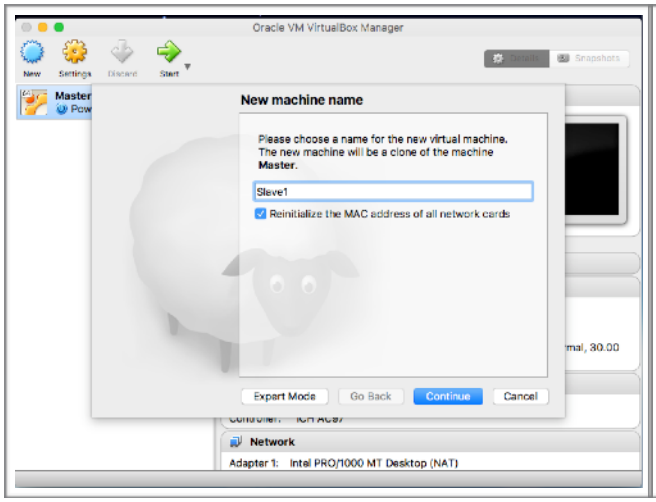
master
slave1
slave2
slave3

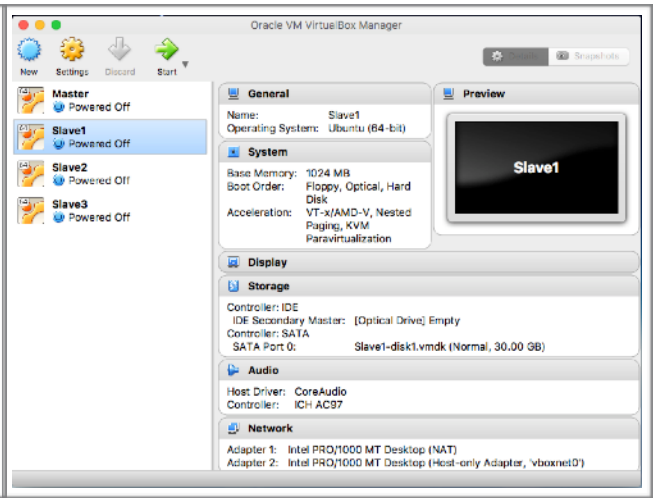
17) ${HADOOP_HOME} 에서 datanode와 namenode를 위해 다음과 같이 빈 directory를 생성 해줍니다.

```
datascience@DataScience ~/hadoop-2.7.3 $ mkdir -p dfs/data
datascience@DataScience ~/hadoop-2.7.3 $ mkdir -p dfs/name
datascience@DataScience ~/hadoop-2.7.3 $ ls dfs/
data  name
```

mkdir -p dfs / data
mkdir -p dfs / name

18) 이제 Hadoop에 대한 모든 설정이 끝났으니, Master Node를 복사하여 Slave를 생성해주고, host name을 수정해 줍니다. 복사(Clone)을 할 때는 반드시 Mac주소를 초기화 시켜주고, Full clone으로 설정합니다.





19) Master에서 slave로 ssh가 되는 것을 확인합니다.

```
datascience@master ~ $ ssh slave1
Welcome to Linux Mint 18.1 Serena (GNU/Linux 4.4.0-53-generic x86_64)

 * Documentation:  https://www.linuxmint.com
Last login: Fri May 12 18:10:14 2017 from 192.168.99.105
```

만약 설정을 변경하면 rsync 명령어를 통해 Node가 파일을 업데이트 합니다.

rsync -avz ~/hadoop-2.7.3 datascience@slave1:/home/datascience/

20) Hadoop을 차례로 실행합니다. 먼저 Master에서 namenode format을 설정하고 start-dfs.sh를 실행합니다.

At $HADOOP_HOME..

bin/hdfs namenode -format
sbin/start-dfs.sh

21) slave1로 접속하여 start-yarn.sh를 실행합니다. yes 명령어를 물어볼때 마다 입력해줍니다.

```
datascience@master ~/hadoop-2.7.3 $ ssh slave1
Welcome to Linux Mint 18.1 Serena (GNU/Linux 4.4.0-53-generic x86_64)

 * Documentation:  https://www.linuxmint.com
Last login: Fri May 12 18:15:33 2017 from 192.168.99.105
datascience@slave1 ~ $ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/datascience/hadoop-2.7.3/logs/yarn-da
tascience-resourcemanager-slave1.out
The authenticity of host 'slave2 (192.168.99.107)' can't be established.
ECDSA key fingerprint is SHA256:6c9yl6xST5hvI48vEKKCvTd7XSuoGC6ANMgswflXHYI.
Are you sure you want to continue connecting (yes/no)? The authenticity of host
'slave1 (192.168.99.106)' can't be established.
```

22) 각 Node들에서 jps명령어를 실행하여 적절하게 process가 실행됬는지 확인합니다.

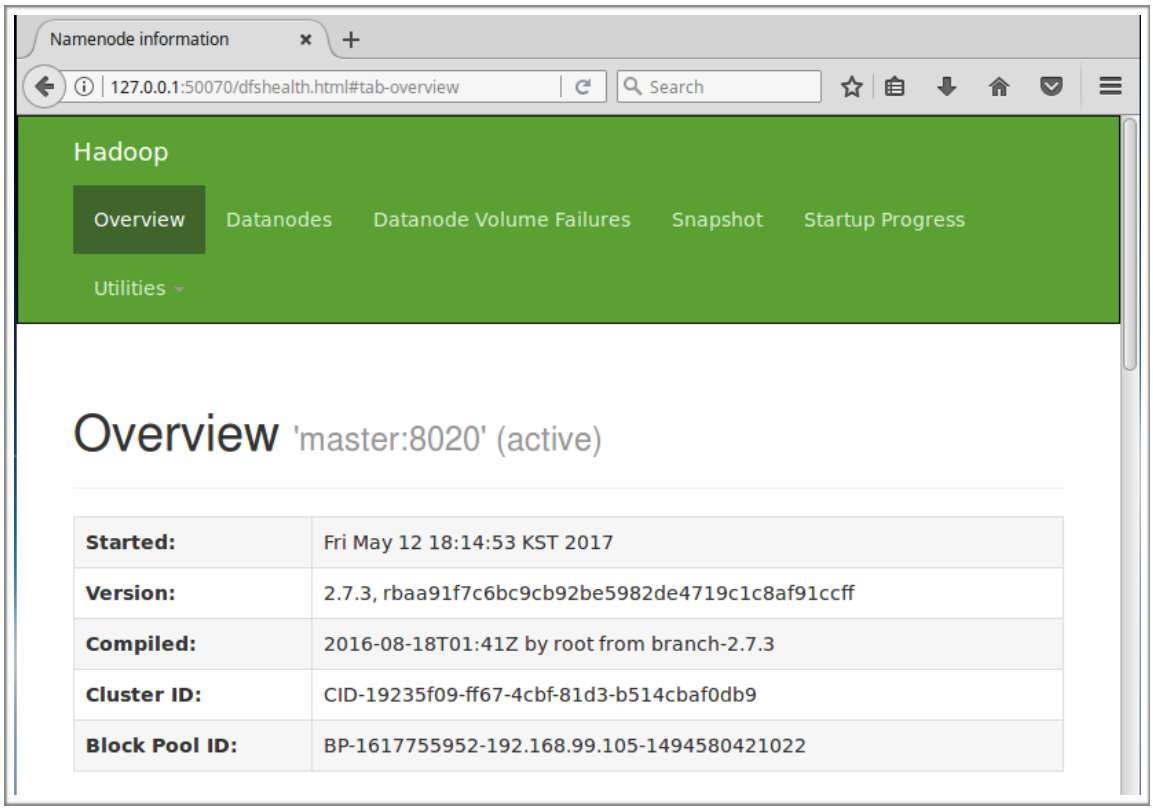
```
datascience@master ~ $ jps
2777 NodeManager
2907 Jps
2315 NameNode
2425 DataNode
2618 SecondaryNameNode
```

```
datascience@slave1 ~ $ jps
2300 ResourceManager
2594 NodeManager
2711 Jps
2121 DataNode
```

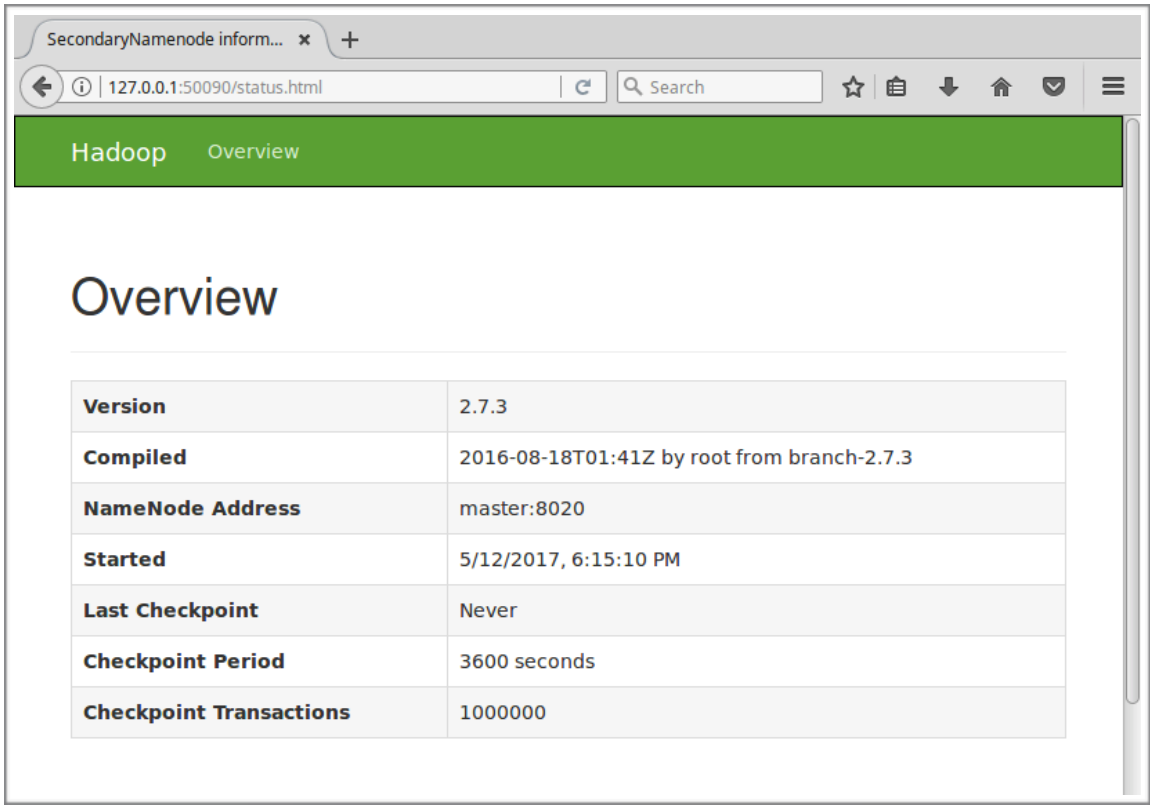
```
datascience@slave2 ~ $ jps
2287 Jps
2150 NodeManager
2039 DataNode
```

```
datascience@slave3 ~ $ jps
2101 DataNode
2349 Jps
2212 NodeManager
```

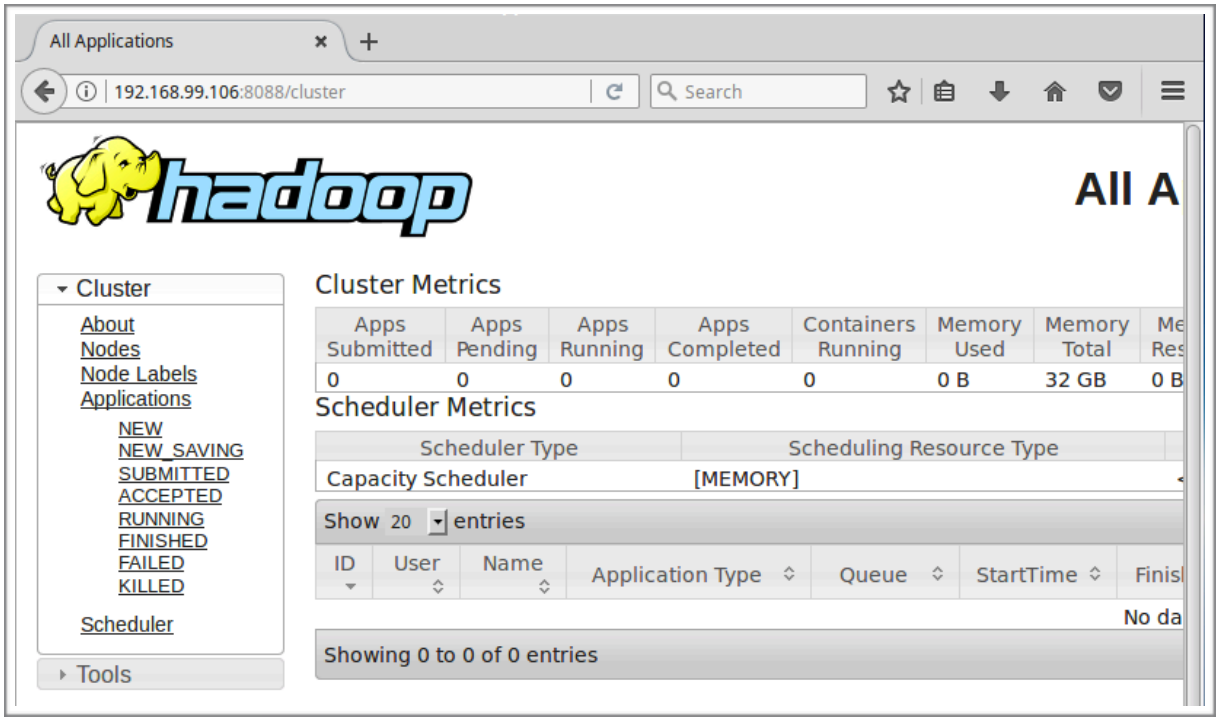
Namenode, SecondaryNameNode 1대, ResourceManager 1대, Datanode, NodeManager가 4대씩 있는 것을 확인 할 수 있습니다.



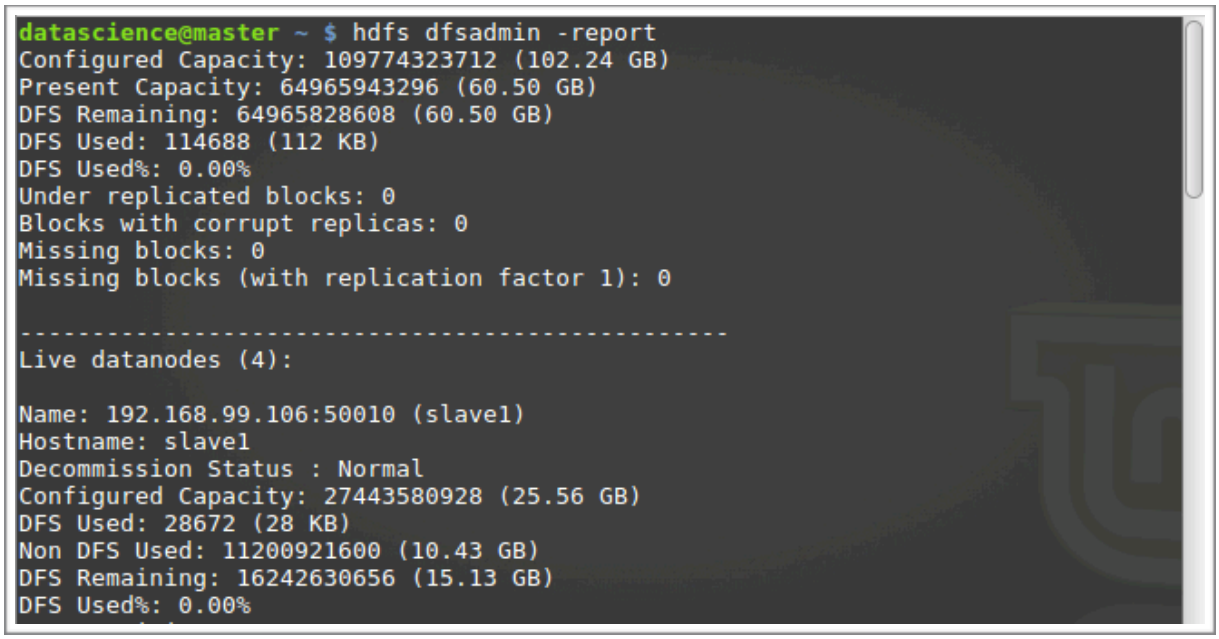
127.0.0.1:50070 Namenode (Master)



127.0.0.1:50090 SecondayNameNode (Master)



192.168.99.106:8088 ResourceManager (Slave1)



hfs dfsadmin -report (Master)

```
datascience@slave1 ~ $ yarn node -list
17/05/12 18:27:14 INFO client.RMProxy: Connecting to ResourceManager at slave1/192.168.99.106:8032
Total Nodes:4
         Node-Id             Node-State Node-Http-Address       Number-of-Running-Containers
    slave2:36232                RUNNING       slave2:8042                                  0
    master:33510                RUNNING       master:8042                                  0
    slave3:40784                RUNNING       slave3:8042                                  0
    slave1:42653                RUNNING       slave1:8042                                  0
```

yarn node -list (Slave1)

2. Word Count

다음으로는 Hadoop을 사용하여 Word Count 예제를 실행하는 실습입니다.

1) Word Count를 실행하기 위해서는 MapReduce 코드가 컴파일된 jar파일이 필요한데, 이것에 대해 Hadoop이 기본적으로 제공하는 hadoop-mapreduce-examples.jar 파일이 존재합니다. 따라서 다음 경로로 이동합니다.

/home/datascience/hadoop-2.7.3/share/hadoop/mapreduce

```
datascience@master ~/hadoop-2.7.3/share/hadoop/mapreduce $ ls
hadoop-mapreduce-client-app-2.7.3.jar
hadoop-mapreduce-client-common-2.7.3.jar
hadoop-mapreduce-client-core-2.7.3.jar
hadoop-mapreduce-client-hs-2.7.3.jar
hadoop-mapreduce-client-hs-plugins-2.7.3.jar
hadoop-mapreduce-client-jobclient-2.7.3.jar
hadoop-mapreduce-client-jobclient-2.7.3-tests.jar
hadoop-mapreduce-client-shuffle-2.7.3.jar
hadoop-mapreduce-examples-2.7.3.jar
lib
lib-examples
sources
```

2) hadoop jar hadoop-mapreduce-examples.jar wordcount 명령어를 사용하여 확인해보니, word count를 위해서는 입력파일과 출력파일을 포함시켜야 한다고 나옵니다.

```
datascience@master ~/hadoop-2.7.3/share/hadoop/mapreduce $ hadoop jar hadoop-map
reduce-examples-2.7.3.jar wordcount
Usage: wordcount <in> [<in>...] <out>
```

이때 입력 파일은 hdfs , 즉 하둡 파일시스템에 올라가 있어야 합니다.

3) 입력파일은 $HADOOP_HOME의 LICENSE, NOTICE, READEME 이렇게 3개의 txt파일을 선택하였고, 하둡 시스템에 /input 디렉토리를 만들고 세개의 txt파일을 put명령어를 통해 넣어 주었습니다. 그리고 ls 명령어를 통해 3개의 파일이 올라간 것을 확인하였습니다.

```
datascience@master ~/hadoop-2.7.3 $ ls
bin  etc      lib      LICENSE.txt  NOTICE.txt  sbin
dfs  include  libexec  logs         README.txt  share
datascience@master ~/hadoop-2.7.3 $ hdfs dfs -mkdir /input
datascience@master ~/hadoop-2.7.3 $ hdfs dfs -put LICENSE.txt /input
datascience@master ~/hadoop-2.7.3 $ hdfs dfs -put NOTICE.txt /input
datascience@master ~/hadoop-2.7.3 $ hdfs dfs -put README.txt /input
```

```
datascience@master ~/hadoop-2.7.3 $ hdfs dfs -ls /input
Found 3 items
-rw-r--r--   4 datascience supergroup      84854 2017-05-12 23:31 /input/LICENSE.txt
-rw-r--r--   4 datascience supergroup      14978 2017-05-12 23:32 /input/NOTICE.txt
-rw-r--r--   4 datascience supergroup       1366 2017-05-12 23:32 /input/README.txt
```

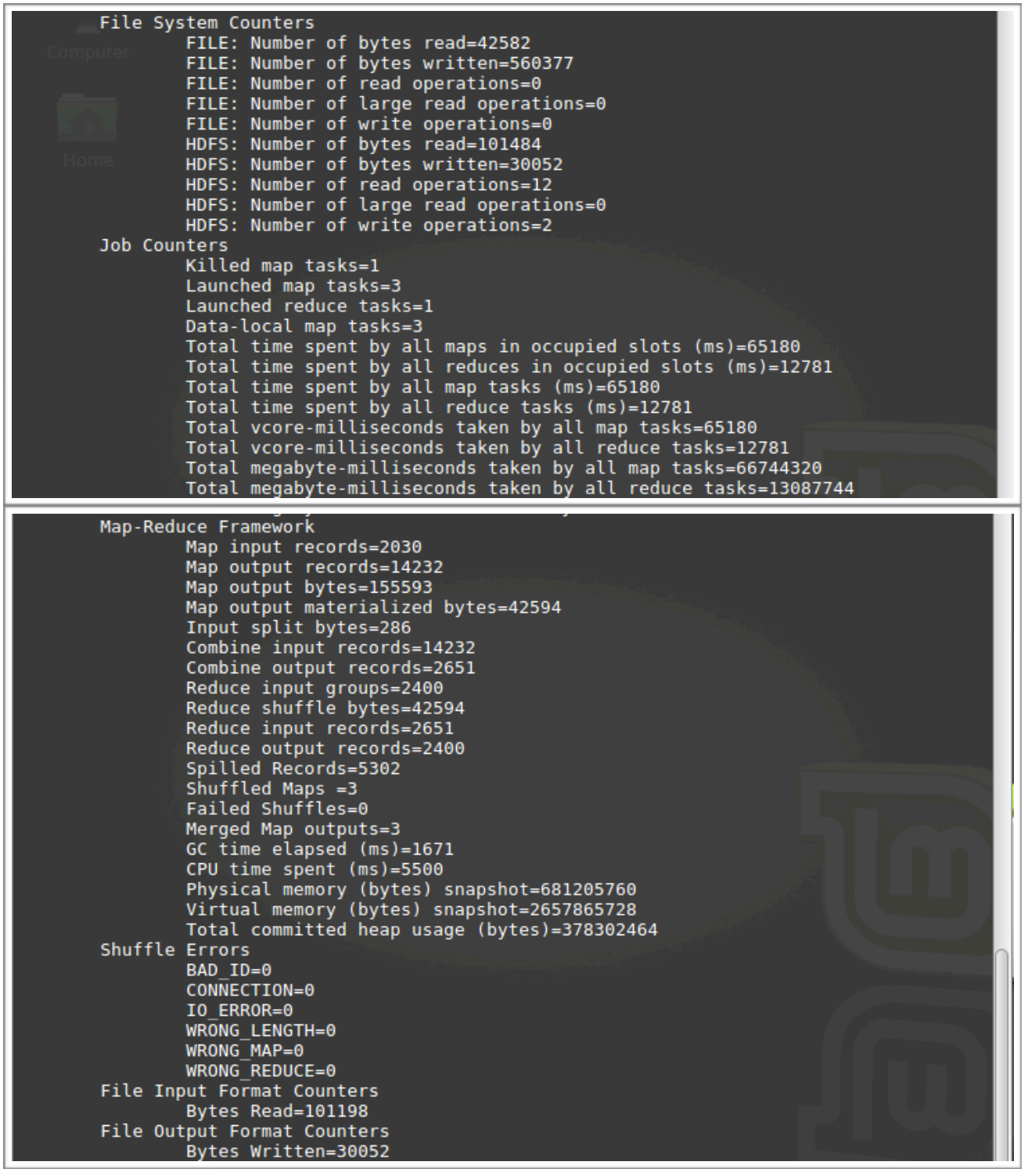
hdfs dfs -mkdir /input
hdfs dfs -put LICENSE.txt /input
hdfs dfs -put NOTICE.txt /input
hdfs dfs -put README.txt /input

hdfs dfs -ls /input

4) hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar wordcount /input /output 명령어를 통해 wordcount를 수행합니다. 입력파일로는 /input 디렉토리를 출력파일로는 /output 디렉토리로 설정하였습니다.

```
datascience@master ~/hadoop-2.7.3 $ hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-exam
ples-2.7.3.jar wordcount /input /output
17/05/12 23:38:26 INFO client.RMProxy: Connecting to ResourceManager at slave1/192.168.99.1
06:8032
17/05/12 23:38:28 INFO input.FileInputFormat: Total input paths to process : 3
17/05/12 23:38:28 INFO mapreduce.JobSubmitter: number of splits:3
17/05/12 23:38:28 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1494598606437
_0001
17/05/12 23:38:29 INFO impl.YarnClientImpl: Submitted application application_1494598606437
_0001
17/05/12 23:38:29 INFO mapreduce.Job: The url to track the job: http://slave1:8088/proxy/ap
plication_1494598606437_0001/
17/05/12 23:38:29 INFO mapreduce.Job: Running job: job_1494598606437_0001
17/05/12 23:38:42 INFO mapreduce.Job: Job job_1494598606437_0001 running in uber mode : fal
se
17/05/12 23:38:42 INFO mapreduce.Job:  map 0% reduce 0%
17/05/12 23:39:07 INFO mapreduce.Job:  map 67% reduce 0%
17/05/12 23:39:09 INFO mapreduce.Job:  map 100% reduce 0%
17/05/12 23:39:22 INFO mapreduce.Job:  map 100% reduce 100%
17/05/12 23:39:22 INFO mapreduce.Job: Job job_1494598606437_0001 completed successfully
17/05/12 23:39:22 INFO mapreduce.Job: Counters: 50
```

slave1의 ResourceManager가 연결되어 File을 split하고 map reduce가 진행 되는 흐름을 볼 수 있습니다.



MapReduce의 결과가 에러없이 잘 마무리된 모습입니다.

5) 결과 값이 저장된 /output 디렉토리를 살펴보니 성공했다는 파일의 표시와 결과가 나타나 있고, part-r-00000 파일은 cat 명령어를 통해 확인해보았습니다.

```
datascience@master ~/hadoop-2.7.3 $ hdfs dfs -ls /output
Found 2 items
-rw-r--r--   4 datascience supergroup          0 2017-05-12 23:39 /output/_SUCCESS
-rw-r--r--   4 datascience supergroup      30052 2017-05-12 23:39 /output/part-r-00000
```

```
datascience@master ~/hadoop-2.7.3 $ hdfs dfs -cat /output/part-r-00000
""AS    2
"AS     17
"COPYRIGHTS     1
"Contribution"  2
"Contributor"   2
"Derivative     1
"GCC    1
"Legal  1
"License"       1
"License");     2
"Licensed       1
"Licensor"      1
"Losses")       1
"NOTICE"        1
"Not    1
"Object"        1
"Program"       1
"Recipient"     1
"Software"),    5
"Source"        1
"Work"  1
"You"   1
"Your") 1
"[]"    1
"control"       1
```

hdfs dfs -ls /output
hdfs dfs -cat /output/part-r-00000

6) /output/part-r-00000 파일을 get 명령어를 통해 하둡파일시스템에서 linux 파일시스템에 출력하도록 하였습니다.

```
datascience@master ~/hadoop-2.7.3 $ hdfs dfs -get /output/part-r-00000 ./wordcount.txt
datascience@master ~/hadoop-2.7.3 $ cat wordcount.txt
""AS    2
"AS     17
"COPYRIGHTS     1
"Contribution"  2
"Contributor"   2
"Derivative     1
"GCC    1
"Legal  1
"License"       1
"License");     2
"Licensed       1
"Licensor"      1
"Losses")       1
"NOTICE"        1
"Not    1
"Object"        1
```

hdfs dfs -get /output/part-r-00000 ./wordcount.txt

7) ResourceManagerIP:8088 로 실행되었던 Job을 확인해 보았습니다.