분산 시스템 Hadoop 환경 구축

1. Hadoop fully Distriduted 설치 및 setup

```
hadoop01@hadoop01-VirtualBox: ~/hadoop-2.7.3
hadoop01@hadoop01-VirtualBox:~$ cd hadoop-2.7.3/
hadoop01@hadoop01-VirtualBox:~/hadoop-2.7.3$ bin/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
  classpath            prints the class path needed to get the
  credential           interact with credential providers
                       Hadoop jar and the required libraries
  daemonlog            get/set the log level for each daemon
  trace                view and modify Hadoop tracing settings

Most commands print help when invoked w/o parameters.
hadoop01@hadoop01-VirtualBox:~/hadoop-2.7.3$
```

=> hadoop-2.7.3 설치 후 실행시 아무 이상 없이 실행이 되는 것을 확인할 수 있다.

```
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol.  Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}

export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}
                                                          33,1          13%
```

=> hadoop_env.sh 의 JAVA_HOME 을 설치한 java8 의 위치에 맞게 변경해준다.

```
127.0.0.1       localhost
127.0.1.1       hadoop01-VirtualBox
192.168.56.101  hadoop00
192.168.56.104  hadoop02
192.168.56.102  hadoop03
192.168.56.103  hadoop04
# 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
~
~
~
~
~
~
~
~
~
~
~
"hosts" [readonly] 12L, 329C                                  1,1           All
```

=> 복제 시켰던 머신들의 호스트 전용 어뎁터의 네트워크 주소를 확인 시켜준후 각각의 머신들의 etc/ hosts 에 머신들의 네트워크를 저장시켜준다. 그리고 각 머신들의 hostname 을 hadoop00, hadoop02, hadoop03, hadoop04 로 바꾼후 저장시킨다.

```
hadoop01@hadoop00:~/hadoop-2.7.3/dfs$ ls
data  name
```

=> datanode 와 namenode 의 데이터를 저장하기 위해서 hadoop 디렉토리에 dfs 디렉토리를 만들고 난후에 data 와 name 디렉토리를 만들어 준다.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://hadoop00</value>
        </property>
</configuration>
~
~
~
"core-site.xml" 24L, 860C                                     1,1           All
```

=> core-site.xml 에 fs.defaultFS 를 namenode 로 설정한 hadoop00 을 hdfs://hadoop00 으로 설정해 줌으로써 namenode 를 지정해준다.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
~
~
~
"mapred-site.xml" 24L, 844C                                   1,1           All
```

=> mapred-site.xml 에 mapreduce.framework.name 을 hadoop2 에서는 yarn 을 사용하기에 value 를 yarn 으로 지정해준다.

```
  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>dfs.replication</name>
                <value>4</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>/home/hadoop01/hadoop-2.7.3/dfs/data</value>
        </property>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>/home/hadoop01/hadoop-2.7.3/dfs/name</value>
        </property>
        <property>
                <name>dfs.hosts</name>
                <value>/home/hadoop01/hadoop-2.7.3/include</value>
        </property>
</configuration>
                                                                36,1          Bot
```

=> hdfs-site.xml 안에 dfs.replication 은 datanode 의 갯수이기에 4 를 값으로 준다.
그리고 dfs.datanode.data.dir datanode 의 data 를 저장해주기위해 전에 만든 data 디렉토리의 위치를 값으로 주어주고 dfs.namenode.name.dir 은 namenode 의 data 를 저장해주는 것이기에 전에 만든 name 디렉토리의 위치를 값으로 준다.
dfs.hosts 는 hadoop 디렉토리에 존재하는 include 디렉토리의 위치를 값으로 지정해준다.
hdfs-site.xml 을 저장시켜준다.

```
hadoop01@hadoop03: ~/hadoop-2.7.3/etc/hadoop
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# User for YARN daemons
export HADOOP_YARN_USER=${HADOOP_YARN_USER:-yarn}

# resolve links - $0 may be a softlink
export YARN_CONF_DIR="${YARN_CONF_DIR:-$HADOOP_YARN_HOME/conf}"

# some Java parameters
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
if [ "$JAVA_HOME" != "" ]; then
  #echo "run java in $JAVA_HOME"
  JAVA_HOME=/usr/lib/jvm/java-8-oracle
fi

if [ "$JAVA_HOME" = "" ]; then
  echo "Error: JAVA_HOME is not set."
                                                                23,43         7%
                                                                      Left ⌘
```

=> yarn-env.sh 에 존재하는 JAVA_HOME 의 환경변수를 java8 의 위치에 맞게 저장시켜준다.

```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>

<!-- Site specific YARN configuration properties -->
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>hadoop03</value>
        </property>
</configuration>
~
"yarn-site.xml" 26L, 890C                                     1,1           All
```

=> yarn-site.xml 에 yarn.nodemanager.aux-services 를 mapreduce_shuffle 로 설정해준다.
그리고 yarn.resourcemanager.hostname 을 resourcemanager 를 관리해줄 hadoop03 으로 값을 주고 yarn-site.xml 을 저장시킨다.

```
hadoop01@hadoop00: ~/hadoop-2.7.3/etc/hadoop
hadoop00
hadoop02
hadoop03
hadoop04
~
~
~
~
~
~
```

=> etc/hadoop 에 존재하는 slaves 파일에 hadoop00, hadoop02, hadoop03, hadoop04 를 모두 입력해준다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ rsync -avz /home/hadoop01/hadoop-2.7.3 hadoop0
1@hadoop02:/home/hadoop01
hadoop01@hadoop00:~/hadoop-2.7.3$ rsync -avz /home/hadoop01/hadoop-2.7.3 hadoop0
1@hadoop03:/home/hadoop01
hadoop01@hadoop00:~/hadoop-2.7.3$ rsync -avz /home/hadoop01/hadoop-2.7.3 hadoop0
1@hadoop04:/home/hadoop01
```

=> hadoop 에 대한것을 hadoop00 에서 설정을 해준 다음 hadoop 디렉토리를 hadoop02, hadoop03, hadoop04 에 rsync 를 이용해서 변경된 부분을 전부 copy 시켜준다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs namenode -format
17/05/13 20:02:36 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = hadoop00/192.168.56.101
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.3
STARTUP_MSG:   classpath = /home/hadoop01/hadoop-2.7.3/etc/hadoop:/home/hadoop01
/hadoop-2.7.3/share/hadoop/common/lib/zookeeper-3.4.6.jar:/home/hadoop01/hadoop-
2.7.3/share/hadoop/common/lib/jettison-1.1.jar:/home/hadoop01/hadoop-2.7.3/share
/hadoop/common/lib/httpclient-4.2.5.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop
/common/lib/jaxb-api-2.2.2.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop/common/l
ib/java-xmlbuilder-0.4.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop/common/lib/j
sr305-3.0.0.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop/common/lib/junit-4.11.j
ar:/home/hadoop01/hadoop-2.7.3/share/hadoop/common/lib/snappy-java-1.0.4.1.jar:/
home/hadoop01/hadoop-2.7.3/share/hadoop/common/lib/xmlenc-0.52.jar:/home/hadoop0
1/hadoop-2.7.3/share/hadoop/common/lib/jets3t-0.9.0.jar:/home/hadoop01/hadoop-2.
7.3/share/hadoop/common/lib/gson-2.2.4.jar:/home/hadoop01/hadoop-2.7.3/share/had
oop/common/lib/protobuf-java-2.5.0.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop/
common/lib/commons-cli-1.2.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop/common/l
ib/hamcrest-core-1.3.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop/common/lib/cur
ator-client-2.7.1.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop/common/lib/common
```

=> hadoop 을 실행시키기 전에 bin/hdfs namenode -format 을 실행 시켜준다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ sbin/start-dfs.sh
Starting namenodes on [hadoop00]
hadoop00: starting namenode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-namenode-hadoop03.out
hadoop04: starting datanode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-datanode-hadoop00.out
hadoop00: starting datanode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-datanode-hadoop03.out
hadoop03: starting datanode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-datanode-hadoop04.out
hadoop02: datanode running as process 7857. Stop it first.
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/hadoop01/hadoop-2.7.3/logs
/hadoop-hadoop01-secondarynamenode-hadoop00.out
```

=> sbin/start-dfs.sh 를 실행 시켜서 namenode 와 secondary namenode 그리고 datanode 를 실행 시켜준다.

```
hadoop01@hadoop03:~/hadoop-2.7.3$ sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn-hadoo
p01-resourcemanager-hadoop03.out
hadoop04: nodemanager running as process 4732. Stop it first.
hadoop00: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop00.out
hadoop02: nodemanager running as process 6767. Stop it first.
hadoop03: nodemanager running as process 5988. Stop it first.
```

=> sbin/start-yarn.sh 를 실행 시켜 resourcemanager 와 nodemanager 를 실행시켜준다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ jps
4130 NodeManager
4308 Jps
3750 DataNode
3944 SecondaryNameNode
3582 NameNode
```

=> hadoop00 의 jps

```
hadoop01@hadoop02:~$ jps
7508 Jps
7318 DataNode
6767 NodeManager
```

=> hadoop02 의 jps

```
hadoop01@hadoop03:~/hadoop-2.7.3$ jps
7378 Jps
5988 NodeManager
6935 DataNode
7065 ResourceManager
```

=> hadoop03 의 jps

```
hadoop01@hadoop04:~$ jps
5289 DataNode
4732 NodeManager
5485 Jps
```

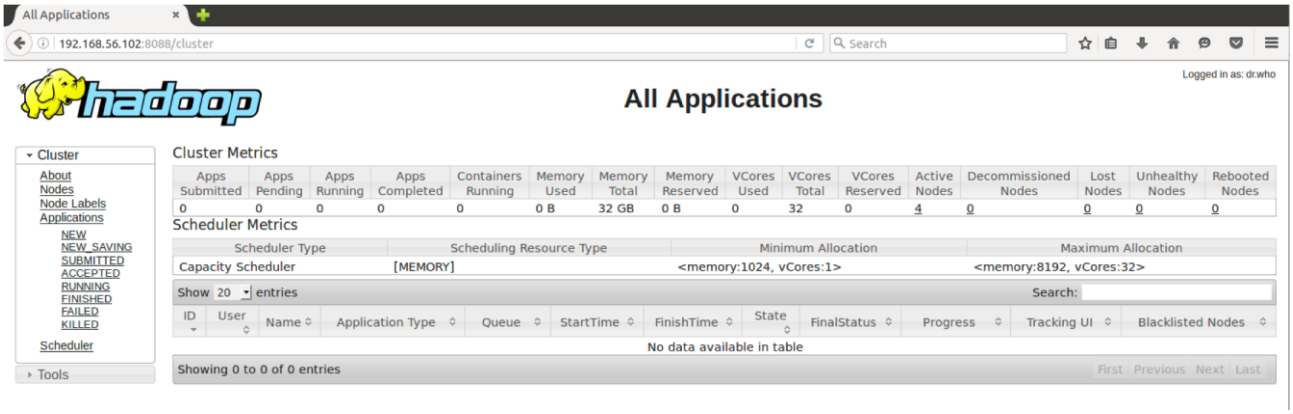
=> hadoop04 의 jps

```
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs dfsadmin -report
Configured Capacity: 81369726976 (75.78 GB)
Present Capacity: 55781777408 (51.95 GB)
DFS Remaining: 55781679104 (51.95 GB)
DFS Used: 98304 (96 KB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-------------------------------------------------
Live datanodes (4):

Name: 192.168.56.101:50010 (hadoop00)
Hostname: hadoop00
Decommission Status : Normal
Configured Capacity: 20342431744 (18.95 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 6353633280 (5.92 GB)
DFS Remaining: 13988773888 (13.03 GB)
DFS Used%: 0.00%
DFS Remaining%: 68.77%
Configured Cache Capacity: 0 (0 B)
```

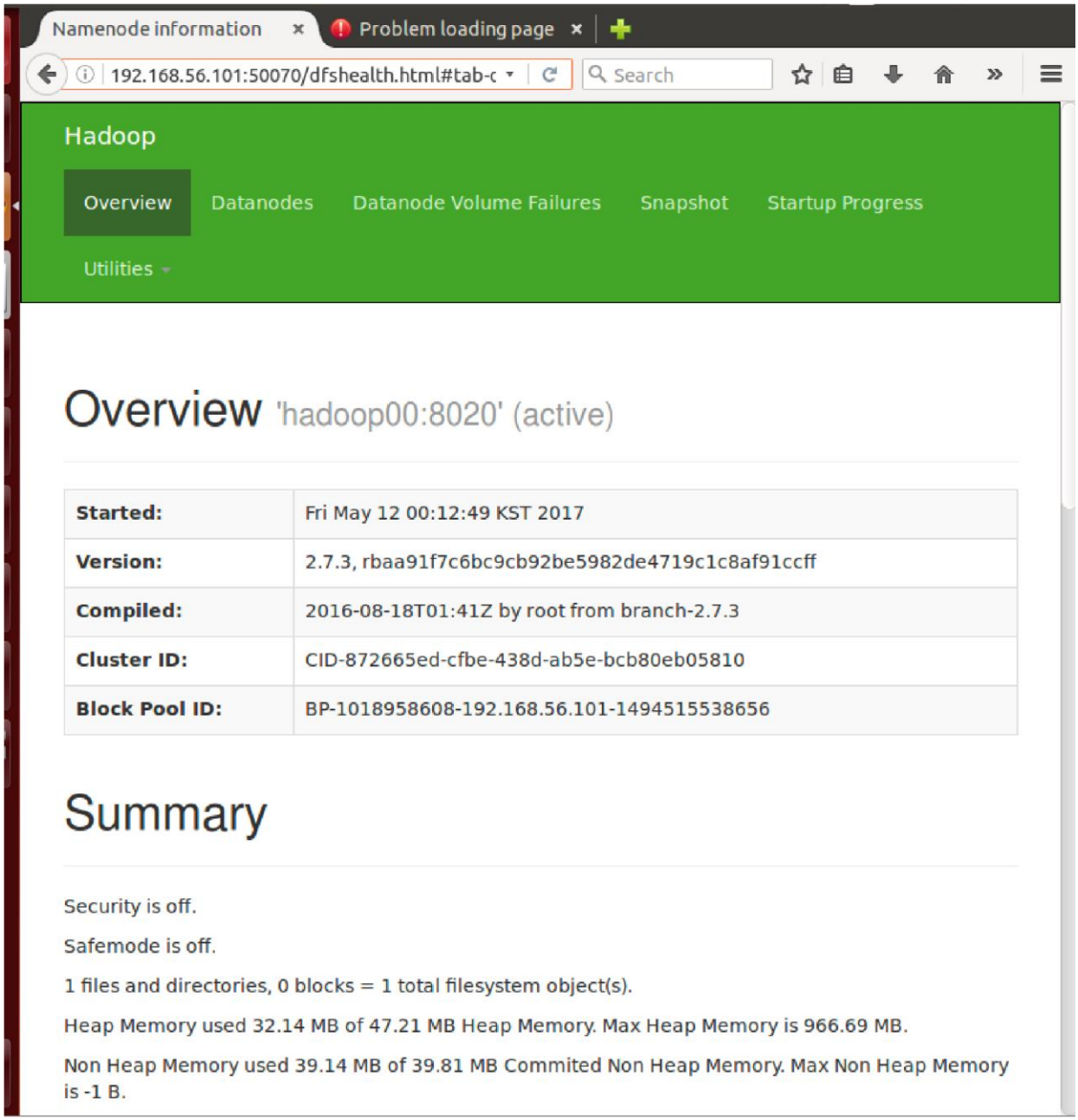
=> bin/hdfs dfsadmin -report 실행 화면

```
hadoop01@hadoop03:~/hadoop-2.7.3$ bin/yarn node -list
17/05/12 01:32:11 INFO client.RMProxy: Connecting to ResourceManager at hadoop03
/192.168.56.102:8032
Total Nodes:4
         Node-Id             Node-State Node-Http-Address       Number-of-Runnin
g-Containers
  hadoop03:38901                RUNNING     hadoop03:8042
             0
  hadoop02:44777                RUNNING     hadoop02:8042
             0
  hadoop04:42355                RUNNING     hadoop04:8042
             0
  hadoop00:36102                RUNNING     hadoop00:8042
             0
```

=> bin/yarn node -list 실행화면



=> resourcemanager 인 192.168.56.102(hadoop03):8088 실행 화면



=> namenode 인 192.168.56.101(hadoop00):50070 실행화면

2. wordcount 예제 (Fully-Distribution 상태에서 진행)

```
hadoop01@hadoop00:~/hadoop-2.7.3$ echo "Hello world in HDFS" > /home/hadoop01/testfile1
hadoop01@hadoop00:~/hadoop-2.7.3$ echo "Hadoop word count example in HDFS" > /home/hadoop
le2
```

=> wordCount 를 하기위해서 file 에 Hello world in HDFS 와 Hadoop word count example in HDFS 를 써서 testfile1, testfile2 에 저장시킨다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs dfs -mkdir /user
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs dfs -mkdir /user/hadoop01
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs dfs -mkdir /user/hadoop01/input
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs dfs -put /home/hadoop01/testfile1 /user/hadoop
01/input
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs dfs -put /home/hadoop01/testfile2 /user/hadoop
01/input
```

=> bin/hdfs dfs 를 이용해서 /user/hadoop01/input 디렉토리를 만든 후 그안에 testfile1 과 testfile2 를 put 해준다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-
examples-2.7.3.jar wordcount /user/hadoop01/input /user/hadoop01/output
17/05/12 02:21:11 INFO client.RMProxy: Connecting to ResourceManager at hadoop03/192.168.
56.102:8032
17/05/12 02:21:13 INFO input.FileInputFormat: Total input paths to process : 2
17/05/12 02:21:13 INFO mapreduce.JobSubmitter: number of splits:2
17/05/12 02:21:13 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14945200673
26_0008
17/05/12 02:21:14 INFO impl.YarnClientImpl: Submitted application application_14945200673
26_0008
17/05/12 02:21:14 INFO mapreduce.Job: The url to track the job: http://hadoop03:8088/prox
y/application_1494520067326_0008/
17/05/12 02:21:14 INFO mapreduce.Job: Running job: job_1494520067326_0008
17/05/12 02:21:27 INFO mapreduce.Job: Job job_1494520067326_0008 running in uber mode : f
alse
17/05/12 02:21:27 INFO mapreduce.Job:  map 0% reduce 0%
17/05/12 02:21:41 INFO mapreduce.Job:  map 100% reduce 0%
17/05/12 02:21:51 INFO mapreduce.Job:  map 100% reduce 100%
17/05/12 02:21:52 INFO mapreduce.Job: Job job_1494520067326_0008 completed successfully
17/05/12 02:21:52 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=120
                FILE: Number of bytes written=356654
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=274
                HDFS: Number of bytes written=62
```

=> share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar 파일을 이용해서 bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar wordcount /user/hadoop01/input /user/hadoop01/output
위에 저장한 testfile 에 대해서 wordcount 를 실시해준후 output 디렉토리에 그 결과를 저장시켜준다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs dfs -ls /user/hadoop01/output
Found 2 items
-rw-r--r--   4 hadoop01 supergroup          0 2017-05-12 02:21 /user/hadoop01/output/_SUC
CESS
-rw-r--r--   4 hadoop01 supergroup         62 2017-05-12 02:21 /user/hadoop01/output/part
-r-00000
hadoop01@hadoop00:~/hadoop-2.7.3$ bin/hdfs dfs -cat /user/hadoop01/output/part-r-00000
HDFS    2
Hadoop  1
Hello   1
count   1
example 1
in      2
word    1
world   1
hadoop01@hadoop00:~/hadoop-2.7.3$
```

=> 저장시켜준 output 파일에 존재하는 part-r-00000 파일을 확인해 보면 위에서 만들어준 2 개의 testfile 을 wordCount 한 결과를 확인 할 수 있다.

1. HIVE 설치 및 실행

```
hadoop01@hadoop00:~$ sudo wget http://apache.mirror.cdnetworks.com/hive/hive-1.2
.2/apache-hive-1.2.2-bin.tar.gz
```

=> wget 을 통해 hive 를 다운로드 받는다.

```
hadoop01@hadoop00:~$ ls
apache-hive-1.2.2-bin          examples.desktop        patterns      testfile1
apache-hive-1.2.2-bin.tar.gz   hadoop-2.7.3            patterns.txt  testfile2
derby.log                      hadoop-2.7.3.tar.gz     Pictures      Videos
Desktop                        jdk-7u80-linux-x64.rpm  Public
Documents                      metastore_db            tashu.csv
Downloads                      Music                   Templates
```

=> hive.tar.gz 압축 파일의 압축을 풀어 준다.

```
hadoop01@hadoop00:~$ export HIVE_HOME=/home/hadoop01/apache-hive-1.2.2-bin
```

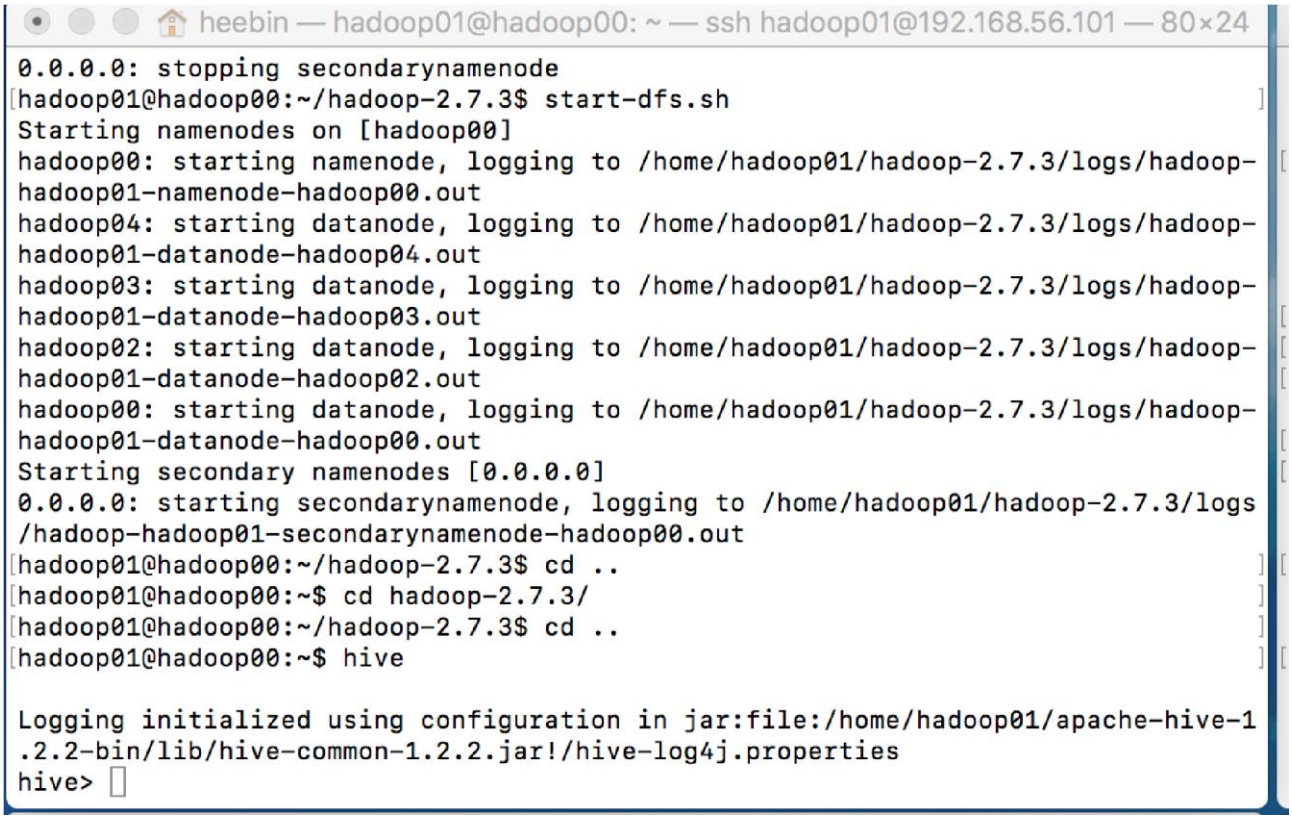
=> HIVE_HOME 의 환경변수를 HIVE 의 경로로 export 한다.

```
hadoop01@hadoop00:~$ export PATH=$HIVE_HOME/bin:$PATH
```

=> PATH 환경변수에 HIVE_HOME/bin 을 추가로 export 한다.

```
hadoop01@hadoop03:~/hadoop-2.7.3$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn-hadoo
p01-resourcemanager-hadoop03.out
hadoop02: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop02.out
hadoop00: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop00.out
hadoop04: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop04.out
hadoop03: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop03.out
hadoop01@hadoop03:~/hadoop-2.7.3$ jps
4322 NodeManager
4054 DataNode
4359 Jps
4183 ResourceManager
hadoop01@hadoop03:~/hadoop-2.7.3$
```

=> 클러스터 4 개를 모두 사용해서 hive 를 이용. start-yarn.sh 를 해서 nodemanager 와 resourcemanager 를 실행시켜준다.



=> start-dfs.sh 를 실행시켜 namenode 와 datanode, secondarynamenode 를 실행 시켜 준다.

그리고 HIVE 를 실행 시켜준다.

2. 타슈 데이터 HIVE 에 업로드하기

```
ohheebinui-MacBook-Pro:Desktop heebin$ rsync -avz /Users/heebin/Desktop/tashu.cs
v hadoop01@192.168.56.101:/home/hadoop01/
hadoop01@192.168.56.101's password:
building file list ... done
tashu.csv

sent 28897936 bytes  received 42 bytes  2512867.65 bytes/sec
total size is 124163945  speedup is 4.30
ohheebinui-MacBook-Pro:Desktop heebin$ rsync -avz /Users/heebin/Desktop/tashu.cs
v hadoop01@192.168.56.102:/home/hadoop01/
hadoop01@192.168.56.102's password:
building file list ... done
tashu.csv

sent 28897936 bytes  received 42 bytes  2512867.65 bytes/sec
total size is 124163945  speedup is 4.30
ohheebinui-MacBook-Pro:Desktop heebin$ rsync -avz /Users/heebin/Desktop/tashu.cs
v hadoop01@192.168.56.103:/home/hadoop01/
hadoop01@192.168.56.103's password:
building file list ... done
tashu.csv

sent 28897936 bytes  received 42 bytes  2311838.24 bytes/sec
total size is 124163945  speedup is 4.30
ohheebinui-MacBook-Pro:Desktop heebin$ rsync -avz /Users/heebin/Desktop/tashu.cs
v hadoop01@192.168.56.104:/home/hadoop01/
hadoop01@192.168.56.104's password:
building file list ... done
tashu.csv

sent 28897936 bytes  received 42 bytes  2752188.38 bytes/sec
total size is 124163945  speedup is 4.30
ohheebinui-MacBook-Pro:Desktop heebin$
```

=> 기존 컴퓨터에 존재하는 tashu.csv 파일을 rsync 를 이용해서 4 개의 클러스터에 모두 옮겨 준다.

```
hive> create table tmp (RENT_STATION int, RENT_DATE string, RETURN_STATION int,
RETURN_DATE string) row format delimited fields terminated by ',';
OK
Time taken: 0.495 seconds
hive> create table tashu (RENT_STATION int, RENT_DATE timestamp, RETURN_STATION
int, RETURN_DATE timestamp) row format delimited fields terminated by ',';
OK
Time taken: 0.119 seconds
hive> load data local inpath '/home/hadoop01/tashu.csv' overwrite into table tmp
;
Loading data to table default.tmp
Table default.tmp stats: [numFiles=1, numRows=0, totalSize=124163945, rawDataSiz
e=0]
OK
Time taken: 4.208 seconds
```

=> RENT_DATE 와 RETURN_DATE 의 date 타입의 데이터베이스를 저장하기 위해 RENT_DATE 와 RETURN_DATE 를 string 으로 가지고 있는 tmp table 을 하나 그리고

RENT_DATE 와 RETURN_DATE 를 date 타입으로 저장하기 위한 tashu table 을 생성한다. 그리고 tmp table 에 tashu.csv 를 overwrite 시켜 디비에 저장 시킨다.

```
hive> insert into table tashu
    > select RENT_STATION,
    > from_unixtime(unix_timestamp(RENT_DATE, 'yyyyMMddHHmmss')),
    > RETURN_STATION,
    > from_unixtime(unix_timestamp(RETURN_DATE, 'yyyyMMddHHmmss'))
    > from tmp;
Query ID = hadoop01_20170601200338_660bbf6e-6672-4df5-8dd6-7bae688c5490
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1496311355444_0001, Tracking URL = http://hadoop03:8088/proxy
/application_1496311355444_0001/
Kill Command = /home/hadoop01/hadoop-2.7.3/bin/hadoop job  -kill job_14963113554
44_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-06-01 20:03:55,034 Stage-1 map = 0%,  reduce = 0%
2017-06-01 20:04:56,007 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 38.13 sec
2017-06-01 20:05:06,756 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 49.67 s
ec
MapReduce Total cumulative CPU time: 49 seconds 670 msec
Ended Job = job_1496311355444_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://hadoop00/user/hive/warehouse/tashu/.hive-staging_hive_201
7-06-01_20-03-38_015_5800191555922286709-1/-ext-10000
Loading data to table default.tashu
Table default.tashu stats: [numFiles=1, numRows=3404664, totalSize=157063263, ra
wDataSize=153658599]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 49.67 sec   HDFS Read: 124168047 HDFS Wr
ite: 157063344 SUCCESS
Total MapReduce CPU Time Spent: 49 seconds 670 msec
OK
Time taken: 91.496 seconds
hive>
```

=> tashu table 에 RENT_DATE 와 RETURN_DATE 를 date 타입으로 저장 하기 위해서 unix_timestamp(RENT_DATE, ‘yyyyMMddHHmmss’)와 unix_timestamp(RETURN_DATE, ‘yyyyMMddHHmmss’) 를 이용해서 tashu table 에 다시 저장 시킨다. 이 과정에서 타입을 바꾸기 때문 인지 mapreduce 를 하지만 map 만 실행시킨다.

```
hive> select * from tashu limit 10;
OK
NULL    NULL    NULL    NULL
43      2013-01-01 05:56:03     34      2013-01-01 06:02:17
97      2013-01-01 06:04:00     NULL    2013-01-01 10:20:37
2       2013-01-01 06:04:06     10      2013-01-01 06:18:59
106     2013-01-01 10:53:05     105     2013-01-01 10:57:43
4       2013-01-01 11:22:23     4       2013-01-01 12:17:53
21      2013-01-01 11:39:53     105     2013-01-01 11:49:43
90      2013-01-01 12:08:33     91      2013-01-01 12:51:36
13      2013-01-01 13:14:29     30      2013-01-01 13:30:39
1       2013-01-01 13:37:42     1       2013-01-01 13:38:15
Time taken: 0.153 seconds, Fetched: 10 row(s)
```

=> tashu table 에 저장된 내용을 확인 limit 10;을 통해서 10 개의 row 만 보여주며 date 타입으로 저장된 것을 확인 할 수 있다.

3. 타슈 데이터에 쿼리를 날려 통계구하기

1) 연도별

```
hive> select year(RENT_DATE),count(*) as cnt from tashu group by year(RENT_DATE)
;
Query ID = hadoop01_20170601201547_886db48b-cefd-4503-81b8-cc650a3d12f7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1496311355444_0004, Tracking URL = http://hadoop03:8088/proxy
/application_1496311355444_0004/
Kill Command = /home/hadoop01/hadoop-2.7.3/bin/hadoop job  -kill job_14963113554
44_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-06-01 20:15:59,892 Stage-1 map = 0%,  reduce = 0%
2017-06-01 20:16:15,470 Stage-1 map = 57%,  reduce = 0%, Cumulative CPU 8.46 sec
2017-06-01 20:16:16,510 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 9.41 se
c
2017-06-01 20:16:23,802 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.84
 sec
MapReduce Total cumulative CPU time: 10 seconds 840 msec
Ended Job = job_1496311355444_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.84 sec   HDFS Read: 157075
302 HDFS Write: 44 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 840 msec
OK
NULL    1
2013    1036614
2014    1200187
2015    1167862
Time taken: 38.587 seconds, Fetched: 4 row(s)
```

=> select year(RENT_DATE), count(*) as cnt from tashu group by year(RENT_DATE); 의 쿼리문을 이용해서 date 타입으로 저장된 RENT_DATE 의 year 를 group by 를 통해서 같은 연도별로 묶어 주고 count 하게 한다.

2) 월별

```
hive> select month(RENT_DATE),count(*) as cnt from tashu group by month(RENT_DAT
E);
Query ID = hadoop01_20170601201739_51163896-47fe-40f9-9fb3-b540515bf6d4
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1496311355444_0005, Tracking URL = http://hadoop03:8088/proxy
/application_1496311355444_0005/
Kill Command = /home/hadoop01/hadoop-2.7.3/bin/hadoop job  -kill job_14963113554
44_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-06-01 20:17:54,227 Stage-1 map = 0%,  reduce = 0%
2017-06-01 20:18:13,248 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.99 se
c
2017-06-01 20:18:23,718 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.83
 sec
MapReduce Total cumulative CPU time: 10 seconds 830 msec
Ended Job = job_1496311355444_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.83 sec   HDFS Read: 157075
311 HDFS Write: 115 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 830 msec
OK
NULL    1
1       99693
2       116987
3       232712
4       290519
5       414934
6       480429
7       358507
8       320058
9       366859
10      360240
11      214771
12      148954
Time taken: 46.168 seconds, Fetched: 13 row(s)
```

=> select month(RENT_DATE), count(*) as cnt from tashu group by month(RENT_DATE); 의 쿼리

문을 이용해서 date 타입으로 저장된 RENT_DATE 의 month 를 group by 를 통해서 같은 월별로 묶어 주고 count 하게 한다.

3) 일별

```
hive> select day(RENT_DATE),count(*) as cnt from tashu group by day(RENT_DATE);
Query ID = hadoop01_20170601201938_25689018-4ee1-48d3-b116-8bfa57189644
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1496311355444_0006, Tracking URL = http://hadoop03:8088/proxy
/application_1496311355444_0006/
Kill Command = /home/hadoop01/hadoop-2.7.3/bin/hadoop job  -kill job_14963113554
44_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-06-01 20:19:48,949 Stage-1 map = 0%,  reduce = 0%
2017-06-01 20:20:03,415 Stage-1 map = 57%,  reduce = 0%, Cumulative CPU 8.36 sec
2017-06-01 20:20:04,445 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 9.22 se
c
2017-06-01 20:20:13,745 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.94
 sec
MapReduce Total cumulative CPU time: 10 seconds 940 msec
Ended Job = job_1496311355444_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.94 sec   HDFS Read: 157075
307 HDFS Write: 305 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 940 msec
OK
NULL    1
1       116298
2       104499
3       106474
4       115649
5       115198
6       110271
7       105940
8       107095
9       117166
10      110197
11      105053
12      104176
13      107960
14      110600
15      120347
16      120099
17      114330
18      105077
19      112020
20      106850
21      113472
22      120796
23      111161
24      110613
25      104908
26      120407
27      106867
28      119053
29      102871
30      114542
31      64674
Time taken: 36.218 seconds, Fetched: 32 row(s)
```

=> select day(RENT_DATE), count(*) as cnt from tashu group by day(RENT_DATE); 의 쿼리문을 이용해서 date 타입으로 저장된 RENT_DATE 의 day 를 group by 를 통해서 같은 일별로 묶어주고 count 하게 한다.

4) 시간별

```
hive> select hour(RENT_DATE),count(*) as cnt from tashu group by hour(RENT_DATE)
;
Query ID = hadoop01_20170601202044_14beddca-005a-4b0f-b194-7e3c40e6f12f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1496311355444_0007, Tracking URL = http://hadoop03:8088/proxy
/application_1496311355444_0007/
Kill Command = /home/hadoop01/hadoop-2.7.3/bin/hadoop job  -kill job_14963113554
44_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-06-01 20:20:54,583 Stage-1 map = 0%,  reduce = 0%
2017-06-01 20:21:10,040 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.92 se
c
2017-06-01 20:21:20,358 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.73
 sec
MapReduce Total cumulative CPU time: 10 seconds 730 msec
Ended Job = job_1496311355444_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.73 sec   HDFS Read: 157075
302 HDFS Write: 218 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 730 msec
OK
NULL    1
0       63022
1       15205
2       199
3       3
4       11
5       16591
6       22152
7       129110
8       186421
9       142126
10      111652
11      96250
12      118768
13      157488
14      173437
15      167018
16      194413
17      250842
18      292905
19      241023
20      259816
21      273561
22      251385
23      241265
Time taken: 37.23 seconds, Fetched: 25 row(s)
```

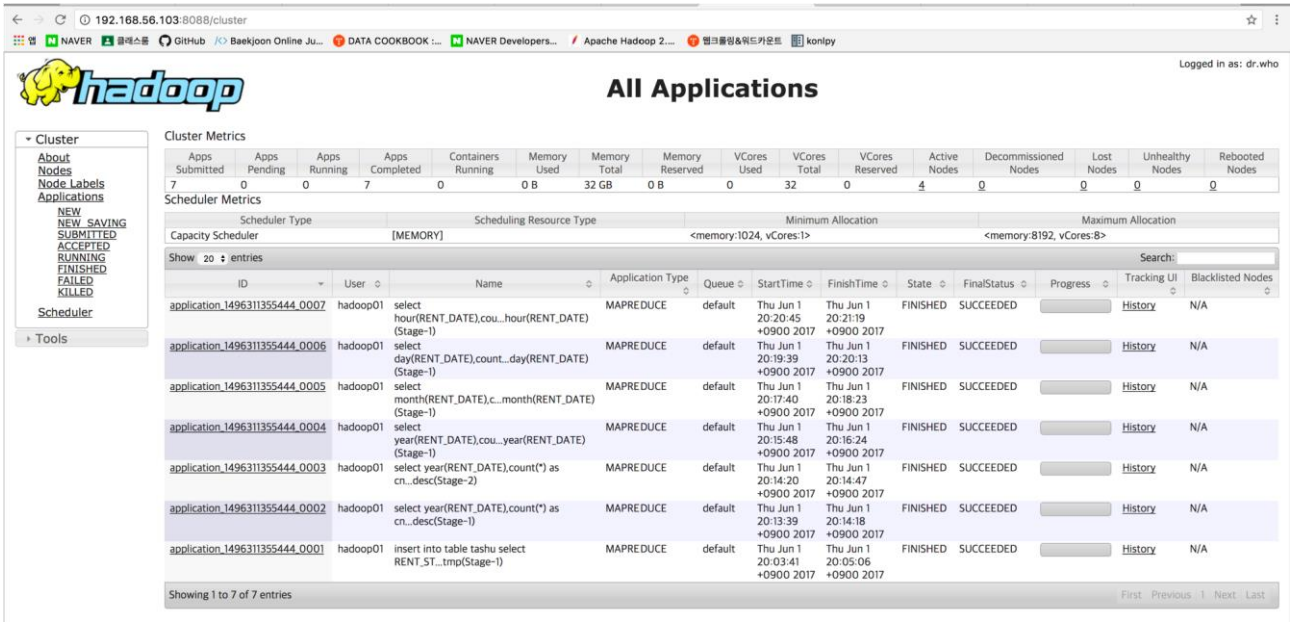
=> select hour(RENT_DATE), count(*) as cnt from tashu group by hour(RENT_DATE); 의 쿼리문을 이용해서 date 타입으로 저장된 RENT_DATE 의 hour 를 group by 를 통해서 같은 시간별로 묶어 주 고 count 하게 한다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfsadmin -report
Configured Capacity: 81369726976 (75.78 GB)
Present Capacity: 53813940224 (50.12 GB)
DFS Remaining: 52671918080 (49.05 GB)
DFS Used: 1142022144 (1.06 GB)
DFS Used%: 2.12%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-------------------------------------------------
Live datanodes (4):

Name: 192.168.56.101:50010 (hadoop00)
Hostname: hadoop00
Decommission Status : Normal
Configured Capacity: 20342431744 (18.95 GB)
DFS Used: 285511680 (272.29 MB)
Non DFS Used: 6925783040 (6.45 GB)
DFS Remaining: 13131137024 (12.23 GB)
DFS Used%: 1.40%
DFS Remaining%: 64.55%
Configured Cache Capacity: 0 (0 B)
```

hdfs dfsadmin -report

```
hadoop01@hadoop03:~$ yarn node -list
17/06/01 20:43:08 INFO client.RMProxy: Connecting to ResourceManager at hadoop03
/192.168.56.103:8032
Total Nodes:4
         Node-Id             Node-State Node-Http-Address       Number-of-Runnin
g-Containers
  hadoop03:42230                RUNNING     hadoop03:8042
            0
  hadoop02:42874                RUNNING     hadoop02:8042
            0
  hadoop00:38715                RUNNING     hadoop00:8042
            0
  hadoop04:41044                RUNNING     hadoop04:8042
            0
hadoop01@hadoop03:~$
```

yarn node -list



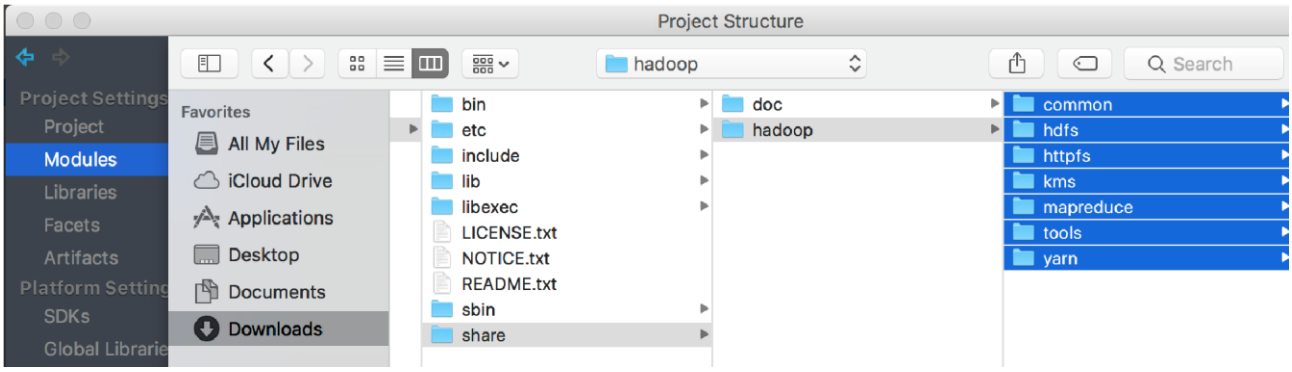
=> 192.168.56.103:8088 을 통해서 mapreduce 결과를 확인 할 수 있다.

1. wordcount V1

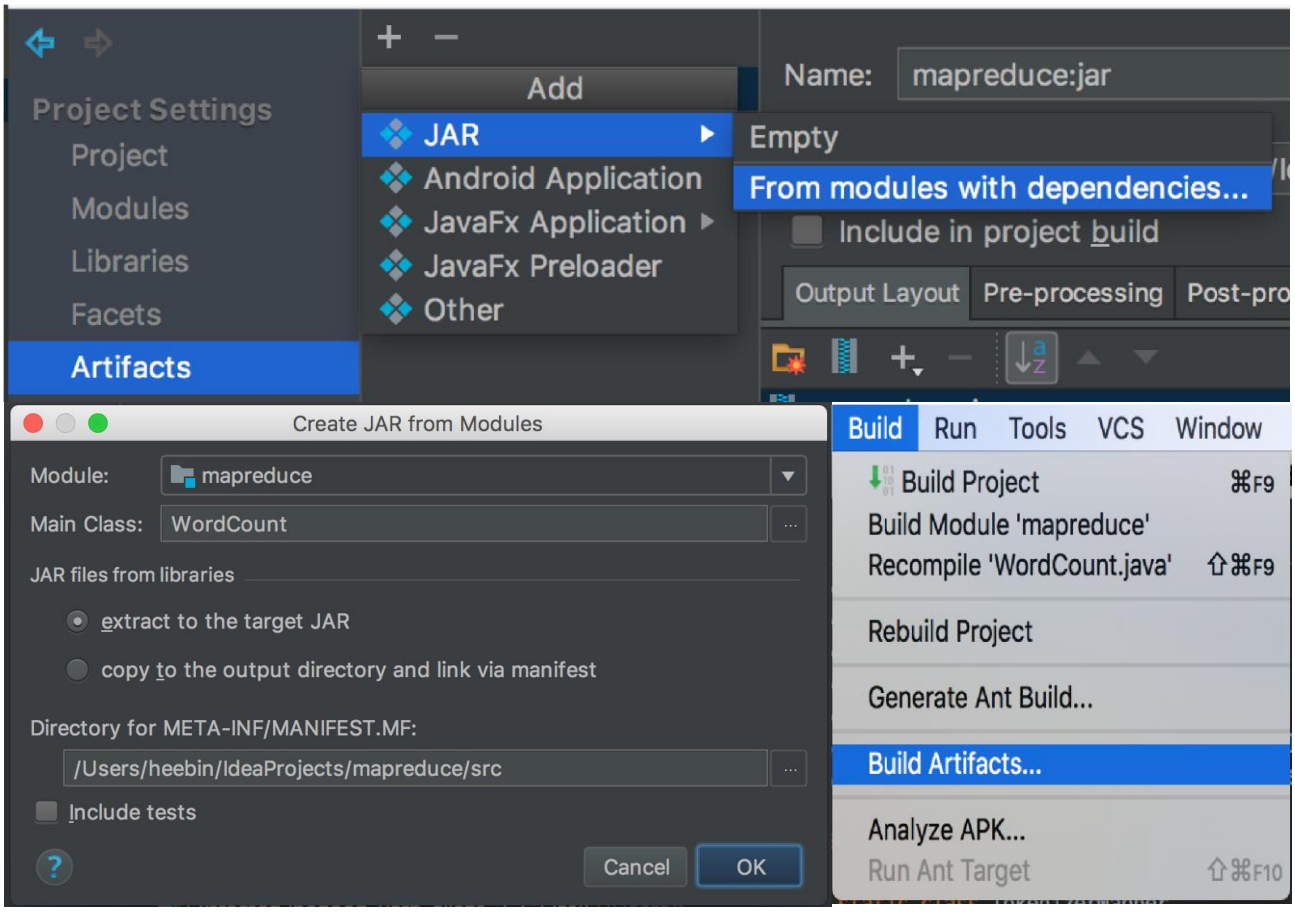
1) java 파일 컴파일 및 build

```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper
            extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable( value: 1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer
            extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
        ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf,  jobName: "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

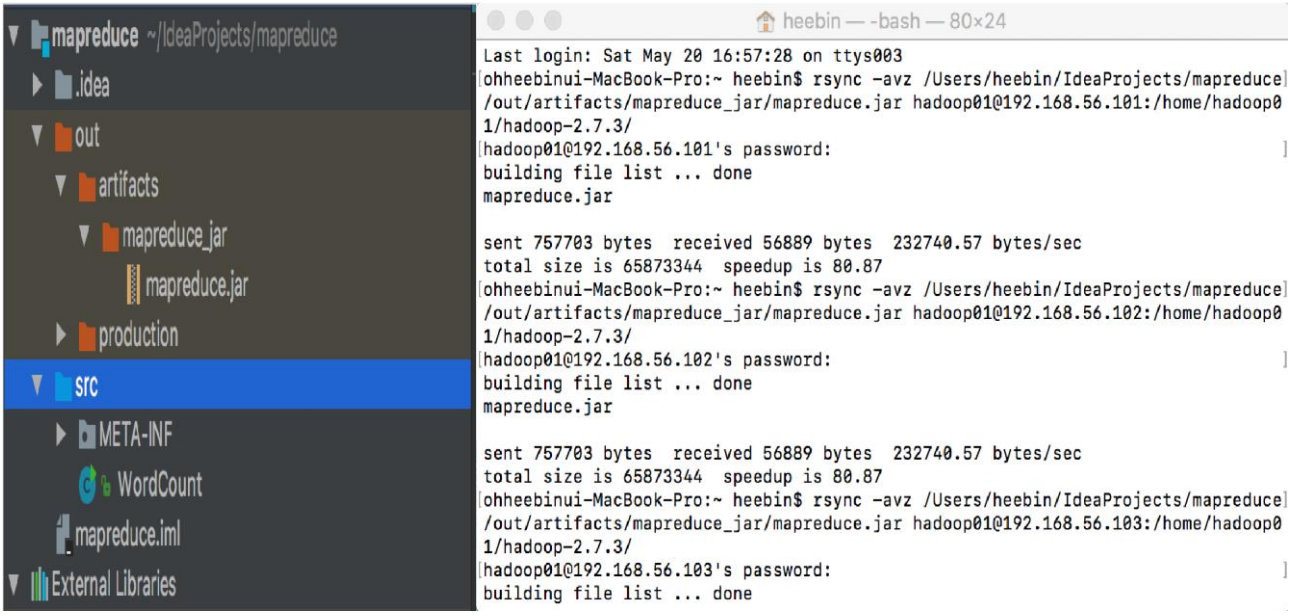
code



=> hadoop wordcount 의 java code 를 intellij 에서 compile 하기 위해서 project 를 생성하고 그 project 의 Project structure 에 들어가서 Modules 에 hadoop 에 사용되는 패키지들을 사용할 수 있게끔 위와 같 이 추가해 준다.



=> 그리고 java code 를 hadoop 에서 사용할 수 있게 jar 파일로 build 할수 있게 설정해준다. Main Class 를 jar 파일로 만들 WordCount class 를 선택하고 Build Artifacts 를 선택해 java code 를 jar 파일로 build 시켜 준다.



=> build 후 jar 파일이 생성된 것을 확인 할 수 있다. 그리고 생성된 jar 파일을 hadoop00, hadoop02, hadoop03, hadoop04 에 rsync 를 이용해서 hadoop 디렉토리에 복사해 준다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ ls
bin  etc      lib      LICENSE.txt  mapreduce.jar  README.txt  share
dfs  include  libexec  logs         NOTICE.txt     sbin
```

=> hadoop 디렉토리에 mapreduce.jar 파일이 복사된 것을 확인 할 수 있다.

2) hadoop 실행 및 wordcount

```
hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs namenode -format
17/05/18 22:39:23 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = hadoop00/192.168.56.101
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.3
STARTUP_MSG:   classpath = /home/hadoop01/hadoop-2.7.3/etc/hadoop:/home/hadoop01
/hadoop-2.7.3/share/hadoop/common/lib/zookeeper-3.4.6.jar:/home/hadoop01/hadoop-
2.7.3/share/hadoop/common/lib/jettison-1.1.jar:/home/hadoop01/hadoop-2.7.3/share
/hadoop/common/lib/httpclient-4.2.5.jar:/home/hadoop01/hadoop-2.7.3/share/hadoop
```

hdfs namenode -format

```
[hadoop01@hadoop00:~/hadoop-2.7.3$ start-dfs.sh
Starting namenodes on [hadoop00]
hadoop00: starting namenode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-namenode-hadoop00.out
hadoop03: starting datanode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-datanode-hadoop03.out
hadoop04: starting datanode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-datanode-hadoop04.out
hadoop02: starting datanode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-datanode-hadoop02.out
hadoop00: starting datanode, logging to /home/hadoop01/hadoop-2.7.3/logs/hadoop-
hadoop01-datanode-hadoop00.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/hadoop01/hadoop-2.7.3/logs
/hadoop-hadoop01-secondarynamenode-hadoop00.out
```

start-dfs.sh

```
[hadoop01@hadoop03:~/hadoop-2.7.3$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn-hadoo
p01-resourcemanager-hadoop03.out
hadoop02: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop02.out
hadoop04: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop04.out
hadoop00: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop00.out
hadoop03: starting nodemanager, logging to /home/hadoop01/hadoop-2.7.3/logs/yarn
-hadoop01-nodemanager-hadoop03.out
```

start-yarn.sh

```
[hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfs -mkdir /input
[hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfs -ls /input
[hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfs -put /home/hadoop01/testfile* /input
[hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfs -ls /input
Found 2 items
-rw-r--r--   4 hadoop01 supergroup         24 2017-05-18 22:41 /input/testfile1
-rw-r--r--   4 hadoop01 supergroup         30 2017-05-18 22:41 /input/testfile2
```

=> 만든 text 파일을 저장할 input 디렉토리를 hdfs dfs -mkdir /input 을 이용해서 생성한 후 text 파일 을 -put 해준다. input 디렉토리에 text 파일이 put 된 것을 확인 할 수 있다.

```
[hadoop01@hadoop00:~/hadoop-2.7.3$ hadoop jar mapreduce.jar /input /output
17/05/18 22:44:31 INFO client.RMProxy: Connecting to ResourceManager at hadoop03
/192.168.56.103:8032
17/05/18 22:44:32 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
17/05/18 22:44:35 INFO input.FileInputFormat: Total input paths to process : 2
17/05/18 22:44:36 INFO mapreduce.JobSubmitter: number of splits:2
17/05/18 22:44:36 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14
95114805970_0002
17/05/18 22:44:37 INFO impl.YarnClientImpl: Submitted application application_14
95114805970_0002
17/05/18 22:44:37 INFO mapreduce.Job: The url to track the job: http://hadoop03:
8088/proxy/application_1495114805970_0002/
17/05/18 22:44:37 INFO mapreduce.Job: Running job: job_1495114805970_0002
17/05/18 22:44:49 INFO mapreduce.Job: Job job_1495114805970_0002 running in uber
 mode : false
17/05/18 22:44:49 INFO mapreduce.Job:  map 0% reduce 0%
17/05/18 22:45:04 INFO mapreduce.Job:  map 100% reduce 0%
17/05/18 22:45:12 INFO mapreduce.Job:  map 100% reduce 100%
17/05/18 22:45:13 INFO mapreduce.Job: Job job_1495114805970_0002 completed succe
ssfully
17/05/18 22:45:13 INFO mapreduce.Job: Counters: 49
        File System Counters
```

=> hadoop jar mapreduce.jar /input /output 을 이용해 /input 디렉토리에 존재하는 text 파일들을 wordcount 한 후 /output 디렉토리에 그 결과를 저장한다. 위와 같이 command 를 입력한 이유는 code 의 마지막 부분에 addInputPath 가 args[0] 즉 command 의 첫 번째 argument 를 input 으로 확인 하기에 mapreduce.jar 다음에 input 할 path 를 입력 해주고 addOutputPath 가 args[1]인 두 번째 argument 를 output 하기에 /input 후에 /output 을 입력해주었다.

```
[hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfs -ls /output
Found 2 items
-rw-r--r--   4 hadoop01 supergroup          0 2017-05-18 22:45 /output/_SUCCESS
-rw-r--r--   4 hadoop01 supergroup         62 2017-05-18 22:45 /output/part-r-00
000
[hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfs -cat /output/part-r-00000
Bye     1
Goodbye 1
Hadoop, 1
Hello   2
World!  1
World,  1
hadoop. 1
hadoop01@hadoop00:~/hadoop-2.7.3$
```

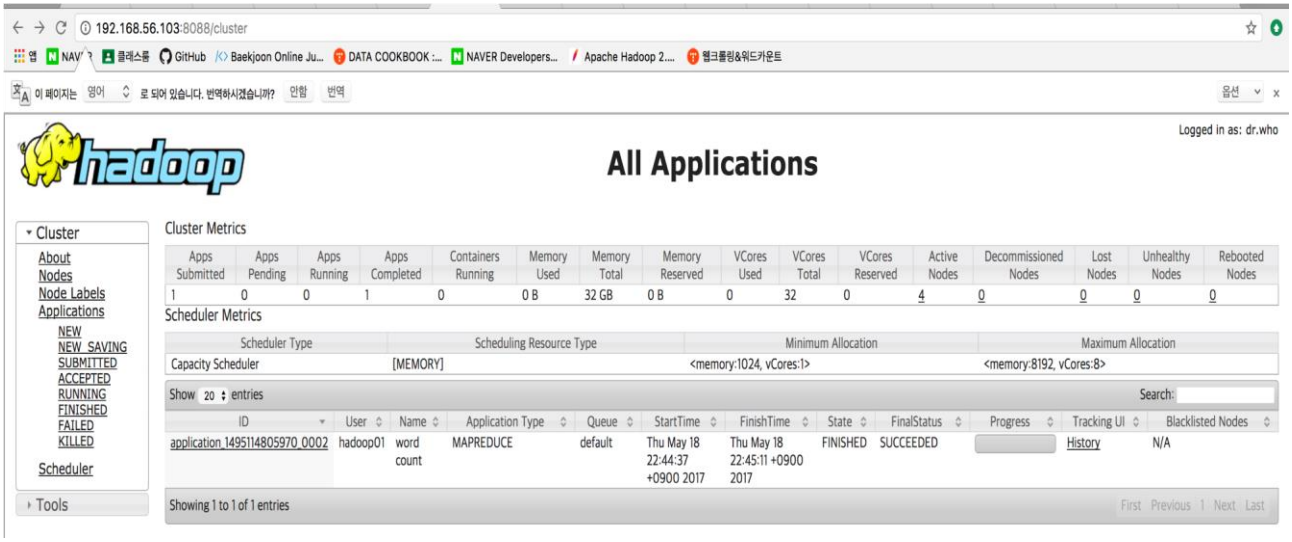
=> output 디렉토리에 input 디렉토리에 put 한 text 파일들이 전부 word-count 된 결과가 저장된 것을 확인 할 수 있다.

```
[hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfsadmin -report
Configured Capacity: 81369726976 (75.78 GB)
Present Capacity: 55003807744 (51.23 GB)
DFS Remaining: 55002857472 (51.23 GB)
DFS Used: 950272 (928 KB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-------------------------------------------------
Live datanodes (4):

Name: 192.168.56.103:50010 (hadoop03)
Hostname: hadoop03
Decommission Status : Normal
Configured Capacity: 20342431744 (18.95 GB)
DFS Used: 237568 (232 KB)
Non DFS Used: 6641909760 (6.19 GB)
DFS Remaining: 13700284416 (12.76 GB)
DFS Used%: 0.00%
DFS Remaining%: 67.35%
Configured Cache Capacity: 0 (0 B)
```

hdfs dfsadmin -report

```
[hadoop01@hadoop03:~/hadoop-2.7.3$ yarn node -list
17/05/18 22:58:58 INFO client.RMProxy: Connecting to ResourceManager at hadoop03
/192.168.56.103:8032
Total Nodes:4
         Node-Id             Node-State Node-Http-Address       Number-of-Runnin
g-Containers
  hadoop00:36066                RUNNING     hadoop00:8042
             0
  hadoop03:42186                RUNNING     hadoop03:8042
             0
  hadoop04:33930                RUNNING     hadoop04:8042
             0
  hadoop02:35541                RUNNING     hadoop02:8042
             0
```

yarn node -list



=> resourceManager site 에 접속 해 보면 방금 실행한 wordCount 에 대해 실행 된 것을 확인 할 수 있 다.

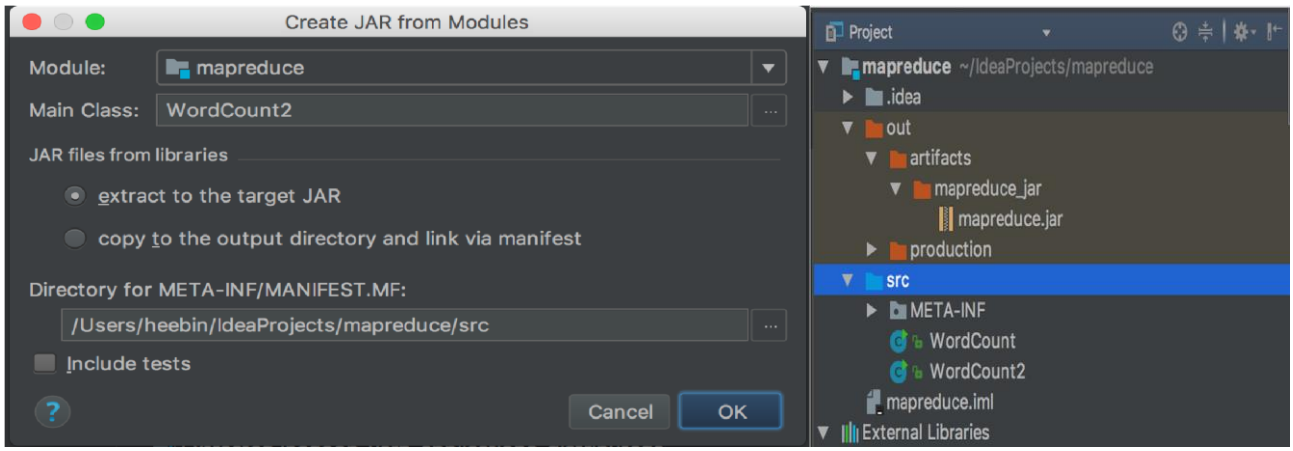
2. wordcount V2 1) java 파일 컴파일 및 build

```java
/**
 * Created by heebin on 2017. 5. 18..
 */
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.net.URI;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.util.StringUtils;

import org.apache.commons.cli.Option;

public class WordCount2 {

    public static class TokenizerMapper
            extends Mapper<Object, Text, Text, IntWritable>{

        static enum CountersEnum { INPUT_WORDS }

        private final static IntWritable one = new IntWritable( value: 1);
        private Text word = new Text();

        private boolean caseSensitive;
        private Set<String> patternsToSkip = new HashSet<String>();

        private Configuration conf;
        private BufferedReader fis;

        @Override
        public void setup(Context context) throws IOException,
                InterruptedException {
            conf = context.getConfiguration();
            caseSensitive = conf.getBoolean( name: "wordcount.case.sensitive",  defaultValue: true);
            if (conf.getBoolean( name: "wordcount.skip.patterns",  defaultValue: false)) {
                URI[] patternsURIs = Job.getInstance(conf).getCacheFiles();
                for (URI patternsURI : patternsURIs) {
                    Path patternsPath = new Path(patternsURI.getPath());
                    String patternsFileName = patternsPath.getName().toString();
                    parseSkipFile(patternsFileName);
                }
            }
        }
```

```
        private void parseSkipFile(String fileName) {
            try {
                fis = new BufferedReader(new FileReader(fileName));
                String pattern = null;
                while ((pattern = fis.readLine()) != null) {
                    patternsToSkip.add(pattern);
                }
            } catch (IOException ioe) {
                System.err.println("Caught exception while parsing the cached file '"
                        + StringUtils.stringifyException(ioe));
            }
        }

        @Override
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            String line = (caseSensitive) ?
                    value.toString() : value.toString().toLowerCase();
            for (String pattern : patternsToSkip) {
                line = line.replaceAll(pattern,  replacement: "");
            }
            StringTokenizer itr = new StringTokenizer(line);
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
                Counter counter = context.getCounter(CountersEnum.class.getName(),
                        CountersEnum.INPUT_WORDS.toString());
                counter.increment( l: 1);
            }
        }
    }

    public static class IntSumReducer
            extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
        ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
```

```
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        GenericOptionsParser optionParser = new GenericOptionsParser(conf, args);
        String[] remainingArgs = optionParser.getRemainingArgs();
        if ((remainingArgs.length != 2) && (remainingArgs.length != 4) && (remainingArgs.length != 5)) {
            System.err.println("Usage: wordcount <in> <out> [-skip skipPatternFile]");
            System.exit( status: 2);
        }
        Job job = Job.getInstance(conf,  jobName: "word count");
        job.setJarByClass(WordCount2.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        List<String> otherArgs = new ArrayList<String>();
        for (int i=0; i < remainingArgs.length; ++i) {
            if ("-skip".equals(remainingArgs[i])) {
                job.addCacheFile(new Path(remainingArgs[++i]).toUri());
                job.getConfiguration().setBoolean( name: "wordcount.skip.patterns",  value: true);
            } else {
                otherArgs.add(remainingArgs[i]);
            }
        }
        FileInputFormat.addInputPath(job, new Path(otherArgs.get(0)));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs.get(1)));

        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

wordCount V2 의 code



=> build 과정에서 Main Class 를 WordCount2 로 한 후 build 를 해서 jar 파일을 생성 한다.

2) 오류 수정

1. code 에 remainingArgs.length != 5 추가 하기.

```
if ((remainingArgs.length != 2) && (remainingArgs.length != 4) && (remainingArgs.length != 5)) {
    System.err.println("Usage: wordcount <in> <out> [-skip skipPatternFile]");
    System.exit( status: 2);
}
```

V2 에서 code 의 오류는 command 를 입력 할 때 argument 의 수에 대한 오류인 것 같다. code 의 main 부분에 처음에는 remainingArgs.legth 가 2 와 4 일 때만 main 함수를 실행 하게 되있었는데 입력에 필요한 command 가 1. wordcount.case.snsitive(대소문자구별) 2. /input 디렉토리의 path 3. /output 디렉토리의 path 4. -skip 5. skip 에 필요한 patterns 파일의 path 총 5 개가 필요하다고 판단 했고 이를 위해 **remainingArgs.legth != 5** 를 추가해 주어서 필요한 argument 를 전부 사용할 수 있게 오류를 수정했다.

2. 1 번의 오류를 수정하지 않고 caseSensitive 의 defaultValue 변경 후 command 를 다르게 실행 시키기

```
@Override
public void setup(Context context) throws IOException,
        InterruptedException {
    conf = context.getConfiguration();
    caseSensitive = conf.getBoolean( name: "wordcount.case.sensitive", defaultValue: true);
    if (conf.getBoolean( name: "wordcount.skip.patterns", defaultValue: false)) {
        URI[] patternsURIs = Job.getInstance(conf).getCacheFiles();
        for (URI patternsURI : patternsURIs) {
            Path patternsPath = new Path(patternsURI.getPath());
            String patternsFileName = patternsPath.getName().toString();
            parseSkipFile(patternsFileName);
        }
    }
}
```

=> 그리고 이 오류를 수정하는 것 말고 command 문제를 해결하는 방법이 있는데 대소문자를 구별해주

는 caseSensitive 의 defaultValue 를 true 로 한번 false 로 한번 씩 build 시키고 hadoop jar mapreduce.jar /input /output -skip /patterns 와 같이 command 에 Sensitive 에 대한 것을 입력해 주

지 않고 true 로 한번 false 로 한번씩 실행 시키면 argument 값이 4 개가 되기에 1 번의 code 를 추가해 주 지 않아도 wordCount V2 에 대해서 대소문자를 구별한 것과 구별하지 않은 2 개의 다른 결과를 얻을 수 있다.

3) wordCount V2 실행

```
heebin — -bash — 80×24
Last login: Sat May 20 16:57:28 on ttys003
ohheebinui-MacBook-Pro:~ heebin$ rsync -avz /Users/heebin/IdeaProjects/mapreduce
/out/artifacts/mapreduce_jar/mapreduce.jar hadoop01@192.168.56.101:/home/hadoop0
1/hadoop-2.7.3/
hadoop01@192.168.56.101's password:
building file list ... done
mapreduce.jar

sent 757703 bytes  received 56889 bytes  232740.57 bytes/sec
total size is 65873344  speedup is 80.87
ohheebinui-MacBook-Pro:~ heebin$ rsync -avz /Users/heebin/IdeaProjects/mapreduce
/out/artifacts/mapreduce_jar/mapreduce.jar hadoop01@192.168.56.102:/home/hadoop0
1/hadoop-2.7.3/
hadoop01@192.168.56.102's password:
building file list ... done
mapreduce.jar

sent 757703 bytes  received 56889 bytes  232740.57 bytes/sec
total size is 65873344  speedup is 80.87
ohheebinui-MacBook-Pro:~ heebin$ rsync -avz /Users/heebin/IdeaProjects/mapreduce
/out/artifacts/mapreduce_jar/mapreduce.jar hadoop01@192.168.56.103:/home/hadoop0
1/hadoop-2.7.3/
hadoop01@192.168.56.103's password:
building file list ... done
```

=> rsync 를 이용해서 build 한 jar 파일을 hadoop00, hadoop02, hadoop03, hadoop04 에 보낸다.

```
hadoop01@hadoop00:~$ vi patterns
hadoop01@hadoop00:~$ cd hadoop-2.7.3/
hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfs -put /home/hadoop01/patterns /user
hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfs -put /home/hadoop01/testfile* /user/i
nput
hadoop01@hadoop00:~/hadoop-2.7.3$ hadoop fs -cat /user/patterns
\.
\,
\!
to
```

=> testfile 과 patterns 파일을 /user/input 디렉토리에 -put 해준다.

```
hadoop01@hadoop00:~/hadoop-2.7.3$ hadoop jar mapreduce.jar -Dwordcount.case.sens
itive=true /user/input /user/output -skip /user/patterns
17/05/20 18:21:43 INFO client.RMProxy: Connecting to ResourceManager at hadoop03
/192.168.56.103:8032
17/05/20 18:21:46 INFO input.FileInputFormat: Total input paths to process : 2
17/05/20 18:21:46 INFO mapreduce.JobSubmitter: number of splits:2
17/05/20 18:21:47 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14
95267316986_0010
17/05/20 18:21:47 INFO impl.YarnClientImpl: Submitted application application_14
95267316986_0010
17/05/20 18:21:47 INFO mapreduce.Job: The url to track the job: http://hadoop03:
8088/proxy/application_1495267316986_0010/
17/05/20 18:21:47 INFO mapreduce.Job: Running job: job_1495267316986_0010
17/05/20 18:21:57 INFO mapreduce.Job: Job job_1495267316986_0010 running in uber
 mode : false
17/05/20 18:21:57 INFO mapreduce.Job:  map 0% reduce 0%
17/05/20 18:22:11 INFO mapreduce.Job:  map 100% reduce 0%
17/05/20 18:22:18 INFO mapreduce.Job:  map 100% reduce 100%
17/05/20 18:22:19 INFO mapreduce.Job: Job job_1495267316986_0010 completed succe
ssfully
17/05/20 18:22:20 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=92
                FILE: Number of bytes written=359673
```

```
hadoop01@hadoop00:~/hadoop-2.7.3$ hadoop fs -cat /user/output/part-r-00000
Bye     1
Goodbye 1
Hadoop  1
Hello   2
World   2
hadoop  1
```

=> hadoop jar mapreduce.jar -Dwordcount.case.sensitive=true /user/input /user/output -skip /

user/patterns command 를 이용해서 대소문자를 구별하고 patterns 에 쓰여 있는 . , , , ! , to 문자들만 제거하는 word-count 를 실행하고 그 결과를 확인

```
[hadoop01@hadoop00:~/hadoop-2.7.3$ hadoop jar mapreduce.jar -Dwordcount.case.sens]
itive=false /user/input /user/output -skip /user/patterns
17/05/20 18:23:16 INFO client.RMProxy: Connecting to ResourceManager at hadoop03
/192.168.56.103:8032
17/05/20 18:23:20 INFO input.FileInputFormat: Total input paths to process : 2
17/05/20 18:23:20 INFO mapreduce.JobSubmitter: number of splits:2
17/05/20 18:23:20 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14
95267316986_0011
17/05/20 18:23:21 INFO impl.YarnClientImpl: Submitted application application_14
95267316986_0011
17/05/20 18:23:21 INFO mapreduce.Job: The url to track the job: http://hadoop03:
8088/proxy/application_1495267316986_0011/
17/05/20 18:23:21 INFO mapreduce.Job: Running job: job_1495267316986_0011
17/05/20 18:23:36 INFO mapreduce.Job: Job job_1495267316986_0011 running in uber
 mode : false
17/05/20 18:23:36 INFO mapreduce.Job:  map 0% reduce 0%
17/05/20 18:23:52 INFO mapreduce.Job:  map 100% reduce 0%
17/05/20 18:24:01 INFO mapreduce.Job:  map 100% reduce 100%
17/05/20 18:24:02 INFO mapreduce.Job: Job job_1495267316986_0011 completed succe
ssfully
17/05/20 18:24:02 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=79
                FILE: Number of bytes written=359650
```

```
[hadoop01@hadoop00:~/hadoop-2.7.3$ hadoop fs -cat /user/output/part-r-00000      ]
bye     1
goodbye 1
hadoop  2
hello   2
world   2
```

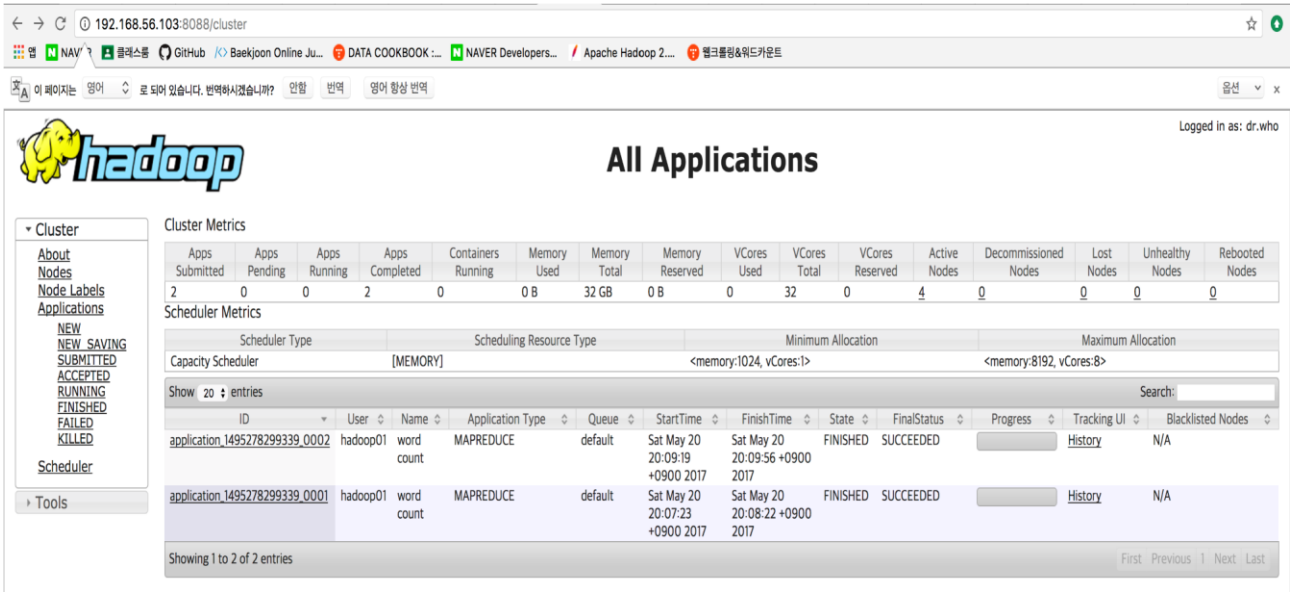
=> hadoop jar mapreduce.jar -Dwordcount.case.sensitive=false /user/input /user/output -skip /

user/patterns command 를 이용해서 대소문구별 없이 전부 소문자로 읽고 patterns 에 쓰여 있는 . , ! , !

, to 문자들을 제거하는 word-count 를 실행하고 그 결과를 확인

sensitive 가 true 일 때와 false 일 때 다른 두가지 결과를 확인 할 수

있다.



=> resourceManager site 에서 확인 한 결과 각각 실행에 대한 결과를 확인 할 수 있다.

yarn node -list

```
[hadoop01@hadoop00:~/hadoop-2.7.3$ hdfs dfsadmin -report
Configured Capacity: 81369726976 (75.78 GB)
Present Capacity: 55242159660 (51.45 GB)
DFS Remaining: 55240773632 (51.45 GB)
DFS Used: 1386028 (1.32 MB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-------------------------------------------------
Live datanodes (4):

Name: 192.168.56.104:50010 (hadoop04)
Hostname: hadoop04
Decommission Status : Normal
Configured Capacity: 20342431744 (18.95 GB)
DFS Used: 346507 (338.39 KB)
Non DFS Used: 6552680053 (6.10 GB)
DFS Remaining: 13789405184 (12.84 GB)
DFS Used%: 0.00%
DFS Remaining%: 67.79%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat May 20 20:10:47 KST 2017


Name: 192.168.56.103:50010 (hadoop03)
Hostname: hadoop03
Decommission Status : Normal
Configured Capacity: 20342431744 (18.95 GB)
DFS Used: 346507 (338.39 KB)
Non DFS Used: 6554232437 (6.10 GB)
DFS Remaining: 13787852800 (12.84 GB)
DFS Used%: 0.00%
DFS Remaining%: 67.78%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat May 20 20:10:47 KST 2017
```

```
Name: 192.168.56.102:50010 (hadoop02)
Hostname: hadoop02
Decommission Status : Normal
Configured Capacity: 20342431744 (18.95 GB)
DFS Used: 346507 (338.39 KB)
Non DFS Used: 6552839797 (6.10 GB)
DFS Remaining: 13789245440 (12.84 GB)
DFS Used%: 0.00%
DFS Remaining%: 67.79%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat May 20 20:10:47 KST 2017


Name: 192.168.56.101:50010 (hadoop00)
Hostname: hadoop00
Decommission Status : Normal
Configured Capacity: 20342431744 (18.95 GB)
DFS Used: 346507 (338.39 KB)
Non DFS Used: 6467815029 (6.02 GB)
DFS Remaining: 13874270208 (12.92 GB)
DFS Used%: 0.00%
DFS Remaining%: 68.20%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat May 20 20:10:46 KST 2017
```

hdfs dfsadmin -report