마. 실습3 - Multi Node Cluster(완전분산모드)

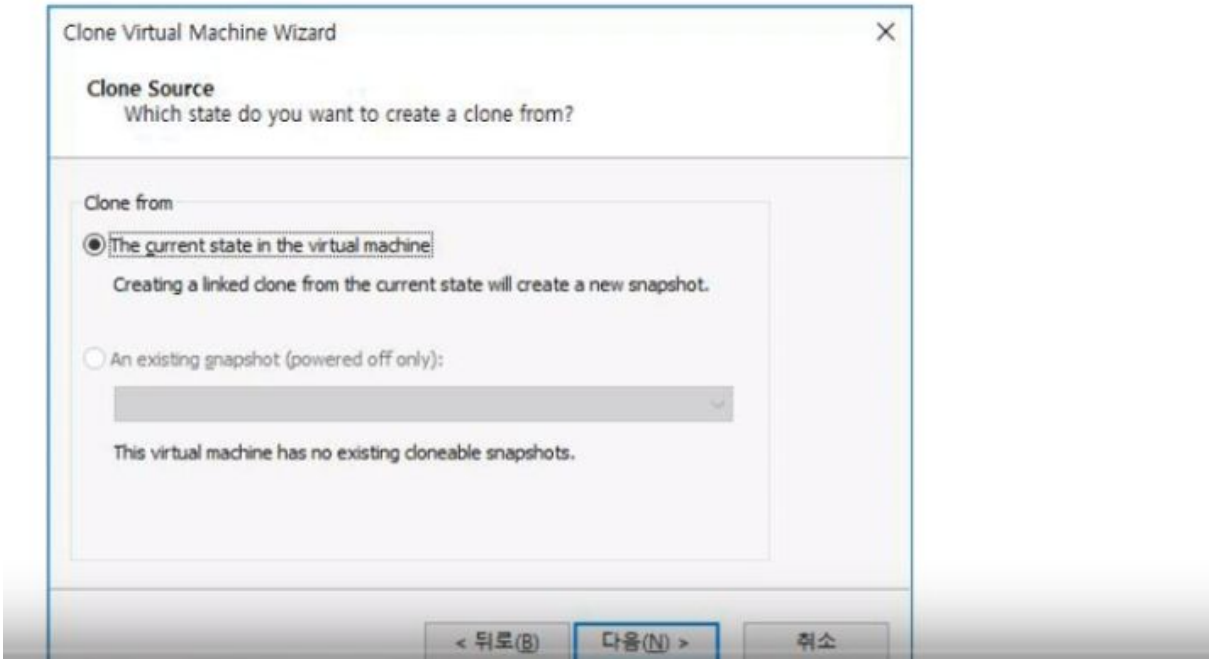
1) VMplayer의 경우 복제 기능이 없으므로 가상머신이 설치된 디렉토리를 복사하여 아래와 같이 디렉토리 구성

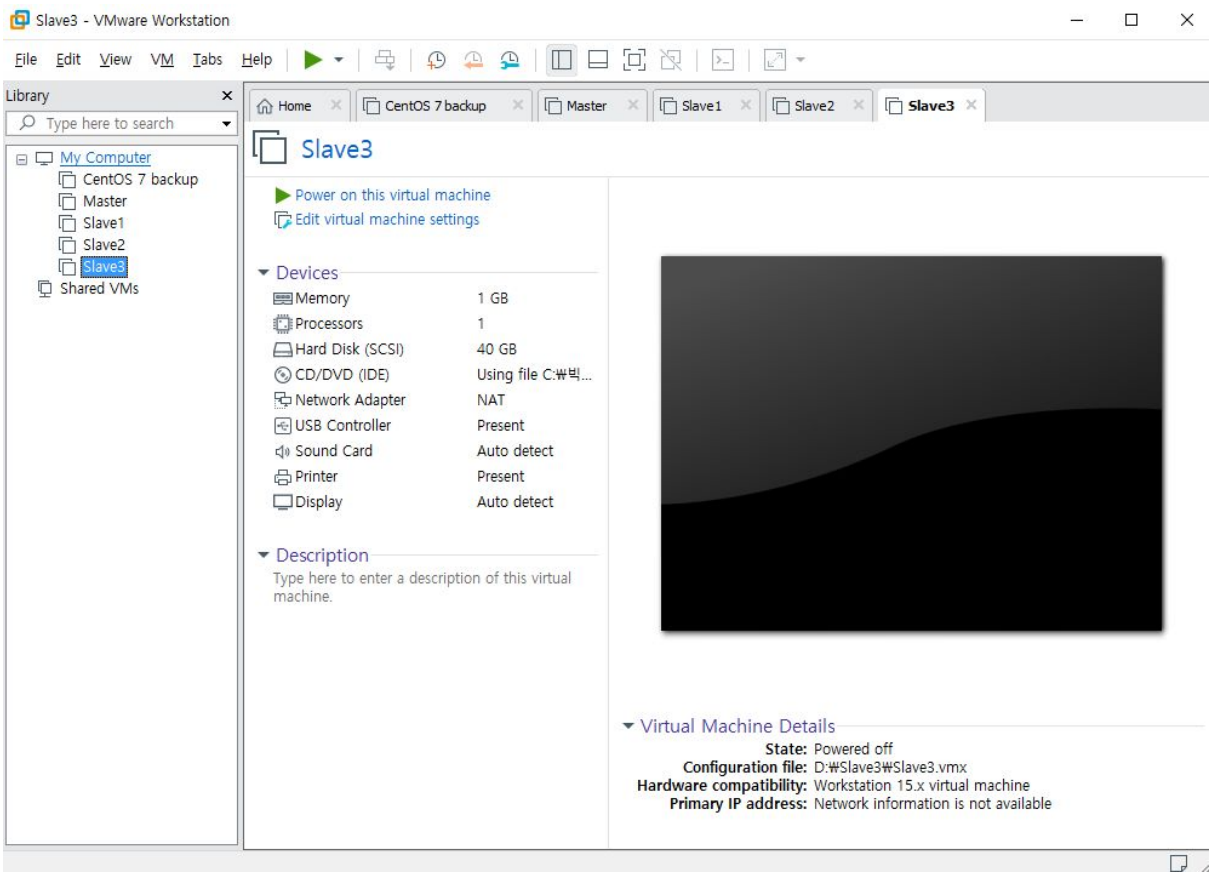
d:₩centos₩backup
d:₩centos₩master
d:₩centos₩slave1
d:₩centos₩slave2
d:₩centos₩slave3

VMplayer버전 같은 경우는

3) VMware Workstation(정식버전)의 가상머신 복제 방법

1대를 설치한 후 vmware에서 clone 복제
Manage – Clone





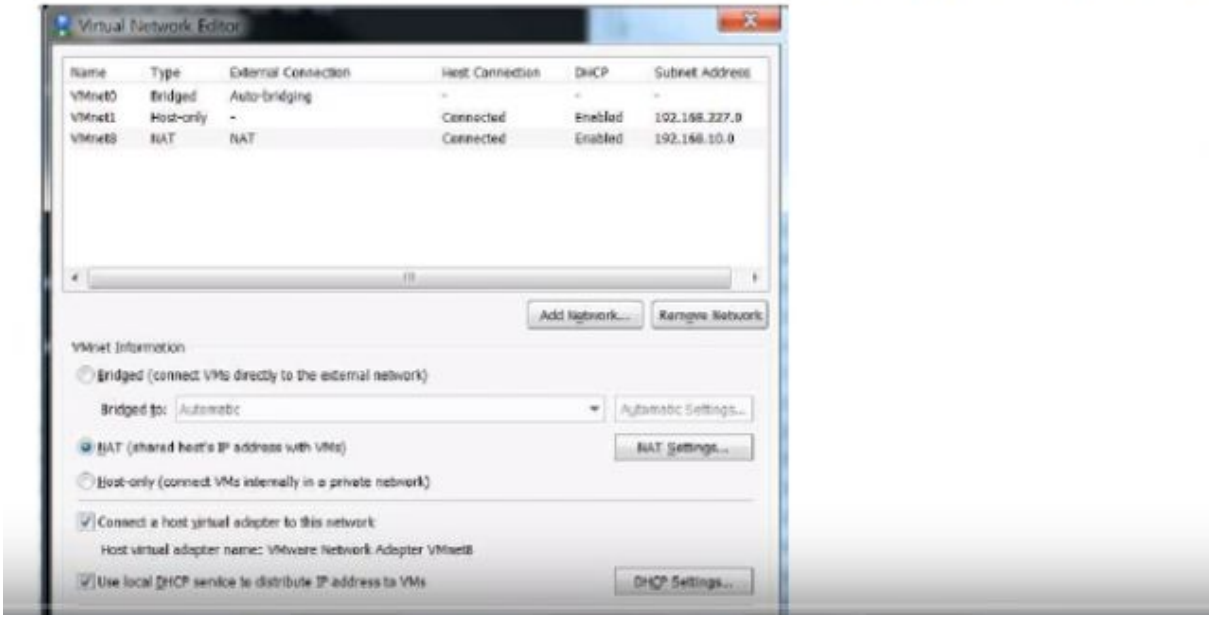
Master를 통해서 slave1,2,3 clone생성한다.

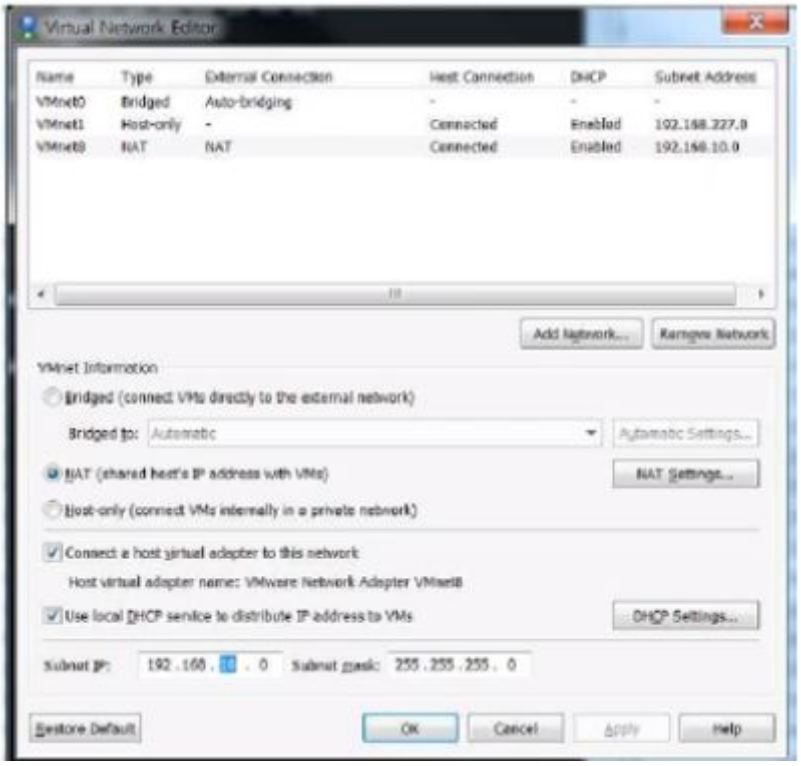
4) 네트워크 설정 작업

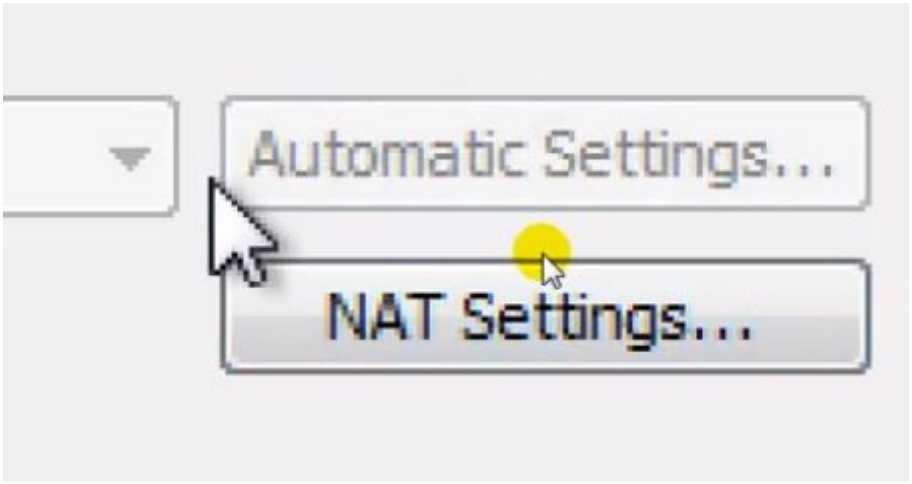
VMware Workstation Pro 버전에서 실행

Edit – Virtual Network Editor

VMnet8을 선택한 후 아래쪽의 Subnet IP에서 3번째 자리를 10으로 변경

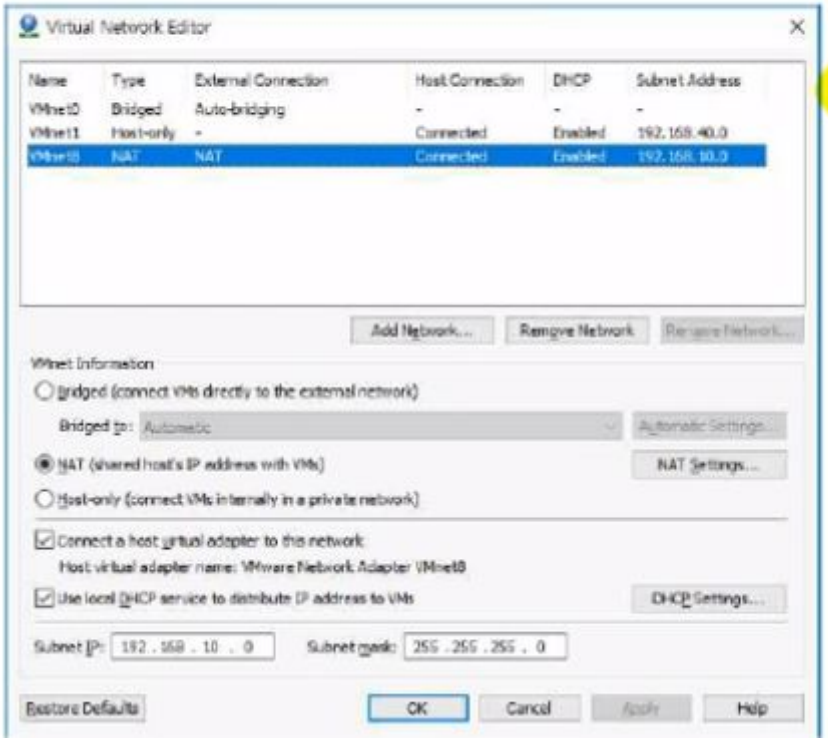






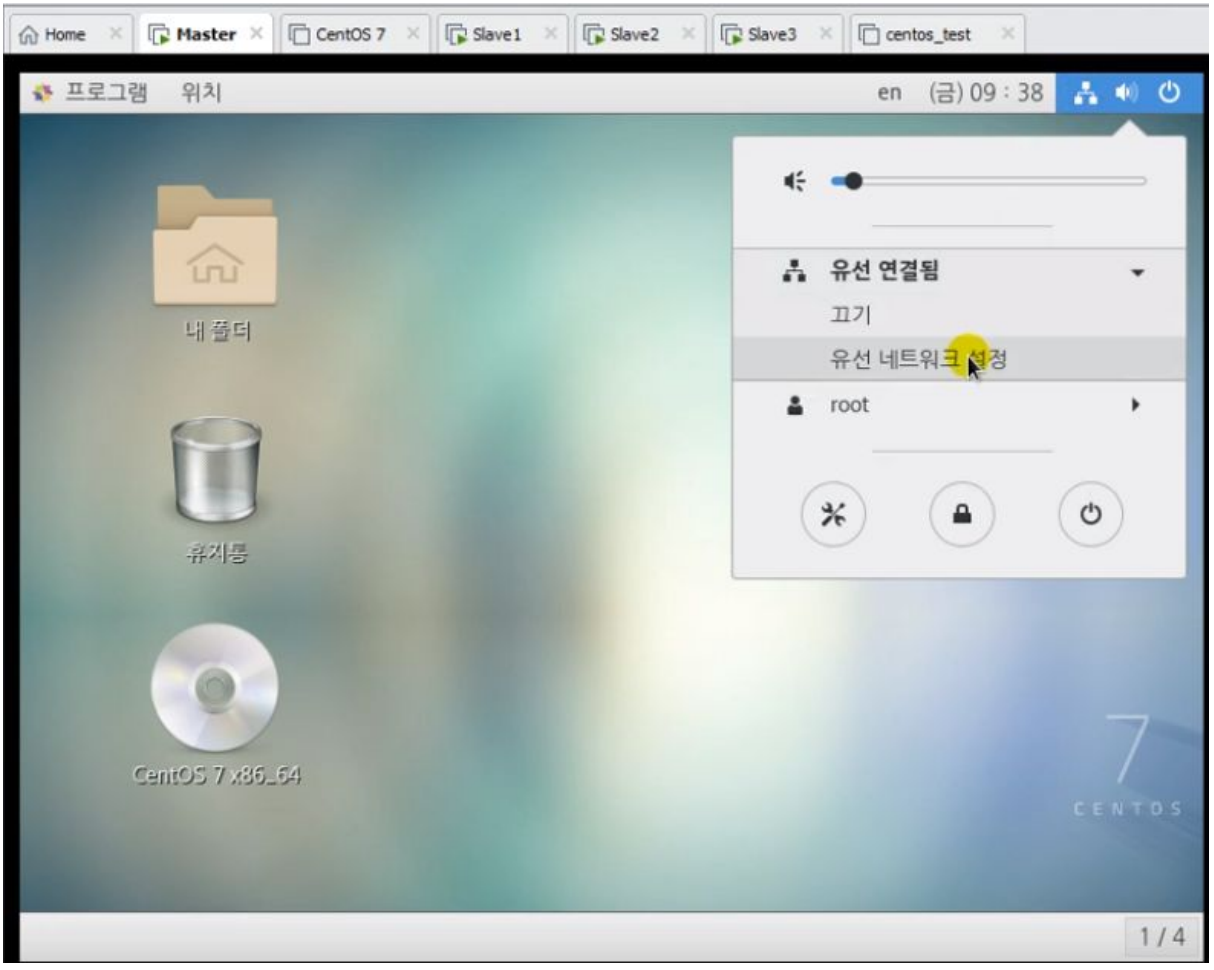
NAT Settings 버튼 클릭

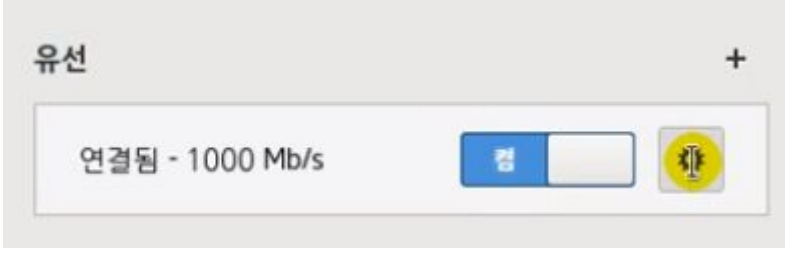
Gateway IP를 192.168.10.2에서 192.168.10.254로 변경

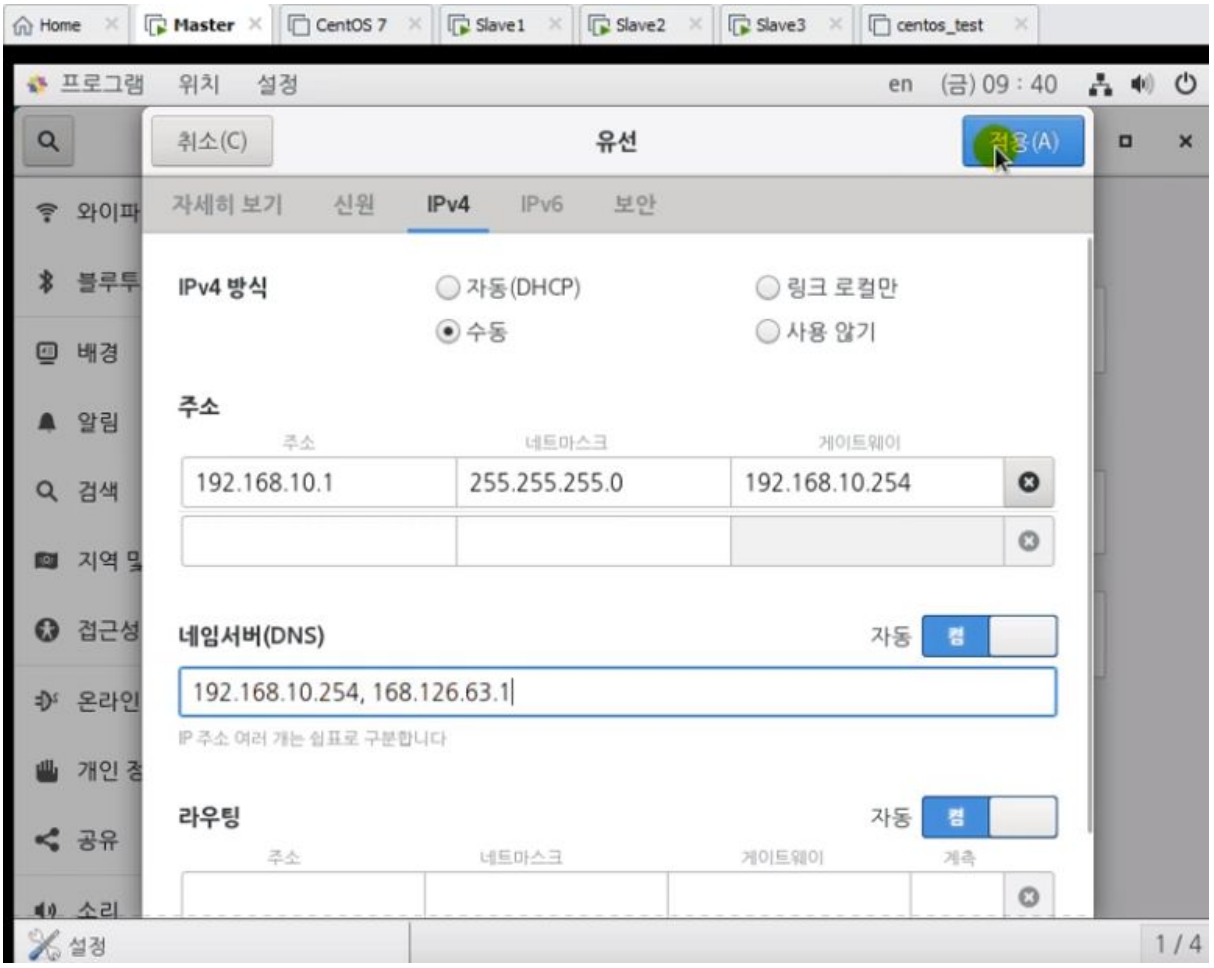


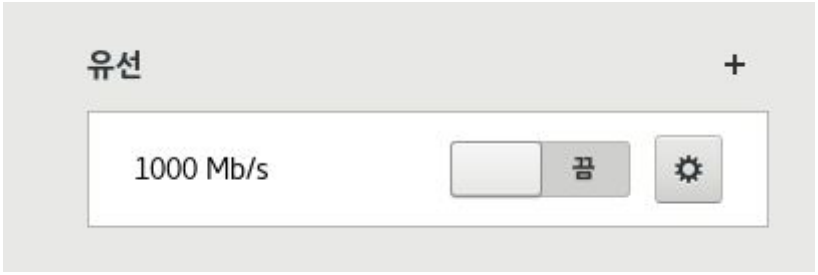
가상 머신 4대를 모두 구동시킨 후 각각 ip를 변경한다.

ip주소
master 192.168.10.1
slave1 192.168.10.2
slave2 192.168.10.3
slave3 192.168.10.4
네트마스크 : 255.255.255.0
게이트웨이 : 192.168.10.254
네임서버 : 192.168.10.254
168.126.63.1





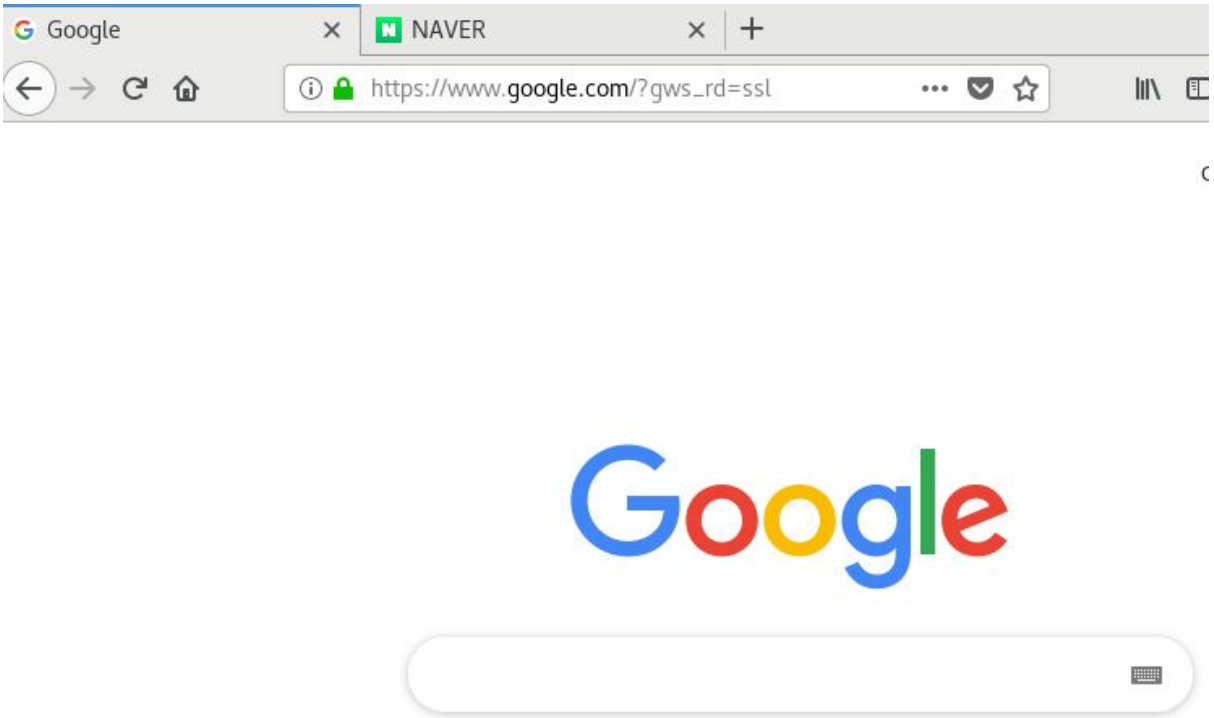




껏다가 켜야 적용된다.

ifconfig, firefox접속을 통해 확인.

```
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.1  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::fc2d:5100:862f:1da3  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:2a:5a:1d  txqueuelen 1000  (Ethernet)
        RX packets 2890  bytes 3342193 (3.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2304  bytes 224941 (219.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



slave1,2,3까지 모두 설정

5) 호스트 설정(모든 노드에서 실행)

```
gedit /etc/hosts
```

```
127.0.0.1 localhost
192.168.10.1 master
192.168.10.2 slave1
192.168.10.3 slave2
192.168.10.4 slave3
```

hosts와 hostname을 일치시키는 작업
master에서 실행

```
gedit /etc/hostname
```

```
master
```

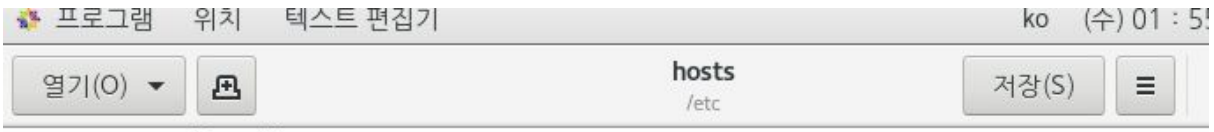
slave1에서 실행

```
gedit /etc/hostname
```

4대 모두 실행

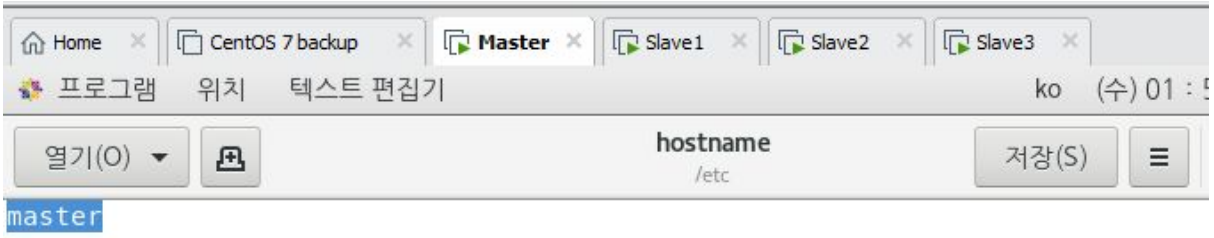
gedit /etc/hostname

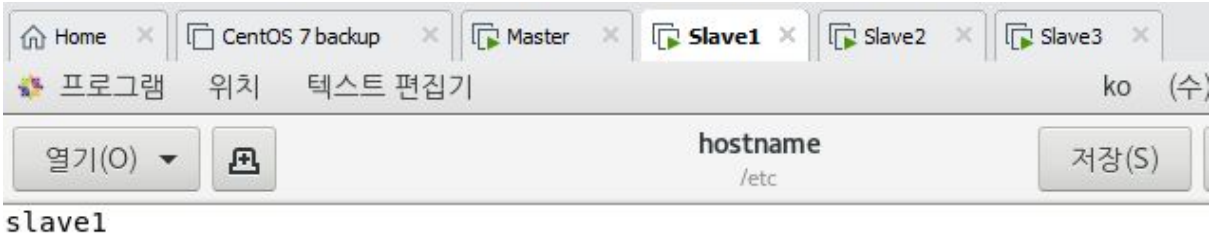
```
[root@localhost ~]# gedit /etc/hosts
```

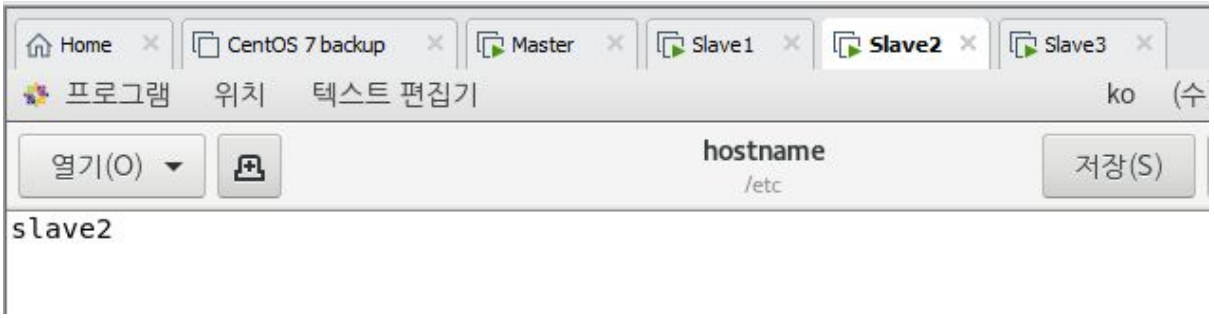


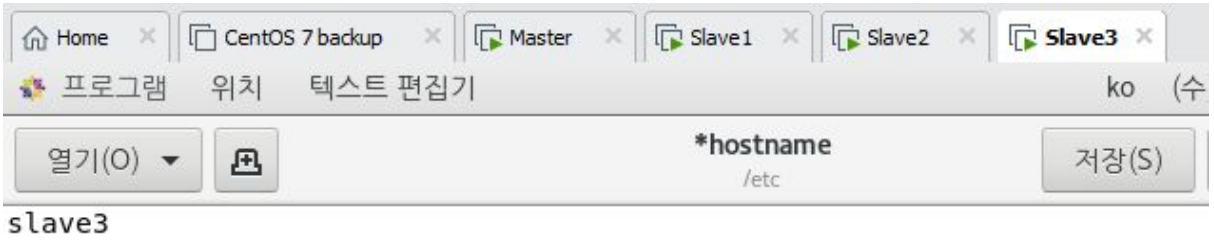

```
127.0.0.1   localhost
192.168.10.1 master
192.168.10.2 slave1
192.168.10.3 slave2
192.168.10.4 slave3
```

```
[root@localhost ~]# gedit /etc/hostname
```

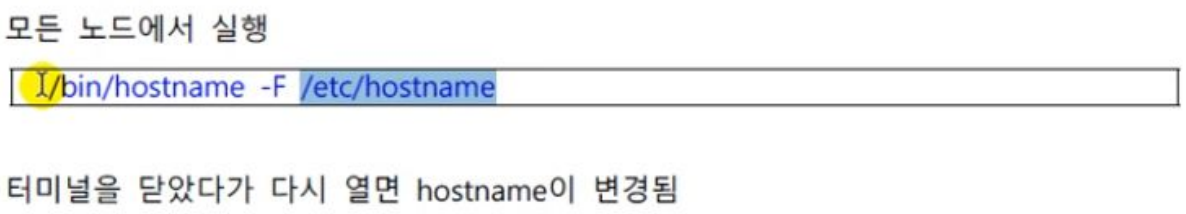




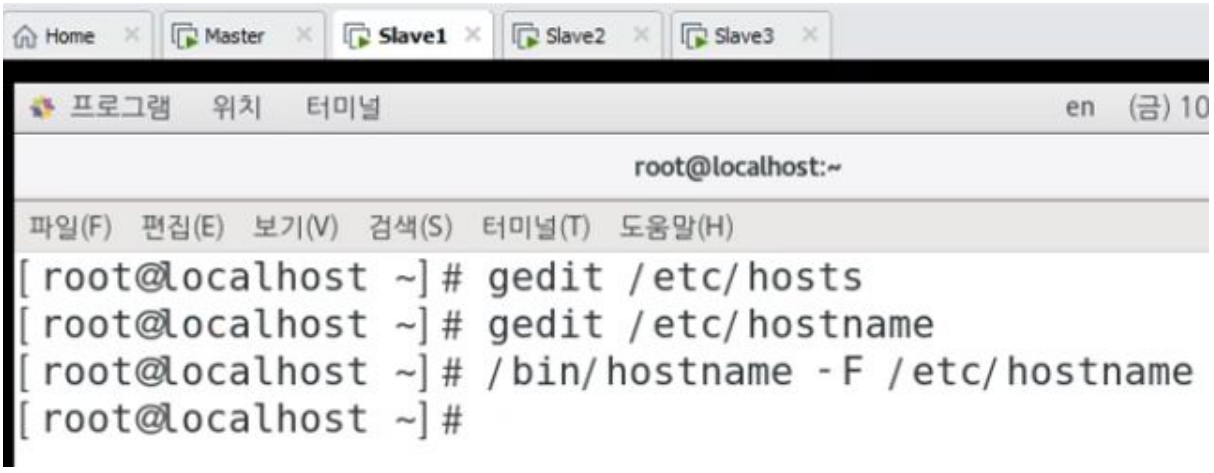




자, 이제 모든 hostname변경된 사항 적용



/bin/hostname -F /etc/hostname



재부팅(모든 노드에서 실행)

```
reboot
```

재부팅 후 hostname이 변경되었는지 확인(모든 노드)

```
hostname
```

ping 테스트

master에서 실행

```
ping slave1
ping slave2
ping slave3
```

slave1에서 실행

```
ping master
ping slave2
ping slave3
```

slave2에서 실행

```
ping master
ping slave1
ping slave3
```

slave3에서 실행

```
ping master
ping slave1
```

```
64 bytes from slave1 (192.168.10.2): icmp_seq=158 ttl=64 time=0.347 ms
64 bytes from slave1 (192.168.10.2): icmp_seq=159 ttl=64 time=0.414 ms
64 bytes from slave1 (192.168.10.2): icmp_seq=160 ttl=64 time=0.391 ms
64 bytes from slave1 (192.168.10.2): icmp_seq=161 ttl=64 time=0.357 ms
64 bytes from slave1 (192.168.10.2): icmp_seq=162 ttl=64 time=0.378 ms
64 bytes from slave1 (192.168.10.2): icmp_seq=163 ttl=64 time=0.484 ms
^C
--- slave1 ping statistics ---
163 packets transmitted, 163 received, 0% packet loss, time 162237ms
rtt min/avg/max/mdev = 0.320/0.900/6.937/0.708 ms
[root@master ~]#
```

Ctrl+C해야 ping테스트가 끝난다.
다음과 같이 했을때, loss가 없다는 것은 잘 연결이 되어있다는 뜻이다.

6) SSH 공개키 복사

로컬 서버 → 원격 서버로 파일 전송

scp [옵션] [원본 경로 및 파일] [계정명]@[원격지IP주소]:[전송할 경로]

scp /home/me/wow.html abc@111.222.333.444:/home/abc/

- master에서 생성한 공개 키를 모든 datanode로 복사(master에서 실행)

```
scp -rp ~/.ssh/authorized_keys root@slave1:~/.ssh/authorized_keys
scp -rp ~/.ssh/authorized_keys root@slave2:~/.ssh/authorized_keys
scp -rp ~/.ssh/authorized_keys root@slave3:~/.ssh/authorized_keys
```

하둡은 분산시스템을 기반으로 작동한다.
즉, 원격 접속을 통해서 데이터를 주고받아야한다.
따라서, 보안 접속이 필요하고 이를 SSH를 사용한다.
이는, 지난 시간에 배웠던 내용이다.

그렇다면, SSH는 공개키와 개인키를 통해서 인증절차를 거치는데,
따라서 공개키를 모두 나눠줘야한다.

모든 노드가 다 키를 주고받아야만 서로 ssh통신이 가능하다.

scp란, 로컬서버와 원격서버가 키를 주고받는 방법이다.

따라서 scp명령어를 통해서 namenode에서 datanode로 뿌려줄것이다.

scp -rp ~/.ssh/authorized_keys root@slave1:~/.ssh/authorized_keys

```
[root@master ~]# scp -rp ~/.ssh/authorized_keys root@slave1:~/.ssh/authorized_keys
The authenticity of host 'slave1 (192.168.10.2)' can't be established.
ECDSA key fingerprint is SHA256:zwUDu9GV9KWYKKTw9X9vXrHLKxD83STLj7YctD8vNLI.
ECDSA key fingerprint is MD5:44:0b:e0:1e:88:f2:ff:e8:39:8e:a5:32:6b:96:52:5f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave1,192.168.10.2' (ECDSA) to the list of known hosts.
authorized_keys                                   100%  408   193.1KB/s   00:00
[root@master ~]# scp -rp ~/.ssh/authorized_keys root@slave2:~/.ssh/authorized_keys
The authenticity of host 'slave2 (192.168.10.3)' can't be established.
ECDSA key fingerprint is SHA256:zwUDu9GV9KWYKKTw9X9vXrHLKxD83STLj7YctD8vNLI.
ECDSA key fingerprint is MD5:44:0b:e0:1e:88:f2:ff:e8:39:8e:a5:32:6b:96:52:5f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave2,192.168.10.3' (ECDSA) to the list of known hosts.
authorized_keys                                   100%  408   310.1KB/s   00:00
[root@master ~]# scp -rp ~/.ssh/authorized_keys root@slave3:~/.ssh/authorized_keys
The authenticity of host 'slave3 (192.168.10.4)' can't be established.
ECDSA key fingerprint is SHA256:zwUDu9GV9KWYKKTw9X9vXrHLKxD83STLj7YctD8vNLI.
ECDSA key fingerprint is MD5:44:0b:e0:1e:88:f2:ff:e8:39:8e:a5:32:6b:96:52:5f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave3,192.168.10.4' (ECDSA) to the list of known hosts.
authorized_keys                                   100%  408   247.0KB/s   00:00
[root@master ~]#
```

```
[root@master ~]# ssh slave1
Last login: Wed Jun  5 02:04:52 2019
[root@slave1 ~]# logout
Connection to slave1 closed.
[root@master ~]# ssh slave2
Last login: Wed Jun  5 02:04:58 2019
[root@slave2 ~]# exit
logout
Connection to slave2 closed.
[root@master ~]# ssh slave3
Last login: Wed Jun  5 02:05:04 2019
[root@slave3 ~]# exit
logout
Connection to slave3 closed.
[root@master ~]#
```

확인 절차

slave 1,2,3도 나머지 서버들에게 이러한 setting을 해줘야한다.
즉, 서로가 키를 주고 받아야한다.
반복 작업 실행!

---

8) hadoop-env.sh 수정(master에서 실행)

```
gedit $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

텍스트 편집기의 줄 번호가 표시되도록 설정(텍스트 편집기 – 기본 설정 – 줄 번호 표시 체크)
25번 라인 JDK 경로 수정 : export JAVA_HOME=/usr/local/jdk1.8
113번 라인 하둡 데몬의 pid 저장 경로 수정
export HADOOP_PID_DIR=/home/centos/hadoop-2.9.2/pids

9) core-site.xml 수정(모든 노드에서 실행)

```
gedit $HADOOP_HOME/etc/hadoop/core-site.xml
```

localhost를 master로 수정

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
```

name노드의 ip주소를 입력하는 구간이다. 우리가 먼저 도메인 네임을 master로 지정해두었기 때문에 이렇게 간편하게 설정이 가능하다.

10) hdfs-site.xml 수정

$HADOOP_HOME 하위에 namenode와 datanode 디렉토리를 만든다.(master)

```
rm -rf $HADOOP_HOME/namenode
mkdir $HADOOP_HOME/namenode
chown root -R $HADOOP_HOME/namenode
chmod 777 $HADOOP_HOME/namenode

rm -rf $HADOOP_HOME/datanode
mkdir $HADOOP_HOME/datanode
chown root -R $HADOOP_HOME/datanode
chmod 777 $HADOOP_HOME/datanode
```

$HADOOP_HOME 하위에 datanode 디렉토리를 만든다.(slave1, slave2, slave3)

```
rm -rf $HADOOP_HOME/datanode
mkdir $HADOOP_HOME/datanode
chown root -R $HADOOP_HOME/datanode
chmod 777 $HADOOP_HOME/datanode
```

hdfs-site.xml 수정(master)

```
gedit $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>2</value> <!-- 2로 변경 -->
    </property>
    <property>
      <name>dfs.permissions</name>
      <value>false</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/home/centos/hadoop-2.9.2/namenode</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/home/centos/hadoop-2.9.2/datanode</value>
    </property>
```

복제하는 파일 갯수를 의미하는 value를 2로 증가시킨다.

그리고 namenode디렉토리와 datanode디렉토리를 지정하게 되어있는데, 이 작업은 directory를 형성하고 진행해야한다.

우리는 분산처리를 할 컴퓨터 대수가 작기 때문에, master가 namenode의 역할과 datanode의 역할을 둘다 하게끔 설정하려고한다.
물론, master는 namenode의 역할만하는 것이 원래는 더 효율적이다.

```
[root@master ~]# rm -rf $HADOOP_HOME/datanode
[root@master ~]# mkdir $HADOOP_HOME/datanode
[root@master ~]# chown root -R $HADOOP_HOME/datanode
[root@master ~]# chmod 777 $HADOOP_HOME/datanode
```

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
        </property>
        <property>
                <name>dfs.permissions</name>
                <value>false</value>
        </property>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>file:/home/centos/hadoop-2.9.2/namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>file:/home/centos/hadoop-2.9.2/datanode</value>
        </property>
</configuration>
```

hdfs-site.xml 수정(slave1,slave2,slave3)

```
gedit $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>2</value> <!-- 2로 변경 -->
    </property>
    <property>
      <name>dfs.permissions</name>
      <value>false</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/home/centos/hadoop-2.9.2/datanode</value>
    </property>
</configuration>
```

gedit $HADOOP_HOME/etc/hadoop/hdfs-site.xml

<configuration>
	<property>
		<name>dfs.replication</name>
		<value>2</value>
	</property>
	<property>
		<name>dfs.permissions</name>
		<value>false</value>
	</property>
	<property>
		<name>dfs.datanode.data.dir</name>
		<value>file:/home/centos/hadoop-2.9.2/datanode</value>
	</property>
</configuration>

11) 잡트래커 설정(모든 노드)

mapred-site.xml.template 파일을 복사하여 mapred-site.xml 만들기

```
cp $HADOOP_HOME/etc/hadoop/mapred-site.xml.template
$HADOOP_HOME/etc/hadoop/mapred-site.xml
```

mapred-site.xml 편집

```
gedit $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

yarn-site.xml 편집

```
gedit $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

map은 분산시키는 것 reduce는 다시 통합시는 작업

cp $HADOOP_HOME/etc/hadoop/mapred-site.xml.template
$HADOOP_HOME/etc/hadoop/mapred-site.xml

gedit $HADOOP_HOME/etc/hadoop/mapred-site.xml

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

yarn-site.xml 편집

```
gedit $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

```
<?xml version="1.0"?>
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

gedit $HADOOP_HOME/etc/hadoop/yarn-site.xml

```
<configuration>

<!-- Site specific YARN configuration properties -->
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
                <value>org.apache.hadoop.mapred.ShuffleHandler</value>
        </property>
</configuration>
```

12) masters , slaves 파일 편집(master에서만 작업)

```
gedit $HADOOP_HOME/etc/hadoop/masters
master
```

```
gedit $HADOOP_HOME/etc/hadoop/slaves
master
slave1
slave2
slave3
```

13) 네임노드 포맷(master에서만 실행), 하둡 가동, HDFS 폴더 생성

```
$HADOOP_HOME/bin/hdfs namenode -format
```

14) 방화벽을 내림(모든 노드)

```
systemctl stop firewalld.service
systemctl disable firewalld.service
```

gedit $HADOOP_HOME/etc/hadoop/masters

gedit $HADOOP_HOME/etc/hadoop/slaves

$HADOOP_HOME/bin/hdfs namenode -format

[root@master ~]# systemctl stop firewalld.service
[root@master ~]# systemctl disable firewalld.service

15) DFS, Yarn 시작(master에서만 실행)

start-dfs.sh (NameNode, SecondaryNameNode, DataNode가 실행됨)

start-yarn.sh (master에서는 ResourceManager와 NodeManager가 실행되고, slave에서는 NodeManager가 실행됨)

```
start-dfs.sh
start-yarn.sh
```

프로세스 확인

```
jps
```

```
2883 ResourceManager
2725 SecondaryNameNode
2502 NameNode
8712 Jps
```

slave1, slave2, slave3에서도 jps를 실행하여 DataNode가 구동되는지 확인한다.

master	slave1	slave2	slave3
[root@master ~]# jps 14082 DataNode 14614 SecondaryNameNode 27398 ResourceManager 27638 NodeManager 27801 Jps 13598 NameNode [root@master ~]#	[root@slave1 ~]# jps 65424 DataNode 65540 NodeManager 65679 Jps [root@slave1 ~]#	[root@slave2 ~]# jps 65397 DataNode 65654 Jps 65514 NodeManager [root@slave2 ~]#	[root@slave3 ~]# jps 65491 NodeManager 65374 DataNode 65630 Jps [root@slave3 ~]#

16) 데이터노드가 올라오지 않을 경우

(slave1,slave2,slave3)

```
rm -rf $HADOOP_HOME/datanode
mkdir $HADOOP_HOME/datanode
chown root -R $HADOOP_HOME/datanode
chmod 777 $HADOOP_HOME/datanode
```

(master)

```
stop-dfs.sh
stop-yarn.sh

rm -rf $HADOOP_HOME/namenode
mkdir $HADOOP_HOME/namenode
chown root -R $HADOOP_HOME/namenode
chmod 777 $HADOOP_HOME/namenode

rm -rf $HADOOP_HOME/datanode
mkdir $HADOOP_HOME/datanode
chown root -R $HADOOP_HOME/datanode
chmod 777 $HADOOP_HOME/datanode

hdfs namenode -format
start-dfs.sh
start-yarn.sh
```

17) 하둡2 웹 인터페이스 확인 (master에서 실행)

http://master:50070

18) yarn 웹 인터페이스(master에서 실행)

http://master:8088

만약에 ip로 한다면 “http://192.168.10.1:50070”(master의 ip주소)이다.

DFS Used:	16 KB (0%)
Non DFS Used:	24.27 GB
DFS Remaining:	111.65 GB (82.15%)
Block Pool Used:	16 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	4 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0

live node클릭시,

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✔master:50010 (192.168.10.1:50010)	http://master:50075	2s	7m	33.98 GB	0	4 KB (0%)	2.9.2
✔slave1:50010 (192.168.10.2:50010)	http://slave1:50075	1s	7m	33.98 GB	0	4 KB (0%)	2.9.2
✔slave2:50010 (192.168.10.3:50010)	http://slave2:50075	1s	7m	33.98 GB	0	4 KB (0%)	2.9.2
✔slave3:50010 (192.168.10.4:50010)	http://slave3:50075	1s	7m	33.98 GB	0	4 KB (0%)	2.9.2

Showing 1 to 4 of 4 entries

Previous 1 Next

19) 분석 프로그램 실행(wordcount)

맵리듀스 job을 실행하기 위해서는 HDFS 디렉토리를 만들어야 함(master에서 실행)

```
hdfs dfs -mkdir /user
hdfs dfs -mkdir /user/root
hdfs dfs -mkdir /user/root/conf
```

테스트용 파일을 HDFS에 업로드

```
hdfs dfs -mkdir /input
hdfs dfs -copyFromLocal /home/centos/hadoop-2.9.2/README.txt /input
hdfs dfs -ls /input
```

wordcount 프로그램 실행

hadoop jar jar파일 클래스 입력파일 출력폴더

```
hadoop jar
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.
jar wordcount /input/README.txt ~/wordcount-output
```

결과 디렉토리 확인

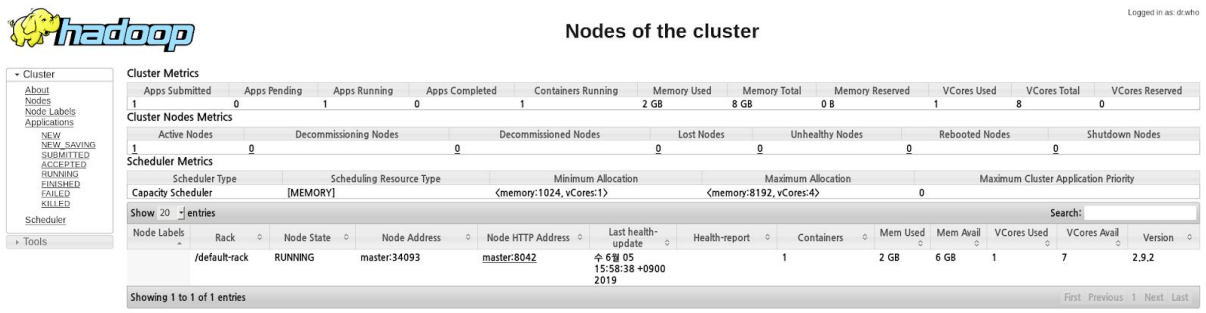
```
hdfs dfs -ls ~/wordcount-output
```

```
Found 2 items
-rw-r--r--   1 root supergroup          0 2018-01-22 23:59
/root/wordcount-output/_SUCCESS
-rw-r--r--   1 root supergroup       1306 2018-01-22 23:59
/root/wordcount-output/part-r-00000
```

실행 결과 확인

```
hdfs dfs -cat ~/wordcount-output/part-r-00000
```

```
(BIS), 1
(ECCN)  1
(TSU)1
```




```
19/06/05 16:03:11 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1836
                FILE: Number of bytes written=400871
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1466
                HDFS: Number of bytes written=1306
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=54645
                Total time spent by all reduces in occupied slots (ms)=71091
                Total time spent by all map tasks (ms)=54645
                Total time spent by all reduce tasks (ms)=71091
                Total vcore-milliseconds taken by all map tasks=54645
                Total vcore-milliseconds taken by all reduce tasks=71091
```

```
        Map-Reduce Framework
                Map input records=31
                Map output records=179
                Map output bytes=2055
                Map output materialized bytes=1836
                Input split bytes=100
                Combine input records=179
                Combine output records=131
                Reduce input groups=131
                Reduce shuffle bytes=1836
                Reduce input records=131
                Reduce output records=131
                Spilled Records=262
                Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
```

```
        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=1366
        File Output Format Counters
                Bytes Written=1306
)5 16:03:12 INFO mapred.ClientServiceDelegate: Application state is completed.
.icationStatus=SUCCEEDED. Redirecting to job history server
)5 16:03:14 INFO ipc.Client: Retrying connect to server: 0.0.0.0/0.0.0.0:10020.
 tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetrie
epTime=1000 MILLISECONDS)
```

등등...

4대의 클러스터 작업을 통해서 데이터가 분석되었다.

```
[root@master ~]# hdfs dfs -ls ~/wordcount-output
Found 2 items
-rw-r--r--   2 root supergroup          0 2019-06-05 16:02 /root/wordcount-output/_SUCC
ESS
-rw-r--r--   2 root supergroup       1306 2019-06-05 16:02 /root/wordcount-output/part-
r-00000
[root@master ~]# hdfs dfs -cat ~/wordcount-output/part-r-00000
(BIS),   1
(ECCN)   1
(TSU)    1
(see     1
5D002.C.1,       1
740.13)  1
<http://www.wassenaar.org/>      1
Administration   1
Apache   1
BEFORE   1
BIS      1
Bureau   1
Commerce,        1
Commodity        1
Control  1
```

실행결과.