Big Data Architecture Building

DONG HO SEO | MIN JAE JIN | KEON HOON LEE | HYUN KYUNG KIM

2018. 11. 23 (Fri)

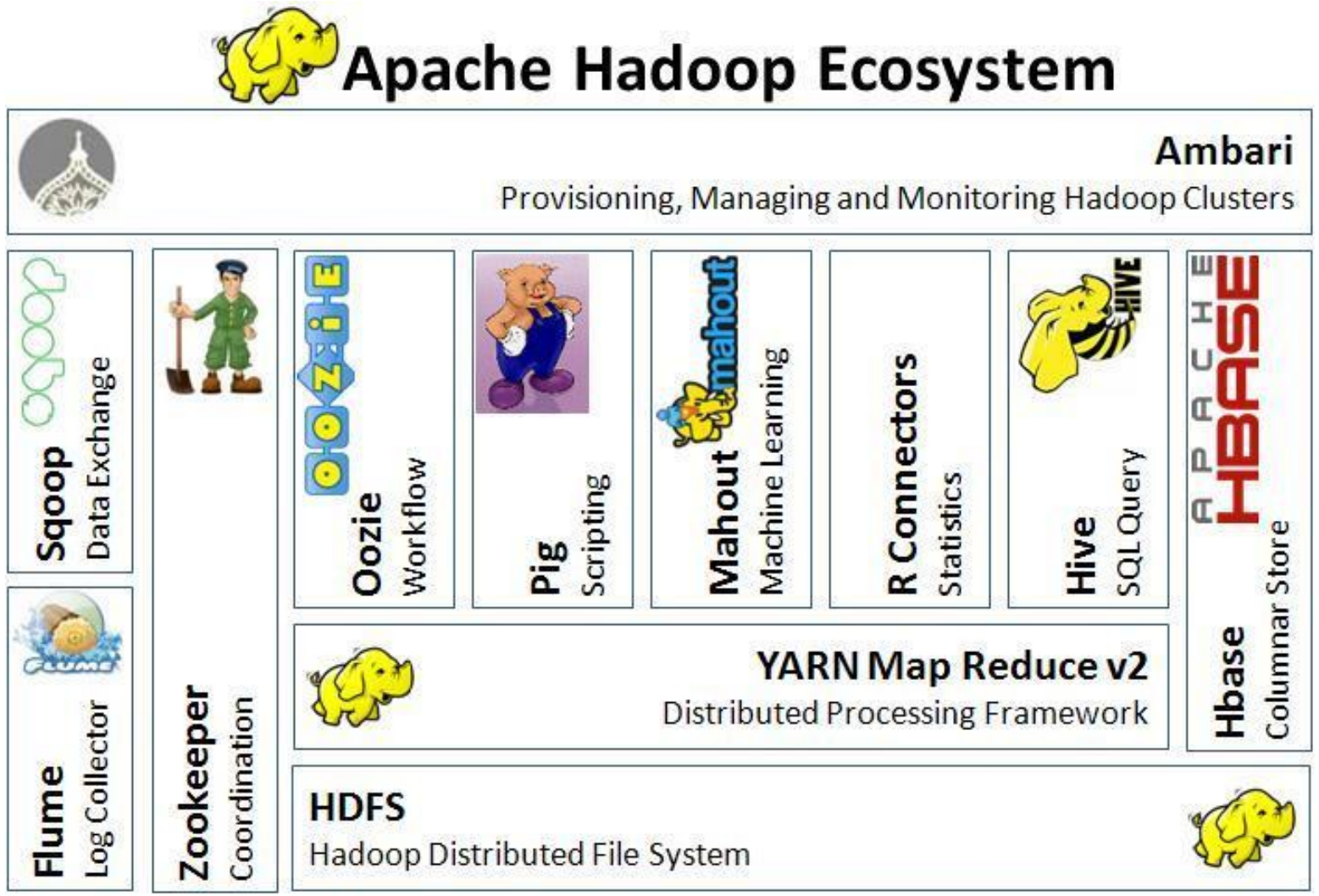
Team4

Contents

HORTONWORKS

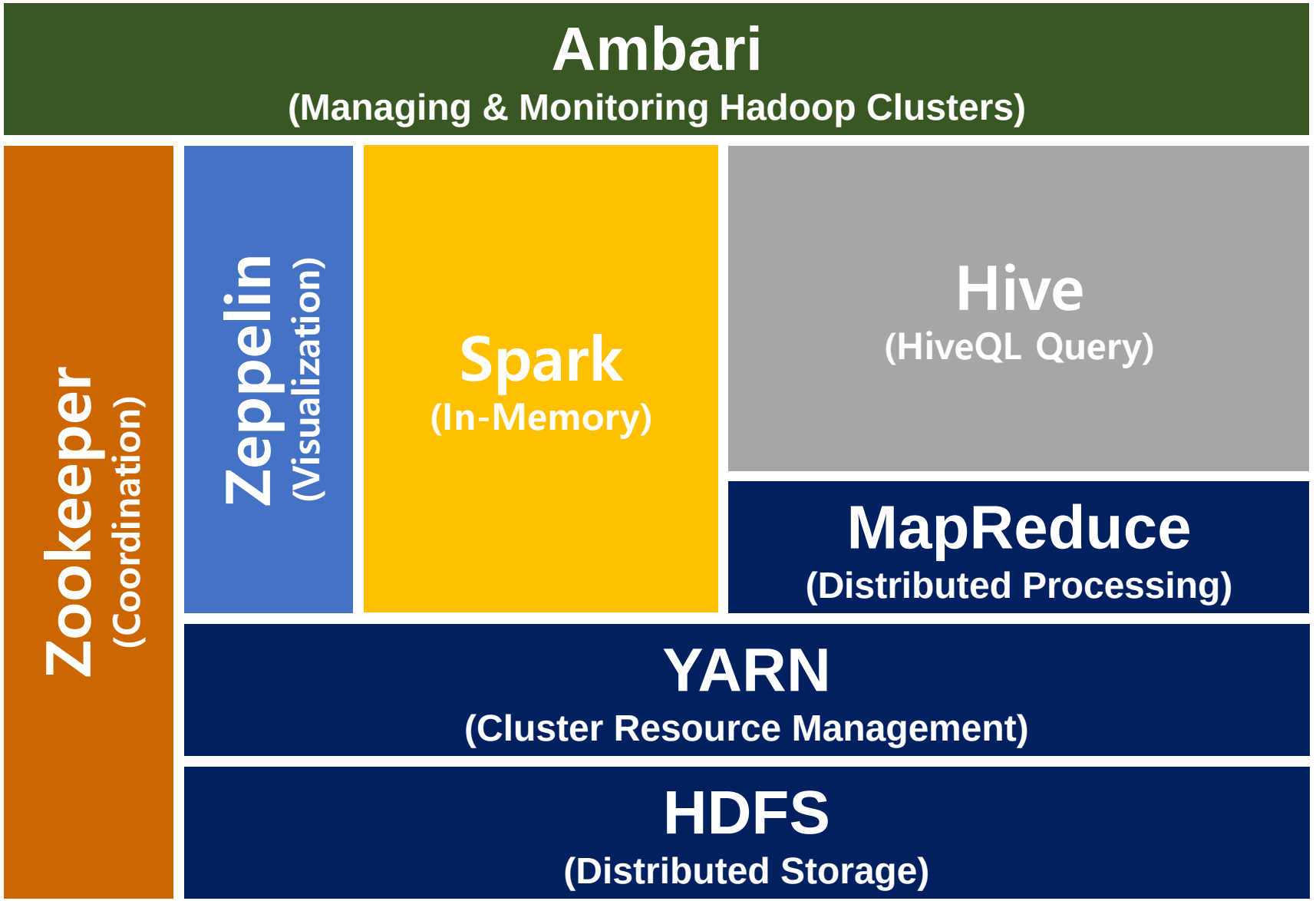
01 | Architecture

1.1 General Architecture

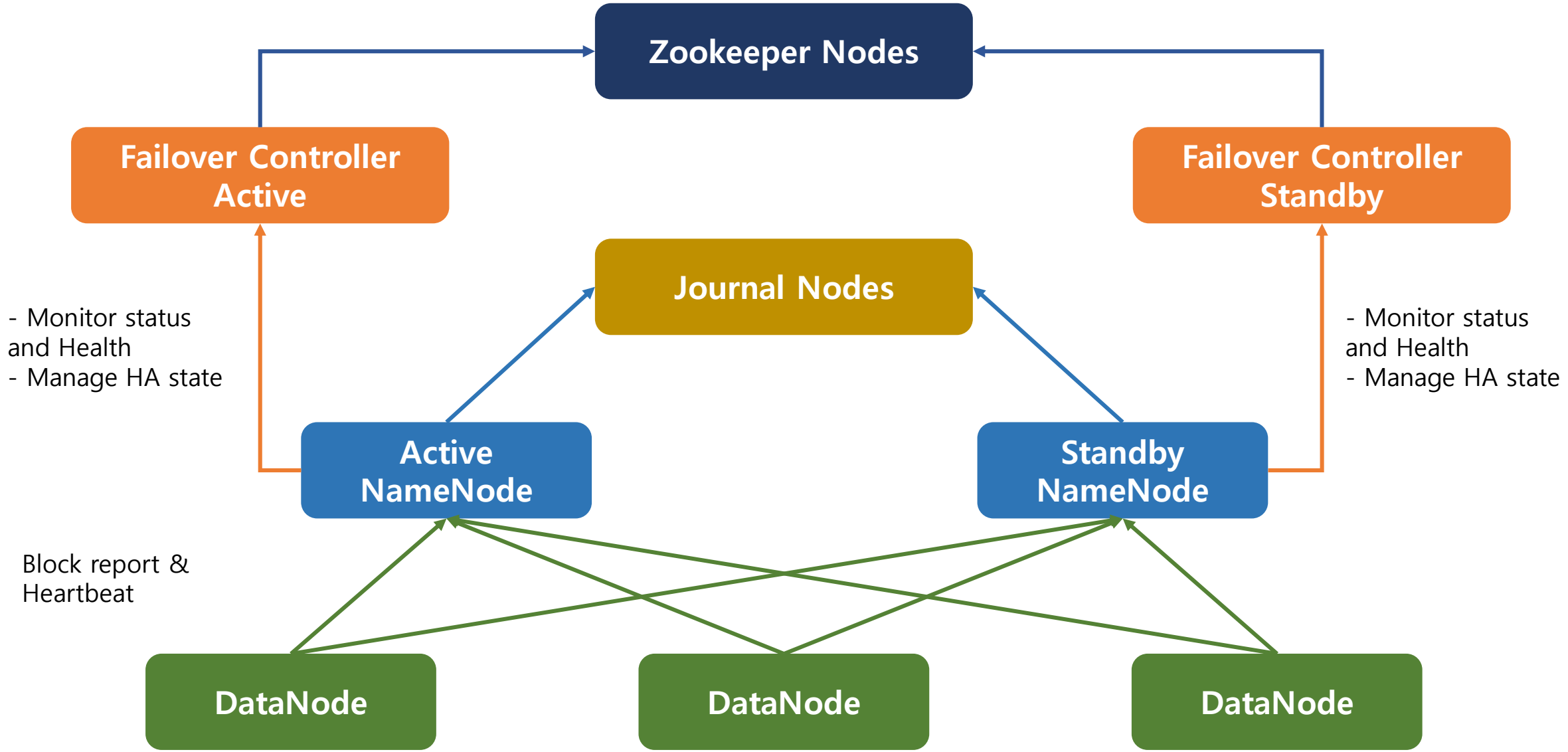


출처 : http://blog.agroknow.com/?p=3810

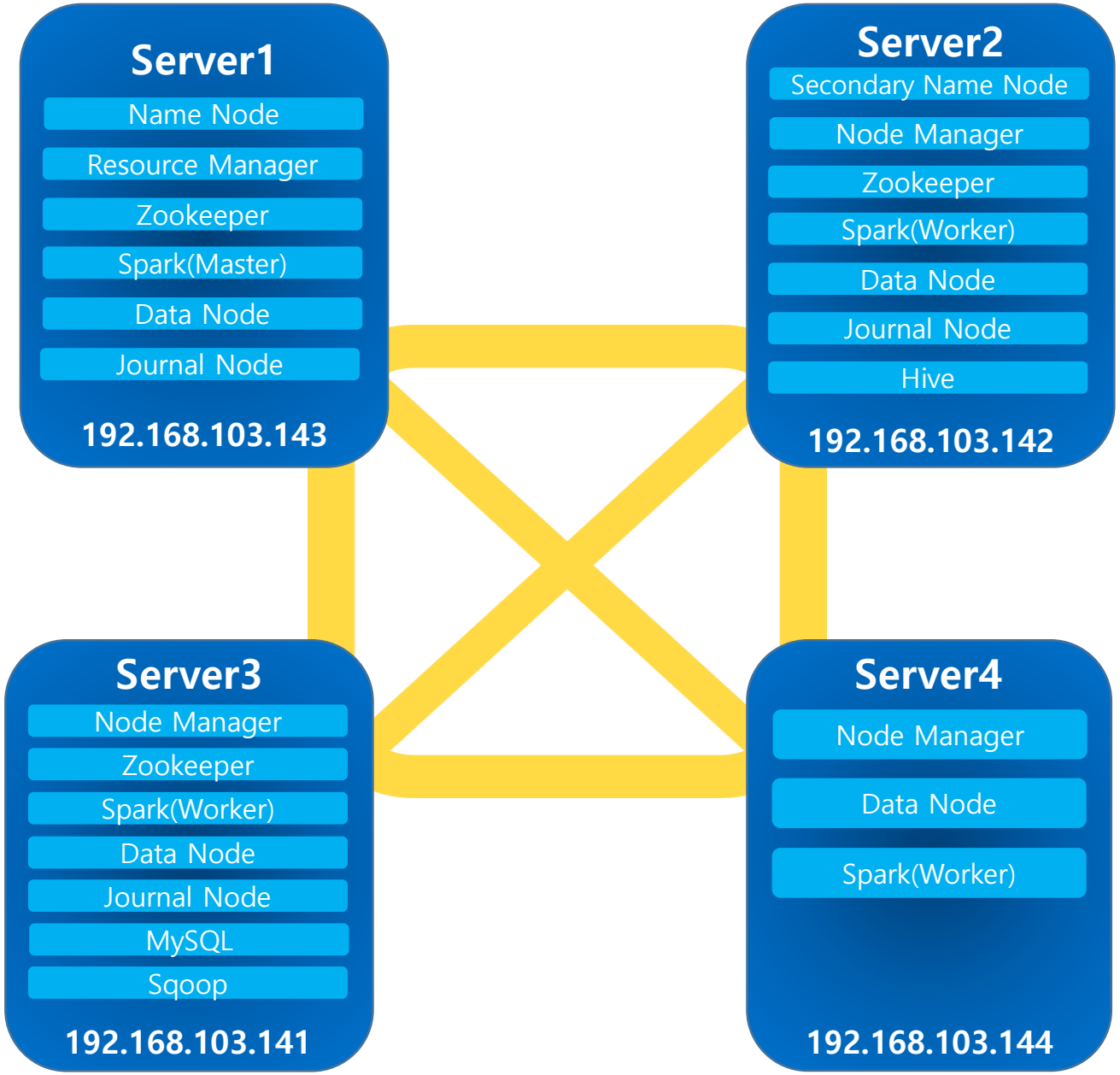
1.2 Our Architecture



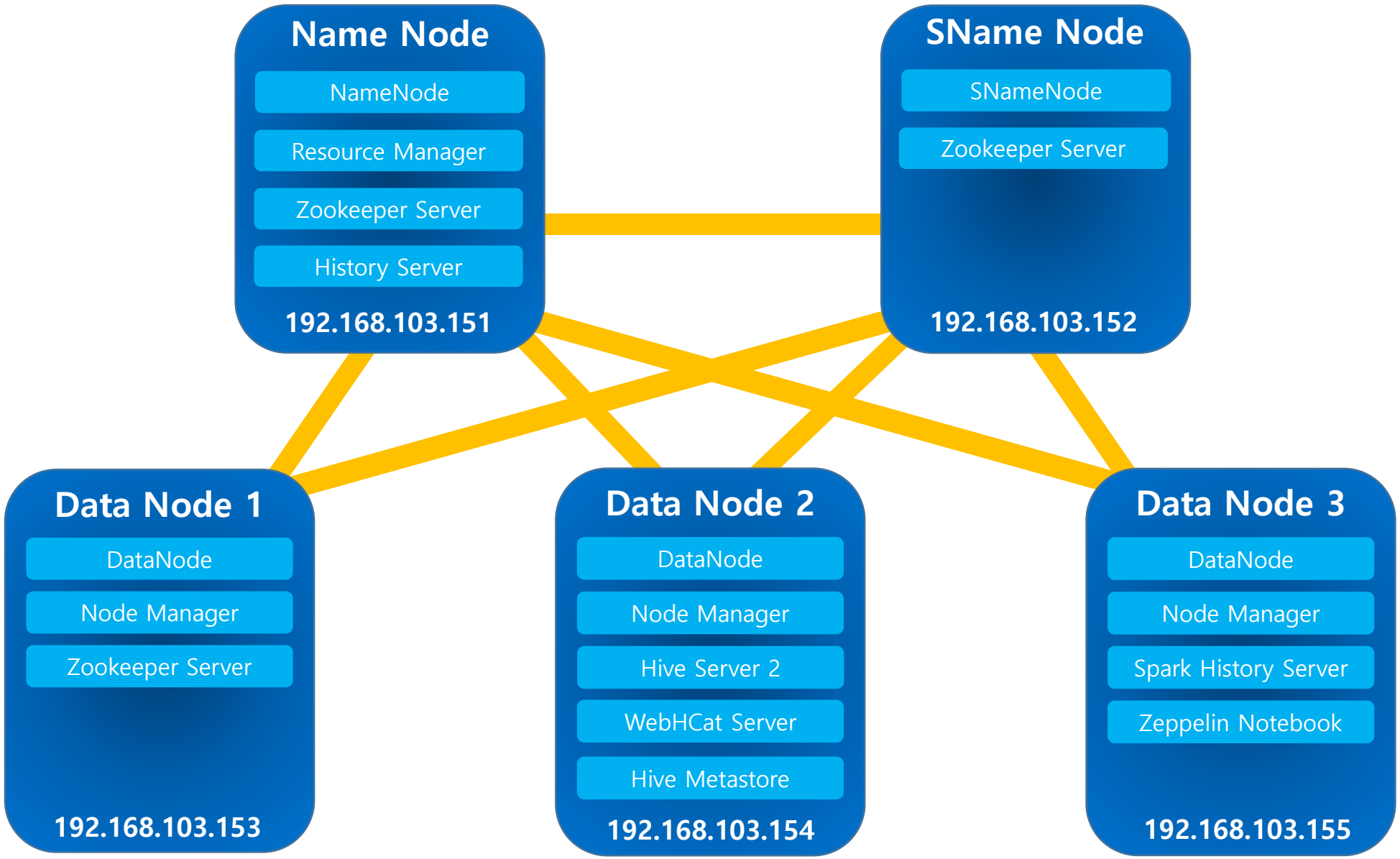
1.3 System Diagram (HDFS High Availability)



1.3.1 System Diagram (HDFS High Availability)



1.4 System Diagram (Ambari)



02 | Requirements

2.1 Test Requirements

3개 이상의 DataNode로 구성된 Cluster를 구성하는 HDFS를 만들 것

- YARN, MR, Zookeeper가 service로 존재
- Data Block Size는 32 MB
- Block Replication은 2
- Namenode와 hive service는 각각 다른 host에서 구동

각 host는 다음 사양 만족

- Host의 Disk Size는 48 ~ 64 GB로 설정
- Host의 CPU Core는 2 ~ 4개로 설정
- Host의 Ram Size는 4.8 GB 이상으로 설정
- HDP 설치 시 SSH private key 입력을 통한 인증 등록

Hive, Spark & Zeppelin notebook 서비스 제공

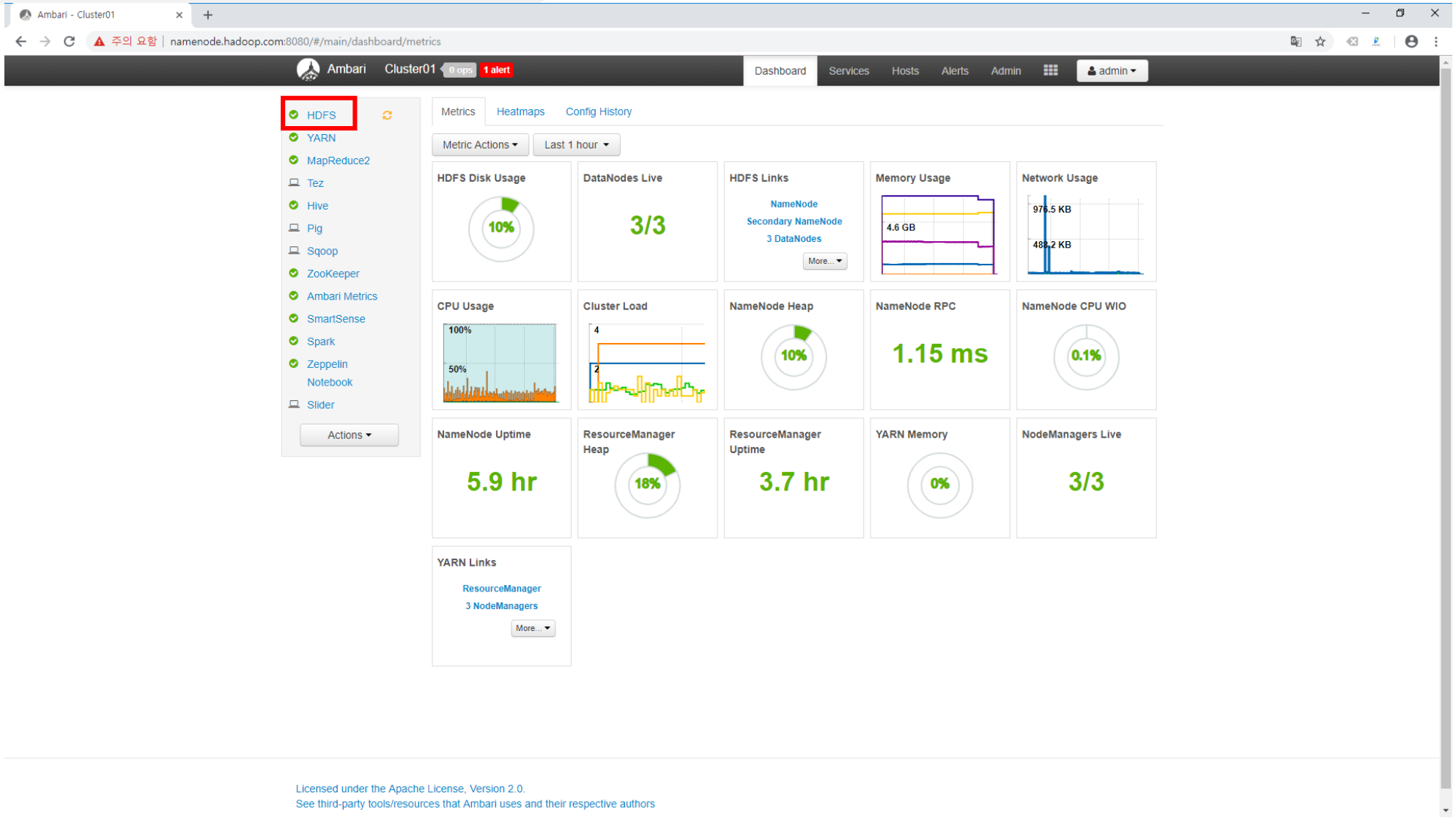
- Hive metastore는 hive server와 동일한 host에 존재
- Hive JDBC 제공
- Data가 존재하는 table 3개 이상 생성

가상 브릿지 네트워크 통신 구성

- Root는 패스워드가 없이 접근 가능
- 클러스터는 외부에서 접근 가능하도록 가상 브릿지 네트워크를 이용한 통신
- 방화벽 서비스는 disable
- 서비스 연결시 host명 return 문제를 해결

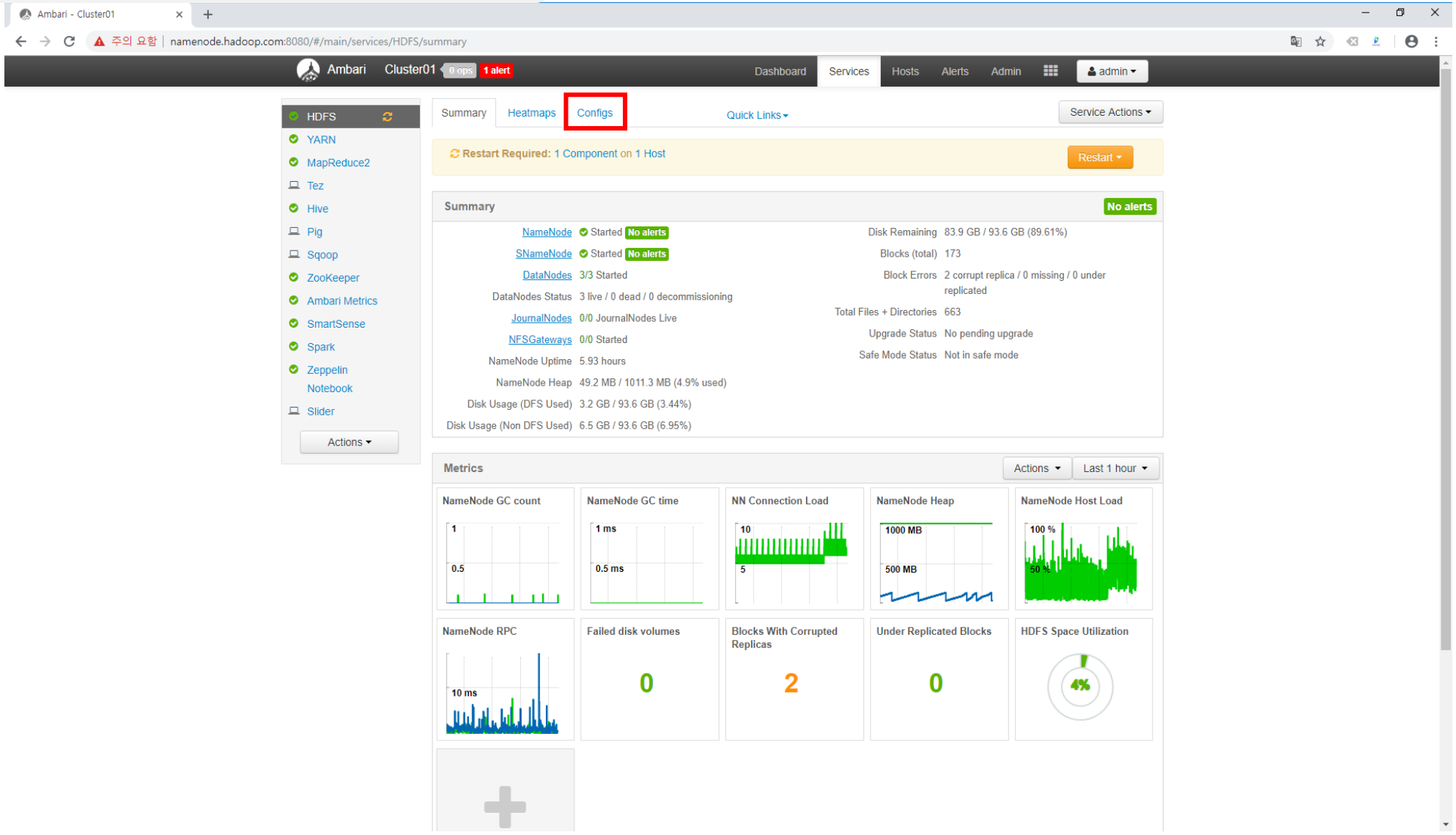
2.2 Requirements in Detail (Ambari)

Data block size 설정 확인



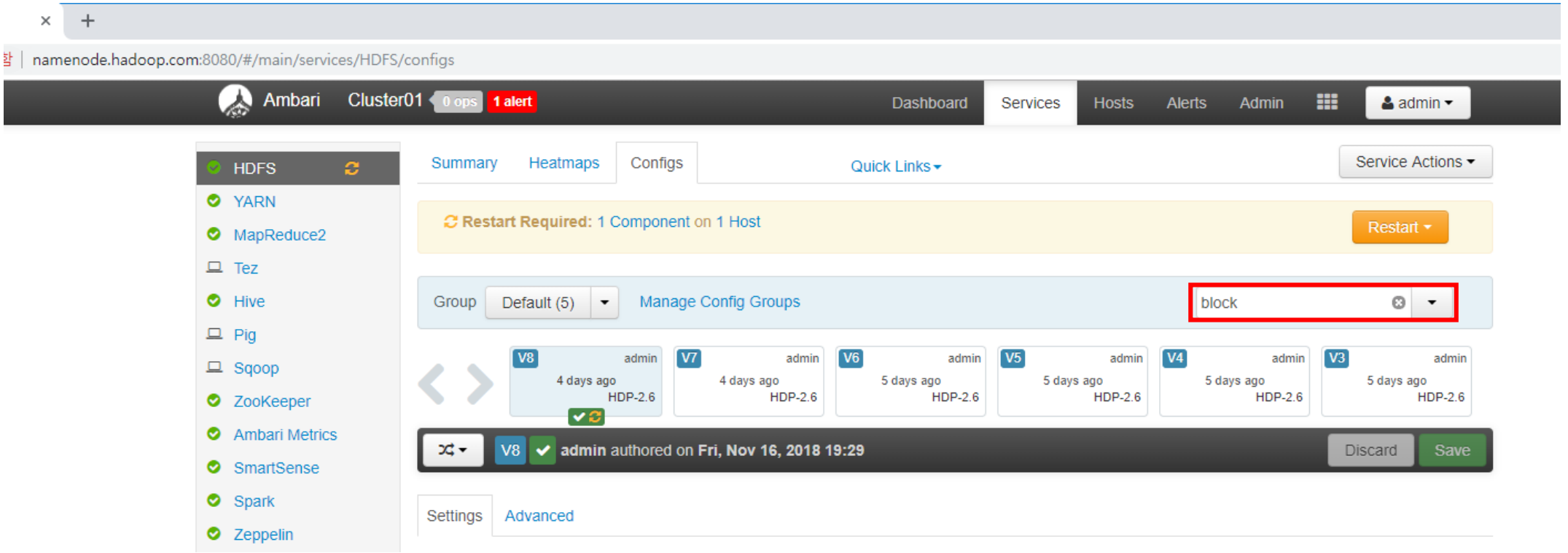
2.2 Requirements in Detail (Ambari)

Data block size 설정 확인



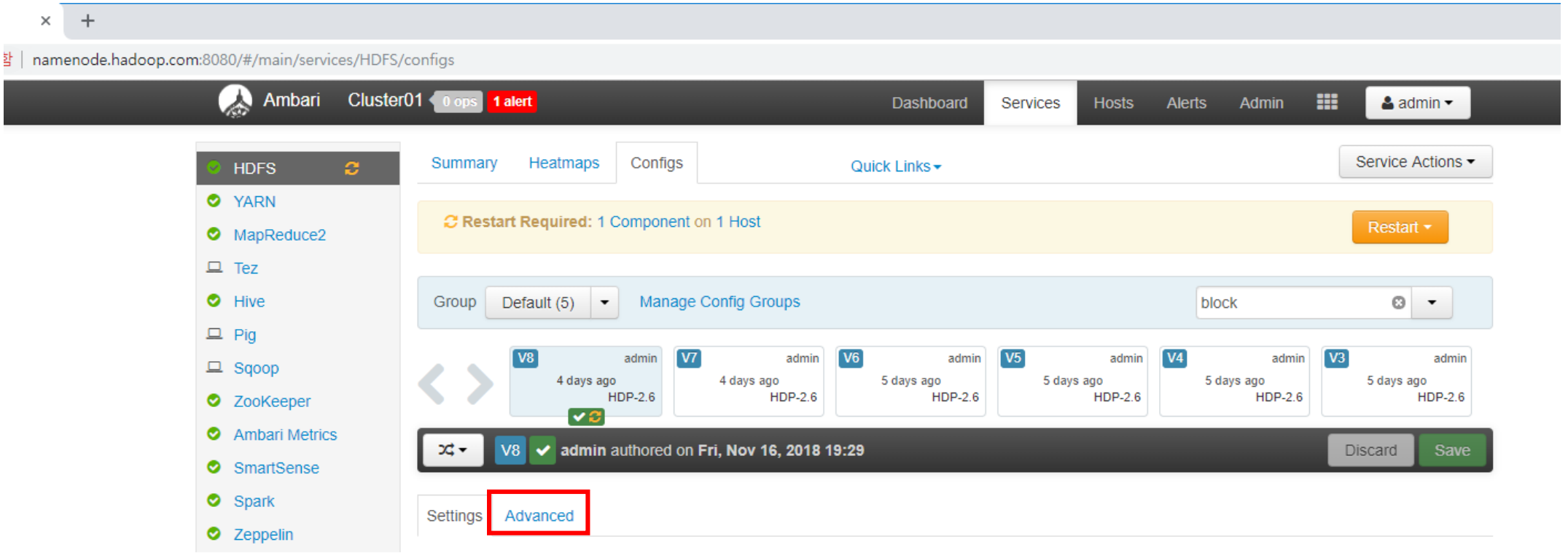
2.2 Requirements in Detail (Ambari)

Data block size 설정 확인



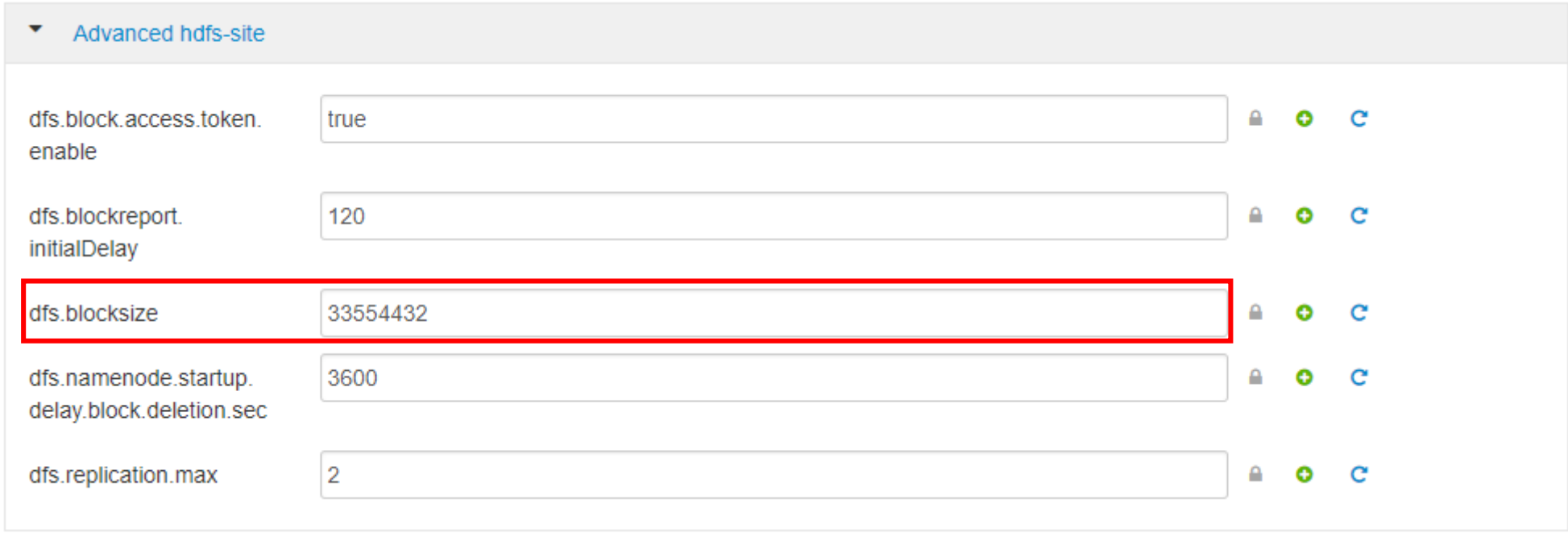
2.2 Requirements in Detail (Ambari)

Data block size 설정 확인



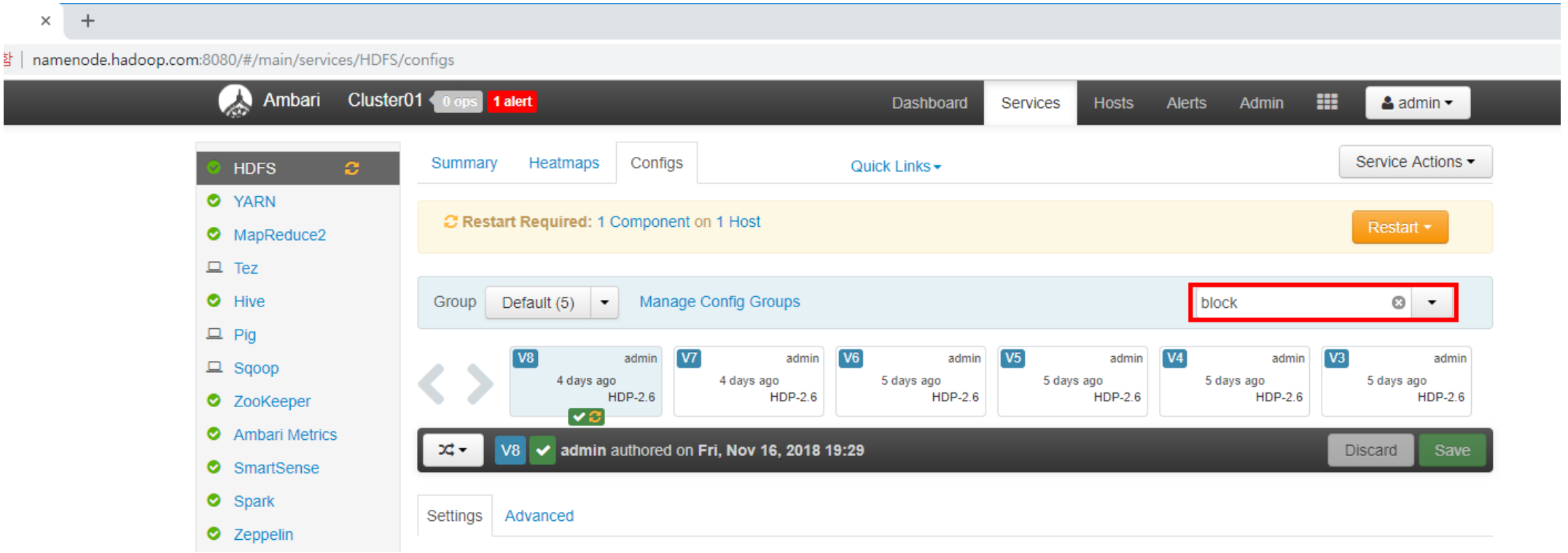
2.2 Requirements in Detail (Ambari)

Data block size 설정 확인



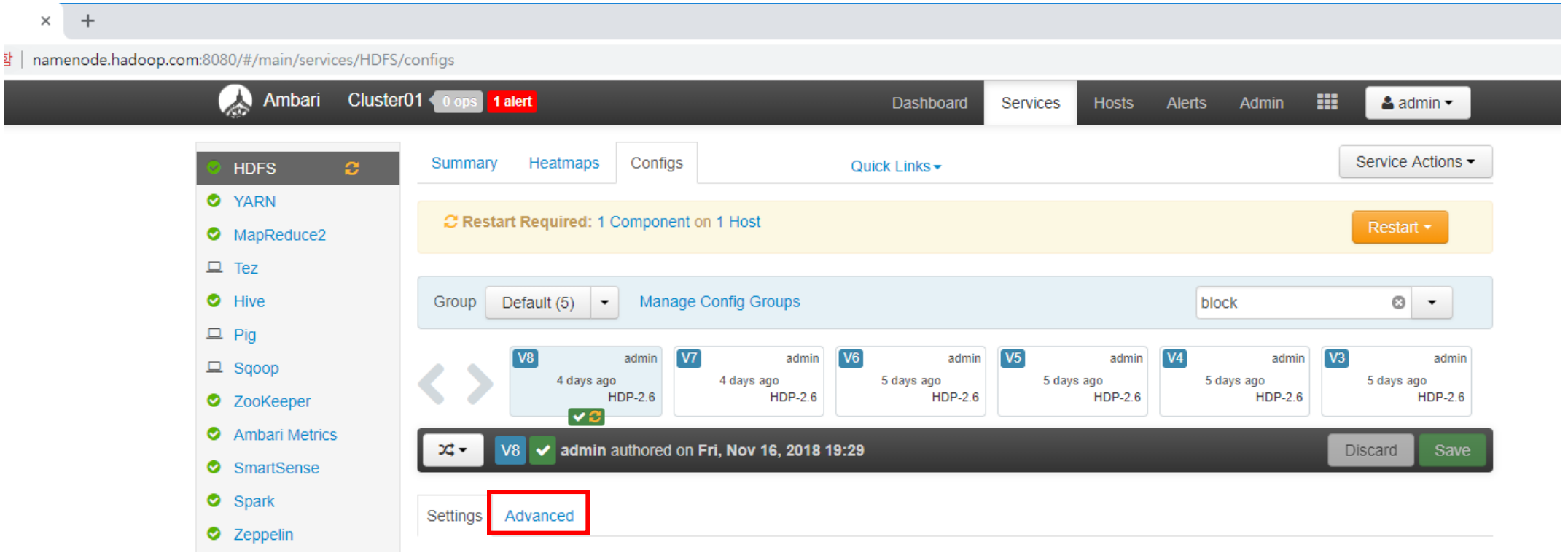
2.2 Requirements in Detail (Ambari)

Block Replication 설정 확인



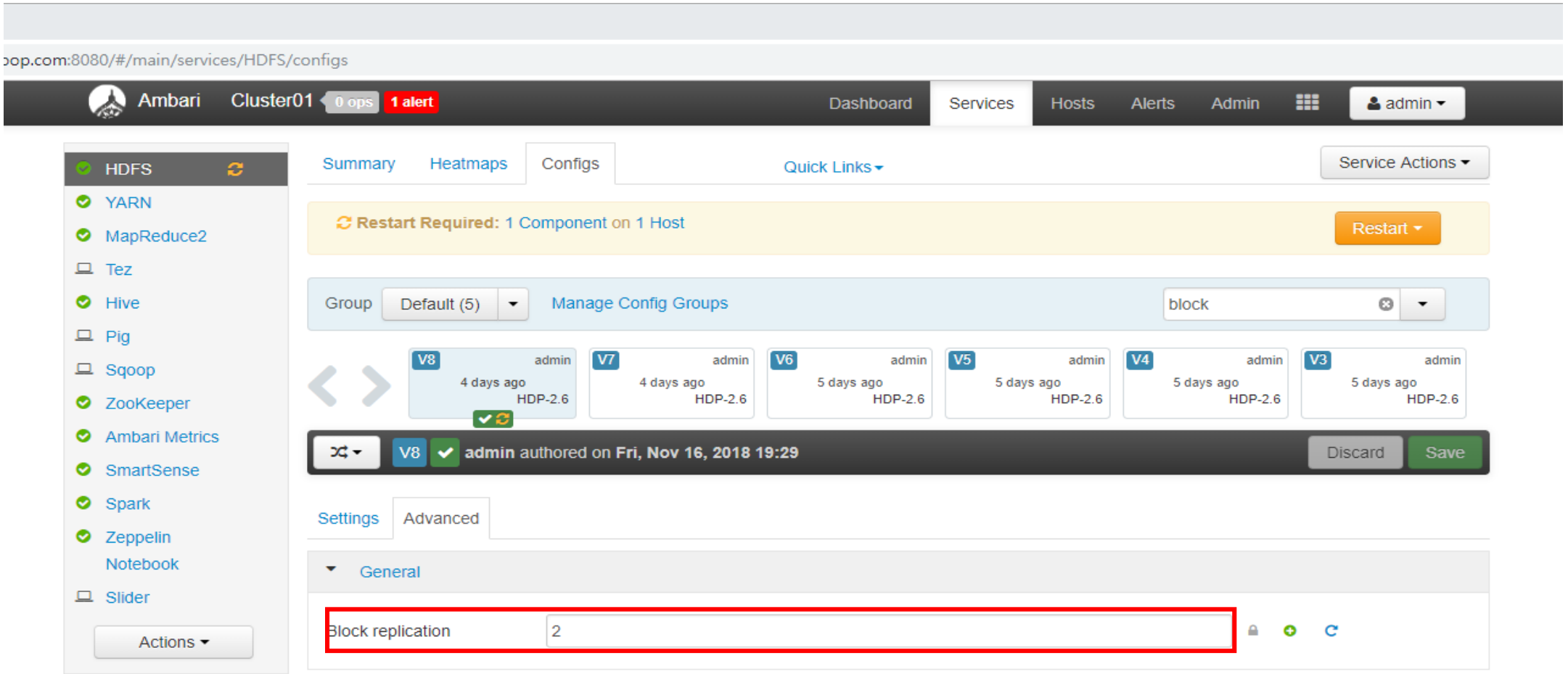
2.2 Requirements in Detail (Ambari)

Block Replication 설정 확인



2.2 Requirements in Detail (Ambari)

Block Replication 설정 확인



2.2 Requirements in Detail (Non-Ambari)

Block Replication 및 size 설정 확인

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
        </property>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>/home/hadoop/data/dfs/namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>/home/hadoop/data/dfs/datanode</value>
        </property>
        <property>
                <name>dfs.journalnode.edits.dir</name>
                <value>/home/hadoop/data/dfs/journalnode</value>
        </property>
        <property>
                <name>dfs.nameservices</name>
                <value>hadoop-cluster</value>
        </property>
        <property>
                <name>dfs.ha.namenodes.hadoop-cluster</name>
                <value>nn1,nn2</value>
        </property>
        <property>
                <name>dfs.namenode.rpc-address.hadoop-cluster.nn1</name>
                <value>jmj:8020</value>
        </property>
        <property>
                <name>dfs.namenode.rpc-address.hadoop-cluster.nn2</name>
                <value>sdh:8020</value>
        </property>
        <property>
                <name>dfs.namenode.http-address.hadoop-cluster.nn1</name>
                <value>jmj:50070</value>
        </property>
        <property>
                <name>dfs.namenode.http-address.hadoop-cluster.nn2</name>
                <value>sdh:50090</value>
        </property>
```

```
        <property>
                <name>dfs.namenode.shared.edits.dir</name>
                <value>qjournal://jmj:8485;sdh:8485;khk:8485/hadoop-cluster</value>
        </property>
         <property>
                <name>dfs.client.failover.proxy.provider.hadoop-cluster</name>
                <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
        </property>
        <property>
                <name>dfs.ha.fencing.methods</name>
                <value>sshfence</value>
        </property>
        <property>
                <name>dfs.ha.fencing.ssh.private-key-files</name>
                <value>/home/hadoop/.ssh/id_rsa</value>
        </property>
        <property>
                <name>dfs.ha.automatic-failover.enabled</name>
                <value>true</value>
        </property>
        <property>
                <name>dfs.block.size</name>
                <value>33554432</value>
        </property>
        <property>
                <name>dfs.hosts.exclude</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/exclude_list</value>
        </property>
        <property>
                <name>mapred.hosts.exclude</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/exclude_list</value>
        </property>
        <property>
                <name>dfs.hosts</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/include_list</value>
        </property>
</configuration>
```

03 | Service Usage Process

HDP 관리 페이지 접속 방법

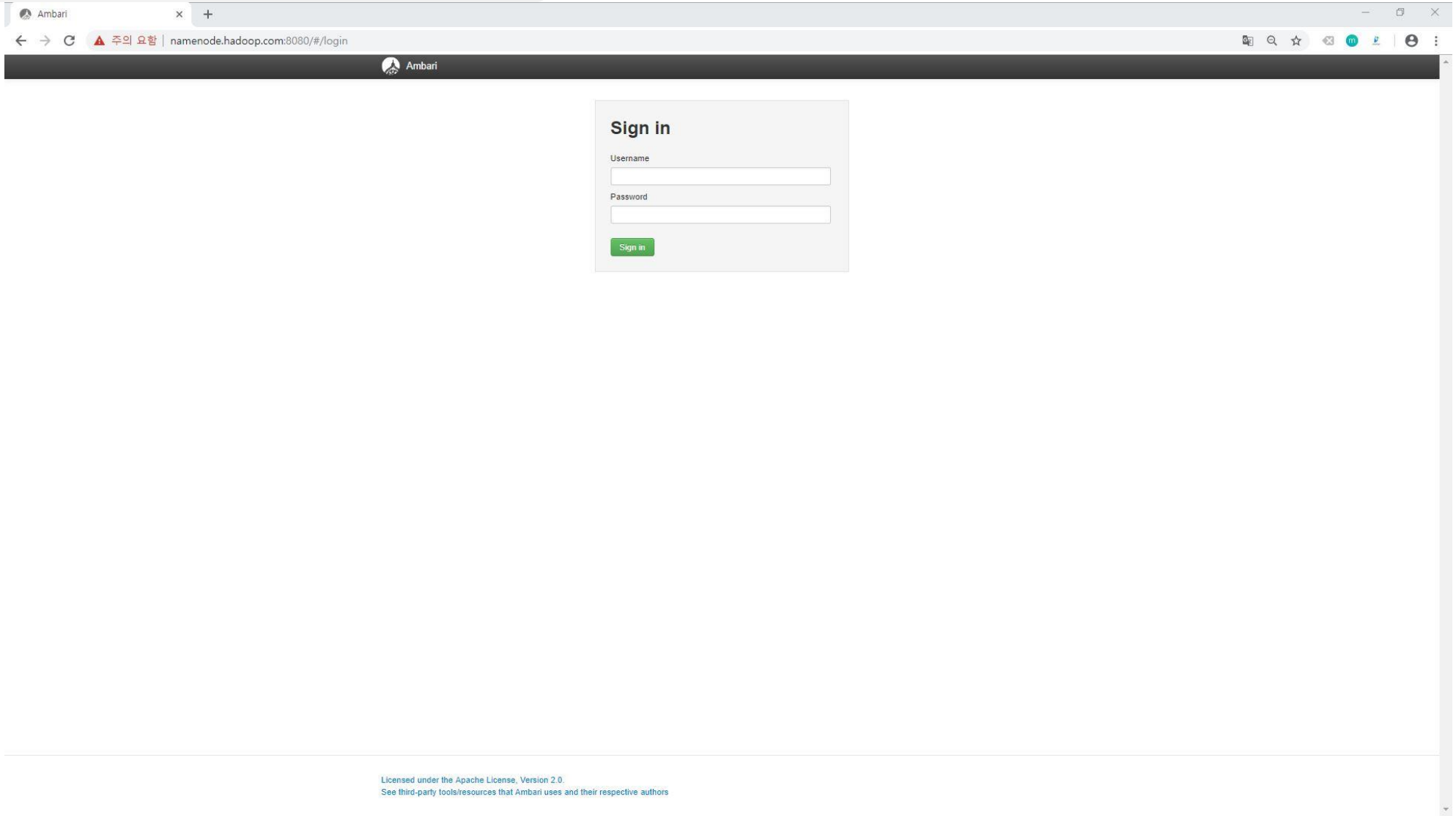
3.1 HDP 관리 페이지 접속 방법

Ambari

```
1. Namenode의 가상 머신에서 다음 과정 진행.

[root@host명 ~]# systemctl stop firewalld (방화벽 해제)
[root@host명 ~]# ambari-server start (ambari server 시작)
[root@host명 ~]# abmari-agent start (ambari agent 시작)

2. 웹 브라우저의 URL 주소 입력부분에 HDP 관리자 페이지를 실행할 URL 입력.
http://'namenode의 host명 또는 ip주소':8080

3. HDP 관리자 페이지가 실행되면 Username : admin / Password : admin 입력
```

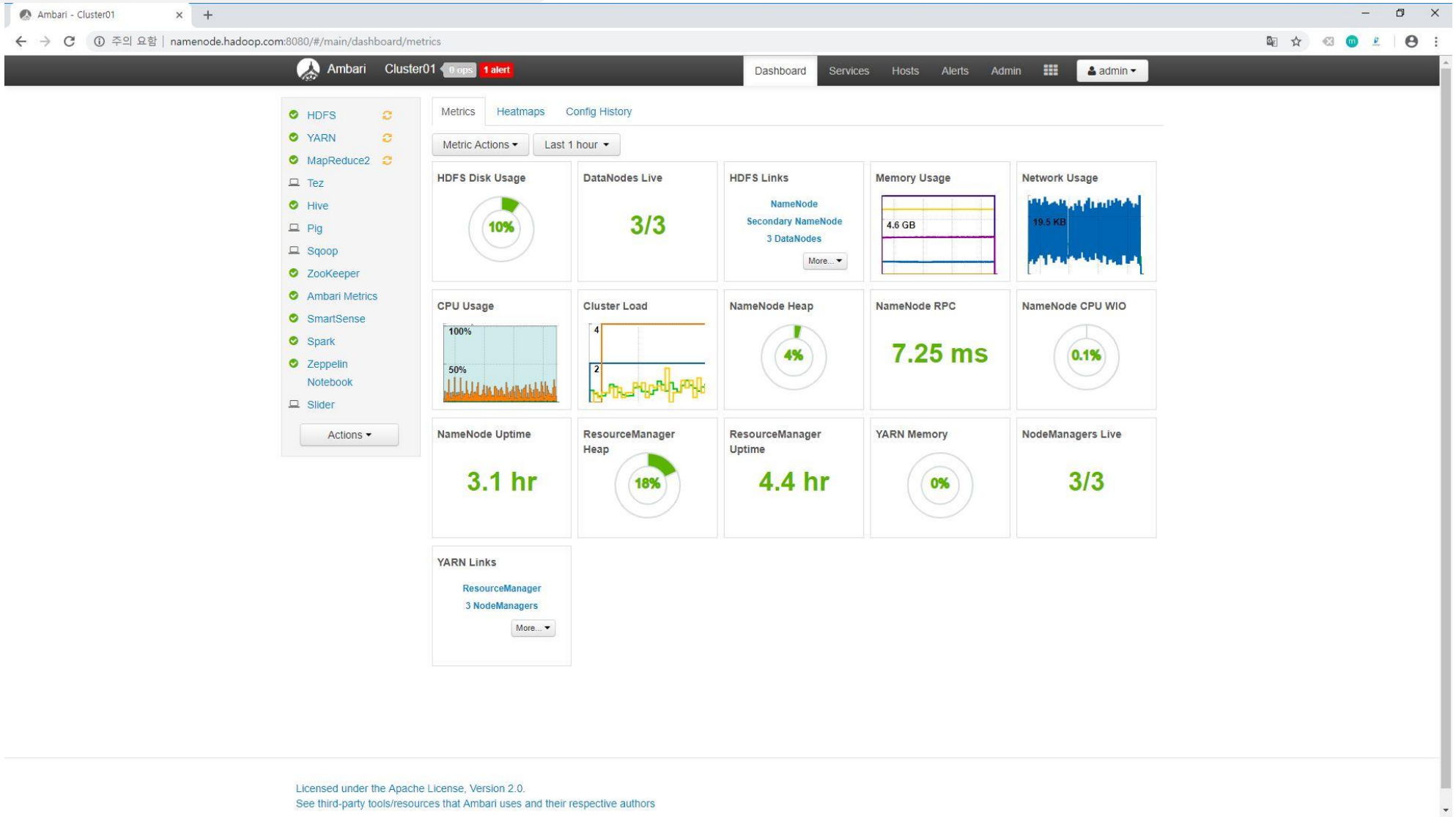
3.1 HDP 관리 페이지 접속 방법

Ambari



3.1 HDP 관리 페이지 접속 방법

Ambari



3.1 HDP 관리 페이지 접속 방법

Non Ambari

```
1. zookeeper가 설치된 각 서버에서 다음 과정 진행
[zookeeper@jmj ~]$ source .bashrc
[zookeeper@jmj ~]$ zkServer.sh .start
[zookeeper@jmj ~]$ zkServer.sh .status

      Mode ghkrdls : leader(1), follower(2)

2. jmj 호스트에서 zookeeper format
[hadoop@jmj ~]$ hdfs zkfc -formatZK

3. 각각의 journalnode 실행
[hadoop@jmj ~]$ hadoop-daemon.sh start journalnode

4. jmj 호스트에서 namenode 포맷
[hadoop@jmj ~]$ hdfs namenode -format

5. jmj 호스트에서 namenode 및 zkfc 실행
[hadoop@jmj ~]$ hadoop-daemon.sh start namenode
[hadoop@jmj ~]$ hadoop-daemon.sh start zkfc
```

3.1 HDP 관리 페이지 접속 방법

Non Ambari

```
6. jmj 호스트에서 datanodes 실행
[hadoop@jmj ~]$ hadoop-daemons.sh start datanode

7. sdh 호스트에서 standbynamenode 및 zkfe 실행
[hadoop@jmj ~]$ hdfs namenode -bootatarpStandby
[hadoop@jmj ~]$ hadoop-daemon.sh start namenode
[hadoop@jmj ~]$ hadoop-daemons.sh start zkfc

8. 하둡 분산 정상 설치 여부 확인 (wordcount실행)
[hadoop@jmj ~]$ hdfs dfs -put /home/hadoop/hadoop2/etc/hadoop/hadoop-env.sh /user/hadoop/conf
[hadoop@jmj ~]$ yarn jaf /home/hadoop/hadoop2/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount conf output

9. jmj 호스트에서 yarn-cluster 실행
[hadoop@jmj ~]$ start-yarn.sh

10. jmj 호스트에서 historyserver 실행
[hadoop@jmj ~]$ mr-jobhistory-daemon.sh start historyserver

11. jmj 호스트에서 proxyserver 실행
[hadoop@jmj ~]$ yarn-demon.sh start proxyserver
```

3.1 HDP 관리 페이지 접속 방법

Non Ambari

```
[hadoop@jmj sbin]$ jps
3504 NameNode
5106 NodeManager
4996 ResourceManager
4203 JobHistoryServer
3595 DFSZKFailoverController
3389 JournalNode
3725 DataNode
5327 Jps
```

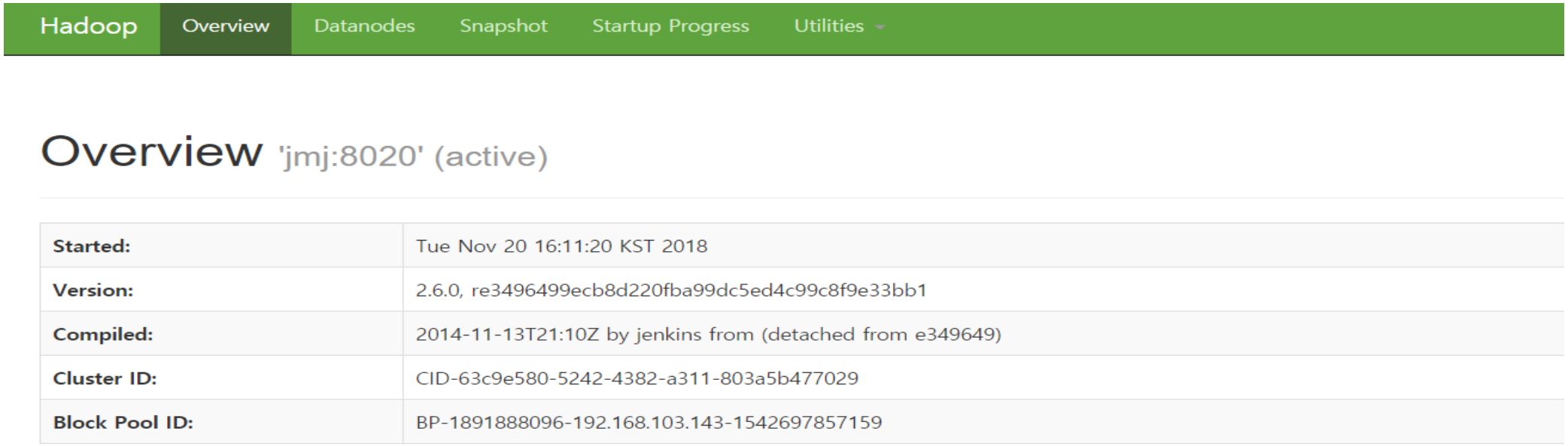
```
[hadoop@sdh ~]$ jps
2085 JournalNode
2168 DataNode
2393 DFSZKFailoverController
2300 NameNode
3983 Jps
```

```
[hadoop@khk hadoop]$ jps
8385 NodeManager
7594 JournalNode
8429 Jps
7677 DataNode
```

```
[hadoop@lkh ~]$ jps
2900 Jps
2842 NodeManager
1947 DataNode
```

3.1 HDP 관리 페이지 접속 방법

Non Ambari



3.1 HDP 관리 페이지 접속 방법

Non Ambari

Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
jmj (192.168.103.143:50010)	1	Decommissioned	31.18 GB	24 KB	6.06 GB	25.12 GB	0	24 KB (0%)	0	2.6.0
sdh (192.168.103.142:50010)	1	In Service	27.44 GB	683.03 MB	3.7 GB	23.07 GB	44	683.03 MB (2.43%)	0	2.6.0
lkh (192.168.103.144:50010)	0	In Service	31.18 GB	747.88 MB	2.87 GB	27.57 GB	50	747.88 MB (2.34%)	0	2.6.0
khk (192.168.103.141:50010)	1	In Service	31.18 GB	834.75 MB	3.41 GB	26.95 GB	50	834.75 MB (2.61%)	0	2.6.0

3.1 HDP 관리 페이지 접속 방법

Non Ambari

Browse Directory

/user/hadoop Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	0	0 B	conf
drwxr-xr-x	hadoop	supergroup	0 B	0	0 B	output

Browse Directory

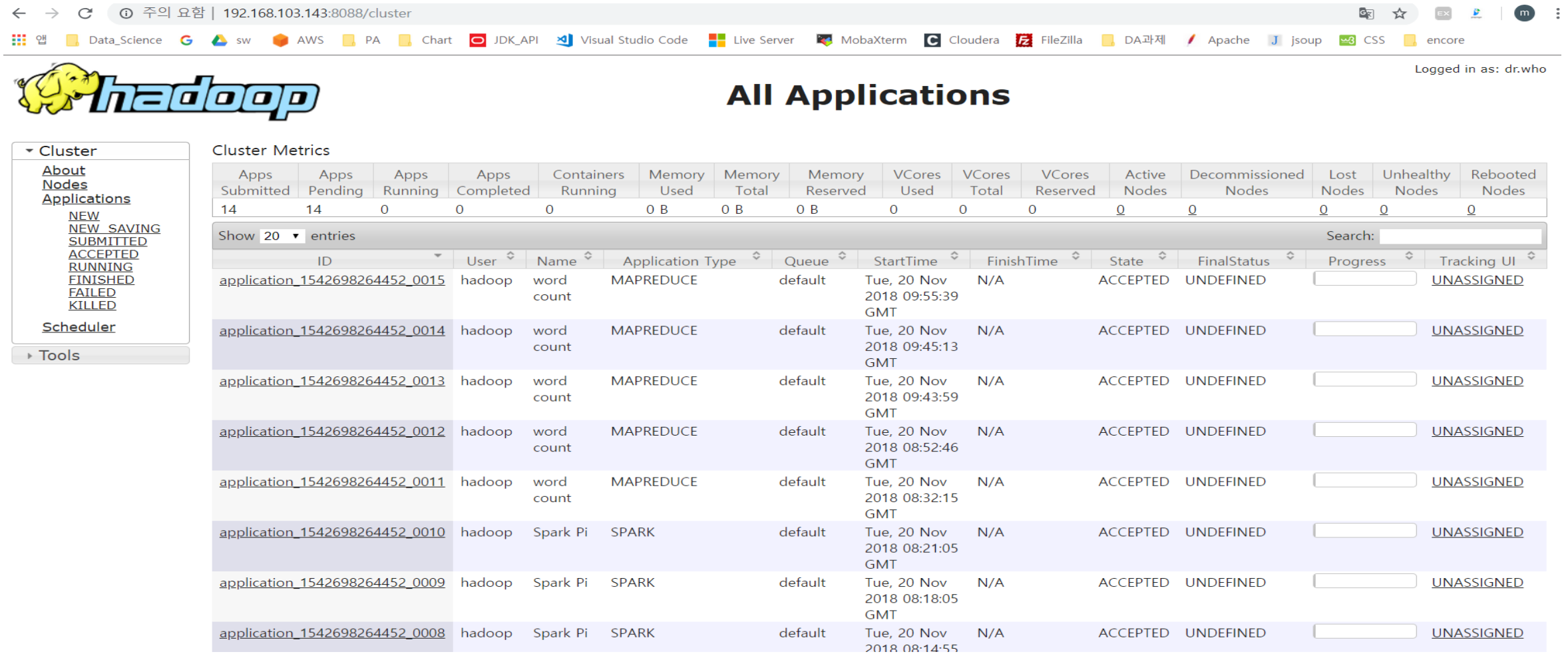
/user/hadoop/output Go!

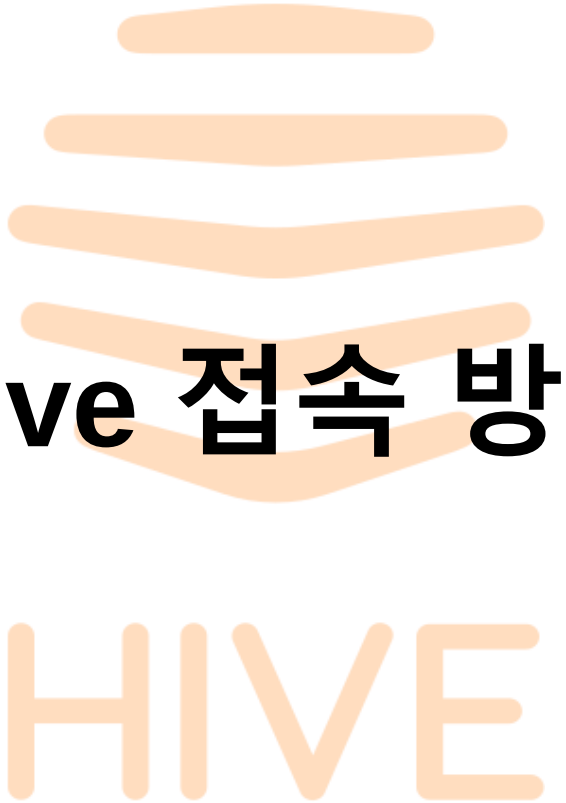
Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	2	32 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	1.46 KB	2	32 MB	part-r-00000

```
[hadoop@client hadoop2]$ hdfs dfs -cat /user/hadoop/output/part-r-00000
10      1
Apart   1
Auto-Save       1
Bookmark        1
Details 1
Disclaimer:     1
For     1
In      2
```

3.1 HDP 관리 페이지 접속 방법

Non Ambari





Hive 접속 방법

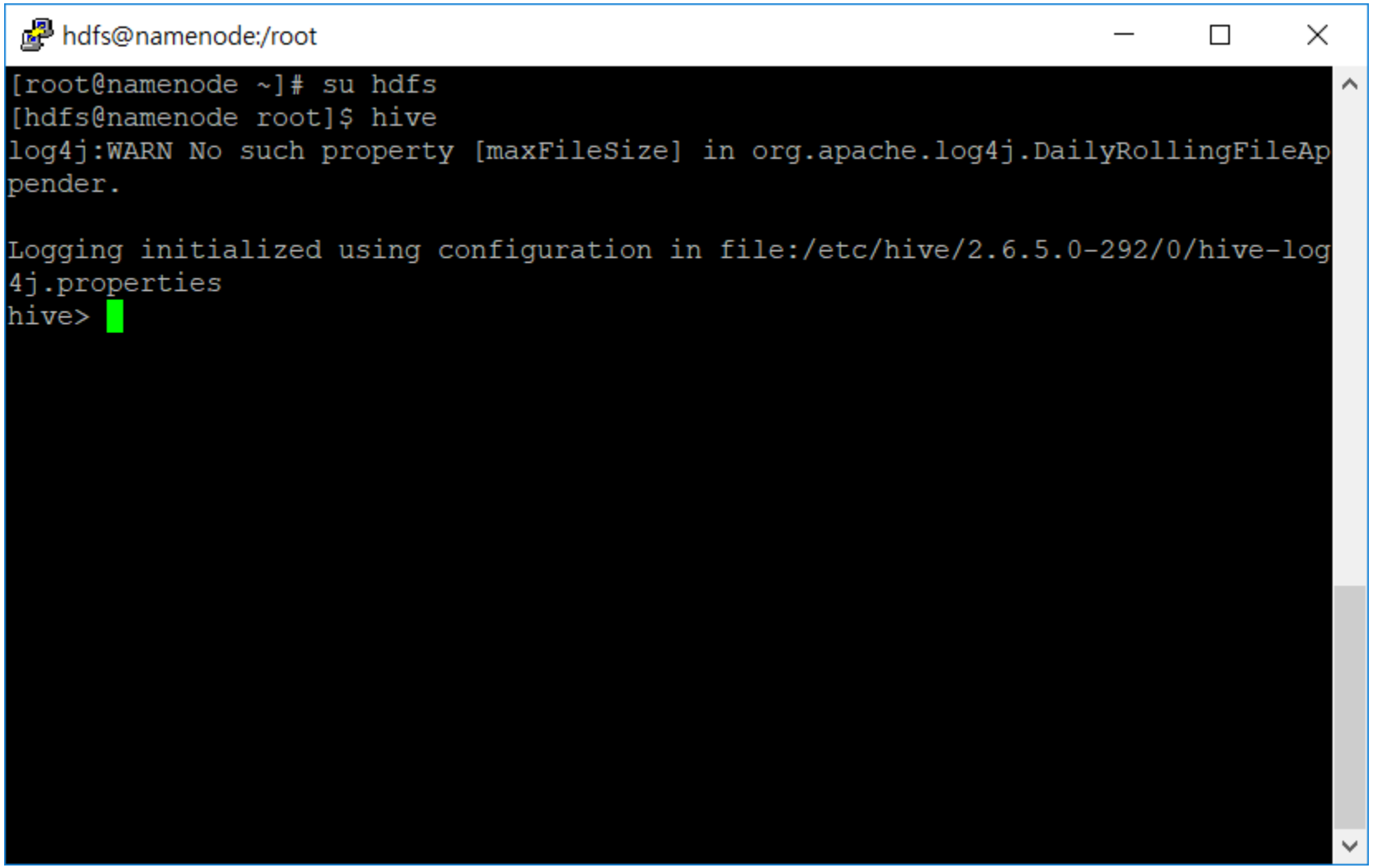
3.2 Hive 접속 방법

Ambari

```
1. Namenode의 가상 머신에서 root 계정에서 hdfs 계정으로 접속.

[root@host명 ~]# su hdfs
[hdfs@host명 root]$

2. hdfs 계정의 /user/hive 디렉토리의 하위에 연습용으로 사용할 tests 디렉토리 생성.
[hdfs@host명 root]$ hdfs dfs -mkdir /user/hive/tests

3. hive 접속 (실행되는 시간 다소 소요)
[hdfs@host명 root]$ hive
hive>

4. hdfs 계정에 새로 생성한 디렉토리인 /user/hive/tests를 확인하기 위해
웹 브라우저에 URL 주소 입력 부분에 다음의 내용을 입력.
http://'namenode의 host명 또는 ip주소':50070

50070은 hdfs의 기본 namenode의 포트번호.

5. 웹 UI 화면이 출력된 후 화면의 상단 메뉴 중 utilites에
Brows the file system을 선택하면 hdfs 계정의 디렉토리 확인 가능.

6. 화면에 나타난 디렉토리를 선택하면서 이동.
user 디렉토리를 선택하고 hive 디렉토리를 선택하면 tests 디렉토리 확인.
```

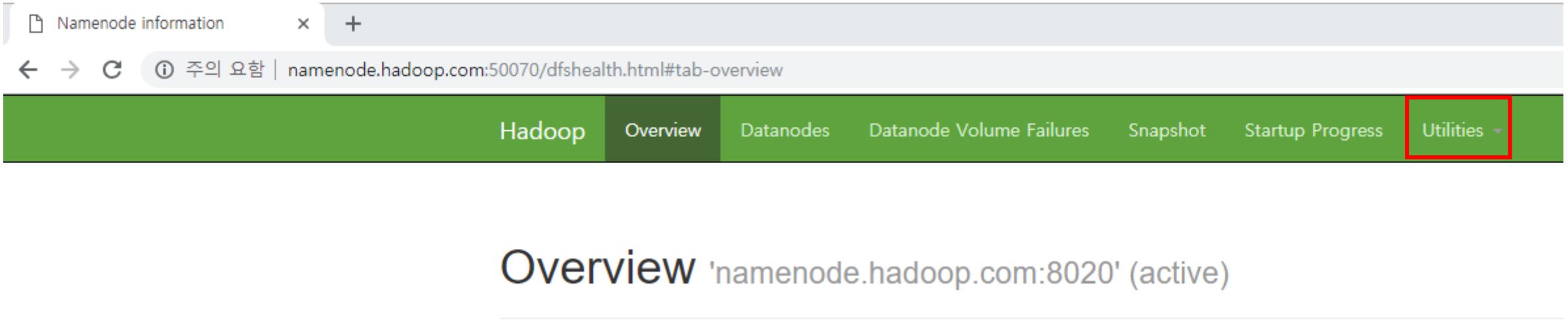
3.2 Hive 접속 방법

Ambari



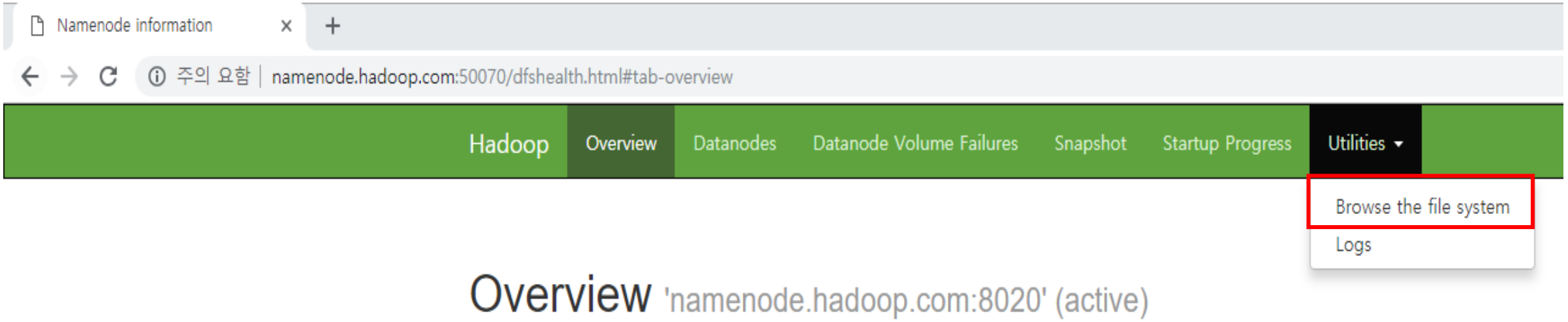
3.2 Hive 접속 방법

Ambari



3.2 Hive 접속 방법

Ambari



3.2 Hive 접속 방법

Ambari

Browse Directory

/ Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	yarn	hadoop	0 B	2018. 11. 15. 오후 5:18:03	0	0 B	app-logs
drwxr-xr-x	hdfs	hdfs	0 B	2018. 11. 15. 오후 4:30:21	0	0 B	apps
drwxr-xr-x	yarn	hadoop	0 B	2018. 11. 15. 오전 9:37:51	0	0 B	ats
drwxr-xr-x	hdfs	hdfs	0 B	2018. 11. 15. 오전 9:37:44	0	0 B	hdp
drwxr-xr-x	mapred	hdfs	0 B	2018. 11. 15. 오전 9:37:40	0	0 B	mapred
drwxrwxrwx	mapred	hadoop	0 B	2018. 11. 15. 오전 9:38:06	0	0 B	mr-history
drwxrwxrwx	spark	hadoop	0 B	2018. 11. 20. 오후 5:41:55	0	0 B	spark-history
drwxrwxrwx	spark	hadoop	0 B	2018. 11. 16. 오후 7:23:56	0	0 B	spark2-history
drwxrwxrwx	hdfs	hdfs	0 B	2018. 11. 15. 오후 12:56:14	0	0 B	tmp
drwxr-xr-x	hdfs	hdfs	0 B	2018. 11. 16. 오후 2:00:09	0	0 B	user

3.2 Hive 접속 방법

Ambari

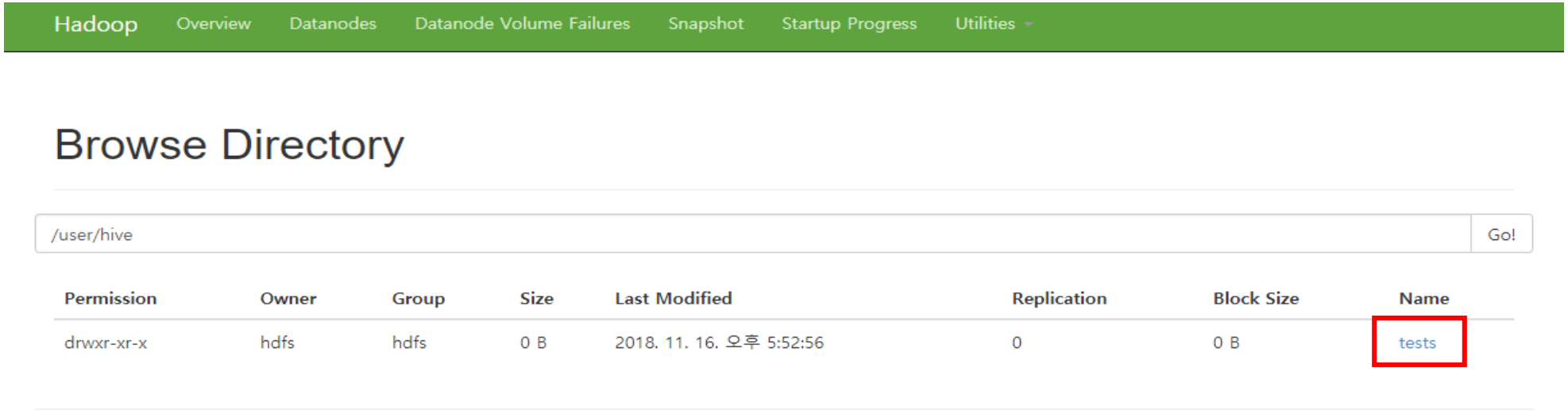
Browse Directory

/user Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hdfs	hdfs	0 B	2018. 11. 16. 오후 2:00:09	0	0 B	admin
drwxrwx---	ambari-qa	hdfs	0 B	2018. 11. 15. 오후 12:54:03	0	0 B	ambari-qa
drwxr-xr-x	hcat	hdfs	0 B	2018. 11. 15. 오후 12:50:39	0	0 B	hcat
drwxr-xr-x	hdfs	hdfs	0 B	2018. 11. 16. 오후 12:34:52	0	0 B	hdfs
drwxr-xr-x	hive	hdfs	0 B	2018. 11. 16. 오후 4:07:19	0	0 B	hive
drwxrwxr-x	spark	spark	0 B	2018. 11. 19. 오전 11:13:56	0	0 B	spark
drwxr-xr-x	zeppelin	hdfs	0 B	2018. 11. 19. 오전 10:06:19	0	0 B	zeppelin

3.2 Hive 접속 방법

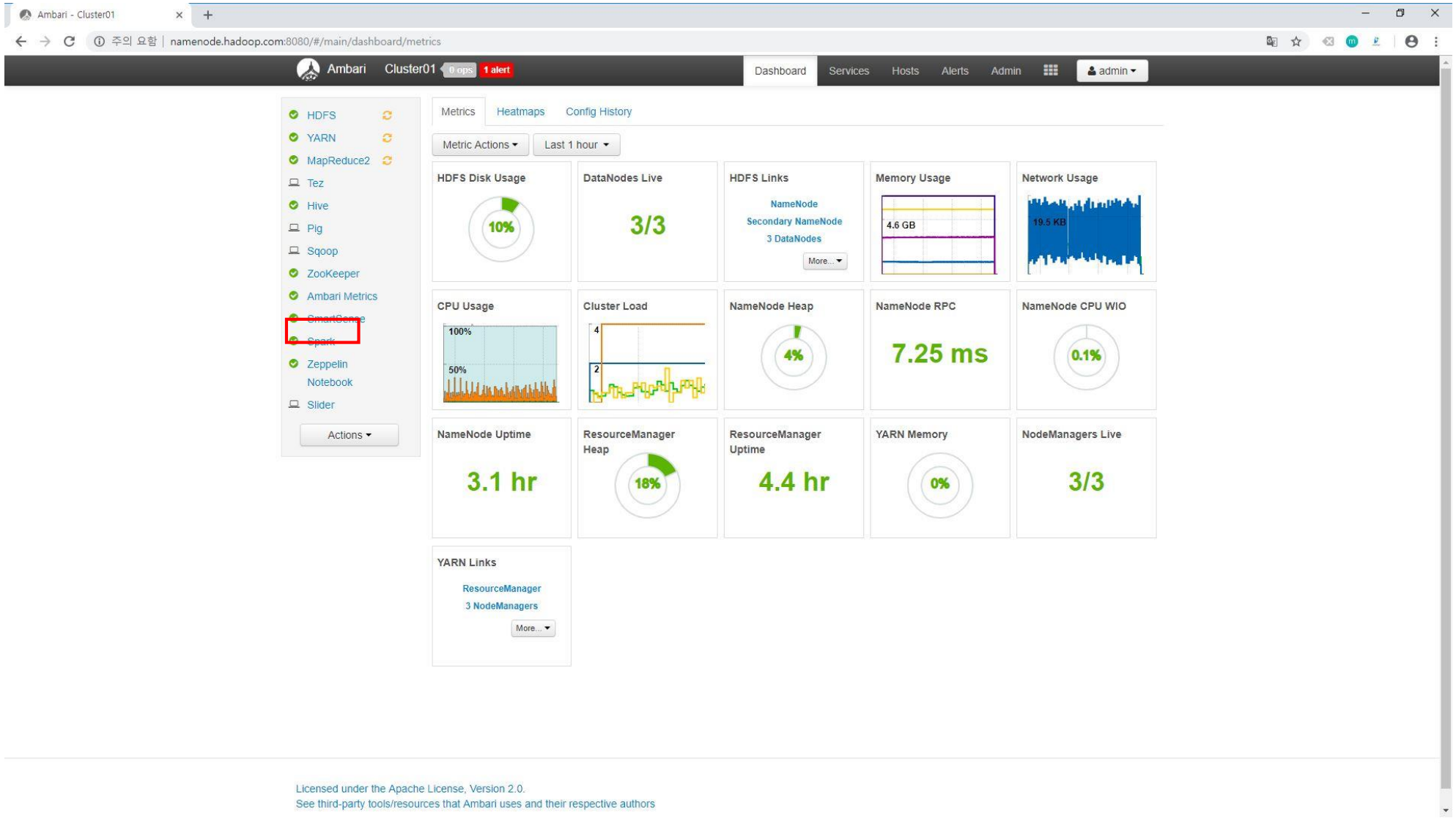
Ambari



Spark History Server 접속 방법

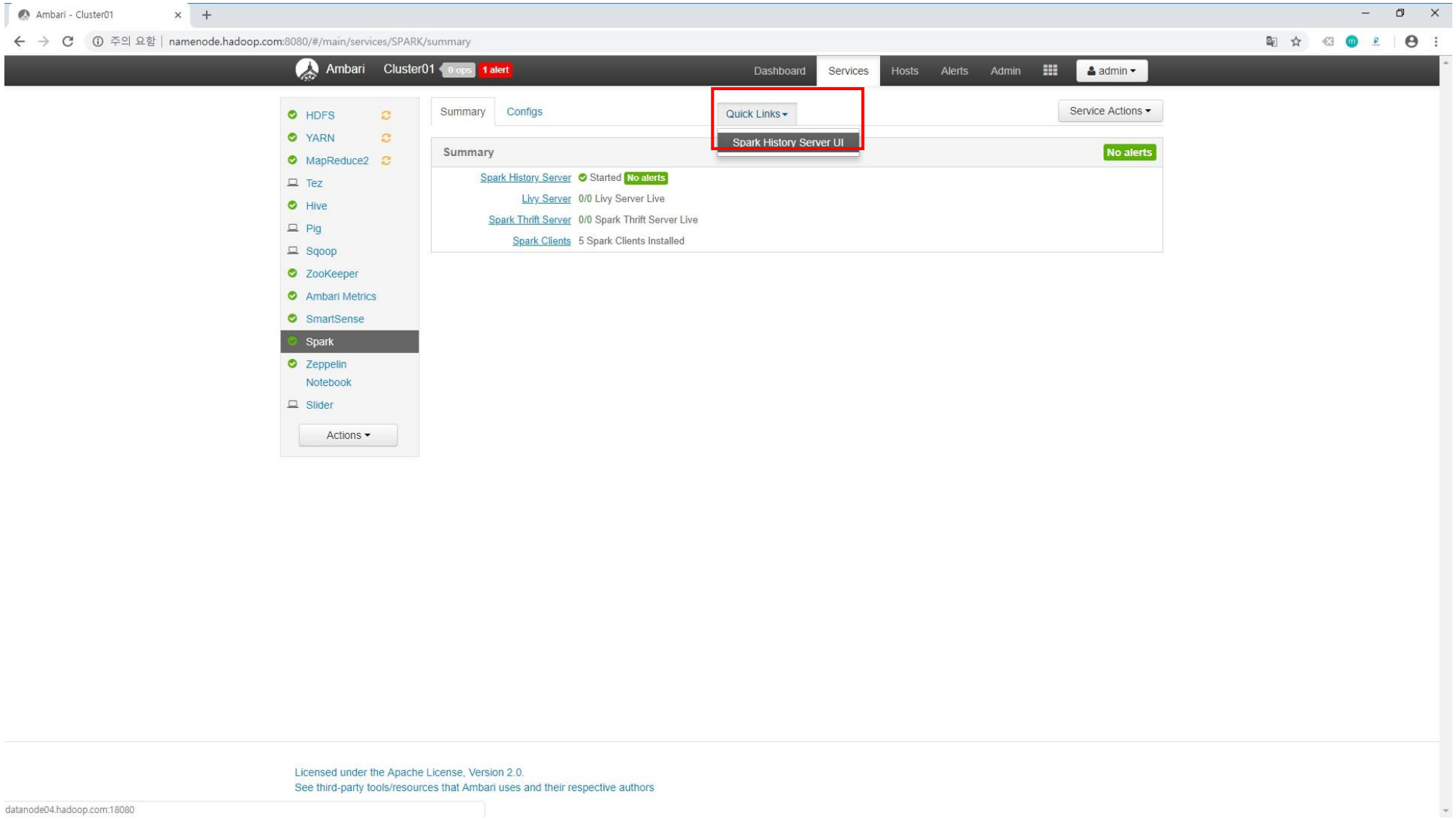
3.3 Spark History Server 접속 방법

Ambari



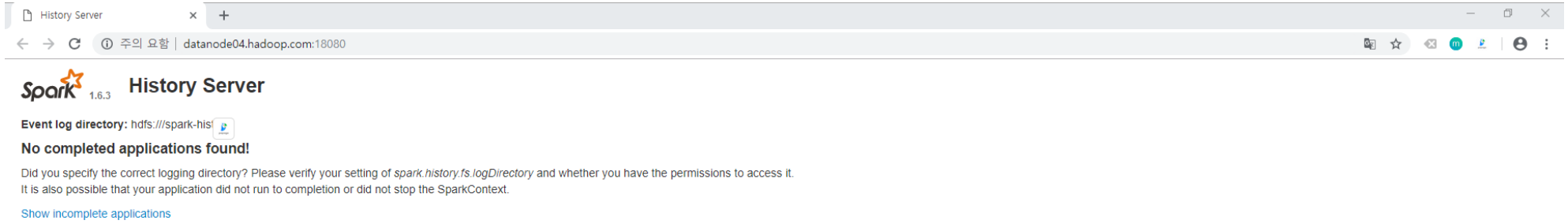
3.3 Spark History Server 접속 방법

Ambari



3.3 Spark History Server 접속 방법

Ambari



3.3 Spark History Server 접속 방법

Non Ambari

```
1. jmj 호스트(master)에서 master, slaves 실행
[hadoop@jmj sbin]$ start-master.sh
[hadoop@jmj sbin]$ start-slaves.sh
2. spark test
[hadoop@lkh sbin]$ spark-submit --master spark://jmj:7077 --class org.apache.spark.examples.SparkPi /home/hadoop/spark2/examples/jars/spark-examples*.jar 10
3. spark history-server 실행
[hadoop@jmj ~]$ hdfs dfs -mkdir /spark-logs
[hadoop@jmj ~]$ hdfs dfs -chmod 777 /spark-logs
[hadoop@jmj conf]$ start-history-server.sh
```

```
[hadoop@jmj sbin]$ jps
3504 NameNode
6609 Master
4996 ResourceManager
6679 Jps
4203 JobHistoryServer
3595 DFSZKFailoverController
3389 JournalNode
3725 DataNode
```

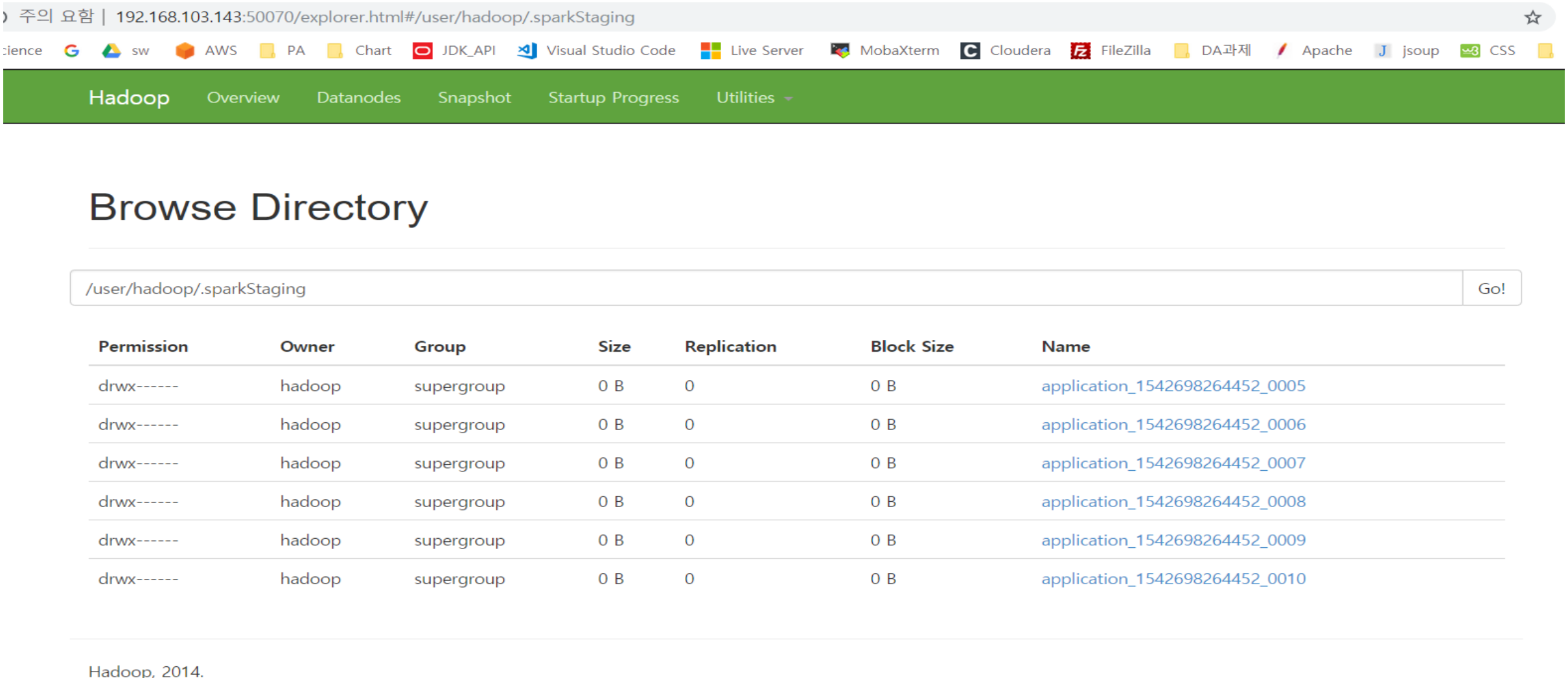
```
[hadoop@sdh ~]$ jps
2085 JournalNode
2168 DataNode
4504 Worker
2393 DFSZKFailoverController
4556 Jps
2300 NameNode
```

```
^C[hadoop@khk hadoop]$ jps
8724 Worker
8806 Jps
7594 JournalNode
7677 DataNode
```

```
[hadoop@lkh ~]$ jps
3207 Worker
3257 Jps
1947 DataNode
```

3.3 Spark History Server 접속 방법

Non Ambari



3.3 Spark History Server 접속 방법

Non Ambari

Spark Master at spark://jmj:7077

URL: spark://jmj:7077
REST URL: spark://jmj:6066 *(cluster mode)*
Alive Workers: 3
Cores in use: 6 Total, 0 Used
Memory in use: 23.6 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20181116182131-192.168.103.141-41356	192.168.103.141:41356	ALIVE	2 (0 Used)	9.4 GB (0.0 B Used)
worker-20181120165443-192.168.103.142-35904	192.168.103.142:35904	ALIVE	2 (0 Used)	9.4 GB (0.0 B Used)
worker-20181120165744-192.168.103.144-38792	192.168.103.144:38792	ALIVE	2 (0 Used)	4.8 GB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

3.3 Spark History Server 접속 방법

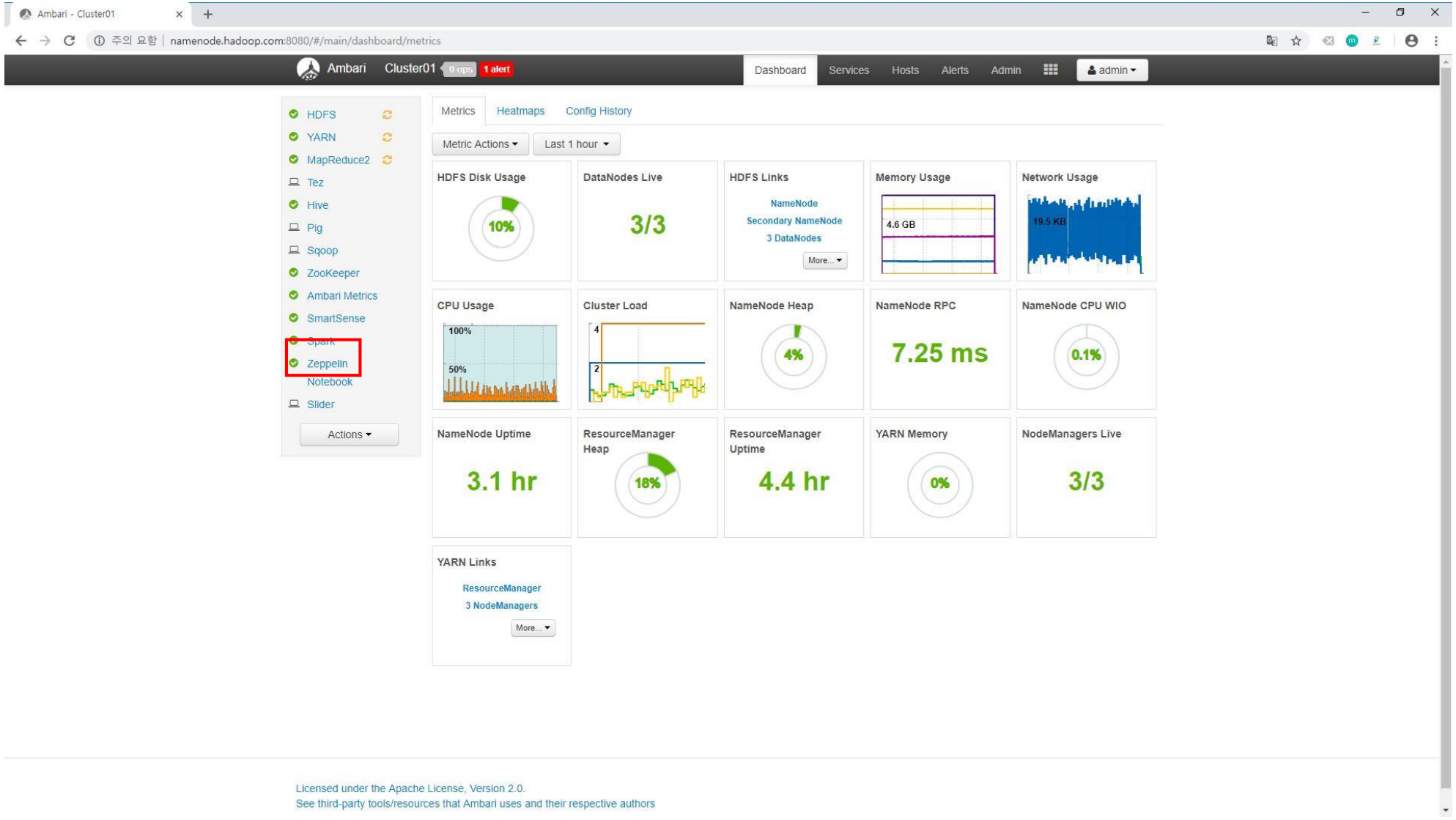
Non Ambari

```
18/11/22 16:46:27 INFO TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in
1347 ms on 192.168.103.144 (executor 1) (9/10)
18/11/22 16:46:27 INFO TaskSetManager: Finished task 3.0 in stage 0.0 (TID 3) in
1315 ms on 192.168.103.144 (executor 1) (10/10)
18/11/22 16:46:27 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) f
inished in 3.556 s
18/11/22 16:46:27 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have a
ll completed, from pool
18/11/22 16:46:27 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38,
took 4.614564 s
Pi is roughly 3.144335144335144
18/11/22 16:46:27 INFO SparkUI: Stopped Spark web UI at http://192.168.103.144:40
40
18/11/22 16:46:27 INFO StandaloneSchedulerBackend: Shutting down all executors
18/11/22 16:46:27 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each
executor to shut down
18/11/22 16:46:28 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndp
oint stopped!
18/11/22 16:46:28 INFO MemoryStore: MemoryStore cleared
18/11/22 16:46:28 INFO BlockManager: BlockManager stopped
```

Zeppelin Notebook 접속 방법

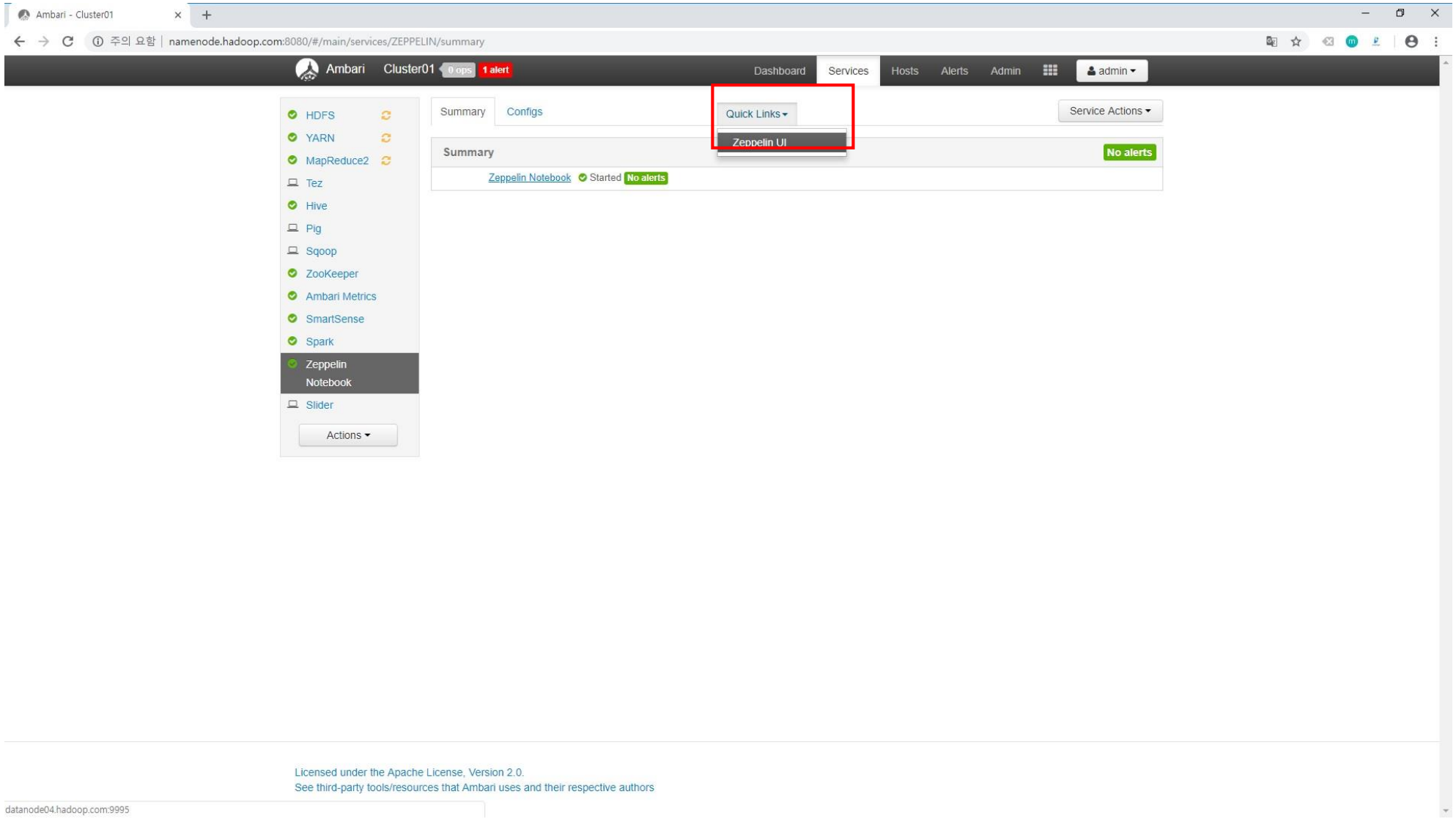
3.4 Zeppelin Notebook 접속 방법

Ambari



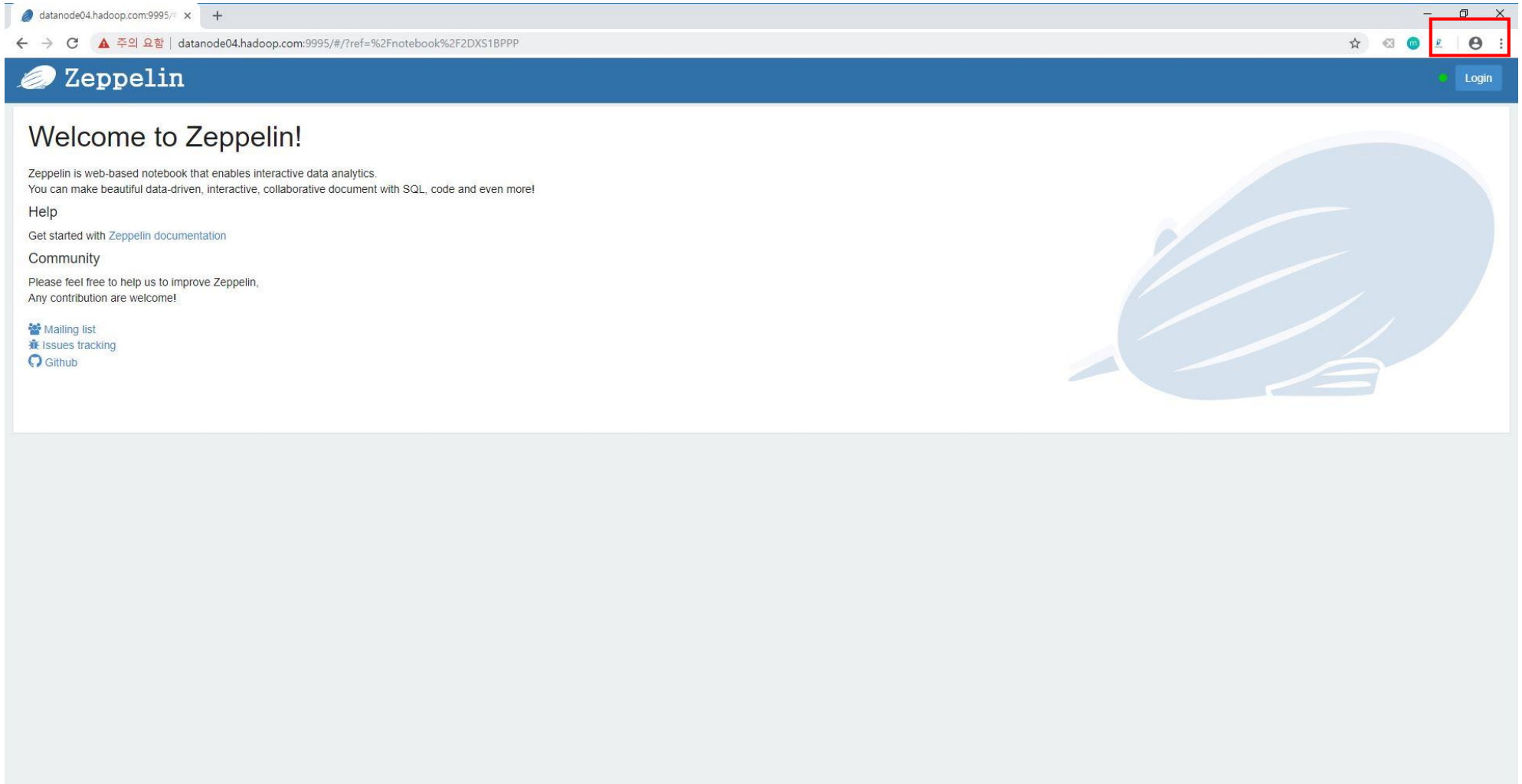
3.4 Zeppelin Notebook 접속 방법

Ambari



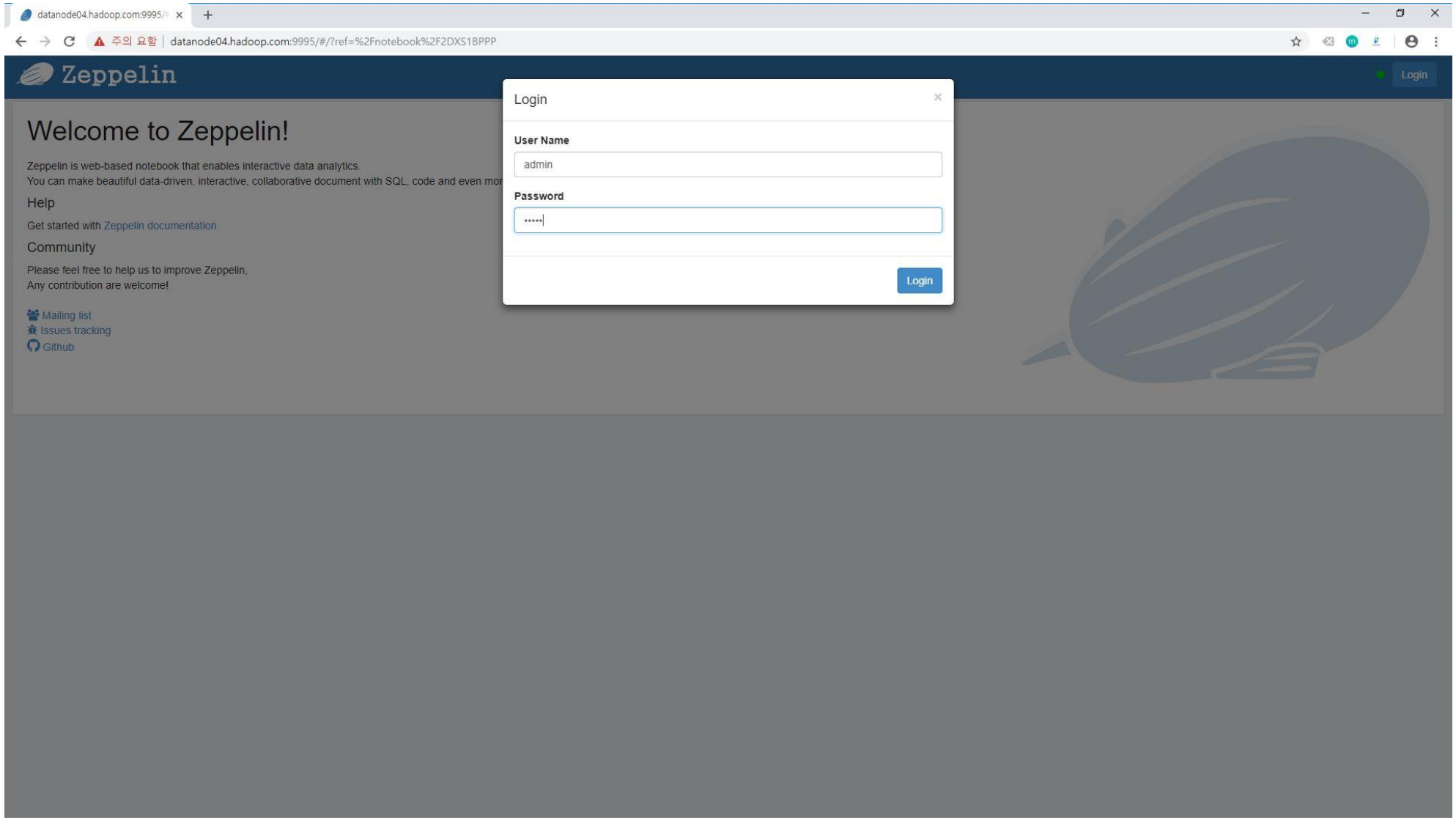
3.4 Spark History Server 접속 방법

Ambari



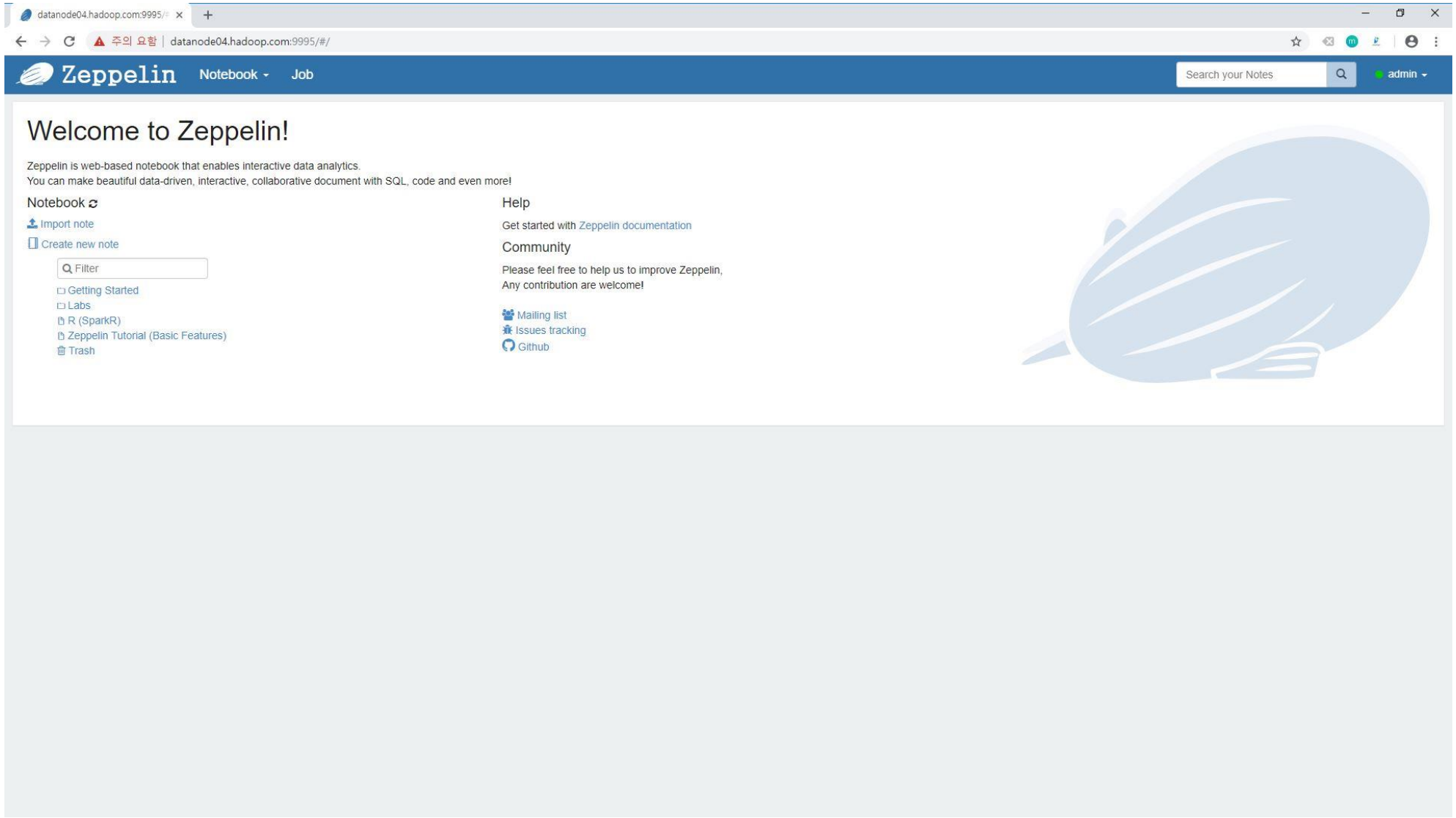
3.4 Spark History Server 접속 방법

Ambari



3.4 Spark History Server 접속 방법

Ambari



3.4 Spark History Server 접속 방법

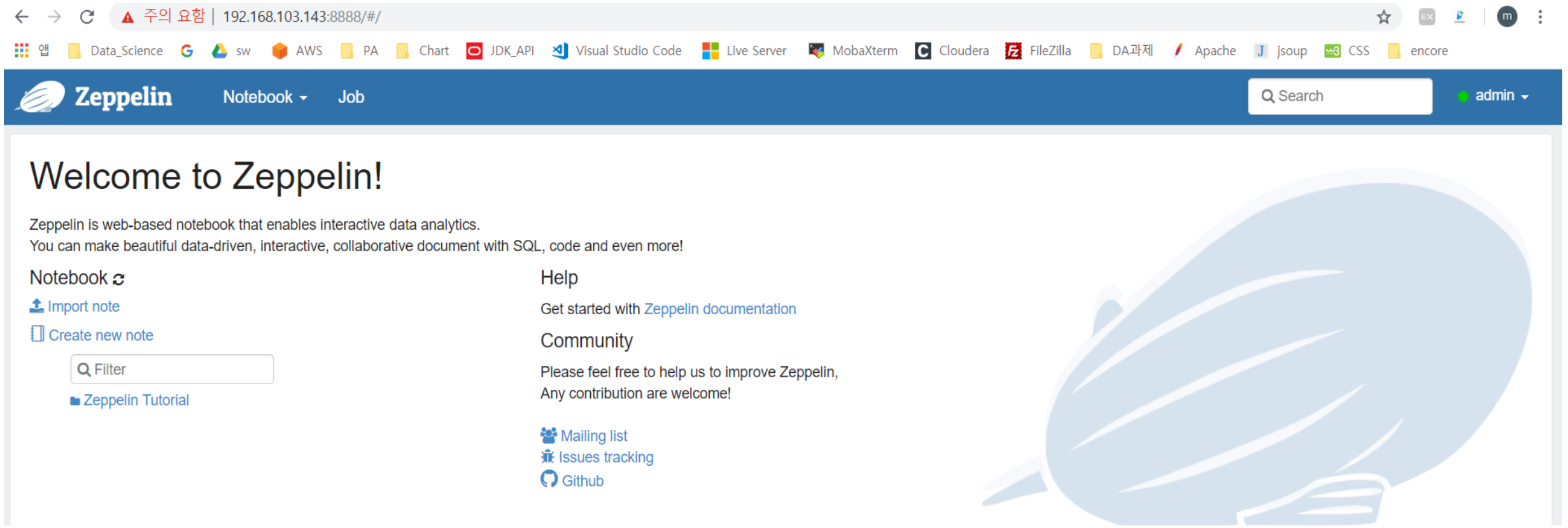
Non Ambari

```
1. zeppelin 실행
[hadoop@jmj bin]$ zeppelin-daemon.sh start
```

```
[hadoop@jmj bin]$ jps
3504 NameNode
6609 Master
4996 ResourceManager
7544 Jps
7481 ZeppelinServer
4203 JobHistoryServer
3595 DFSZKFailoverController
3389 JournalNode
3725 DataNode
```

3.4 Spark History Server 접속 방법

Non Ambari



04 | Demo

05 | Errors & Corrections

4. Errors in Detail

Ambari

오류 발생 상황

- 모든 host의 ambari server 및 agent에 대한 설정이 완료된 후 ambari를 실행했을 때 정상적으로 실행.
- 다음 날, 모든 host를 재가동하고 ambari를 실행했을 때 서비스들이 정상적으로 실행이 되지 않는 현상 발생.

해결 방법

- Ambari-agent.ini 파일 내용 수정
- Cert-verification.cfg 파일 내용 수정

```
[root@host명 ~]# vi /etc/ambari-agent/conf/ambari-agent.ini

    수정 1 (hostname 부분 수정)
    hostname=localhost  -->  hostname=namenode의 host명  (모든 host에 적용)

    수정 2 (프로토콜 설정 추가)
    [security] 부분
    force_https_protocol=PROTOCOL_TLSv1_2

[root@host명 ~]# vi /etc/python/cert-verification.cfg

    수정 1 (verify 수정)
    [https] 부분
    'verify=platform_default'  -->  'verify=disable'
```

4. Errors in Detail

Non Ambari

오류 발생 상황 1

- Probuf-2.5.0 설치후 configure 파일 실행 시 실행 되지 않음

해결 방법

- 컴파일러 설치 유무 확인 및 설치
  [root@localhost protobuf-2.5.0]# rpm –qa | grep gcc (컴파일러 설치 유무 확인)
  [root@localhost protobuf-2.5.0]# yum –y install gcc (gcc 설치)

- Development Tools 설치
  [root@localhost protobuf-2.5.0]# yum group install "Development Tools"

오류 발생 상황 2

- Hadoop-2.6.0 설치 후 conf 디렉터리에서 설정 시 수정 불가

해결 방법

- Hadoop user에서 Hadoop-2.6.0을 재 다운로드하여 설치

오류 발생 상황 3

- JPS 실행 시 datanode가 실행 되지 않음

해결 방법

- Hadoop Home 하위의 data – dfs – datanode 디렉터리 삭제

4. Errors in Detail

Non Ambari

오류 발생 상황 4

- Spark 2.4.0 설치 시 Hadoop-2.6.0과 호환되지 않음

해결 방법

- Spark-2.1.3 재설치

오류 발생 상황 5

- JDK 7 version은 Spark-2.1.3에서 호환되지 않음

해결 방법

- JDK-8u-181 재설치

오류 발생 상황 6

- Spark history server UI (192.168.103.143:18080) 접속 불가

해결 방법

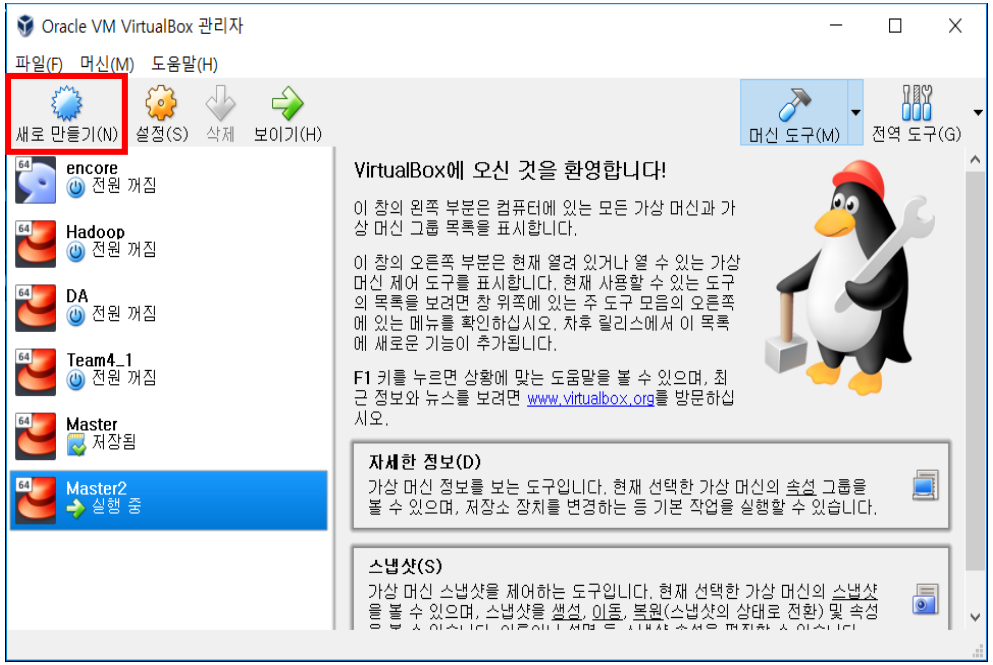
- Hadoop Namenode UI의 spark-logs 디렉터리의 권한 반영(777) 및 그룹명 변경 (supergroup -> Hadoop)
- 해결하지 못함

06 | Appendix

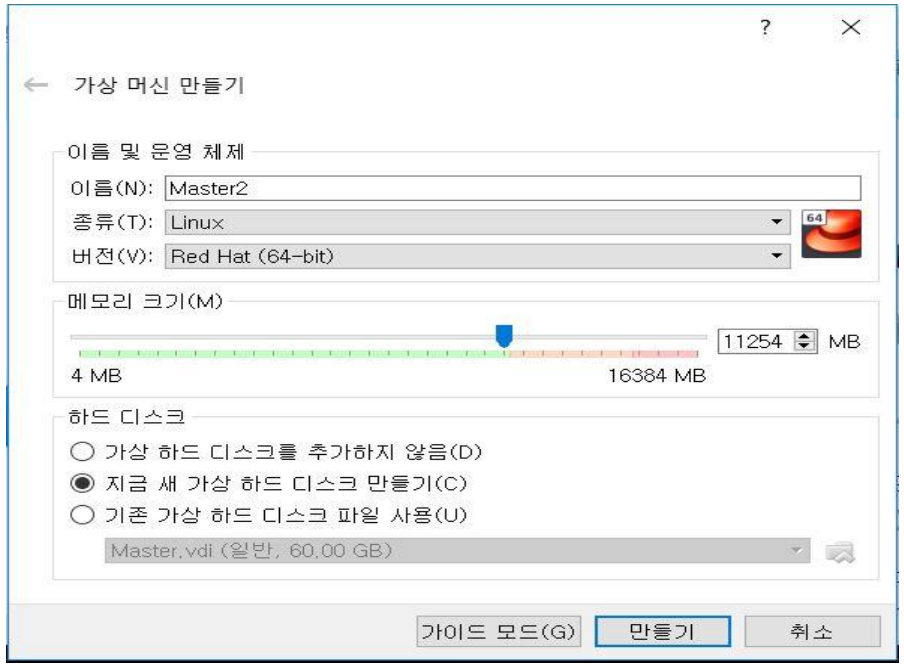
가상머신 생성

가상머신 생성

- 버추얼 박스 실행 -> 새로 만들기 선택



가상머신 생성

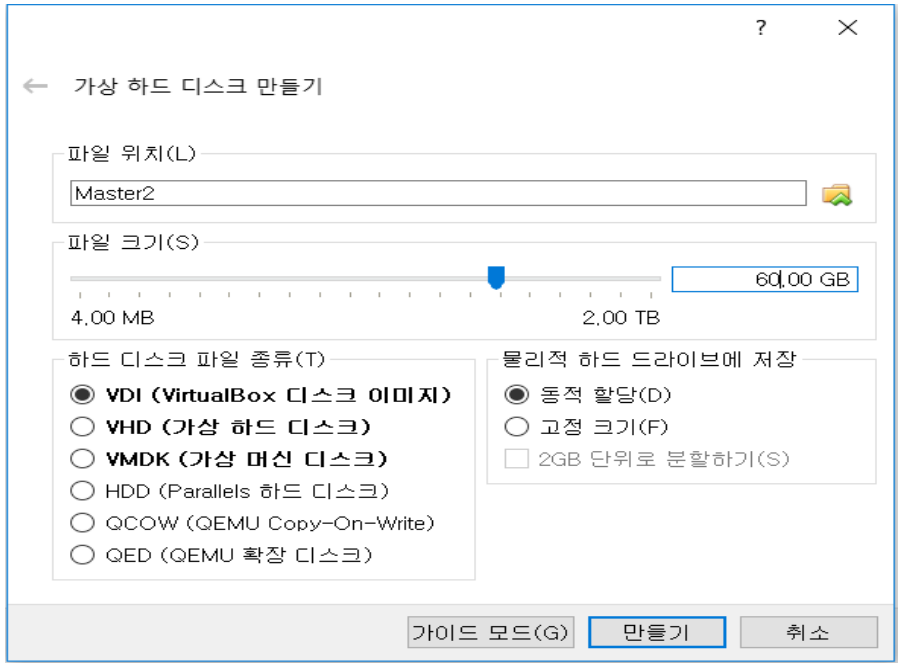


- 종류 : Linux
- 버전 : Red Hat (64-bit)

- 메모리 크기 : 녹색 부분에서 최대치

- (나머지 설정 부분은 그대로 한 후 '만들기' 선택)

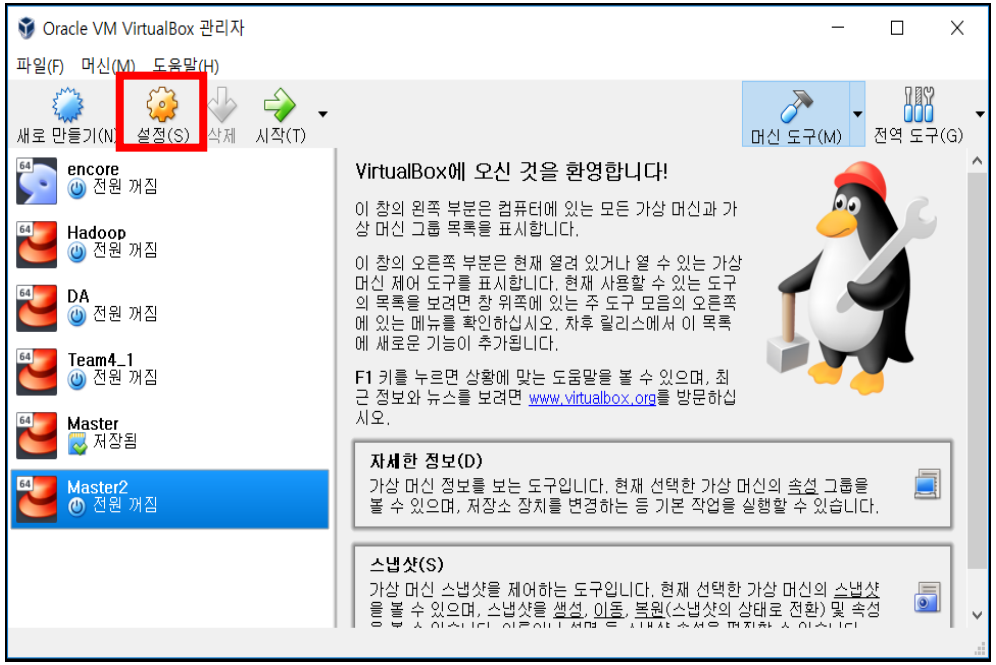
가상머신 생성



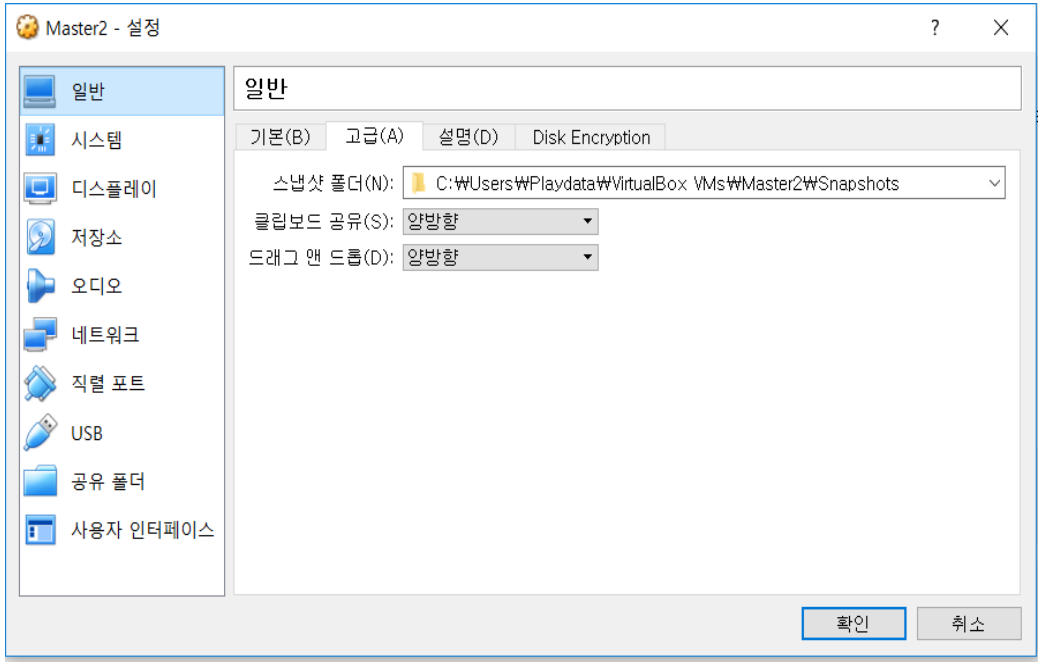
- 파일 크기 : 60 GB
- (나머지 설정 부분은 그대로 한 후 '만들기' 선택)

가상머신 생성

- 가상 머신 생성 후 **설정** 선택



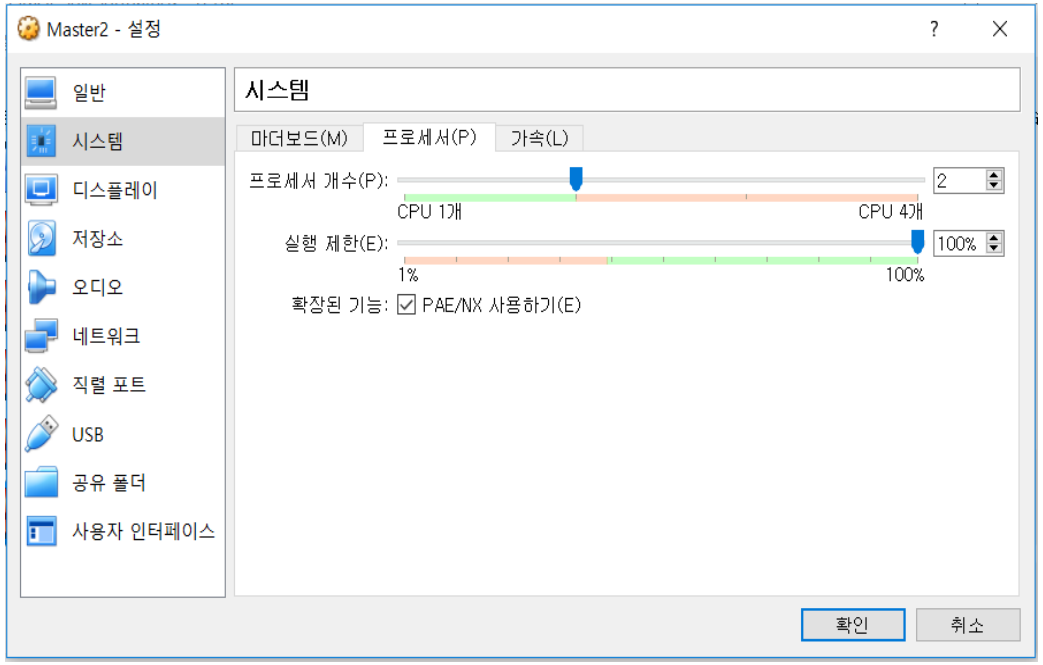
가상머신 생성



- 설정 선택 후

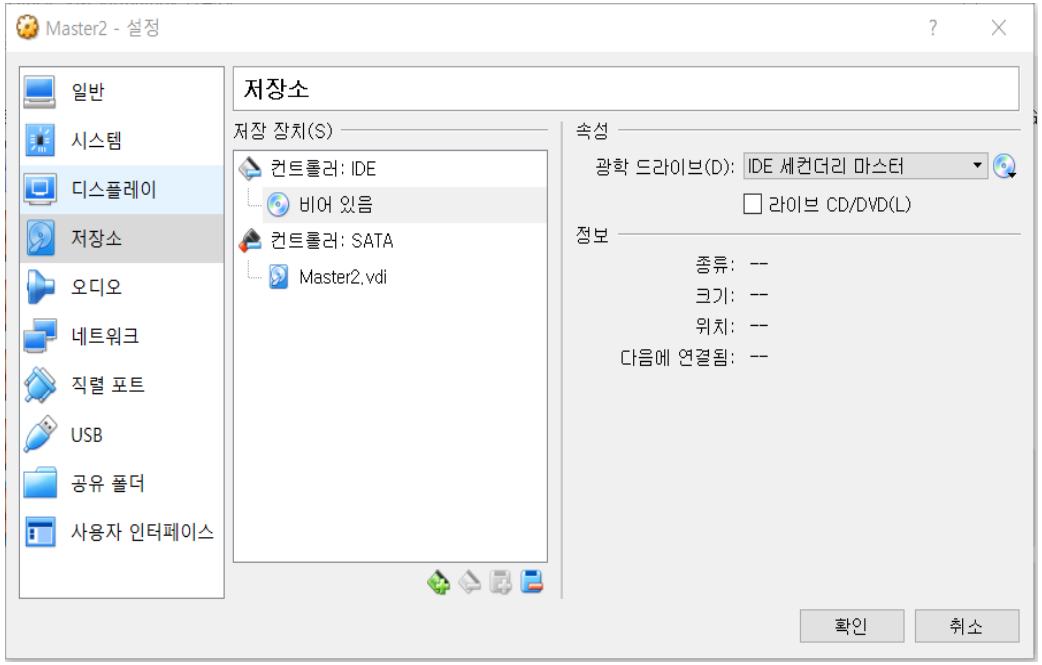
- 일반 메뉴 선택 -> 고급 탭 ->
- 클립보드 공유, 드래그 앤 드롭 모두 양방향 선택

가상머신 생성



- 시스템 -> 프로세서 탭 ->
- 프로세서 개수 : 녹색 부분에서 최대치 (2개)

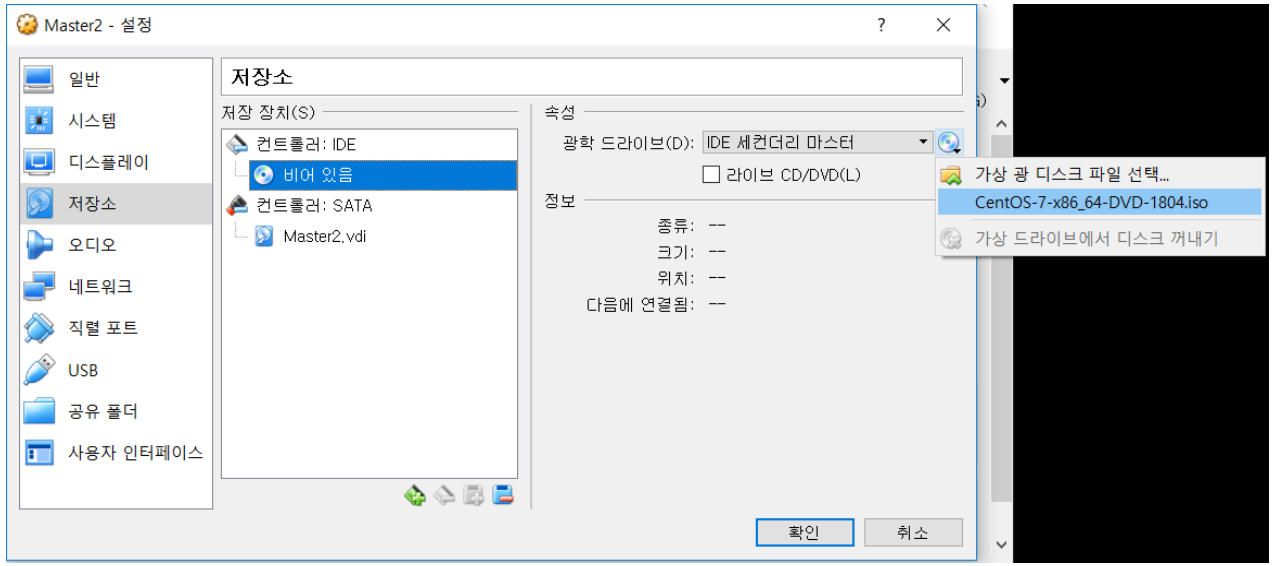
가상머신 생성



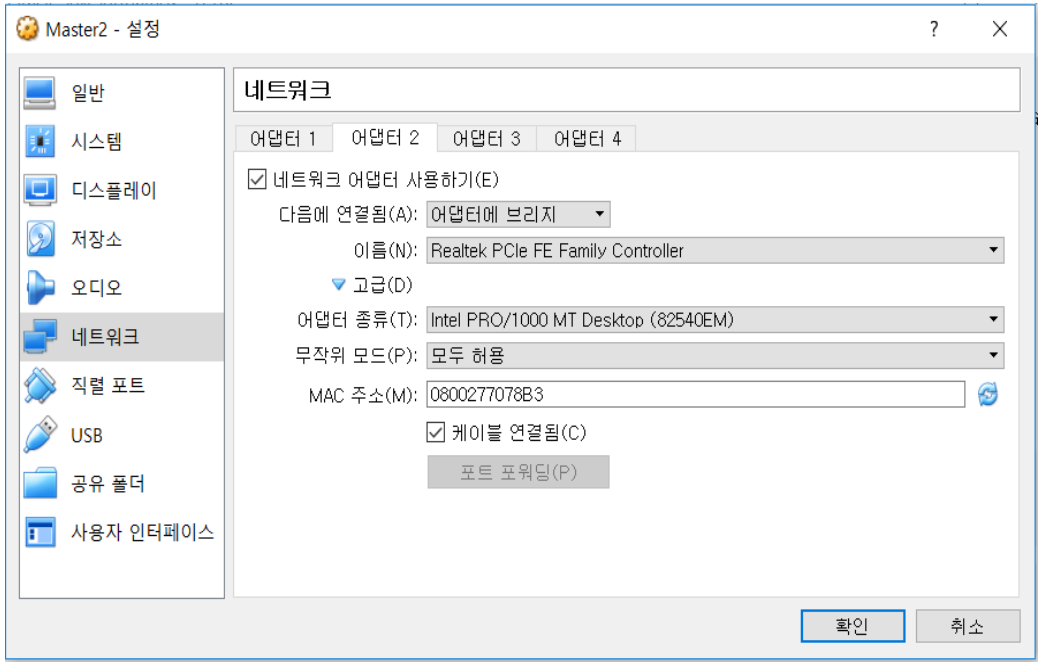
- 저장소 -> 컨트롤러: IDE 밑에 있는 비어있음 선택 ->
- 우측에 있는 속성 밑에 있는 광학 드라이브(D): IDE 세컨더리 마스터 옆에 있는 디스크 모양 아이콘 선택

가상머신 생성

- 디스크 모양 아이콘 선택 후 보유하고 있는
- CentOS 7 iso 파일 선택



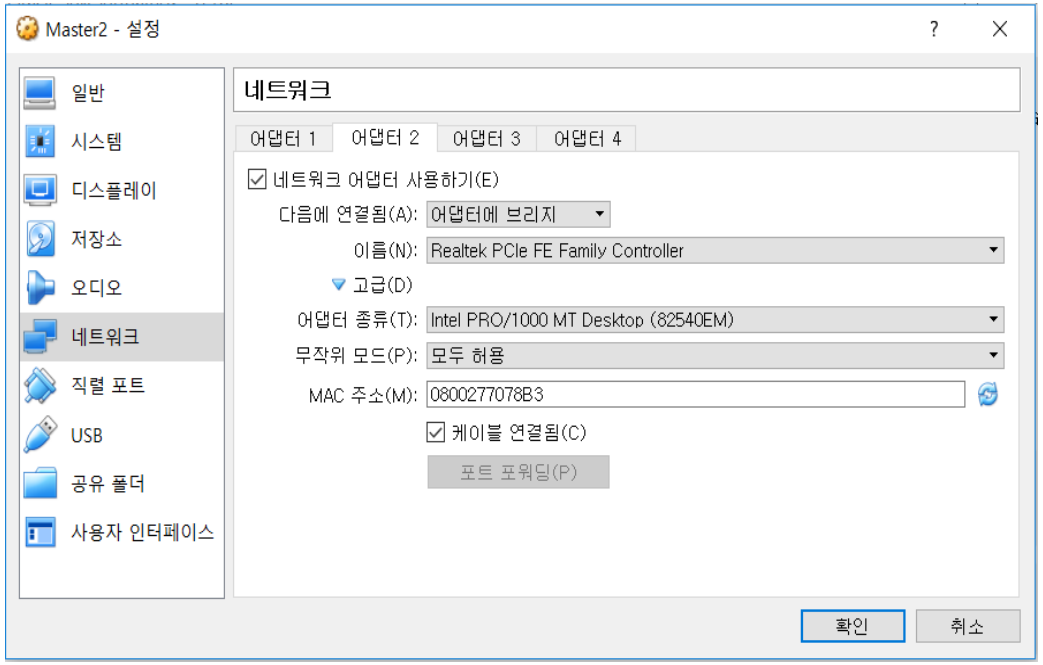
가상머신 생성



- 네트워크 -> 어댑터 1 ->
- 다음에 연결됨 : NAT 선택
- (어댑터 1 나머지 설정부분은 그대로)

- 현재 네트워크는 Windows OS에
- 이더넷으로 연결되어 있는 상황 (LAN)

가상머신 생성

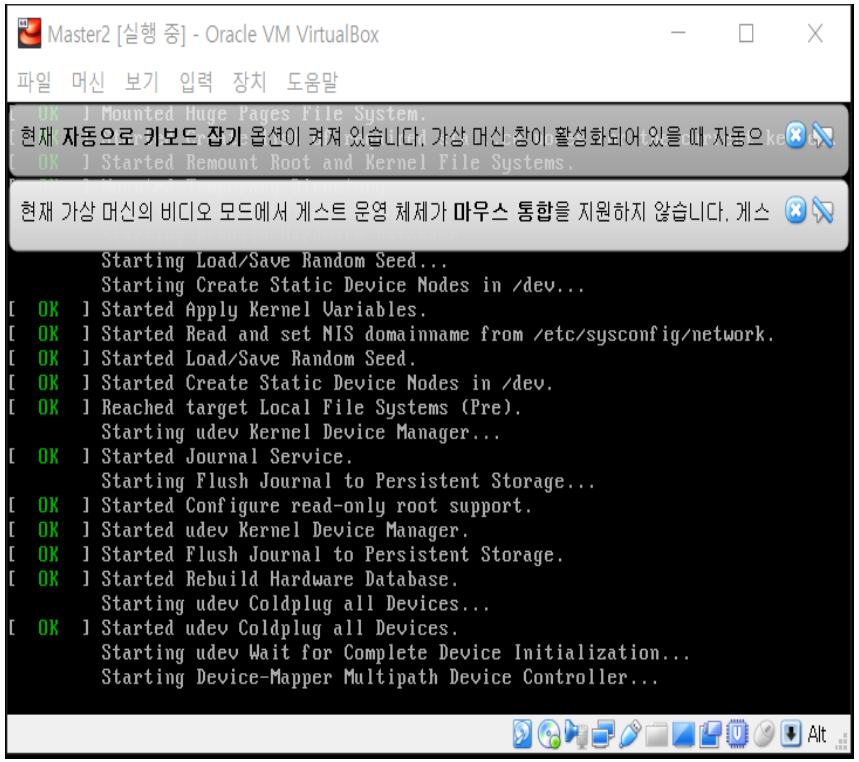


- 어댑터 2 ->
- 다음에 연결됨을 어댑터에 브릿지로 선택,
- 무작위 모드를 모두 허용으로 선택 후 확인

가상머신 생성

- 가상 머신 생성 후 **시작** 선택
- (나머지 설정 부분은 그대로 한 후 '만들기' 선택)

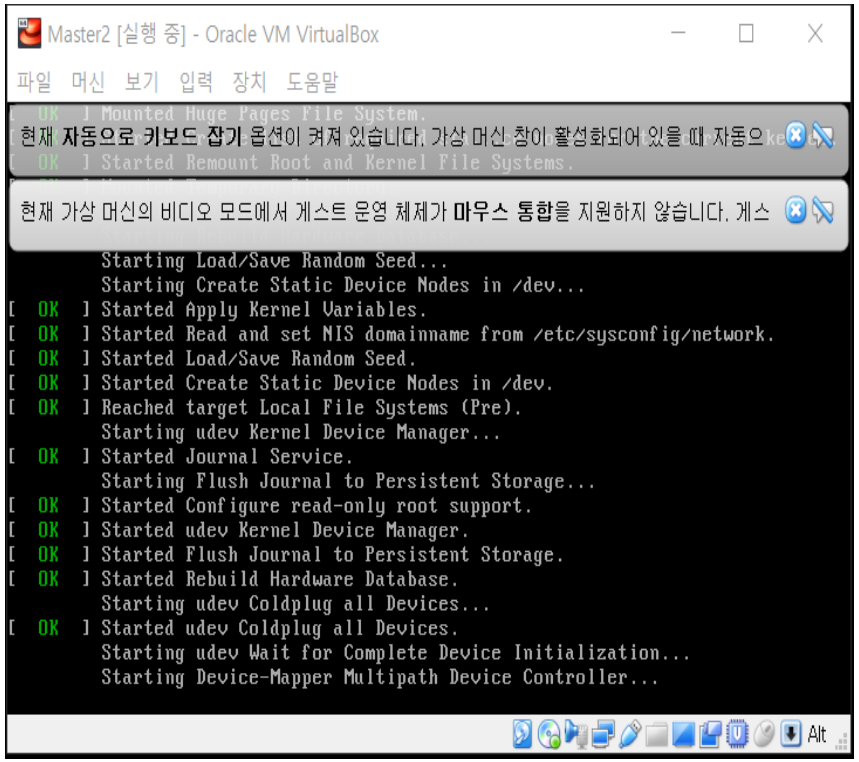
가상머신 생성



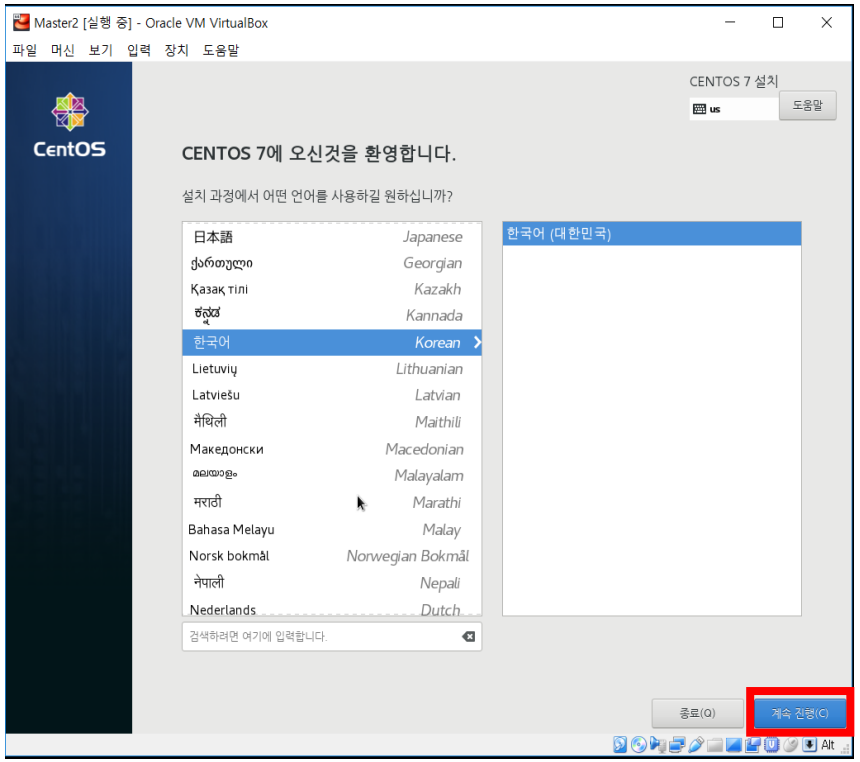
- 시작 후 Install 키보드를 이용해 Install CentOS 7 선택

가상머신 생성

- 설치 진행 중



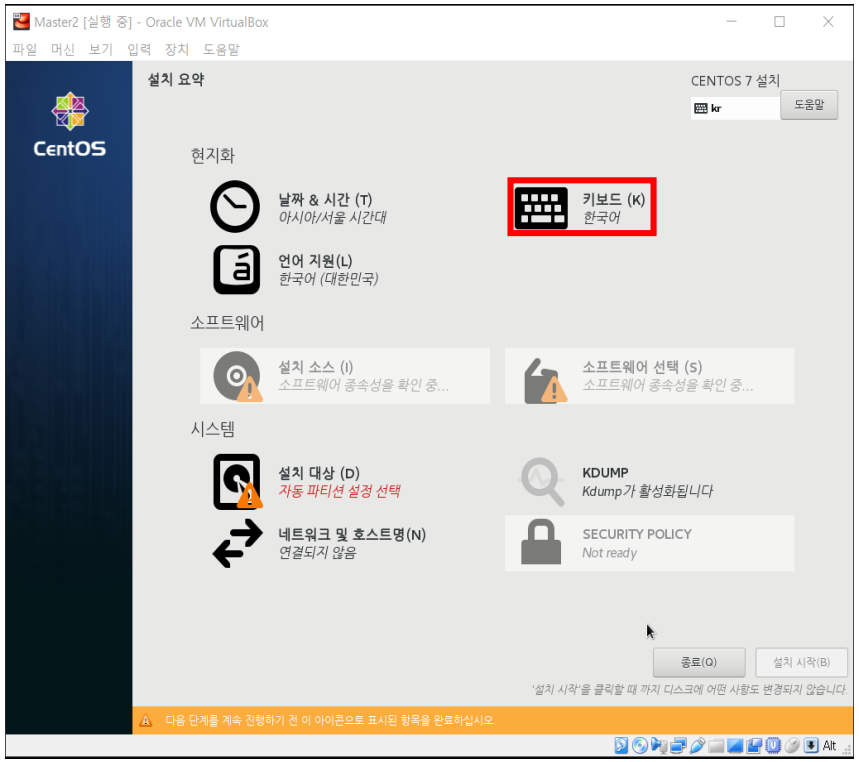
가상머신 생성



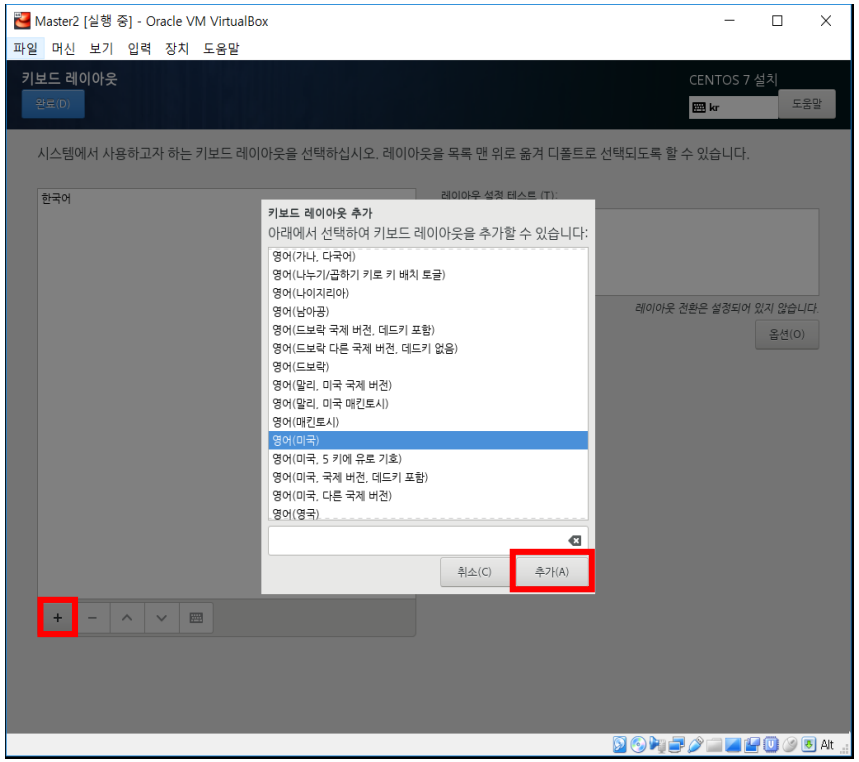
- 언어 설정 : 한국어 선택 후
- 계속 진행 선택

가상머신 생성

- 키보드 선택

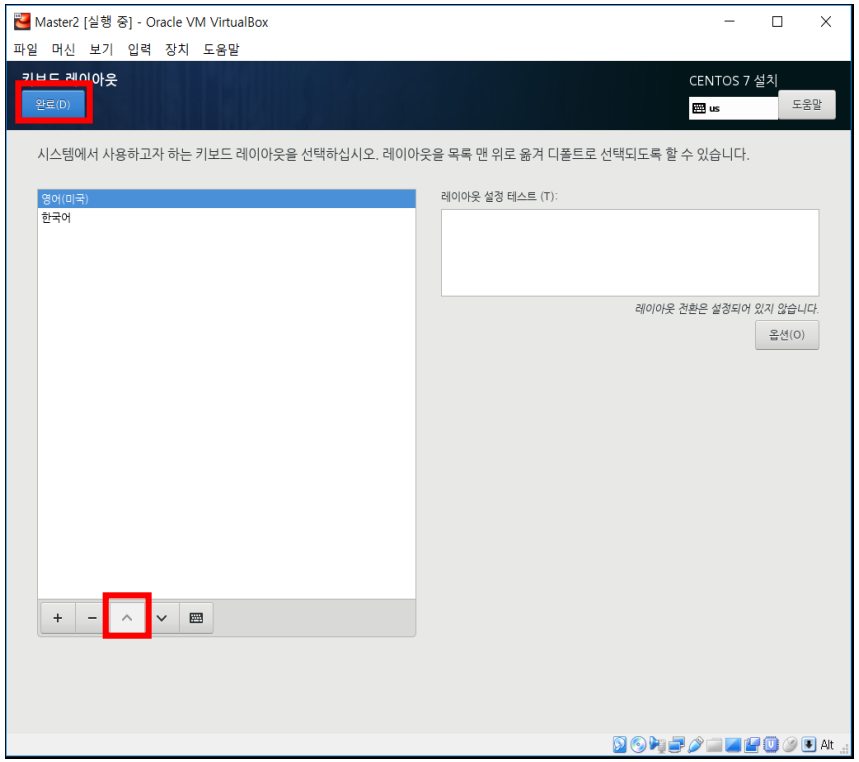


가상머신 생성



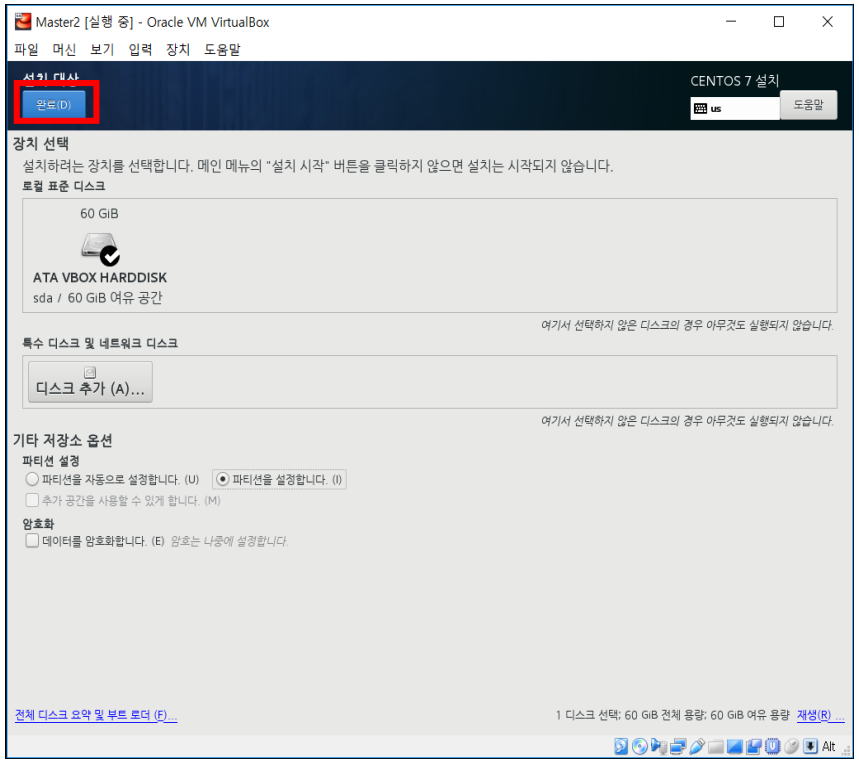
- "+" 선택
- 키보드 레이아웃 중 영어(미국) 선택 후 추가

가상머신 생성



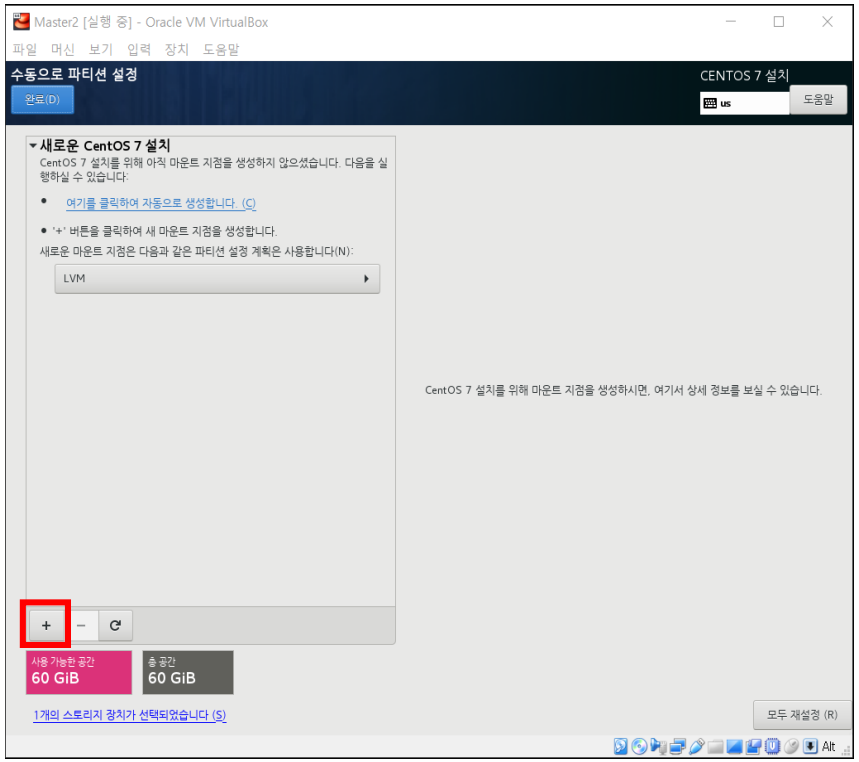
- 위로 이동 버튼을 이용해
- 영어(미국) 키보드 레이아웃을 한국어 위로 이동 후 완료 선택

가상머신 생성



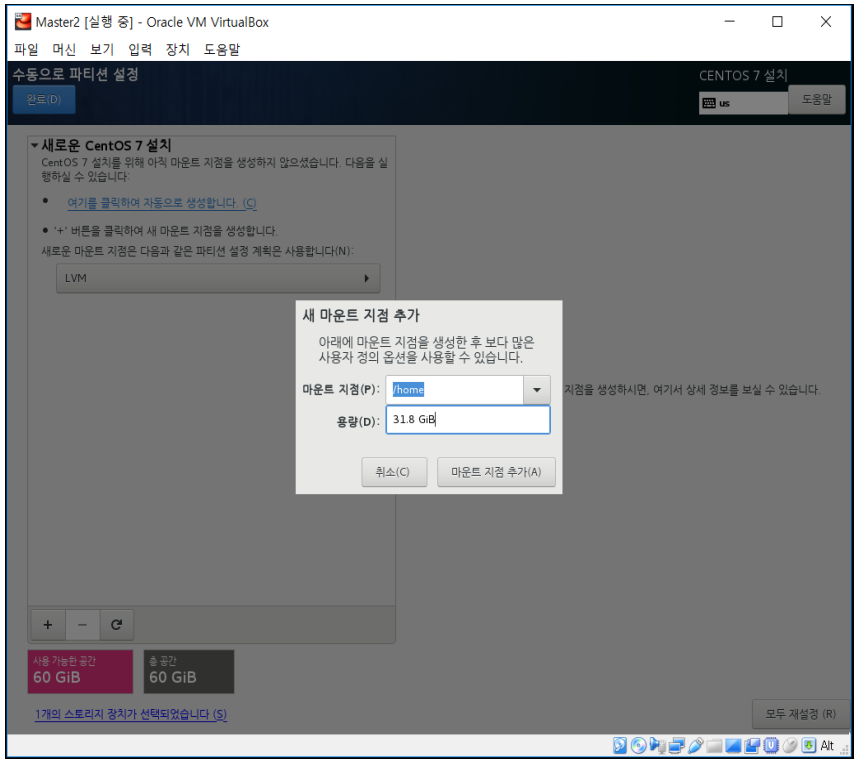
- ATA VBOX HARDDISK를 선택 ->
- 기타 저장소 옵션 밑에 있는
- 파티션을 설정합니다 선택 후 완료

가상머신 생성



- 파티션을 직접 조정할 수 있는 화면
- "+" 버튼을 선택해 디렉토리 별로 용량을 직접 선택

가상머신 생성



- [/home] : 31.8 GiB
- [/boot] : 200 MiB
- [/] : 24 GiB
- [swap] : 4096 MiB

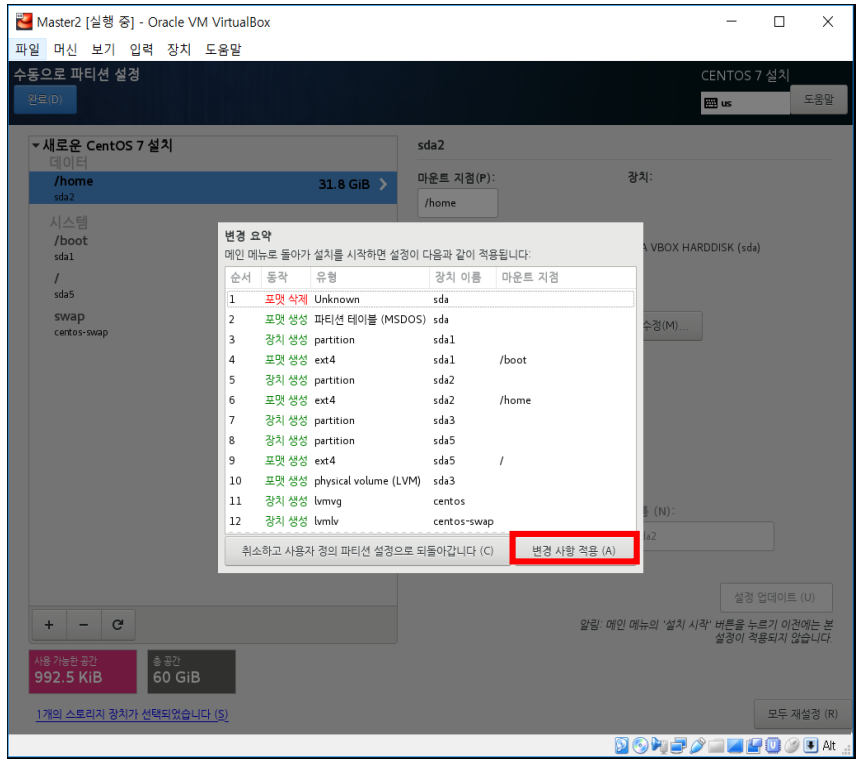
- 입력 한 후 마운트 지점 추가 선택

- [/home], [/boot], [/]는
- 장치 유형을 표준 파티션
- 파일 시스템을 ext4 로 선택 후 완료

가상머신 생성

- [/home], [/boot], [/]는
- 장치 유형을 표준 파티션
- 파일 시스템을 ext4 로 선택 후 완료

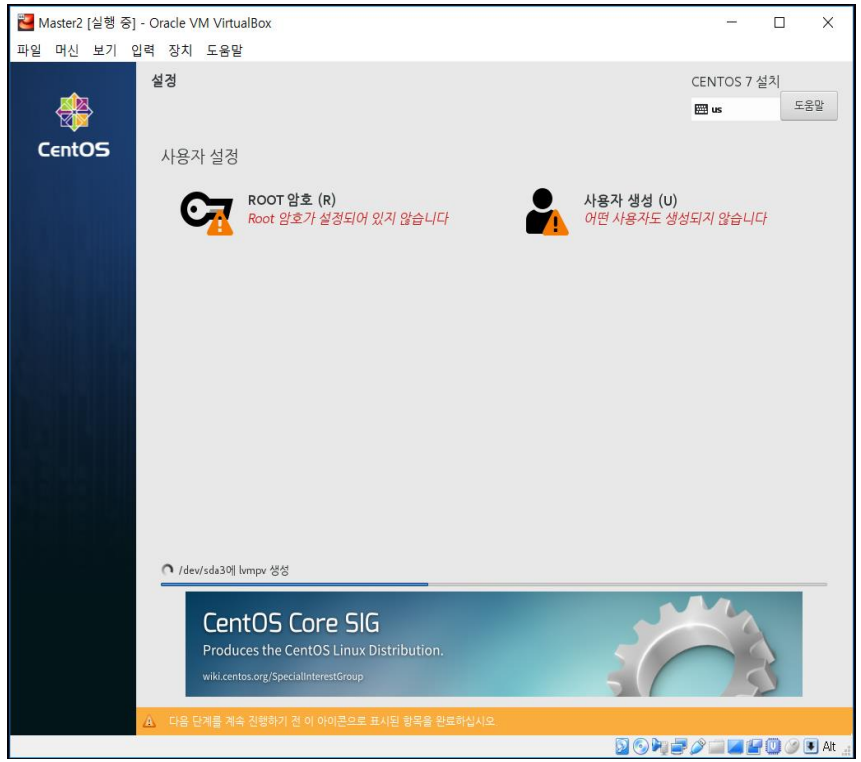
가상머신 생성



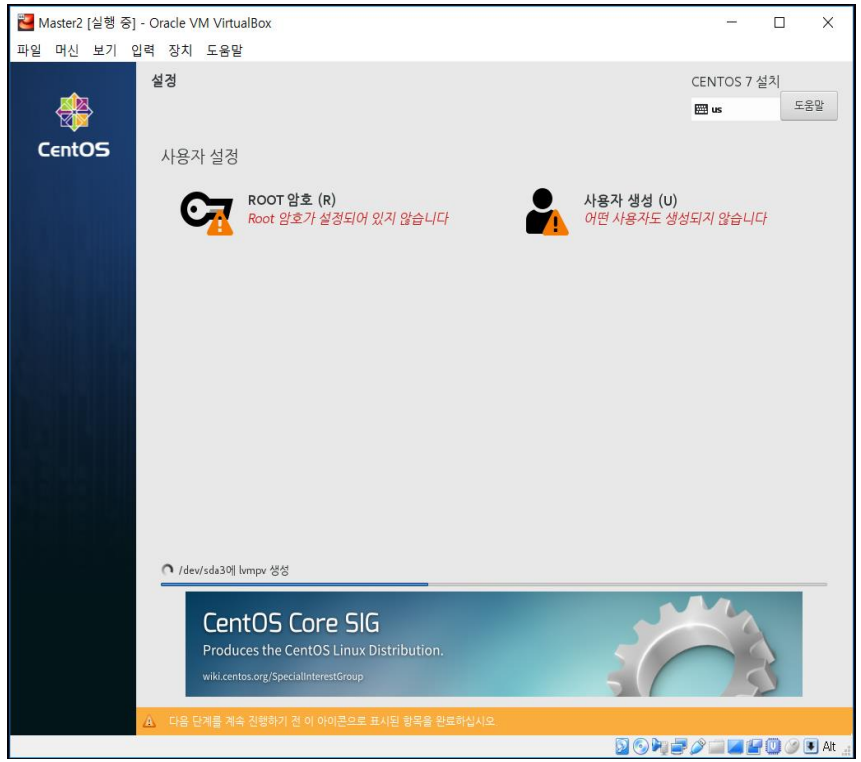
- 완료 후 변경 요약 창에서 변경 사항 적용 선택

가상머신 생성

- 설치 시작 선택

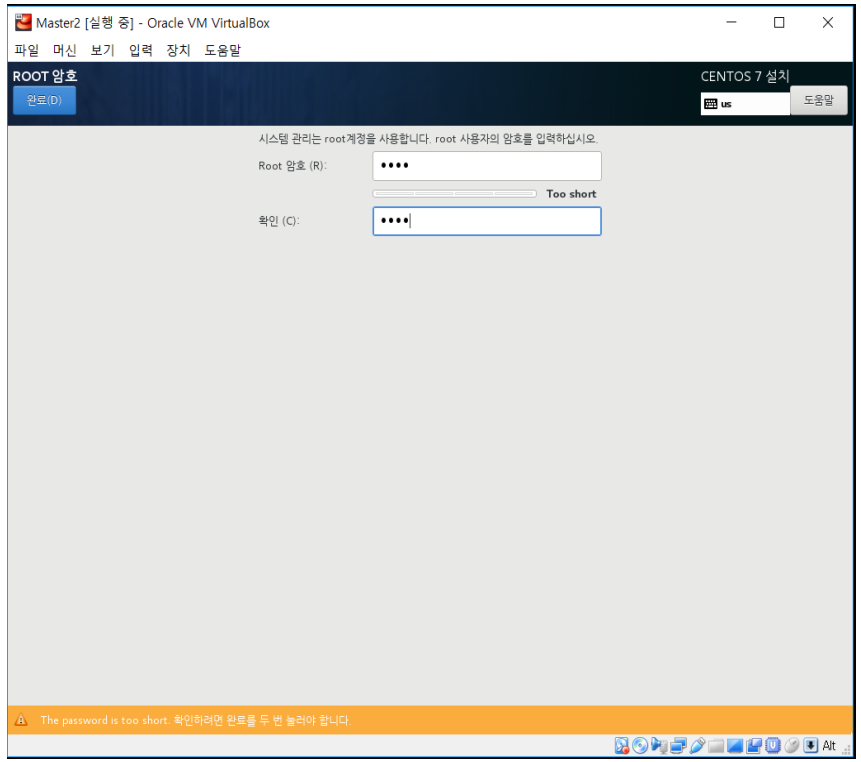


가상머신 생성



- Root 암호, 사용자 생성은
- 사용하는 본인이 알아서 진행하면 될 듯 합니다.

가상머신 생성

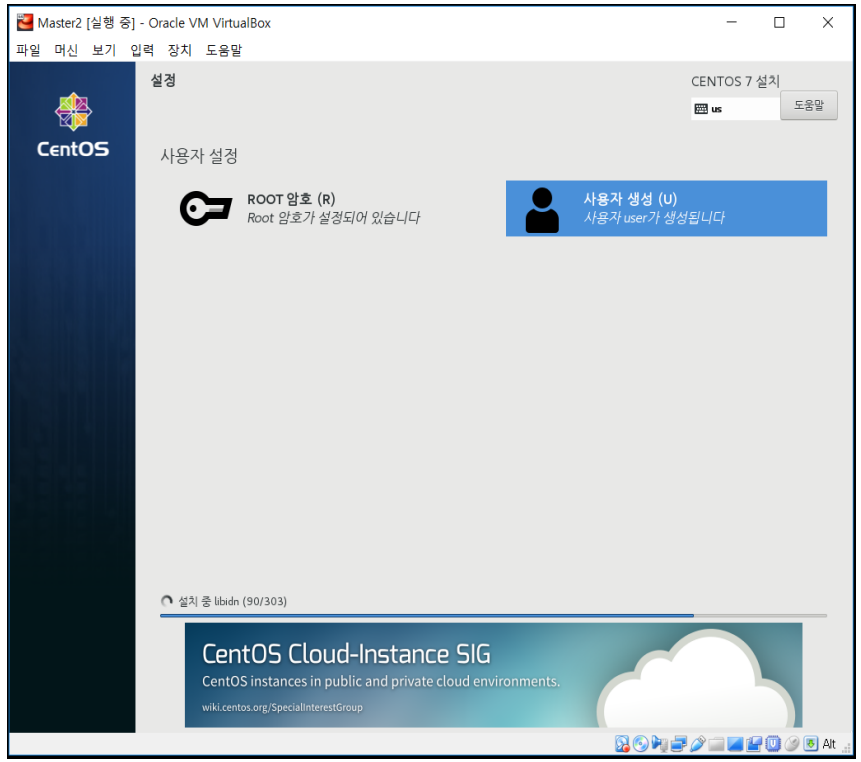


- Root 암호 메뉴 선택 후
- Root 계정에 대한 암호 설정

가상머신 생성

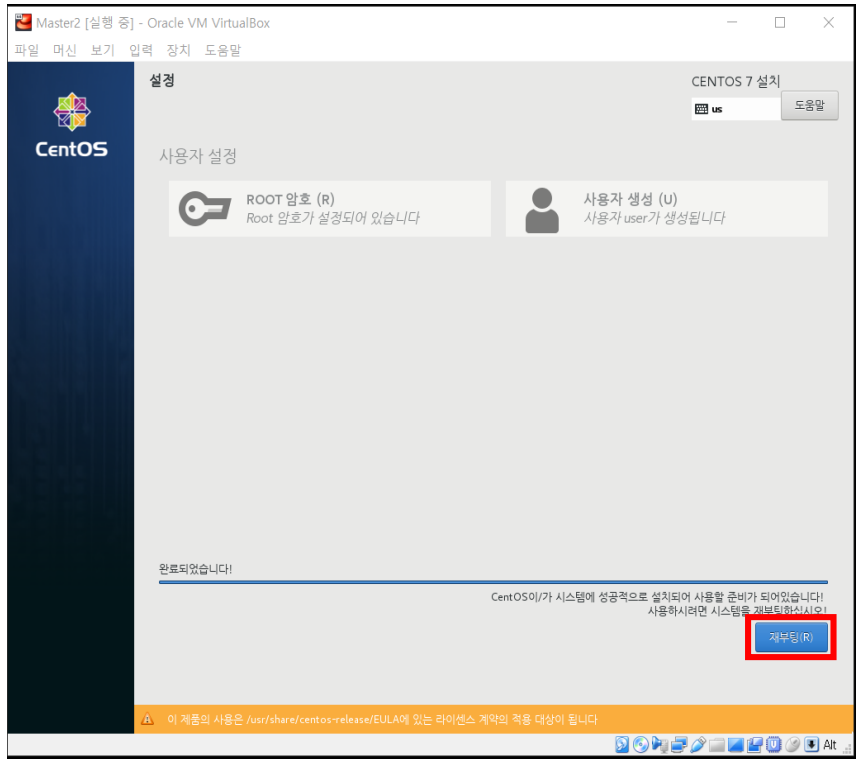
- 사용자 계정 추가 및 암호 설정 진행 후 완료 선택

가상머신 생성



- 두 가지 설정 후 CentOS 7 설치가 계속 진행 됨

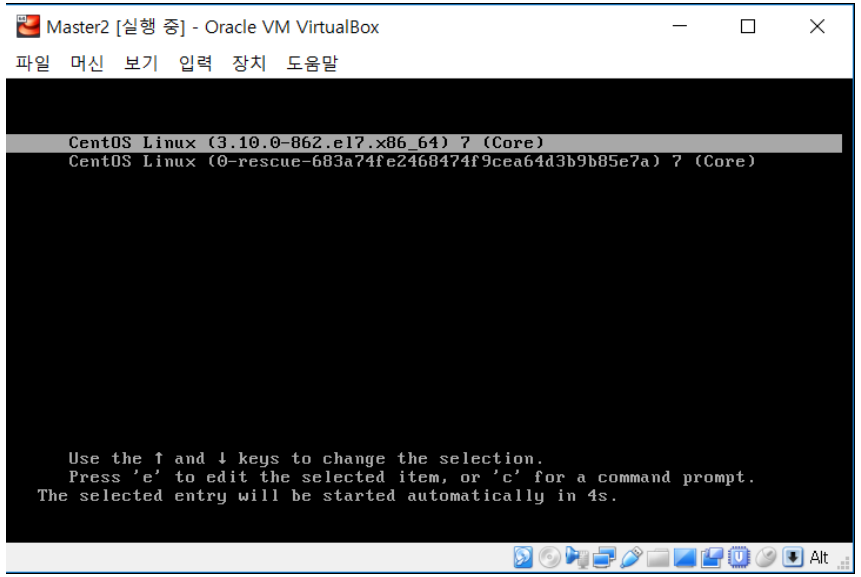
가상머신 생성



- 가상 머신이 생성이 완료된 후 재부팅 선택

가상머신 실행

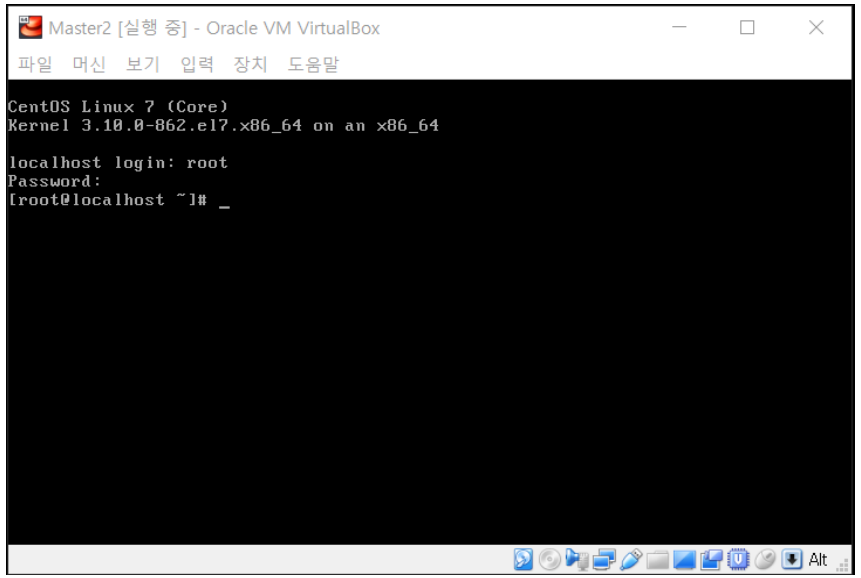
가상머신 실행



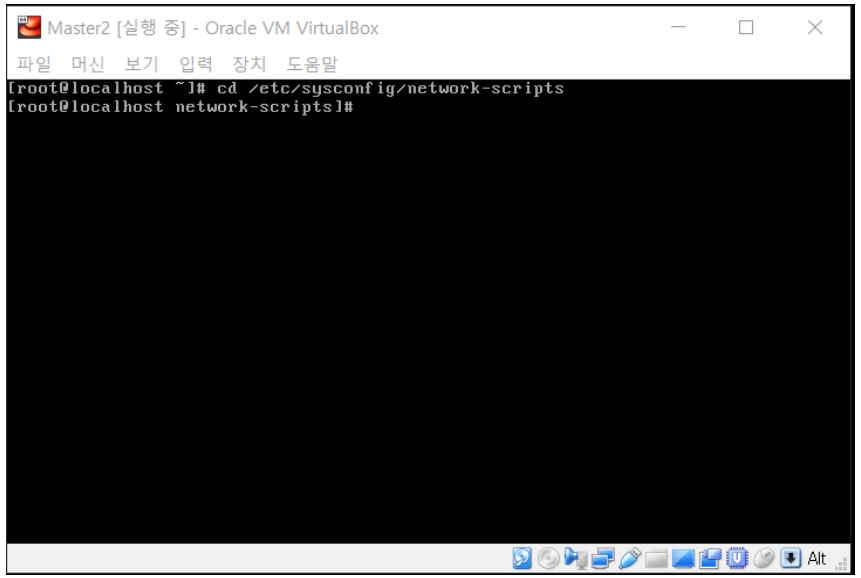
- 두 가지 항목 중
- CentOS Linux (3.10.0-862.e17.x86_64) 7 (Core) 선택

가상머신 실행

- Root 계정으로 CentOS 7에 로그인



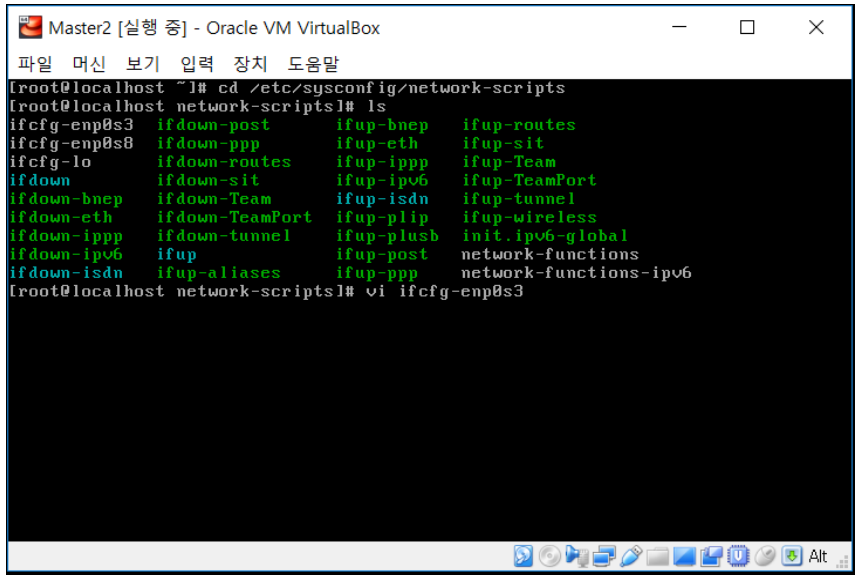
가상머신 실행



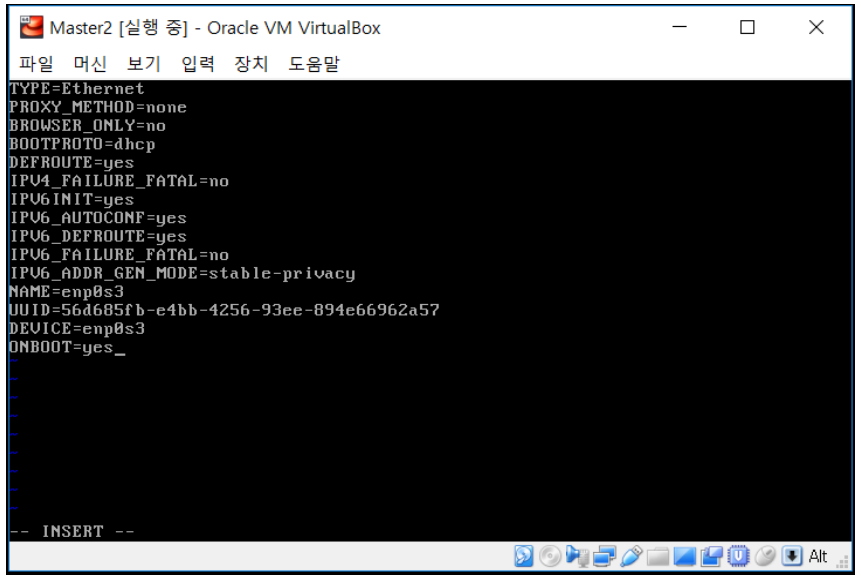
- 네트워크 설정을 위해 root ("~") 디렉토리에서
- /etc/sysconfig/network-scripts 디렉토리로 이동

가상머신 실행

- vi ifcfg-enp0s3 실행



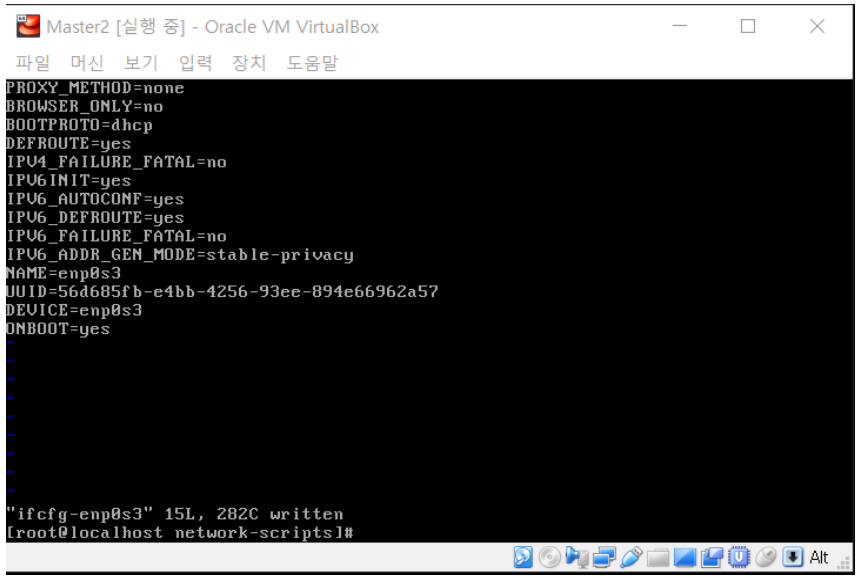
가상머신 실행



- ONBOOT의 항목을 no에서 yes로 변경한 후 저장
- 수정하기 위해 insert 실행
- insert는 키보드로 "i"를 실행하면 가능

- 수정 및 작성한 후
- 저장하는 명령 esc -> : (콜론) -> wq

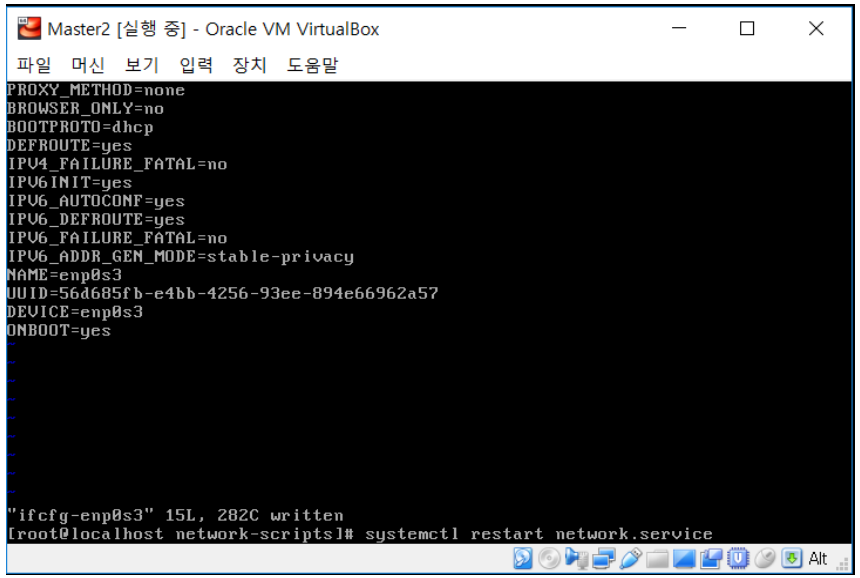
가상머신 실행



- ONBOOT의 항목을 no에서 yes로 변경한 후 저장
- 수정하기 위해 insert 실행
- insert는 키보드로 "i"를 실행하면 가능

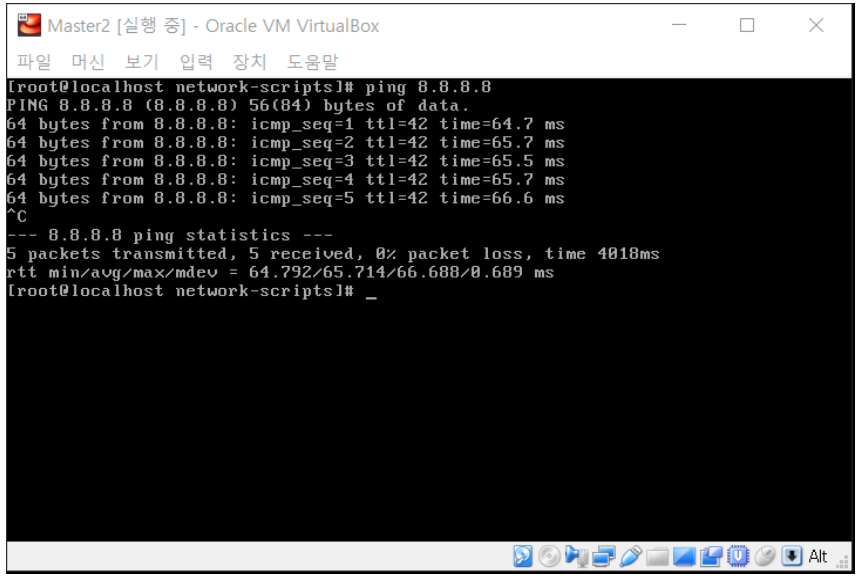
- 수정 및 작성한 후
- 저장하는 명령 esc -> : (콜론) -> wq

가상머신 실행



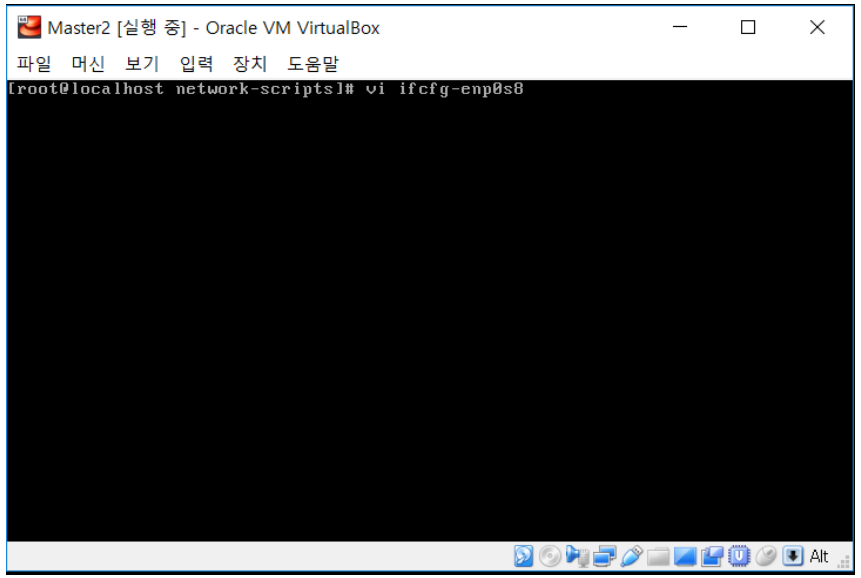
- network 서비스를 재시작하기 위해
- [root@localhost network-scripts]# systemctl restart network.service 입력

가상머신 실행



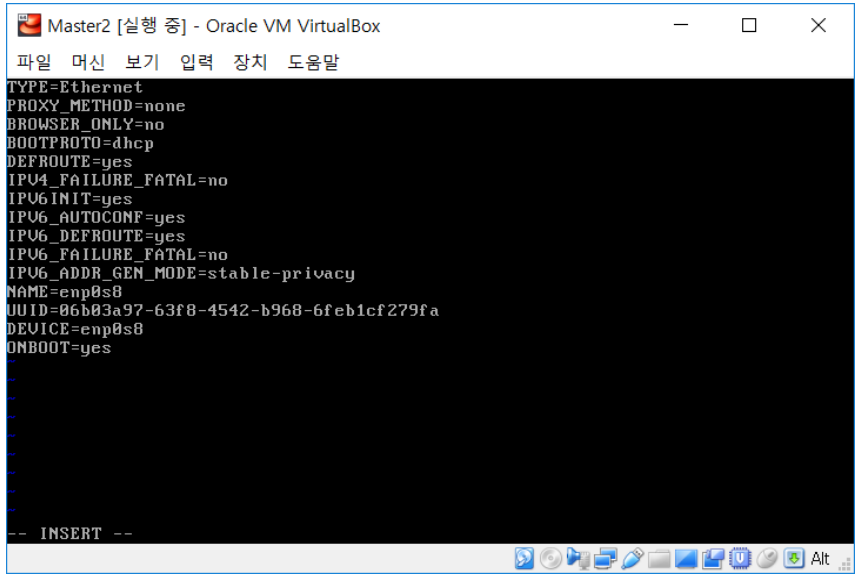
- network 가동 확인을 위해 ping 실행
- [root@localhost network-scripts]# ping 8.8.8.8 입력

가상머신 실행



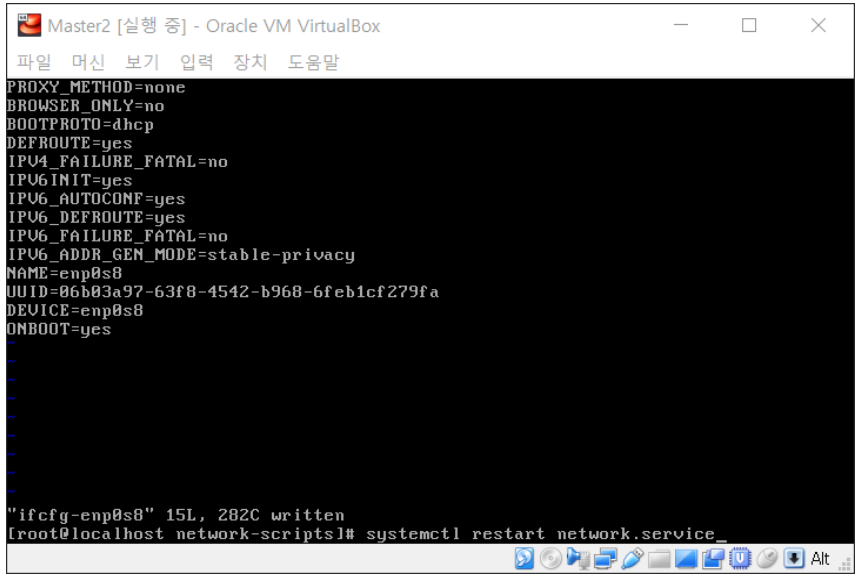
- 고정 IP 설정
- [root@localhost network-scripts]# vi ifcfg-enp0s8

가상머신 실행



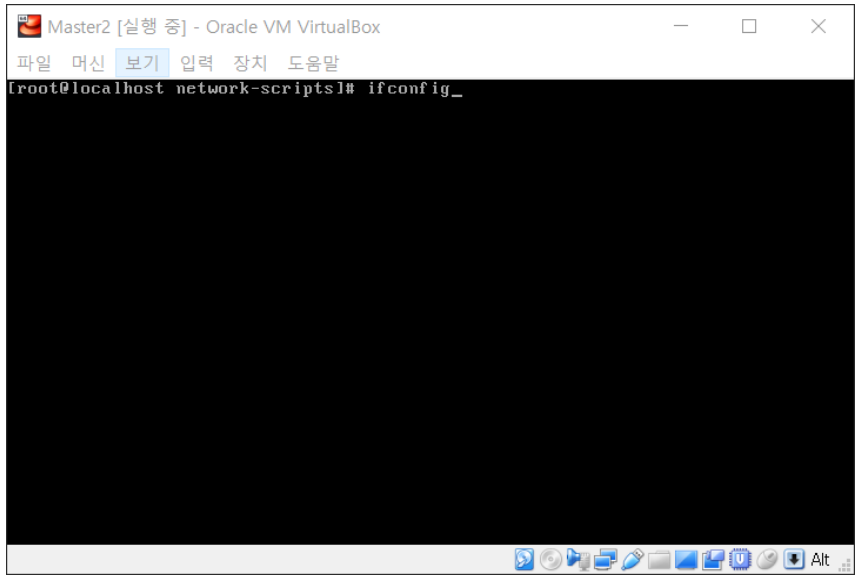
- [root@localhost network-scripts]# vi ifcfg-enp0s8
- ONBOOT부분에서 no를 yes로 변경 후 저장

가상머신 실행



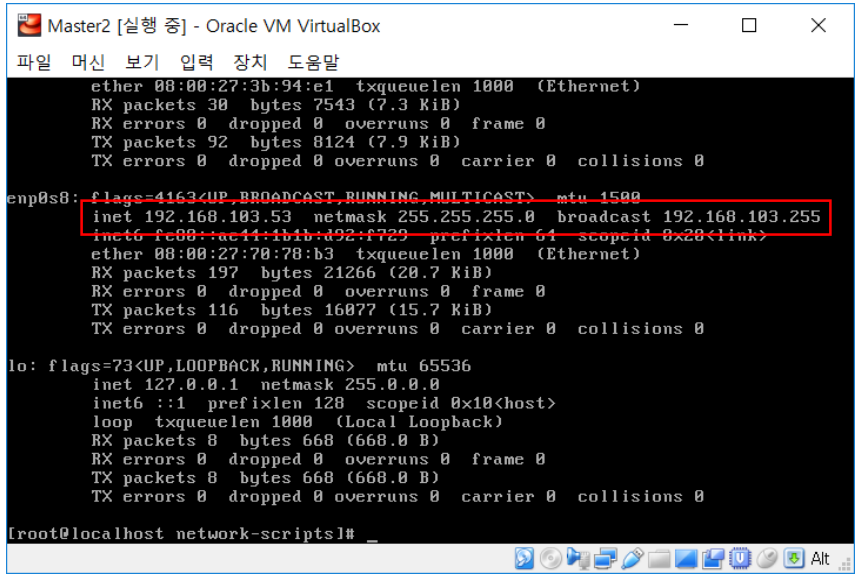
- 네트워크 서비스 재시작
- [root@localhost network-scripts]# systemctl restart network.service

가상머신 실행



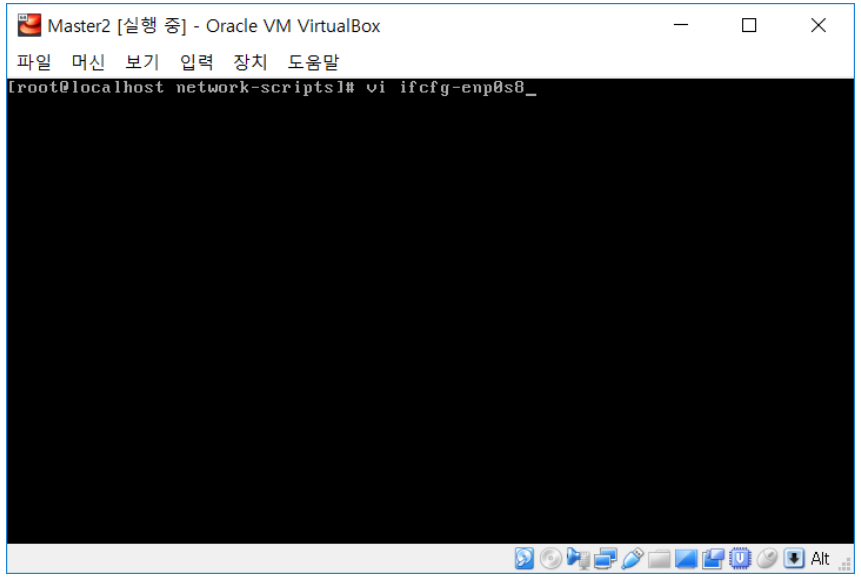
- ip 확인하기 위해
- [root@localhost network-scripts]# ifconfig 입력

가상머신 실행



- ip 확인하기 위해
- [root@localhost network-scripts]# ip addr 입력

가상머신 실행

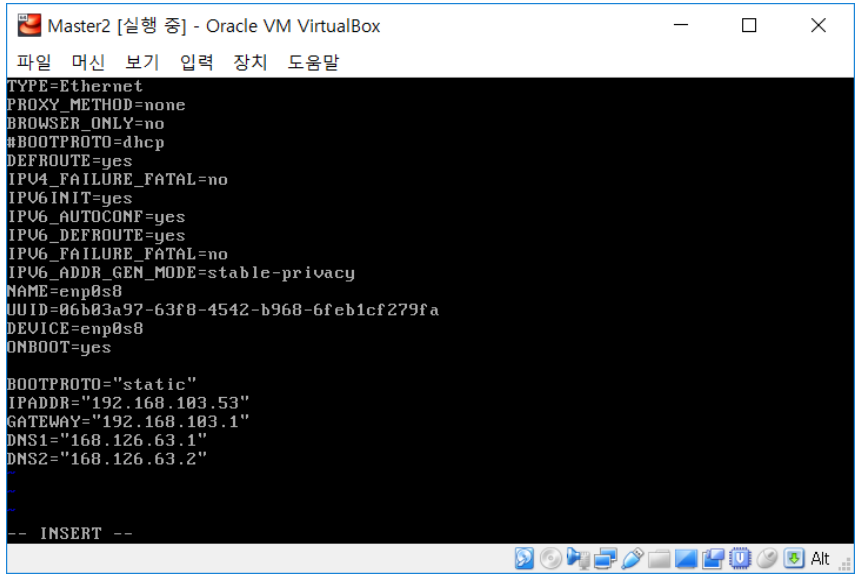


- 현재 ifcfg-enp0s8에 할당 받은
- ip는 192.168.103.53

- 이 ip를 고정 ip로 설정하기 위해 다시

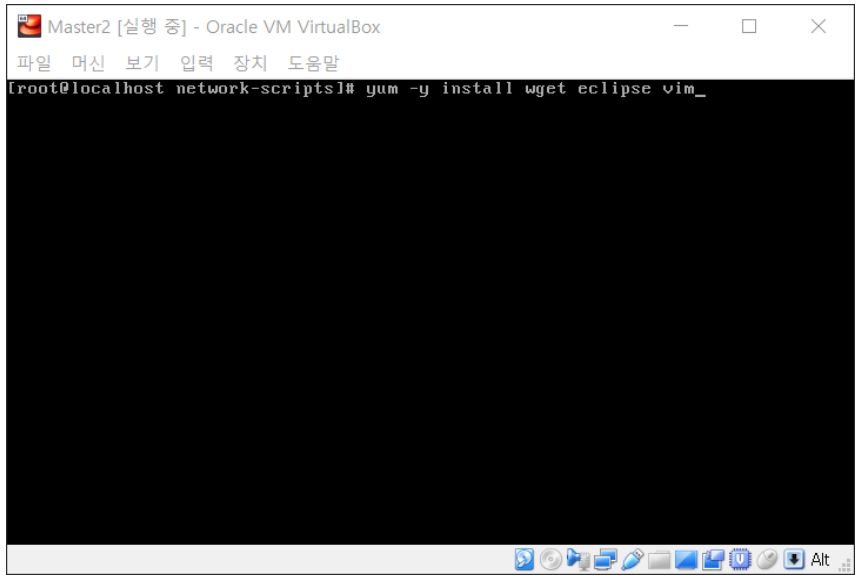
- [root@localhost network-scripts]# vi ifcfg-enp0s8 입력

가상머신 실행



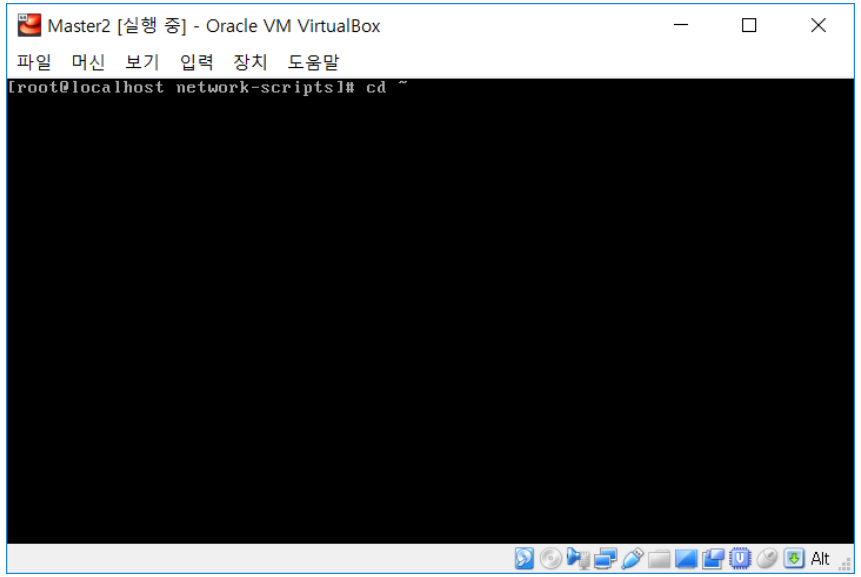
- 외부에서 접속하기 위해 고정 IP 설정
- 기존 BOOTPROTO=dhcp를 주석처리하고
- ONBOOT 하단에 다음과 같이 작성
- BOOTPROTO="static"
- IPADDR="잠시 전 ifcfg-enp0s8에 할당받은 ip 입력"
- GATEWAY="ifcfg-enp0s8에 할당받은 ip 자리 중 앞 세 위치 자리 값 입력" - ex) 192.168.103.1
- DNS1="168.126.63.1"
- DNS2="168.126.63.2"
- 입력 후 저장

가상머신 실행



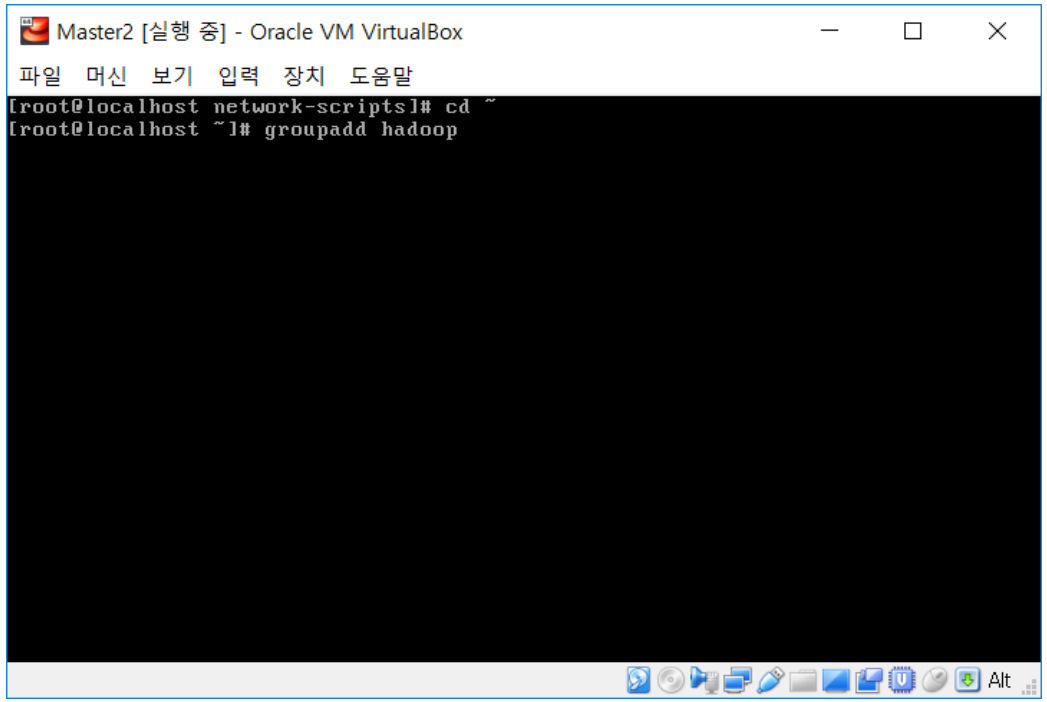
- 필요한 프로그램 설치
- [root@localhost network-scripts]# yum –y install wget vim

가상머신 실행



- ~ 디렉토리로 이동
- [root@localhost network-scripts]# cd ~

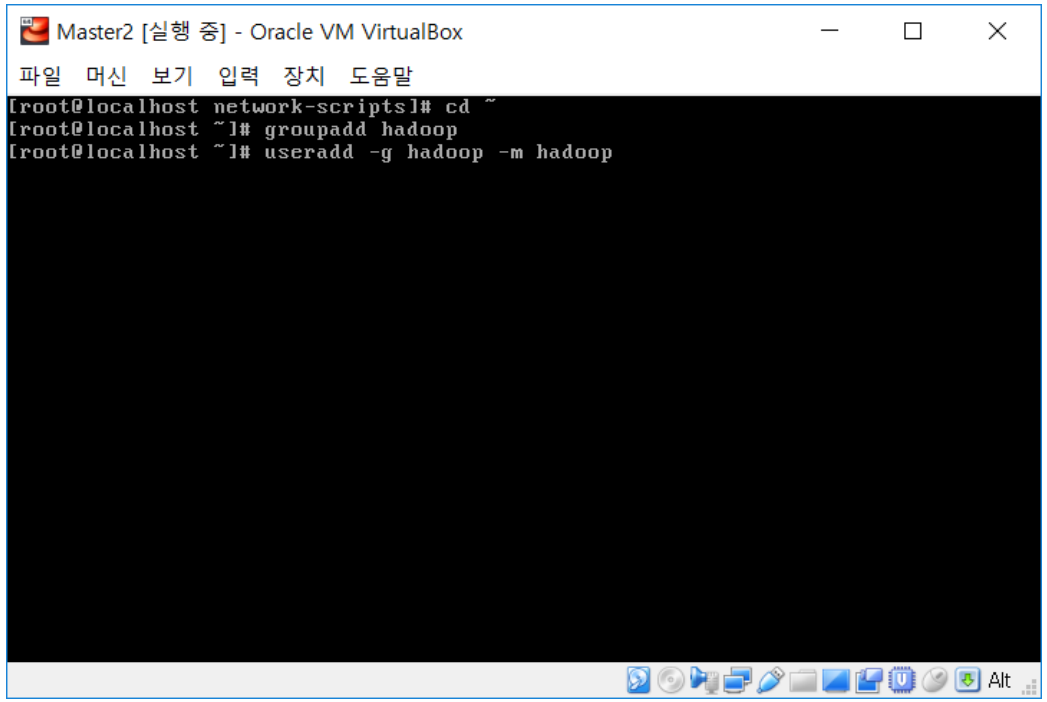
가상머신 실행



hadoop 그룹 생성

[root@localhost ~]# groupadd hadoop

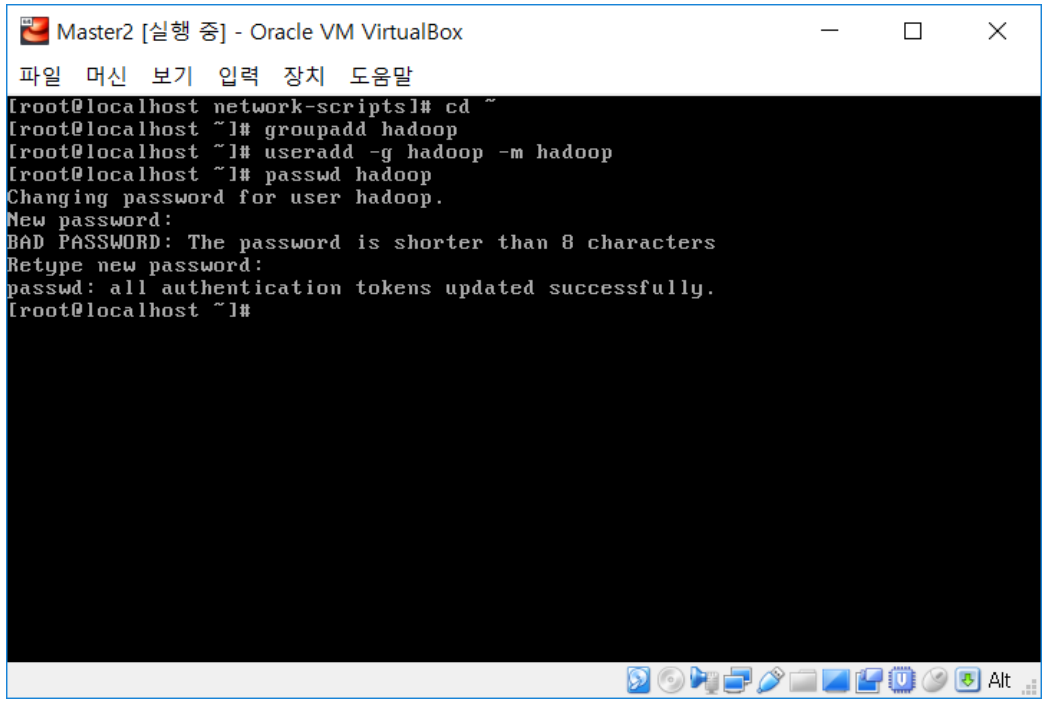
가상머신 실행



hadoop 그룹에 속하는 hadoop 이름의 사용자 계정 생성

[root@localhost ~]# useradd –g hadoop –m hadoop

가상머신 실행



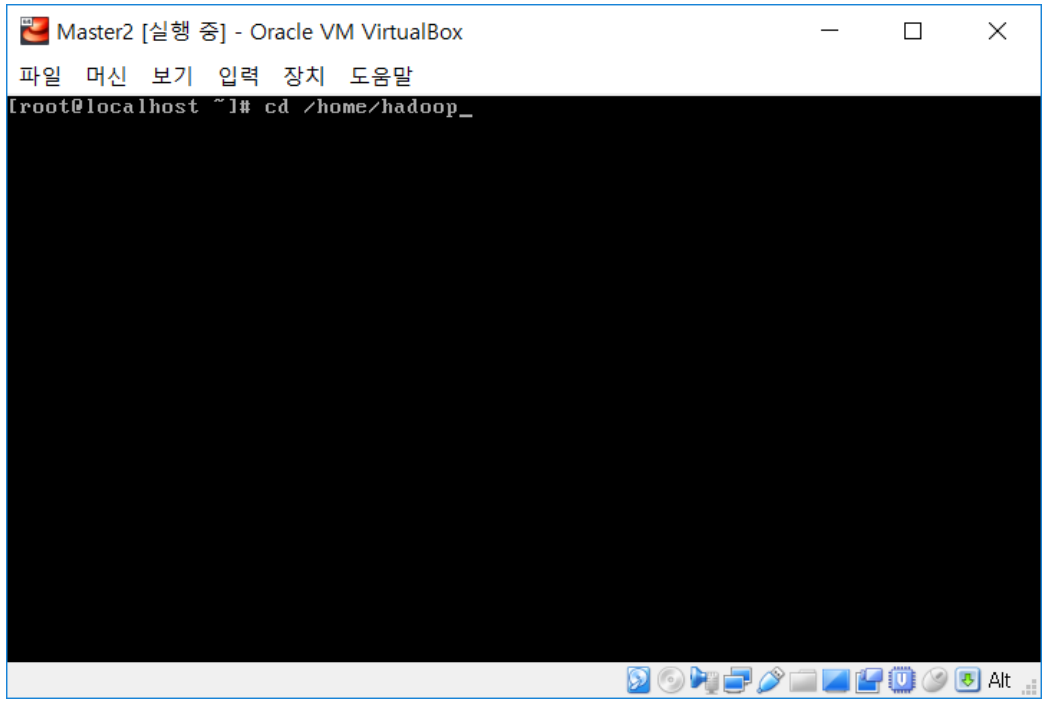
hadoop 계정의 password 생성

[root@localhost ~]# passwd hadoop X

이후 본인이 hadoop 계정에 생성할 password 입력 (확인 입력까지 총 2번)

hadoop 계정의 password 생성 하지 말기

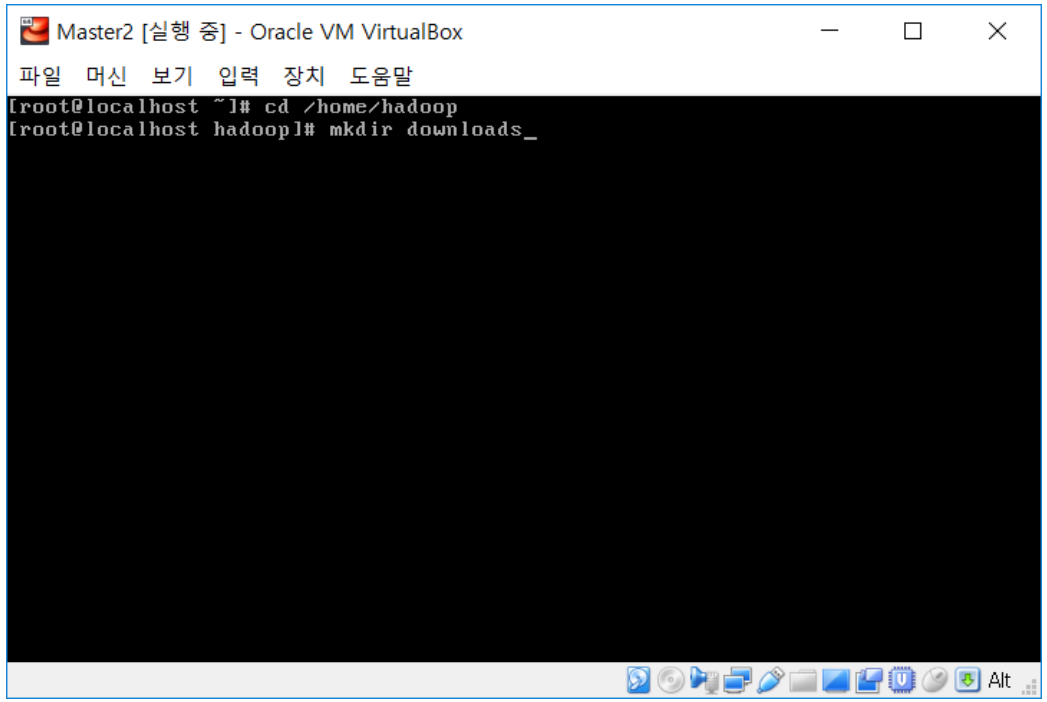
가상머신 실행



hadoop 디렉토리 이동

[root@localhost ~]# cd /home/hadoop

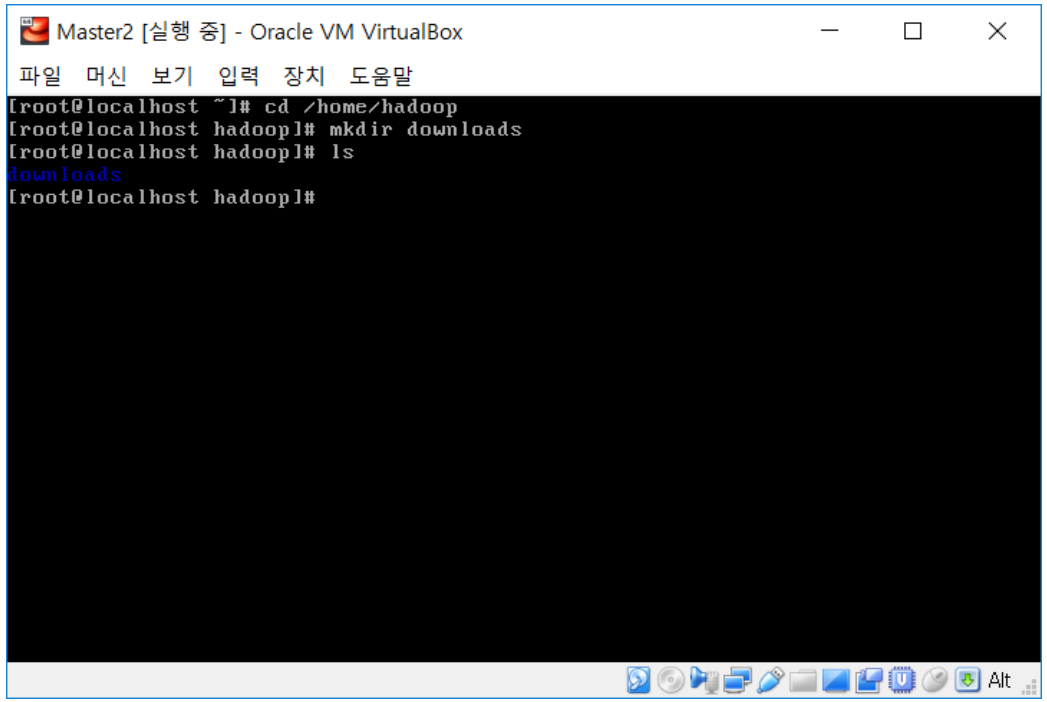
가상머신 실행



hadoop 디렉토리 하위에 downloads 디렉토리 생성

[root@localhost hadoop]# mkdir downloads

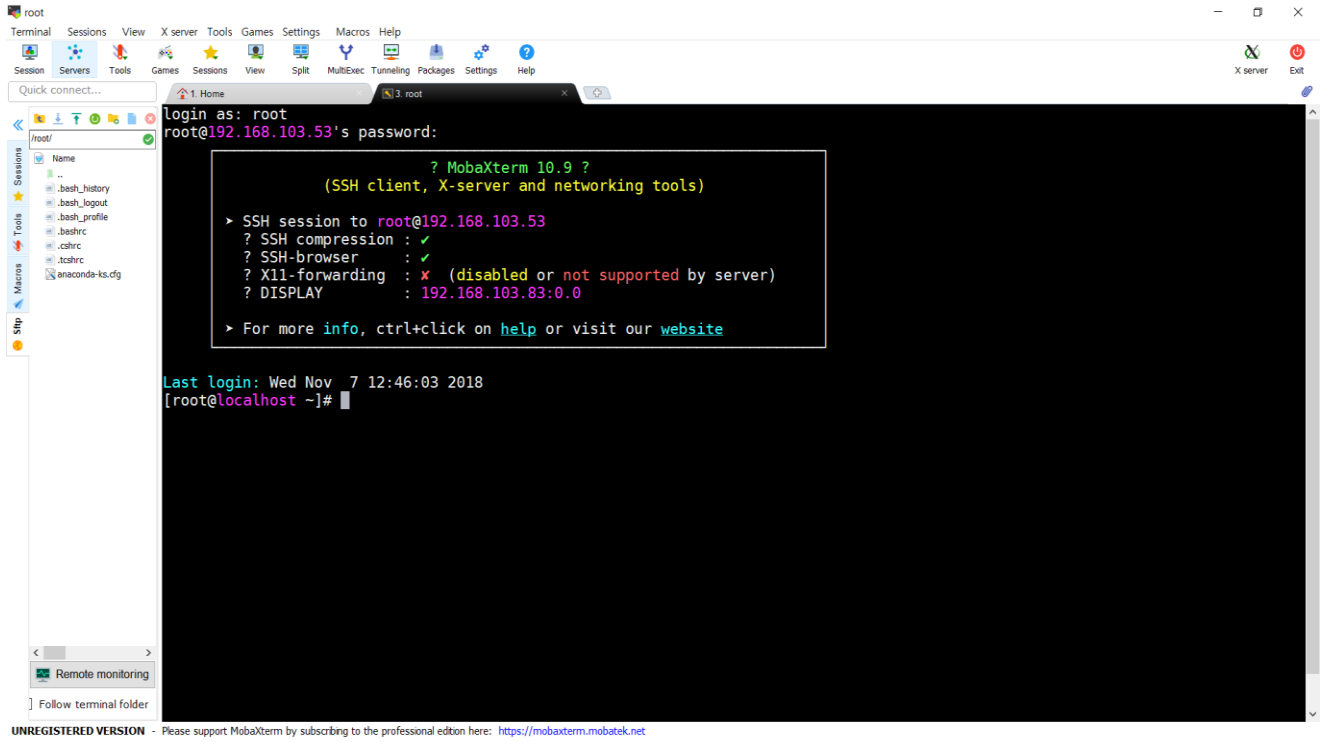
가상머신 실행



hadoop 디렉토리 하위에 downloads 디렉토리 생성

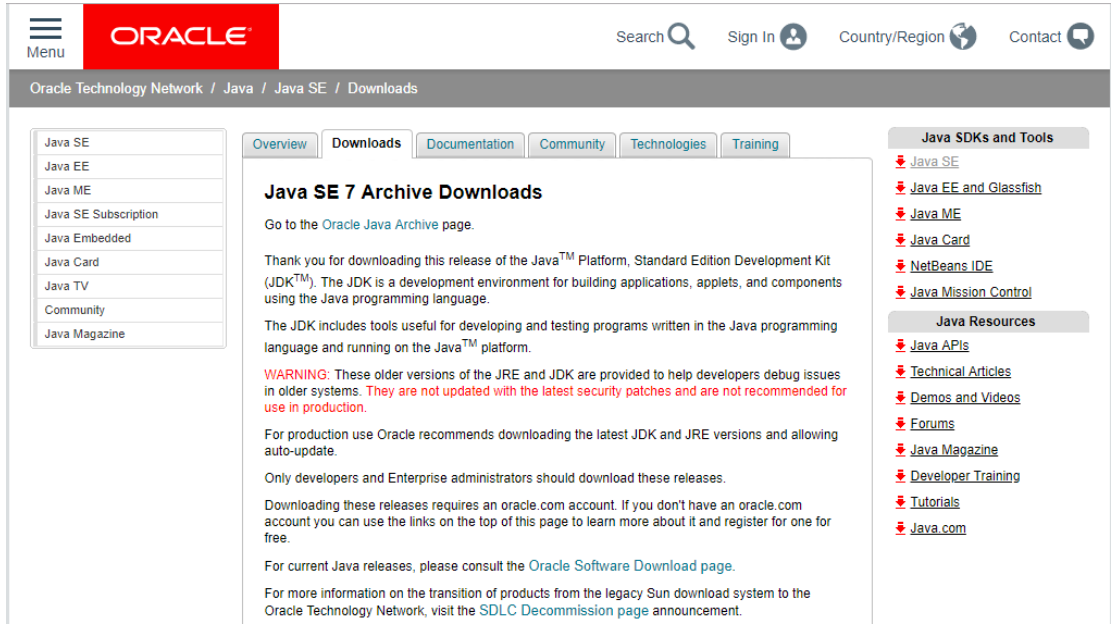
[root@localhost hadoop]# mkdir downloads

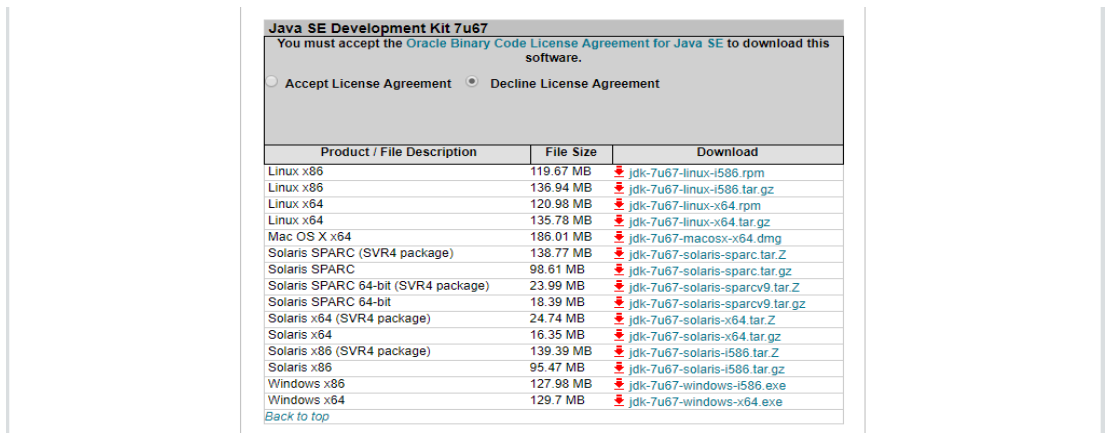
가상머신 실행



Mobaxterm으로 가상 머신 접속

가상머신 실행



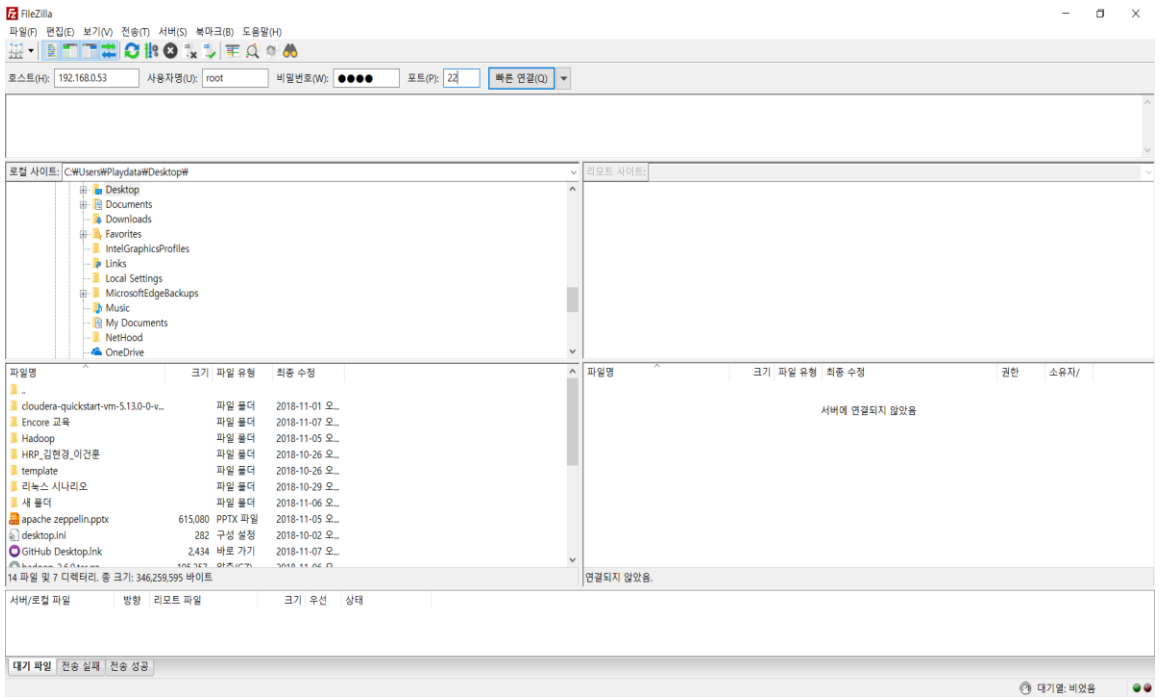


Jdk 1.7 (jdk-7u67-linux-x64.tar.gz)을 받기 위해
오라클 홈페이지에서 다운로드 받은 후
"Filezilla"를 통해 /home/hadoop/downloads에 업로드

Url
https://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html

File
Java SE Development Kit 7u67
Linux x64 (jdk-7u67-linux-x64.tar.gz, 135.78MB)

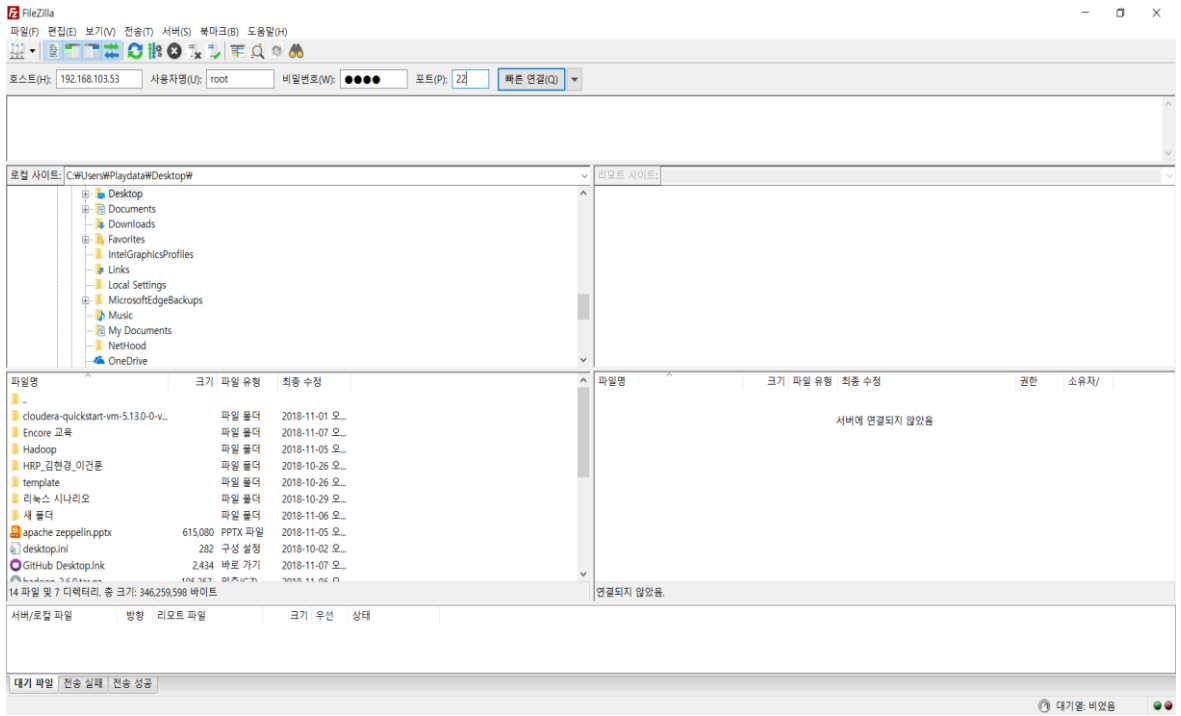
가상머신 실행



Filezilla 실행

호스트: 고정 ip (ifcfg-enp0s8의 고정 ip)
사용자명 : root
비밀번호 : 사용자명의 비밀번호
포트 : 22

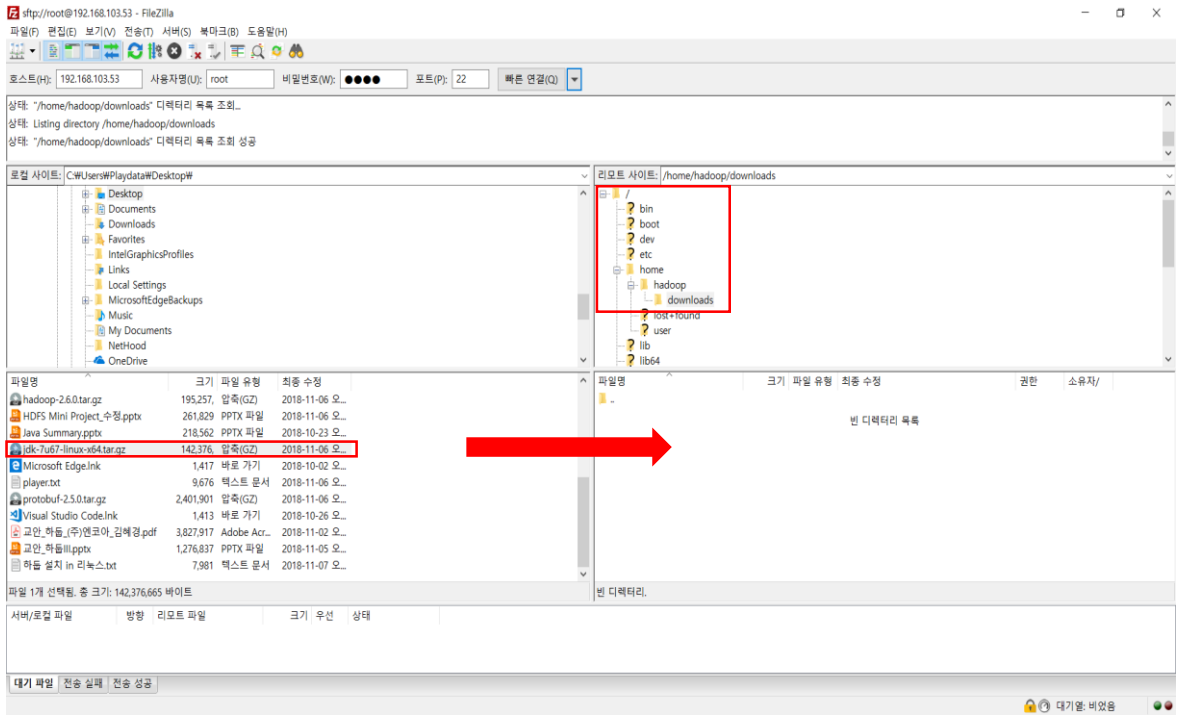
가상머신 실행



Filezilla 실행

호스트: 고정 ip (ifcfg-enp0s8의 고정 ip)
사용자명 : root
비밀번호 : 사용자명의 비밀번호
포트 : 22

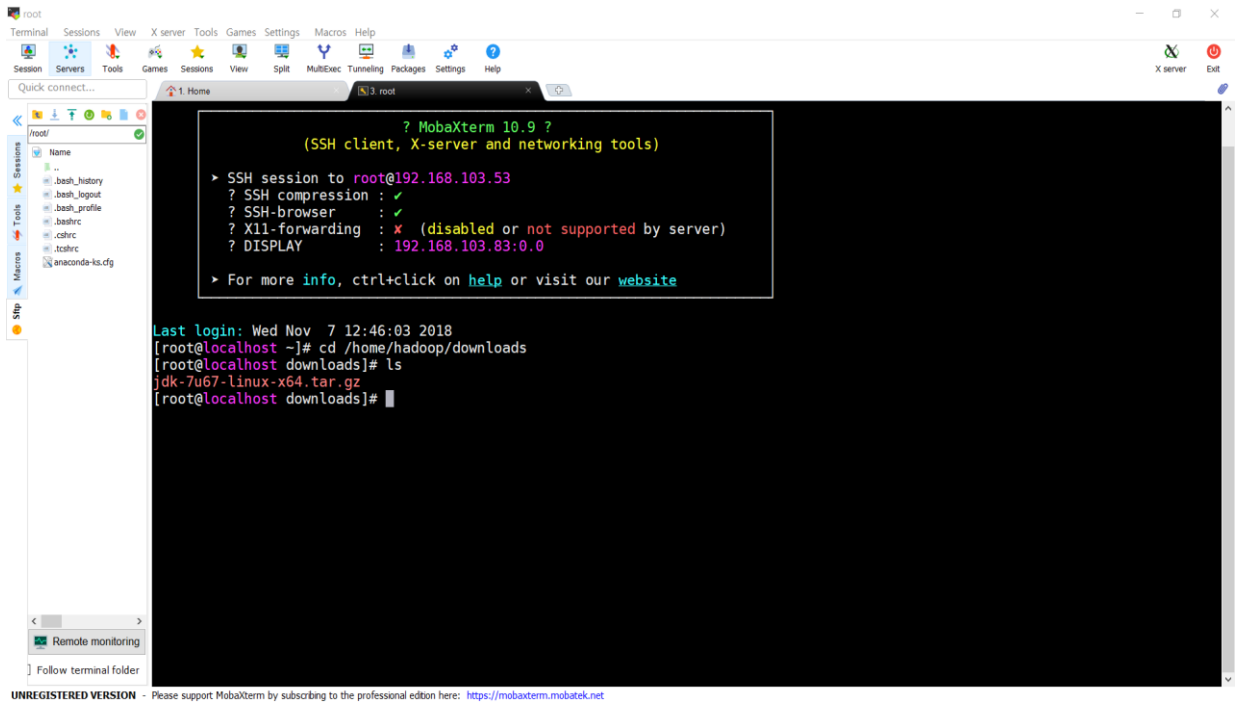
가상머신 실행



Filezilla 실행

오라클 사이트에서 다운로드 받은 jdk-7u67-linux-x64.tar.gz 파일을 가상머신의 /home/hadoop/downloads 디렉토리에 복사

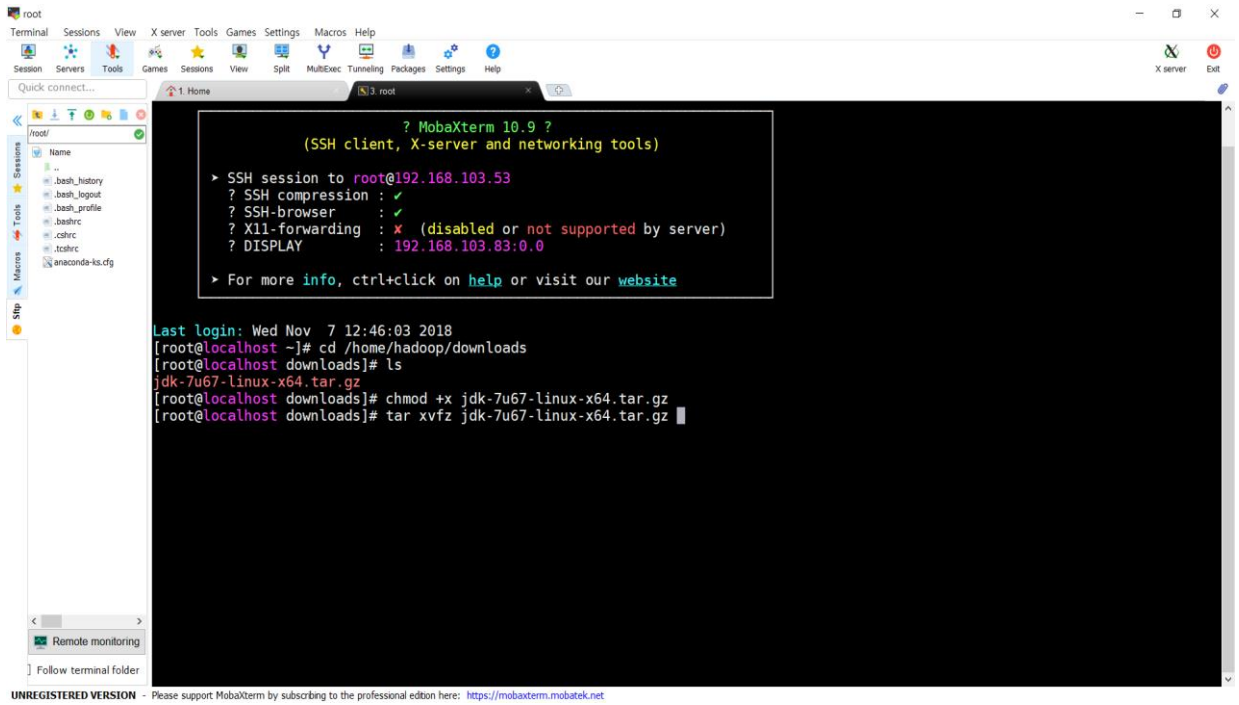
가상머신 실행



/home/hadoop/downloads 디렉토리에 다운로드 받은 jdk-7u76-linux-x64.tar.gz 파일 확인

[root@localhost downloads]# ls

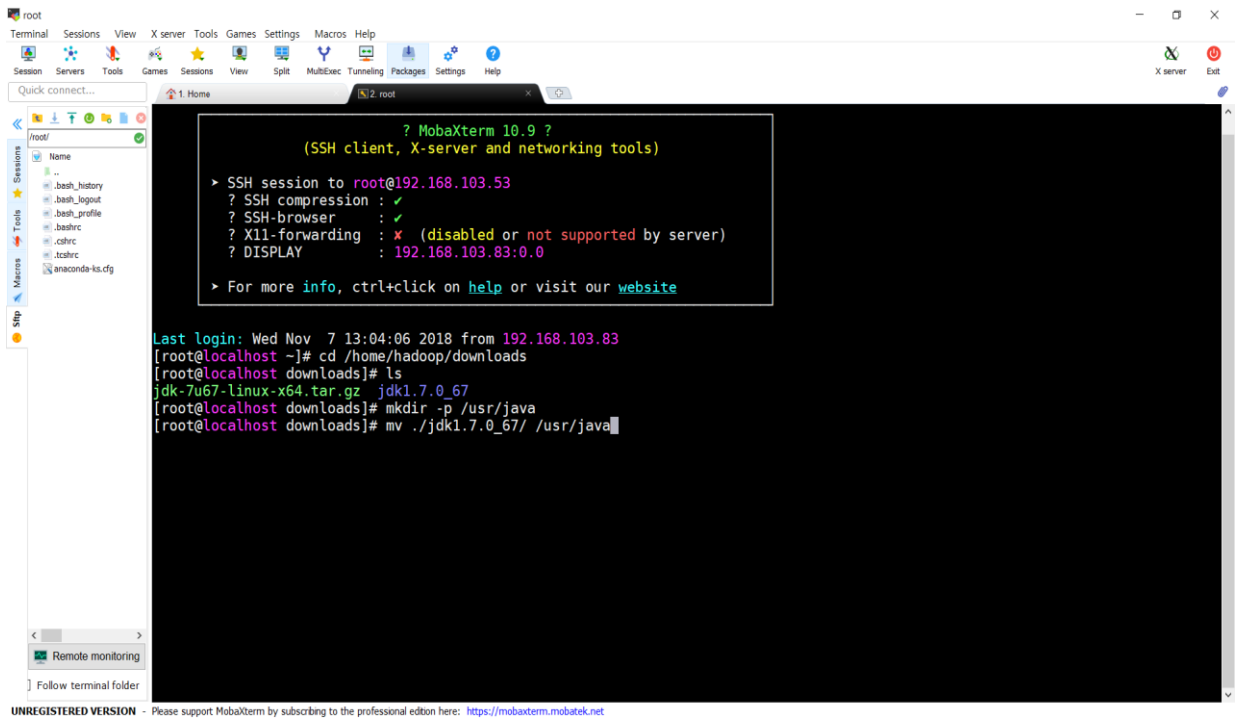
가상머신 실행



/home/hadoop/downloads 디렉토리에 다운로드 받은 jdk-7u76-linux-x64.tar.gz 파일 압축 해제

[root@localhost downloads]# chmod +x jdk-7u67-linux-x64.tar.gz
[root@localhost downloads]# tar xvfz jdk-7u67-linux-x64.tar.gz

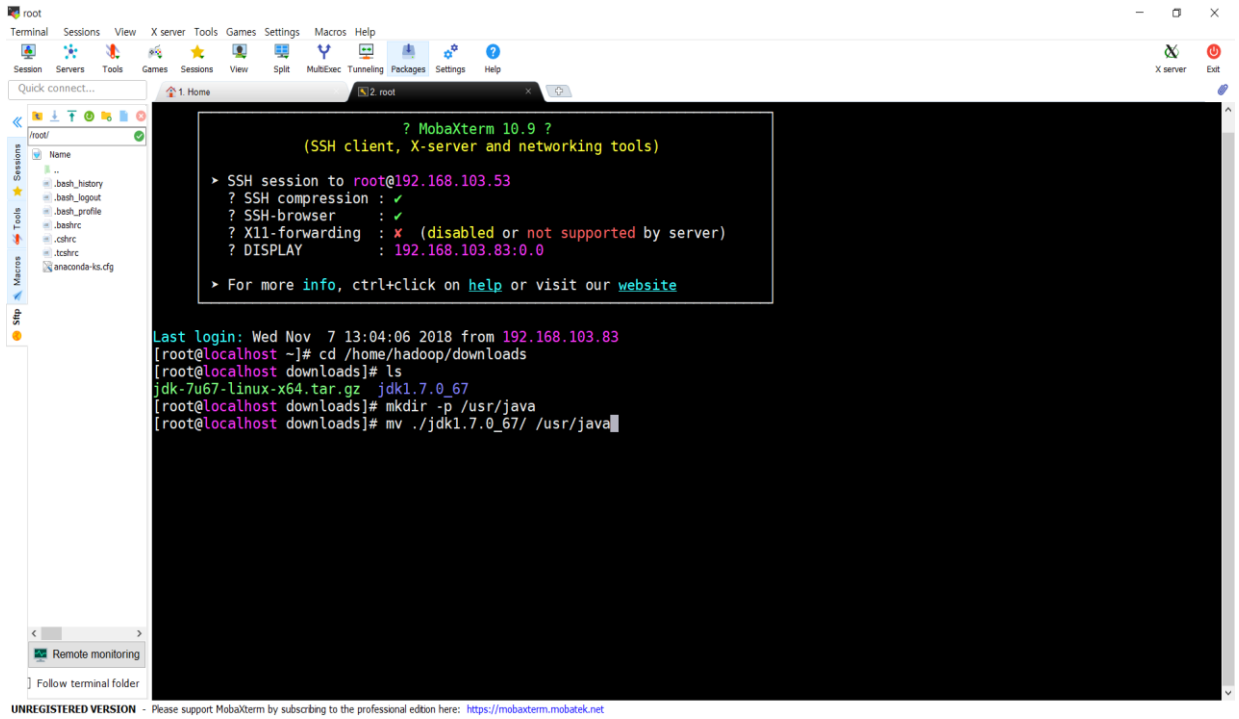
가상머신 실행



/usr 하위에 java 디렉토리 생성

[root@localhost downloads]# mkdir -p /usr/java

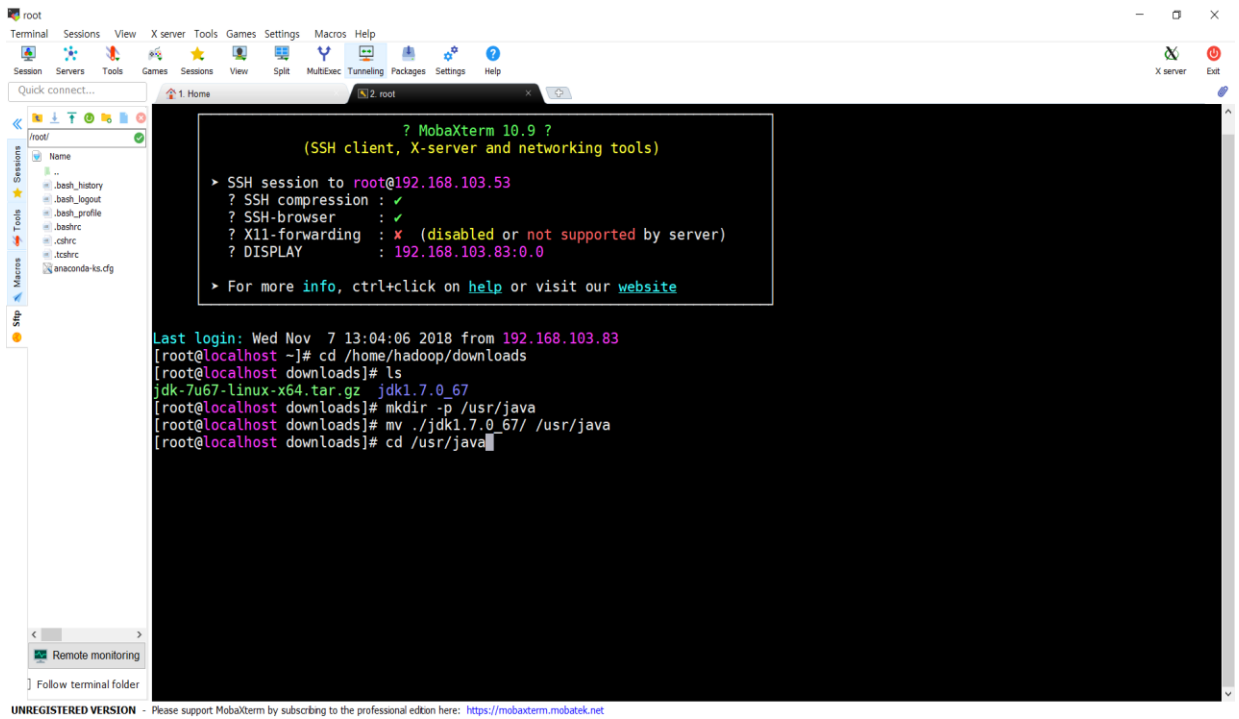
가상머신 실행



압축 해제한 jdk 파일을 /usr/java 디렉토리로 이동

[root@localhost downloads]# mv ./jdk1.7.0_67/ /usr/java

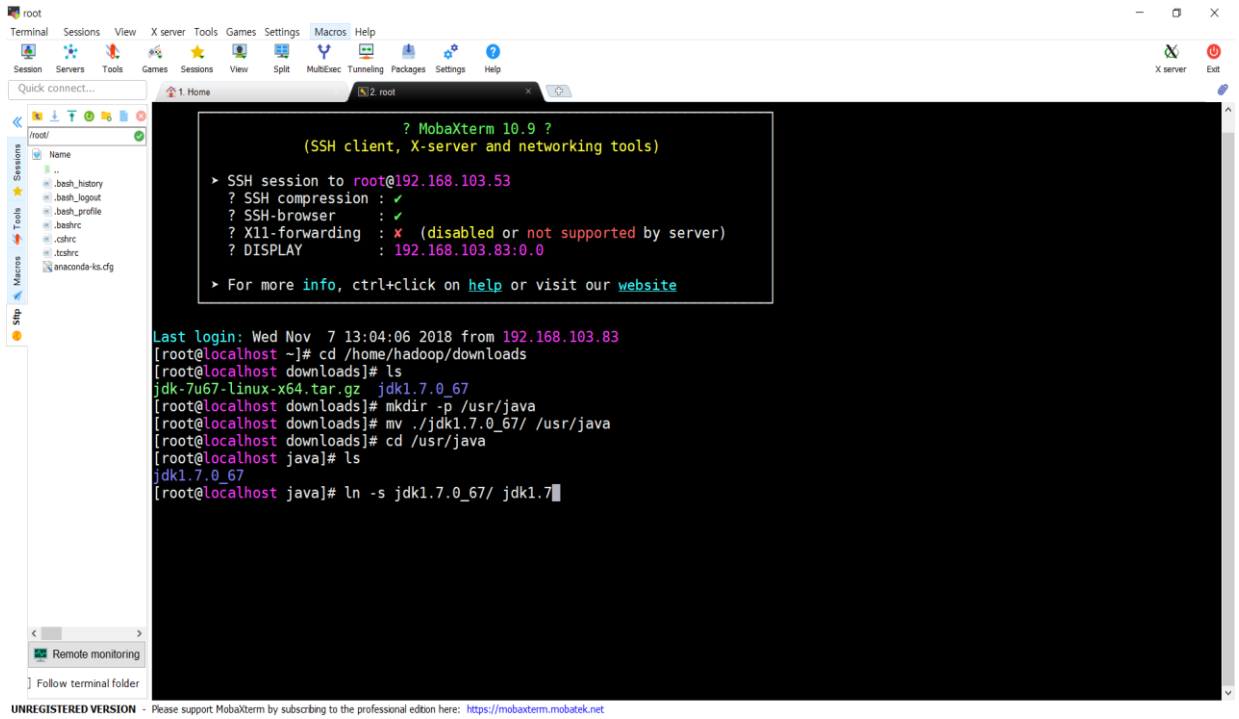
가상머신 실행



/usr/java 디렉토리로 이동

[root@localhost downloads]# cd /usr/java

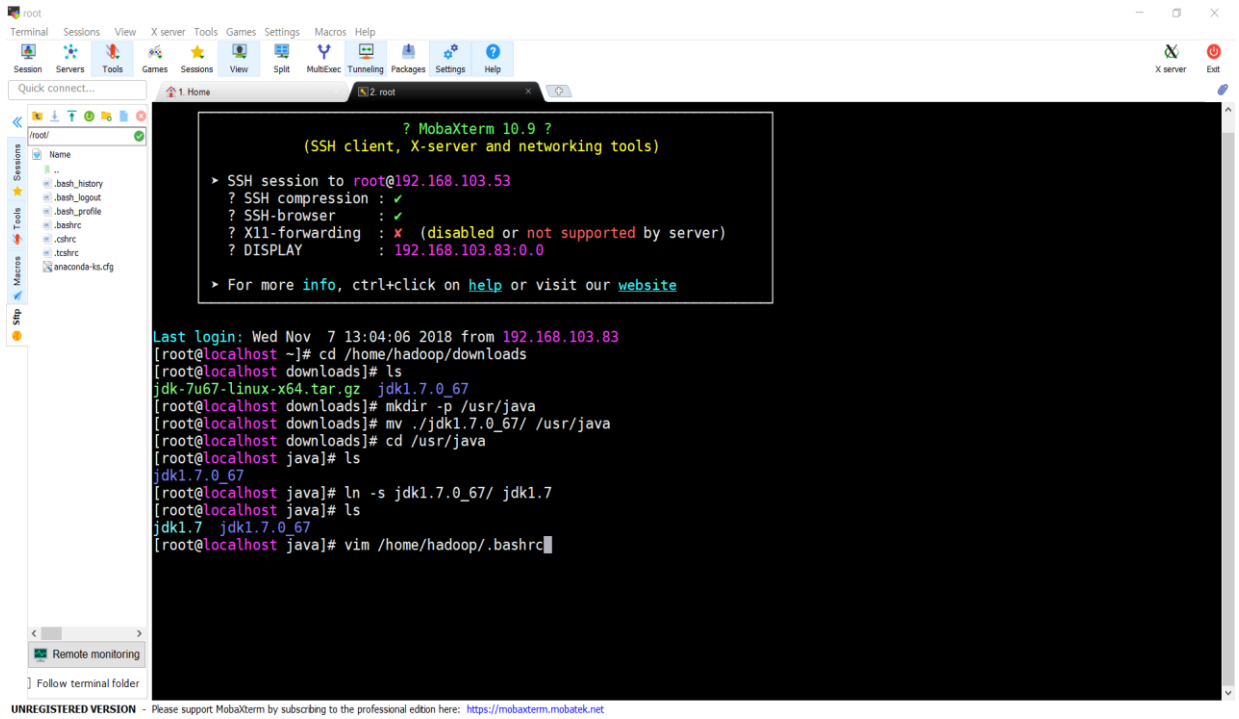
가상머신 실행



jdk1.7.0_67의 명칭 변경해서 새로 생성

[root@localhost java]# ln –s jdk1.7.0_67/ jdk1.7

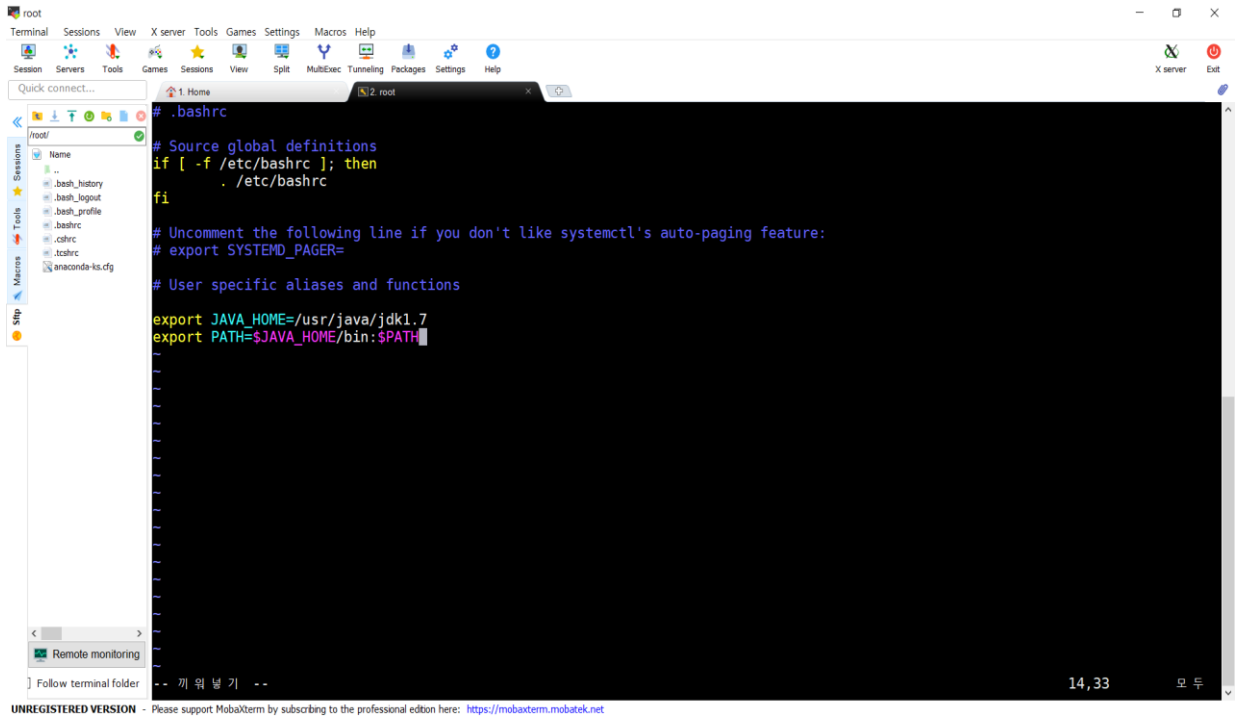
가상머신 실행



/home/hadoop/.bashrc를 vim으로 실행해 편집

[root@localhost java]# vim /home/hadoop/.bashrc

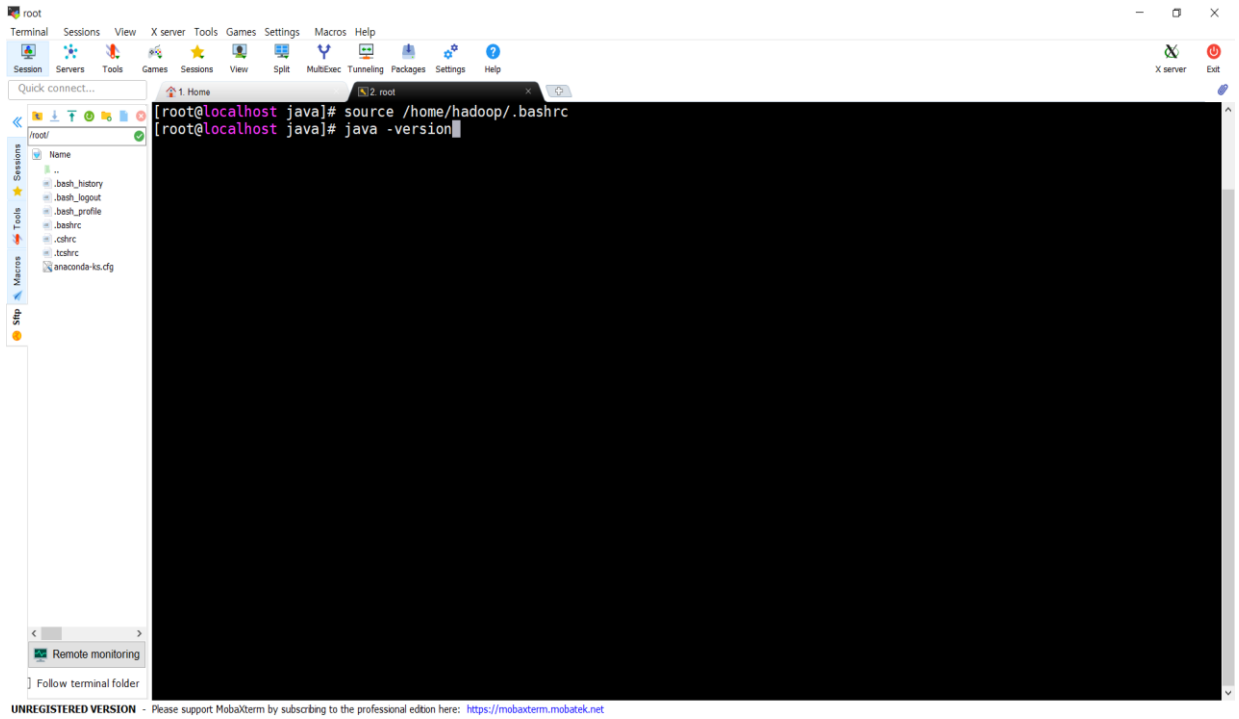
가상머신 실행



/home/hadoop/.bashrc를 vim으로 실행해 편집

export JAVA_HOME=/usr/java/jdk1.7
export PATH=$JAVA_HOME/bin:$PATH

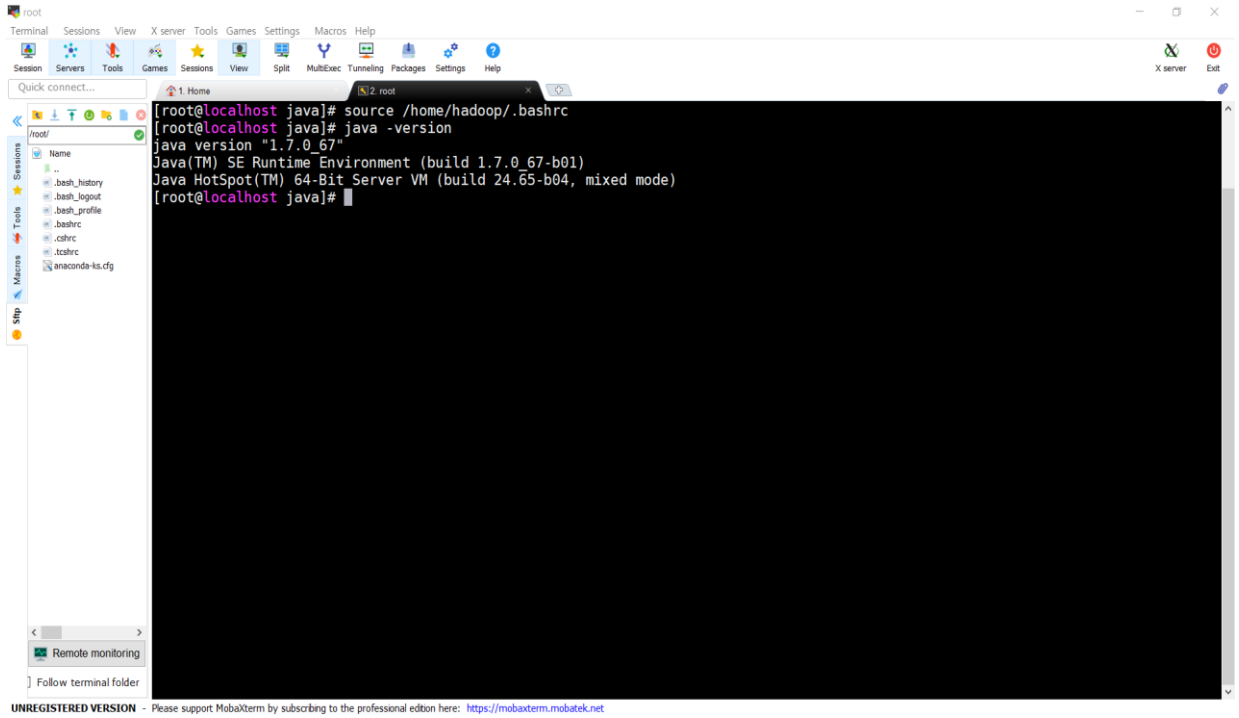
가상머신 실행



/home/hadoop/.bashrc 적용 및 java 버전 확인

[root@localhost java]# source /home/hadoop/.bashrc
[root@localhost java]# java -version

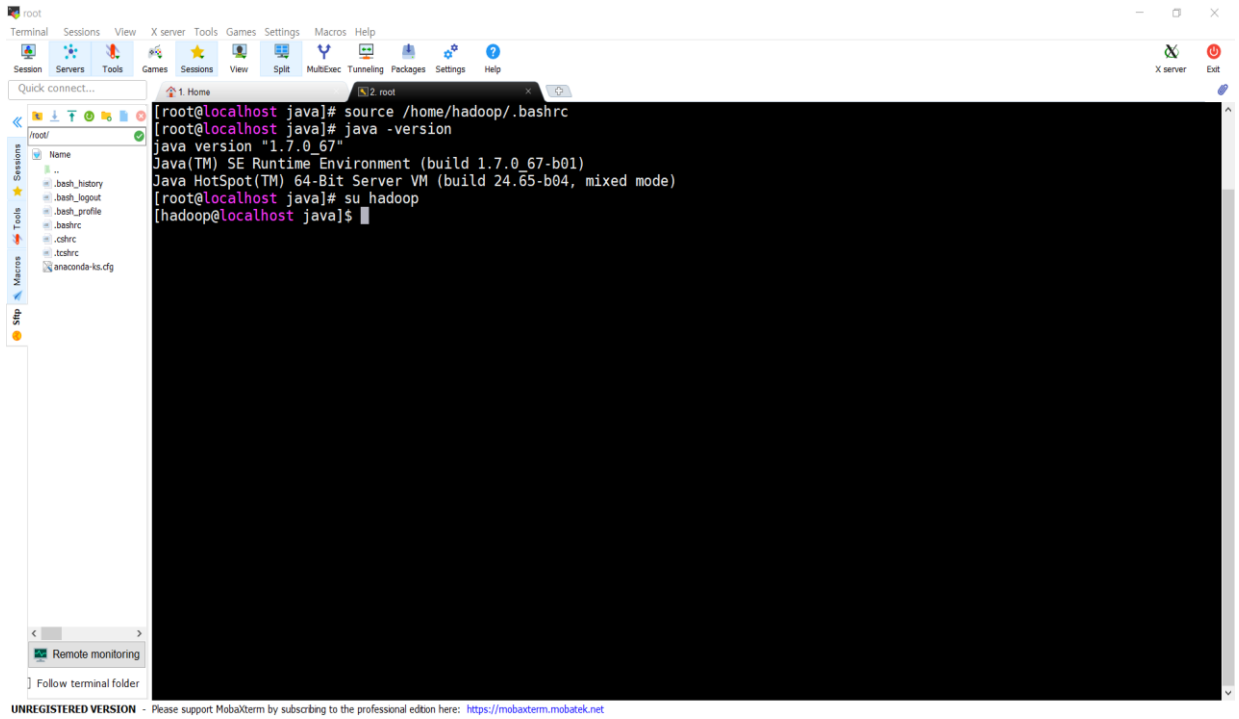
가상머신 실행



/home/hadoop/.bashrc 적용 및 java 버전 확인

[root@localhost java]# source /home/hadoop/.bashrc
[root@localhost java]# java -version

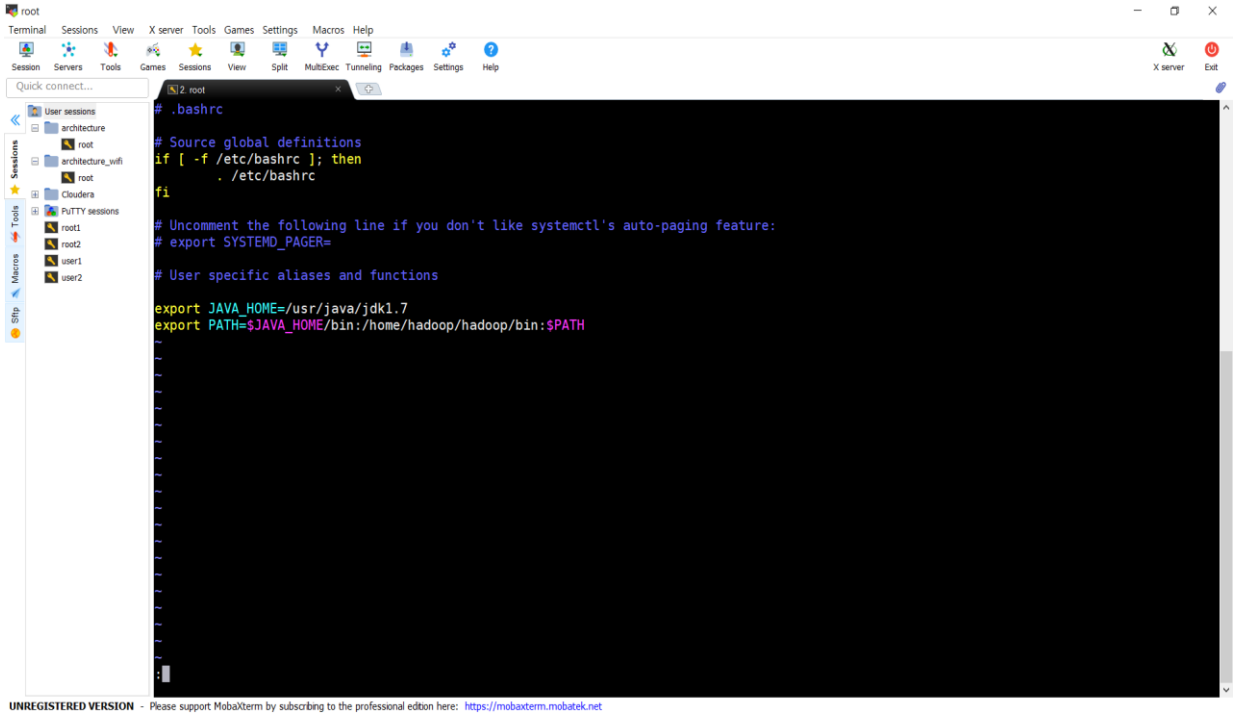
가상머신 실행



hadoop 계정으로 사용자 변경

[root@localhost java]# su hadoop

가상머신 실행

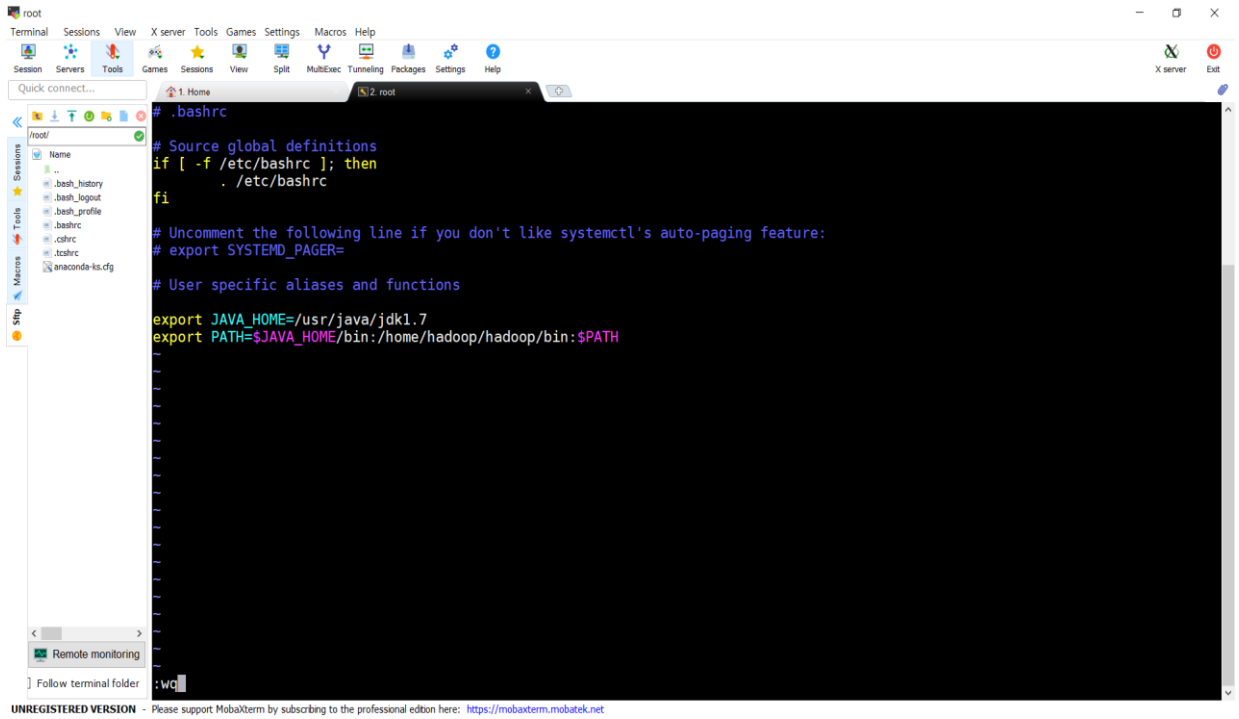


/home/hadoop/.bashrc 파일 vim으로 편집

[hadoop@localhost java]$ vim /home/hadoop/.bashrc

export PATH=$JAVA_HOME/bin:$PATH를
export PATH=$JAVA_HOME/bin:/home/hadoop/hadoop/bin:$PATH

가상머신 실행

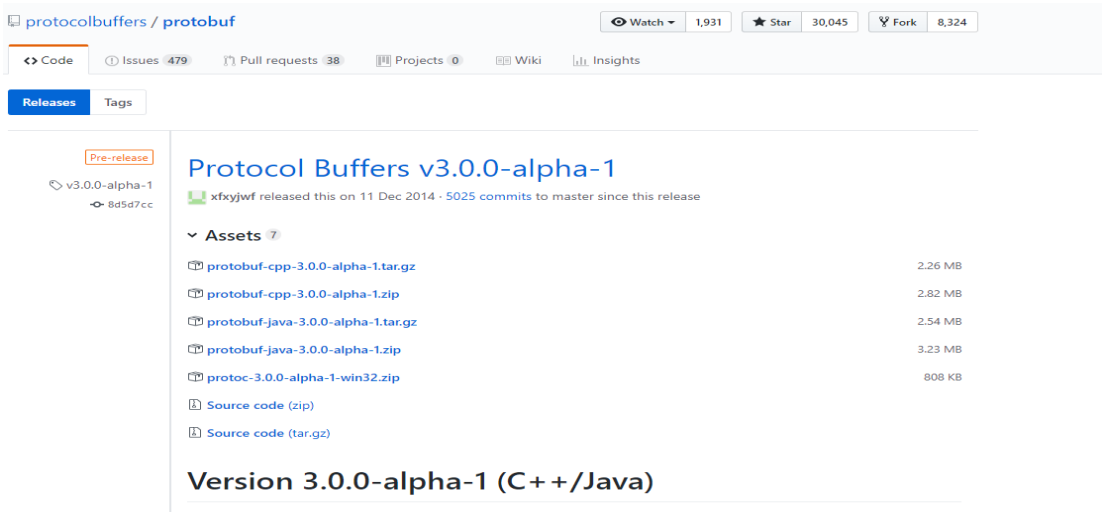


/home/hadoop/.bashrc 파일 vim으로 편집

[hadoop@localhost java]$ vim /home/hadoop/.bashrc

export PATH=$JAVA_HOME/bin:$PATH를
export PATH=$JAVA_HOME/bin:/home/hadoop/hadoop/bin:$PATH

가상머신 실행

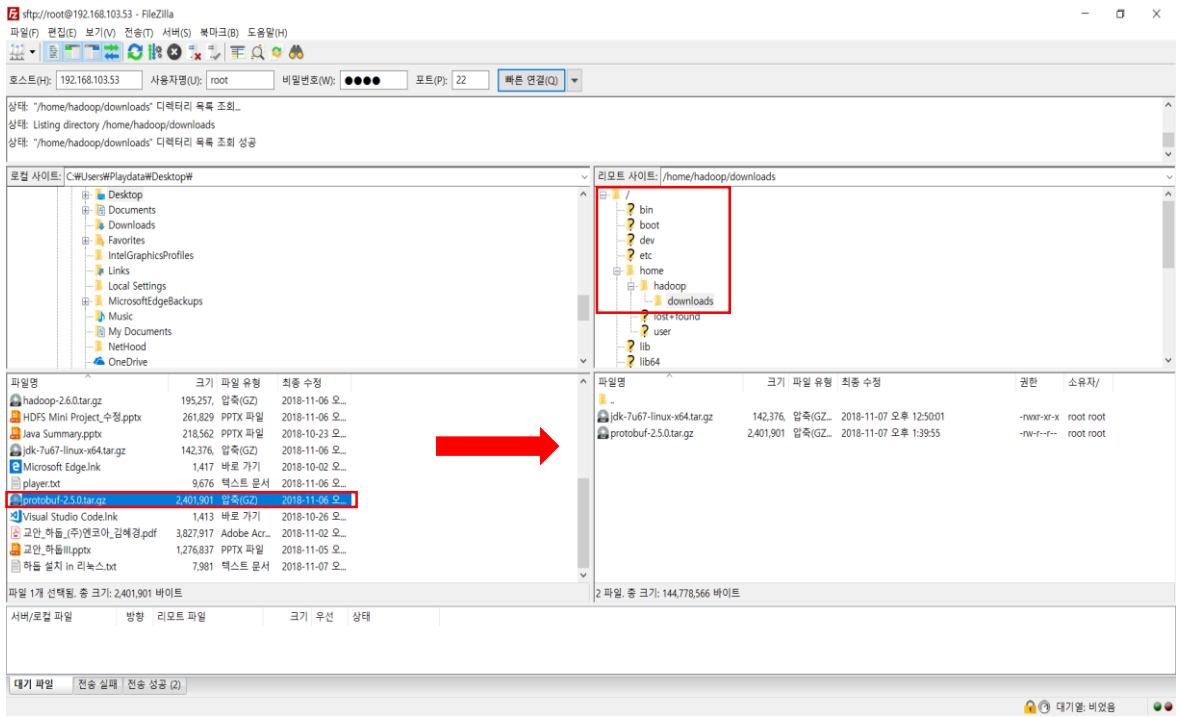


Protocol Buffer 2.5.0 버전 다운로드 받은 후
"Filezilla"를 통해 /home/hadoop/downloads에 업로드

Url
https://github.com/protocolbuffers/protobuf/releases?after=v3.0.0-alpha-2

File
Protocol Buffers v2.5.0
protobuf-2.5.0.tar.gz, 2.29MB

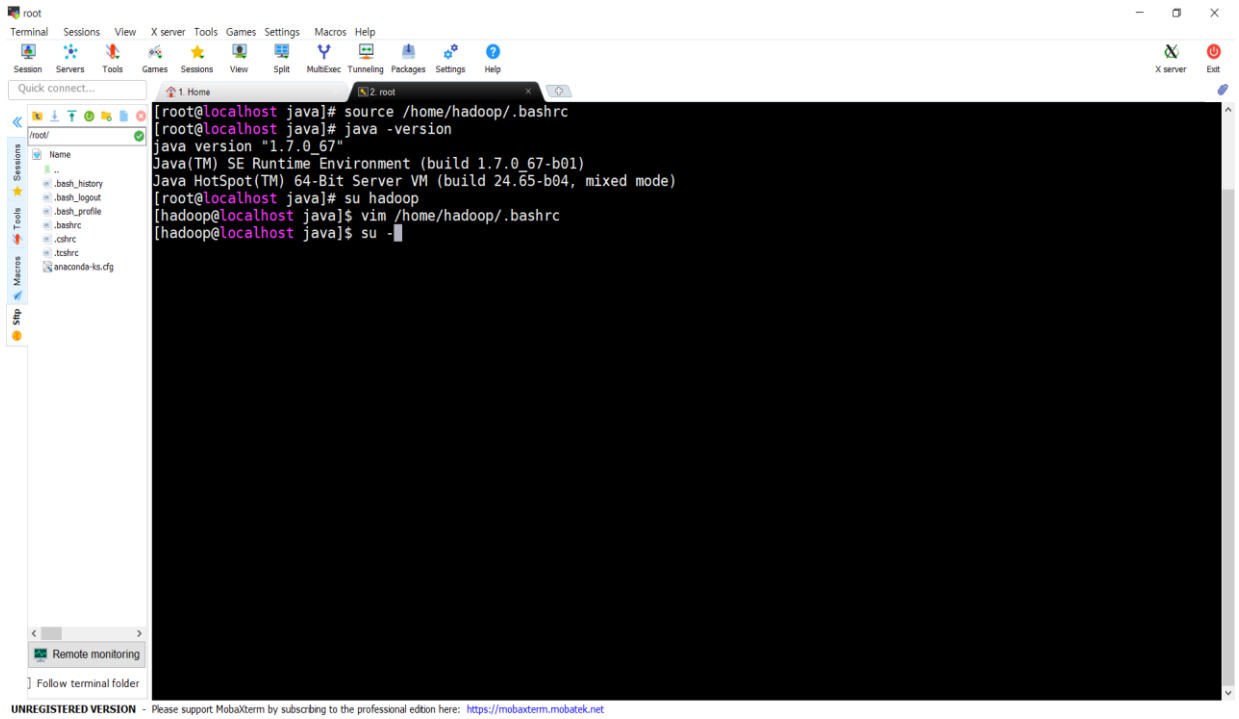
가상머신 실행



Filezilla 실행

Protocol buffers github 사이트에서 다운로드 받은 protobuf-2.5.0.tar.gz 파일을 가상머신의 /home/hadoop/downloads 디렉토리에 복사

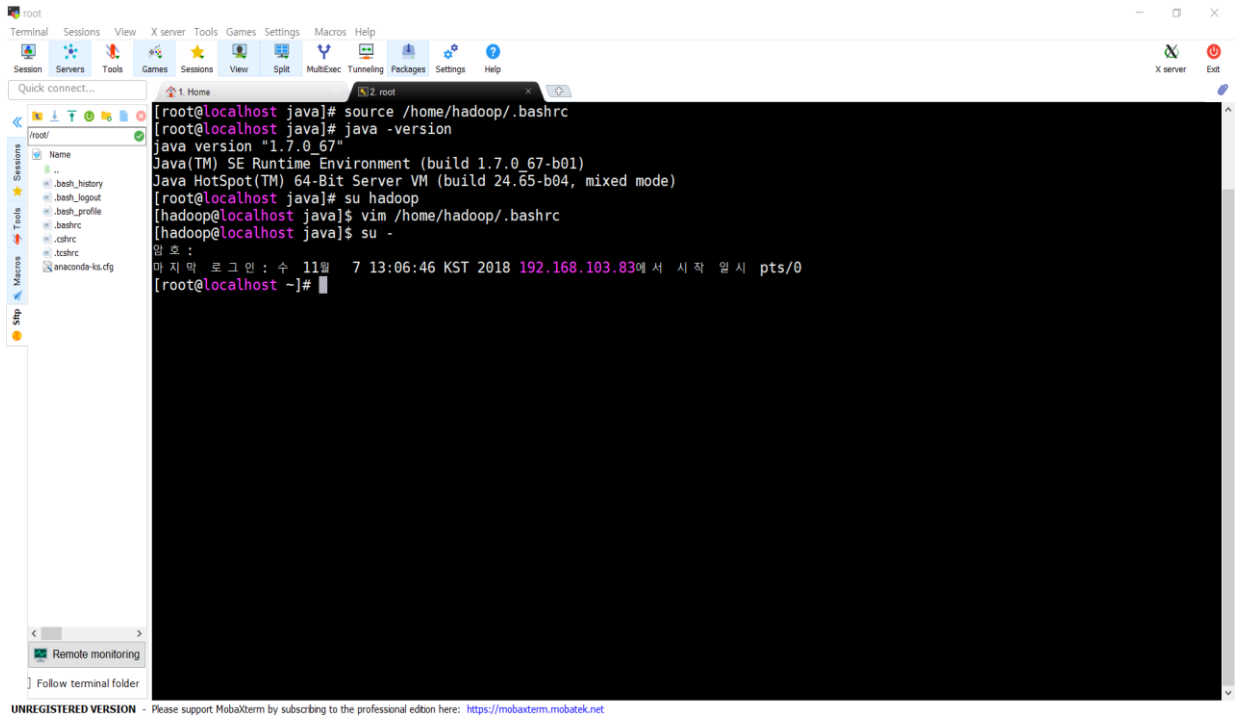
가상머신 실행



root 계정으로 사용자 변경

[hadoop@localhost java]$ su -

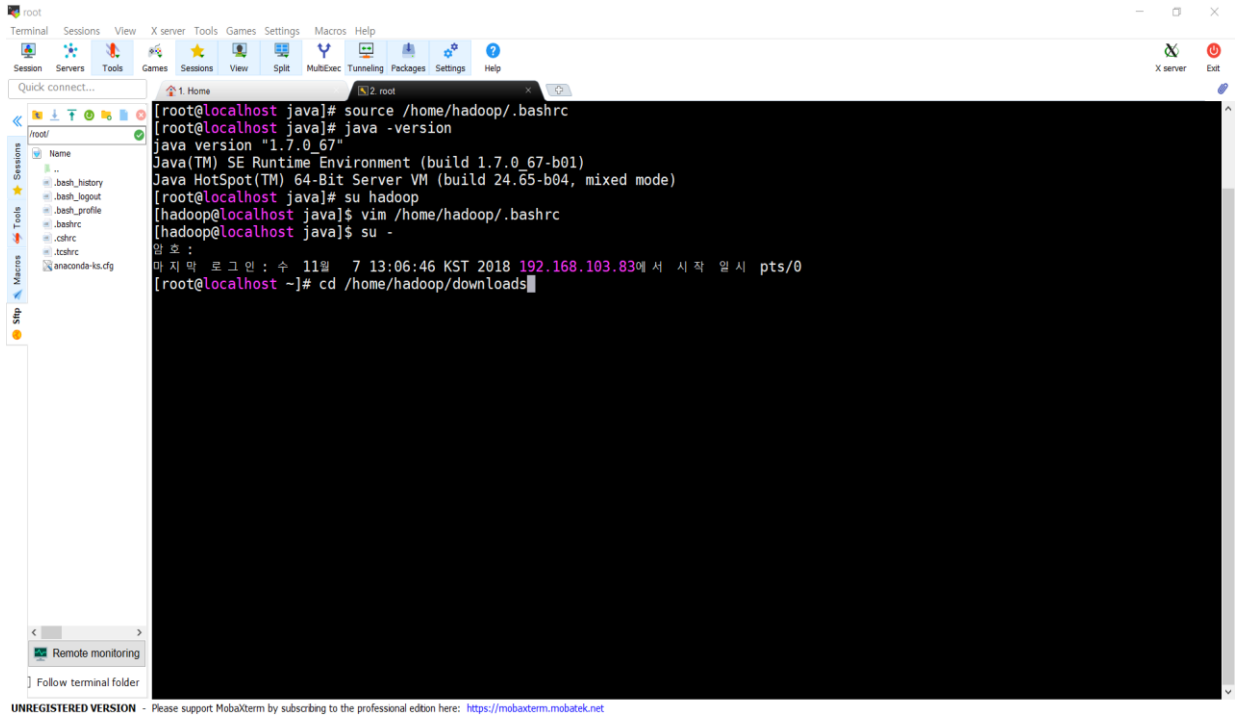
가상머신 실행



root 계정으로 사용자 변경

[hadoop@localhost java]$ su -

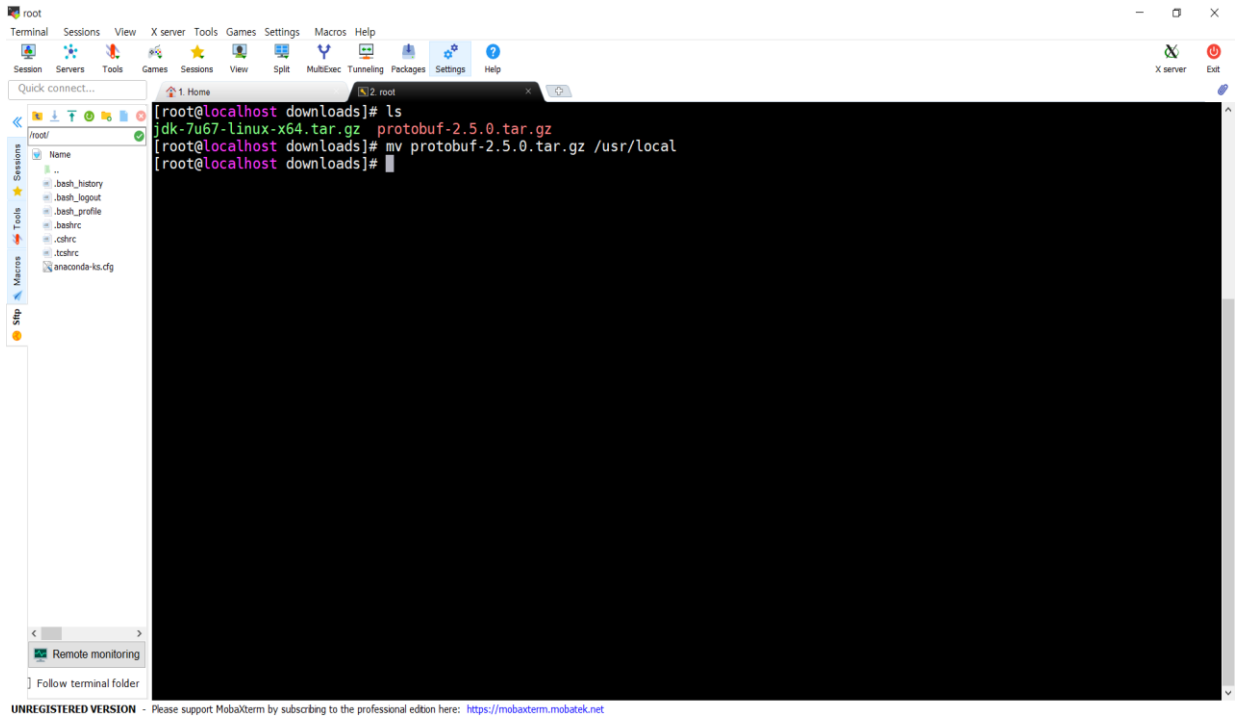
가상머신 실행



/home/hadoop/downloads 디렉토리로 이동

[root@localhost ~]# cd /home/hadoop/downloads

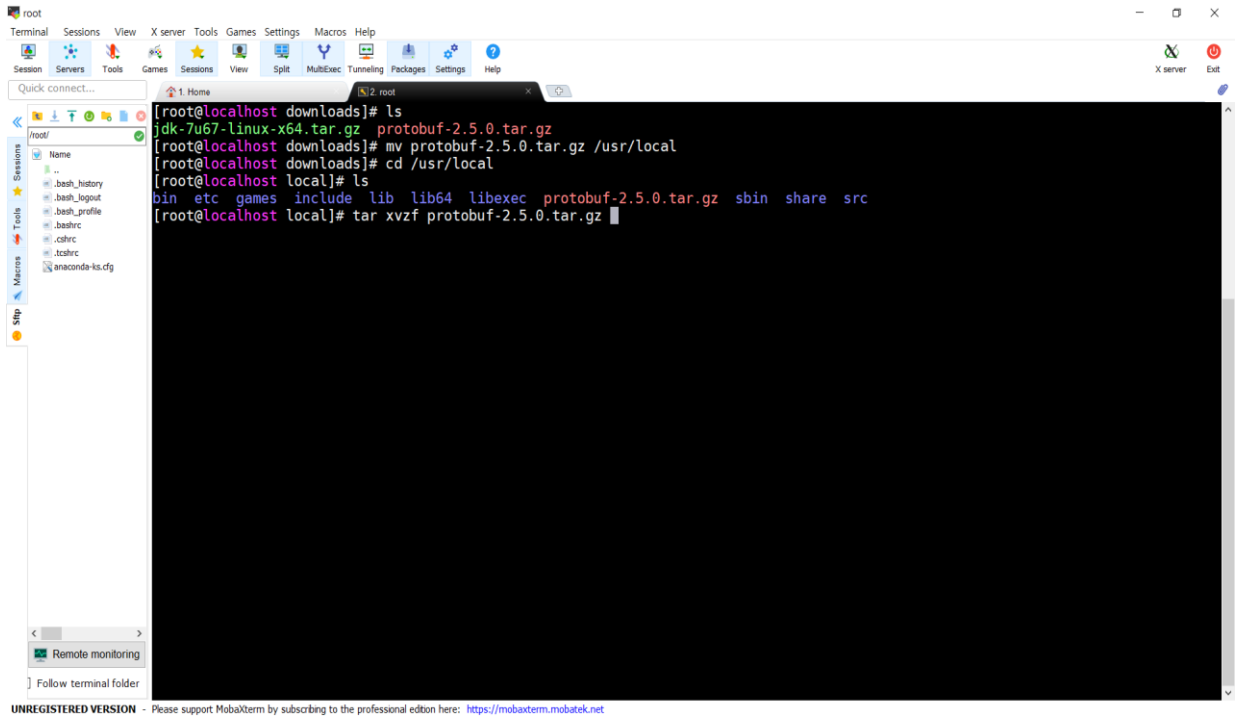
가상머신 실행



downloads 디렉토리에 위치한 protobuf-2.5.0.tar.gz 파일을 /usr/local로 이동

[root@localhost ~]# mv protobuf-2.5.0.tar.gz /usr/local

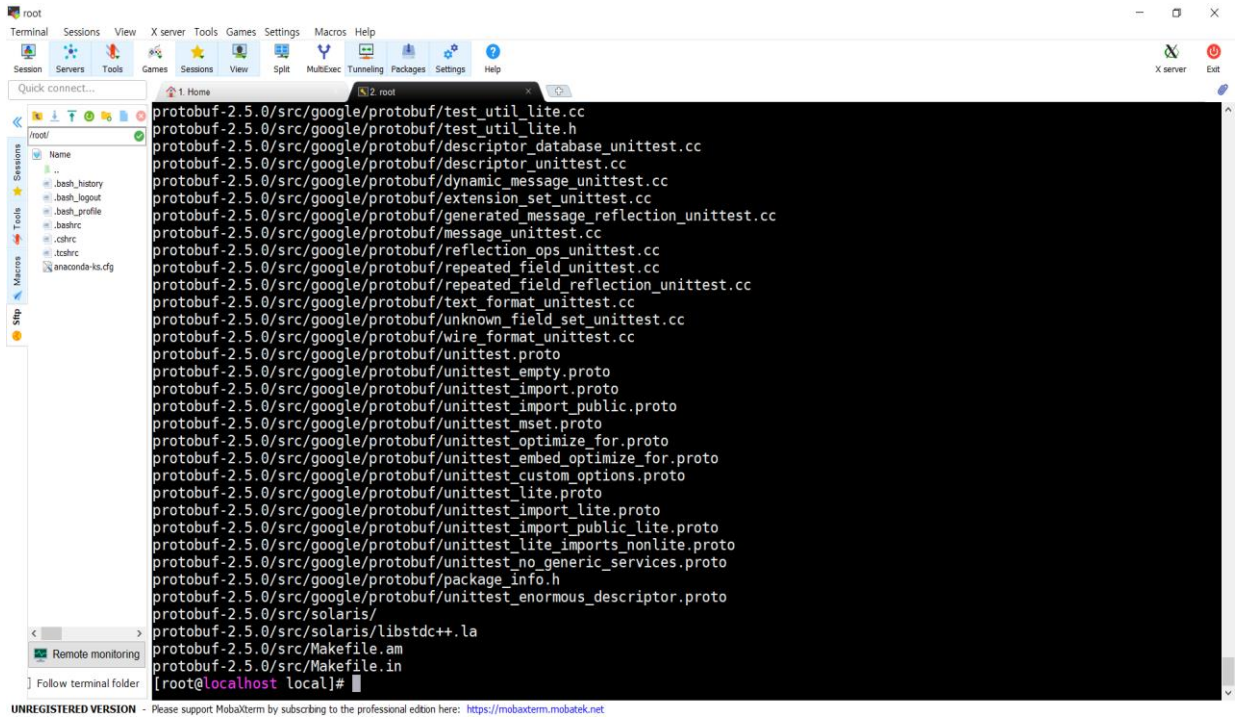
가상머신 실행



/usr/local 디렉토리에서 protobuf-2.5.0.tar.gz 파일 압축 해제

[root@localhost ~]# cd /usr/local
[root@localhost local]# tar xvzf protobuf-2.5.0.tar.gz

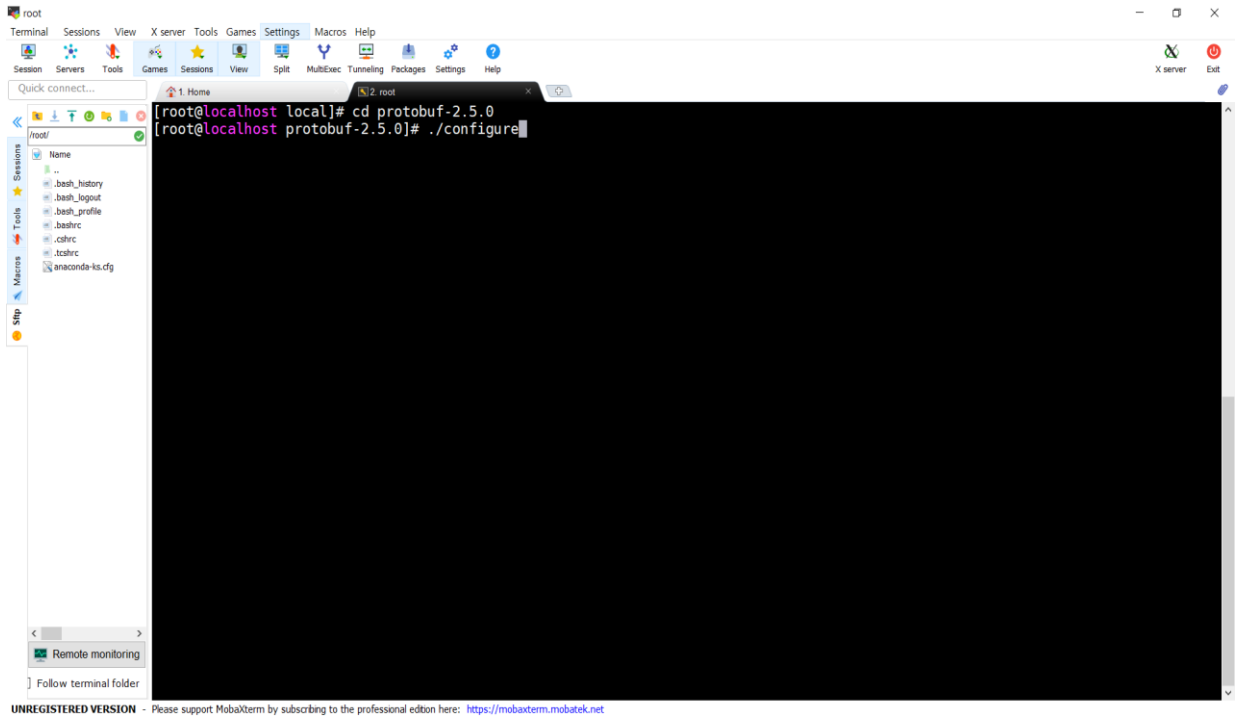
가상머신 실행



/usr/local 디렉토리에서 protobuf-2.5.0.tar.gz 파일 압축 해제

[root@localhost ~]# cd /usr/local
[root@localhost local]# tar xvzf protobuf-2.5.0.tar.gz

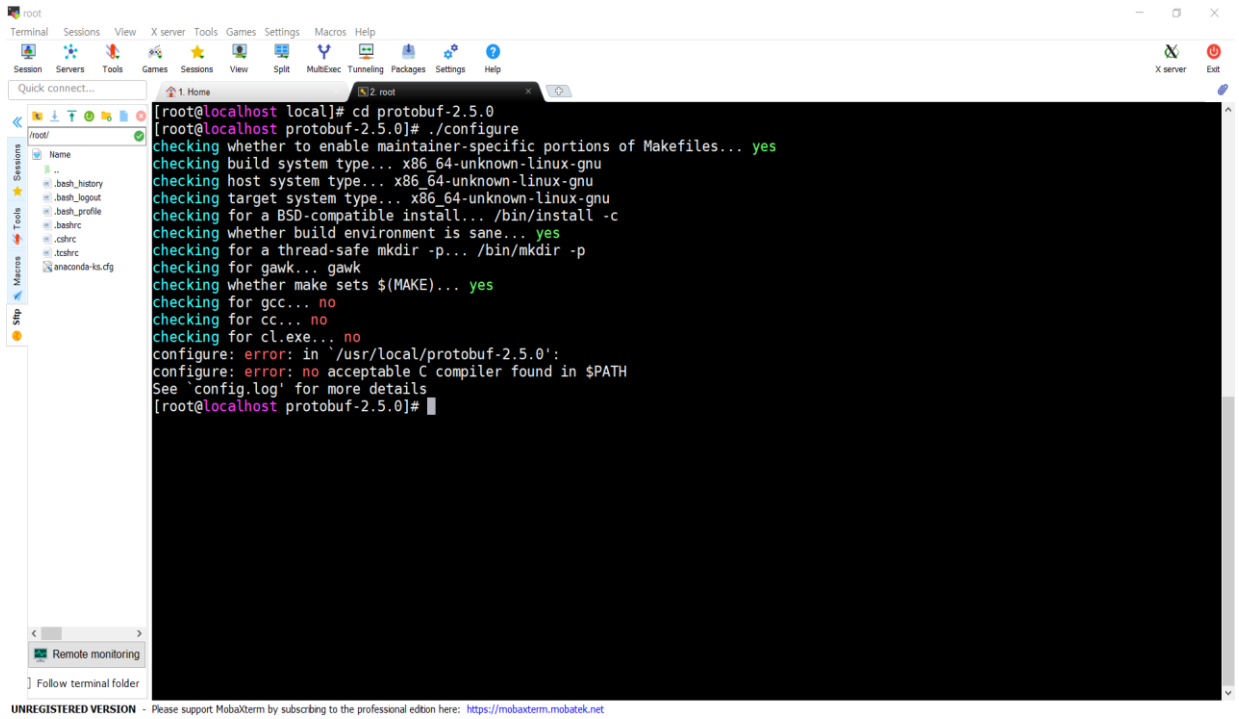
가상머신 실행



Protobuf-2.5.0 디렉토리로 이동 후 configure 파일 실행

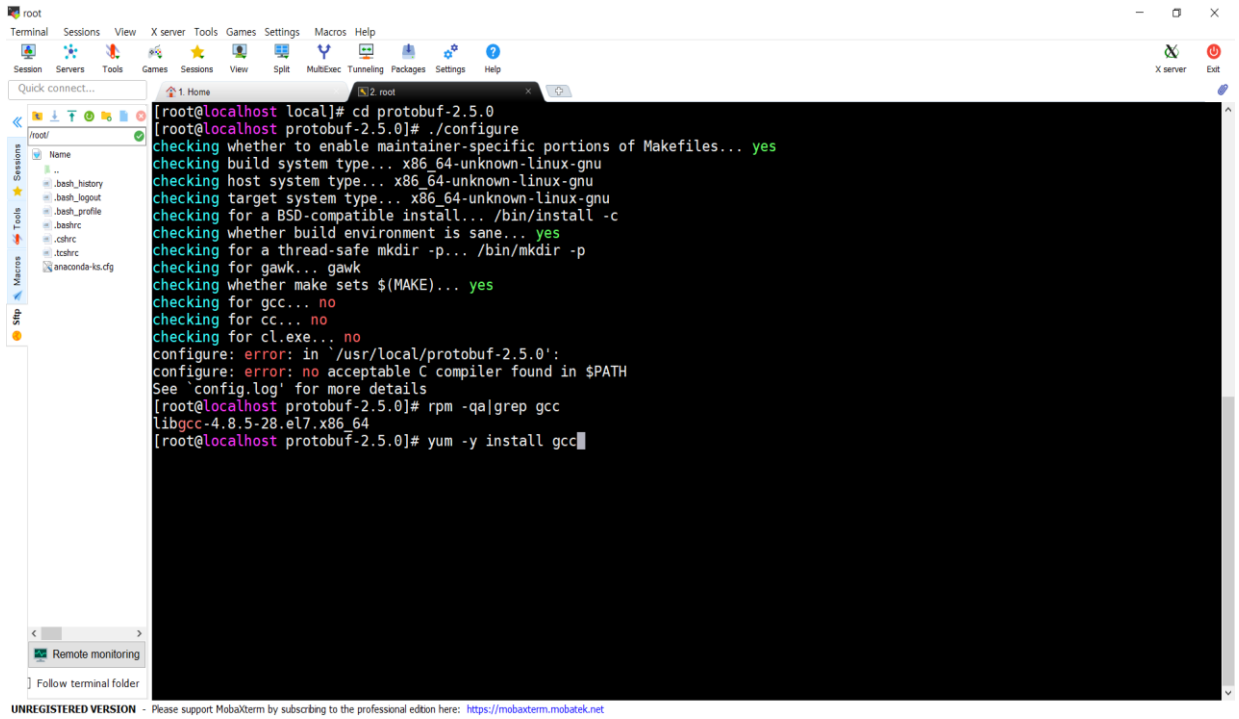
[root@localhost local]# cd protobuf-2.5.0
[root@localhost protobuf-2.5.0]# ./configure

가상머신 실행



Protobuf-2.5.0 디렉토리로 이동 후 configure 파일 실행
오류 발생!!!!!!

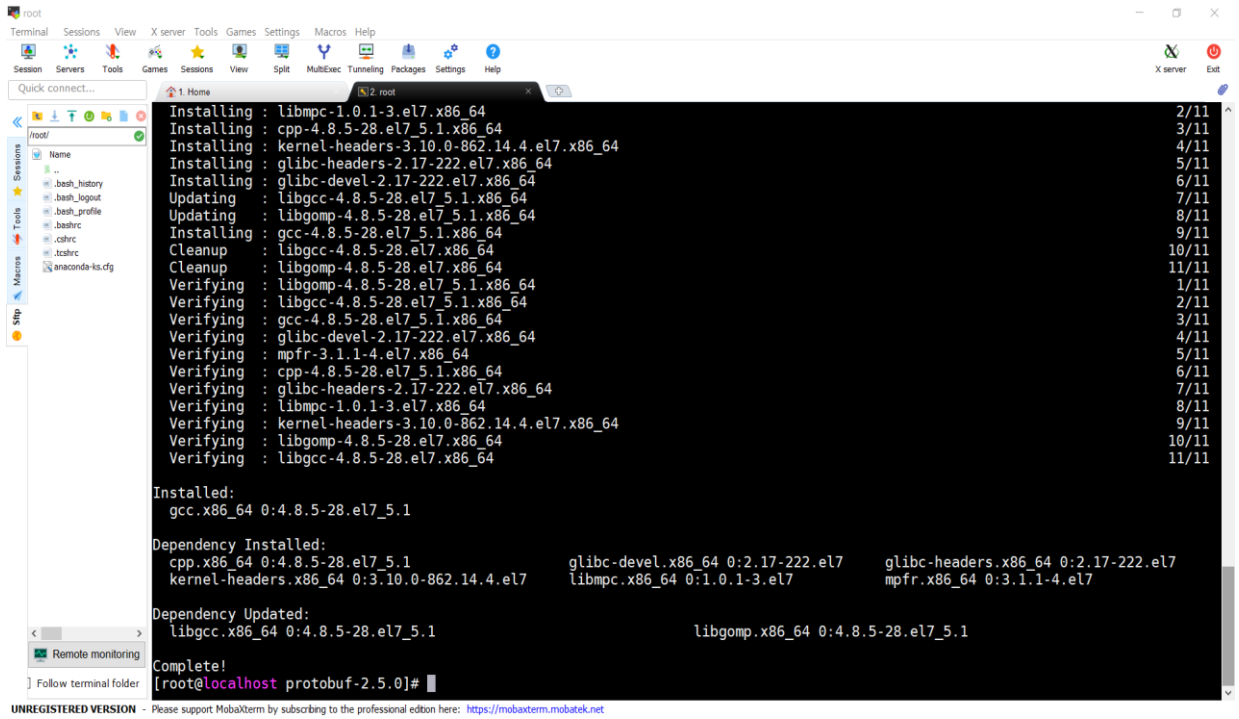
가상머신 실행



Protobuf-2.5.0 디렉토리로 이동 후 configure 파일 실행 오류 발생!!!!!! 해결하기 1

[root@localhost protobuf-2.5.0]# rpm –qa | grep gcc (컴파일러 설치 유무 확인)
[root@localhost protobuf-2.5.0]# yum –y install gcc (gcc 설치)

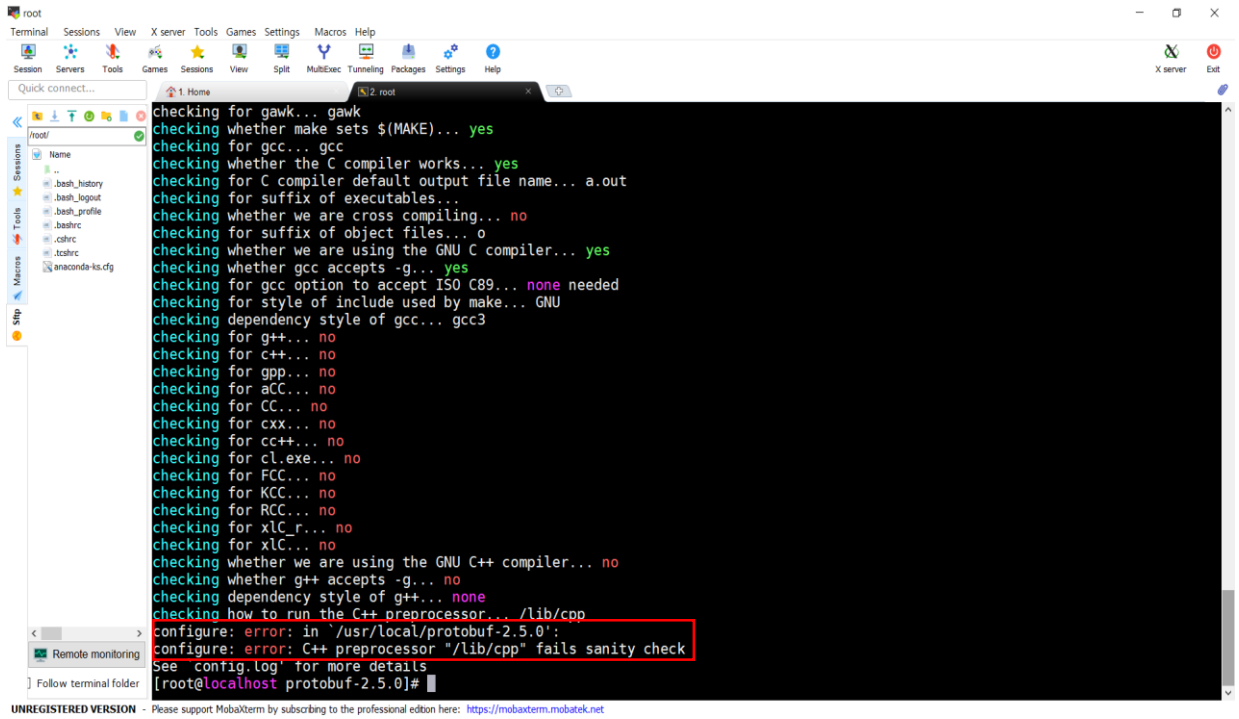
가상머신 실행



Protobuf-2.5.0 디렉토리로 이동 후 configure 파일 실행 오류 발생!!!!!! 해결하기 1

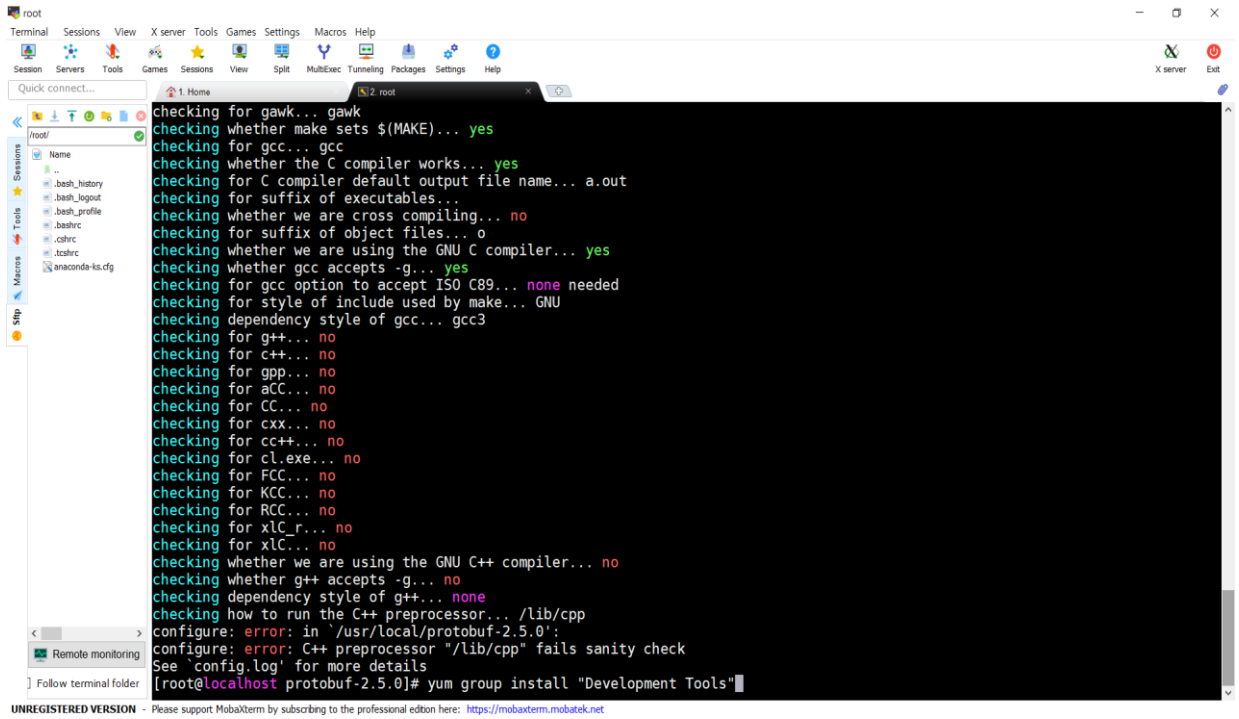
[root@localhost protobuf-2.5.0]# rpm –qa|grep gcc (컴파일러 설치 유무 확인)
[root@localhost protobuf-2.5.0]# yum –y install gcc (gcc 설치)

가상머신 실행



gcc 설치한 후에도 configure 파일 실행 오류가 발생하면.....

가상머신 실행

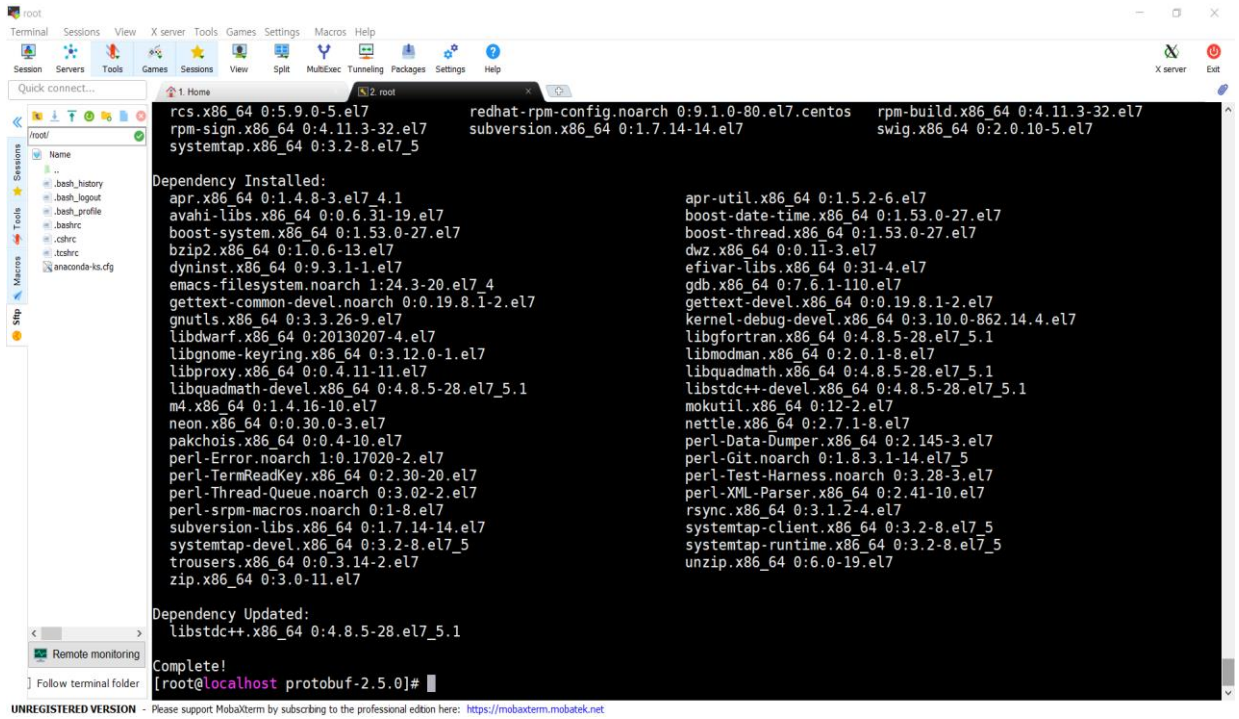


gcc 설치한 후에도 configure 파일 실행 오류가 발생하면.....

해결하기 2

[root@localhost protobuf-2.5.0]# yum group install "Development Tools"

가상머신 실행

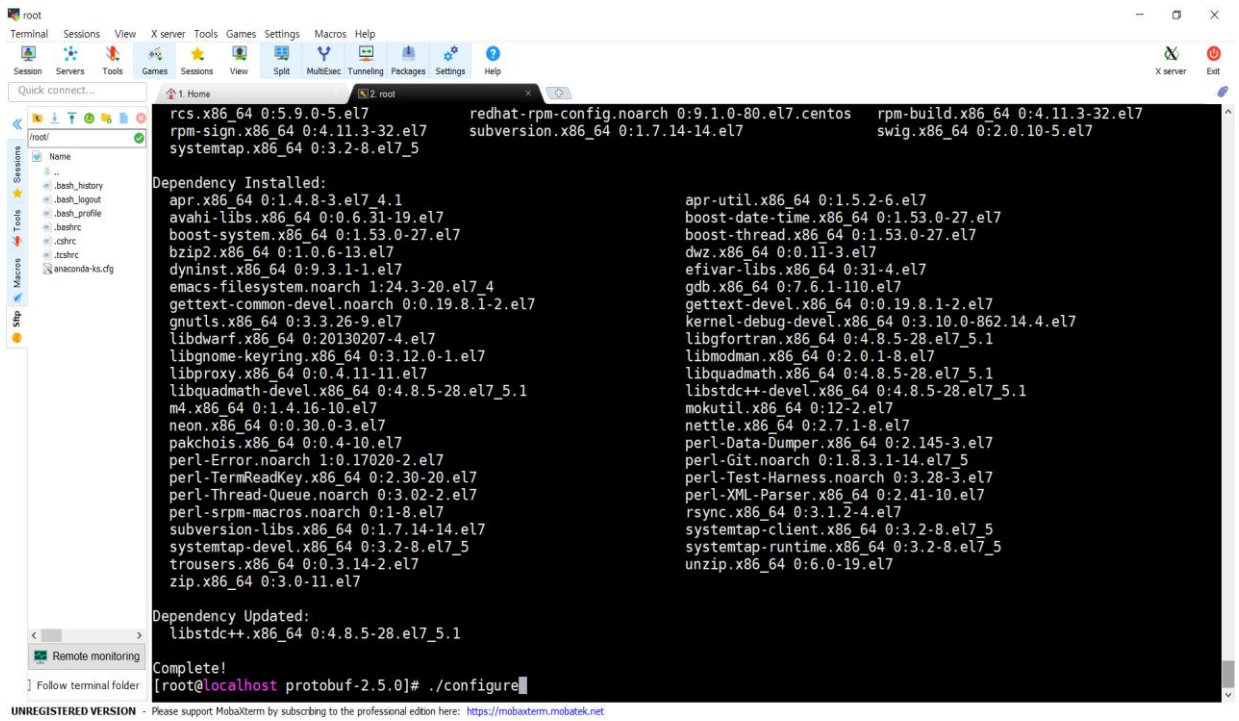


gcc 설치한 후에도 configure 파일 실행 오류가 발생하면.....

해결하기 2

[root@localhost protobuf-2.5.0]# yum group install "Development Tools"

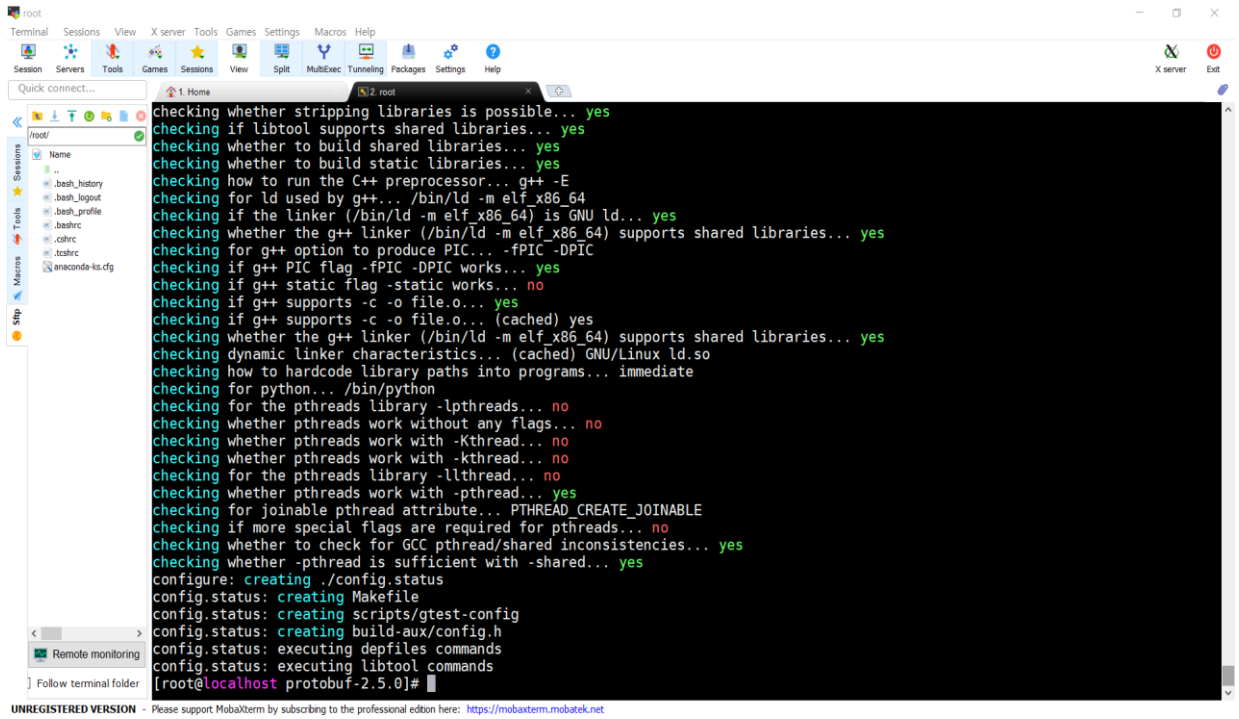
가상머신 실행



Development Tools 설치한 후 configure 파일 실행

[root@localhost protobuf-2.5.0]# ./configure

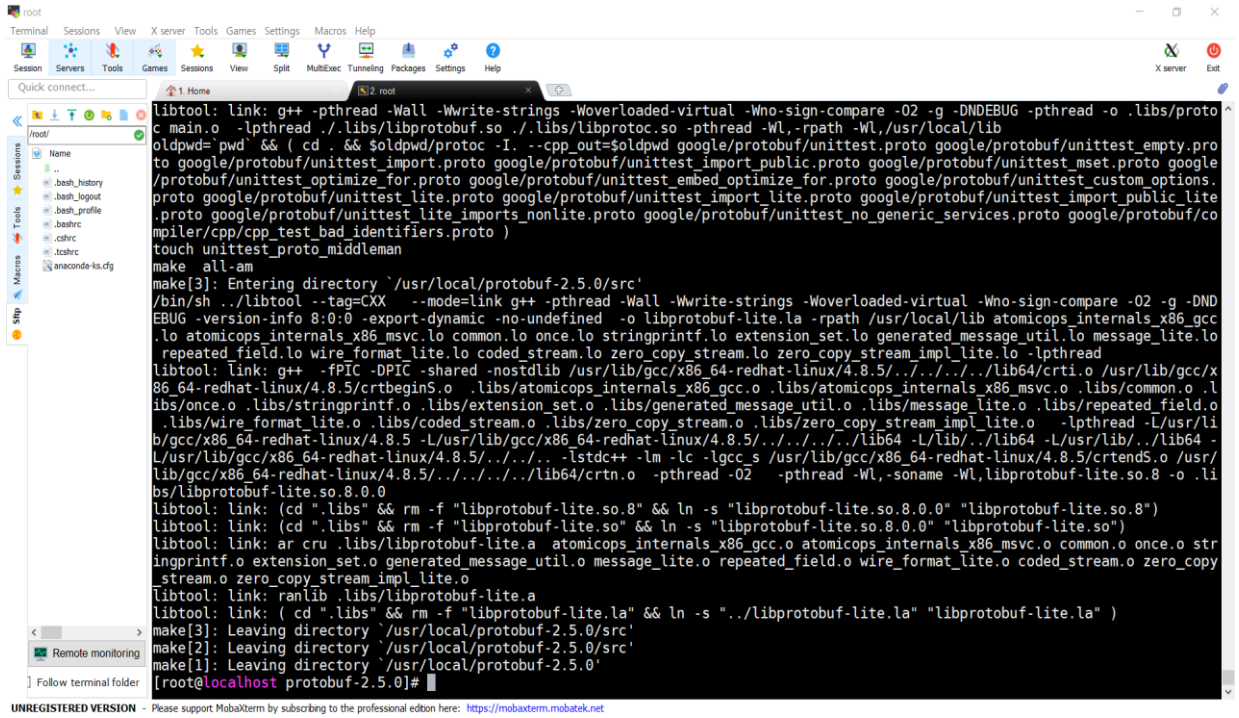
가상머신 실행



Development Tools 설치한 후 configure 파일 실행

[root@localhost protobuf-2.5.0]# ./configure

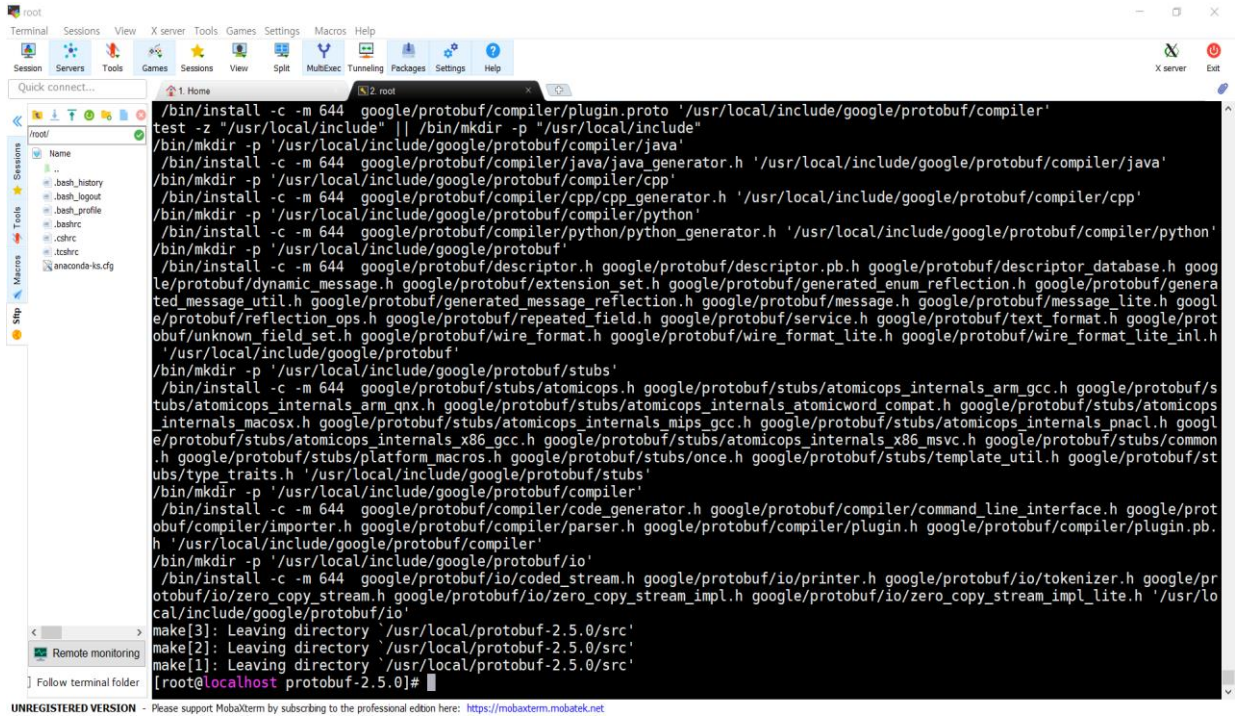
가상머신 실행



Development Tools 설치한 후 configure 파일 실행이 정상적으로 된다면 make 명령어 실행

[root@localhost protobuf-2.5.0]# make

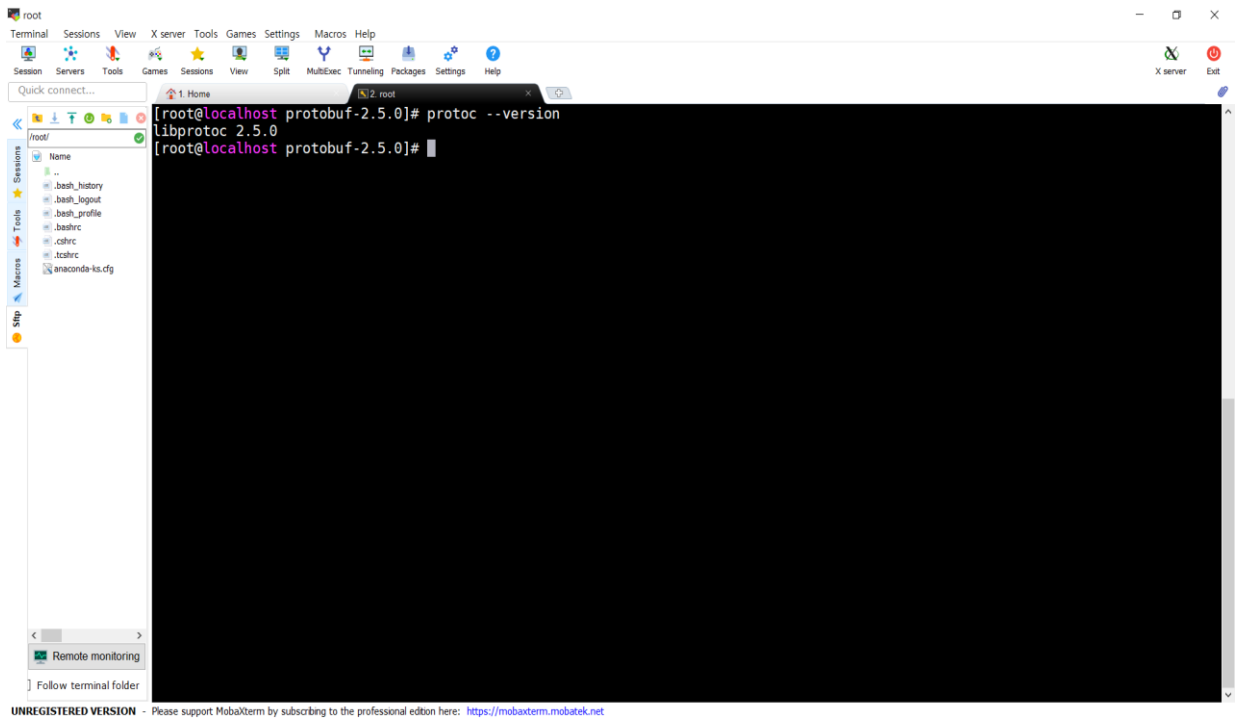
가상머신 실행



Development Tools 설치한 후 configure 파일 실행이 정상적으로 된다면 make install명령어 실행

[root@localhost protobuf-2.5.0]# make install

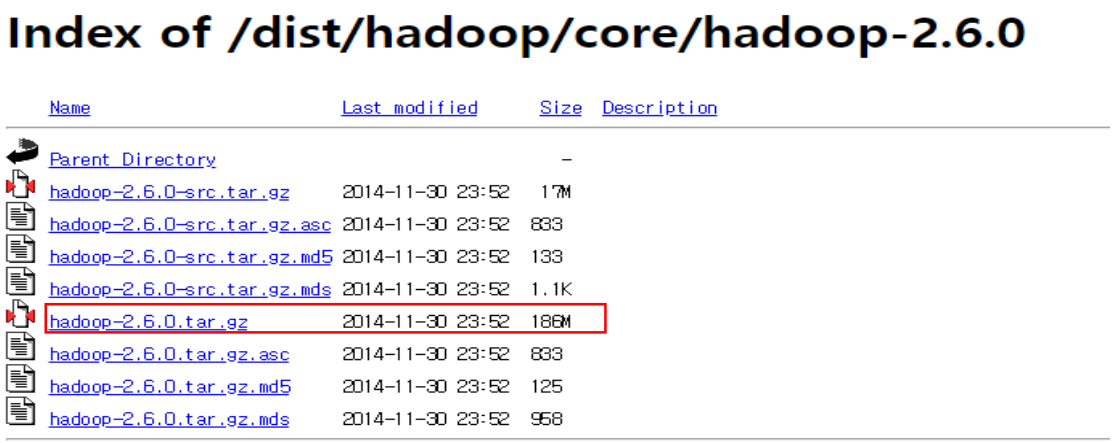
가상머신 실행



Protocol buffer 버전 확인

[root@localhost protobuf-2.5.0]# protoc --version

가상머신 실행



Index of /dist/hadoop/core/hadoop-2.6.0

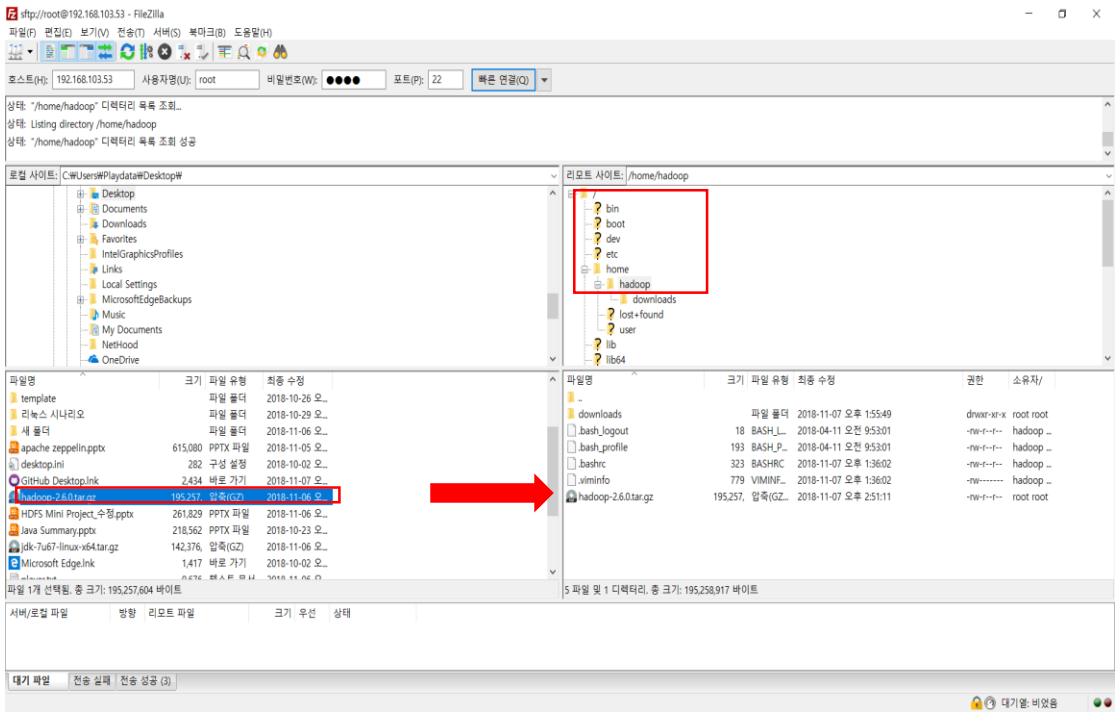
Name	Last modified	Size	Description
Parent Directory		-	
hadoop-2.6.0-src.tar.gz	2014-11-30 23:52	17M	
hadoop-2.6.0-src.tar.gz.asc	2014-11-30 23:52	833	
hadoop-2.6.0-src.tar.gz.md5	2014-11-30 23:52	133	
hadoop-2.6.0-src.tar.gz.mds	2014-11-30 23:52	1.1K	
hadoop-2.6.0.tar.gz	2014-11-30 23:52	186M	
hadoop-2.6.0.tar.gz.asc	2014-11-30 23:52	833	
hadoop-2.6.0.tar.gz.md5	2014-11-30 23:52	125	
hadoop-2.6.0.tar.gz.mds	2014-11-30 23:52	958	

하둡 2.6.0 버전 다운로드 받은 후
"Filezilla"를 통해 /home/hadoop에 업로드

Url
https://archive.apache.org/dist/hadoop/core/hadoop-2.6.0/

File
hadoop-2.6.0.tar.gz, 186MB

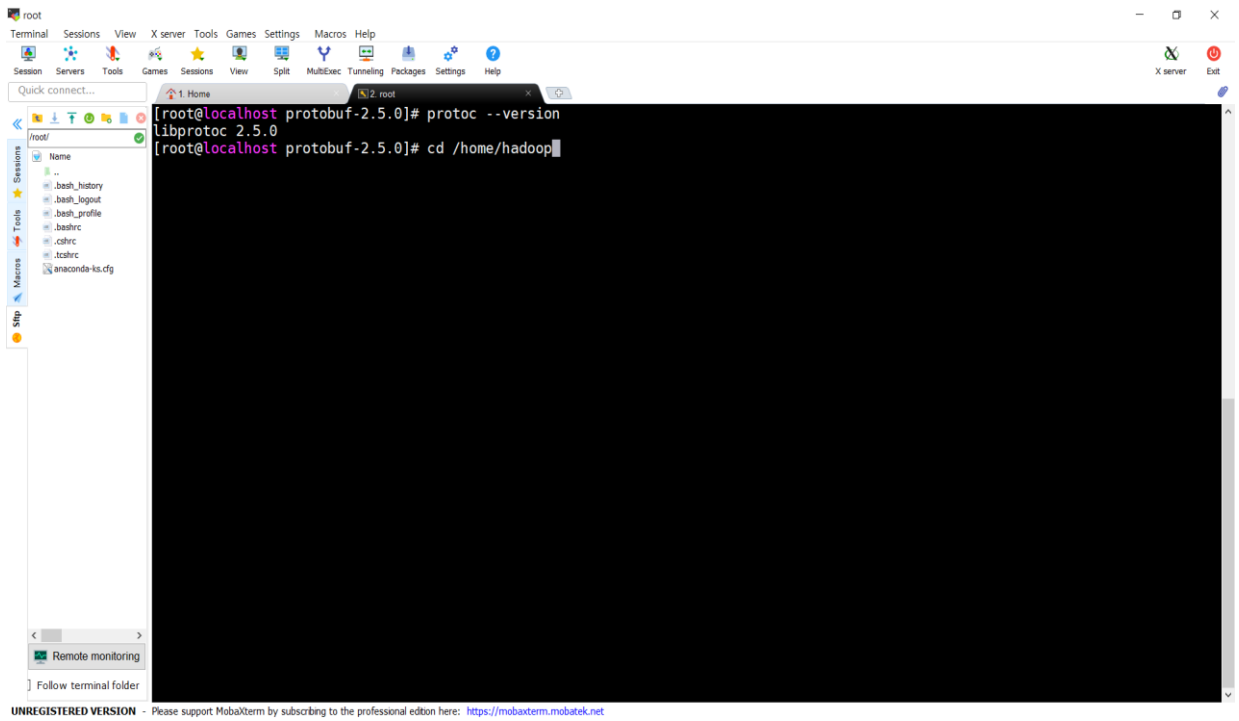
가상머신 실행



Filezilla 실행

hadoop 다운로드 사이트에서 다운로드 받은 hadoop-2.6.0.tar.gz 파일을 가상머신의 /home/hadoop 디렉토리에 복사

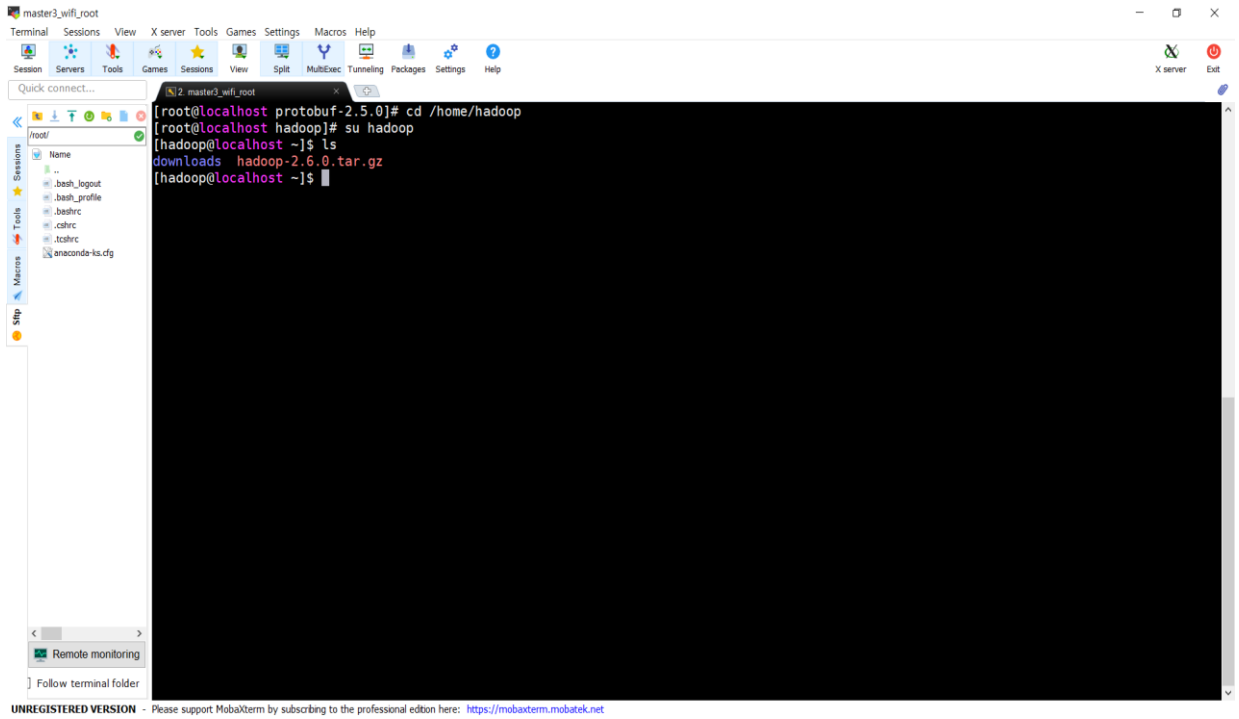
가상머신 실행



/home/hadoop 디렉토리로 이동

[root@localhost protobuf-2.5.0]# cd /home/hadoop

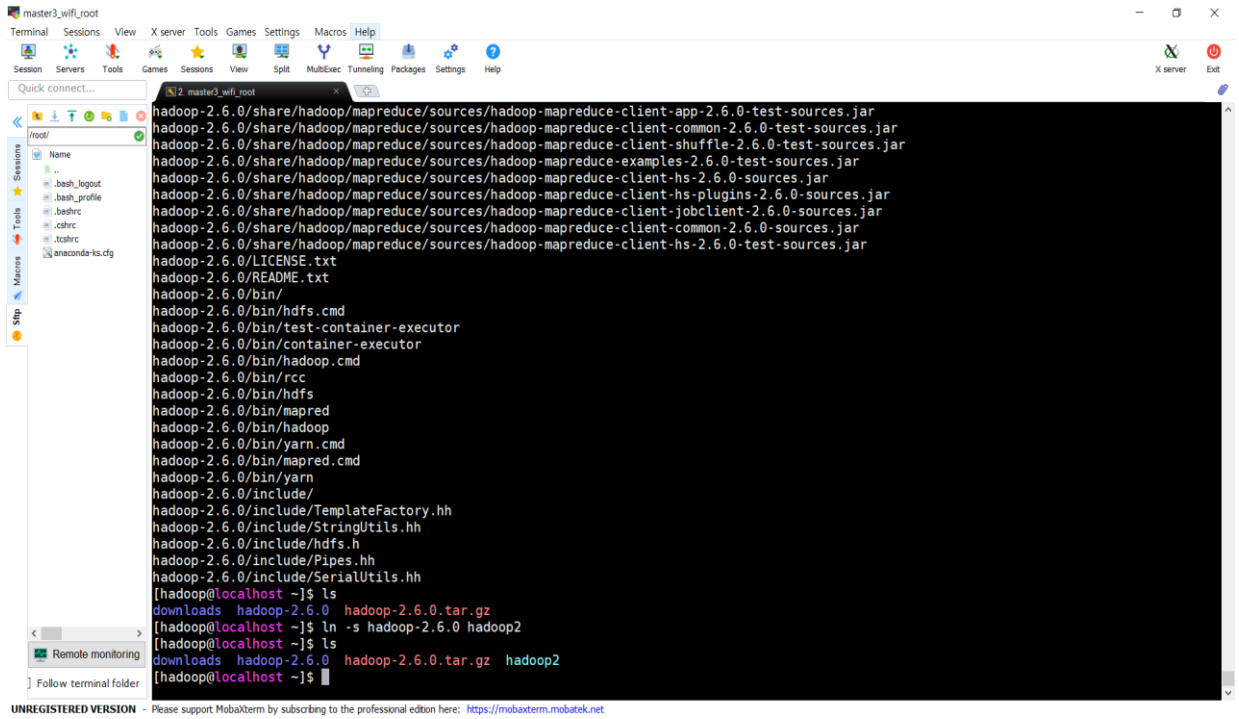
가상머신 실행



root 계정에서 hadoop 계정으로 계정 사용 변경

[root@localhost hadoop]# su hadoop
[hadoop@localhost ~]$ ls

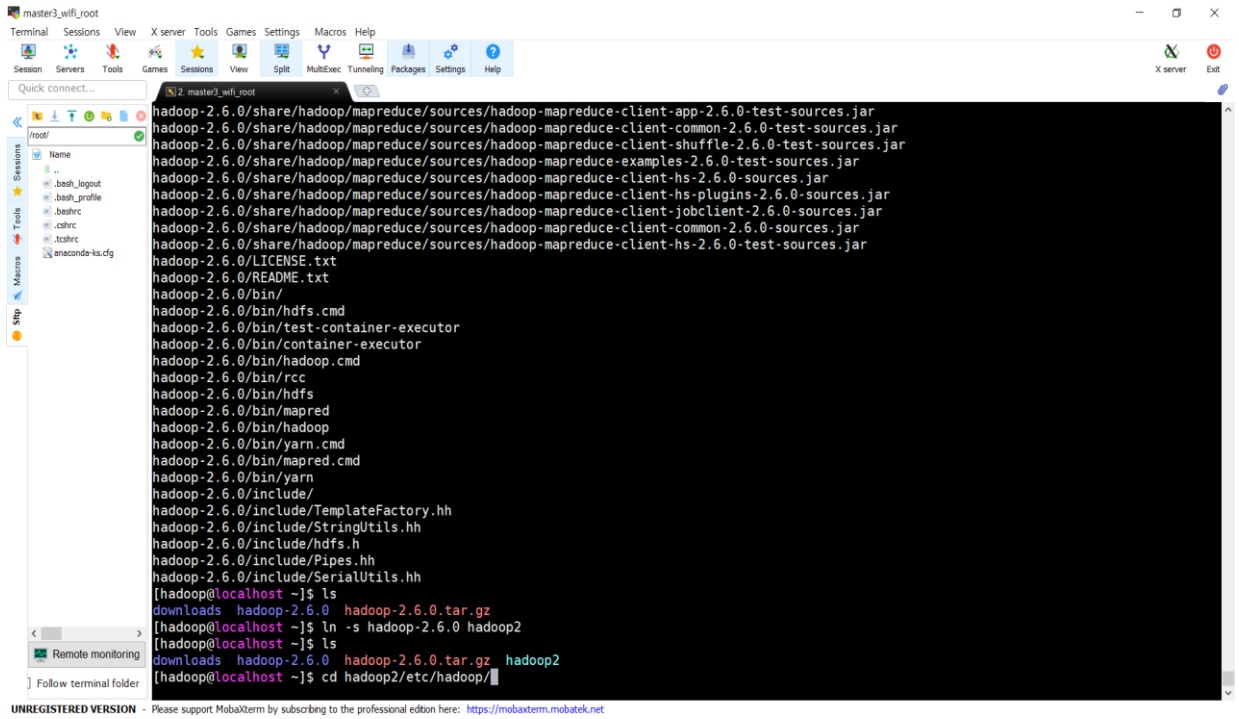
가상머신 실행



hadoop-2.6.0.tar.gz 파일 압축 해제 및 명칭 변경

[hadoop@localhost ~]$ tar xvzf hadoop-2.6.0.tar.gz (파일 압축 해제)
[hadoop@localhost ~]$ ln –s hadoop-2.6.0 hadoop2 (명칭 변경)

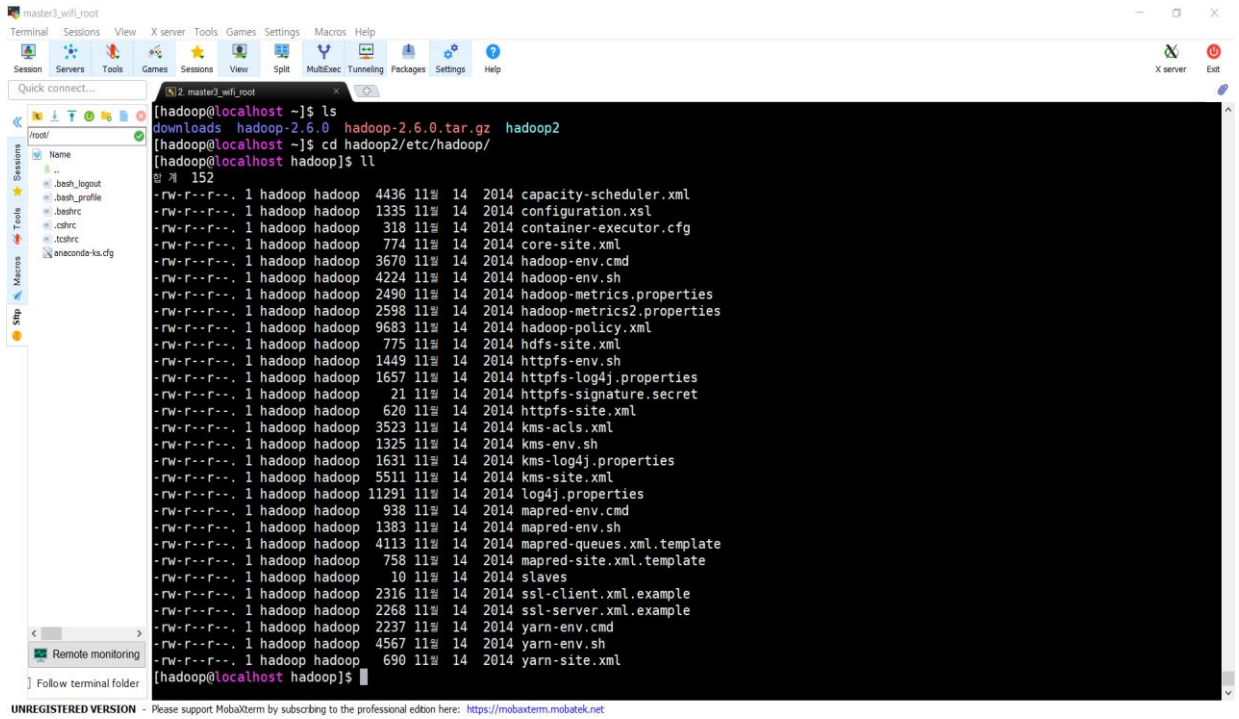
가상머신 실행



hadoop2/etc/hadoop/ 디렉토리로 이동

[hadoop@localhost ~]$ cd hadoop2/etc/hadoop/

가상머신 실행



hadoop2/etc/hadoop/ 디렉토리에서 "ll" 명령어로 파일 확인하고 환경설정해야 할 파일 확인

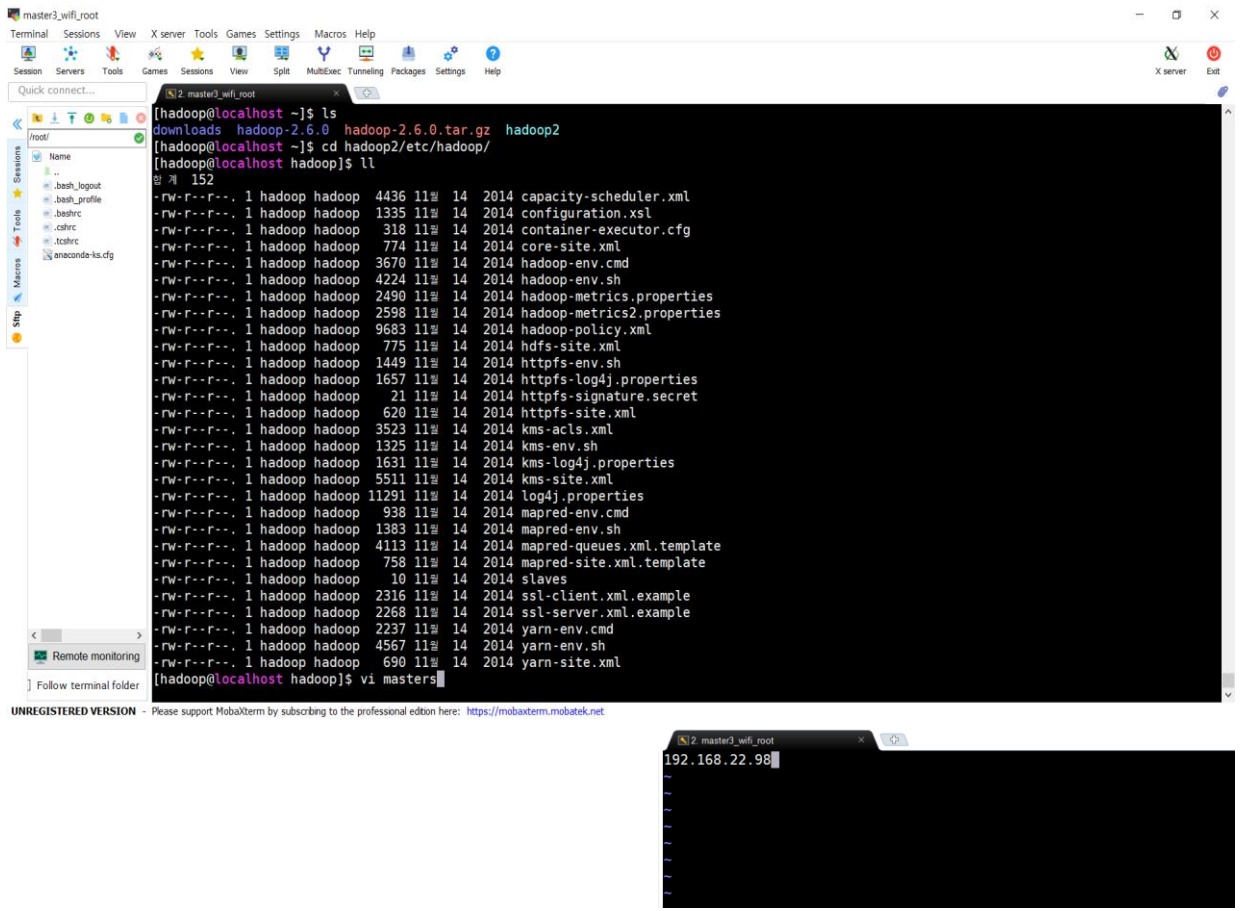
[hadoop@localhost hadoop]$ ll

환경 설정해야 할 파일들
core-site.xml
hadoop-env.sh
hadoop-env.sh
hdfs-site.xml
mapred-site.xml.template
slaves
yarn-env.sh

가상 분산 모드로 진행

(하나의 머신에 Master와 Slave 설치)

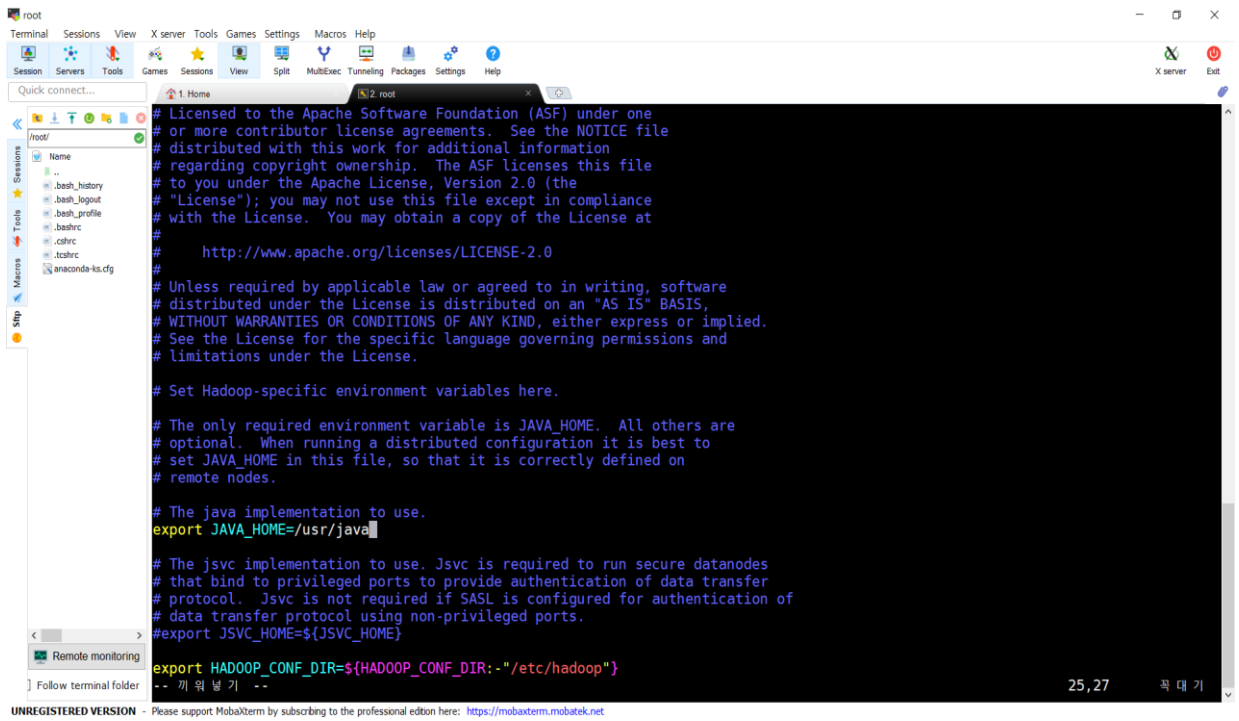
가상 분산 모드



hadoop2/etc/hadoop/ 디렉토리에서 vi 편집기로 masters 파일 생성
파일 내 내용은 현재 사용하는 host의 고정 ip 입력

[hadoop@localhost hadoop]$ vi masters

가상 분산 모드



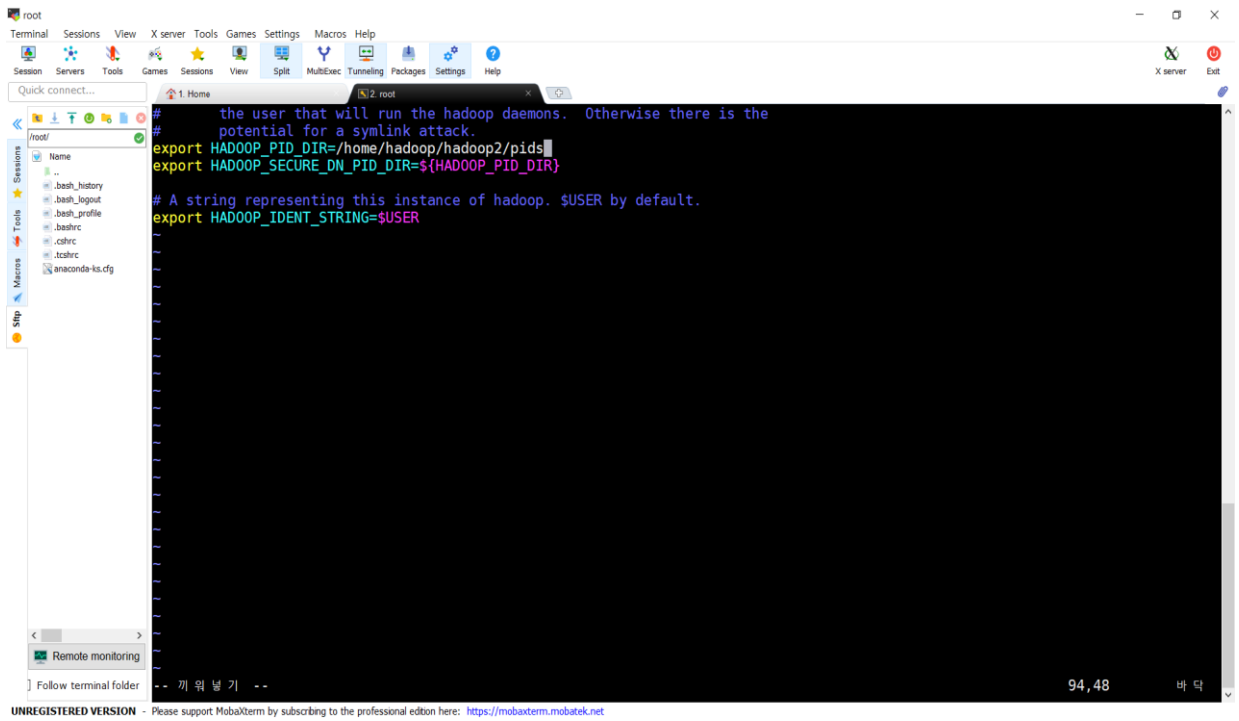
hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 hadoop-env.sh 파일 편집

[hadoop@localhost hadoop]$ vim hadoop-env.sh

"JAVA_HOME=${JAVA_HOME}"을 "JAVA_HOME=/usr/java"로 변경

"HADOOP_PID_DIR=${HADOOP_PID_DIR}"을
"HADOOP_PID_DIR=/home/hadoop/hadoop2/pids"로 변경

가상 분산 모드



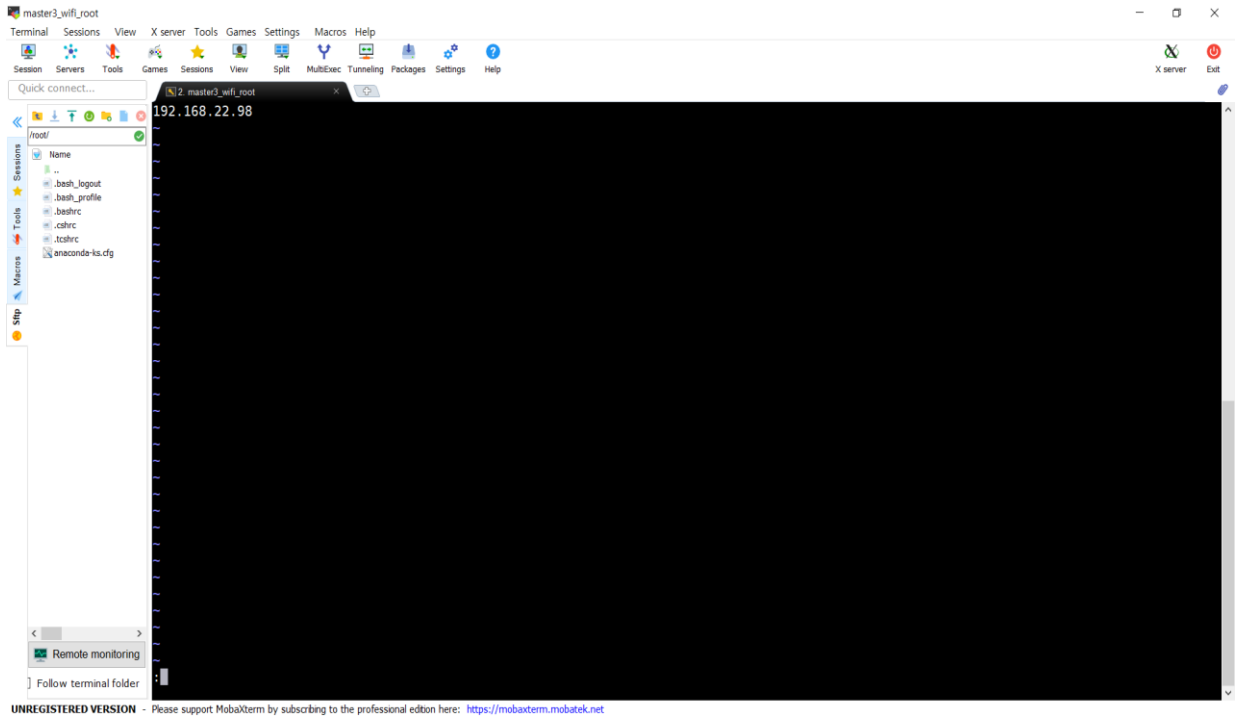
hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 hadoop-env.sh 파일 편집

[hadoop@localhost hadoop]$ vim hadoop-env.sh

"JAVA_HOME=${JAVA_HOME}"을 "JAVA_HOME=/usr/java"로 변경

"HADOOP_PID_DIR=${HADOOP_PID_DIR}"을
"HADOOP_PID_DIR=/home/hadoop/hadoop2/pids"로 변경

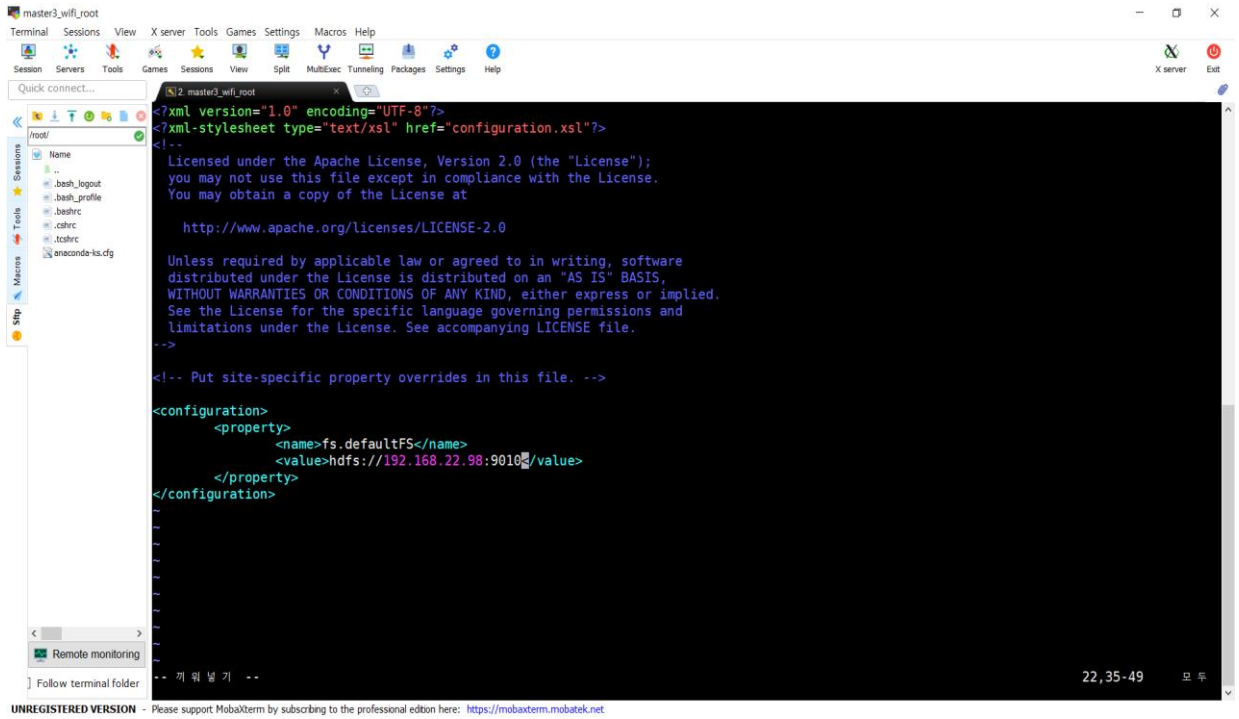
가상 분산 모드



hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 slaves 파일 편집
파일 내 내용은 현재 사용하는 host의 고정 ip 입력

[hadoop@localhost hadoop]$ vim slaves

가상 분산 모드



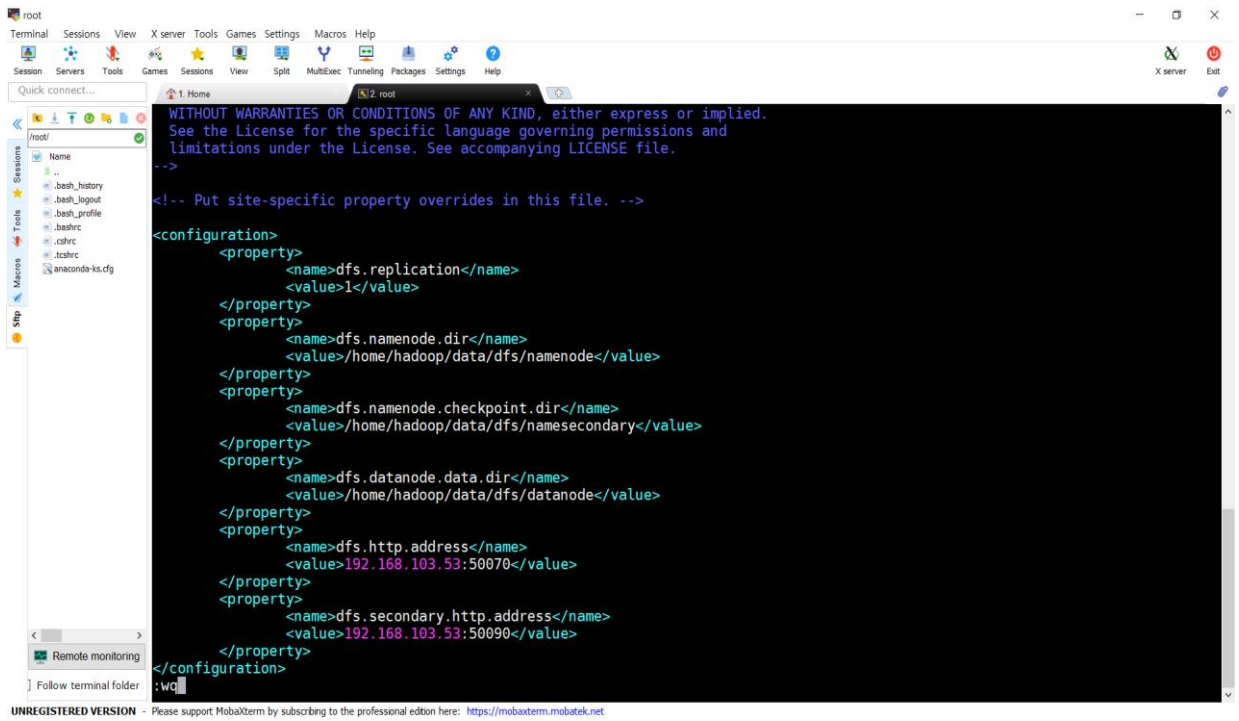
hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 core-site.xml 파일 편집

[hadoop@localhost hadoop]$ vim core-site.xml

* <configuration></configuration> 부분에서 안 부분을 수정해야함

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://호스트의 아이피:9010</value>
    </property>
</configuration>
```

가상 분산 모드



hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 hdfs-site.xml 파일 편집

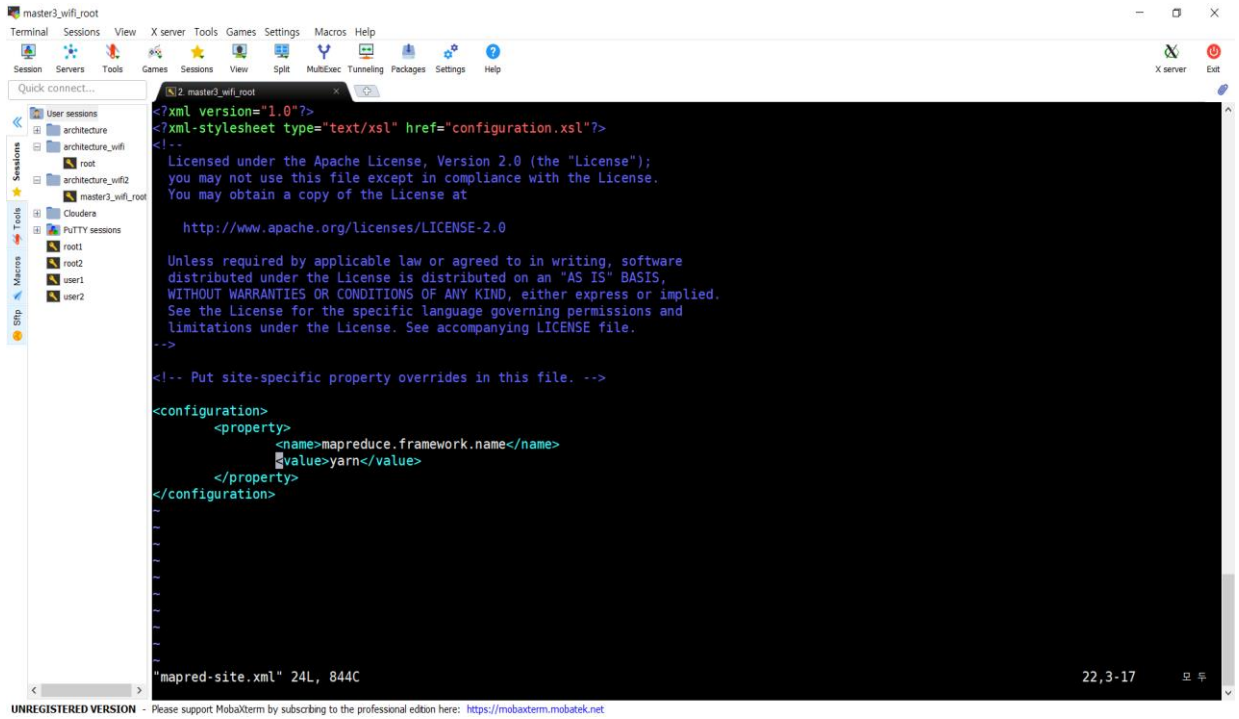
[hadoop@localhost hadoop]$ vim hdfs-site.xml

* <configuration></configuration> 부분에서 안 부분을 수정해야함

다음과 같음

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value> (1 : 복제본이 하나라는 뜻 -> 수행과제는 2로 해야함)
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/home/hadoop/data/dfs/namenode</value>
    </property>
    <property>
        <name>dfs.namenode.checkpoint.dir</name>
        <value>/home/hadoop/data/dfs/namesecondary</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/home/hadoop/data/dfs/datanode</value>
    </property>
    <property>
        <name>dfs.http.address</name>
        <value>고정 ip:50070</value>
    </property>
    <property>
        <name>dfs.secondary.http.address</name>
        <value>고정 ip:50090</value>
    </property>
</configuration>
```

가상 분산 모드



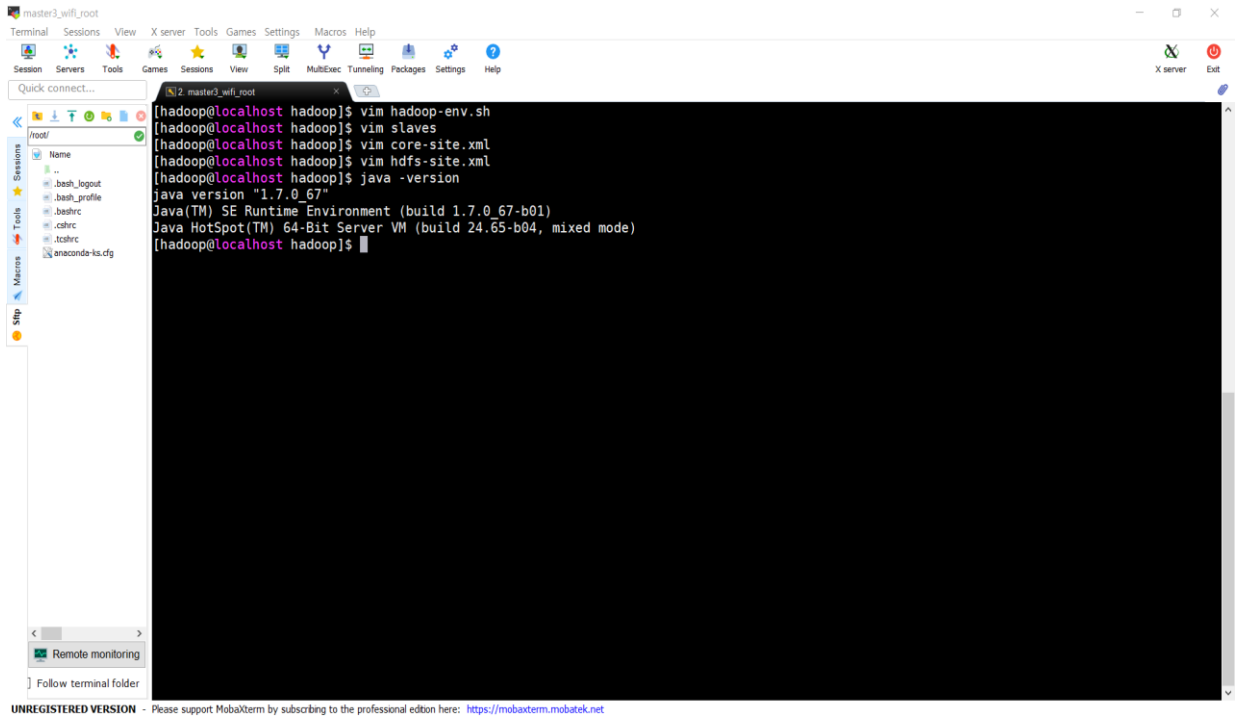
hadoop2/etc/hadoop/ 디렉토리에서 mapred-site.xml.template를 mapred-site.xml로 mv 이후 vim 편집기로 mapred-site.xml 파일 편집

[hadoop@localhost hadoop]$ mv mapred-site.xml.template mapred-site.xml
[hadoop@localhost hadoop]$ vim mapred-site.xml

* <configuration></configuration> 부분에서 안 부분을 수정해야함

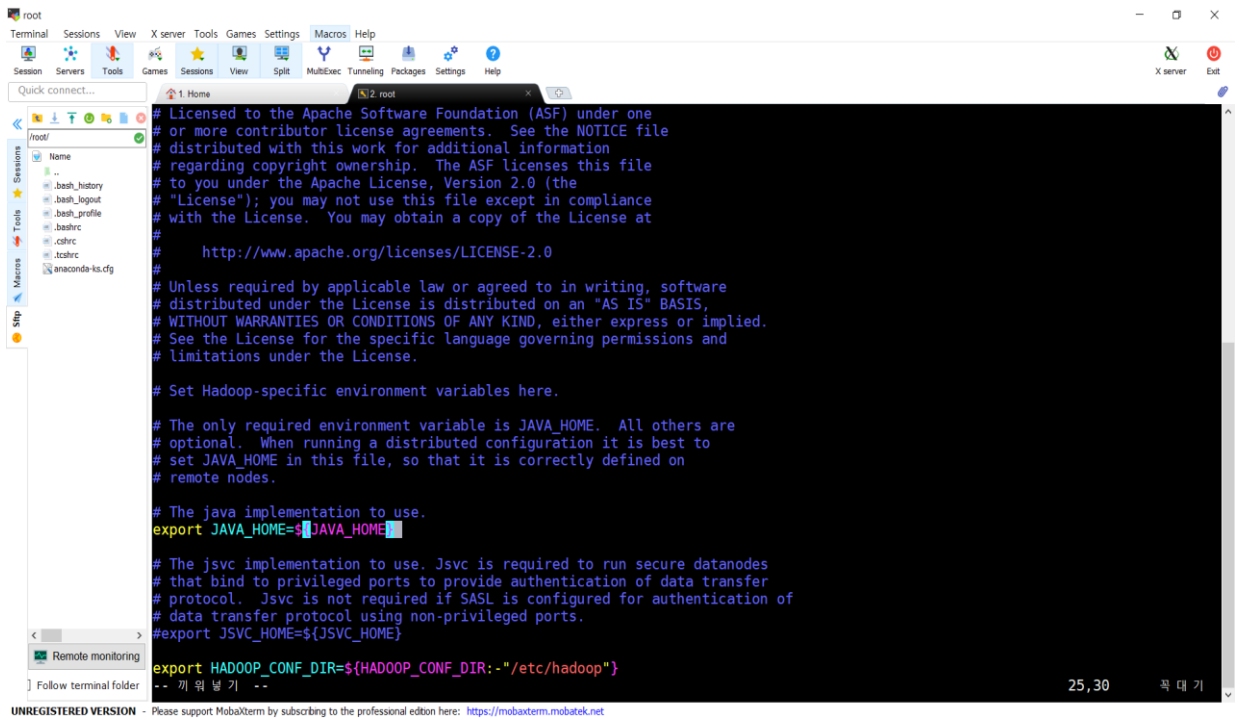
```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property
</configuration>
```

가상 분산 모드



[hadoop@localhost hadoop]$ java –version
java 버전 확인 발생

가상 분산 모드



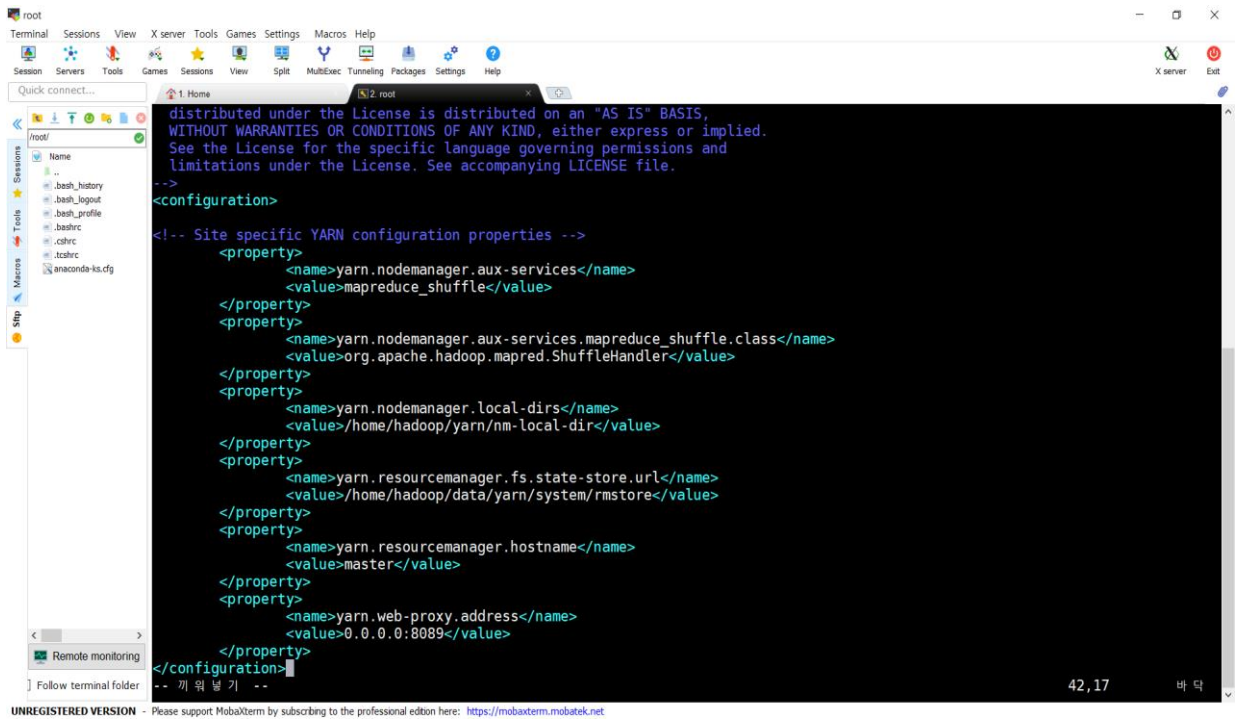
hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 hadoop-env.sh 파일 편집

[hadoop@localhost hadoop]$ vim hadoop-env.sh

JAVA_HOME 부분 수정

export JAVA_HOME=/usr/java를
export JAVA_HOME=${JAVA_HOME}으로 변경

가상 분산 모드



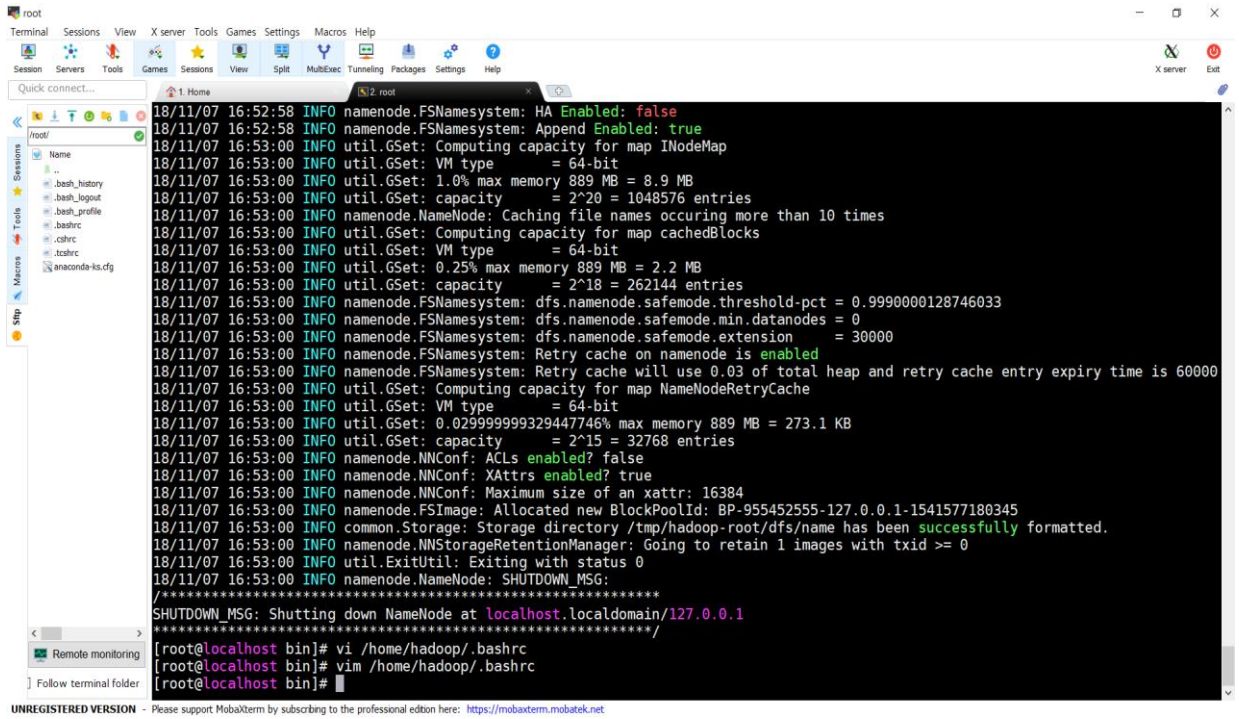
hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 yarn-site.xml 파일 편집

[hadoop@localhost hadoop]$ vim yarn-site.xml

<configuration></configuration> 부분 안에 수정해야함

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
    <property>
        <name>yarn.nodemanager.local-dirs</name>
        <value>/home/hadoop/yarn/nm-local-dir</value>
    </property>
    <property>
        <name>yarn.resourcemanager.fs.state-store.url</name>
        <value>/home/hadoop/data/yarn/system/rmstore</value>
    </property>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>master</value> (root @ localhost 에서 localhost에 해당하는 명을 작성해야함)
    </property>
    <property>
        <name>yarn.web-proxy.address</name>
        <value>0.0.0.0:8089</value>
    </property>
</configuration>
```

가상 분산 모드



hadoop2/bin 디렉토리로 이동
/home/hadoop/.bashrc 파일을 source로 실행해 적용
hdfs namenode –format 실행

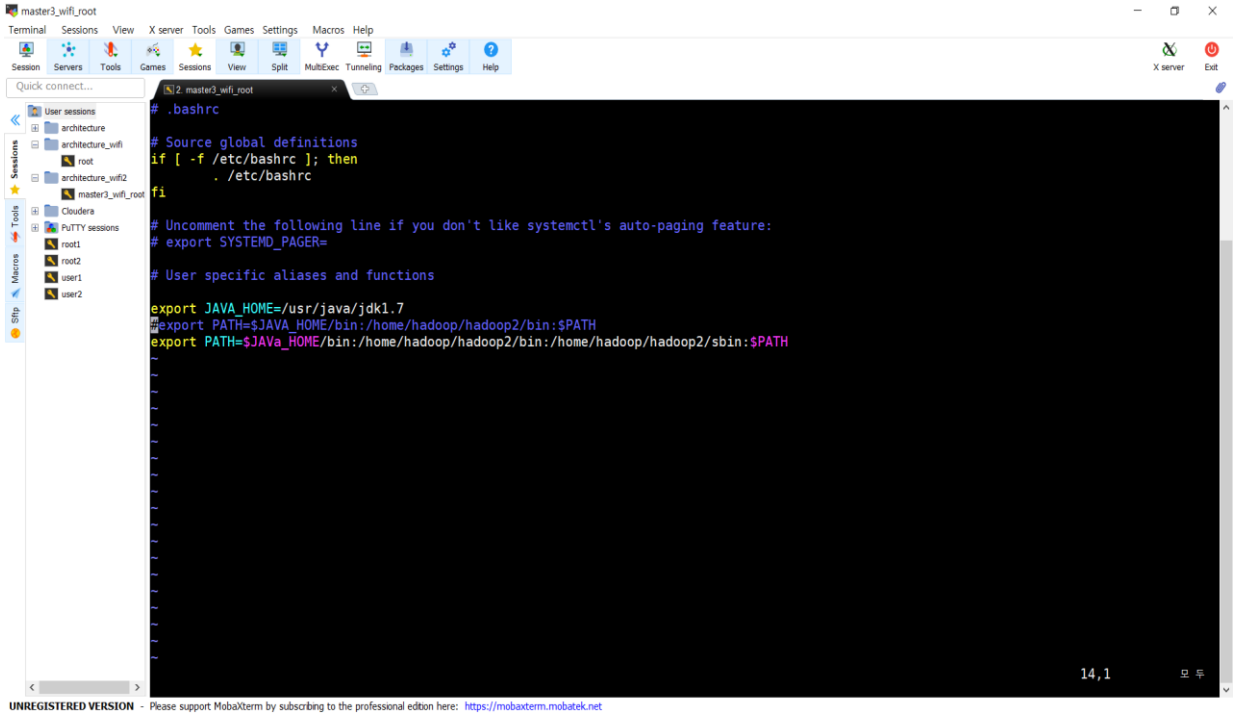
[root@localhost hadoop]# cd hadoop2/bin

vim으로 /home/hadoop/.bashrc 에 들어가서
export PATH=$JAVA_HOME/bin:/home/hadoop/hadoop/bin:$PATH 를
export PATH=$JAVA_HOME/bin:/home/hadoop/hadoop2/bin:$PATH 로 변경한 후

Source /home/hadoop/.bashrc 실행

[root@localhost bin]# source /home/hadoop/.bashrc
[root@localhost bin]# hdfs namenode -format

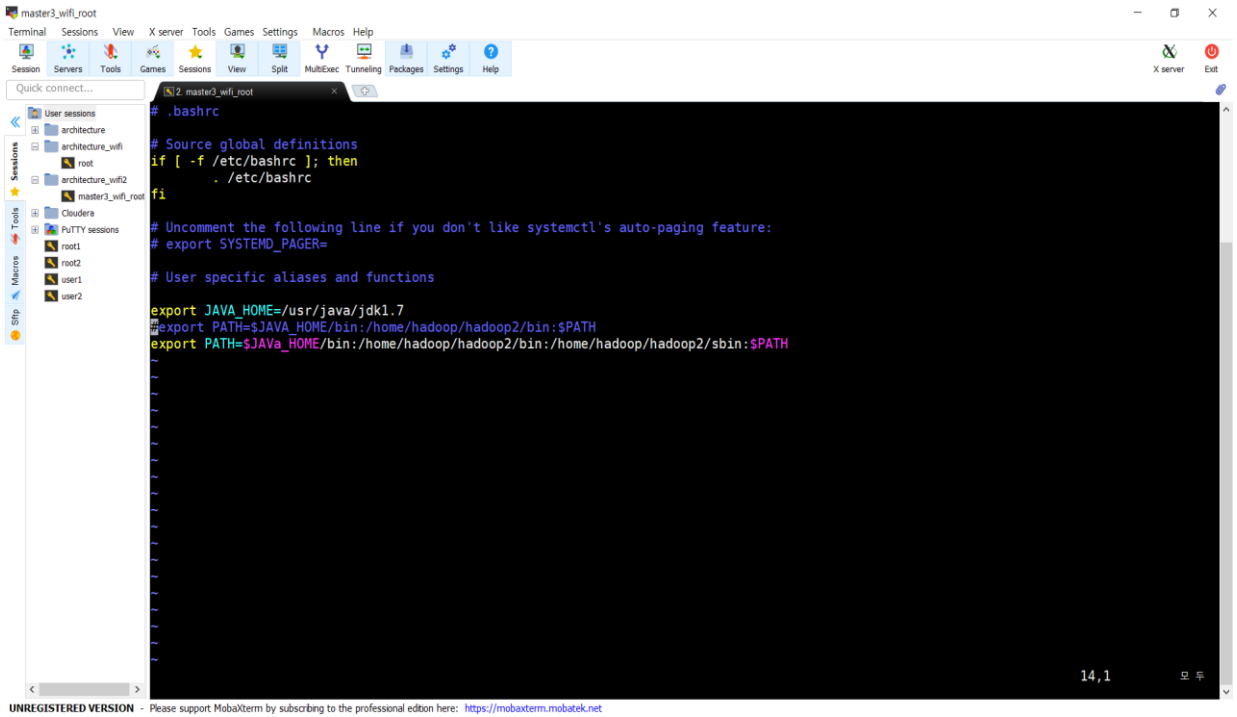
가상 분산 모드



hadoop2/bin 디렉토리로에서 hadoop2/sbin 디렉토리로 이동

[hadoop@localhost bin]$ cd ..
[hadoop@localhost hadoop2]$ cd sbin/
[hadoop@localhost sbin]$

가상 분산 모드



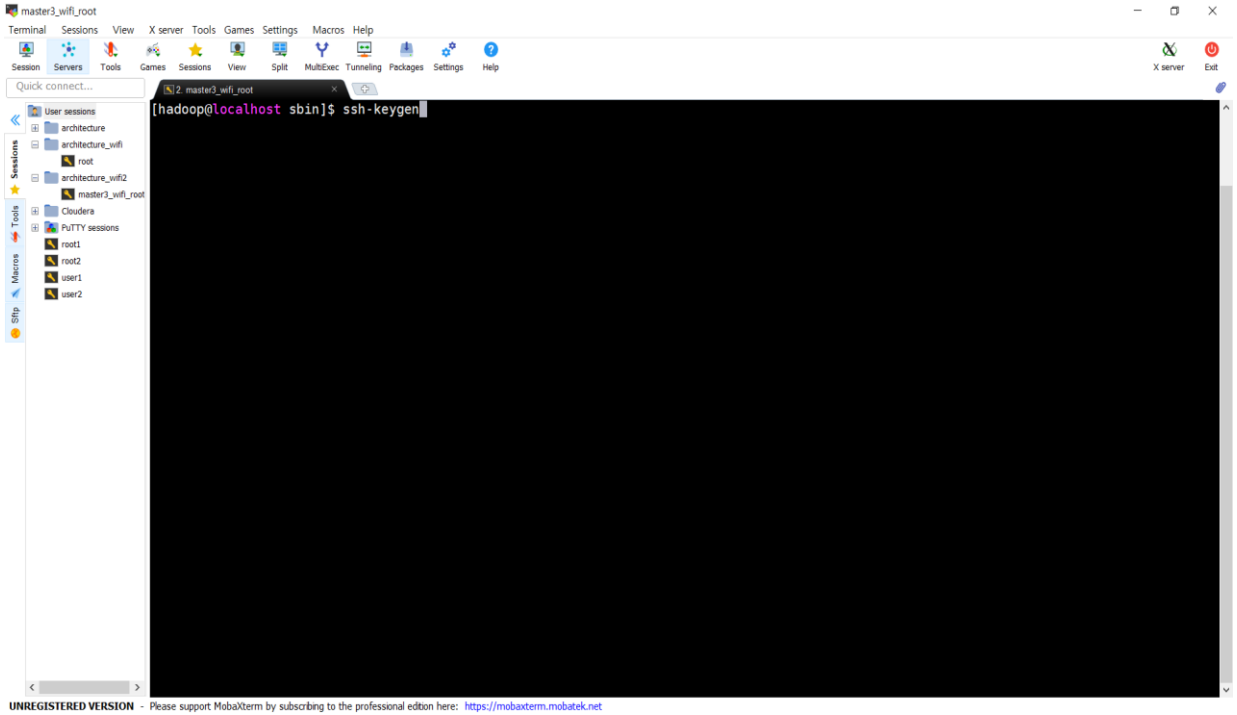
hadoop2/sbin 디렉토리에서 vim 편집기로 /home/hadoop/.bashrc 내용 편집

[hadoop@localhost sbin]$ vim /home/hadoop/.bashrc

export PATH=$JAVA_HOME/bin:/home/hadoop/hadoop2/bin:$PATH 를

export PATH=$JAVA_HOME/bin:/home/hadoop/hadoop2/bin:/home/hadoop/hadoop2/sbin:$PATH로 변경

가상 분산 모드

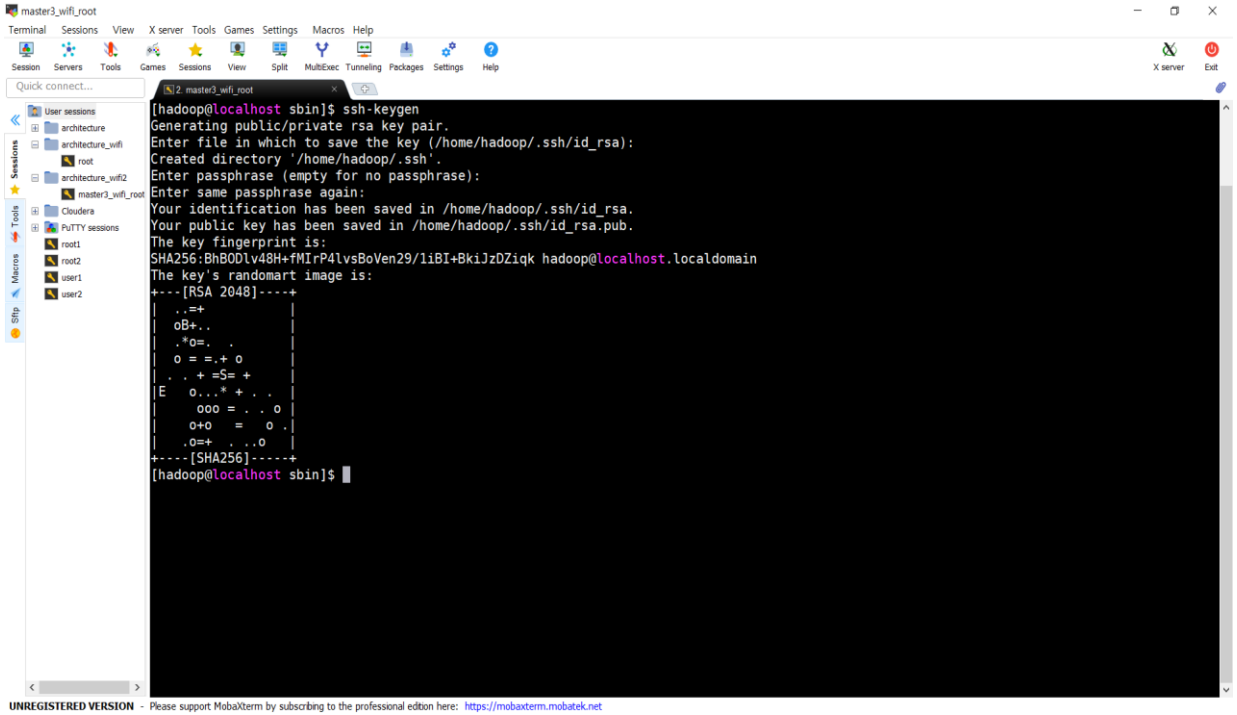


hadoop2/sbin 디렉토리 위치에서
hadoop 계정의 ssh key 등록 명령어 실행

[hadoop@localhost sbin]$ ssh-keygen

Ssh는 서버 간 통신을 할때 ssh key를 통해 통신!!!!!!!

가상 분산 모드



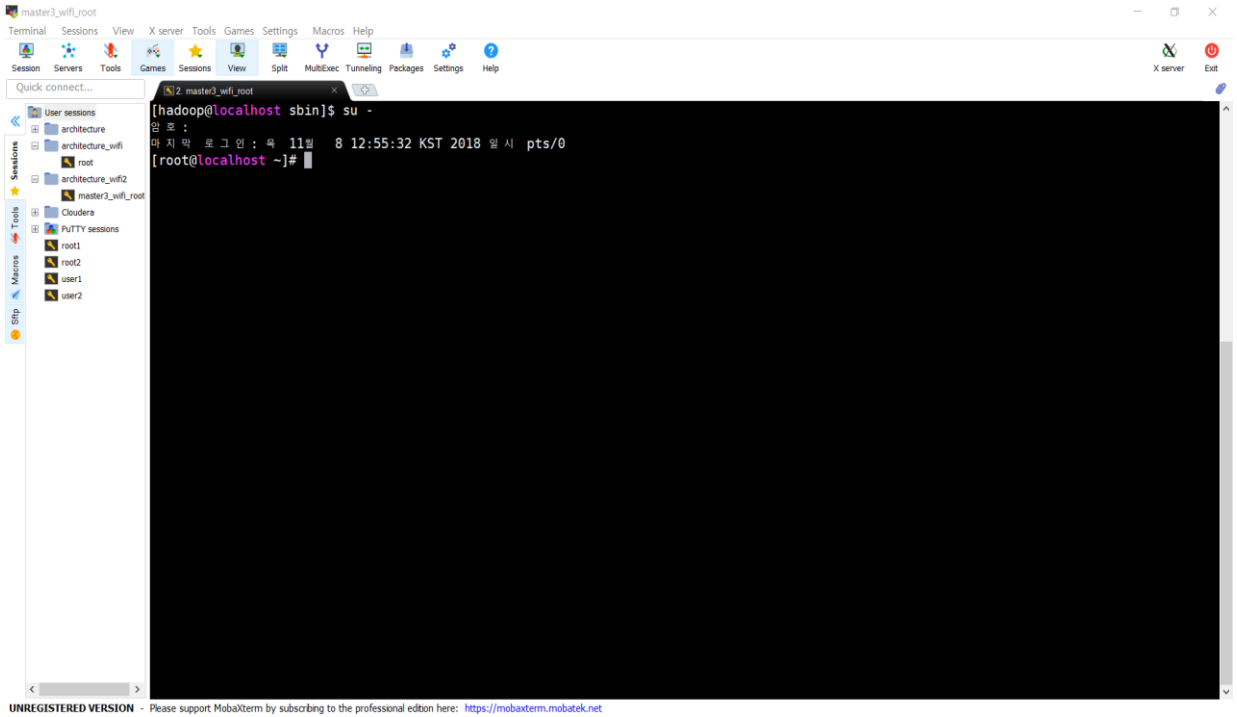
hadoop2/sbin 디렉토리 위치에서
hadoop 계정의 ssh key 등록 명령어 실행

[hadoop@localhost sbin]$ ssh-keygen

Ssh는 서버 간 통신을 할때 ssh key를 통해 통신!!!!!!!

*****
생성할 때 총 3번의 enter key를 눌러야함. 아무 입력 없이 3번 enter key 누르기!!!!

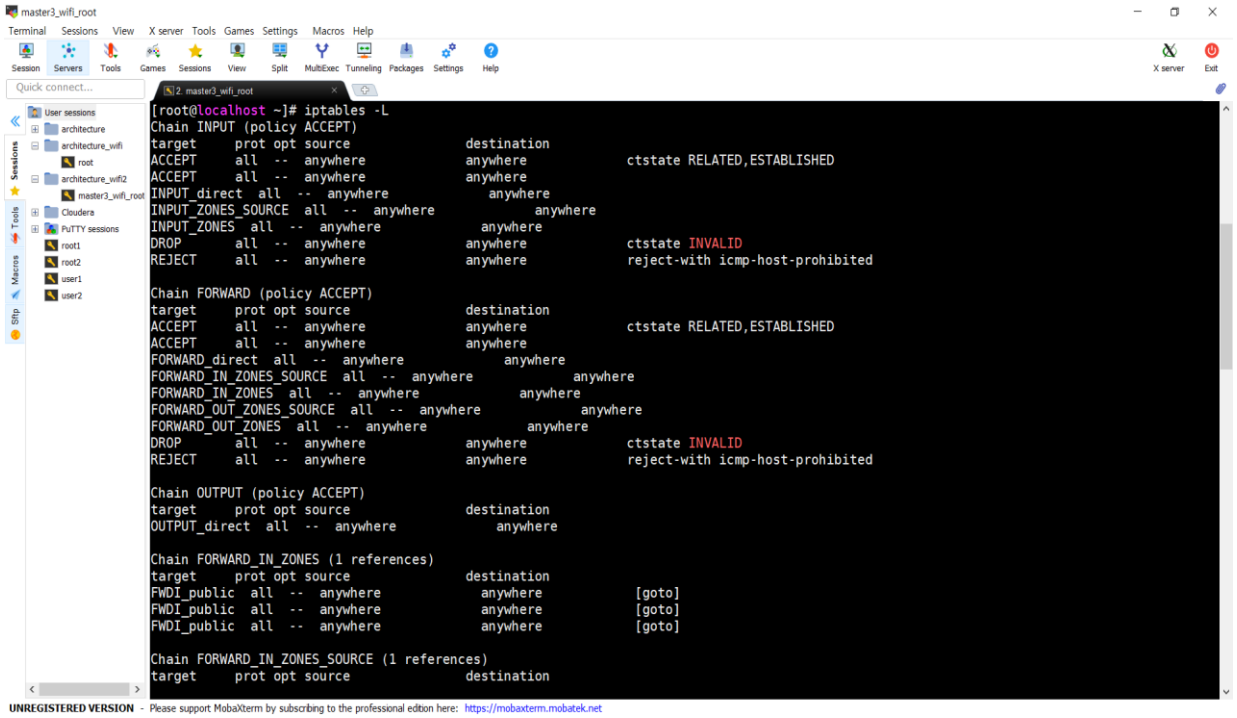
가상 분산 모드



hadoop2/sbin 디렉토리 위치에서
root 계정로 변경 후 방화벽 확인 및 종료

[hadoop@localhost sbin]$ su -
[root@localhost ~]#

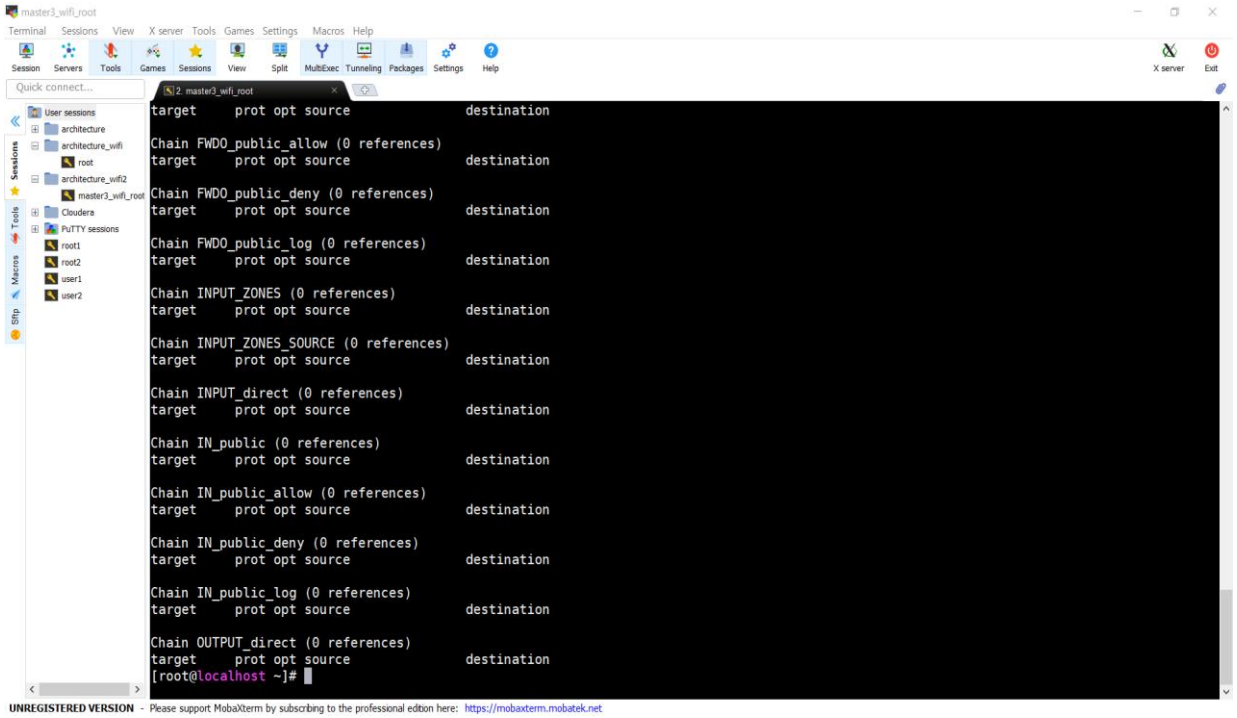
가상 분산 모드



hadoop2/sbin 디렉토리 위치에서
root 계정로 변경 후 방화벽 확인 및 종료

[root@localhost ~]# iptables –L
[root@loalchost ~]# iptables –F
[root@localhost ~]# iptables -L

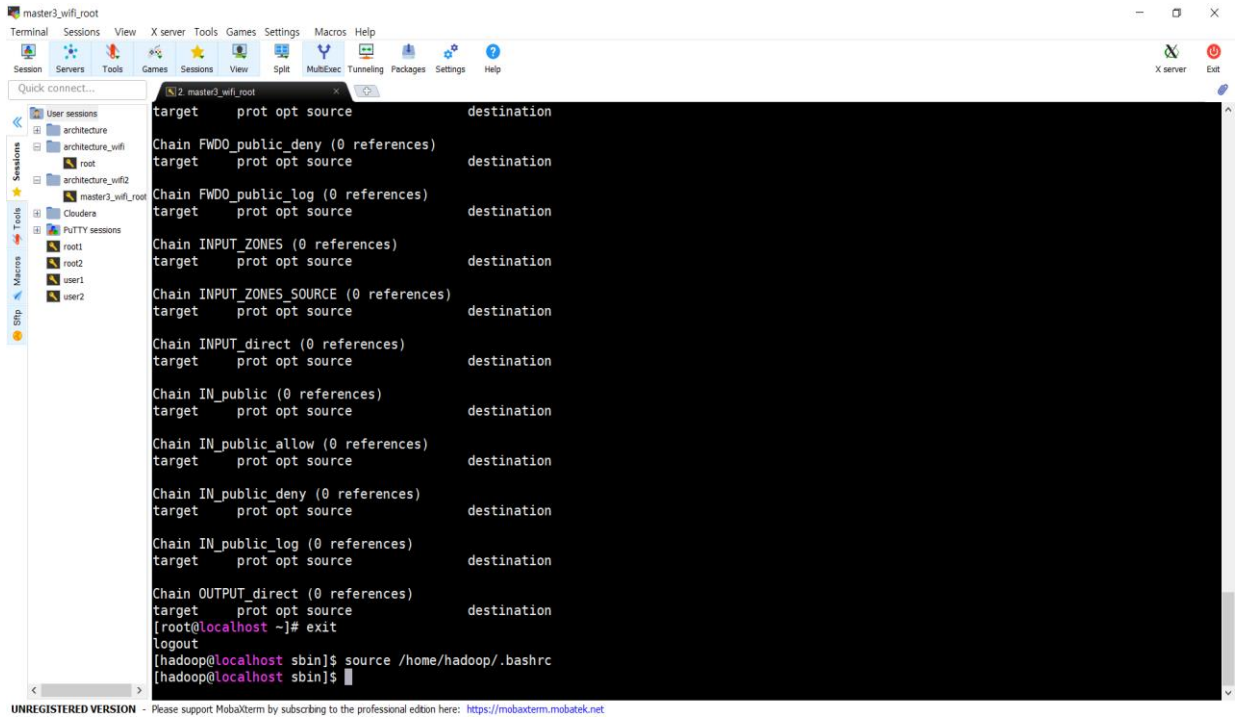
가상 분산 모드



hadoop2/sbin 디렉토리 위치에서
root 계정로 변경 후 방화벽 확인 및 종료

[root@localhost ~]# iptables –L
[root@loalchost ~]# iptables –F
[root@localhost ~]# iptables -L

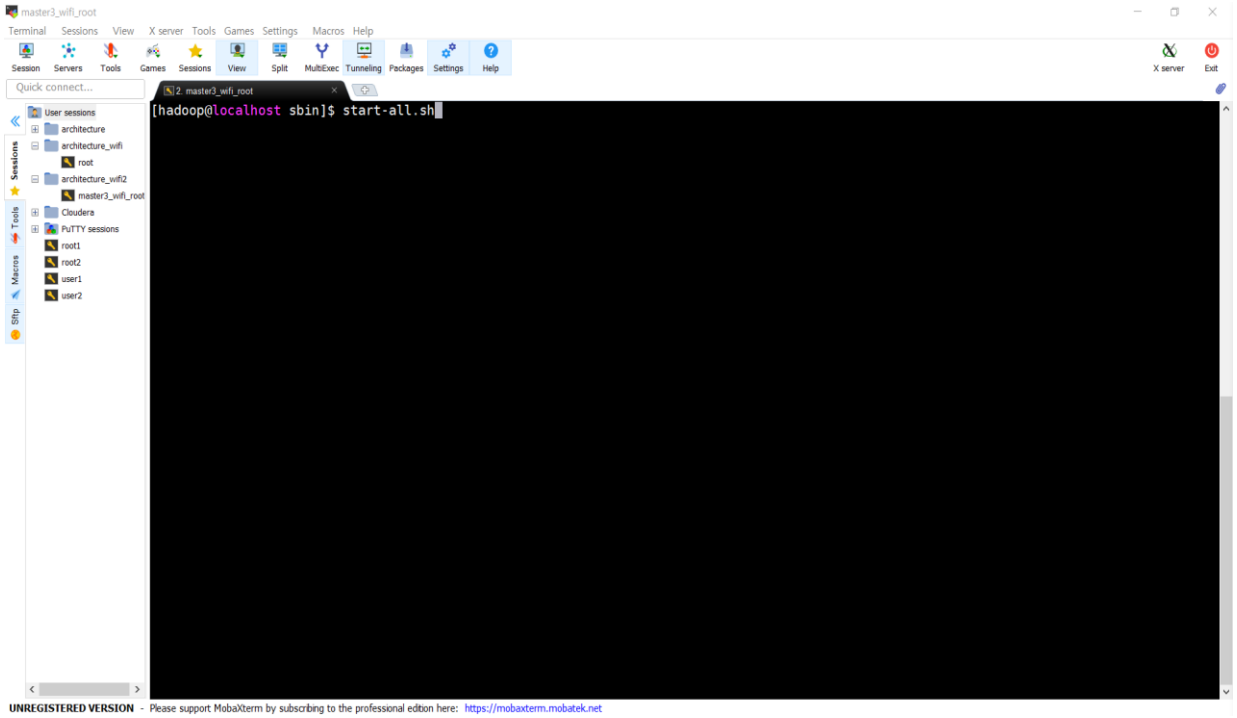
가상 분산 모드



root 계정 exit 한 뒤 /home/hadoop/.bashrc를 source로 실행

[root@localhost ~]# exit
[hadoop@localhost sbin]$ source /home/hadoop/.bashrc

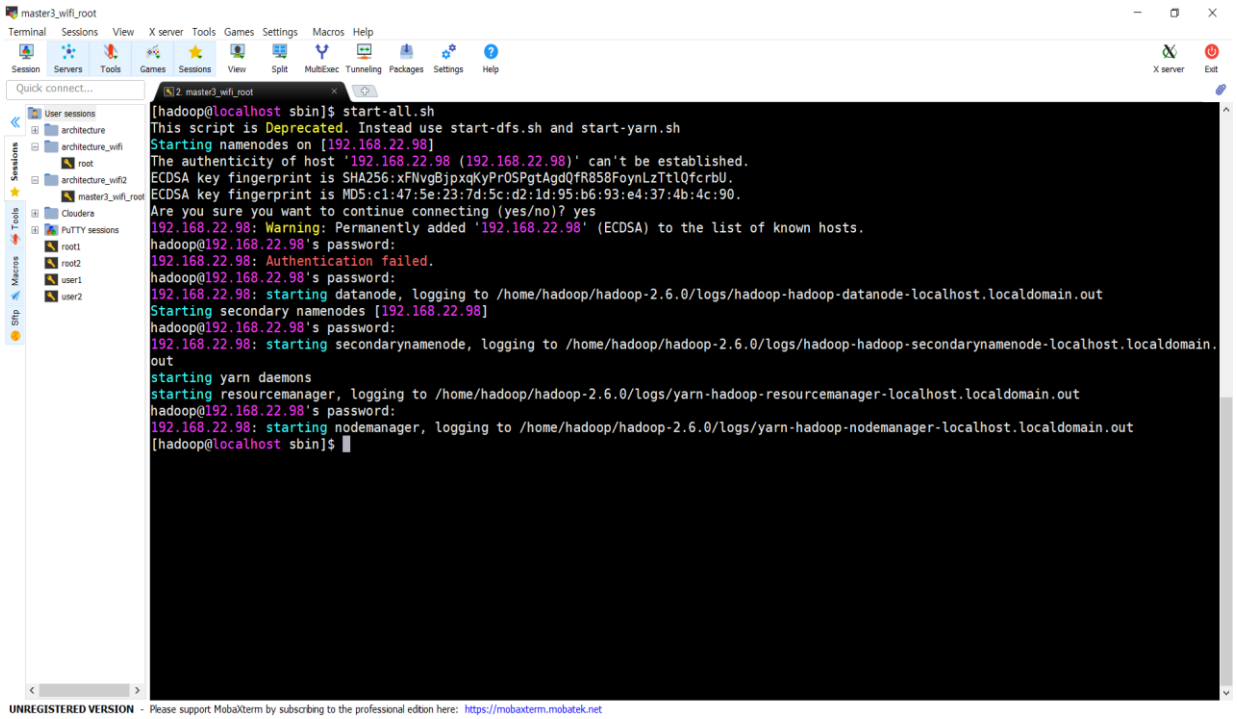
가상 분산 모드



Start-all.sh 실행

[hadoop@localhost sbin]$ start-all.sh

가상 분산 모드



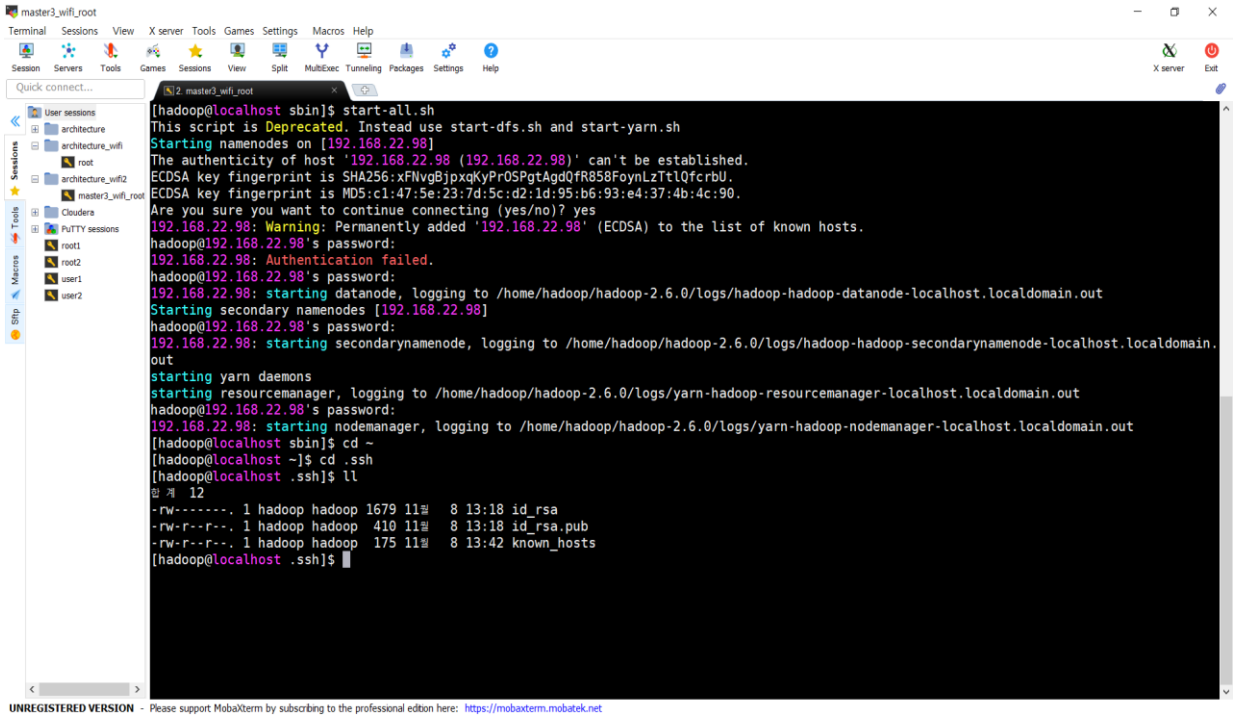
Start-all.sh 실행

[hadoop@localhost sbin]$ start-all.sh

Are you sure you want to continue connecting (yes/no)? **Yes** 입력

단계적으로 hadoop 계정의 비밀번호 입력하기....

가상 분산 모드



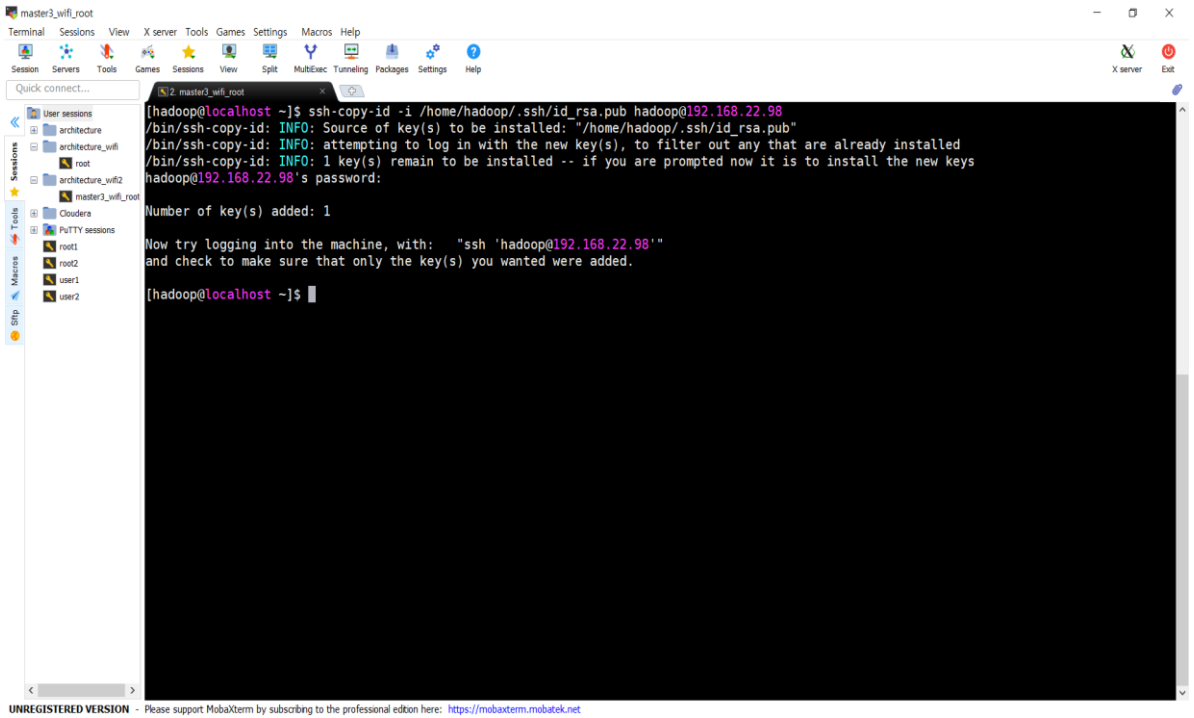
.ssh 디렉토리로 이동한 뒤 key 확인

[hadoop@localhost sbin]$ cd ~
[hadoop@localhost ~]$ cd .ssh
[hadoop@localhost .ssh]$ ll

rsa 키 확인한 후 상위 디렉토리로 이동

[hadoop@localhost .ssh]$ cd ..

가상 분산 모드

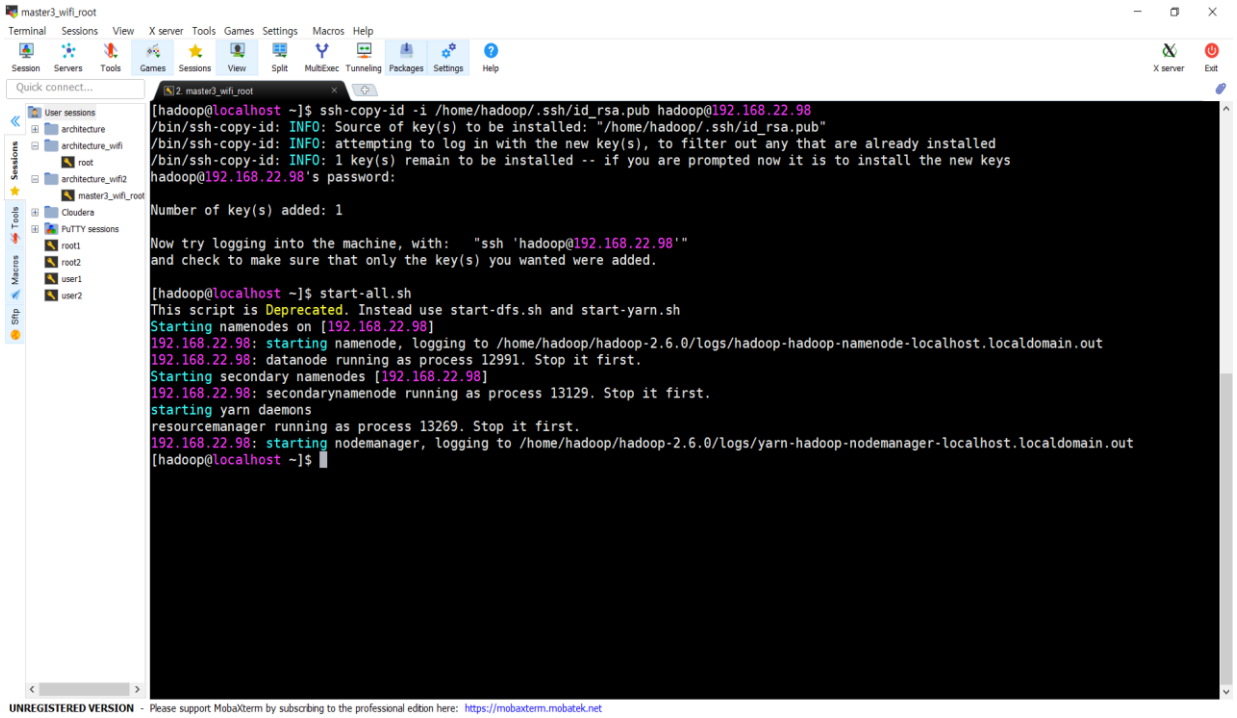


Id_rsa.pub key를 hadoop 계정으로 복사

[hadoop@localhost ~]$ ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub hadoop@192.168.22.98

hadoop 계정의 password 입력

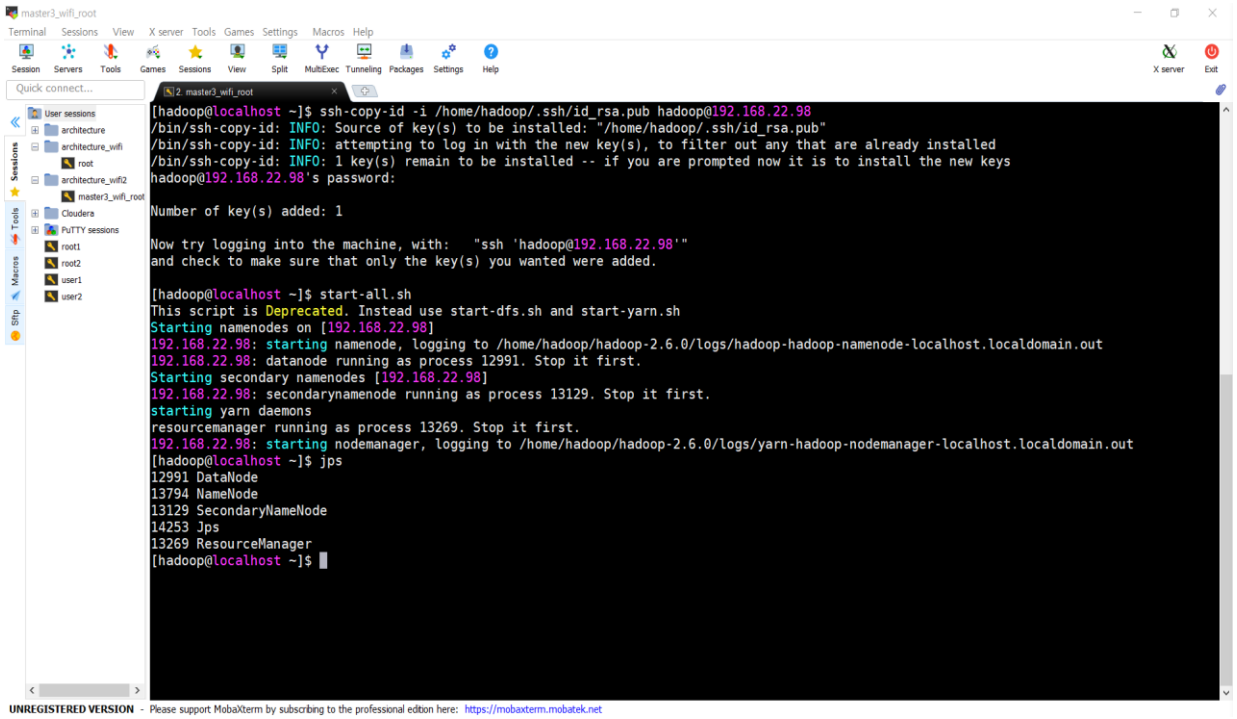
가상 분산 모드



Start-all.sh 실행

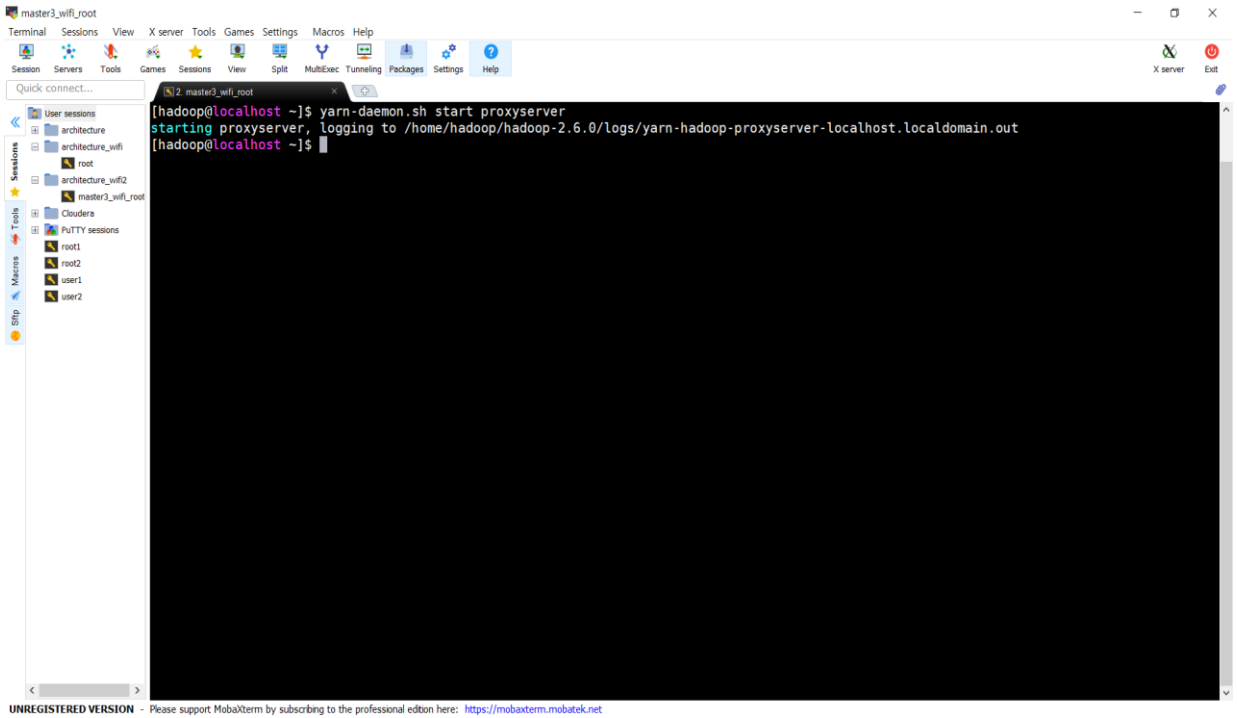
[hadoop@localhost ~]$ start-all.sh

가상 분산 모드



프로세스 세팅 확인

[hadoop@localhost ~]$ jps



프록시 서버 시작

[hadoop@localhost ~]$ yarn-daemon.sh start proxyserver

가상 분산 모드

```
drwx------. 3 hadoop hadoop 4096 11월   8 20:51 datanode
drwxrwxr-x. 3 hadoop hadoop 4096 11월   8 20:51 namenode
drwxrwxr-x. 3 hadoop hadoop 4096 11월   8 20:51 namesecondary
[hadoop@team4 dfs]$ cd datanode
[hadoop@team4 datanode]$ ll
합계 4
drwxrwxr-x. 3 hadoop hadoop 4096 11월   7 23:57 current
[hadoop@team4 datanode]$ stop-all.sh
This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh
Stopping namenodes on [team4]
team4: stopping namenode
192.168.100.107: no datanode to stop
Stopping secondary namenodes [team4]
team4: stopping secondarynamenode
stopping yarn daemons
stopping resourcemanager
192.168.100.107: stopping nodemanager
no proxyserver to stop
[hadoop@team4 datanode]$ rm -r datanode
rm: cannot remove `datanode': 그런 파일이나 디렉터리가 없습니다
[hadoop@team4 datanode]$ cd ..
[hadoop@team4 dfs]$ rm -r datanode
[hadoop@team4 dfs]$ ll
합계 8
drwxrwxr-x. 3 hadoop hadoop 4096 11월   8 20:53 namenode
drwxrwxr-x. 3 hadoop hadoop 4096 11월   8 20:53 namesecondary
[hadoop@team4 dfs]$ mkdir datanode
[hadoop@team4 dfs]$ cd datanode
[hadoop@team4 datanode]$ hdfs namenode -format
18/11/08 20:54:40 INFO namenode.NameNode: STARTUP_MSG:
```

Namenode 미실행시
$hdfs namenode –format를 통해 포맷하여 준다

Datanode 미실행시
/home/hadoop/data/dfs/datanode 를 삭제 후 hadoop을 재실행(start-def.sh , start-yarn.sh)

가상 분산 모드

```
[root@team4 etc]# vim hosts
[root@team4 etc]#
```

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.100.107 team4
192.168.100.108 Master
192.168.100.104 slav101
192.168.100.105 slave02
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:
```

주키퍼설치를 위한 사전 작업

1. Cluser를 구성하고자 하는 server 중 3대의 server Root 계정에서 user, pw 설정
   #adduser zookeeper
   #passwd zookeeper
   이후 비밀번호 설정

2. Host에서 클러스를 구성하고자 하는 가상머신의 호스트 네임과 ip 주소를 등록한다.

가상 분산 모드

```
[zookeeper@team4 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zookeeper/.ssh/id_rsa):
Created directory '/home/zookeeper/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/zookeeper/.ssh/id_rsa.
Your public key has been saved in /home/zookeeper/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:IwkklanaaWfBr3b8MFUdc+ciL1NscmtBbY5IeX6e0uA zookeeper@team4
The key's randomart image is:
+---[RSA 2048]----+
|   ..oo        ..+o|
|    oo        oo.oB|
|    ..  .    .o+O+o|
|   .   . .o   .Xo+o|
|  o . .o So  = *o.|
| . . . ..o... E o.|
|        o oo    .  |
|         o oo      |
|          .. .     |
+----[SHA256]-----+
```

```
[zookeeper@team4 ~]$ ssh-copy-id -i .ssh/id_rsa.pub zookeeper@192.168.100.107
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/id_rsa.pub"
The authenticity of host '192.168.100.107 (192.168.100.107)' can't be established.
ECDSA key fingerprint is SHA256:MKMZIPwu483V7gXT3c8ENO86BNoH8Wk6bksVk2jxUko.
ECDSA key fingerprint is MD5:a2:d0:75:55:ab:b5:ad:62:34:30:b6:43:e2:86:41:b1.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are alr
eady installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to inst
all the new keys
zookeeper@192.168.100.107's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'zookeeper@192.168.100.107'"
and check to make sure that only the key(s) you wanted were added.

[zookeeper@team4 ~]$ ssh-copy-id -i .ssh/id_rsa.pub zookeeper@192.168.100.108
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/id_rsa.pub"
The authenticity of host '192.168.100.108 (192.168.100.108)' can't be established.
ECDSA key fingerprint is SHA256:aC+8toxSzWZxjsUfMoZSBXZd36eBupTW2qxlNh+DTZo.
ECDSA key fingerprint is MD5:f1:be:4e:87:ef:64:1c:7a:de:b8:43:8b:cf:6f:b4:30.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are alr
eady installed
```

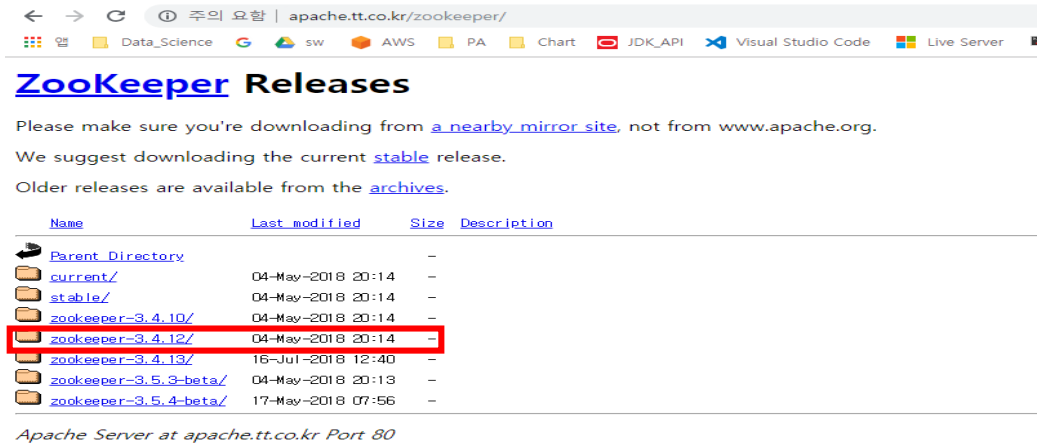
```
[zookeeper@team4 ~]$ ssh zookeeper@192.168.100.108
[zookeeper@Master ~]$ exit
logout
Connection to 192.168.100.108 closed.
[zookeeper@team4 ~]$
```

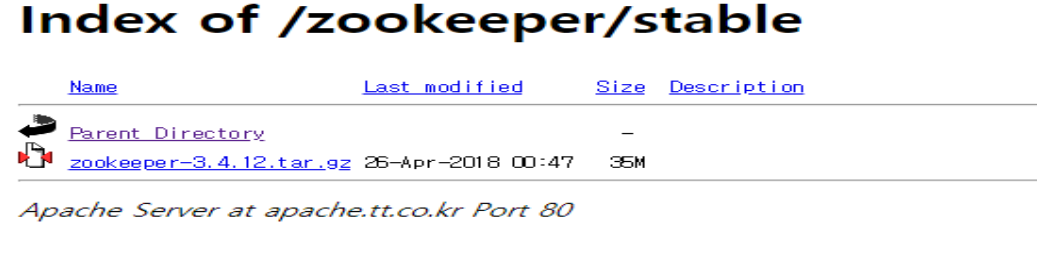
3. 각 서버의 zookeeper 계정에서 ssh-key 값을 설정한다.
   $ssh-keygen
   :이 때 '엔터'를 연속으로 클릭하여 비밀 번호를 설정하지 않는다.

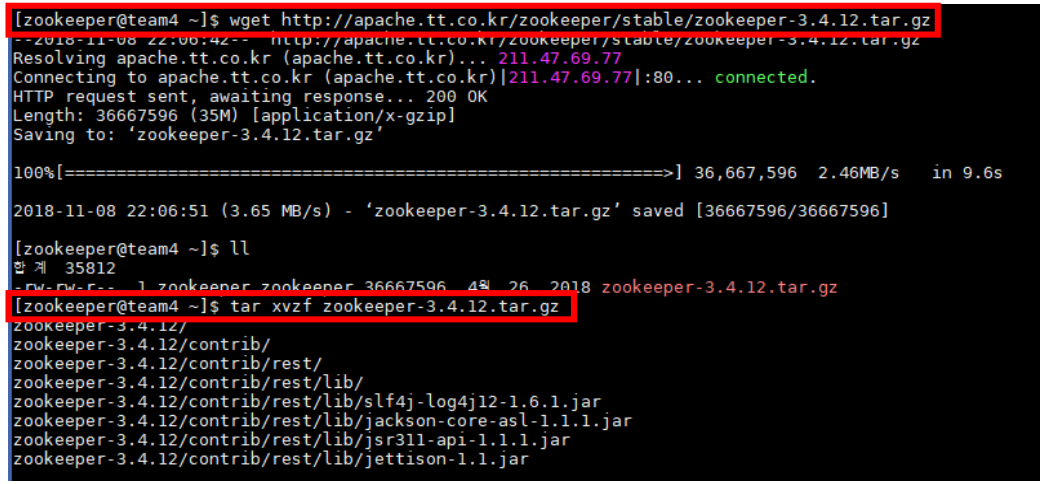
4. 서버간에 ssh-key값을 공유한다.
   $ssh-copy-id –I /home/zookeeper/.ssh/id_rsa.pub zookeeper@ip주소

5. Ssh zookeeper@ip주소를 통해 접속여부를 확인한다.

가상 분산 모드







6. 주키퍼 다운로드 및 압축을 풀어준다.

- http://apache.tt.co.kr/zookeeper/ 사이트에서 zookeeper-3.4.12.tar.gz 확인
- wget http://apache.tt.co.kr/zookeeper/stable/zookeeper-3.4.12.tar.gz
- tar xvzf zookeeper-3.4.12.gz

가상 분산 모드

```
drwxr-xr-x. 10 zookeeper zookeeper     4096  3월  27  2018 zookeeper-3.4.12
-rw-rw-r--.  1 zookeeper zookeeper 3667596  4월  26  2018 zookeeper-3.4.12.tar.gz
[zookeeper@team4 ~]$ cd zookeeper-3.4.12
[zookeeper@team4 zookeeper-3.4.12]$ ll
합계 1624
-rw-rw-r--.  1 zookeeper zookeeper   11938  3월  27  2018 LICENSE.txt
-rw-rw-r--.  1 zookeeper zookeeper    3132  3월  27  2018 NOTICE.txt
-rw-rw-r--.  1 zookeeper zookeeper    1585  3월  27  2018 README.md
-rw-rw-r--.  1 zookeeper zookeeper    1770  3월  27  2018 README_packaging.txt
drwxr-xr-x.  2 zookeeper zookeeper    4096  3월  27  2018 bin
-rw-rw-r--.  1 zookeeper zookeeper   87945  3월  27  2018 build.xml
drwxr-xr-x.  2 zookeeper zookeeper    4096  3월  27  2018 conf
drwxr-xr-x. 10 zookeeper zookeeper    4096  3월  27  2018 contrib
drwxr-xr-x.  2 zookeeper zookeeper    4096  3월  27  2018 dist-maven
drwxr-xr-x.  6 zookeeper zookeeper    4096  3월  27  2018 docs
-rw-rw-r--.  1 zookeeper zookeeper    8197  3월  27  2018 ivy.xml
-rw-rw-r--.  1 zookeeper zookeeper    1709  3월  27  2018 ivysettings.xml
drwxr-xr-x.  4 zookeeper zookeeper    4096  3월  27  2018 lib
drwxr-xr-x.  5 zookeeper zookeeper    4096  3월  27  2018 recipes
drwxr-xr-x.  8 zookeeper zookeeper    4096  3월  27  2018 src
-rw-rw-r--.  1 zookeeper zookeeper 1483366  3월  27  2018 zookeeper-3.4.12.jar
-rw-rw-r--.  1 zookeeper zookeeper     819  3월  27  2018 zookeeper-3.4.12.jar.asc
```

```
-rw-rw-r--. 1 zookeeper zookeeper  535  3월  27  2018 configuration.xsl
-rw-rw-r--. 1 zookeeper zookeeper 2161  3월  27  2018 log4j.properties
-rw-rw-r--. 1 zookeeper zookeeper  922  3월  27  2018 zoo_sample.cfg
[zookeeper@team4 conf]$ cp zoo_sample.cfg zoo.cfg
[zookeeper@team4 conf]$ ll
합계 16
-rw-rw-r--. 1 zookeeper zookeeper  535  3월  27  2018 configuration.xsl
-rw-rw-r--. 1 zookeeper zookeeper 2161  3월  27  2018 log4j.properties
-rw-rw-r--. 1 zookeeper zookeeper  922 11월   8 22:12 zoo.cfg
-rw-rw-r--. 1 zookeeper zookeeper  922  3월  27  2018 zoo_sample.cfg
[zookeeper@team4 conf]$ vi zoo.cfg
[zookeeper@team4 conf]$
```

7. 간편한 사용을 위하여 명칭을 변경하여 준다. (별도의 캡처사진은 없음)
   $ln –s zookeeper-3.4.12 zookeeper3

8. /home/zookeeper/zookeeper3/conf 의 zoo_sample.cfg를 zoo.cfg로 변경한다.
   $cp zoo_sample.cfg zoo.cfg

가상 분산 모드

```
#dataDir=/tmp/zookeeper
dataDir=/home/zookeeper/data
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
maxClientCnxns=0
maxSessionTimeout=180000
server.1=team4:2888:3888
server.2=Master:2888:3888
server.3=slave01:2888:3888
#
```

```
[zookeeper@master ~]$ ls
data  zookeeper-3.4.12  zookeeper-3.4.12.tar.gz  zookeeper.out  zookeeper3
[zookeeper@master ~]$ cd data
[zookeeper@master data]$ ls
myid  version-2  zookeeper_server.pid
[zookeeper@master data]$ vim myid
```

```
2
~
~
```

```
PATH=$PATH:$HOME/.local/bin:$HOME/bin:$ZOOKEEPER_HOME/bin:$JAVA_HOME/bin
#PATH=$PATH:$HOME/bin:$ZOOKEEPER_HOME/bin:$JAVA_HOME/bin
#PATH=$JAVA_HOME/bin:ZOOKEEPER_HOME/bin:$PATH
export PATH
export ZOOKEEPER_HOME=/home/zookeeper/zookeeper3
export JAVA_HOME=/usr/java/jdk1.7
~
```

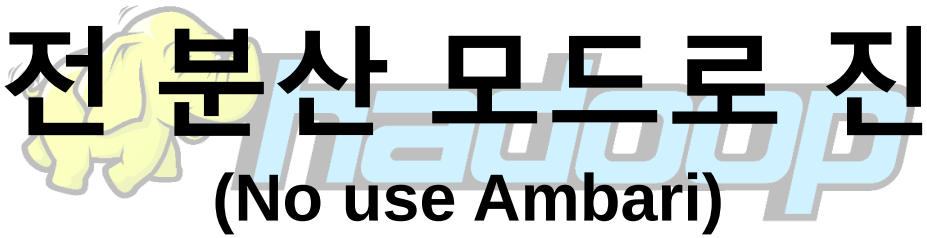
7. Vim zoo.cfg를 통하여 zoo.cfg를 설정한다.
   dataDir=/home/zookeeper/data
   maxClientCnxns=0(무한대를 의미한다)
   maxSessionTimeout=180000
   server.1=host_name:2888:3888
   server.2=host_name:2888:3888
   server.3=host_name:2888:3888
   2888 : 각각의 통신하기 위한 포트 3888 : 마스터 선출을 위한 포트

8. /home/zookeeper의 하위디렉토리에 data라는 디렉토리를 생성하고(없을 경우)
   myid 파일을 생성한다.
   $vim myid

9. 7번에서 생성한 server의 번호를 입력한다.

10. 경로를 설정한다.
    $vim ~/.bashrc

    PATH=$PATH:$HOME/.local/bin:$HOME/bin:$ZOOKEEPER_HOME/bin:$JAVA_HOME/bin
    export ZOOKEEPER_HOME=/home/zookeeper/zookeeper3
    export JAVA_HOME=/usr/java/jdk1.7



완전 분산 모드로 진행

(No use Ambari)

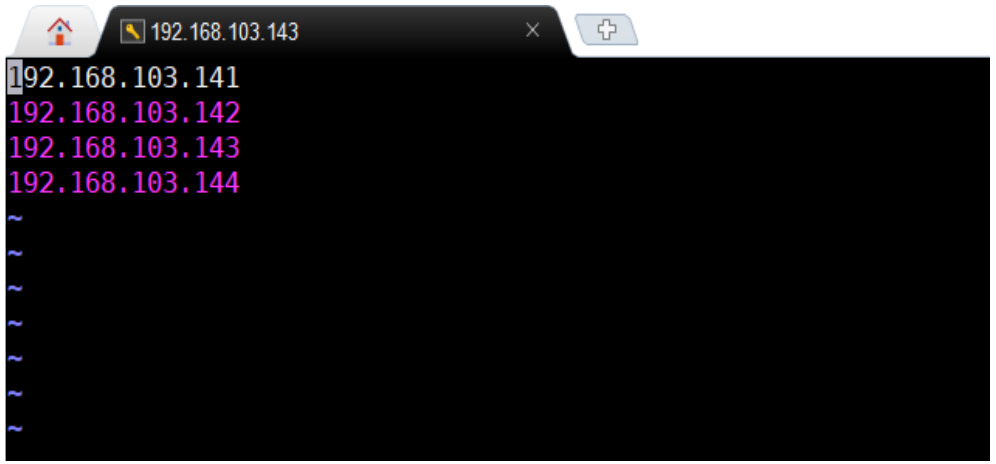
완전 분산 모드

1.가상분산모드로 설정된 가상컴퓨터에서 하둡만 삭제한다.
 - java, Protocol Buffer는 삭제하지 않는다
 - 다만, 이후 Spark, hive 설치시 버전 문제로 java를 업데이트해야 하이며 이는 해당 과정에서 설명 한다.

2. Namenode가 설정된 서버를 set up 하고 다른 서버에 배포한다.

3. 서버 설정 시 가상분산모드와 다른 설정만 제시하도록 한다.

완전 분산 모드



hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 slaves 파일 편집
파일 내 내용은 현재 cluster를 형성하는 호스트ip 주소 입력

[hadoop@localhost hadoop]$ vim slaves

완전 분산 모드

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

<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://hadoop-cluster</value>
        </property>
        <property>
                <name>ha.zookeeper.quorum</name>
                <value>jmj:2181,sdh:2181,khk:2181</value>
        </property>
        <property>
                <name>ipc.client.connect.max.retries</name>
                <value>10</value>
        </property>
        <property>
                <name>ipc.client.connect.retry.interval</name>
                <value>5000</value>
        </property>
</configuration>
                                                              36,1          바 닥
```

hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 core-site.xml 파일 편집

[hadoop@localhost hadoop]$ vim core-site.xml

* <configuration></configuration> 부분에서 안 부분을 수정해야함

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://hadoop-cluster</value>
        </property>
        <property>
                <name>ha.zookeeper.quorum</name>
                <value>localhost:2181,master:2181,slave1:2181</value>
        </property>
</configuration>
```

완전 분산 모드

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
        </property>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>/home/hadoop/data/dfs/namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>/home/hadoop/data/dfs/datanode</value>
        </property>
        <property>
                <name>dfs.journalnode.edits.dir</name>
                <value>/home/hadoop/data/dfs/journalnode</value>
        </property>
        <property>
                <name>dfs.nameservices</name>
                <value>hadoop-cluster</value>
        </property>
        <property>
                <name>dfs.ha.namenodes.hadoop-cluster</name>
                <value>nn1,nn2</value>
        </property>
        <property>
                <name>dfs.namenode.rpc-address.hadoop-cluster.nn1</name>
                <value>jmj:8020</value>
        </property>
        <property>
                <name>dfs.namenode.rpc-address.hadoop-cluster.nn2</name>
                <value>sdh:8020</value>
        </property>
        <property>
                <name>dfs.namenode.http-address.hadoop-cluster.nn1</name>
                <value>jmj:50070</value>
        </property>
        <property>
                <name>dfs.namenode.http-address.hadoop-cluster.nn2</name>
                <value>sdh:50090</value>
        </property>
```

hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 hdfs-site.xml 파일 편집(1)

[hadoop@localhost hadoop]$ vim hdfs-site.xml

* <configuration></configuration> 부분에서 안 부분을 수정해야함

다음과 같음


```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
        </property>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>/home/hadoop/data/dfs/namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>/home/hadoop/data/dfs/datanode</value>
        </property>
        <property>
                <name>dfs.journalnode.edits.dir</name>
                <value>/home/hadoop/data/dfs/journalnode</value>
        </property>
        <property>
                <name>dfs.nameservices</name>
                <value>hadoop-cluster</value>
        </property>
        <property>
                <name>dfs.ha.namenodes.hadoop-cluster</name>
                <value>nn1,nn2</value>
        </property>
        <property>
                <name>dfs.namenode.rpc-address.hadoop-cluster.nn1</name>
                <value>jmj:8020</value>
        </property>
        <property>
                <name>dfs.namenode.rpc-address.hadoop-cluster.nn2</name>
                <value>sdh:8020</value>
        </property>
        <property>
                <name>dfs.namenode.http-address.hadoop-cluster.nn1</name>
                <value>jmj:50070</value>
        </property>
        <property>
                <name>dfs.namenode.http-address.hadoop-cluster.nn2</name>
                <value>sdh:50090</value>
        </property>
```

완전 분산 모드

```
        <property>
                <name>dfs.namenode.shared.edits.dir</name>
                <value>qjournal://jmj:8485;sdh:8485;khk:8485/hadoop-cluster</value>
        </property>
         <property>
                <name>dfs.client.failover.proxy.provider.hadoop-cluster</name>
                <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
        </property>
        <property>
                <name>dfs.ha.fencing.methods</name>
                <value>sshfence</value>
        </property>
        <property>
                <name>dfs.ha.fencing.ssh.private-key-files</name>
                <value>/home/hadoop/.ssh/id_rsa</value>
        </property>
        <property>
                <name>dfs.ha.automatic-failover.enabled</name>
                <value>true</value>
        </property>
        <property>
                <name>dfs.block.size</name>
                <value>33554432</value>
        </property>
        <property>
                <name>dfs.hosts.exclude</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/exclude_list</value>
        </property>
        <property>
                <name>mapred.hosts.exclude</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/exclude_list</value>
        </property>
        <property>
                <name>dfs.hosts</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/include_list</value>
        </property>
</configuration>
```

hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기로 hdfs-site.xml 파일 편집(2)

[hadoop@localhost hadoop]$ vim hdfs-site.xml

* <configuration></configuration> 부분에서 안 부분을 수정해야함

다음과 같음

```
<property>
                <name>dfs.namenode.shared.edits.dir</name>
                <value>qjournal://jmj:8485;sdh:8485;khk:8485/hadoop-cluster</value>
        </property>
         <property>
                <name>dfs.client.failover.proxy.provider.hadoop-cluster</name>
                <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
        </property>
        <property>
                <name>dfs.ha.fencing.methods</name>
                <value>sshfence</value>
        </property>
        <property>
                <name>dfs.ha.fencing.ssh.private-key-files</name>
                <value>/home/hadoop/.ssh/id_rsa</value>
        </property>
        <property>
                <name>dfs.ha.automatic-failover.enabled</name>
                <value>true</value>
        </property>
        <property>
                <name>dfs.block.size</name>
                <value>33554432</value>
        </property>
        <property>
                <name>dfs.hosts.exclude</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/exclude_list</value>
        </property>
        <property>
                <name>mapred.hosts.exclude</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/exclude_list</value>
        </property>
        <property>
                <name>dfs.hosts</name>
                <value>/home/hadoop/hadoop2/etc/hadoop/include_list</value>
        </property>
</configuration>
```

완전 분산 모드

```
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle,spark_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
                <value>org.apache.hadoop.mapred.ShuffleHandler</value>
        </property>
        <property>
                <name>yarn.naodemanager.aux-services.spark_shuffle.class</name>
                <value>org.apache.spark.network.yarn.YarnShuffleService</value>
        </property>
        <property>
                <name>spark.yarn.shuffle.stopOnFailure</name>
                <value>false</value>
        </property>
        <property>
                <name>yarn.nodemanager.local-dirs</name>
                <value>/home/hadoop/yarn/nm-local-dir</value>
        </property>
        <property>
                <name>yarn.resourcemanager.fs.state-store.url</name>
                <value>/home/hadoop/data/yarn/system/rmstore</value>
        </property>
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>jmj</value>
        </property>
        <property>
                <name>yarn.web-proxy.address</name>
                <value>0.0.0.0:8089</value>
        </property>
        <property>
                <name>yarn.nodemanager.recovery.enabled</name>
                <value>true</value>
        </property>
        <property>
                <name>yarn.nodemanager.address</name>
                <value>0.0.0.0:45454</value>
        </property>
        <property>
                <name>yarn.nodemanager.pmem-check-enabled</name>
                <value>false</value>
        </property>
        <property>
                <name>yarn.nodemanager.vmem-check-enabled</name>
                <value>false</value>
        </property>
</configuration>
```

hadoop2/etc/hadoop/ 디렉토리에서 vim 편집기 yarn-site.xml 파일 편집

[hadoop@localhost hadoop]$ vim yarn-site.xml

<configuration></configuration> 부분 안에 수정해야함


```
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle,spark_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
                <value>org.apache.hadoop.mapred.ShuffleHandler</value>
        </property>
        <property>
                <name>yarn.naodemanager.aux-services.spark_shuffle.class</name>
                <value>org.apache.spark.network.yarn.YarnShuffleService</value>
        </property>
        <property>
                <name>spark.yarn.shuffle.stopOnFailure</name>
                <value>false</value>
        </property>
        <property>
                <name>yarn.nodemanager.local-dirs</name>
                <value>/home/hadoop/yarn/nm-local-dir</value>
        </property>
        <property>
                <name>yarn.resourcemanager.fs.state-store.url</name>
                <value>/home/hadoop/data/yarn/system/rmstore</value>
        </property>
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>jmj</value>
        </property>
        <property>
                <name>yarn.web-proxy.address</name>
                <value>0.0.0.0:8089</value>
        </property>
        <property>
                <name>yarn.nodemanager.recovery.enabled</name>
                <value>true</value>
        </property>
        <property>
                <name>yarn.nodemanager.address</name>
                <value>0.0.0.0:45454</value>
        </property>
        <property>
                <name>yarn.nodemanager.pmem-check-enabled</name>
                <value>false</value>
        </property>
        <property>
                <name>yarn.nodemanager.vmem-check-enabled</name>
                <value>false</value>
        </property>
</configuration>
```

완전 분산 모드

```
drwxrwxr-x. 2 hadoop hadoop      4096 11월   7 21:59 Downloads
drwxr-xr-x. 3 hadoop hadoop      4096 11월   7 23:31 data
drwxr-xr-x. 9 hadoop hadoop      4096 11월  14  2014 hadoop-2.6.0
-rw-rw-r--. 1 hadoop hadoop 195257604 11월   7 21:57 hadoop-2.6.0.tar.gz
lrwxrwxrwx. 1 hadoop hadoop        12 11월   9 22:16 hadoop2 -> hadoop-2.6.0
drwxr-xr-x. 3 hadoop hadoop      4096 11월   7 23:32 yarn
[hadoop@localhost ~]$ pwd
/home/hadoop
[hadoop@localhost ~]$ tar cvzf hadoop.tar.gz hadoop-2.6.0
hadoop-2.6.0/
```

```
[hadoop@localhost ~]$ scp hadoop.tar.gz hadoop@192.168.100.110:/home/hadoop
hadoop.tar.gz                                                 100%  188MB  93.6MB/
[hadoop@localhost ~]$ ll
합계 382992
drwxrwxr-x. 2 hadoop hadoop      4096 11월   7 21:59 Downloads
drwxr-xr-x. 3 hadoop hadoop      4096 11월   7 23:31 data
drwxr-xr-x. 9 hadoop hadoop      4096 11월  14  2014 hadoop-2.6.0
-rw-rw-r--. 1 hadoop hadoop 195257604 11월   7 21:57 hadoop-2.6.0.tar.gz
-rw-rw-r--. 1 hadoop hadoop 196904543 11월  10 00:15 hadoop.tar.gz
lrwxrwxrwx. 1 hadoop hadoop        12 11월   9 22:16 hadoop2 -> hadoop-2.6.0
drwxr-xr-x. 3 hadoop hadoop      4096 11월   7 23:32 yarn
[hadoop@localhost ~]$ exit
```

```
drwxrwxr-x.  2 hadoop hadoop      4096 11월   7 21:59 Downloads
drwxr-xr-x.  3 hadoop hadoop      4096 11월   7 23:31 data
drwxr-xr-x. 11 hadoop hadoop      4096 11월   7 23:21 hadoop-2.6.0
-rw-rw-r--.  1 hadoop hadoop 195257604 11월   7 21:57 hadoop-2.6.0.tar.gz
-rw-rw-r--.  1 hadoop hadoop 196904543 11월  10 00:08 hadoop.tar.gz
lrwxrwxrwx.  1 hadoop hadoop        12 11월   7 22:20 hadoop2 -> hadoop-2.6.0
drwxr-xr-x.  3 hadoop hadoop      4096 11월   7 23:32 yarn
[hadoop@slave1 ~]$ tar xvzf hadoop.tar.gz
hadoop-2.6.0/
hadoop-2.6.0/bin/
```

설정이 완료된 hadoop을 압축하여 배포.

$tar cvzf Hadoop.tar.gz Hadoop-2.6.0

$scp Hadoop.tar.gz Hadoop@ip주소(또는 hostname):/home/Hadoop

배포받은 hadoop 압축 파일을 풀고 설치.

$ tar xvzf Hadoop.tar.gz

완전 분산 모드

```
1. zookeeper가 설치된 모든 서버에서 다음 과정 진행
[zookeeper@jmj ~]$ source .bashrc
[zookeeper@jmj ~]$ zkServer.sh .start
[zookeeper@jmj ~]$ zkServer.sh .status
      Mode ghkrdls : leader(1), follower(2)
2. jmj 호스트에서 zookeeper format
[hadoop@jmj ~]$ hdfs zkfc -formatZK
3. 각각의 journalnode 실행
[hadoop@jmj ~]$ hadoop-daemon.sh start journalnode
4. jmj 호스트에서 namenode 포맷
[hadoop@jmj ~]$ hdfs namenode -format
5. jmj 호스트에서 namenode 및 zkfc 실행
[hadoop@jmj ~]$ hadoop-daemon.sh start namenode
[hadoop@jmj ~]$ hadoop-daemon.sh start zkfc
6. jmj 호스트에서 datanodes 실행
[hadoop@jmj ~]$ hadoop-daemons.sh start datanode
```

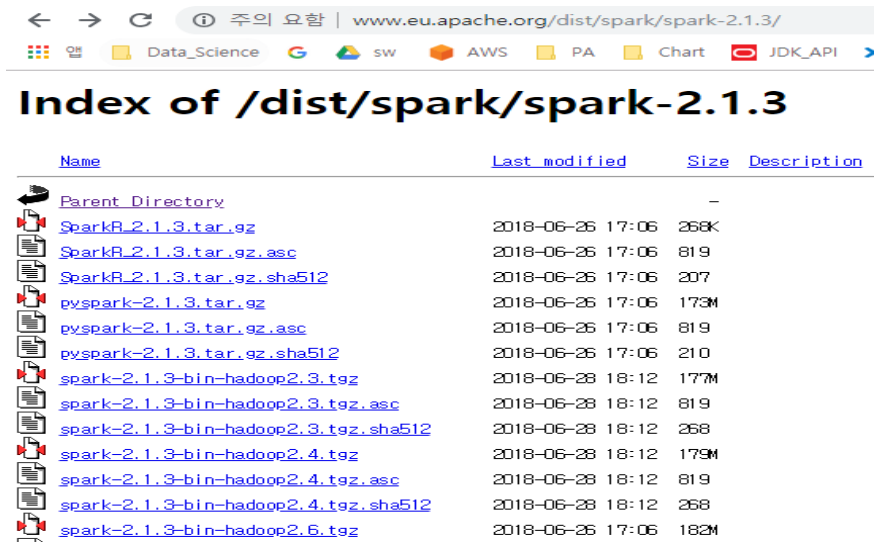
완전 분산 모드

```
7. sdh 호스트에서 standbynamenode 및 zkfe 실행
[hadoop@sdh ~]$ hdfs namenode -bootstrapStandby
[hadoop@sdh ~]$ hadoop-daemon.sh start namenode
[hadoop@sdh ~]$ hadoop-daemon.sh start zkfc
8. jmj 호스트에서 yarn-cluster 실행
[hadoop@jmj ~]$ start-yarn.sh
9. jmj 호스트에서 historyserver 실행
[hadoop@jmj ~]$ mr-jobhistory-daemon.sh start historyserver
10. jmj 호스트에서 proxyserver 실행
[hadoop@jmj ~]$ yarn-demon.sh start proxyserver
11. 하둡 분산 정상 설치 여부 확인 (wordcount실행)
[hadoop@jmj ~]$ hdfs haadmin -getServiceState nn1
[hadoop@jmj ~]$ hdfs haadmin -getServiceState nn2
[hadoop@jmj ~]$ hdfs dfs -put /home/hadoop/hadoop2/etc/hadoop/hadoop-env.sh /user/hadoop/conf
[hadoop@jmj ~]$ yarn jaf\
                /home/hadoop/hadoop2/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar\
                wordcount conf output
[hadoop@jmj ~]$ jps
```

Hadoop Eco System

(Spark | Zeppelin | Hive | Sqoop)

Hadoop Eco System



1. Spark 버젼 확인

아래 사이트에 접속하여 설치하고자 하는 spark의 버젼을 확인합니다.

http://spark.apache.org/downloads.html

2. Spark 2.1.3 설치

```
# 설치파일 다운로드

[hadoop@jmj ~]$ wget http://www.eu.apache.org/dist/spark/spark-2.1.3/spark-2.1.3-bin-hadoop2.6.tgz
```

```
# 압축 해제

[hadoop@jmj ~]$ tar -xvzf spark-2.1.3-bin-hadoop2.6.tgz

# 디렉토리명 변경

[hadoop@jmj ~]$ ln -s spark-2.1.3-bin-hadoop2.6 spark2

# 환경 변수 설정

[hadoop@jmj ~]$ vi .bashrc

_____

export YARN_CONF_DIR=/home/hadoop/etc/hadoop2
export SPARK_HOME=/home/hadoop/spark2
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
```

Hadoop Eco System

```
3. Spark on YARN

샘플 예제 실행

 * cluster 모드는 다른 node에서 driver를 실행시키는 방식이고,
   client 모드는 자신의 node에서 driver를 실행시키는 방식

client 방식 시행
[hadoop@jmj ~]$spark-submit --deploy-mode cluster \
 --class org.apache.spark.examples.SparkPi \
 /home/hadoop/spark2/examples/jars/spark-examples*.jar
 10

cluster 방식 시행
[hadoop@jmj ~]$spark-submit --deploy-mode client \
 --class org.apache.spark.examples.SparkPi \
 /home/hadoop/spark2/examples/jars/spark-examples*.jar
 10

    참고

    Spark in Standalone
    Spark는 독립적인 모드로 설치 및 수행 가능
```

```
4. slaves에 설치/배포

[hadoop@jmj ~]$ tar cvzf spark.tar.gz spark-2.1.3-bin-hadoop2.6
[hadoop@jmj ~]$ scp spark.tar.gz hadoop@sdh:/home/hadoop
[hadoop@jmj ~]$ scp spark.tar.gz hadoop@khk:/home/hadoop
[hadoop@jmj ~]$ scp spark.tar.gz hadoop@lkh:/home/hadoop

5. slaves에서 압축 해제 및 설치

[hadoop@sdh ~]$ tar xvzf spark.tar.gz
[hadoop@sdh ~]$ ln -s spark-2.1.3-bin-hadoop2.6 spark2
[hadoop@khk ~]$ tar xvzf spark.tar.gz
[hadoop@khk ~]$ ln -s spark-2.1.3-bin-hadoop2.6 spark2
[hadoop@lkh ~]$ tar xvzf spark.tar.gz
[hadoop@lkh ~]$ ln -s spark-2.1.3-bin-hadoop2.6 spark2

### 이후 과정은 master 서버에서만 진행
```

Hadoop Eco System

6. slave 서버 등록

```
[hadoop@jmj ~]$ vim spark2/conf/slaves

_____________________-
sdh
khk
lkh
```

7. 환경 설정

```
[hadoop@jmj ~]$ vim spark2/conf/spark-site.xml

_____________________-

export JAVA_HOME=/usr/java/jdl1.7
export HADOOP_CONF_DIR=/home/hadoop/hadoop2/etc/hadoop

[hadoop@jmj ~]$ vim spark2/conf/spark-de

_____________________-
spark.master                    yarn
spark.eventLog.enabled          true
spark.eventLog.dir              hdfs://jmj:9000/spark-logs
spark.history.fs.logDirectory   hdfs://jmj:9000/spark-logs
spark.history.provider          org.apache.spark.deploy.history.FsHistoryProvider
spark.history.fs.update.interval 10
spark.history.ui.port           18080
```

8. 서비스 시작/종료

```
[hadoop@jmj ~]$ start-master.sh
[hadoop@jmj ~]$ start-slaves.sh
```

9. web ui for spark

```
http://192.168.103.143:8080
```

10. 테스트

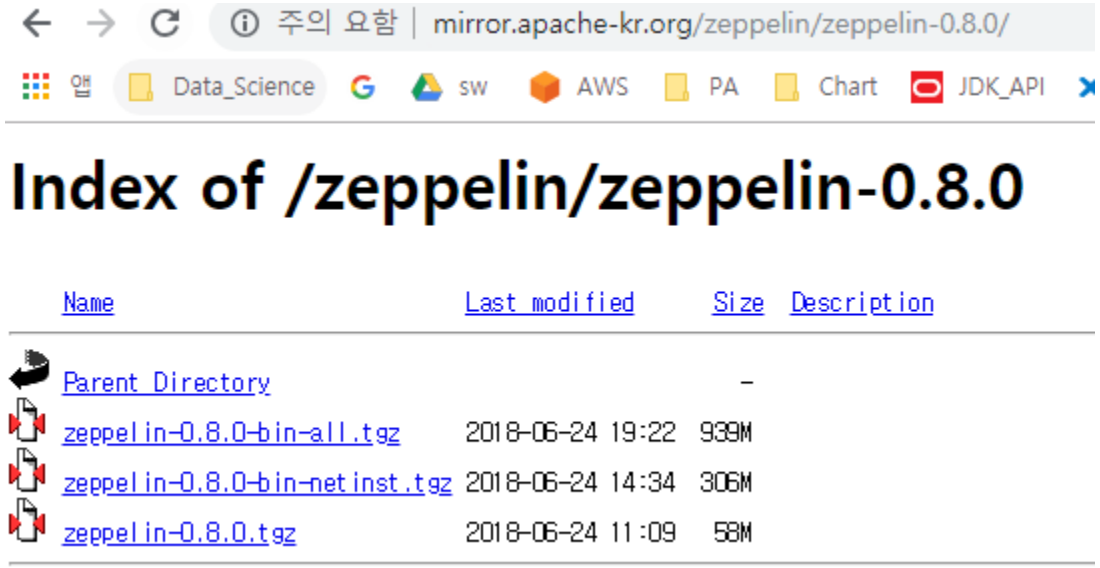
```
[hadoop@jmj ~]spark-submit \
 --master spark://jmj:7077 \
 --class org.apache.spark.examples.SparkPi \
 /home/hadoop/spark2/examples/jars/spark-examples*.jar
 100
```

11. Spark history server

```
[hadoop@jmj ~] hdfs dfs -mkdir /spark-logs
[hadoop@jmj ~] hdfs dfs -chmod /spark-logs
[hadoop@jmj ~] hdfs dfs -chown hadoop:hadoop /spark-logs
[hadoop@jmj ~] start-history-server.sh
```

12. web ui for spark history server:18080

Hadoop Eco System




```
1. Zeppelin 파일 다운 및 압축 해제

[hadoop@jmj ~]wget http://mirror.apache-kr.org/zeppelin/zeppelin-0.8.0/zeppelin-0.8.0-bin-all.tgz
[hadoop@jmj ~]tar zxfv zeppelin-0.8.0-bin-all.tgz
[hadoop@jmj ~]ln -s zeppelin-0.8.0-bin-all.tgz zeppelin
```

Hadoop Eco System

2. Zeppelin 설정

```
[hadoop@jmj ~]$ vi .bashrc
```

```
export ZEPPELIN_HOME=/home/hadoop/zeppelin
export PATH=$PATH:$ZEPPELIN_HOME/bin
```

```
[zeppelin-env.sh]
[hadoop@jmj conf]$ cp zeppelin-env.sh.template zeppelin-env.sh
```

```
export SPARK_HOME=/home/hadoop/zeppelin
export HADOOP_CONF_DIR=/home/hadoop/etc/hadoop2
export ZEPPELIN_PORT=8888
export JAVA_HOME=/usr/java/jdl1.7
export MASTER=spark://jmj:7077
```

```
[zeppelin-site.xml]
[hadoop@jmj conf]$ cp zeppelin-site.xml.template zeppelin-site.xml
```

```
#Spark Web UI와 포트가 겹치니 수정해준다.

<property>
    <name>zeppelin.server.port</name>
    <value>8888</value>
</property>
```

```
[shiro.ini]
[hadoop@jmj conf]$ cp shiro.ini.template shiro.ini
```

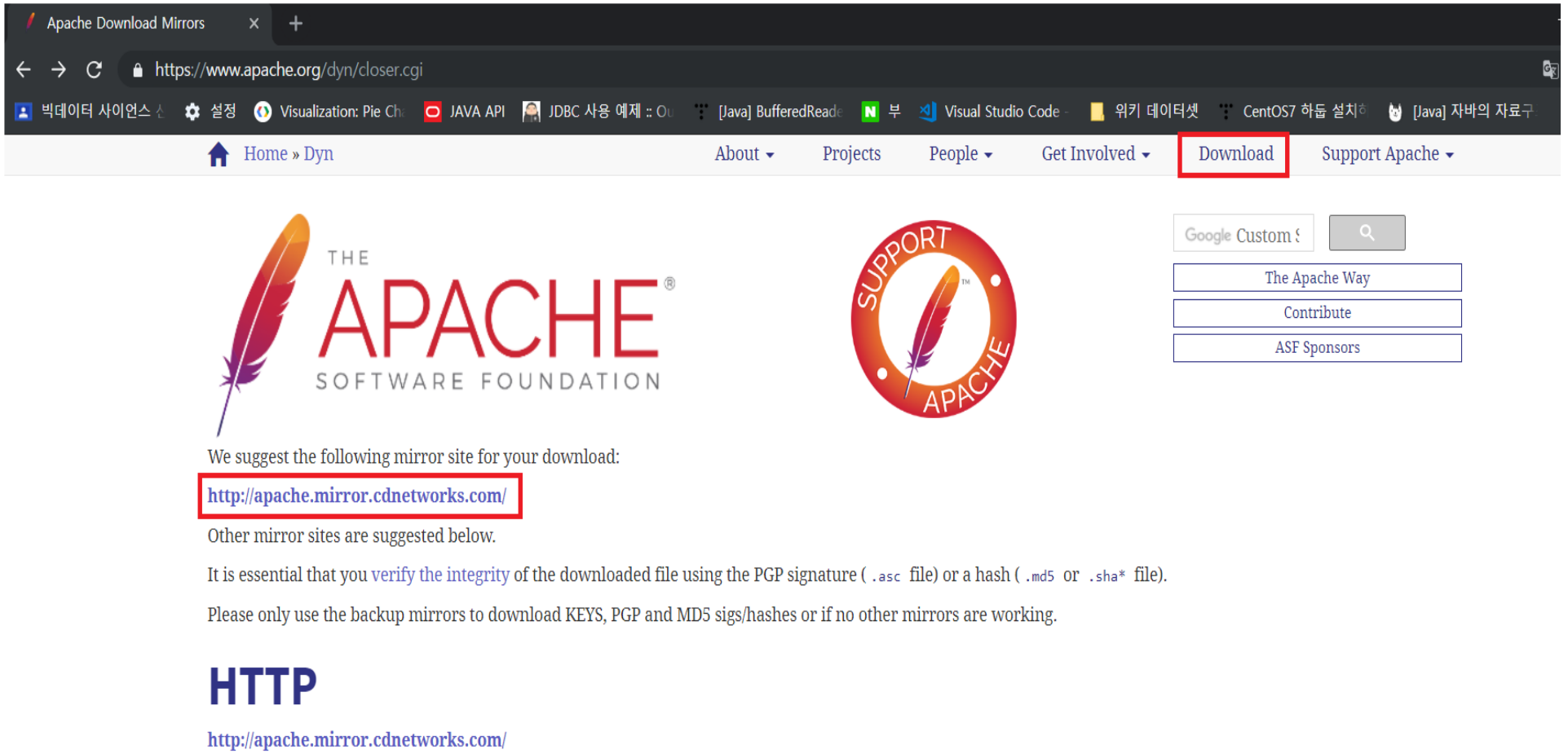
```
#사용자 정보를 수정한다.

[users]

admin = admin, admin
```

```
[interpreter.json]
[hadoop@jmj interpreter]$ vim interpreter.json
#Spark 클러스터 정보를 입력한다.
#spark.Utils.InvolkeMethod 장애 발생 시 useHiveContext 설정 값을 false로 변경한다.

"master": "spark://jmj:7077"
"zeppelin.spark.useHiveContext": "false"
```

```
3. zeppelin 실행
[hadoop@jmj interpreter]$zeppelin-daemon.sh start

4. web ui for zeppelin
http://192.168.103.143:8888
```

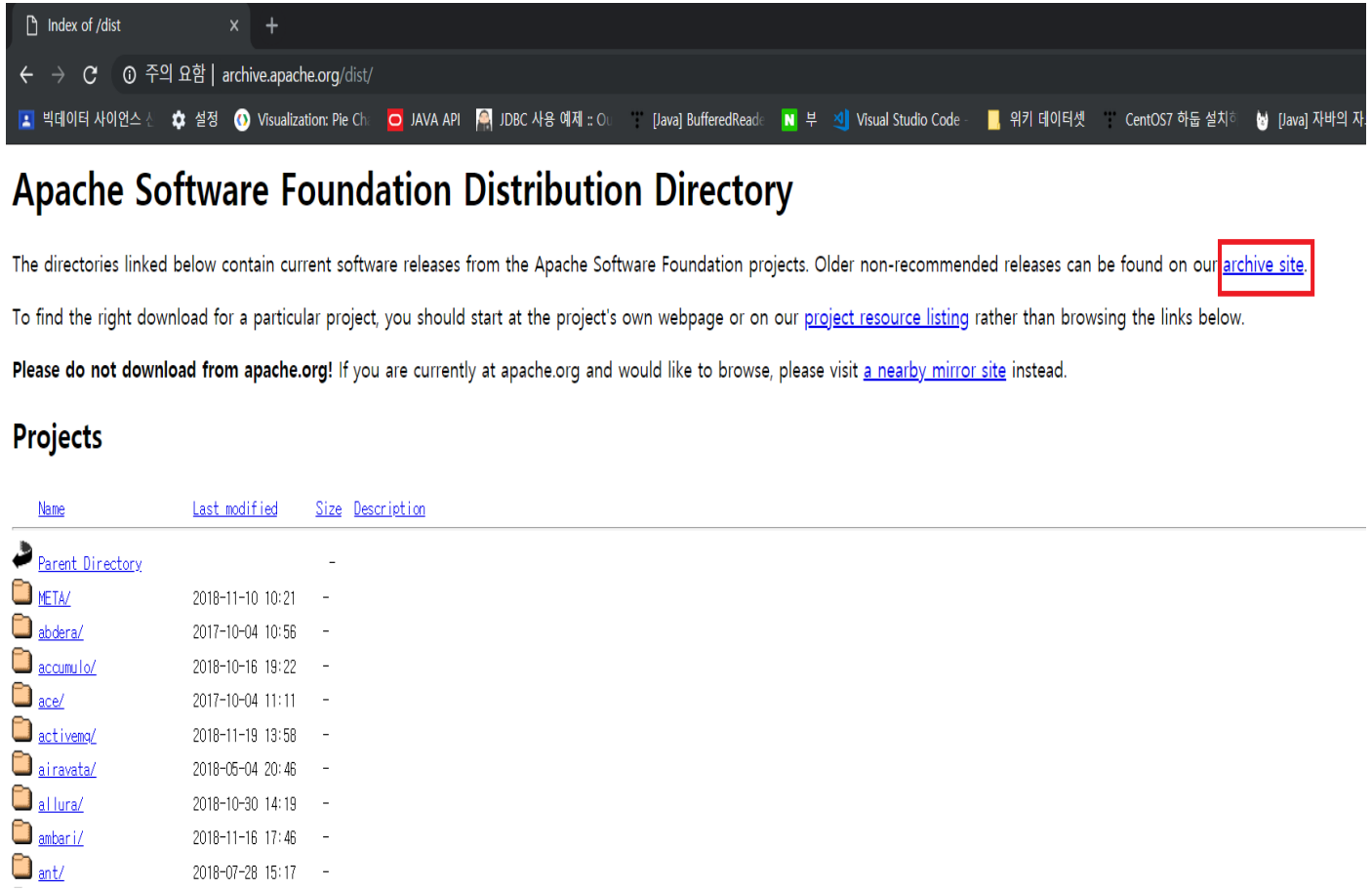
Hadoop Eco System

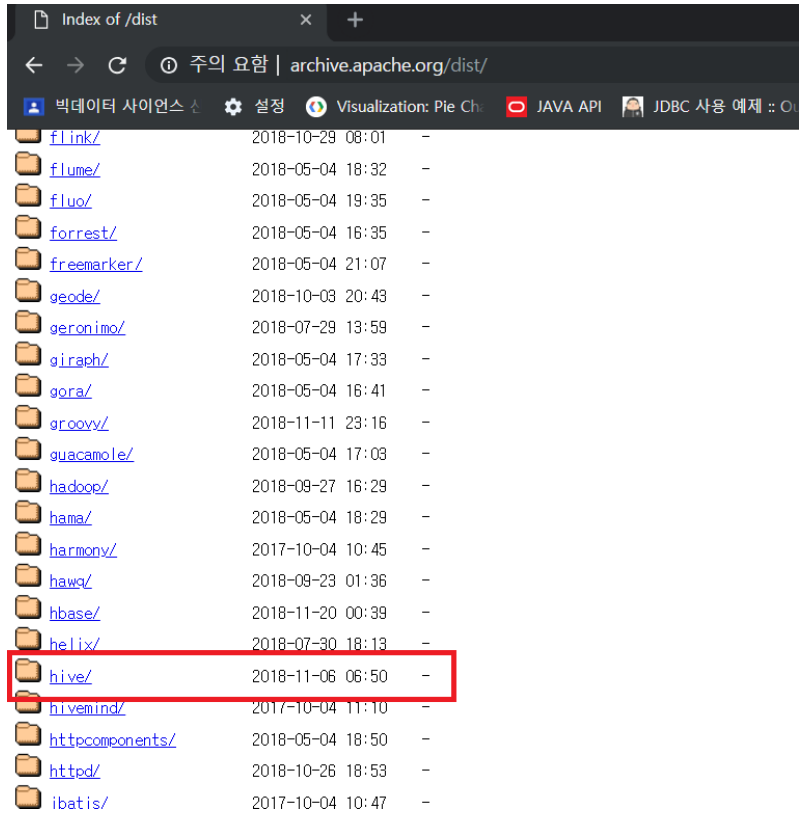
http://www.apache.org/dyn/closer.cgi/hive/ 에 접속해서 [Download] – [http://apache.mirror.cdnetworks.com/]에 접속한다.



Hadoop Eco System

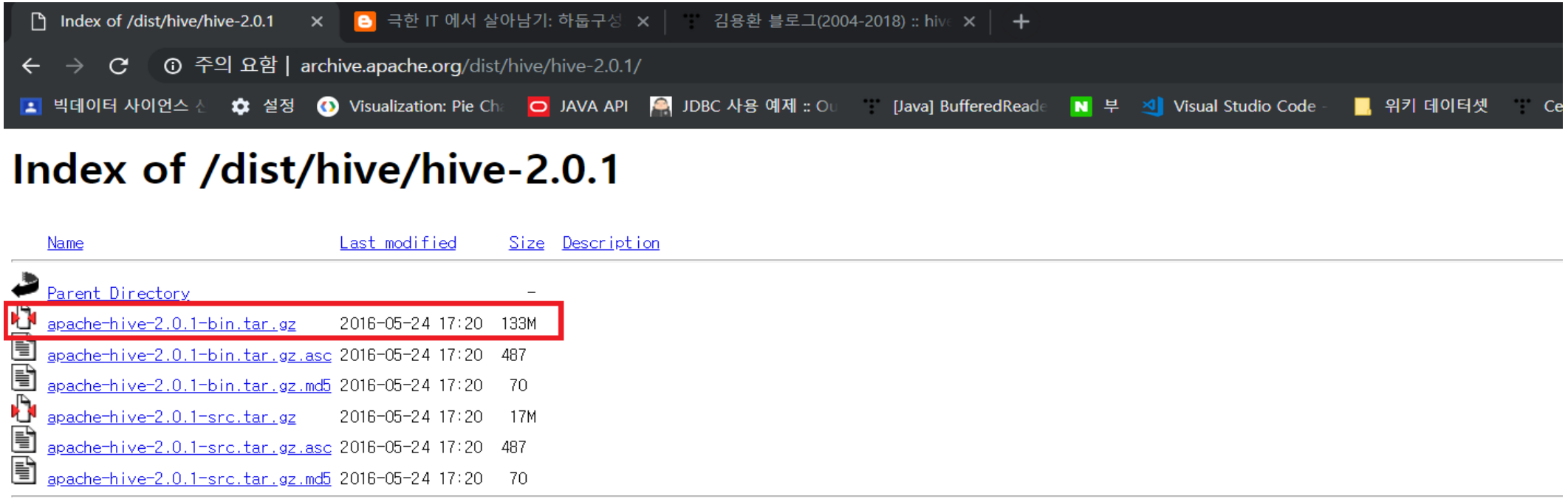
아파치 mirror 사이트에 들어가서 archive site를 클릭하면 아래와 같이 목록이 출력된다.
목록 중에 hive를 입력하고 들어간다.





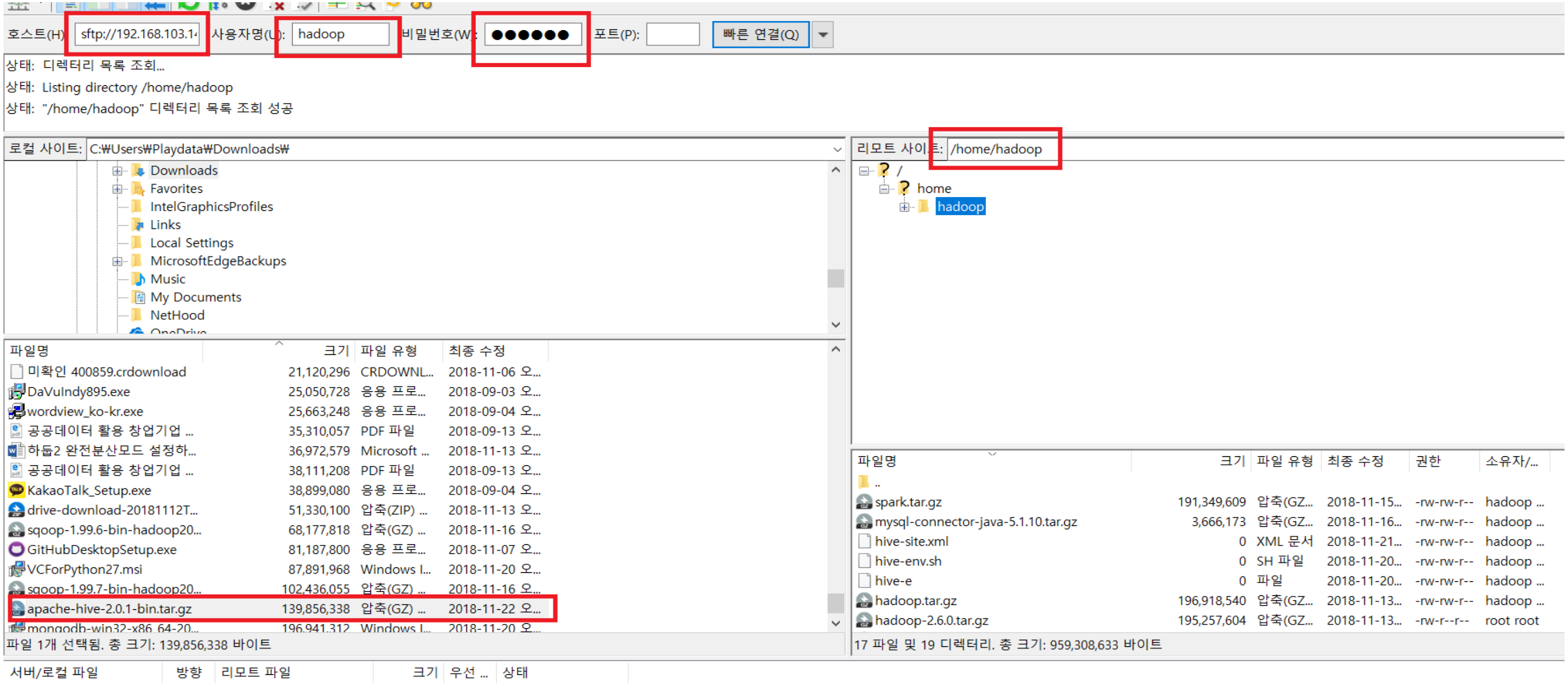
Hadoop Eco System

Hive 폴더에 들어가서 hive-2.0.1 버전을 다운로드 받는다.



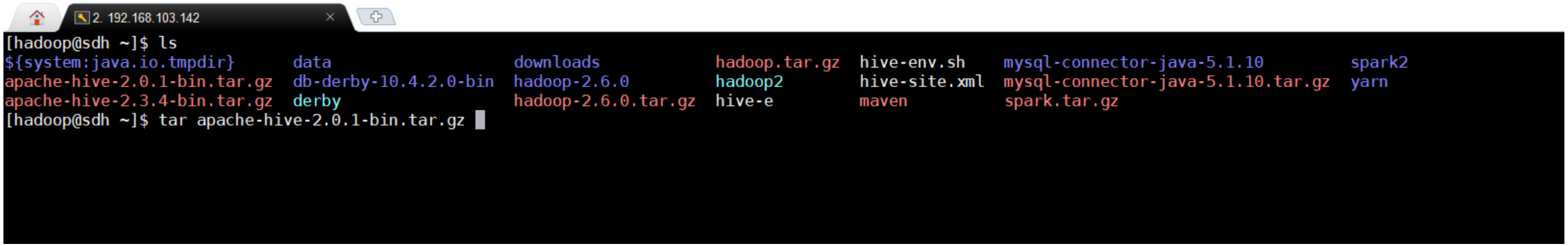
Hadoop Eco System

Filezilla로 hadoop 아이디로 접속해서 /home/hadoop 폴더로 hive 파일을 전송한다.
Wget도 가능하다.



Hadoop Eco System

apache-hive-2.0.1-bin.tar.gz
Tar xvzf apache-hive-2.0.1-bin.tar.gz로 압축푼다.



Hadoop Eco System

apache-hive-2.0.1-bin.tar.gz을 압축 푼 후
$ ln –s apache-hive-2.0.1-bin hive 로
해당 폴더를 hive 폴더로 축약한다.

```
[hadoop@sdh ~]$
[hadoop@sdh ~]$ ls
${system:java.io.tmpdir}      apache-hive-2.3.4-bin.tar.gz  derby         hadoop-2.6.0.tar.gz  hive-e         maven                                spark.tar.gz
apache-hive-2.0.1-bin         data                          downloads     hadoop.tar.gz        hive-env.sh    mysql-connector-java-5.1.10          spark2
apache-hive-2.0.1-bin.tar.gz  db-derby-10.4.2.0-bin        hadoop-2.6.0  hadoop2              hive-site.xml  mysql-connector-java-5.1.10.tar.gz   yarn
[hadoop@sdh ~]$ ln -s apache-hive-2.0.1-bin hive
[hadoop@sdh ~]$ ls
${system:java.io.tmpdir}      data                  hadoop-2.6.0        hive          maven                                spark2
apache-hive-2.0.1-bin         db-derby-10.4.2.0-bin hadoop-2.6.0.tar.gz hive-e        mysql-connector-java-5.1.10          yarn
apache-hive-2.0.1-bin.tar.gz  derby                 hadoop.tar.gz       hive-env.sh   mysql-connector-java-5.1.10.tar.gz
apache-hive-2.3.4-bin.tar.gz  downloads             hadoop2             hive-site.xml spark.tar.gz
[hadoop@sdh ~]$
```

Hadoop Eco System

$ cd ~
$ vi .bashrc로 들어가서
export HIVE_HOME=/home/hadoop/hive
export HIVE_CONF_DIR=$HIVE_HOME/conf
export PATH=$PATH:$HIVE_HOME/bin
를 입력한 후 wq로 저장하고 나온다.
그리고 $ source .bashrc 를 입력해서 환경설정 파일을 적용해준다.

```
# export SYSTEMD_PAGER=

# User specific aliases and functions

export JAVA_HOME=/usr/java/jdk1.7
#export PATH=$JAVA_HOME/bin:$PATH
export PATH=$JAVA_HOME/bin:/home/hadoop/hadoop2/bin:/home/hadoop/hadoop2/sbin:/home/hadoop/spark2/bin:/home/hadoop/spark2/sbin:$PATH

export HIVE_HOME=/home/hadoop/hive
export PATH=$PATH:$HIVE_HOME/bin:$PATH
export HIVE_CONF_DIR=$HIVE_HOME/conf
export HIVE_CLASS_PATH=$HIVE_CONF_DIR
export HADOOP_USER_CLASSPATH_FIRST=true
export HADOOP_HOME=/home/hadoop/hadoop2
export CLASSPATH=$CLASSPATH:/home/hadoop/hadoop2/lib/*:.
export CASSSPATH=$CLASSPATH:/usr/local/hive/lib/*:.

export DERBY_HOME=/home/hadoop/derby
export DERBY_INSTALL=/home/hadoop/derby
export PATH=$PATH:$DERBY_HOME/bin

export CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar
export SPARK_HOME=/home/hadoop/spark2


export ECLIPSE_HOME=/usr/java/eclipse
export PATH=$PATH:$ECLIPSE_HOME
export CLASSPATH=$JAVA_HOME/lib/tools.jar
export JAVA_HOME CLASSPATH
export SPARK_SUBMIT=/home/hadoop/spark2/bin/spark-submit
export PATH
```

Hadoop Eco System

```
#MySQL 연결 라이브러리 추가
$ wget http://repo.maven.apache.org/maven2/mysql/mysql-connector-java/5.1.9/mysql-connector-java-5.1.9.jar

#MySQL 연결 라이브러리 복사
$ mv mysql-connector-java-5.1.9.jar ./lib

#hive-site.xml 파일 수정
$ vi conf/hive-site.xml
```

```
<?xml version="1.0"?><?xml-stylesheet type="text/xsl" href="configuration.xsl"?><configuration>
<property>
   <name>hive.metastore.local</name>
   <value>true</value>
</property>
<property>
   <name>javax.jdo.option.ConnectionURL</name>
   <value>jdbc:mysql://localhost:3306/metastore?createDatabaseIfNotExist=true&amp;useUnicode=true&amp;characterEncoding=UTF-8</value>
</property>
<property>
   <name>javax.jdo.option.ConnectionDriverName</name>
   <value>com.mysql.jdbc.Driver</value>
</property>
<property>
   <name>javax.jdo.option.ConnectionUserName</name>
   <value>username</value>
</property>
<property>
   <name>javax.jdo.option.ConnectionPassword</name>
   <value>password</value>
</property>
<property>
   <name>hive.metastore.uris</name>
   <value>thrift://localhost:10000</value>
</property>
</configuration>
```

Hadoop Eco System

```
$ cp conf/hive-log4j.properties.template conf/hive-log4j.properties
$ vi conf/hive-log4j.properties

아래 변수를 찾아 다르면 아래 항목으로 수정
log4j.appender.EventCounter=org.apache.hadoop.log.metrics.EventCounter

하둡 디렉토리 구성: hive.metastore.warehove.dir 설정(./conf/hive-default.xml) 가능.
$ hadoop fs -mkdir /tmp
$ hadoop fs -mkdir /user/hive/warehouse
$ hadoop fs -chmod g+w /tmp
$ hadoop fs -chmod g+w /user/hive/warehouse3.

Hive 실행

서버로 실행
$ hive --service hiveserver &
$ hive --service metastore &

실행
$ hive
```

Hadoop Eco System

```
-- sqoop을 설치할 디렉토리 생성

[root@host명 home]# mkdir sqoop
    sqoop 설치할 디렉토리 생성

[root@host명 home]# cd sqoop
    sqoop 디렉토리로 이동
```

```
-- sqoop을 설치 및 변경

http://apache.mirror.cdnetworks.com/sqoop/1.4.7/sqoop-1.4.7.bin_hadoop-2.6.0.tar.gz
    apache.mirror에서 sqoop-1.4.7.bin_hadoop-2.6.0.tar.gz을 다운로드 받은 후 filezilla를 통해 파일 옮기기

[root@host명 sqoop]# tar xvfz sqoop-1.4.7.bin_hadoop-2.6.0.tar.gz
    파일 압축 해제

[root@host명 sqoop]# ln -s sqoop-1.4.7.bin_hadoop-2.6.0 sqoop2
    파일 이름 변경
```

Hadoop Eco System

```
-- sqoop 경로 설정

[root@host명 sqoop]# vim /home/hadoopo/.bashrc
    export PATH=$JAVA_HOME/bin:/home/hadoop/hadoop2/bin:/home/hadoop/hadoop2/sbin:
                /home/hadoop/spark2/bin:/home/hadoop/spark2/sbin:/home/sqoop/sqoop/bin:$PATH
    export SQOOP_HOME=/home/sqoop/sqoop

    --> sqoop 설치경로를 hadoop에 경로 설정


[root@host명 ~]# source /home/hadoop/.bashrc
    수정된 사항 적용
```

```
-- sqoop 경로 설정

[root@host명 sqoop2]# vim sqoop-env-template.sh
    #Set path to where bin/hadoop is available
    export HADOOP_COMMON_HOME=/home/hadoop/hadoop2

    #Set path to where hadoop-*-core.jar is available
    export HADOOP_MAPRED_HOME=/home/hadoop/hadoop2

    --> hadoop 경로 설정

[root@host명 sqoop2]# source sqoop-env-template.sh
    수정된 사항 적용
```

Hadoop Eco System

```
-- sqoop 실행  오류

[root@host명 ~]# sqoop list-tables --connect jdbc:mysql://localhost:3306/test
    --username 'mysql계정명' --password 'mysql계정의 비밀번호'

아래 오류의 해결 방법은 /home/sqoop/sqoop/lib에 mysql 드라이버를 설치하면 해결됨
    ERROR sqoop.Sqoop: Got exception running Sqoop: java.lang.RuntimeException:
    Could not load db driver class: com.mysql.jdbc.Driver
```

```
-- sqoop 실행 오류 해결법

https://dev.mysql.com/downloads/file/?id=481158
    mysql 디렉토리를 다운로드하고 filezilla를 이용해 lib 디렉토리 위치에 설치

[root@host명 ~]# cd /home/sqoop/sqoop2/lib
    디렉토리

[root@host명 lib]# tar xvfz mysql-connector-nodejs-8.0.13.tar.gz
    압축 해제

[root@host명 ~]# vim /home/hadoop/.bashrc
    CLASSPATH=.:$JAVA_HOME/lib/tools.jar:$JAVA_HOME/home/sqoop/sqoop2/lib/package
    export CLASSPATH
    경로 설정

[root@host명 ~]# source /home/hadoop/.bashrc
    수정된 사항 적용
```

완전 분산 모드로 진행

(Use Ambari)

완전 분산 모드

본 장은 하둡 관리 소프트웨어인 Ambari를 이용한 내용을 기술합니다.

CentOS 7을 설치 및 네트워크 설정 과정은 생략하고 난 후의

설치 및 설정 과정을 기술합니다.

완전 분산 모드

Host명 변경

```
[root@localhost ~]# hostnamectl set-hostname [변경할 host명]
[root@localhost ~]# reboot
```

Fully Qualified Domain Name 설정

```
[root@host명 ~]# vi /etc/hosts
```

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

# FQDN
192.168.103.151 namenode.hadoop.com namenode
192.168.103.152 datanode01.hadoop.com datanode01
192.168.103.153 datanode02.hadoop.com datanode02
192.168.103.154 datanode03.hadoop.com datanode03
192.168.103.155 datanode04.hadoop.com datanode04
```

완전 분산 모드

방화벽 설정 및 해제

```
[root@host명 ~]# yum install firwalld
[root@host명 ~]# systemctl stop firewalld
[root@host명 ~]# systemctl disable firewalld
```

SSH Key 생성 및 SSH 접속 설정

```
[root@host명 ~]# ssh-keygen
[root@host명 ~]# cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
[root@host명 ~]# cat ~/.ssh/id_rsa.pub | ssh root@namenode.hadoop.com 'cat >> ~/.ssh/authorized_keys'
[root@host명 ~]# cat ~/.ssh/id_rsa.pub | ssh root@datanode01.hadoop.com 'cat >> ~/.ssh/authorized_keys'
[root@host명 ~]# cat ~/.ssh/id_rsa.pub | ssh root@datanode02.hadoop.com 'cat >> ~/.ssh/authorized_keys'
[root@host명 ~]# cat ~/.ssh/id_rsa.pub | ssh root@datanode03.hadoop.com 'cat >> ~/.ssh/authorized_keys'
[root@host명 ~]# cat ~/.ssh/id_rsa.pub | ssh root@datanode04.hadoop.com 'cat >> ~/.ssh/authorized_keys'
```

완전 분산 모드

Hugepage 편집

```
[root@host명 ~]# vi /etc/rc.local
```

```
if test -f /sys/kernel/mm/transparent_hugepage/enabled; then
/bin/echo never > /sys/kernel/mm/transparent_hugepage/enabled
fi
if test -f /sys/kernel/mm/transparent_hugepage/defrag; then
/bin/echo never > /sys/kernel/mm/transparent_hugepage/defrag
fi
```

SELinux 비활성화 설정

```
[root@host명 ~]# vi /etc/selinux/config
```

```
SELINUX=enforcing  -->  SELINUX=disabled
```

완전 분산 모드

Swap 활용도 설정

```
[root@host명 ~]# vi /etc/sysctl.conf
```

```
vm.swappiness=10  <-- 추가
```

Ambari repository 다운로드

```
wget -nv http://public-repo-1.hortonworks.com/ambari/centos7/
2.x/updates/2.6.1.0/ambari.repo -O /etc/yum.repos.d/ambari.repo
```

Ambari server 다운로드

```
yum install ambari-server
```

완전 분산 모드

MySQL 연동

```
[root@host명 ~]# yum install mysql-connector-java
[root@host명 ~]# ambari-server setup --jdbc-db=mysql --jdbc-dirver=/usr/share/java/mysql-connector-java.jar
```

Ambari server 설정

```
[root@host명 ~]# ambari-server setup
```

Ambari agent 다운로드

```
[root@host명 ~]# yum install ambari-agent
```

완전 분산 모드

ambari agent 설정

```
[root@host명 ~]# vi /etc/ambari-agent/ambari-agent.ini
```

```
hostname=localhost  -->  hostname=master의 host명

[security] 부분에 아래 내용 추가
force_https_protocol=PROTOCOL_TLSv1_2
```

```
[root@host명 ~]# vi /etc/python/cert-verification.cfg
```

```
[https]
verify=platform_default  -->  verify=disable
```

완전 분산 모드

Host 간 NTP 시간 동기화

```
[root@host명 ~]# yum remove chrony
[root@host명 ~]# yum install ntp
[root@host명 ~]# systemctl stop ntpd
```

NTP 설정

```
[root@host명 ~]# vi /etc/ntp.conf
```

```
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst

주석 후 아래 내용 추가

server 'namenode의 ip주소' iburst
```

완전 분산 모드

Windows hosts 파일 설정

```
메모장을 관리자 권한으로 실행한 후 C:\Windows\System32\drivers\etc\에
위치한 hosts 파일의 내용 수정

ambari-server에 할당한 ip와 host명 등록
ex) xxx.xxx.xxx.xxx master.hadoop.com
```

Ambari server 실행

```
[root@host명 ~]# ambari-server start
```

Ambari agent 실행

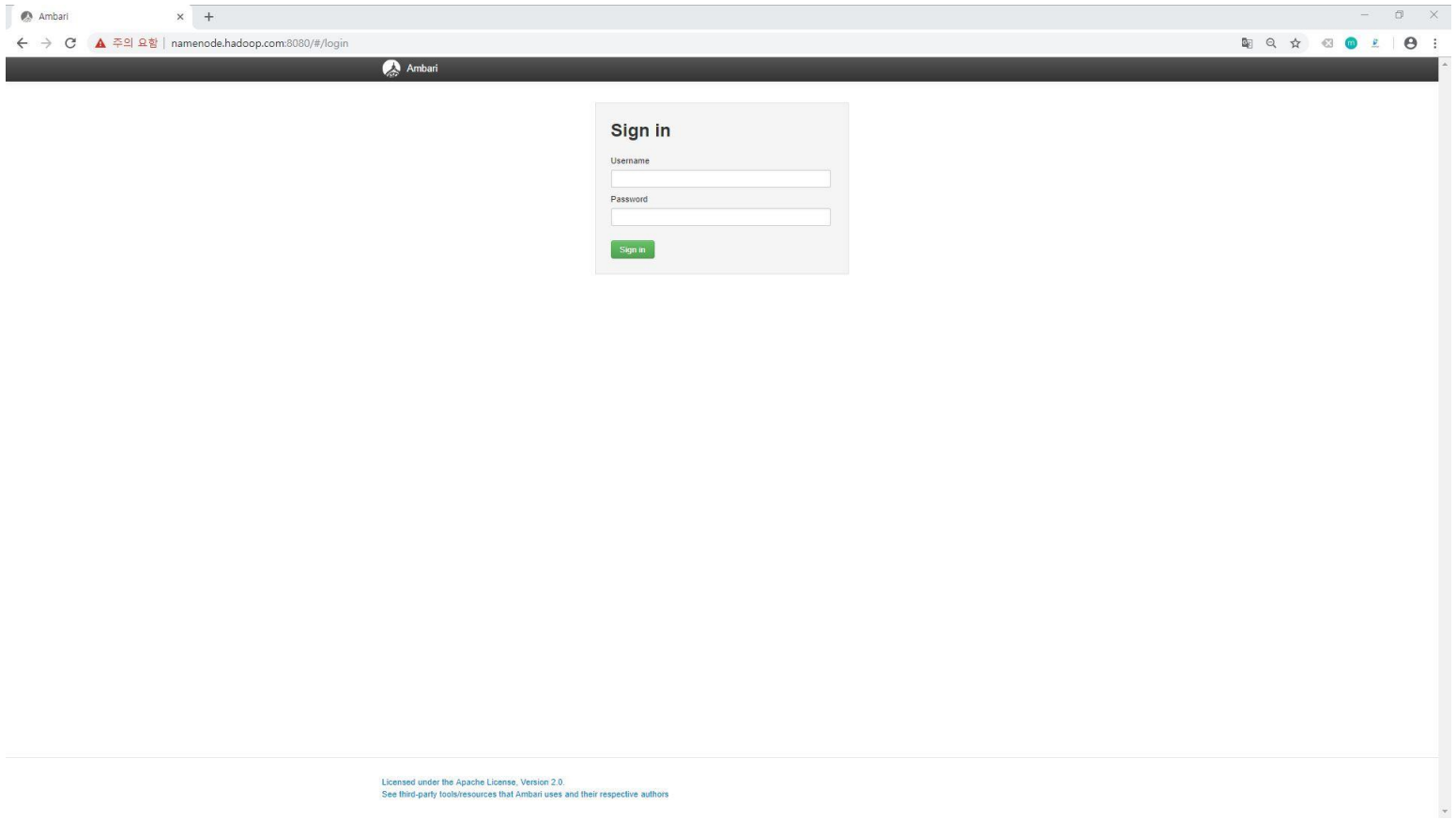
```
[root@host명 ~]# ambari-agent start
```

완전 분산 모드

Web browser에서 Ambari 실행 및 접속

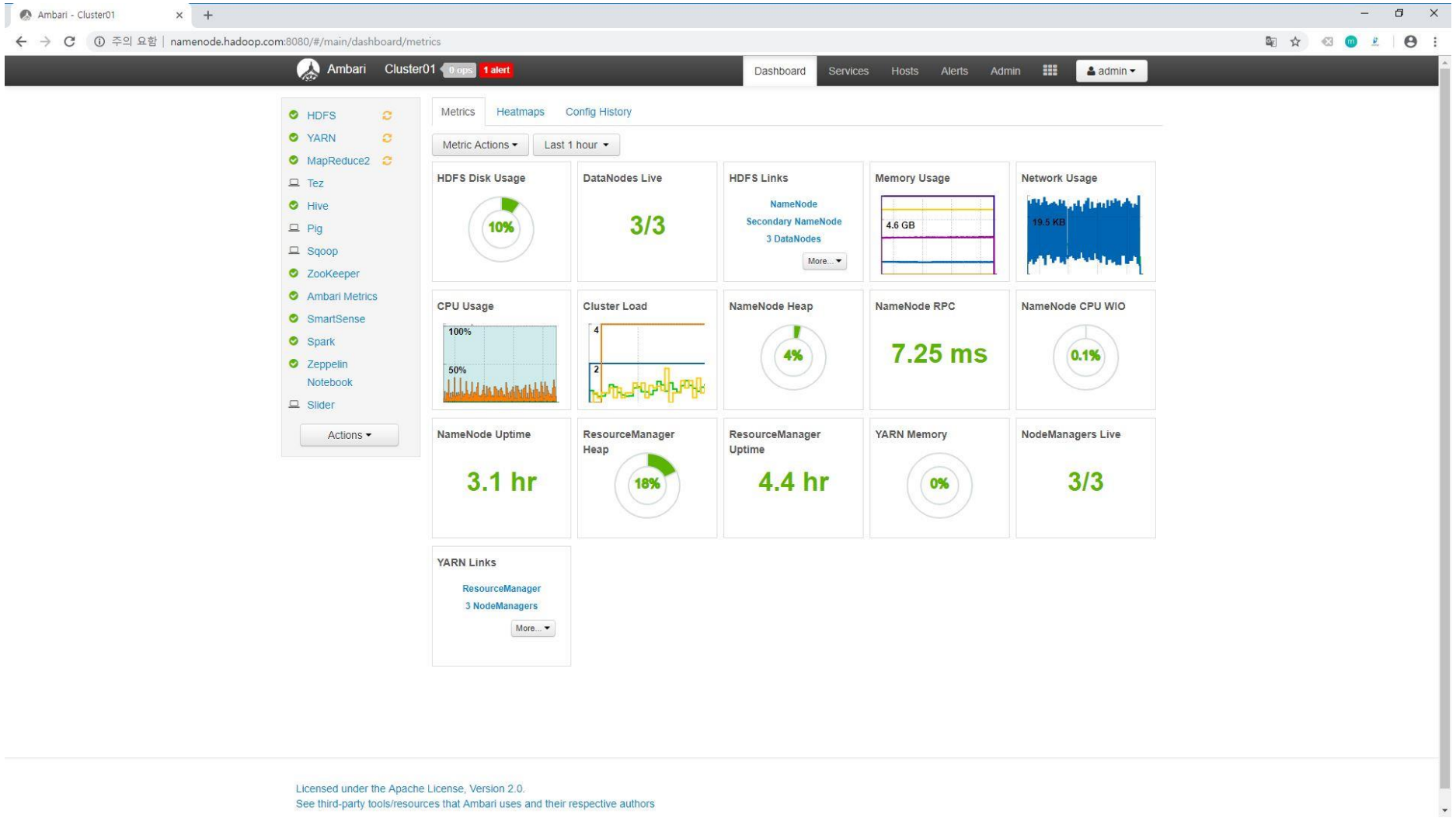
```
1 http://'master의 host명 또는 mater의 ip':8080
```

```
1 username : admin, password : admin 입력 후 login
```



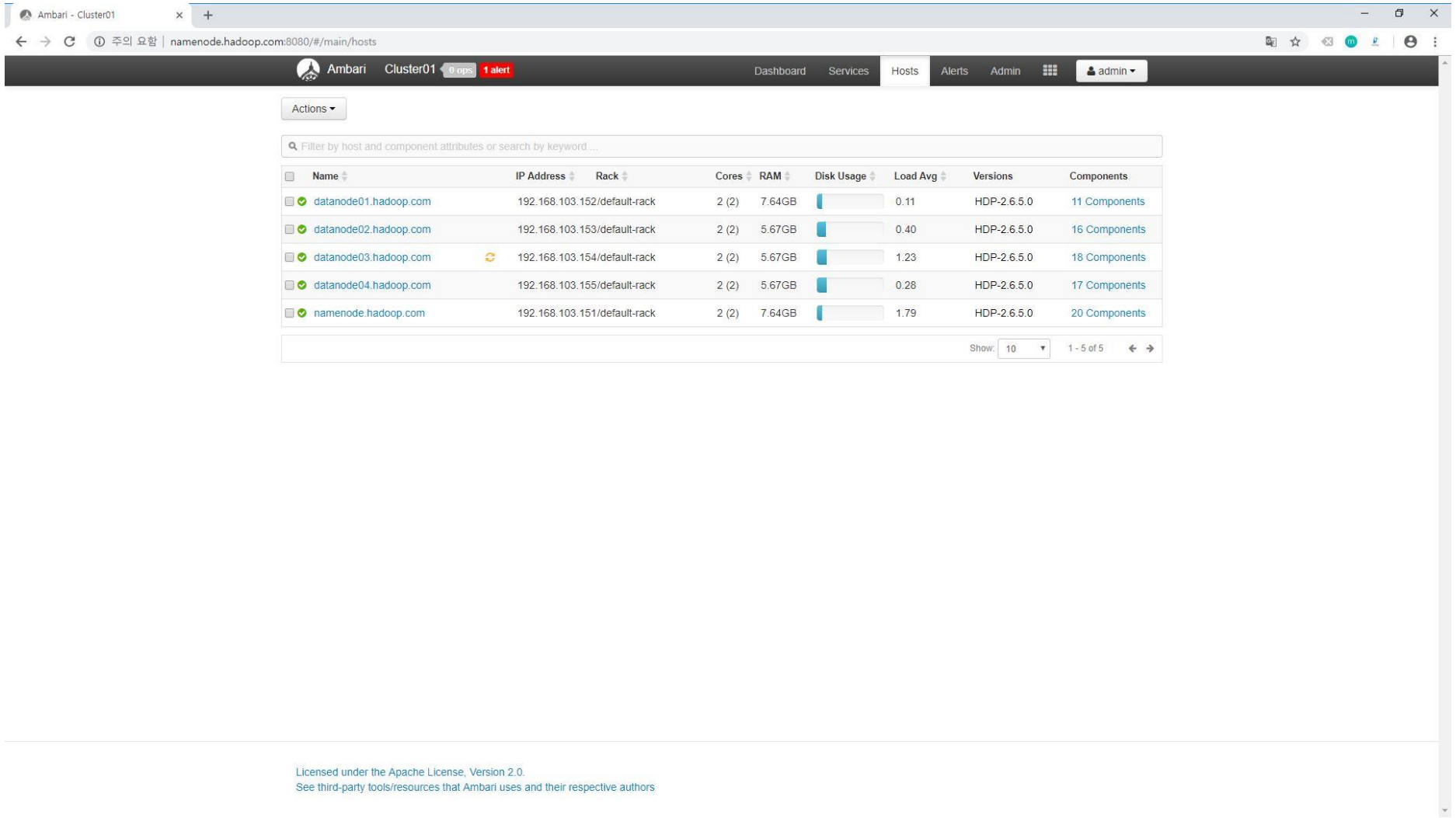
완전 분산 모드

Ambari 메인 화면



완전 분산 모드

Cluster에 등록된 Hosts 목록 화면



완전 분산 모드 (Host에 MySQL 및 Hive 설치)

mysql-connector-java 권한 변경 (namenode host에서 진행)

```
[root@host명 ~]# chmod 644 /usr/share/java/mysql-connector-java.jar
```

mysql repo 다운로드 (hive 서비스를 설치할 node에서 진행)

```
[root@Host명 ~]# wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
[root@Host명 ~]# sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm
```

mysql server 다운로드 (hive 서비스를 설치할 node에서 진행)

```
[root@Host명 ~]# yum install mysql-server
```

완전 분산 모드 (Host에 MySQL 및 Hive 설치)

mysql 시작 (hive 서비스를 설치할 node에서 진행)

```
[root@Host명 ~]# sudo systemctl start mysqld
```

mysql 사용 (hive 서비스를 설치할 node에서 진행)

```
[root@Host명 ~]# mysql -u root -p
    root 계정의 처음 비밀번호는 없으므로 비밀번호는 입력하는 부분이 나왔을 때 그냥 'enter key' 입력
```

완전 분산 모드 (Host에 MySQL 및 Hive 설치)

root 계정으로 mysql을 접속한 후 아래 내용 작성 후 exit로 종료 (새로운 계정 생성 및 사용을 위한 설정)

```
mysql> CREATE USER 'mysql에서 사용할 계정명'@'%' IDENTIFIED BY 'mysql에서 사용할 계정명의 비밀번호';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'mysql에서 사용할 계정명'@'%';
mysql> CREATE USER 'mysql에서 사용할 계정명'@'localhost' IDENTIFIED BY 'mysql에서 사용할 계정명의 비밀번호';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'mysql에서 사용할 계정명'@'localhost';
mysql> CREATE USER 'mysql에서 사용할 계정명'@'hive 서비스를 설치할 node host명' IDENTIFIED BY 'mysql에서 사용할 계정명의 비밀번호';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'mysql에서 사용할 계정명'@'hive 서비스를 설치할 node의 host명';
mysql> FLUSH PRIVILEGES;
mysql> exit;
```

```
ex)
mysql> CREATE USER 'ambari'@'%' IDENTIFIED BY 'ambari';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'ambari'@'%';
mysql> CREATE USER 'ambari'@'localhost' IDENTIFIED BY 'ambari';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'ambari'@'localhost';
mysql> CREATE USER 'ambari'@'datanode03.hadoop.com' IDENTIFIED BY 'ambari';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'ambari'@'datanode03.hadoop.com';
mysql> FLUSH PRIVILEGES;
mysql> exit;
```

완전 분산 모드 (Host에 MySQL 및 Hive 설치)

새로 생성한 계정으로 mysql 접속

```
[root@host명 ~]# mysql -u 새로운 계정명 -p
    ex) mysql -u ambari -p
```

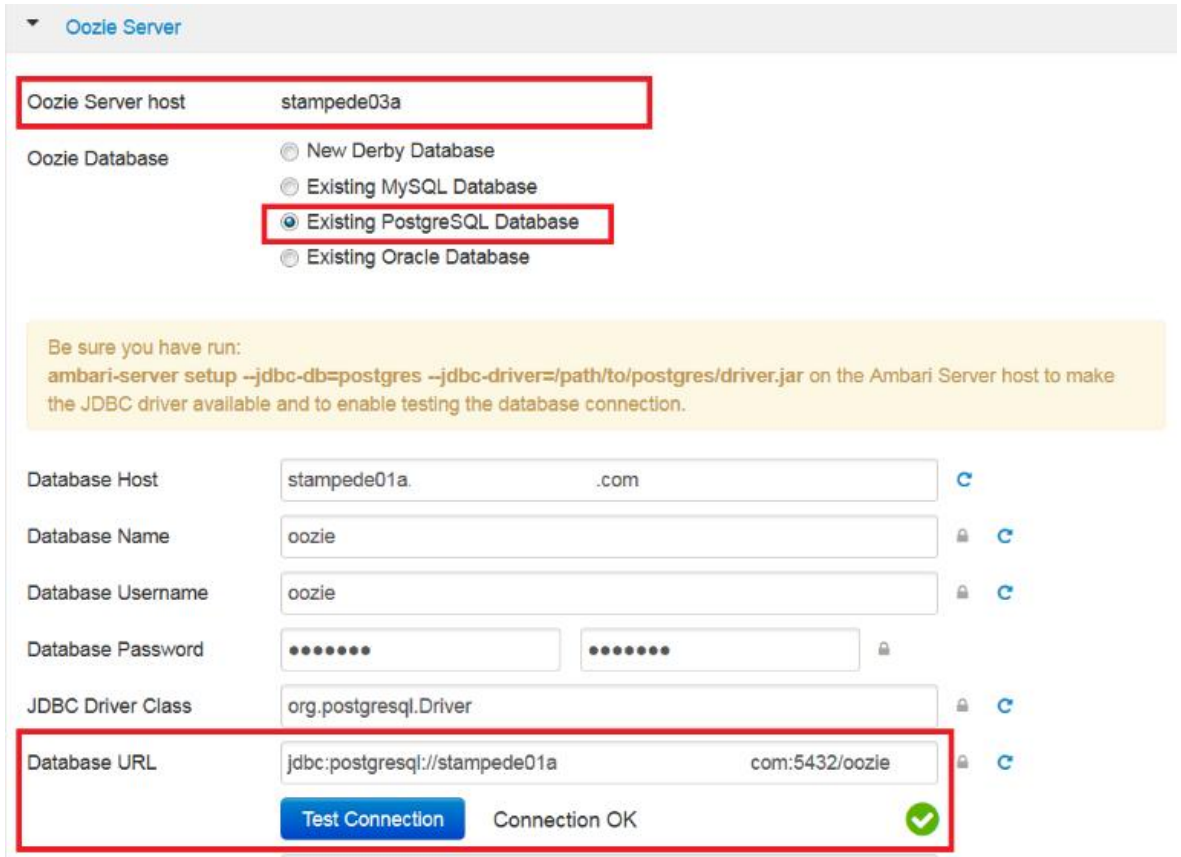
다음 mysql 설정 진행

```
mysql> CREATE DATABASE 새로 생성할 DB명;
mysql> USE 새로 생성한 DB명;
mysql> SOURCE Ambari-DDL-MySQL-CREATE.sql;
```

```
ex)
mysql> CREATE DATABASE ambari;
mysql> USE ambari;
mysql> SOURCE Ambari-DDL-MySQL-CREATE.sql;
    별도로 첨부한 Ambari-DDL-MySQL-CREATE.sql 파일을 편집기로 실행한 뒤
    전체 내용을 복사 붙여넣기 한 후 종료
```

완전 분산 모드 (Host에 MySQL 및 Hive 설치)

Ambari 관리 Page에서 서비스 추가




```
Oozie Database
    사용할 DB 선택
        MySQL을 미리 설치했을 경우, Existing MySQL Database 선택

[Database Host] : Hive를 설치할 host
[Database Name] : 사용할 db 명
[Database Username] : 사용할 mysql의 사용자 명
[Database Password] : 사용할 mysql의 사용자 password

위 사항을 입력한  Test Connection 실행  --> Connection OK가 뜨면 정상 실행
화면 하단에 위치한 next 버튼 실행
```

완전 분산 모드 (Host에 MySQL 및 Hive 설치)

Ambari 관리 Page에서 Hive View를 실행하기 위한 설정 (namenode host에서 진행)

```
[root@host명 ~]# su hdfs
[hdfs@host명 root]$ hdfs dfs -chown admin:hadoop /user/${username}
    ${username} : ambari 관리 페이지에 접속해 사용하는 사용자 명
        ex) hdfs dfs -chown admin:hadoop /user/admin

reference url
https://docs.hortonworks.com/HDPDocuments/Ambari-2.6.2.0/bk_ambari-views/content/setup_HDFS_user_directory.html
```

Hive 실행

```
[root@host명 ~]# su hdfs
[hdfs@host명 root]$ hive
    hive 실행 명령 동작 시 시간 다소 소요

hive>
```

완전 분산 모드 (Host에 MySQL 및 Hive 설치)

Ambari 관리자 계정인 admin의 정보가 저장되어 있는 위치 확인

```
[root@host명 ~]# cd /lib/ambari-server/web/data/users

[root@host명 users]# ls
    privileges.json  privileges_admin.json  user_admin.json  user_user.json  users.json

[root@host명 users]# cat privileges_admin.json
```

완전 분산 모드 (Host에 MySQL 및 Hive 설치)

Ambari 관리자 계정인 admin의 정보가 저장되어 있는 위치 확인

hdfs@namenode:/lib/ambari-server/web/data/users

```
[root@namenode users]# cat privileges_admin.json
{
  "href" : "http://c6401.ambari.apache.org:8080/api/v1/privileges?PrivilegeInfo/principal_name=admin&fields=PrivilegeInfo/*",
  "items" : [
    {
      "href" : "http://c6401.ambari.apache.org:8080/api/v1/privileges/1",
      "PrivilegeInfo" : {
        "permission_name" : "AMBARI.ADMINISTRATOR",
        "principal_name" : "admin",
        "principal_type" : "USER",
        "privilege_id" : 1
      }
    }
  ]
}
[root@namenode users]#
```

Thank you very much for your attention