■ 하이브(Apache Hive)

1) 하둡 이전의 (빅)데이터 분석업무를 맡은 사람들은 SQL등을 활용하여 업무를 수행했음

2) 하지만 하둡의 맵리듀스는 애플리케이션을 구현해야 처리 / 분석을 할 수 있었음

3) 그래서 기존의 데이터분석가를 위해 '페이스북'이 하이브를 개발

4) 현재는 하둡플랫폼으로 빅데이터를 처리하는 회사에서 다양하게 사용하였음

- 페이스북, 넷플릭스 등

■ 하이브(Apache Hive)의 개념

1) 하둡 데이터(파일)를 SQL과 비슷한 쿼리를 이용해서 다룰 수 있게 해줌

- 하이브QL 지원

2) DW(Data Warehouse) 어플리케이션에 적합

- 하둡의 기반으로 대량의 데이터를 배치 연산 가능
- 레코드 단위 갱신/삭제
- 트랜잭션 제한적인 지원(0.13 버전 이전에는 아예 지원을 안했음)
- HiveQL로 분석, 데이터 작업은 용이하나 내부적으로는 Mapreduce로 변환되어 진행되기 때문에 빠른 처리는 불가능

■ 하이브 설치

- sever01에 bigdata로 접속한 뒤 홈디렉토리에 하이브 설치

```
$ wget http://mirror.navercorp.com/apache/hive/hive-2.3.3/apache-hive-2.3.3-bin.tar.gz
```

- 설치된 apache-hive-2.3.3-bin.tar.gz 파일 압축 풀기

```
$ tar xvfz apache-hive-2.3.3-bin.tar.gz
```

- 하이브 홈디렉토리의 conf 디렉토리로 이동

```
$ cd apache-hive-2.3.3-bin/conf
```

- hive-env.sh.template로 hive-env.sh 생성

```
$ mv hive-env.sh.template conf/hive-env.sh
```

- hive-env.sh 파일에 아래처럼 하둡 홈디렉터리 경로 입력

```
HADOOP_HOME=/home/bigdata/hadoop-2.9.1
```

- vi에디터로 hive-site.xml 만들기

```
$ vi hive-site.xml
```

- hive-site.xml 파일에 아래의 설정을 입력

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
   <property>
     <name>hive.metastore.warehouse.dir</name>
     <value>/user/hive/warehouse</value>
   </property>
   <property>
     <name>hive.cli.print.header</name>
     <value>true</value>
   </property>
   <property>
     <name>javax.jdo.option.ConnectionURL</name>
     <value>jdbc:oracle:thin:@192.168.0.103:1521:xe</value>
   </property>
   <property>
     <name>javax.jdo.option.ConnectionDriverName</name>
     <value>oracle.jdbc.OracleDriver</value>
   </property>
   <property>
     <name>hive.metastore.local</name>
     <value>false</value>
   </property>
   <property>
     <name>javax.jdo.option.ConnectionUserName</name>
     <value>hive</value>
   </property>
   <property>
     <name>javax.jdo.option.ConnectionPassword</name>
     <value>hive</value>
   </property>
</configuration>
```

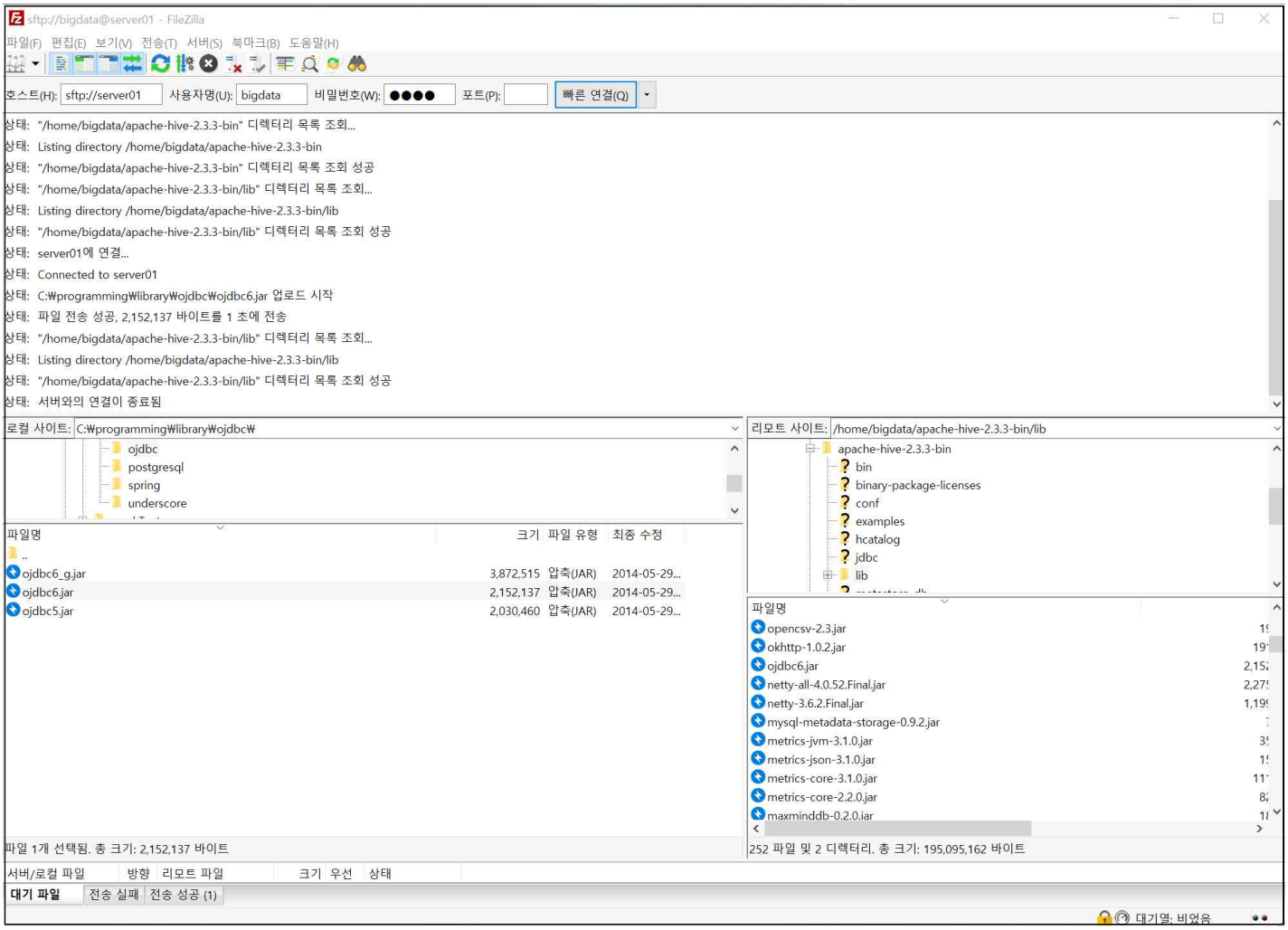
◦ 하이브는 기본적으로 아파치 더비디비를 메타스토어로 사용

◦ 만약 다른 DBMS를 메타스토어로 사용하려면 해당 DBMS의 JDBC드라이브를 하이브의 lib 디렉터리에 복사한 뒤 hive-site를 수정

◦ 우리는 Oracle을 메타스토어로 사용

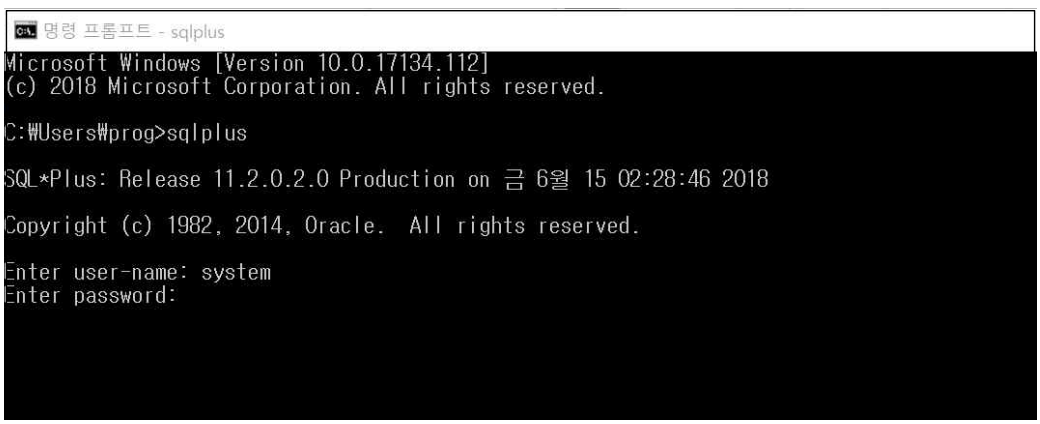
- FileZilla를 통해 ojdbc6.jar 파일을 hive의 lib에 가져다 놓음

/home/bigdata/apache-hive-2.3.3-bin/lib



- CMD창을 띄우고 sqlplus 입력 후 오라클 사용자 계정으로 접속

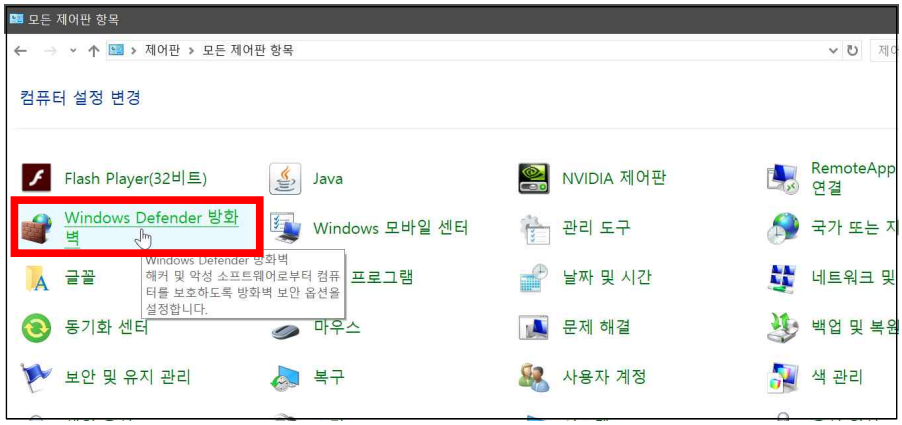
```
Enter user-name: system
Enter password: 1111
```



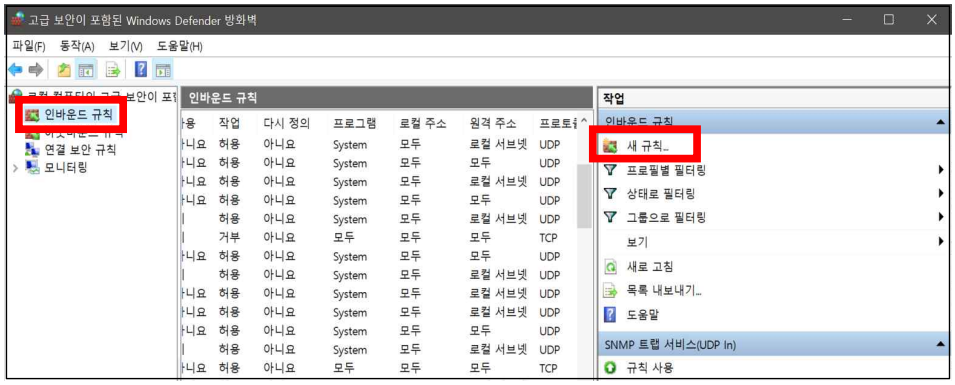
- 접속후 아래와 같이 입력

```
SQL> create user hive identified by hive;
grant connect to hive;
grant all privileges to hive;
```

- 제어판 -> 방화벽 -> 고급설정 을 차례로 클릭

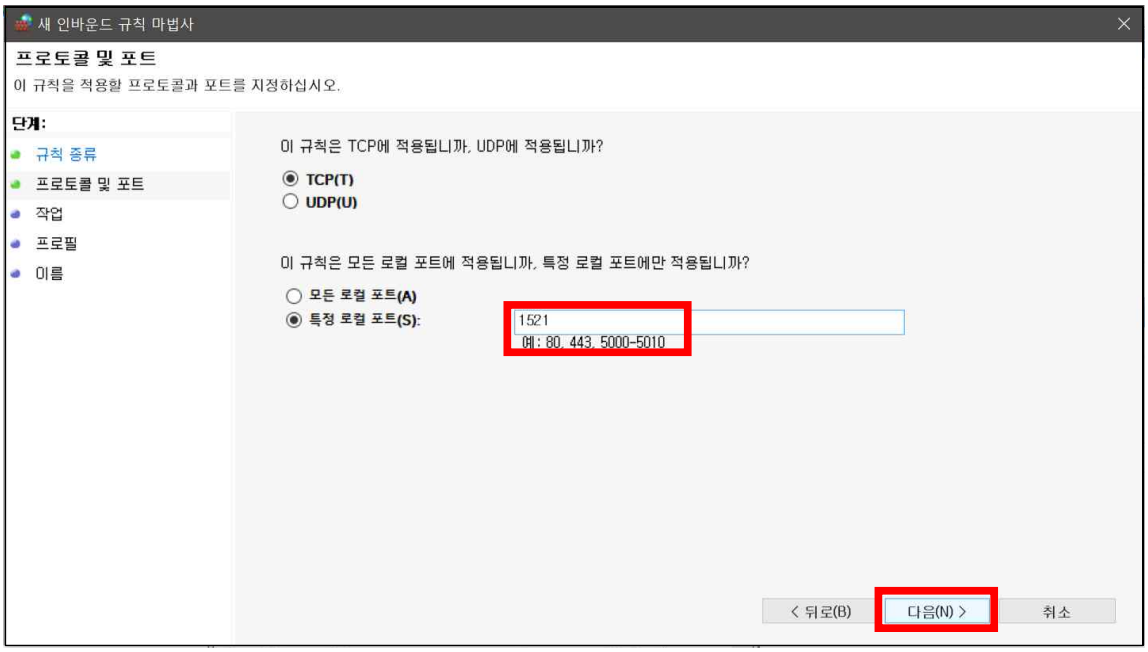


- 고급설정의 인바운드 규칙 클릭 후 오른편 새 규칙 클릭

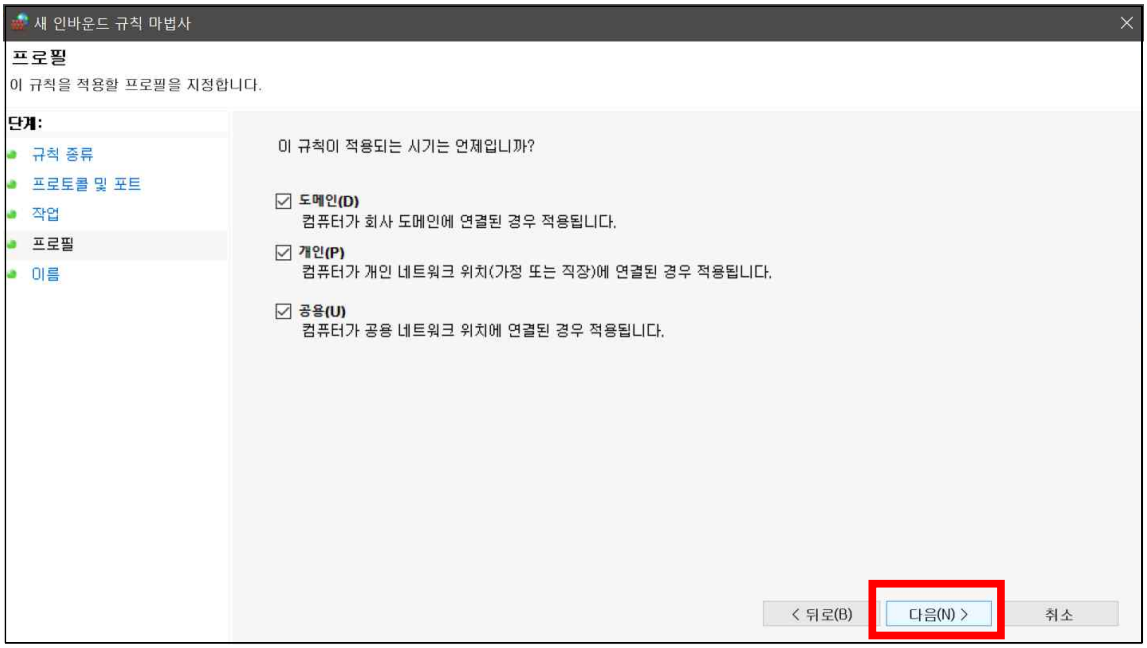


- 포트 선택 후 다음 클릭

- 특정 로컬 포트에 1521 작성 후 다음 클릭



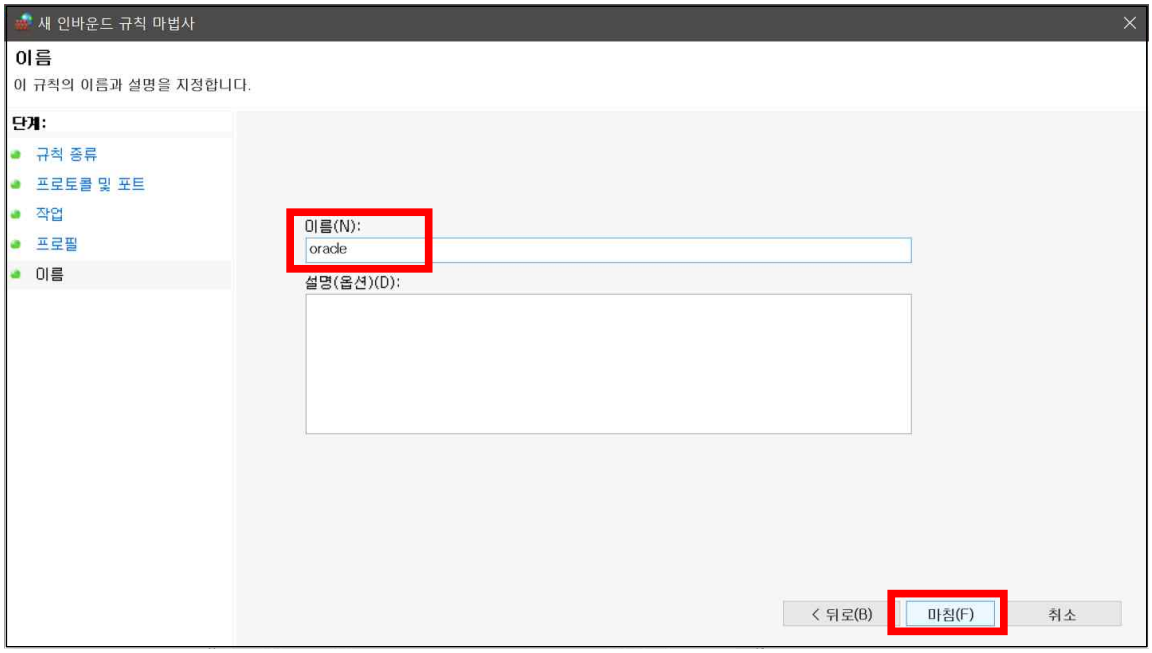
- 다음 클릭


새 인바운드 규칙 마법사
프로필
이 규칙을 적용할 프로필을 지정합니다.
단계:
규칙 종류
프로토콜 및 포트
작업
프로필
이름
이 규칙이 적용되는 시기는 언제입니까?
도메인(D)
컴퓨터가 회사 도메인에 연결된 경우 적용됩니다.
개인(P)
컴퓨터가 개인 네트워크 위치(가정 또는 직장)에 연결된 경우 적용됩니다.
공용(U)
컴퓨터가 공용 네트워크 위치에 연결된 경우 적용됩니다.
< 뒤로(B)
다음(N) >
취소

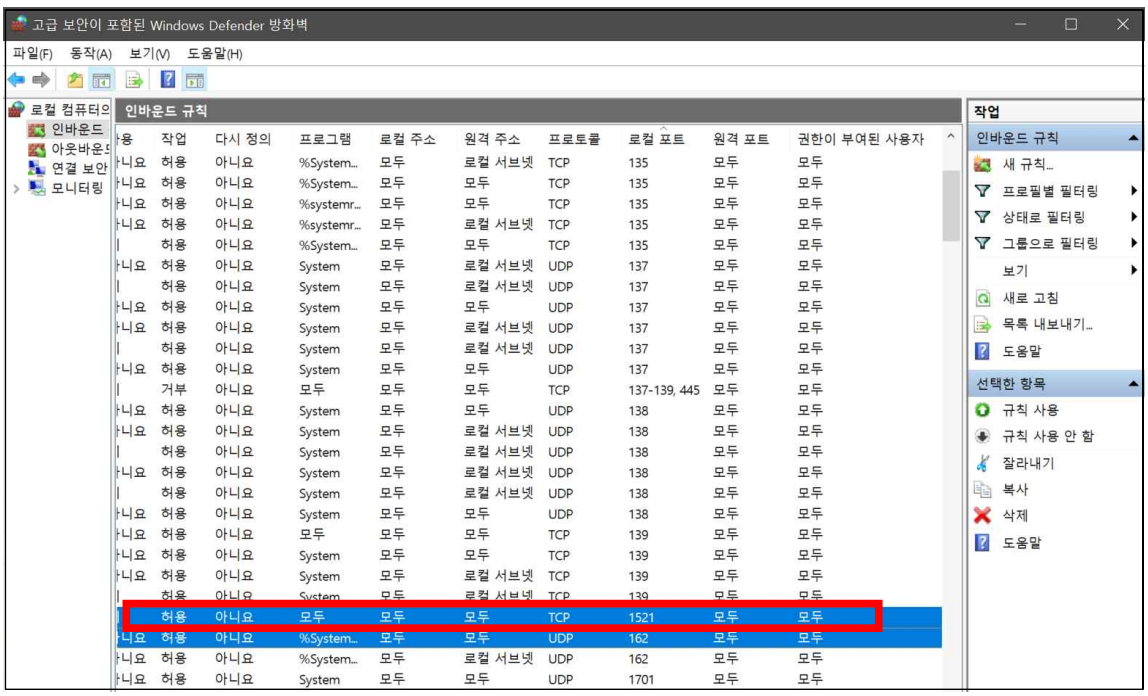
- 다음 클릭


새 인바운드 규칙 마법사
작업
규칙에 지정된 조건과 연결이 일치할 때 수행할 작업을 지정합니다.
단계:
규칙 종류
프로토콜 및 포트
작업
프로필
이름
지정된 조건과 연결이 일치할 경우 어떤 작업을 수행해야 합니까?
연결 허용(A)
IPsec으로 보호되는 연결과 보호되지 않은 연결이 포함됩니다.
보안 연결만 허용(C)
IPsec을 사용하여 인증된 연결만 포함됩니다. 연결 보안 규칙 노드의 IPsec 속성 및 규칙 설정을 사용하여 연결이 보호됩니다.
사용자 지정(Z)...
연결 차단(K)
< 뒤로(B)
다음(N) >
취소

- 이름에 oracle 작성 후 마침 클릭



- 1521 포트 확인 가능



- server01에 bigdata로 로그인 후 하둡으로 이동해 tmp 디렉터리가 없다면 생성

```
$ cd /home/bigdata/hadoop-2.9.1
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -mkdir /tmp
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -mkdir /tmp/hive
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -mkdir /tmp/hadoop-yarn
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -mkdir /tmp/hadoop-yarn/staging
```

○ 하이브에서 업로드하는 데이터는 HDFS의 /user/hive/warehouse 에 저장

○ 하이브에서 실행하는 잡의 여유 공간으로 HDFS의 /tmp/hive-유저명 디렉터리 사용

- HDFS에 필요한 디렉터리를 생성

```
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -mkdir /user/hive
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -mkdir /user/hive/warehouse
```

- 권한 변경

```
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -chmod 777 /tmp
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -chmod 777 /tmp/hive
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -chmod 777 /tmp/hadoop-yarn
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -chmod 777 /tmp/hadoop-yarn/staging
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -chmod 777 /user/hive/warehouse
```

- 하이브 디렉터리로 이동후 메타스토어 초기화

```
$ cd /home/bigdata/apache-hive-2.3.3-bin/
[bigdata@server01 apache-hive-2.3.3-bin]$ ./bin/schematool -initSchema -dbType oracle
```

- 하이브 시작

```
[bigdata@server01 apache-hive-2.3.3-bin]$ ./bin/hive
```

- 하이브 설치 확인

```
hive> show databases;
```

```
hive> show databases;
OK
default
Time taken: 0.066 seconds, Fetched: 1 row(s)
```

■ Beeline

1) HiveServer2는 HiveServer2 작동하는 새로운 명령 셸 Beeline를 지원

2) SQLLine CLI을 기반으로 JDBC 클라이언트

3) HiveServer2 에 접속하여 command shell 을 수행할 수 있도록 도와주는 client

- 하둡의 core-site.xml에 아래의 속성 추가

```
$ /home/bigdata/hadoop-2.9.1/etc/hadoop
$ vi core-site.xml
```

```
<property>
     <name>hadoop.proxyuser.bigdata.hosts</name>
     <value>*</value>
</property>
<property>
     <name>hadoop.proxyuser.bigdata.groups</name>
     <value>*</value>
</property>
```

- 하둡 및 server01~04를 종료 한뒤 다시 시작

- 하이브 bin 디렉터리로 이동

```
$ cd /home/bigdata/apache-hive-2.3.3-bin/bin
```

- hiveserver2 시작

```
$ ./hiveserver2
```

```
[bigdata@server01 apache-hive-2.3.3-bin]$ ./bin/hiveserver2
which: no hbase in (/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/bigd
ata/hadoop-2.9.1/bin:/home/bigdata/apache-flume-1.8.0-bin/bin:/usr/local/java/bi
n:/home/bigdata/apache-hive-2.3.3-bin/bin:/usr/local/apache-maven-3.5.3/bin:/hom
e/bigdata/.local/bin:/home/bigdata/bin)
2018-06-12 13:20:34: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/apache-hive-2.3.3-bin/lib/log4j-
slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop-2.9.1/share/hadoop/common
/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

- bigdata계정으로 server01 세션을 하나 더 열어서 하이브 bin 디렉터리로 이동

```
$ cd /home/bigdata/apache-hive-2.3.3-bin/bin
```

- beeline 실행

```
$ ./beeline
```

```
beeline> !connect jdbc:hive2://localhost:10000
```

- hiveserver2와 연결이 되었으면 beeline 테스트

```
0: jdbc:hive2://localhost:10000> show databases;
```

```
0: jdbc:hive2://localhost:10000> show databases;
+----------------+
| database_name  |
+----------------+
| default        |
+----------------+
1 row selected (2.891 seconds)
```

- 종료시 '!q' 입력

4) beeline 실행시 자동으로 연결시키기 위하여 beeline-hs2-connection.xml 과 beeline-site.xml이 필요함

- beeline-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
  <name>beeline.hs2.jdbc.url.tcpUrl</name>
  <value>jdbc:hive2://localhost:10000/default</value>
</property>

<property>
  <name>beeline.hs2.jdbc.url.httpUrl</name>

<value>jdbc:hive2://localhost:10000/default;transportMode=http;httpPath=cliservice</val
ue>
</property>

<property>
  <name>beeline.hs2.jdbc.url.default</name>
  <value>tcpUrl</value>
</property>
</configuration>
```

- beeline-hs2-connection.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
  <name>beeline.hs2.connection.user</name>
  <value>hive</value>
</property>
<property>
  <name>beeline.hs2.connection.password</name>
  <value>hive</value>
</property>
</configuration>
```

■ 하이브 DataBase

- 하이브의 데이터베이스 생성

```
CREATE DATABASE [IF NOT EXISTS] DB이름
```

◦ CREATE DATABASE DB이름: 데이터베이스를 만드는 구문

◦ IF NOT EXISTS: DB이름과 같은 데이터베이스가 없을 시 생성

- userdb 생성

```
CREATE DATABASE IF NOT EXISTS userdb;
```

- hdfs에서 확인해보면 DB 경로 생성 확인 가능

```
[bigdata@server01 example_data]$ hdfs dfs -ls /user/hive/warehouse
Found 3 items
drwxrwxrwx   - hive supergroup          0 2018-06-14 11:07 /user/hive/warehouse
drwxrwxrwx   - hive supergroup          0 2018-06-14 11:53 /user/hive/warehouse/userdb.db
```

- 데이터베이스 지우기

```
DROP DATABASE [IF EXISTS] DB명;
```

- userdb 지우기

```
DROP DATABASE IF EXISTS userdb;
```

■ 하이브 테이블

- RDBMS 테이블과 유사한 스키마를 가짐
- 파일 시스템의 파일을 테이블 데이터로 사용
- 테이블 종류

• 관리(managed)테이블 : 하이브가 관리하며 테이블 삭제 시 메타 정보와 파일 함께 삭제 됨

• 외부(external)테이블 : 테이블 생성 시 설정한 경로로 데이터를 저장하며 테이블 삭제 시 메타 정보만 삭제되고 데이터가 보존됨

- 하이브 데이터 타입

데이터 타입	설명
TINYINT	1바이트 크기 정수
SMALLINT	2바이트 크기 정수
INT	4바이트 크기 정수
BIGINT	8바이트 크기 정수
BOOLEAN	TRUE 또는 FALSE
FLOAT	4바이트 부동소수점
DOUBLE	8바이트 부동소수점
STRING	문자열
TIMESTAMP	타임스탬프 (1970년 1월 1일 0시 이후 흘러간 시간)

기타: BINARY, STRUCT, MAP, ARRAY

- Putty로 server01세션을 하나 더 열어 bigdata계정으로 접속 후 하이브 실습용 데이터를 저장할 디렉토리 생성

```
$ mkdir example_data
```

- FileZilla를 통해 HDFS에 call_chicken_04month.csv 파일 업로드
- SK Big Data Hub(https://www.bigdatahub.co.kr/index.do)에서 제공하는 파일

- SK 텔레콤에서 수집한 데이터를 SK Big Data Hub 가입후 무료로 이용 가능

- call_chicken_04month.csv: 서울 지역 치킨집에 대한 한달 통화량을 기반으로 지역별 이용 현황 데이터를 제공

1	기준일	요일	성별	연령대	시도	시군구	읍면동	업종	통화건수
2	20180401	일	남	10대	서울특별시	강남구	논현동	치킨	5
3	20180401	일	남	10대	서울특별시	강남구	삼성동	치킨	11
4	20180401	일	남	10대	서울특별시	강남구	역삼동	치킨	10
5	20180401	일	남	20대	서울특별시	강남구	개포동	치킨	14
6	20180401	일	남	20대	서울특별시	강남구	도곡동	치킨	5
7	20180401	일	남	20대	서울특별시	강남구	삼성동	치킨	33
8	20180401	일	남	30대	서울특별시	강남구	개포동	치킨	12
9	20180401	일	남	30대	서울특별시	강남구	도곡동	치킨	5
10	20180401	일	남	30대	서울특별시	강남구	일원동	치킨	5

◦ 시도/시군구/읍면동: 이용자의 발신지역을 기준으로 서울지역 동단위까지 제공

◦ 연령대: 10세 단위로 6개 연령대(10대, 20대, 30대, 40대, 50대, 60대 이상)

◦ 통화건수: 이용자의 치킨집 통화건수(통화량 5건 미만은 5건으로 표시)

- 업로드한 csv파일에 따옴표와 컬럼명이 있다면 제거

◦ find: 디스크에 저장된 파일/디렉터리 검색

◦ 문자열찾은 후 치환: find . -exec perl -pi -e 's/찾을문자열/바꿀문자열/g' {} \;

◦ sed: 필터링과 텍스트를 변환하는 스트림 편집기

```
#따옴표 제거
$ find . -name call_chicken_04month.csv -exec perl -pi -e 's/"//g' {} \;
#맨 위 한줄 제거(컬럼명 제거)
$ sed -e '1d' call_chicken_04month.csv > call_chicken_04month_new.csv
#파일 원래 이름으로
$ mv call_chicken_04month_new.csv call_chicken_04month.csv
```

- 관리테이블 생성

```
CREATE TABLE 테이블명 (칼럼명 칼럼 타입, ...)
COMMENT '테이블 설명'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS '파일 포맷형식';
```

◦ CREATE TABLE 테이블명 (칼럼명 칼럼 타입, …) -> 테이블 만드는 최소 조건

◦ ROW FORMAT: 해당 테이블 내의 데이터가 어떤 형식으로 저장될지 설정

◦ 필드를 콤마 기준으로 구분(만약 필드가 탭으로 구분되어있다면: \t)

◦ 행과 행은 \n 값으로 구분

◦ STORED AS: 데이터 저장 파일 포맷

◦ COMMENT: 테이블의 설명을 참고용으로 등록

◦ DBNAME: 데이터베이스명 -> 생략시 default DB 사용

- chicken_04month 테이블 생성

```
CREATE TABLE chicken_04month(sysdate STRING, dayOfWeek STRING, gender STRING, age
STRING , city STRING , district STRING , town STRING, type STRING, calls INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

- 외부테이블 생성

```
CREATE EXTERNAL TABLE chicken_04month_ex(sysdate STRING, dayOfWeek STRING, gender STRING, age STRING , city STRING , district STRING , town STRING, type STRING, calls INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/hadoop/chicken_04month';
```

◦ EXTERNAL 키워드로 외부 테이블 생성

◦ LOCATION: 데이터를 저장할 경로 생성

◦ 외부 테이블은 Drop을 해도 데이터가 보존됨

◦ HDFS에서 외부테이블로 저장한 데이터 확인 가능

```
[bigdata@server01 ~]$ ./hadoop-2.9.1/bin/hdfs dfs -ls -R /user/hadoop
drwxr-xr-x   - bigdata supergroup          0 2018-06-08 14:51 /user/hadoop/chicken_04month
```

- 파티션 된 테이블

```
CREATE TABLE chicken_calls(sysdate STRING, dayOfWeek STRING, gender STRING, age STRING , city STRING , district STRING , town STRING, type STRING, calls INT)
PARTITIONED BY (year string, month string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

◦ PARTITIONED BY (파티션키 파티션타입): 테이블의 파티션 설정

◦ 파티션키는 해당 테이블의 새로운 칼럼으로 추가됨

◦ 하이브는 쿼리문의 수행 속도를 향상 시키기 위해 파티션 설정 가능

◦ 파티션을 설정하면 해당 테이블의 데이터를 파티션 별로 디렉터리 생성해 저장

◦ 아래의 사진은 생성된 테이블에 데이터를 로딩한 결과 HDFS 결과

```
$ hdfs dfs -ls -R /user/hive/warehouse/chicken_calls
```

```
/user/hive/warehouse/chicken_calls
/user/hive/warehouse/chicken_calls/year=2018
/user/hive/warehouse/chicken_calls/year=2018/month=04
/user/hive/warehouse/chicken_calls/year=2018/month=04/call_chicken_04month.csv
```

■ 데이터 업로드

- HDFS에서 데이터를 로딩

```
LOAD DATA INPATH '디렉터리 경로'
OVERWRITE INTO TABLE 테이블 명;
```

◦ LOAD DATA INPATH '디렉토리 경로': 디렉토리에 포함된 모든 파일을 로딩

◦ OVERWRITE: 기존의 파일들을 삭제

- 로컬에서 데이터를 로딩

```
LOAD DATA LOCAL INPATH '디렉터리 경로'
OVERWRITE INTO TABLE 테이블 명;
```

◦ LOCAL: 로컬 파일을 테이블 데이터 경로에 로딩

- 파티션된 테이블에 데이터 로딩

```
LOAD DATA local INPATH '디렉토리 경로'
OVERWRITE INTO TABLE 테이블 명
PARTITION (파티션 키='값');
```

- 외부(로컬) 데이터를 하이브 테이블로 로딩하기 :

/home/bigdata/example_data/ 경로에 있는 call_chicken_04month.csv 업로드

```
LOAD DATA LOCAL INPATH '/home/bigdata/example_data/call_chicken_04month.csv'
OVERWRITE INTO TABLE chicken_04month;
```

```
[bigdata@server01 example_data]$ hdfs dfs -ls /user/hive/warehouse
Found 2 items
drwxrwxrwx   - hive supergroup          0 2018-06-14 11:07 /user/hive/warehouse/chicken_04month
drwxrwxrwx   - hive supergroup          0 2018-06-14 11:53 /user/hive/warehouse/userdb.db
```

- select로 chicken_04month 테이블의 데이터를 5개만 확인

```
SELECT sysdate, dayofweek, gender, age, city, district, town, type, calls
FROM chicken_04month limit 5;
```

sysdate	dayofweek	gender	age	city	district	town	type	calls
20180401	일	남	10대	서울특별시	강남구	논현동	치킨	5
20180401	일	남	10대	서울특별시	강남구	삼성동	치킨	11
20180401	일	남	10대	서울특별시	강남구	역삼동	치킨	10
20180401	일	남	20대	서울특별시	강남구	개포동	치킨	14
20180401	일	남	20대	서울특별시	강남구	도곡동	치킨	5

- 외부(로컬) 데이터를 하이브 파티션 테이블로 로딩하기 :

/home/bigdata/example_data/ 경로에 있는 call_chicken_04month.csv 업로드

```
LOAD DATA LOCAL INPATH '/home/bigdata/example_data/call_chicken_04month.csv'
OVERWRITE INTO TABLE chicken_calls
PARTITION(year='2018', month='04');
```

- select로 chicken_calls 테이블의 데이터를 5개만 확인

```
SELECT sysdate, dayofweek, gender, age, district, calls, year, month
FROM chicken_calls
limit 5;
```

sysdate	dayofweek	gender	age	district	calls	year	month
20180401	일	남	10대	강남구	5	2018	04
20180401	일	남	10대	강남구	11	2018	04
20180401	일	남	10대	강남구	10	2018	04
20180401	일	남	20대	강남구	14	2018	04
20180401	일	남	20대	강남구	5	2018	04

◦ 파티션 키(year, month)가 컬럼이 된 것을 확인

■ 테이블 지우기

- 테이블 구조 + HDFS에 저장된 데이터 모두 삭제

```
DROP TABLE [IF EXISTS] 테이블명;
```