

리눅스 기초 및 하둡



1. 리눅스 기초

1. VMWare를 이용한 가상화
2. 디렉터리와 파일 사용하기
3. 리눅스 vi 에디터 사용법

2. 빅데이터 개념

1. 빅데이터 처리 시스템 개요
2. 빅데이터 처리 인프라 및 S/W
3. 하둡클러스터 동작 방식

2. 하둡

1. Hadoop 종류와 구성요소
2. 하둡 클러스터 구축을 위한 설정
Full Distributed Mode 하둡 클러스터 설치
3. Spark를 이용한 데이터 분석
4. 하둡 클러스터 구성 관련 참고사항

4. 호튼웍스 샌드박스

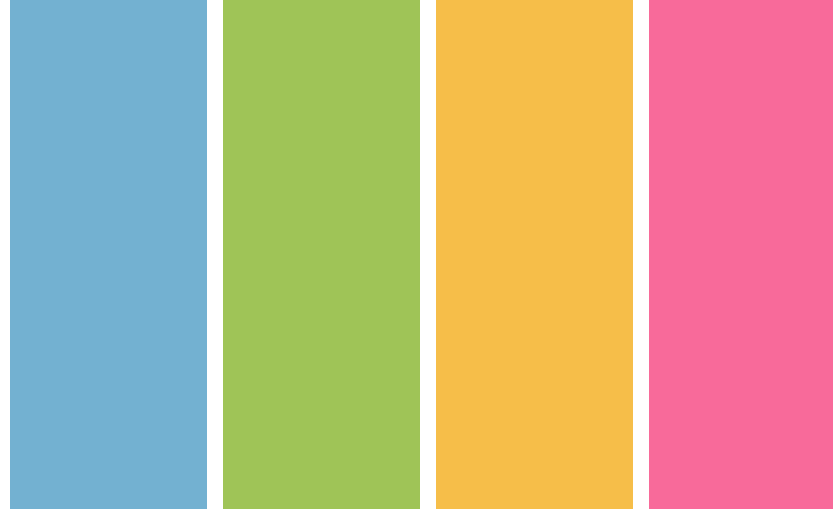
1. 호튼웍스 샌드박스 설치
2. 하둡 기본 명령어
3. Sqoop
4. Pig
5. Hive

<https://bit.ly/33ISGQk>



04 호튼웍스 샌드박스


1. 호튼웍스 샌드박스 설치
2. 하둡 기본 명령어
3. Sqoop
4. Pig
5. Hive



1. 호튼웍스 샌드박스 설치

가상화 기술

1. 호튼웍스 샌드박스 설치



[Hortonworks_Sandbox_2.1.ova](#)

http://hortonassets.s3.amazonaws.com/2.1/virtualbox/Hortonworks_Sandbox_2.1.o...

33.2KB/초 - 2.6GB중 1,954KB, 22시간 남음

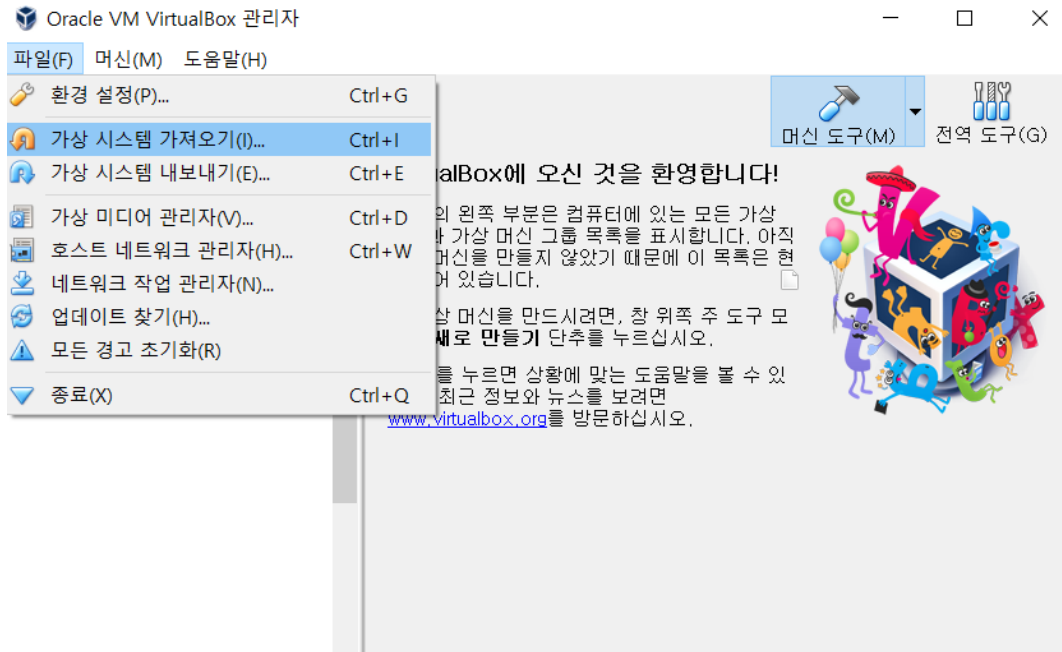
[일시중지](#)[취소](#)

http://hortonassets.s3.amazonaws.com/2.1/virtualbox/Hortonworks_Sandbox_2.1.ova

위의 링크를 선택하면 다운 받을 수 있다

가상화 기술

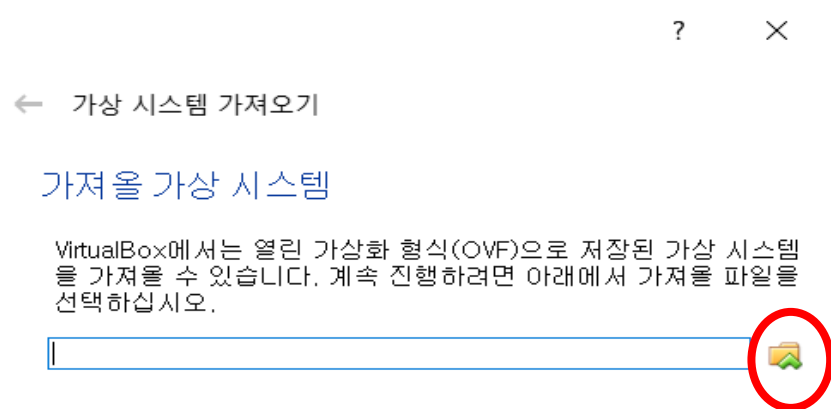
1. 호튼웍스 샌드박스 설치



좌측상단 파일 -> 가상 시스템 가져오기 선택

가상화 기술

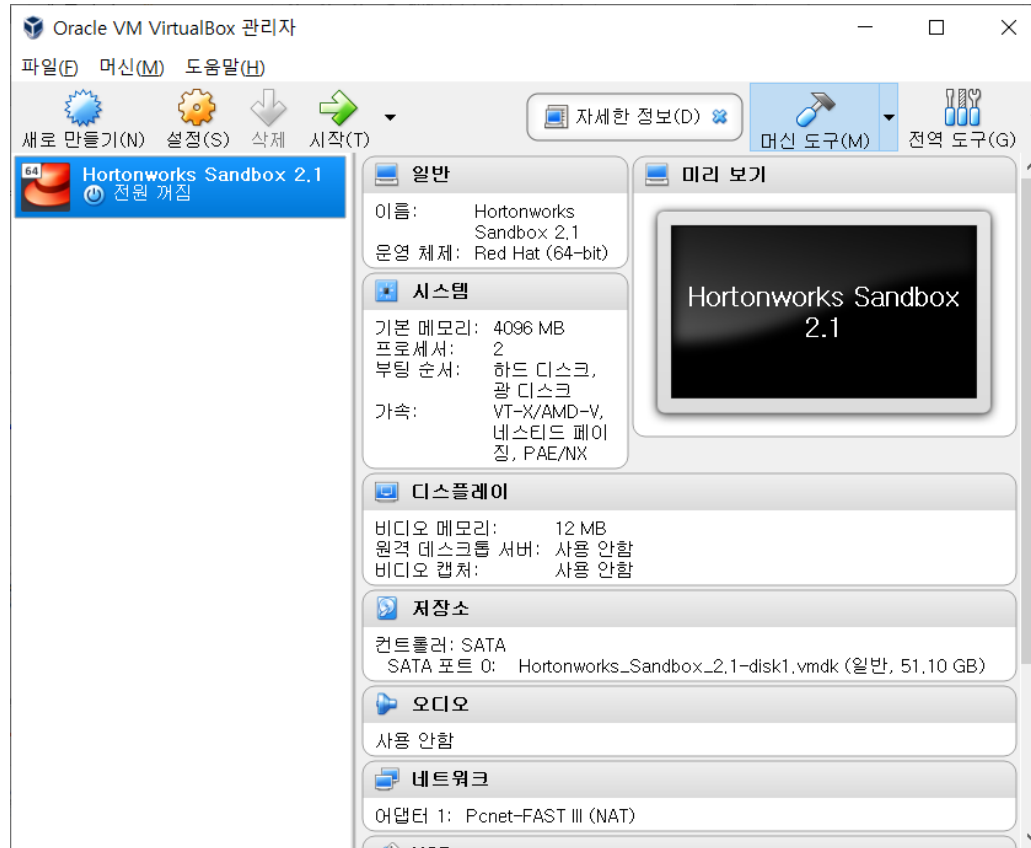
1. 호튼웍스 샌드박스 설치



다운 받은 Hortonworks_Sandbox_2.1.ova 연결하기위해 파일 찾아보기 버튼 클릭하여 가상 시스템을 가져온다

가상화 기술

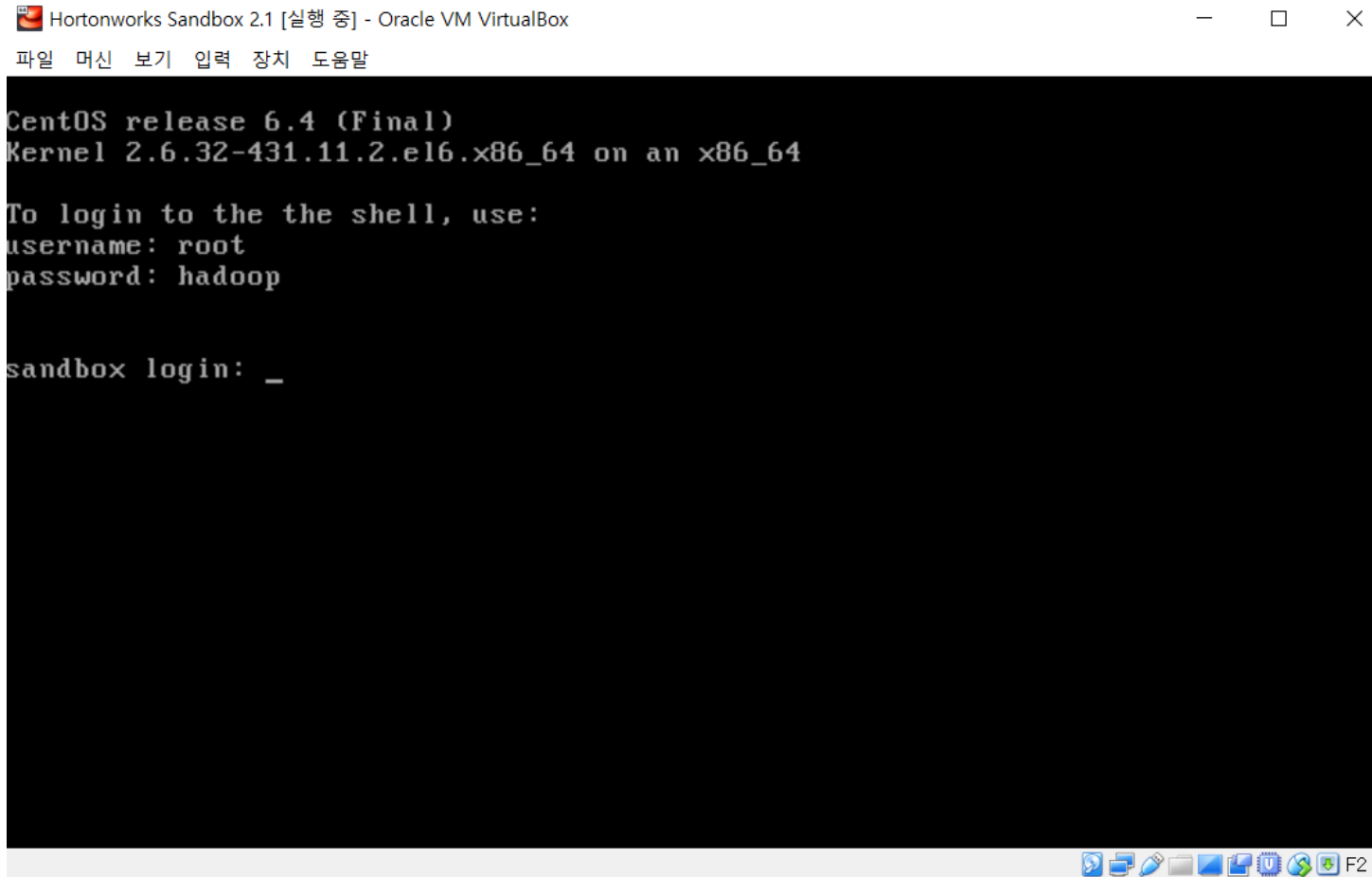
1. 호튼웍스 샌드박스 설치



완료되면 왼쪽에 **Sandbox 2.1**이 올라간 것을 확인할 수 있으며 시작버튼을 눌러 가상 시스템을 시작한다
혹시 오류가 났을 경우 VirtualBox버전을 확인한 후 버전에 맞는 **Extension Pack** 다운 하여 저장하여야 한다
VirtualBox 버전은 **5.2.32** 버전

가상화 기술

1. 호튼웍스 샌드박스 설치



```
Hortonworks Sandbox 2.1 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

CentOS release 6.4 (Final)
Kernel 2.6.32-431.11.2.el6.x86_64 on an x86_64

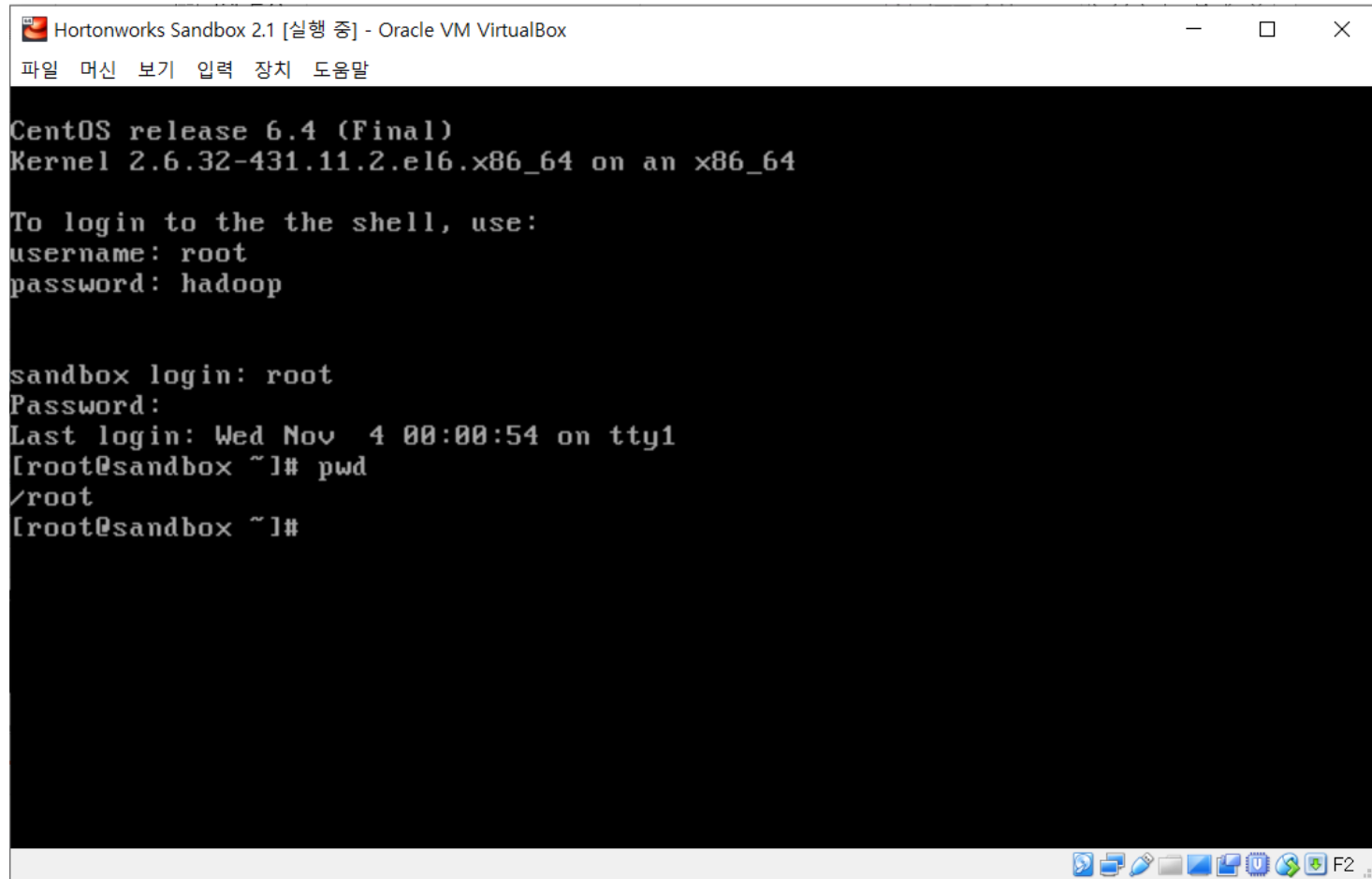
To login to the the shell, use:
username: root
password: hadoop

sandbox login: _
```

로그인한다(login : root / password:hadoop)

가상화 기술

1. 호튼웍스 샌드박스 설치



```
Hortonworks Sandbox 2.1 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

CentOS release 6.4 (Final)
Kernel 2.6.32-431.11.2.el6.x86_64 on an x86_64

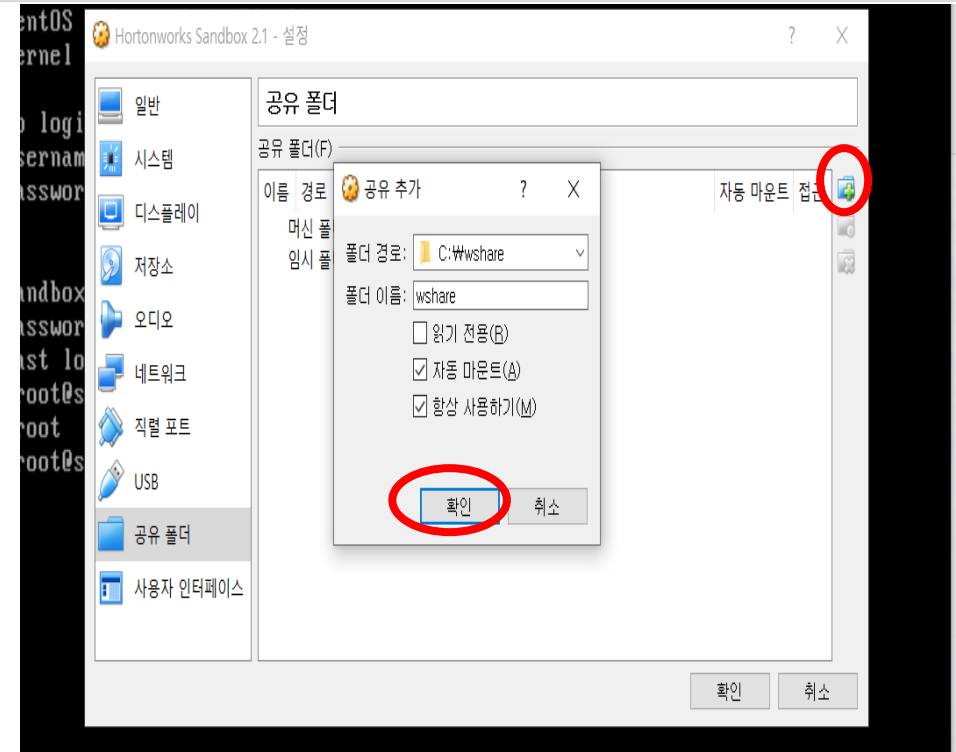
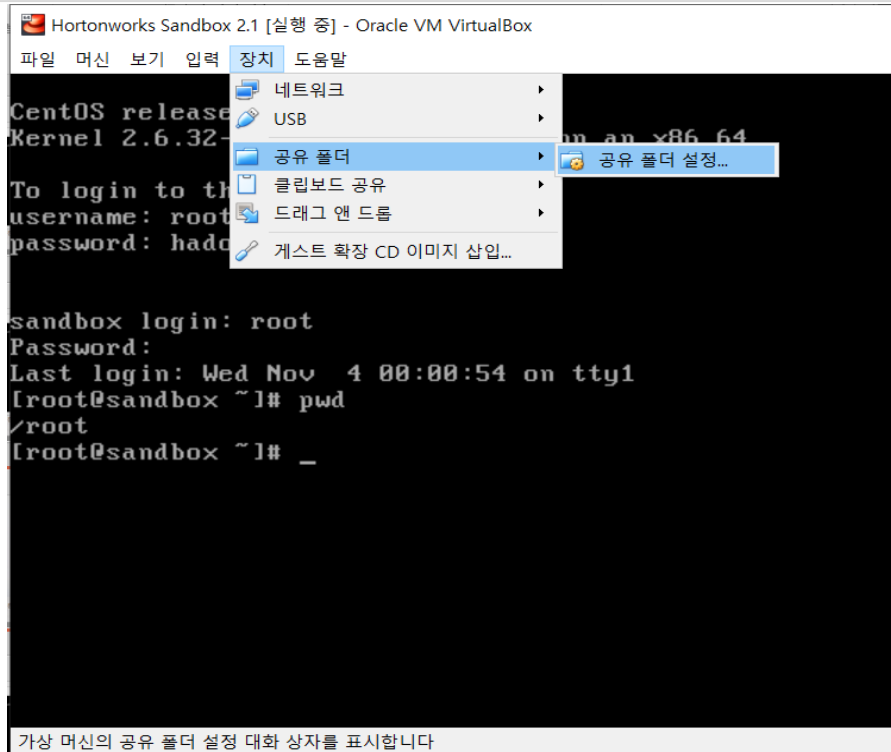
To login to the the shell, use:
username: root
password: hadoop

sandbox login: root
Password:
Last login: Wed Nov  4 00:00:54 on tty1
[root@sandbox ~]# pwd
/root
[root@sandbox ~]#
```

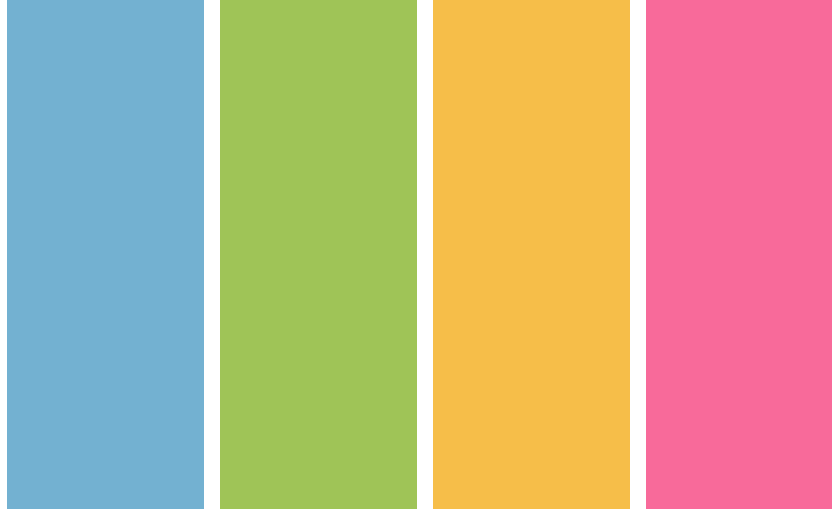
정상로그인 된 후 호튼웍 리눅스의 홈디렉토리는 /root임을 알 수 있다

가상화 기술

1. 호튼웍스 샌드박스 설치



- (1) 장치 - 공유폴더 - 공유폴더 설정
- (2) 공유할 폴더 추가 - 폴더경로와 이름을 설정 - 자동마운트, 항상사용하기체크 - 확인클릭
- (3) mkdir lshare
- (4) sudo mount -t vboxsf wshare lshare
- (5) 마운트가 성공하면 ls하여 확인시 마운트된 폴더가 초록색으로 보임



2. 하둡 기본 명령어

하둡 명령어 – hdfs에 데이터 넣기

2. 하둡 명령어

```
lshare]# hadoop fs -put 2_stocks.csv stocks.csv
```

리눅스경로및파일 하둡경로및파일

```
lshare]# hadoop fs -ls
```

```
lshare]# hadoop fs -ls /
```

```
lshare]# hadoop fs -ls /user/root    하둡의 홈디렉토리 / 리눅스홈디렉토리 /root
```

```
lshare]# hadoop fsck /user/root/stocks.csv    블록수 1개 확인
```

```
lshare]# hadoop fs -stat %r stocks.csv
```

```
lshare]# hadoop fs -rm stocks.csv    하둡에 저장된 stocks.csv를 삭제
```

```
lshare]# hadoop fs -ls    삭제 확인
```

```
lshare]# hadoop fs -rm -R .Trash    삭제 히스토리도 삭제
```

```
lshare]# hadoop fs-put 2008.csv    블록사이즈보다 큰 파일
```

```
lshare]# hadoop fs -ls    삭제 확인
```

```
lshare]# hadoop fsck /user/root/2008.csv    블록수 확인
```

하둡 명령어 – hdfs에 데이터 넣기

2. 하둡 명령어

```
lshare]# hadoop fs -mkdir test
lshare]# hadoop fs -mkdir test/test1
lshare]# hadoop fs -mkdir test/test2
lshare]# hadoop fs -mkdir test/test2/test3
lshare]# hadoop fs -ls -R 하둡 홈디렉토리 하위디렉토리 확인
```

```
lshare]# hadoop fs -rm -R test/test2 하둡의 test2 삭제한다
lshare]# hadoop fs -rm -R .Trash
```

```
lshare]# hadoop fs -put 2_data.txt test/data.txt
lshare]# hadoop fs -cp test/data.txt test/test1/data2.txt
lshare]# hadoop fs -ls -R test
```

data2.txt와 test1 폴더를 삭제하세요

```
lshare]# hadoop fs -mv test/data.txt test/data1.txt
```

```
lshare]# hadoop fs -cat test/data1.txt | head -10
lshare]# hadoop fs -cat test/data1.txt | tail -10
```

```
lshare]# hadoop fs -tail test/data1.txt
```

```
lshare]# hadoop fs -ls 확인
```

하둡 명령어 – hdfs에서 데이터 가져오기 및 병합

2. 하둡 명령어

```
root@sandbox ~]# mkdir tmp
```

```
root@sandbox ~]# hadoop fs -get test/data1.txt tmp/
```

```
root@sandbox ~]# ls tmp/
```

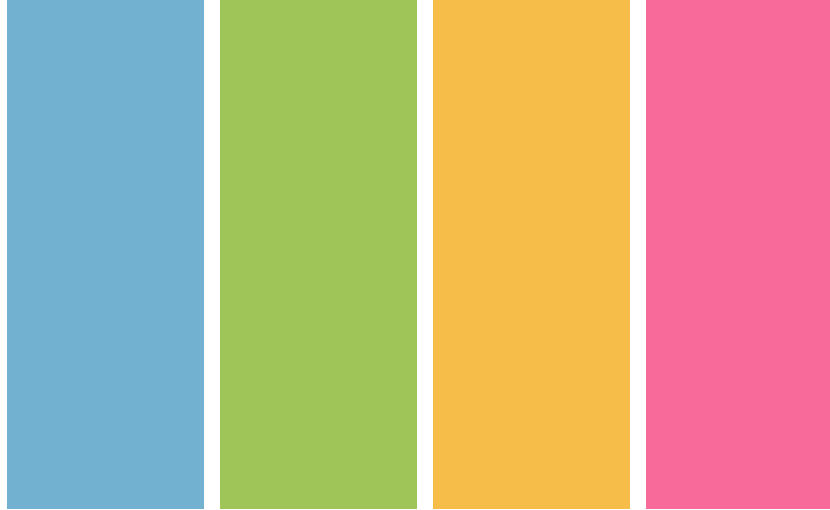
```
root@sandbox ~]# hadoop fs -put lshare/2_stocks.csv test/
```

```
root@sandbox ~]# hadoop fs -ls -R test/
```

```
root@sandbox ~]# hadoop fs -getmerge test/ lshare/merged.txt 병합
```

```
root@sandbox ~]# cat lshare/merged.txt
```

```
root@sandbox ~]# hadoop version
```

3. Sqoop

스쿱에 액세스 할 데이터 mysql에 넣기

3. Sqoop

- 호튼웍스 샌드박스에는 mySql과 sqoop이 설치되어 있음

```
root@sandbox ~]# mysql -u root -p
mysql> show databases;
mysql> use test; 호튼웍스에서는 test만 스쿱에서 액세스하도록 config
mysql> create table salaries (
            gender varchar(1),
            age      int,
            salary double,
            zipcode int);
mysql> show tables;
mysql> desc salaries;
mysql> alter table salaries add column `id` int(10) unsigned primary key auto_increment;
mysql> desc salaries; 스쿱에서 액세스하기 위해서는 반드시 주키가 있어야 함
mysql> insert into salaries (gender, age, salary, zipcode) values
            ( 'm' , 40, 76000,95102);
mysql> insert into salaries (gender, age, salary, zipcode) values
            ( 'f' , 20, 40000,89081);
mysql> insert into salaries (gender, age, salary, zipcode) values
            ( 'm' , 30, 60000,94311);
mysql> select * from salaries;
```

스쿱을 활용한 수집

3. Sqoop

- 스쿱 실행

```
root@sandbox ~]# sudo sqoop import --connect jdbc:mysql://localhost/test --table salaries  
--username root --target-dir /sqoop_out
```

```
root@sandbox ~]# hadoop fs -ls /sqoop_out
```

```
sqoop_out/__SUCCESS
```

```
sqoop_out/part-m-00000
```

```
sqoop_out/part-m-00001
```

```
sqoop_out/part-m-00002 m은 map단계에서 만들어짐. r은 reduce단계에서 만들어진 파일
```

```
root@sandbox ~]# hadoop fs -cat /sqoop_out/*
```

```
M, 40,76000.0, 95102, 1
```

```
...
```

```
root@sandbox ~]# mysql -u root -p
```

```
mysql> use test;
```

```
mysql> create table test1 (
```

스큐프를 활용한 수집

3. Sqoop

```
root@sandbox ~]# mysql -u root -p
mysql> use test;
mysql> create table test1 (big1 varchar(1), big2 varchar(2));
mysql> create table test2 (big3 varchar(1), big4 varchar(2));
mysql> create table test3 (
    id int(10) unsigned primary key auto_increment, big5 varchar(1), big6 varchar(2));
mysql> insert into test1 values ( 'A' , 'AA' );
mysql> insert into test1 values ( 'B' , 'BB' );
mysql> insert into test1 values ( 'C' , 'CC' );
mysql> insert into test2 values ( 'D' , 'DD' );
mysql> insert into test2 values ( 'E' , 'EE' );
mysql> insert into test2 values ( 'F' , 'FF' );
mysql> insert into test3 values ( 'G' , 'GG' );
mysql> insert into test3 values ( 'HH' , 'HH' );
mysql> insert into test3 values ( 'I' , 'II' );
mysql> alter table test1 add column `id` int(10) unsigned primary key auto_increment;
mysql> alter table test2 add column `id` int(10) unsigned primary key auto_increment;
mysql> select * from test1;
mysql> select * from test2;
mysql> select * from test3;
```

스큐어를 활용한 수집

3. Sqoop

```
root@sandbox ~]# sudo sqoop import --connect jdbc:mysql://localhost/test --table salaries --username root --target-dir /sqoop_out
```

```
root@sandbox ~]# sudo sqoop import-all-tables --connect jdbc:mysql://localhost/test --username root --warehouse-dir /sqoop_out2 복수테이블 import 시 --target-dir 대신 --warehouse-dir
```

```
root@sandbox ~]# hadoop fs -ls /sqoop_out2
```

```
root@sandbox ~]# hadoop fs -cat /sqoop_out2/test1/*
```

```
root@sandbox ~]# sudo sqoop import-all-tables --connect jdbc:mysql://localhost/test --username root --exclude-tables test3 --warehouse-dir /sqoop_out3 제외할 테이블--exclude
```

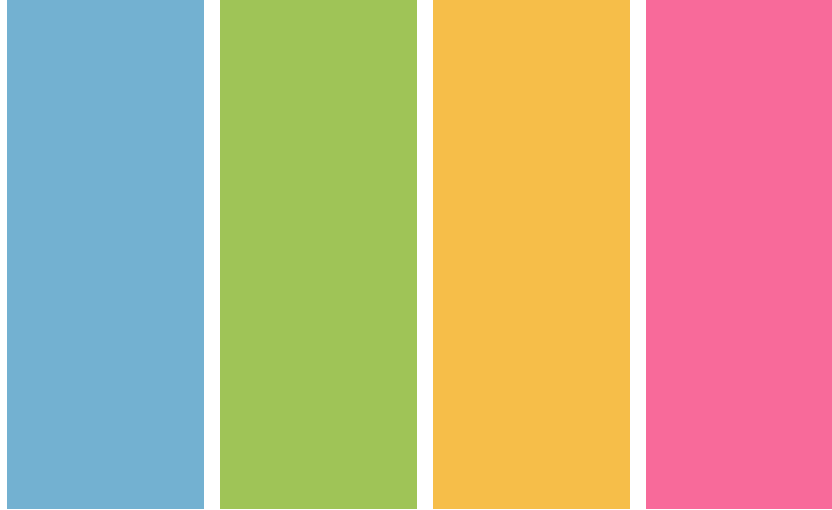
```
root@sandbox ~]# hadoop fs -ls /sqoop_out3
```

```
root@sandbox ~]# hadoop fs -cat /sqoop_out2/test1
```

```
root@sandbox ~]# sudo sqoop import --connect jdbc:mysql://localhost/test --table salaries --username root --target-dir /sqoop_out4 --where 'gender= "m" '
```

```
root@sandbox ~]# hadoop fs -cat /sqoop_out4/*
```

```
sudo mount -t vboxsf wshare lshare
```



4. Pig

- 1) 데이터 구조를 자세하게 검토할 명령어 제공
- 2) 입력 데이터의 대표 부분집합에 대해 표본실행

Pig 명령어

4. Pig

```
root@sandbox ~]# pig
```

```
grunt> quit
```

```
root@sandbox ~]# cd lshare
```

```
root@sandbox lshare]# cat pigdemo.txt
```

```
root@sandbox lshare]# pig
```

```
grunt> ls
```

```
grunt> mkdir demo
```

```
grunt> copyFromLocal /root/lshare/pigdemo.txt demo/
```

```
grunt> cd demo
```

```
grunt> ls
```

```
grunt> employees = LOAD 'pigdemo.txt' AS (state, name);
```

```
grunt> describe employees;
```

```
grunt> DUMP employees;
```


Pig 명령어

4. Pig

```
grunt> ca_only = FILTER employees BY (state=='CA');  
grunt> DUMP ca_only;
```

```
grunt> emp_group = GROUP employees BY state;  
grunt> describe emp_group;  
Grunt> DUMP emp_group;
```

```
grunt> STORE emp_group INTO 'emp_group.txt' ;  
grunt> ls
```

```
grunt> STORE emp_group INTO 'emp_group.csv' USING PigStorage(',') ;  
함수명은대소문자구문함
```

```
grunt> cat emp_group.csv
```

```
grunt> STORE employees INTO 'emp.csv' USING PigStorage(',') ;  
함수명은대소문자구문함
```

```
grunt> cat emp.csv
```

```
grunt> aliases; 릴레이션들은 pig가 종료되면 사라진다
```

Pig를 활용한 빅데이터 처리

4. Pig

```
root@sandbox lshare]# unzip whitehouse_visits.zip
```

```
root@sandbox lshare]# tail whitehouse_visits.txt (실수로 cat을 하면 빅데이터라 한세월 멈춘은  
ctrl + c)
```

```
root@sandbox lshare]# pig
```

```
grunt> mkdir whitehouse
```

```
grunt> copyFromLocal /root/lshare/whitehouse_visits.txt whitehouse/visits.txt
```

```
grunt> ls whitehouse
```

```
grunt> A = LOAD '/user/root/whitehouse/visits.txt' USING TextLoader();
```

```
grunt> describe A; A의 구성을 확인 스키마를 알수 없을 것이라고 출력
```

```
grunt> A_limit = LIMIT A 3;
```

```
grunt> DUMP A_limit;
```

Pig를 활용한 빅데이터 처리

4. Pig

```
grunt> visits = LOAD '/user/root/whitehouse/visits.txt' USING PigStorage( ',' );  
grunt> firstten = FOREACH visits GENERATE $0..$9; 0~9열만 뽑음  
grunt> firstten_limit = LIMIT firstten 50;  
grunt> DUMP firstten_limit;
```

```
grunt> lastfields = FOREACH visits GENERATE $19..$25;  
grunt> lastfields_limit = LIMIT lastfields 500;  
grunt> DUMP lastfields_limit
```

```
grunt> potus = FILTER visits BY $19 MATCHES 'POTUS' ;  
grunt> potus = FILTER visits BY ($19== 'POTUS' );  
grunt> potus_limit = LIMIT potus 500;  
grunt> DUMP potus_limit;
```

Pig를 활용한 빅데이터 처리

4. Pig

```
grunt> potus_details = FOREACH potus GENERATE  
>> $0 AS lname:chararray,  
>> $1 AS fname:chararray,  
>> $6 AS arrival_time:chararray,  
>> $19 AS visitee:chararray;
```

```
grunt> DESCRIBE potus_details;  
grunt> STORE potus_details INTO 'potus' USING PigStorage('\t');  
grunt> cd potus  
grunt> ls 하둡에 블록으로 나누어 저장된 것을 볼 수 있다  
grunt> cat part-m-00000
```

```
root@sandbox ~]# hadoop fs -cat potus/* 모두 볼 수 있음  
root@sandbox ~]# hadoop fs -cat potus/* > lshare/potus.csv 윈도우wshare에서 볼 수 있음  
root@sandbox ~]# cd lshare  
root@sandbox lshare]# hadoop fs -get potus/* potus/ 윈도우wshare에서 볼 수 있음
```

Pig를 활용한 빅데이터 처리

4. Pig

```
root@sandbox ~]# cd lshare
root@sandbox lshare]# pig 4_wh_visits.pig
root@sandbox lshare]# hadoop fs -ls /user/hive/warehouse/wh_visits/
root@sandbox lshare]# hadoop fs -cat /user/hive/warehouse/wh_visits/*
```

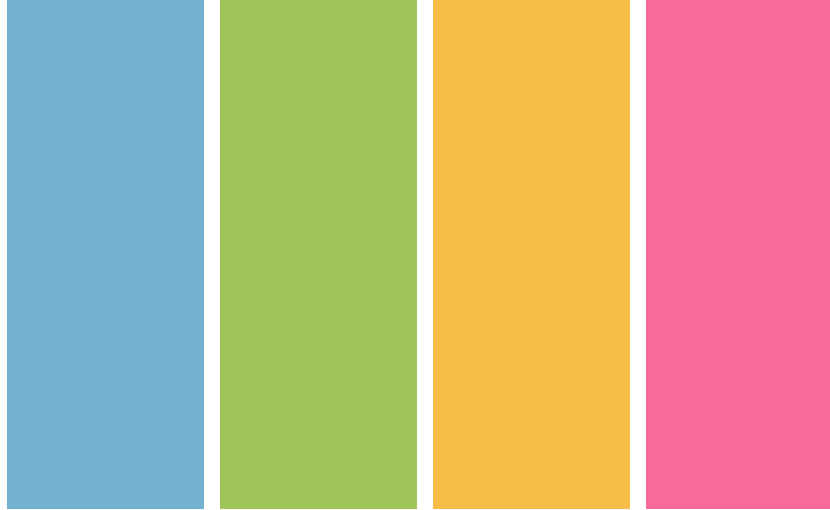
```
-- 4_wh_visits.pig : potus extract
```

```
visits = LOAD '/user/root/whitehouse/visits.txt' USING PigStorage(',');
```

```
potus = FILTER visits BY $19 MATCHES 'POTUS';
```

```
project_potus = FOREACH potus GENERATE
    $0 AS lname:chararray,
    $1 AS fname:chararray,
    $6 AS time_of_arrival:chararray,
    $11 AS appt_scheduled_time:chararray,
    $21 AS location:chararray,
    $25 as comment:chararray;
```

```
STORE project_potus INTO '/user/hive/warehouse/wh_visits/';
```



5. Hive

Hive를 활용한 빅데이터 처리

5. Hive

```
root@sandbox ~]# hadoop fs -cat /user/hive/warehouse/wh_visits/*
root@sandbox ~]# hive
hive> create table wh_visits (
  > lname string,
  > fname string,
  > time_of_arrival string,
  > appt_scheduled_time string,
  > meeting_location string,
  > info_comment string)
  > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t '
  > LOCATION '/user/hive/warehouse/wh_visits/' ;
hive> show tables;
hive> describe wh_visits;
hive> select * from wh_visits limit 20;
```

Table을 먼저 만들고 데이터를 나중에 넣어도 적용

5. Hive

```
hive> create table names (  
  > id int,  
  > name string)  
  > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t '  
  > LOCATION '/user/hive/warehouse/names/' ;
```

```
Hive> show tables;
```

```
Hive> describe names;
```

```
hive> select * from names limit 20;      names
```

```
root@sandbox ~]# hadoop fs -put lshare/6_names.txt /user/hive/warehouse/names/names.txt
```

```
root@sandbox ~]# hive
```

```
hive> select * from names;
```


Table을 먼저 만들고 데이터를 나중에 넣어도 적용

5. Hive

```
hive> drop table names;
root@sandbox ~]# hadoop fs -rm /user/hive/warehouse/names/names.txt
root@sandbox ~]# hadoop fs -put lshare/6_names.txt names.txt
root@sandbox ~]# hadoop fs -ls
root@sandbox ~]# hive -f /roor/lshare/6_create_names.hive
```

-- 6_create_names.hive

```
create external table names (      external 테이블은 drop table 하여도 데이터 남음
    id int,                        일반 테이블은 drop table을 하면 데이터, 폴더도 삭제
    name string)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t'
LOCATION '/user/hive/warehouse/names/';
```

```
hive> load data inpath '/user/root/names.txt' into table names;
hive> select * from names;
```

```
root@sandbox ~]# hadoop fs -ls /user/hive/warehouse/names/
```

Table을 먼저 만들고 데이터를 나중에 넣어도 적용

5. Hive

```
root@sandbox ~]# hive -f lshare/5_select_names.hive
```

```
-- 5_select_names.hive
```

```
select * from wh_visits  
  where time_of_arrival != ""  
  order by unix_timestamp(time_of_arrival, 'mm/dd/yyyy hh:mm')  
  limit 10;
```