

스파크를 이용한 데이터 이동



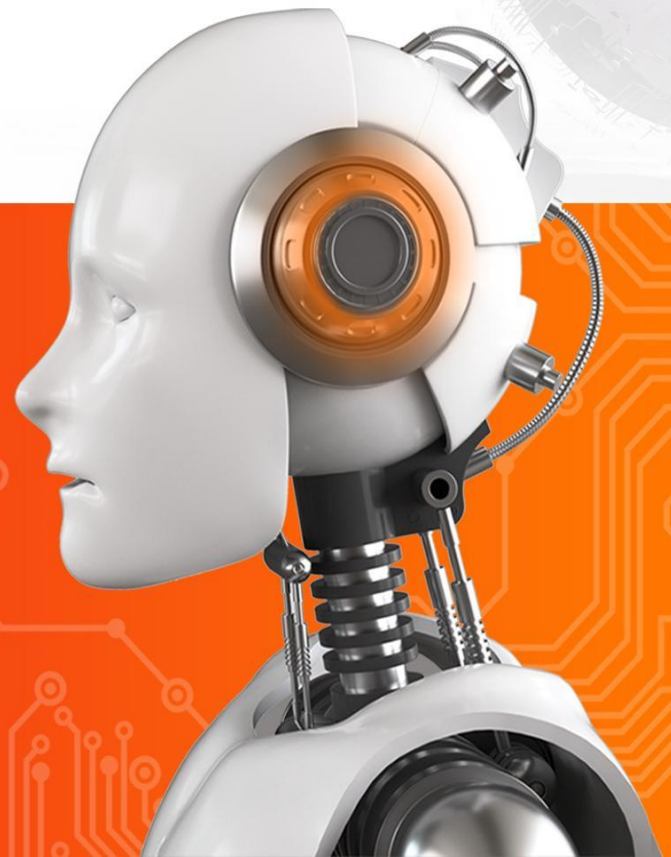


1.실시간 운행 정보 저장





1.1 실시간 운행 정보 생성



- 스마트 카가 없으므로 가상으로 스마트카의 운행정보를 생성하는 자바로 구현한 프로그램 실행

실시간 운행정보 SmartCarDriverInfo.log 파일의 내용

- SmartCarDriverInfo.log 파일의 내용은 다음과 같다

운전중 브레이크 강도
세게 누를수록 큰값

헤드라이트 상태
N(꺼짐),R(오른쪽 켜짐),L(왼쪽 켜짐)

```
Driver Status Infomation, CarNum, AccStep, BrkStep, WheelStep, DirLightStep, Speed, AreaNum
202001140000000, V0010, 5, 0, L2, L, 40, B08
202001140000002, M0005, 3, 0, F, N, 20, B01
Driver Status Infomation, CarNum, AccStep, BrkStep, WheelStep, DirLightStep, Speed, AreaNum
202001140000000, T0011, 1, 0, F, N, 5, B04
202001140000002, G0006, 4, 0, L3, L, 20, D07
202001140000004, M0001, 0, 1, F, N, 0, A10
202001140000002, V0007, 0, 0, L3, L, 5, B10
```

현재 운행중인 지역 번호

속도

실시간 운행
년도월일시분초
년도 4자리
월 2자리
일 2자리
시 2자리
분 2자리
초 2자리

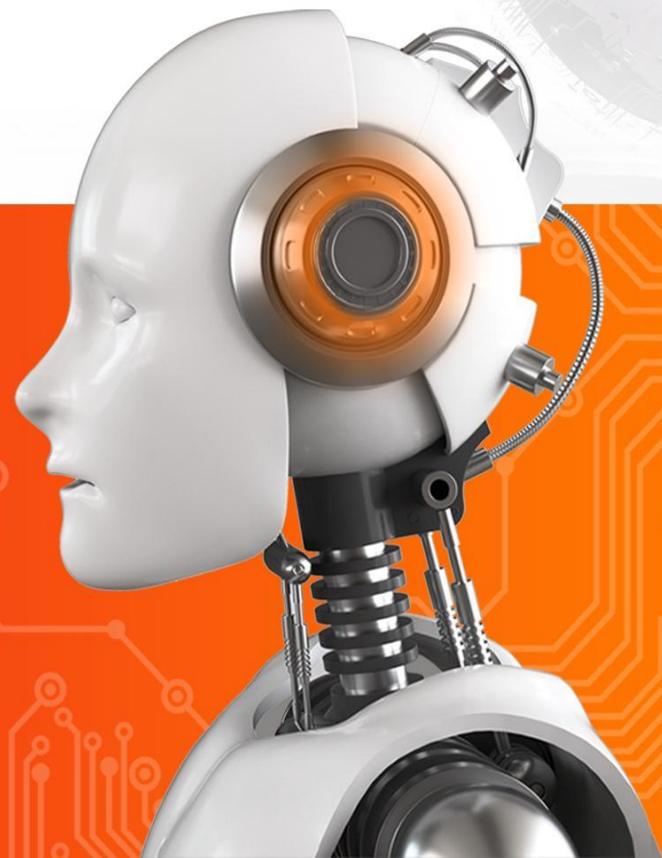
자동차
번호

운전중
가속 패달
강도
세게
누를수록
큰값

운전중 운전대 조작 강도
L(왼쪽), R(오른쪽), F(정지) 세게 조작할 수록 큰값



1.2 람다 함수



- 람다 표현식 사용하기

- 람다 표현식은 식 형태로 되어 있다고 해서 람다 표현식(lambda expression)이라고 부름
- 람다 표현식은 함수를 간편하게 작성할 수 있어서 다른 함수의 인수로 넣을 때 주로 사용함

람다 표현식으로 함수 만들기

- 람다 표현식으로 함수 만들기

```
>>> def plus_ten(x):  
...     return x + 10  
...  
>>> plus_ten(1)  
11
```

- 람다 표현식은 다음과 같이 `lambda`에 매개변수를 지정하고 `:`(콜론) 뒤에 반환값으로 사용할 식을 지정함

- `lambda` 매개변수들: 식

```
>>> lambda x: x + 10  
<function <lambda> at 0x02C27270>
```


람다 표현식으로 함수 만들기

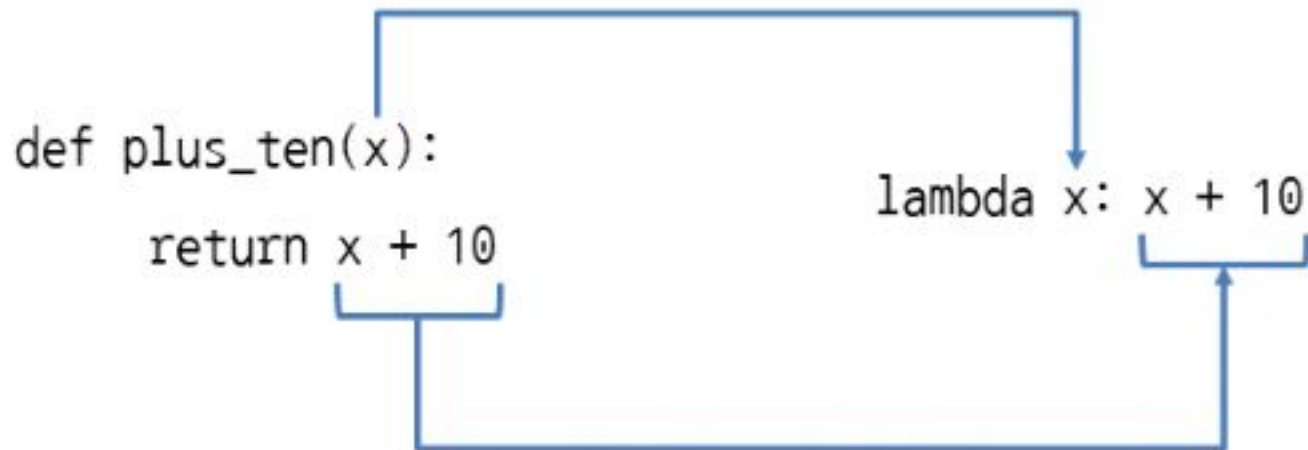
- 람다 표현식으로 함수 만들기

- 실행을 해보면 함수 객체가 나오는데, 이 상태로는 함수를 호출할 수 없음
- 람다 표현식은 이름이 없는 함수를 만들기 때문임
- 람다 표현식을 익명 함수(anonymous function)로 부르기도 함
- `lambda`로 만든 익명 함수를 호출하려면 다음과 같이 람다 표현식을 변수에 할당해주면 됨

```
>>> plus_ten = lambda x: x + 10
>>> plus_ten(1)
11
```

- 람다 표현식을 살펴보면 `lambda x: x + 10`은 매개변수 `x` 하나를 받고, `x`에 `10`을 더해서 반환한다는 뜻임
- 매개변수, 연산자, 값 등을 조합한 식으로 반환값을 만드는 방식임

▼ 그림 32-1 def로 만든 함수와 람다 표현식



람다 표현식으로 함수 만들기

- 람다 표현식 자체를 호출하기
 - 람다 표현식은 변수에 할당하지 않고 람다 표현식 자체를 바로 호출할 수 있음

• (lambda 매개변수들: 식)(인수들)

```
>>> (lambda x: x + 10)(1)
11
```

람다 표현식으로 함수 만들기

- 람다 표현식 안에서는 변수를 만들 수 없다
 - 반환값 부분은 변수 없이 식 한 줄로 표현할 수 있어야 함
 - 변수가 필요한 코드일 경우에는 **def**로 함수를 작성하는 것이 좋음

- ```
>>> (lambda x: y = 10; x + y)(1)
```

```
SyntaxError: invalid syntax
```

- 네름은 매개변수 x와 함수 표현식 바깥에 쓰인 변수 y를 너에서 건넌 것

```
>>> y = 10
>>> (lambda x: x + y)(1)
11
```

# 람다 표현식으로 함수 만들기

- 람다 표현식을 인수로 사용하기
  - 람다 표현식을 사용하기 전에 먼저 **def**로 함수를 만들어서 **map**을 사용해보자

```
>>> def plus_ten(x):
... return x + 10
...
>>> list(map(plus_ten, [1, 2, 3]))
[11, 12, 13]
```

- 이제 람다 표현식으로 함수를 만들어서 **map**에 넣어보자

```
>>> list(map(lambda x: x + 10, [1, 2, 3]))
[11, 12, 13]
```

# 람다 표현식과 map, filter, reduce 함수 활용하기

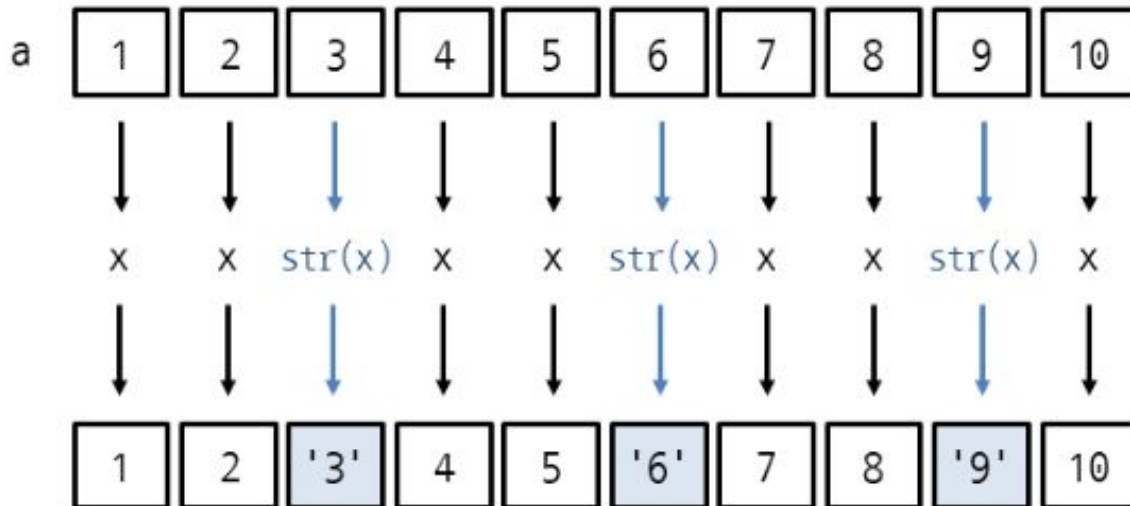
- 람다 표현식에 조건부 표현식 사용하기
  - 다음은 map을 사용하여 리스트 a에서 3의 배수를 문자열로 변환함
    - lambda 매개변수들: 식1 if 조건식 else 식2

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(map(lambda x: str(x) if x % 3 == 0 else x, a))
[1, 2, '3', 4, 5, '6', 7, 8, '9', 10]
```

- map은 리스트의 요소를 각각 처리하므로 lambda의 반환값도 요소라야 함
- 여기서는 요소가 3의 배수일 때는 str(x)로 요소를 문자열로 만들어서 반환했고, 3의 배수가 아닐 때는 x로 요소를 그대로 반환함

## ▼ 그림 32-2 map에 람다 표현식 사용하기

```
list(map(lambda x: str(x) if x % 3 == 0 else x, a))
```





# 람다 표현식과 map, filter, reduce 함수 활용하기

- 람다 표현식에 조건부 표현식 사용하기
  - 람다 표현식 안에서 조건부 표현식 **if, else**를 사용할 때는 **:(콜론)**을 붙이지 않음
  - **if, else**와 문법이 다르므로 주의해야 함
  - 조건부 표현식은 식1 **if** 조건식 **else** 식2 형식으로 사용하며 식1은 조건식이 참일 때, 식2는 조건식이 거짓일 때 사용할 식임
  - 특히 람다 표현식에서 **if**를 사용했다면 반드시 **else**를 사용해야 함
  - 다음과 같이 **if**만 사용하면 문법 에러가 발생하므로 주의해야 함

```
>>> list(map(lambda x: str(x) if x % 3 == 0, a))
SyntaxError: invalid syntax
```

# 람다 표현식과 map, filter, reduce 함수 활용하기

- 람다 표현식에 조건부 표현식 사용하기
  - 람다 표현식 안에서는 `elif`를 사용할 수 없음
  - 조건부 표현식은 식1 `if` 조건식1 `else` 식2 `if` 조건식2 `else` 식3 형식처럼 `if`를 연속으로 사용해야 함

• `lambda` 매개변수들: 식1 `if` 조건식1 `else` 식2 `if` 조건식2 `else` 식3

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(map(lambda x: str(x) if x == 1 else float(x) if x == 2 else x + 10, a))
['1', 2.0, 13, 14, 15, 16, 17, 18, 19, 20]
```

- 람다 표현식에 조건부 표현식 사용하기
  - 별로 복잡하지 않은 조건인데도 알아보기가 힘든 경우에는 억지로 람다 표현식을 사용하기 보다는 그냥 **def**로 함수를 만들고 **if**, **elif**, **else**를 사용하는 것을 권장함

```
>>> def f(x):
... if x == 1:
... return str(x)
... elif x == 2:
... return float(x)
... else:
... return x + 10
...
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(map(f, a))
['1', 2.0, 13, 14, 15, 16, 17, 18, 19, 20]
```

- map에 객체를 여러 개 넣기
  - 다음은 두 리스트의 요소를 곱해서 새 리스트를 만듦

```
>>> a = [1, 2, 3, 4, 5]
>>> b = [2, 4, 6, 8, 10]
>>> list(map(lambda x, y: x * y, a, b))
[2, 8, 18, 32, 50]
```

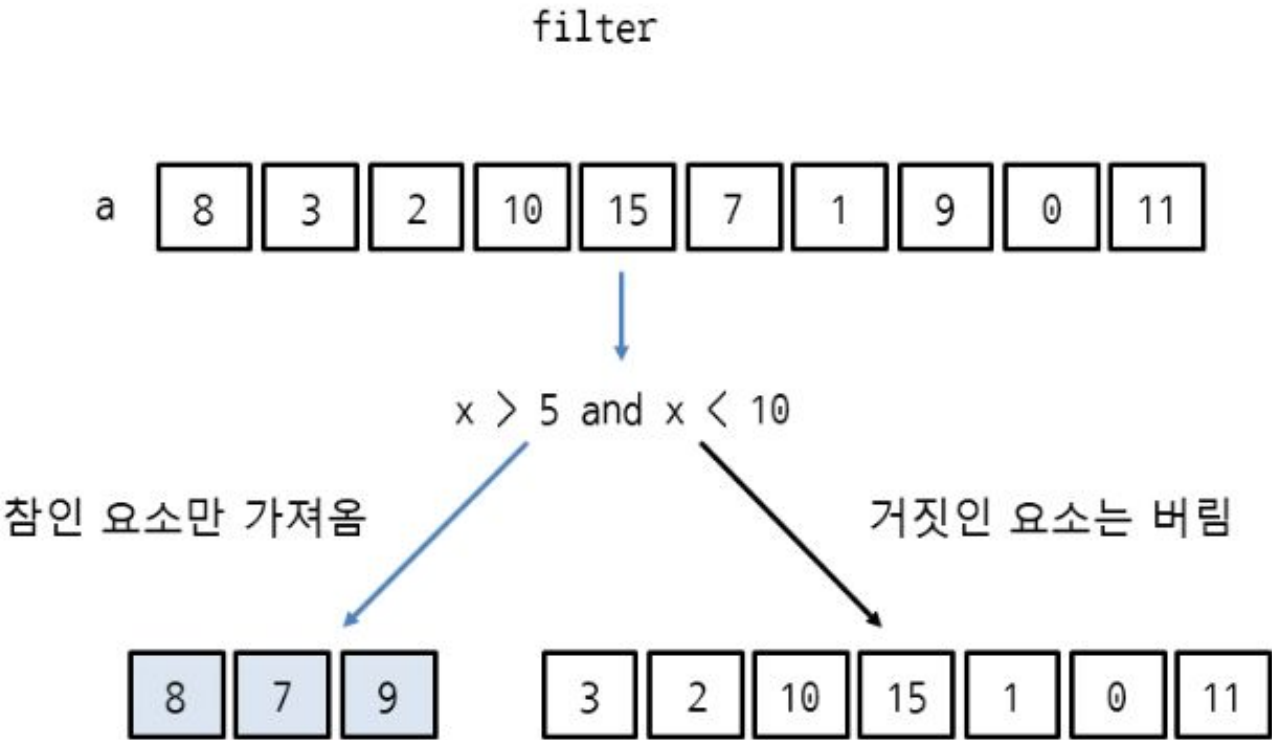
- filter 사용하기

- filter는 반복 가능한 객체에서 특정 조건에 맞는 요소만 가져오는데, filter에 지정한 함수의 반환값이 True일 때만 해당 요소를 가져옴

- filter(함수, 반복가능한객체)

```
>>> def f(x):
... return x > 5 and x < 10
...
>>> a = [8, 3, 2, 10, 15, 7, 1, 9, 0, 11]
>>> list(filter(f, a))
[8, 7, 9]
```

▼ 그림 32-3 filter 함수





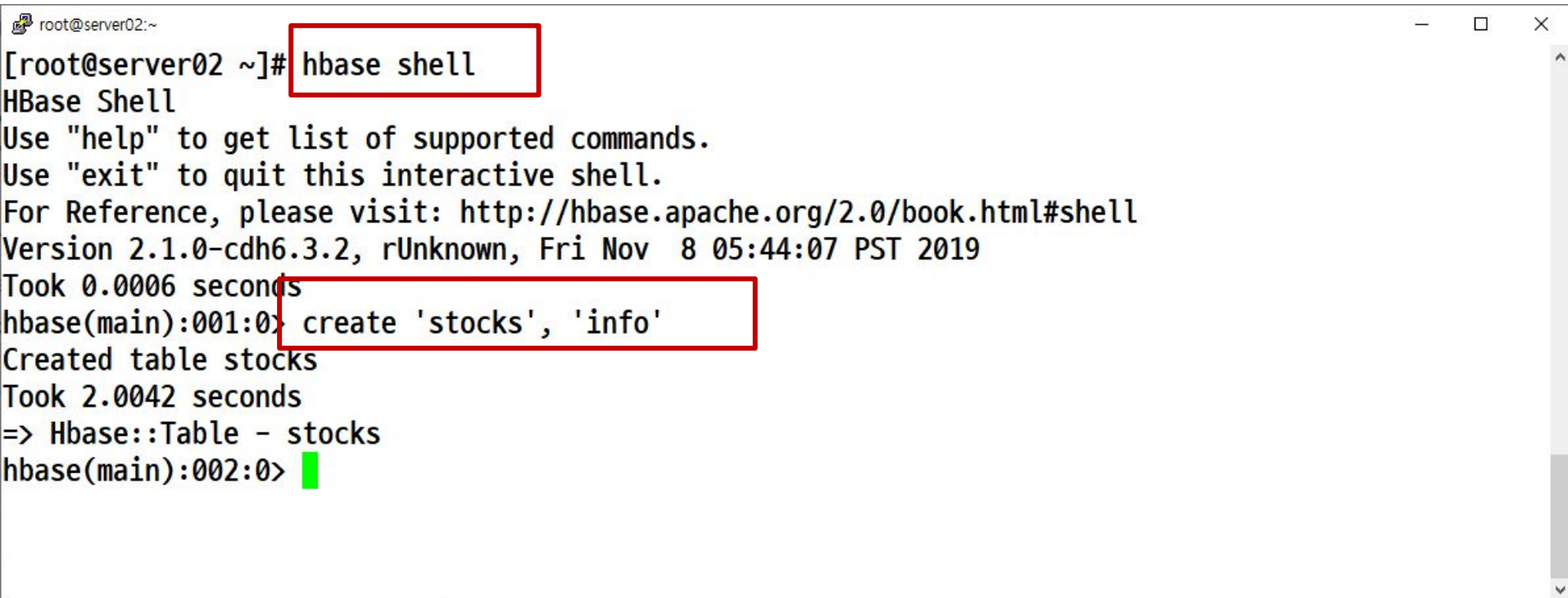
## 1.3 파이썬을 이용한 Hbase 사용





## Hbase 테이블 생성

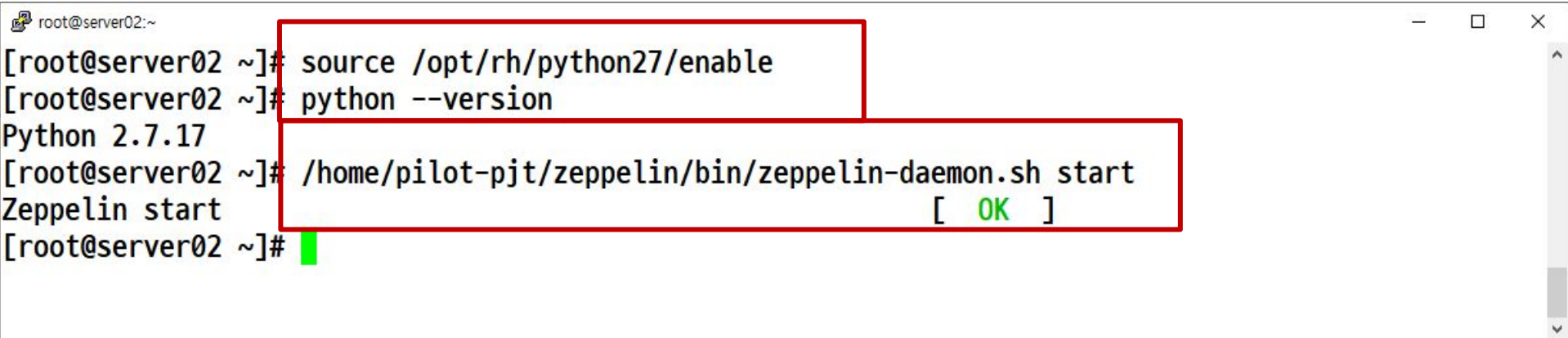
- server02 에 root 계정으로 접속 한 후 hbase 에서 다음과 같이 입력한다



```
root@server02:~
[root@server02 ~]# hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.1.0-cdh6.3.2, rUnknown, Fri Nov 8 05:44:07 PST 2019
Took 0.0006 seconds
hbase(main):001:0> create 'stocks', 'info'
Created table stocks
Took 2.0042 seconds
=> Hbase::Table - stocks
hbase(main):002:0>
```

# Spark 개발환경 실행

- server02 에서 다음과 같이 입력한다



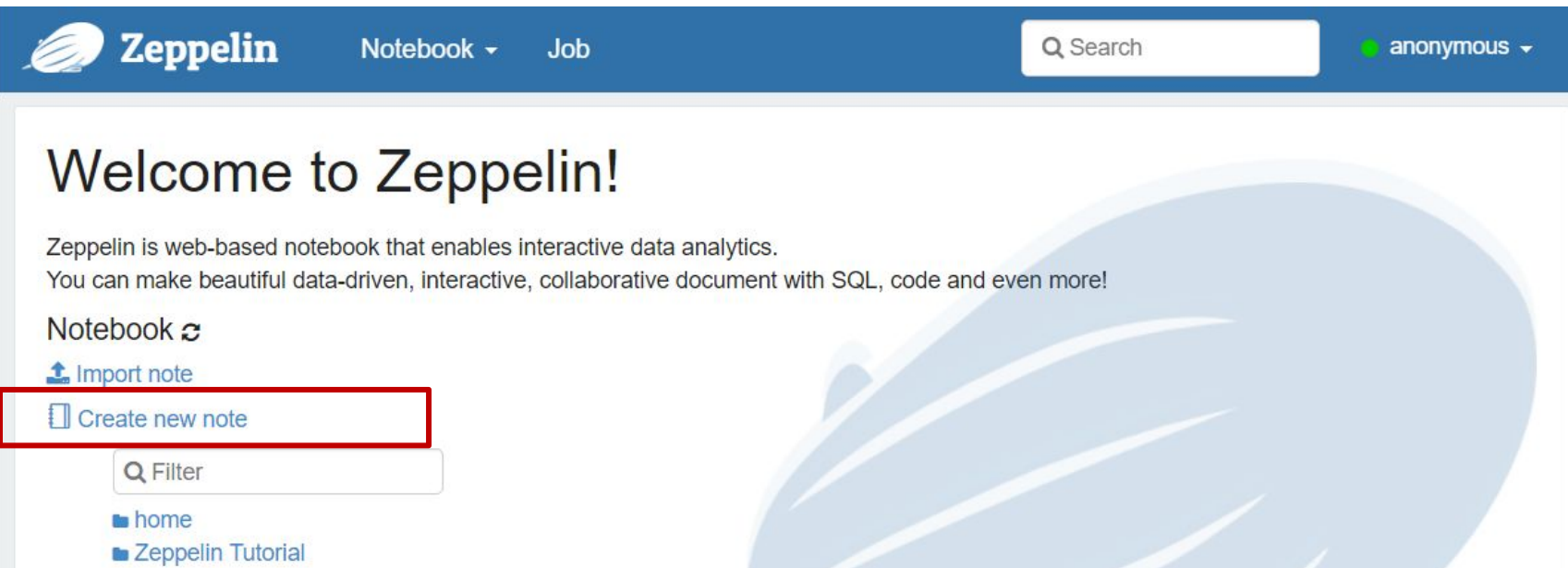
```
root@server02:~
[root@server02 ~]# source /opt/rh/python27/enable
[root@server02 ~]# python --version
Python 2.7.17
[root@server02 ~]# /home/pilot-pjt/zeppelin/bin/zeppelin-daemon.sh start
Zeppelin start [OK]
[root@server02 ~]#
```

## Spark hbase 연동 라이브러리 설치

```
root@server02:~
[root@server02 ~]# /home/pilot-pjt/zeppelin/bin/zeppelin-daemon.sh start
Zeppelin start [OK]
[root@server02 ~]# pip install happybase
Requirement already satisfied (use --upgrade to upgrade): happybase in /opt/rh/python27/root/usr/lib/python2.7/site-packages
Requirement already satisfied (use --upgrade to upgrade): thriftpy2>=0.4 in /opt/rh/python27/root/usr/lib64/python2.7/site-packages (from happybase)
Requirement already satisfied (use --upgrade to upgrade): six in /opt/rh/python27/root/usr/lib/python2.7/site-packages (from happybase)
Requirement already satisfied (use --upgrade to upgrade): ply<4.0,>=3.4 in /opt/rh/python27/root/usr/lib/python2.7/site-packages (from thriftpy2>=0.4->happybase)
You are using pip version 8.1.2, however version 21.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
[root@server02 ~]#
```

# HBase 접속 Spark 프로그래밍

- <http://server02.hadoop.com:8081/> 실행
- create new notebook 클릭
- 파일명은 hbase\_01 을 입력한다



# HBase 접속 Spark 프로그래밍

- 다음의 내용을 입력한다

```
%spark.pyspark
import time
import happybase
```

Took 0 sec. Last updated by anonymous at April 21 2021, 4:46:50 AM. (outdated)

Shift+Enter 실행

```
%spark.pyspark
HBase 의 버전
CDH6_HBASE_THRIFT_VER='0.92'

connect = happybase.Connection(
 host='server02', # HBASE 저장된 서버
 port=9090, # HBASE 포트
 table_prefix=None,
 compat=CDH6_HBASE_THRIFT_VER, # HBASE 버전
 timeout=None,
 autoconnect=True,
 transport='framed',
 protocol='compact'
)
```

Took 0 sec. Last updated by anonymous at April 21 2021, 4:46:50 AM. (outdated)

Shift+Enter 실행

```
%spark.pyspark
connect.tables() # HBASE 에 저장된 테이블이 조회되는지 확인
```

Shift+Enter 실행

```
['DriverCarInfo', 'books', 'stocks', 'test', 'testtable']
```

HBASE 에 저장된 테이블 리스트가 출력 되는지 확인

## HBASE stock 테이블에 데이터 추가

```
%spark.pyspark
stocks 테이블에 primary key r1, 컬럼명 c1 데이터 v11 추가
table.put("r1", {"info:c1": "v11"})
stocks 테이블에 primary key r1, 컬럼명 c2 데이터 v12 추가
table.put("r1", {"info:c2": "v12"})
stocks 테이블에 primary key r2, 컬럼명 c1 데이터 v21 추가
table.put("r2", {"info:c1": "v21"})
stocks 테이블에 primary key r2, 컬럼명 c2 데이터 v22 추가
table.put("r2", {"info:c2": "v22"})
```

**Shift+Enter 실행**



## HBASE stock 테이블에 추가 확인

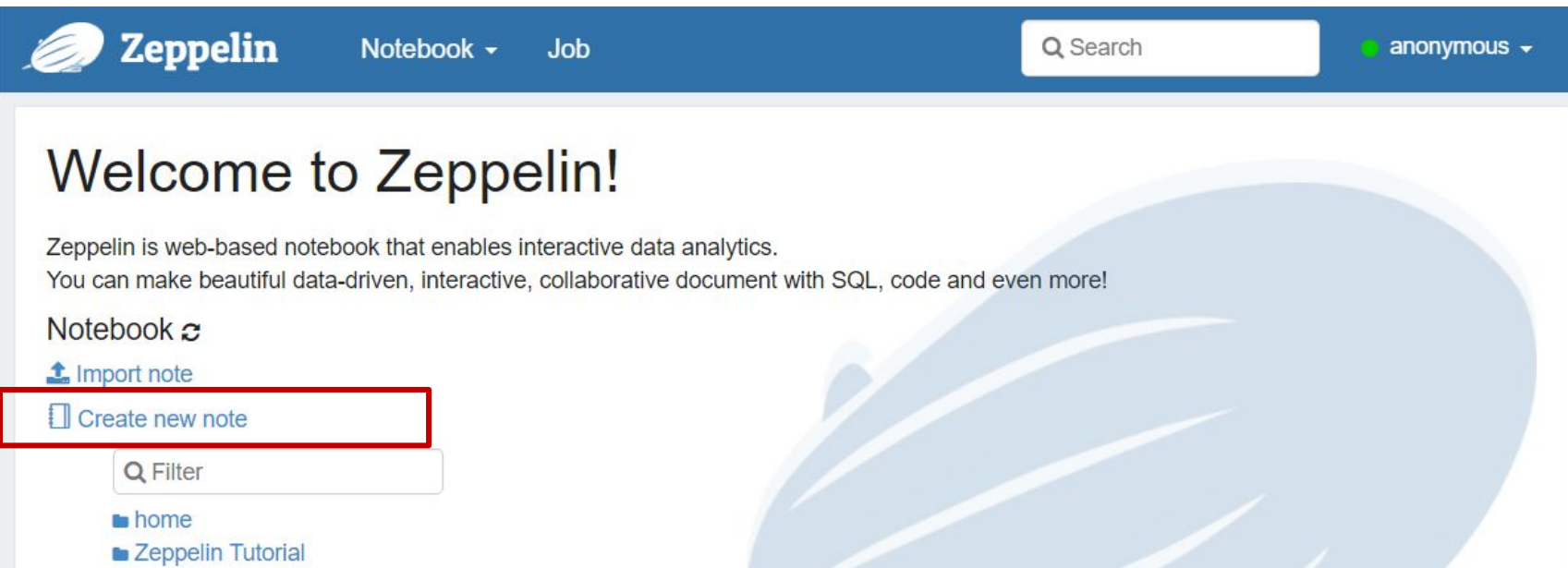
```
root@server02:~
[root@server02 ~]# hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.1.0-cdh6.3.2, rUnknown, Fri Nov 8 05:44:07 PST 2019
Took 0.0137 seconds
hbase(main):001:0> scan 'stocks'
ROW COLUMN+CELL
r1 column=info:c1, timestamp=1619044061573, value=v11
r1 column=info:c2, timestamp=1619044061592, value=v12
r2 column=info:c1, timestamp=1619044061602, value=v21
r2 column=info:c2, timestamp=1619044061616, value=v22
2 row(s)
Took 0.7927 seconds
hbase(main):002:0>
```

stocks 테이블에 데이터가 추가 되었는지 확인



# Spark 를 이용한 Hbase 테이블 레코드 조회

- <http://server02.hadoop.com:8081/> 실행
- create new notebook 클릭
- 파일명은 hbase\_02 을 입력한다



# Spark 를 이용한 Hbase 테이블 레코드 조회

```
%spark.pyspark
import time
import happybase
```

Took 0 sec. Last updated by anonymous at April 22 2021, 7:48:25 AM.

```
%spark.pyspark
HBase 의 버전
CDH6_HBASE_THRIFT_VER='0.92'

connect = happybase.Connection(
 host='server02', # HBASE 저장된 서버
 port=9090, # HBASE 포트
 table_prefix=None,
 compat=CDH6_HBASE_THRIFT_VER, # HBASE 버전
 timeout=None,
 autoconnect=True,
 transport='framed',
 protocol='compact'
)
```

## Spark 를 이용한 Hbase 테이블 레코드 조회

```
%spark.pyspark
import time
import happybase
```

Took 0 sec. Last updated by anonymous at April 22 2021, 7:48:25 AM.

```
%spark.pyspark
HBase 의 버전
CDH6_HBASE_THRIFT_VER='0.92'

connect = happybase.Connection(
 host='server02', # HBASE 저장된 서버
 port=9090, # HBASE 포트
 table_prefix=None,
 compat=CDH6_HBASE_THRIFT_VER, # HBASE 버전
 timeout=None,
 autoconnect=True,
 transport='framed',
 protocol='compact'
)
```

## Spark 를 이용한 Hbase 테이블 레코드 조회

```
%spark.pyspark
stocks 테이블에 접속
table = connect.table('stocks')
table

<happybase.table.Table name='stocks'>
```

Took 0 sec. Last updated by anonymous at April 22 2021, 7:48:31 AM.

```
%spark.pyspark
table.scan() : stocks 테이블의 모든 데이터 조회해서 primary 키와 컬럼값 리턴
key : stocks 테이블의 primary key 를 저장할 변수
data : stocks 테이블의 컬럼값을 저장할 변수
for key, data in table.scan():
 print("pk=",key," columns=", data)

('pk=', 'r1', ' columns=', {'info:c1': 'v11', 'info:c2': 'v12'})
('pk=', 'r2', ' columns=', {'info:c1': 'v21', 'info:c2': 'v22'})
```

**stocks 테이블의 레코드가 조회되는지 확인**

## 프로그램의 종료

- 스파크 스트림은 백그라운드 작업 ( 눈에 보이지 않게 무한 루프로 실행) 으로 데이터를 옮기기 때문에 개발환경의 실행이 끝나도 눈에 안보이게 무한루프가 실행 중임
- 프로그램을 종료 하기 위해서는 강제로 종료 해야함

The screenshot shows the Zeppelin Notebook interface in a web browser. The browser tab is titled 'hbase01 - Zeppelin'. The address bar shows the URL '192.168.56.102:8081/#/notebook/2G5UZKAHY'. The Zeppelin logo and 'Notebook' tab are visible in the header. A search bar and a user profile dropdown (currently showing 'anonymous') are on the right. Below the header, the notebook title 'hbase01' is displayed with various action icons. A dropdown menu is open from the 'anonymous' user profile, with the 'Interpreter' option highlighted by a red box. Other options in the menu include 'About Zeppelin', 'Notebook Repos', 'Credential', 'Helium', and 'Configuration'. The main content area shows a code block with the following text:

```
%spark.pyspark
connect.tables() # HBASE 에 저장된 테이블이 조회되는지 확인

['DriverCarInfo', 'books', 'stocks', 'test', 'testtable']
```

Below the code, it says 'Took 1 sec. Last updated by anonymous at April 22 2021, 7:27:40 AM.'

The screenshot shows the Zeppelin web interface. The browser tab is labeled 'Zeppelin' and the address bar shows '192.168.56.102:8081/#/interpreter'. The Zeppelin logo and navigation links 'Notebook' and 'Job' are in the top blue bar. A search bar and a user profile 'anonymous' are also present. The main section is titled 'Interpreters' and includes a 'Repository' button and a '+ Create' button. Below this, a text input field contains 'spark' and is highlighted with a red box. A search icon is to its right. Underneath, the 'spark' interpreter is listed with its configuration: '%spark, %sql, %dep, %pyspark, %ipyspark, %r'. To the right of this configuration are four buttons: 'spark ui', 'edit', 'restart' (highlighted with a red box), and 'remove'. Below the configuration, the word 'Option' is followed by a description: 'The interpreter will be instantiated' with dropdown menus for 'Globally' and 'in shared', and the word 'process' with an information icon.

Zeppelin Notebook Job Search anonymous

## Interpreters

Repository + Create

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

spark

spark %spark, %sql, %dep, %pyspark, %ipyspark, %r

spark ui edit restart remove

Option

The interpreter will be instantiated Globally in shared process

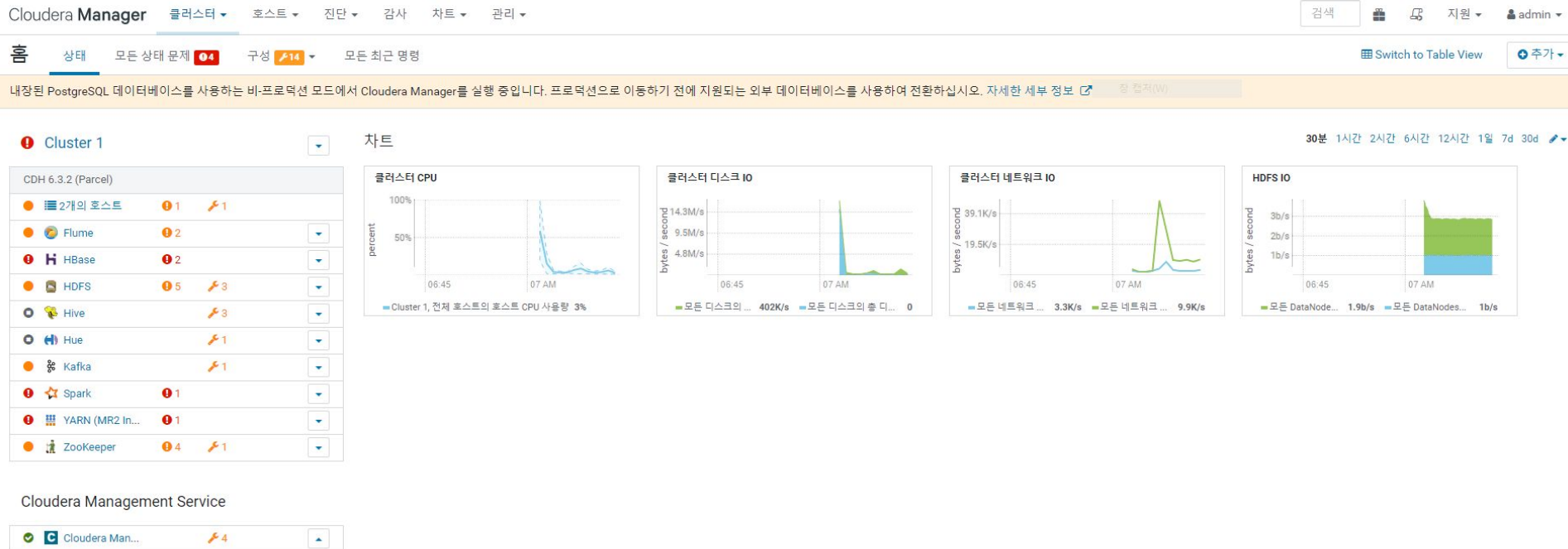
# Apache Spark Streaming

- **Kafka** 에 저장된 데이터를 읽어 오거나 다른곳으로 옮기는 작업
- 저장된 데이터를 다른 곳으로 옮기는것

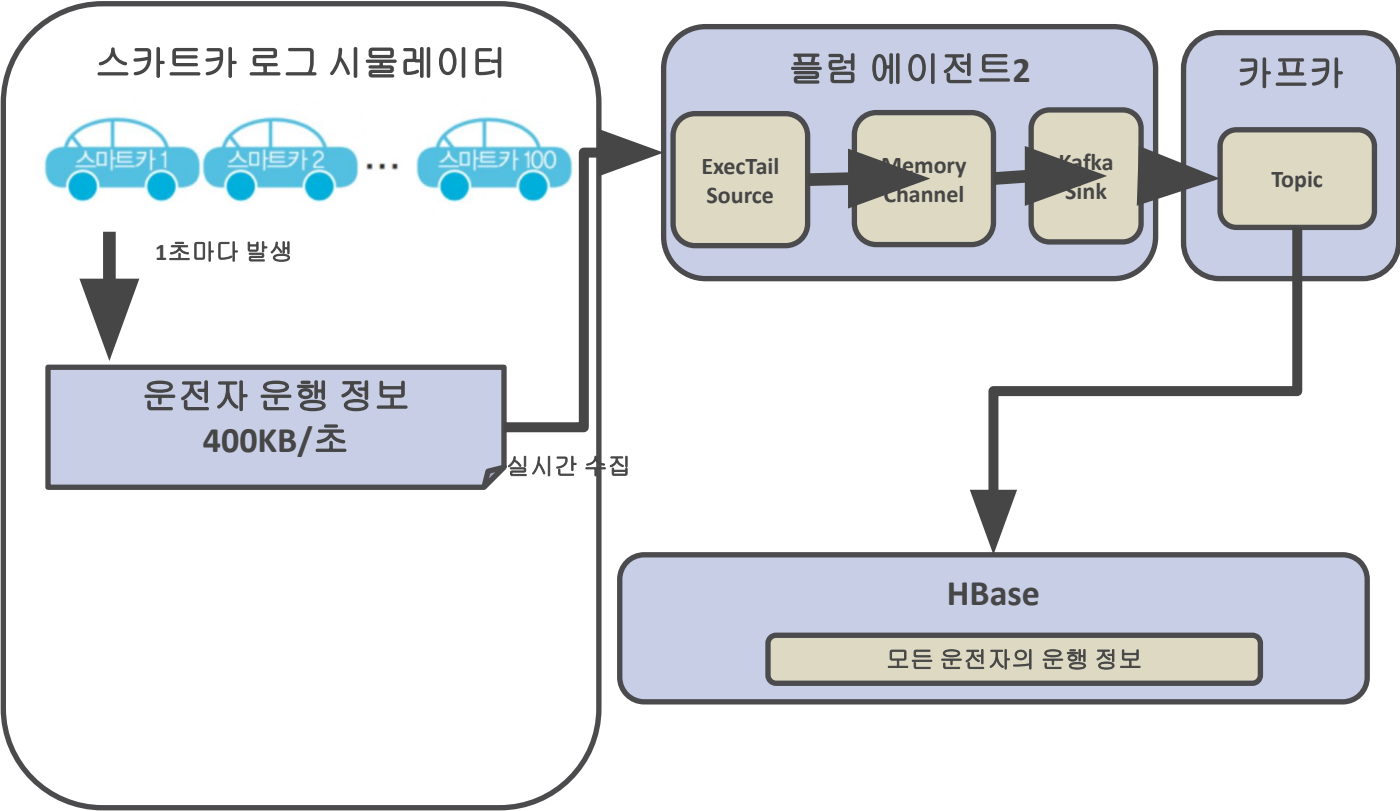


# Kafka 에 저장된 데이터 읽어 오기

- <http://server01.hadoop.com:7180>
- Flume, HDFS, Kafka, Spark, Yarn, Zookeeper 만 시작하고 나머지 서비스는 중지



# 실시간 적재 환경 구성 (1)



- 스마트 카의 로그 정보를 파일로 저장하는 프로그램
- **Server02**에서 실행
- **Server02**에 **root**계정으로 로그인

- 실시간으로 차량의 운행정보를 가짜로 만드는 프로그램
- `com.wikibook.bigdata.smartcar.loggen` 패키지
- `/home/pilot-pjt/working/bigdata.smartcar.loggen-1.0.jar` 에 포함되 있음

## DriverLogMain 실행 (2)

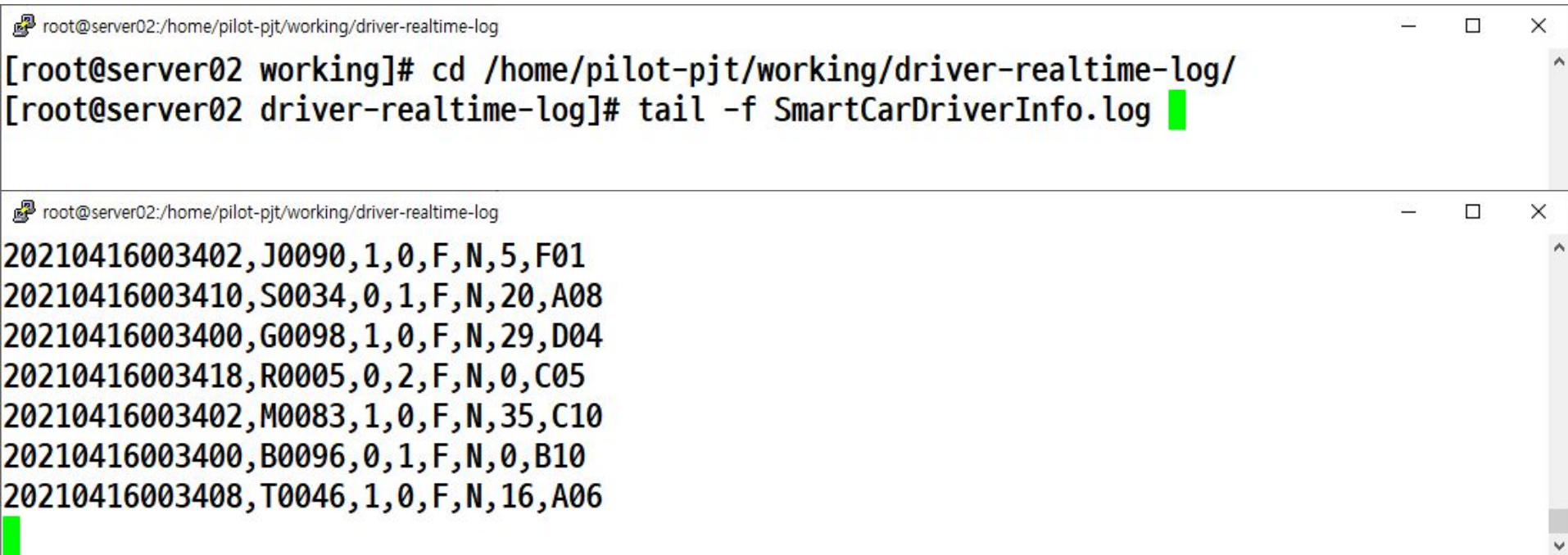
- 터미널 창에서 다음과 같이 DriverLogMain 실행
- 20210416 은 날짜
- 100은 로그 정보를 수집할 자동차의 수
- 100대의 자동차로 부터 2021년 4월 16일의 실시간 운행 정보 가짜로 생성

```
root@server02:/home/pilot-pjt/working
[root@server02 ~]# cd /home/pilot-pjt/working/
[root@server02 working]# java -cp bigdata.smartcar.loggen-1.0.jar com.wikibook.bigdata.smartcar.loggen.DriverLogMain 20210416 100
```

무한루프를 도는 프로그램이므로  
약간의 시간이 지난후에 ctrl+c 를 눌러서 프로그램 종료

## DriverLogMain 실행 (3)

- 가짜로 생성된 자동차 위치정보를 출력한다



```
root@server02:/home/pilot-pjt/working/driver-realtime-log
[root@server02 working]# cd /home/pilot-pjt/working/driver-realtime-log/
[root@server02 driver-realtime-log]# tail -f SmartCarDriverInfo.log

root@server02:/home/pilot-pjt/working/driver-realtime-log
20210416003402,J0090,1,0,F,N,5,F01
20210416003410,S0034,0,1,F,N,20,A08
20210416003400,G0098,1,0,F,N,29,D04
20210416003418,R0005,0,2,F,N,0,C05
20210416003402,M0083,1,0,F,N,35,C10
20210416003400,B0096,0,1,F,N,0,B10
20210416003408,T0046,1,0,F,N,16,A06
```

위치정보의 의미는 교재 32페이지 표2.2의 데이터 스키마를 참조

- 실시간 운행정보를 수집해서 카프카 SmartCar-Topic 창고에 저장되도록 플럼 설정

<http://server01.hadoop.com:7180> 접속

The screenshot shows the Cloudera Manager web interface. The top navigation bar includes '홈', '상태', '모든 상태 문제', '구성', and '모든 최근 명령'. The main content area displays 'Cluster 1' with a list of components: CDH 6.3.2 (Parcel), 2개의 호스트, Flume, HDFS, Kafka, YARN (MR2 In...), and ZooKeeper. The '구성' button is highlighted with a red box. The '플럼' component is also highlighted with a red box. The right side of the interface shows three charts: '클러스터 CPU', '클러스터 디스크 IO', and '클러스터 네트워크 IO'. The '클러스터 CPU' chart shows a usage of 2.4% for Cluster 1. The '클러스터 디스크 IO' chart shows a peak of 977K/s. The '클러스터 네트워크 IO' chart shows a peak of 9.8K/s.

# 플럼 설정

구성 - Flume - Cloudera Manag x +

← → ↻ ⚠ 주의 요함 | 192.168.56.101:7180/cmf/services/10/config

flume

Agent 이름

Agent Default Group ↗ ?

SmartCar\_Agent

구성 파일

Agent Default Group ↗ ?

timestamp  
SmartCar\_Agent.sources.SmartCarInfo\_SpoolSource.interceptors.timeInterceptor.preserveExisting = true

Flume 홈 디렉토리

Agent Default Group ?

/var/lib/flume-ng

변경 내용 저장

기존 내용을 모두 삭제하고

[https://drive.google.com/drive/folders/1aD-8RscE5pyovLBZfL40TE1pjYhu5C9\\_?usp=sharing](https://drive.google.com/drive/folders/1aD-8RscE5pyovLBZfL40TE1pjYhu5C9_?usp=sharing) 의 SmartCar\_Agent\_Realtime.txt 를 더블클릭 후 내용을 복사해서 붙여 넣기



# 플럼 설정

구성 - Flume - Cloudera Manag x +

← → ↺ 주의 요함 | 192.168.56.101:7180/cmf/services/10/config

flume

Agent 이름 Agent Default Group SmartCar\_Agent

구성 파일 Agent Default Group

```
SmartCar_Agent.sources.DriverCarInfo_TailSource.channels = DriverCarInfo_Channel
SmartCar_Agent.sinks.DriverCarInfo_KafkaSink.channel = DriverCarInfo_Channel
```

Flume 홈 디렉토리 Agent Default Group /var/lib/flume-ng

플러그인 디렉토리 Agent Default Group

1 Edited Value 변경 이유

변경 내용 저장

- 실시간 운행정보를 수집해서 카프카 SmartCar-Topic 창고에 저장되도록 플럼 설정

The screenshot shows the Cloudera Manager web interface. At the top, there's a navigation bar with 'Cloudera Manager' and various tabs like '클러스터', '호스트', '진단', '감사', '차트', and '관리'. Below this, there's a summary section for 'Cluster 1' showing its status and configuration. The 'Flume' service is highlighted with a red box, indicating it's the focus of the setup. Below the service list, there are several monitoring charts: '클러스터 CPU' (Cluster CPU), '클러스터 디스크 IO' (Cluster Disk IO), '클러스터 네트워크 IO' (Cluster Network IO), and 'HDFS IO'. The 'Flume' service is currently in a 'Stopped' state, as indicated by the red 'X' icon next to it.

Cluster 1

CDH 6.3.2 (Parcel)

- 2개의 호스트
- Flume
- HBase
- HDFS
- Kafka
- YARN (MR2 In...)
- ZooKeeper

Cloudera Management Service

- Cloudera Man...

차트

클러스터 CPU

클러스터 디스크 IO

클러스터 네트워크 IO

HDFS IO

# 플럼 설정

오래된 구성 - Cloudera Manager x +

← → ↺ 주의 요함 | 192.168.56.101:7180/cm/clusters/1/staleness/view?service=flume

Cloudera Manager 클러스터 호스트 진단 감사 차트 관리

Cluster 1  
오래된 구성

필터 모두 지우기

- ▼ 파일  
파일: flume.conf 1
- ▼ 서비스 지우기  
Flume 1
- ▼ 역할 유형  
Agent 1

파일: flume.conf flume(1) 표시

```
... @@ -1,29 +1,18 @@
1 -SmartCar_Agent.sources=SmartCarInfo_SpoolSource
2 -SmartCar_Agent.channels=SmartCarInfo_Channel
3 -SmartCar_Agent.sinks=SmartCarInfo_HdfsSink
4 -SmartCar_Agent.sources.SmartCarInfo_SpoolSource.type=spooldir
5 -SmartCar_Agent.sources.SmartCarInfo_SpoolSource.spoolDir=/home/pilot-pjt/working/car-batch-log
6 -SmartCar_Agent.sources.SmartCarInfo_SpoolSource.deletePolicy=immediate
7 -SmartCar_Agent.sources.SmartCarInfo_SpoolSource.batchSize=1000
8 -SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors=timeInterceptor
9 -SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.timeInterceptor.type=timestamp
10 -SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.timeInterceptor.preserveExisting=true
11 -SmartCar_Agent.channels.SmartCarInfo_Channel.type=memory
12 -SmartCar_Agent.channels.SmartCarInfo_Channel.capacity=100000
13 -SmartCar_Agent.channels.SmartCarInfo_Channel.transactionCapacity=10000
14 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.type=hdfs
15 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.path=/pilot-pjt/collect/car-batch-log/wrk_date=%Y%m%d
16 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.filePrefix=car-batch-log
17 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.fileSuffix=.log
18 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.fileType=DataStream
19 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.writeFormat=Text
20 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.batchSize=10000
21 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.rollInterval=0
22 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.rollCount=0
23 -SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.idleTimeout=100
```

클러스터 새로고침

Cloudera Manager

← → ↺ 주의 요함 | 192.168.56.101:7180/cm/clusters/1/staleness/restartWizard?returnUrl=%2Fcmf%2F#step=commandDetailsStep

Cloudera Manager

지원 admin

클러스터 새로고침

명령 세부 정보

클러스터 새로고침 명령

상태 실행 중 컨텍스트 Cluster 1 Apr 16, 7:17:51 AM 중단

0/1 단계가 완료되었습니다.

Show All Steps

Show Only Failed Steps

Show Only Running Steps

Update all refreshable configuration files in the cluster. Will not restart any roles.

0/2 단계가 완료되었습니다.

Apr 16, 7:17:51 AM

뒤로

완료

Cloudera Manager

← → ↻ 주의 요함 | 192.168.56.101:7180/cm/clusters/1/staleness/restartWizard?returnUrl=%2Fcmf%2F#step=commandDetailsStep

Cloudera Manager

지원 admin

클러스터 새로고침

명령 세부 정보

클러스터 새로고침 명령

상태 완료됨 컨텍스트 Cluster 1 Apr 16, 7:17:51 AM 37.13s

Successfully refreshed roles in the cluster.

1/1 단계가 완료되었습니다.

Show All Steps

Show Only Failed Steps

Show Only Running Steps

Update all refreshable configuration files in the cluster. Will not restart any roles.

Apr 16, 7:17:51 AM 37.13s

뒤로

완료

# 플럼 설정

<http://server01.hadoop.com:7180> 접속

Cloudera Manager   클러스터   호스트   진단   감사   차트   관리

홈   상태   모든 상태 문제 3   구성 10   모든 최근 명령

내장된 PostgreSQL 데이터베이스를 사용하는 비-프로덕션 모드에서 Cloudera Manager를 실행 중입니다. 프로덕션으로 0

Cluster 1

차트

CDH 6.3.2 (Parcel)

- 2개의 호스트 1 1
- Flume
- HBase 1 1
- HDFS 1 1
- Kafka
- YARN (MR2 In...
- ZooKeeper

Flume개 작업

시작

중지

재시작

롤링 재시작

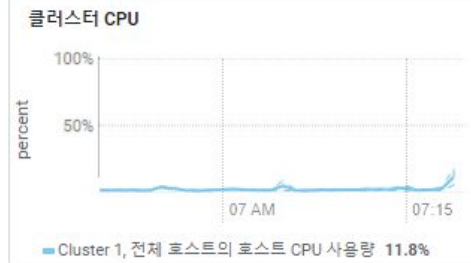
인스턴스

구성

역할 인스턴스 추가

이름 바꾸기

삭제



재시작



**Flume** 서비스에 재시작 명령을 실행하시겠습니까?

취소


재시작

재시작 ✕

상태 실행 중 컨텍스트 [Flume](#)   Apr 16, 7:21:51 AM 중단

▼ 0/2단계가 완료되었습니다.

☒ Show All Steps ☐ Show Only Failed Steps ☐ Show Only Running Steps

|                                                                                                                                                 |                    |                 |
|-------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|-----------------|
| ▶ <span>실행 중</span> 서비스 Flume에서 명령 중지 실행 <span>Flume</span>  | Apr 16, 7:21:51 AM | <span>중단</span> |
| ○ 서비스 Flume에서 명령 시작 실행                                                                                                                          |                    |                 |

중단 닫기



재시작

✕

상태 완료됨 컨텍스트 [Flume](#) Apr 16, 7:21:51 AM 27.95s

Successfully restarted service.

▼ 2/2단계가 완료되었습니다.

☒ Show All Steps    ☐ Show Only Failed Steps    ☐ Show Only Running Steps

|                         |                       |                    |        |
|-------------------------|-----------------------|--------------------|--------|
| >  서비스 Flume에서 명령 중지 실행 | <a href="#">Flume</a> | Apr 16, 7:21:51 AM | 3.96s  |
| >  서비스 Flume에서 명령 시작 실행 | <a href="#">Flume</a> | Apr 16, 7:21:55 AM | 23.96s |

닫기

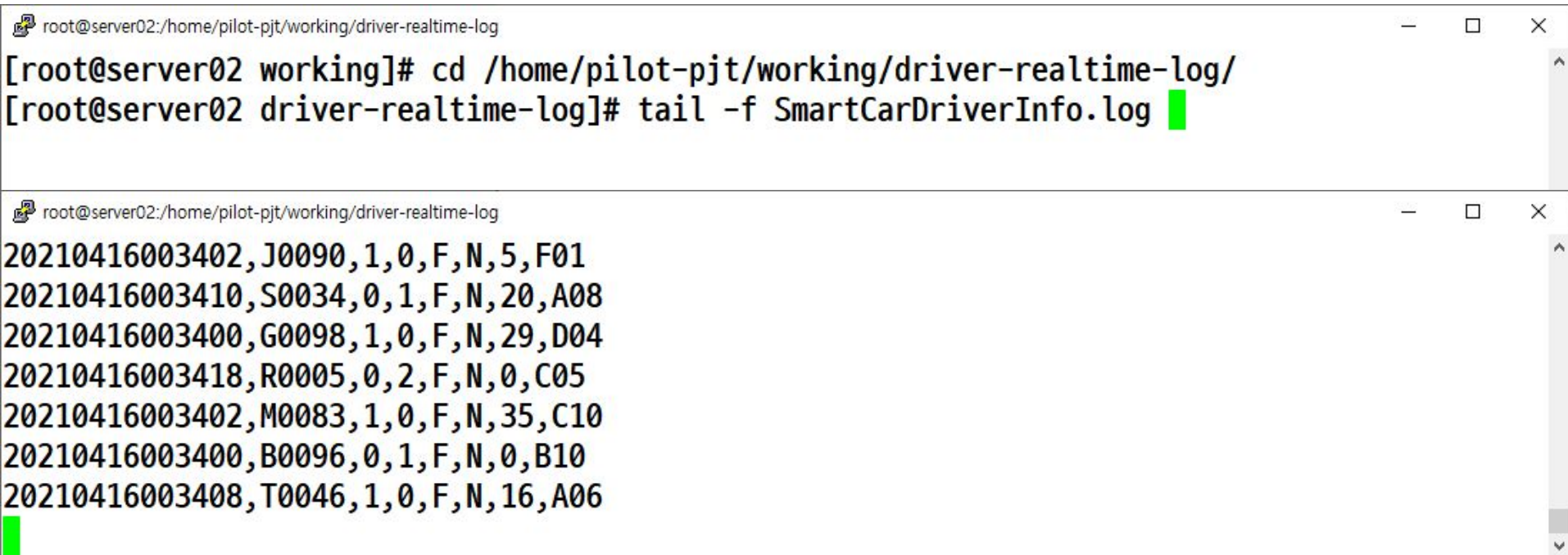
# DriverLogMain 실행

- 터미널 창에서 다음과 같이 DriverLogMain 실행
- 20210416 은 날짜
- 100은 로그 정보를 수집할 자동차의 수
- 100대의 자동차로 부터 2021년 4월 16일의 실시간 운행 정보 가짜로 생성

```
root@server02:/home/pilot-pjt/working
[root@server02 /]# cd /home/pilot-pjt/working/
[root@server02 working]# java -cp bigdata.smartcar.loggen-1.0.jar com.wikibook.bigdata.smartcar.loggen.DriverLogMain 20210416 100 &
[2] 10900
[root@server02 working]#
```

무한루프를 도는 프로그램이므로  
명령어 끝에 & 를 입력 했으므로 백그라운드로 실행

- 가짜로 생성된 자동차 위치정보를 출력한다



```
root@server02:/home/pilot-pjt/working/driver-realtime-log
[root@server02 working]# cd /home/pilot-pjt/working/driver-realtime-log/
[root@server02 driver-realtime-log]# tail -f SmartCarDriverInfo.log

root@server02:/home/pilot-pjt/working/driver-realtime-log
20210416003402,J0090,1,0,F,N,5,F01
20210416003410,S0034,0,1,F,N,20,A08
20210416003400,G0098,1,0,F,N,29,D04
20210416003418,R0005,0,2,F,N,0,C05
20210416003402,M0083,1,0,F,N,35,C10
20210416003400,B0096,0,1,F,N,0,B10
20210416003408,T0046,1,0,F,N,16,A06
```

위치정보의 의미는 교재 32페이지 표2.2의 데이터 스키마를 참조  
실시간 위치 정보를 실시간으로 생성중이므로 Ctrl+C 눌러서  
파일에서 나올것

## 카프카 토픽 확인

- 실시간 운행 정보가 카프카의 메시지를 저장하는 창고 토픽 (SmartCar-Topic)에 저장되었는지 확인

root@server02:/home/pilot-pjt/working

```
[root@server02 working]: kafka-console-consumer --bootstrap-server server02.hadoop.com:9092 --topic SmartCar-Topic --partition 0 --from-beginning
```

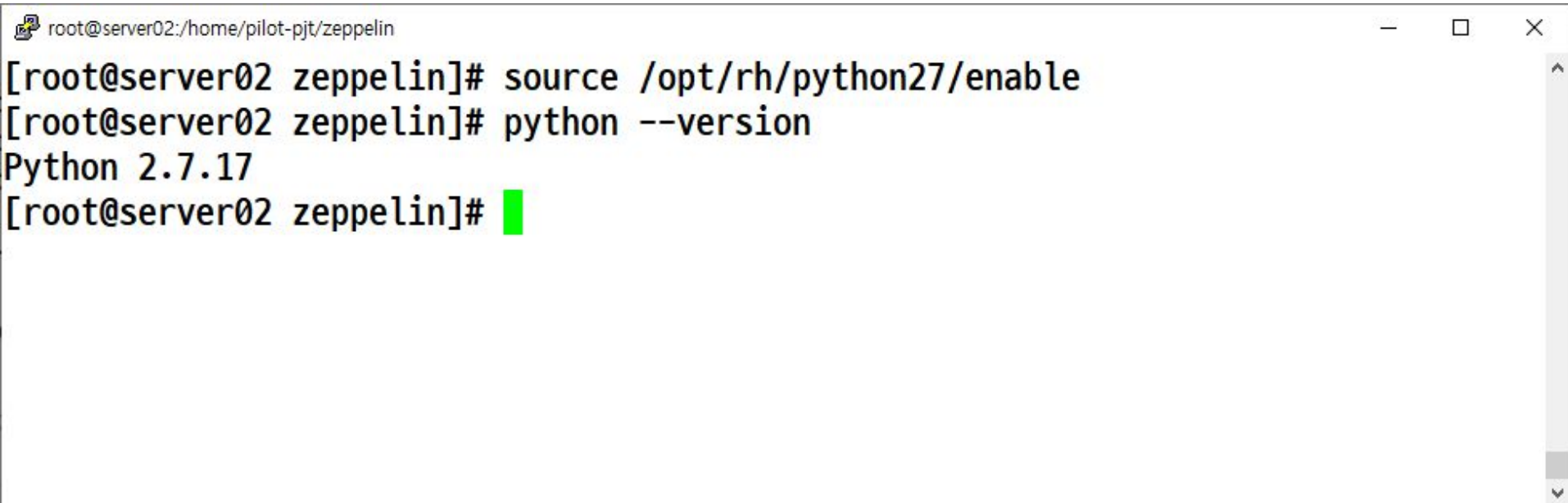
root@server02:/home/pilot-pjt/working

```
20210416000520,J0063,0,0,F,N,20,D09
20210416000520,F0065,3,0,F,N,66,C02
20210416000524,V0041,0,1,L3,L,28,A10
20210416000518,B0081,0,1,F,N,23,E01
20210416000518,H0083,3,0,F,N,62,A05
20210416000526,G0031,0,2,F,N,1,E02
20210416000532,I0006,2,0,L3,L,240,C06
20210416000518,A0082,0,1,R2,R,0,E01
20210416000526,A0030,1,0,F,N,12,B05
```

카프카의 메시지 저장 창고 SmartCar-Topic  
에 실시간 운행 정보가 저장되었음

## 카프카 토픽에 저장된 데이터 읽어 오기

- Server02에서 실행
- Server02에 root계정으로 로그인



```
root@server02:/home/pilot-pjt/zeppelin
[root@server02 zeppelin]# source /opt/rh/python27/enable
[root@server02 zeppelin]# python --version
Python 2.7.17
[root@server02 zeppelin]#
```

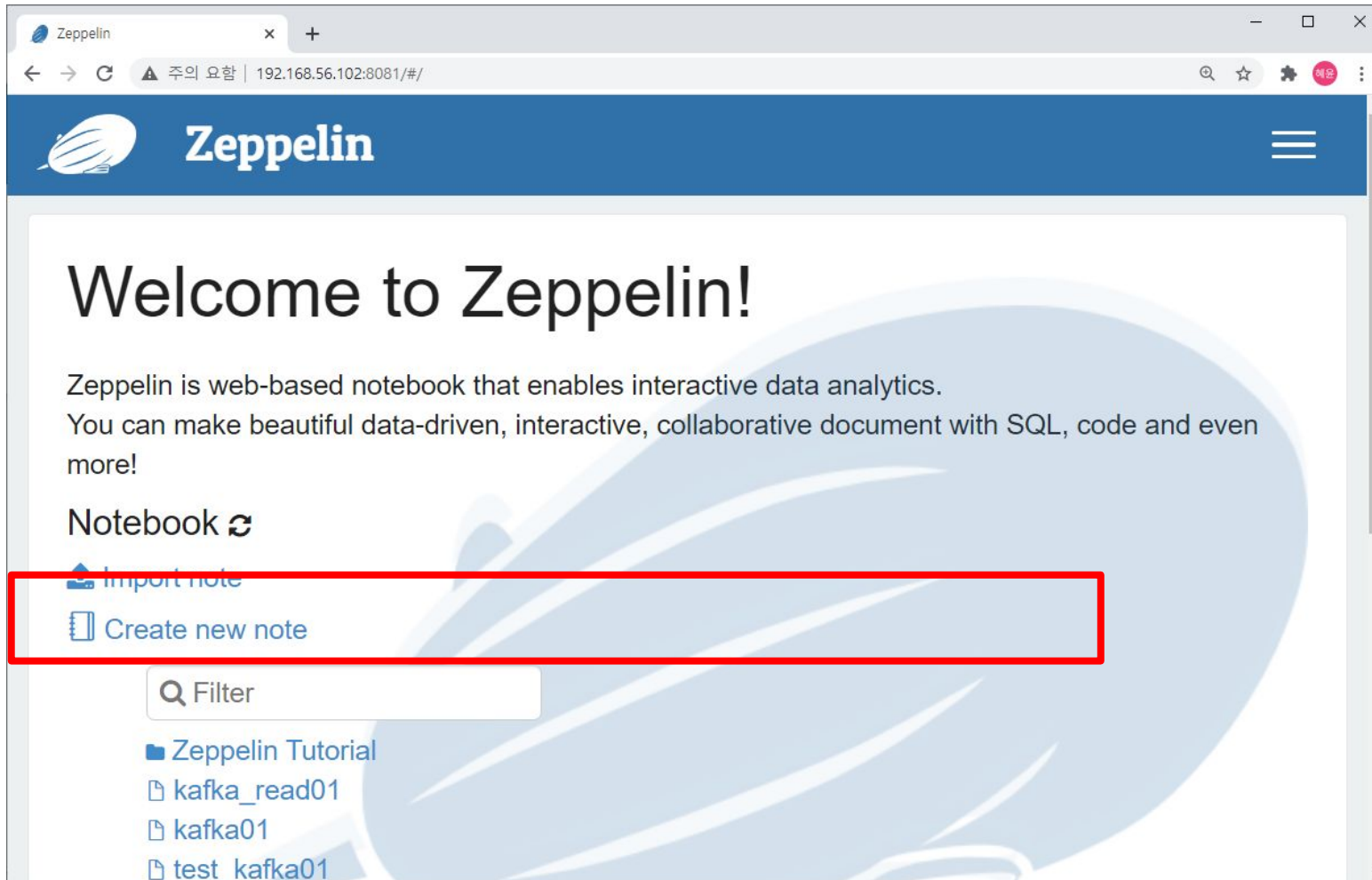
## 카프카 토픽에 저장된 데이터 읽어 오기



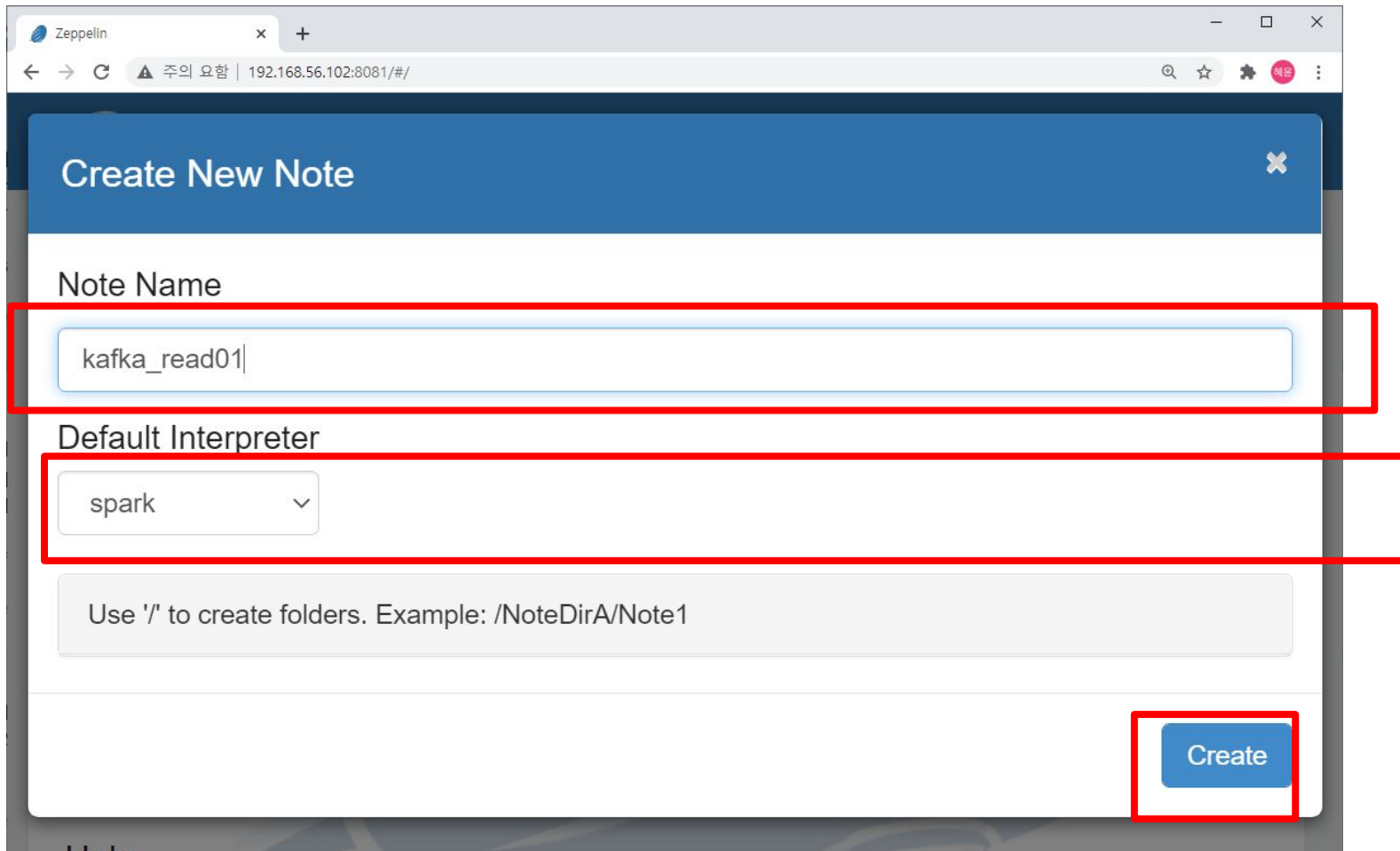
```
root@server02:/home/pilot-pjt/zeppelin
[root@server02 /]# cd /home/pilot-pjt/zeppelin
[root@server02 zeppelin]# zeppelin-daemon.sh start
Zeppelin start [OK]
[root@server02 zeppelin]#
```

A terminal window with a title bar showing standard window controls (minimize, maximize, close) and a scrollbar on the right. The terminal text shows the user navigating to the Zeppelin directory and starting the daemon, which returns an 'OK' status.

- <http://server02.hadoop.com:8081/>



- <http://server02.hadoop.com:8081/>



The screenshot shows the Zeppelin web interface for creating a new note. The browser address bar shows the URL `192.168.56.102:8081/#/`. The form has a blue header with the title "Create New Note" and a close button. Below the header, there are two main sections: "Note Name" and "Default Interpreter". The "Note Name" section contains a text input field with the value "kafka\_read01". The "Default Interpreter" section contains a dropdown menu with the value "spark". At the bottom right of the form is a blue "Create" button. Red rectangular boxes are drawn around the "Note Name" input field, the "Default Interpreter" dropdown, and the "Create" button. Below the "Default Interpreter" section, there is a light gray box containing the text "Use '/' to create folders. Example: /NoteDirA/Note1".

Create New Note

Note Name

kafka\_read01

Default Interpreter

spark

Use '/' to create folders. Example: /NoteDirA/Note1

Create



# 스파크 개발환경 실행

- <http://server02.hadoop.com:8081/>

## kafka\_read01



kafka\_read01

```
%spark.pyspark
import time
```

Took 28 sec. Last updated by anonymous at April 19 2021, 7:49:23 AM.

**Shift+Enter 실행**

```
%spark.pyspark
from pyspark import SparkContext, SparkConf
from pyspark.streaming import StreamingContext
저장소에 저장된 데이터를 다른 곳으로 옮기는 Spark 프로그램을 설정
SparkConf().setMaster("local") : spark 프로그램이 설치된 컴퓨터의 IP (같은 컴퓨터 server02.hadoop.com에 저장되어 있으므로 local)
setAppName("kafka_read01") : 실행 중인 프로그램의 이름을 설정 setAppName("kafka_read01")
conf = SparkConf().setMaster("local").setAppName("kafka_read01")
```

Took 0 sec. Last updated by anonymous at April 19 2021, 7:49:23 AM. (outdated)

**Shift+Enter 실행**

## 스파크 개발환경 실행

```
%spark.pyspark
Kafka 서버가 저장된 아이피
brokers = "server02.hadoop.com:9092"
Kafka Topic
topic = "SmartCar-Topic"
```

Shift+Enter 실행

Took 0 sec. Last updated by anonymous at April 19 2021, 7:49:24 AM. (outdated)

```
%spark.pyspark
from pyspark.streaming.kafka import KafkaUtils
KafkaUtils.createDirectStream : Kafka 에서 데이터를 읽어 올것임
ssc : Kafka에서 2초마다 데이터를 읽어올 StreamingContext
[topic] : 카프카 토픽 (데이터 저장 참고)
brokers : Kafka가 실행중인 컴퓨터
kvs = KafkaUtils.createDirectStream(ssc, [topic], {"metadata.broker.list": brokers})
데이터를 1줄씩 읽어서 리턴
lines = kvs.map(lambda x: x[1])
#데이터를 , 를 기준으로 분리해서 리턴
logs = lines.map(lambda line: line.split(","))
#logs에 저장된 데이터 출력
logs.pprint()
```

Shift+Enter 실행

Took 1 sec. Last updated by anonymous at April 19 2021, 7:49:25 AM. (outdated)

```
%spark.pyspark
ssc : 카프카에서 데이터 읽어오기 시작
ssc.start()
100초 대기
time.sleep(100)
카프카에서 데이터 읽어오기 멈춤
ssc.stop()
```

Shift+Enter 실행

```
[u'20210416041542', u'B0019', u'0', u'0', u'F', u'N', u'38', u'A08']
[u'20210416041534', u'B0059', u'1', u'0', u'F', u'N', u'96', u'E08']
[u'20210416041540', u'C0030', u'0', u'1', u'L3', u'L', u'170', u'E10']
...
```

Kafka에 저장된 실시간 운행정보가 출력되는지 확인

```

Time: 2021-04-19 07:49:28

```

```
[u'20210416041606', u'Z0098', u'2', u'0', u'F', u'N', u'48', u'A01']
[u'20210416041614', u'H0076', u'0', u'2', u'R2', u'R', u'8', u'C02']
[u'20210416041610', u'U0094', u'4', u'0', u'F', u'N', u'154', u'C02']
[u'20210416041616', u'D0054', u'3', u'0', u'F', u'N', u'95', u'C01']
[u'20210416041616', u'V0071', u'2', u'0', u'R2', u'R', u'35', u'E09']
[u'20210416041622', u'Q0026', u'0', u'0', u'F', u'N', u'55', u'B01']
[u'20210416041622', u'B0019', u'2', u'0', u'F', u'N', u'41', u'A07']
[u'20210416041630', u'X0006', u'0', u'1', u'R3', u'R', u'74', u'A09']
[u'20210416041624', u'G0021', u'0', u'1', u'L2', u'L', u'39', u'D03']
```

## 프로그램의 종료

- 스파크 스트림은 백그라운드 작업 ( 눈에 보이지 않게 무한 루프로 실행) 으로 데이터를 옮기기 때문에 개발환경의 실행이 끝나도 눈에 안보이게 무한루프가 실행 중
- 프로그램을 종료 하기 위해서는 강제로 종료 해야함

The screenshot shows the Zeppelin Notebook web interface. The browser tab is 'hbase01 - Zeppelin'. The address bar shows the URL '192.168.56.102:8081/#/notebook/2G5UZKAHY'. The Zeppelin logo and 'Notebook' tab are visible. The user is logged in as 'anonymous'. A dropdown menu is open, showing options: 'About Zeppelin', 'Interpreter' (highlighted with a red box), 'Notebook Repos', 'Credential', 'Helium', and 'Configuration'. The notebook title is 'hbase01'. Below the title, there are execution controls and a status bar indicating 'Took 27 sec. Last updated by anonymous at April 22 2021, 7:27:39 AM.' The code area contains the following text:

```
%spark.pyspark
connect.tables() # HBASE 에 저장된 테이블이 조회되는지 확인

['DriverCarInfo', 'books', 'stocks', 'test', 'testtable']
```

Below the code, the status bar indicates 'Took 1 sec. Last updated by anonymous at April 22 2021, 7:27:40 AM.'



The screenshot shows the Zeppelin web interface. The browser tab is labeled 'Zeppelin' and the address bar shows '192.168.56.102:8081/#/interpreter'. The Zeppelin logo and navigation links 'Notebook' and 'Job' are visible. A search bar contains the text 'spark' and is highlighted with a red box. Below the search bar, the 'Interpreters' section is displayed. It includes a 'Repository' button and a '+ Create' button. A message states: 'Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.' Below this, the 'spark' interpreter is listed with its configuration: '%spark, %sql, %dep, %pyspark, %ipyspark, %r'. To the right of the configuration are buttons for 'spark ui', 'edit', 'restart' (highlighted with a red box), and 'remove'. Below the configuration, the 'Option' section is visible, showing 'The interpreter will be instantiated' followed by dropdown menus for 'Globally' and 'in shared', and the word 'process' with an information icon.

## 실시간 운행정보 생성 중지

```
root@server02:/home/pilot-pjt/zeppelin
[root@server02 zeppelin]# ps -ef | grep smartcar.log
root 10966 7210 2 07:36 pts/0 00:00:34 java -cp bigdata.smartcar.loggen-1.0.jar com.wikibook.bigdata.smartcar.loggen.DriverLogMain 20211016 100
root 15369 7210 0 08:00 pts/0 00:00:00 grep smartcar.log
[root@server02 zeppelin]# kill -9 10966
[root@server02 zeppelin]#
```

윗줄의 pid 번호를 확인 (프로그램마다 다름)  
하고 kill -9 뒤에 입력해서 프로그램 종료

## Kafka 데이터 HBASE 테이블에 데이터 추가

- Zeppline 에서 `kafka_read_hbase_write_01` 노트북 생성
- 다음과 같은 코드를 입력한다

# Kafka 데이터 HBASE 테이블에 데이터 추가

```
%spark.pyspark
import time
import happybase
```

Shift+Enter 실행

Took 27 sec. Last updated by anonymous at April 21 2021, 5:29:57 AM.

```
%spark.pyspark
from pyspark import SparkContext, SparkConf
from pyspark.streaming import StreamingContext
저장소에 저장된 데이터를 다른 곳으로 옮기는 Spark 프로그램을 설정
SparkConf().setMaster("local") : spark 프로그램이 설치된 컴퓨터의 IP (같은 컴퓨터 server02.hadoop.com에 저장되 있으므로 local)
setAppName("kafka_read01") : 실행 중인 프로그램의 이름을 설정 setAppName("kafka_read01")
conf = SparkConf().setMaster("local").setAppName("kafka_read_hbase_write_01")
```

Shift+Enter 실행

Took 0 sec. Last updated by anonymous at April 21 2021, 5:30:01 AM.

```
%spark.pyspark
SparkContext.getOrCreate(conf=conf) : conf 에 저장한 설정으로 데이터를 읽어올 Spark 프로그램 객체 (SparkContext)
sc = SparkContext.getOrCreate(conf=conf)
StreamingContext(sc, 2) : Kafka에 저장된 데이터는 sc 프로그램이 2초마다 읽어 올 것임
ssc = StreamingContext(sc, 2)
```

Shift+Enter 실행

Took 0 sec. Last updated by anonymous at April 21 2021, 5:30:03 AM.



```
%spark.pyspark
Kafka 서버가 저장된 아이피
brokers ="server02.hadoop.com:9092"
Kafka Topic
topic = "SmartCar-Topic"
```

Took 0 sec. Last updated by anonymous at April 21 2021, 5:30:05 AM.

```
%spark.pyspark
def put_hbase(line):
 # HBASE 버전
 CDH6_HBASE_THRIFT_VER='0.92'

 connect = happybase.Connection(
 host='server02', # HBASE 저장된 서버
 port=9090, # HBASE 포트
 table_prefix=None,
 compat=CDH6_HBASE_THRIFT_VER, # HBASE 버전 설정
 timeout=None,
 autoconnect=True,
 transport='framed',
 protocol='compact'
)

 # HBASE 의 'DriverCarInfo' 에 데이터를 추가 할 객체 생성
 table = connect.table('DriverCarInfo')
 # 카프카에서 읽어온 데이터를 , 를 기준으로 분리
 line_split_col = line.split(",")
 # HBASE 에 추가할 새로운 레코드 생성
 new_row = {
 'cf1:date': line_split_col[0],
 'cf1:car_number':line_split_col[1],
 "cf1:speed_pedal":line_split_col[2],
 "cf1:break_pedal":line_split_col[3],
 "cf1:steer_angle":line_split_col[4],
 "cf1:direct_light":line_split_col[5],
 "cf1:speed":line_split_col[6],
 "cf1:area_number":line_split_col[7]
 }

 # HBASE 레코드 추가
 # put(primary key, 레코드)
 table.put(line_split_col[0]+"-"+line_split_col[1], new_row)

 return new_row
```

# Kafka 데이터 HBASE 테이블에 데이터 추가

```
%spark.pyspark
from pyspark.streaming.kafka import KafkaUtils
KafkaUtils.createDirectStream : Kafka 에서 데이터를 읽어 올것임
ssc : Kafka에서 2초마다 데이터를 읽어올 StreamingContext
[topic] : 카프카 토픽 (데이터 저장 참고)
brokers : Kafka가 실행중인 컴퓨터
kvs = KafkaUtils.createDirectStream(ssc, [topic], {"metadata.broker.list": brokers})
데이터를 1줄씩 읽어서 리턴
lines = kvs.map(lambda x: x[1])
hbase_insert_number = lines.map(lambda line: put_hbase(line))
hbase_insert_number.pprint()
```

Took 1 sec. Last updated by anonymous at April 21 2021, 5:30:11 AM.

```
%spark.pyspark
ssc : 카프카에서 데이터 읽어오기 시작
ssc.start()
100 초 대기
time.sleep(100)
카프카에서 데이터 읽기 중지
ssc.stop()
```

-----  
Time: 2021-04-21 05:30:16  
-----

```
{'cf1:area_number': u'E04', 'cf1:speed_pedal': u'4', 'cf1:speed': u'73', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'F', 'cf1:direct_
{'cf1:area_number': u'C07', 'cf1:speed_pedal': u'1', 'cf1:speed': u'90', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'F', 'cf1:direct_
{'cf1:area_number': u'E03', 'cf1:speed_pedal': u'1', 'cf1:speed': u'29', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'F', 'cf1:direct_
{'cf1:area_number': u'B06', 'cf1:speed_pedal': u'5', 'cf1:speed': u'110', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'F', 'cf1:direct_
{'cf1:area_number': u'C09', 'cf1:speed_pedal': u'2', 'cf1:speed': u'83', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'F', 'cf1:direct_
{'cf1:area_number': u'C10', 'cf1:speed_pedal': u'1', 'cf1:speed': u'70', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'F', 'cf1:direct_
{'cf1:area_number': u'E10', 'cf1:speed_pedal': u'3', 'cf1:speed': u'48', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'F', 'cf1:direct_
{'cf1:area_number': u'D04', 'cf1:speed_pedal': u'1', 'cf1:speed': u'55', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'F', 'cf1:direct_
{'cf1:area_number': u'B10', 'cf1:speed_pedal': u'2', 'cf1:speed': u'250', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'L3', 'cf1:direc
{'cf1:area_number': u'C02', 'cf1:speed_pedal': u'1', 'cf1:speed': u'104', 'cf1:break_pedal': u'0', 'cf1:steer_angle': u'L2', 'cf1:direc
```

## hbase 추가 확인

```
root@server02:~
[root@server02 ~]# hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.1.0-cdh6.3.2, rUnknown, Fri Nov 8 05:44:07 PST 2019
Took 0.0008 seconds
hbase(main):001:0> scan 'DriverCarInfo'
```

```
root@server02:~
20210420123410-M0007 column=cf1:speed_pedal, timestamp=1618950632857, value=0
20210420123410-M0007 column=cf1:steer_angle, timestamp=1618950632857, value=F
20210420123410-P0001 column=cf1:area_number, timestamp=1618950632832, value=D06
20210420123410-P0001 column=cf1:break_pedal, timestamp=1618950632832, value=1
20210420123410-P0001 column=cf1:car_number, timestamp=1618950632832, value=P0001
20210420123410-P0001 column=cf1:date, timestamp=1618950632832, value=20210420123410
20210420123410-P0001 column=cf1:direct_light, timestamp=1618950632832, value=N
20210420123410-P0001 column=cf1:speed, timestamp=1618950632832, value=30
20210420123410-P0001 column=cf1:speed_pedal, timestamp=1618950632832, value=0
20210420123410-P0001 column=cf1:steer_angle, timestamp=1618950632832, value=F
```



## 프로그램의 종료

- 스파크 스트림은 백그라운드 작업 ( 눈에 보이지 않게 무한 루프로 실행) 으로 데이터를 옮기기 때문에 개발환경의 실행이 끝나도 눈에 안보이게 무한루프가 실행 중
- 프로그램을 종료 하기 위해서는 강제로 종료 해야함

The screenshot shows the Zeppelin Notebook web interface. The browser tab is 'hbase01 - Zeppelin'. The address bar shows the URL '192.168.56.102:8081/#/notebook/2G5UZKAHY'. The Zeppelin logo and 'Notebook' tab are visible. The notebook title is 'hbase01'. A dropdown menu for the user 'anonymous' is open, with the 'Interpreter' option highlighted. The notebook content shows a Spark command to connect to HBase tables.

hbase01

Took 27 sec. Last updated by anonymous at April 22 2021, 7:27:39 AM.

```
%spark.pyspark
connect.tables() # HBASE 에 저장된 테이블이 조회되는지 확인

['DriverCarInfo', 'books', 'stocks', 'test', 'testtable']
```

Took 1 sec. Last updated by anonymous at April 22 2021, 7:27:40 AM.

The screenshot shows the Zeppelin web interface. The browser tab is labeled 'Zeppelin' and the address bar shows '192.168.56.102:8081/#/interpreter'. The Zeppelin logo and navigation links 'Notebook' and 'Job' are visible. A search bar contains the text 'spark' and is highlighted with a red box. Below the search bar, the 'Interpreters' section is displayed. It includes a 'Repository' button and a '+ Create' button. A message states: 'Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.' Below this, a list of interpreters is shown. The first entry is 'spark', followed by a list of languages: '%spark, %sql, %dep, %pyspark, %ipyspark, %r'. To the right of this entry are four buttons: 'spark ui', 'edit', 'restart', and 'remove'. The 'restart' button is highlighted with a red box. Below the interpreter list, there is an 'Option' section with the text 'The interpreter will be instantiated' followed by two dropdown menus: 'Globally' and 'in shared', and the word 'process' with an information icon.

Zeppelin

Notebook Job

Search

anonymous

## Interpreters

Repository + Create

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

spark

spark %spark, %sql, %dep, %pyspark, %ipyspark, %r

spark ui edit restart remove

### Option

The interpreter will be instantiated Globally in shared process