

3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

SmartCar 로그 시뮬레이터 작동 (1/3)

01. 먼저 Server02에 SSH 접속을 하고 bigdata.smartcar.loggen-1.0.jar가 위치한 곳으로 이동한다.

```
$ cd /home/pilot-pjt/working
```

02. 다음 명령으로 2개의 스마트카 로그 시뮬레이터를 백그라운드 방식으로 실행한다.

```
$ java -cp bigdata.smartcar.loggen-1.0.jar com.wikibook.bigdata.smartcar.loggen.CarLogMain  
20160101 3 &
```

```
$ java -cp bigdata.smartcar.loggen-1.0.jar com.wikibook.bigdata.smartcar.loggen.  
DriverLogMain 20160101 3 &
```

2016년 1월 1일에 3대의 스마트카에 대한 상태 정보와 운전자의 운행 정보가 생성되기 시작한다.

3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

SmartCar 로그 시뮬레이터 작동 (2/3)

03. 정상적으로 시뮬레이터가 작동되고 있는지 아래의 내용으로 확인해 본다.

- /home/pilot-pjt/working/SmartCar 경로에 SmartCarStatusInfo_20160101.txt 파일이 생성됐는지 확인한다.
SmartCarStatusInfo_20160101.txt 파일의 내용을 확인해 보면 3대의 스마트카 상태 정보가 기록된 것을 볼 수 있다.

```
$ cd /home/pilot-pjt/working/SmartCar
```

```
$ vi SmartCarStatusInfo_20160101.txt
```

- /home/pilot-pjt/working/driver-realtime-log 경로에 SmartCarDriverInfo.log 파일이 생성됐는지 확인한다.
tail -f SmartCarDriverInfo.log 명령을 통해 3대의 스마트카 운전자의 운행 정보가 실시간으로 발생하는 것을 볼 수 있다.

```
$ cd /home/pilot-pjt/working/driver-realtime-log
```

```
$ tail -f SmartCarDriverInfo.log
```

3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

SmartCar 로그 시뮬레이터 작동 (3/3)

04. 마지막으로 /home/pilot-pjt/working/SmartCar 경로에 만들어진 SmartCarStatusInfo_20160101.txt 파일을 플럼 SmartCarInfo 에이전트의 SpoolDir 경로로 옮긴다.

```
$ mv /home/pilot-pjt/working/SmartCar/SmartCarStatusInfo_20160101.txt /home/pilot-pjt/working/car-batch-log/
```

3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

플럼 에이전트 작동

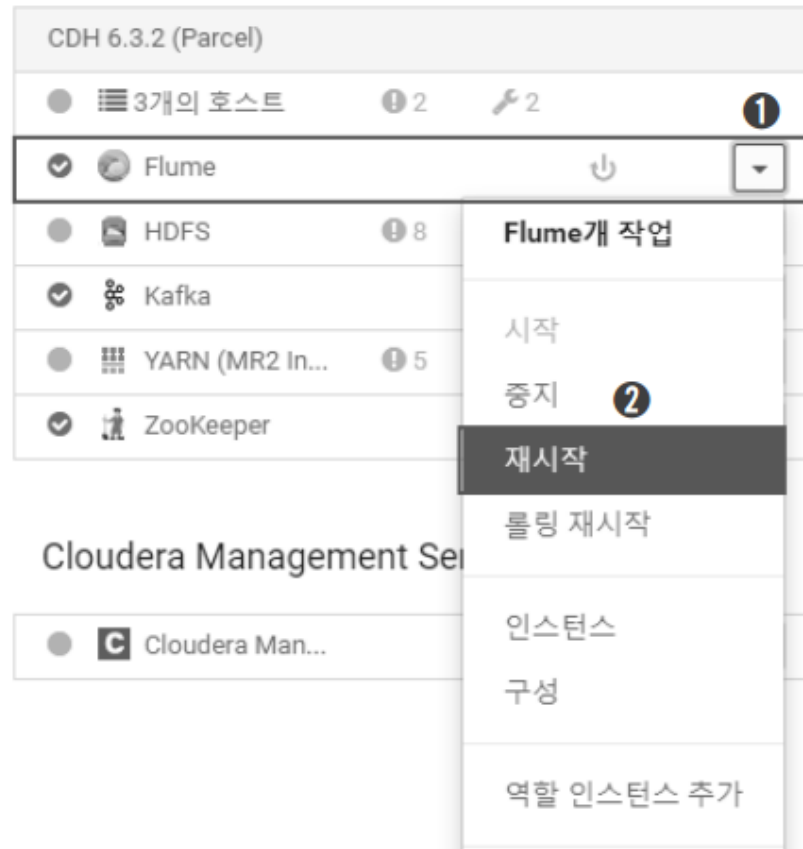


그림 3.33 CM을 이용한 플럼 에이전트 재시작

3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

카프카 Consumer 작동

```
$ kafka-console-consumer --bootstrap-server server02.hadoop.com:9092 --topic SmartCar-Topic  
--partition 0
```

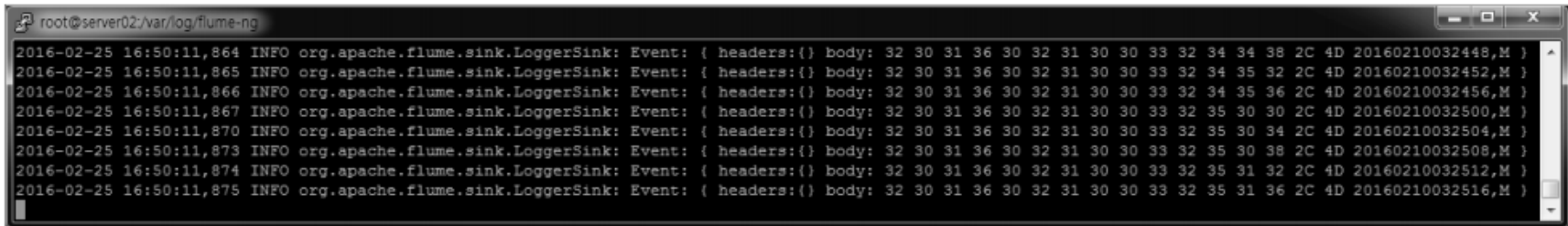
3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

수집 기능 점검 (1/3)

01. 스마트카의 상태 정보 로그 파일이 플럼의 표준 출력 로그로 전송됐는지 리눅스 tail 명령어를 통해 확인한다.

```
$ tail -f /var/log/flume-ng/flume-cmf-flume-AGENT-server02.hadoop.com.log
```

그림 3.34처럼 수집된 스마트카의 상태값 데이터가 출력되면 성공적으로 수집되고 있는 것이다.



```
root@server02:/var/log/flume-ng
2016-02-25 16:50:11,864 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 32 30 31 36 30 32 31 30 30 33 32 34 34 38 2C 4D 20160210032448,M }
2016-02-25 16:50:11,865 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 32 30 31 36 30 32 31 30 30 33 32 34 35 32 2C 4D 20160210032452,M }
2016-02-25 16:50:11,866 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 32 30 31 36 30 32 31 30 30 33 32 34 35 36 2C 4D 20160210032456,M }
2016-02-25 16:50:11,867 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 32 30 31 36 30 32 31 30 30 33 32 35 30 30 2C 4D 20160210032500,M }
2016-02-25 16:50:11,870 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 32 30 31 36 30 32 31 30 30 33 32 35 30 34 2C 4D 20160210032504,M }
2016-02-25 16:50:11,873 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 32 30 31 36 30 32 31 30 30 33 32 35 30 38 2C 4D 20160210032508,M }
2016-02-25 16:50:11,874 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 32 30 31 36 30 32 31 30 30 33 32 35 31 32 2C 4D 20160210032512,M }
2016-02-25 16:50:11,875 INFO org.apache.flume.sink.LoggerSink: Event: { headers:{} body: 32 30 31 36 30 32 31 30 30 33 32 35 31 36 2C 4D 20160210032516,M }
```

그림 3.34 수집 기능 점검 1 - 플럼의 표준 출력 로그 파일로 수집된 데이터 확인

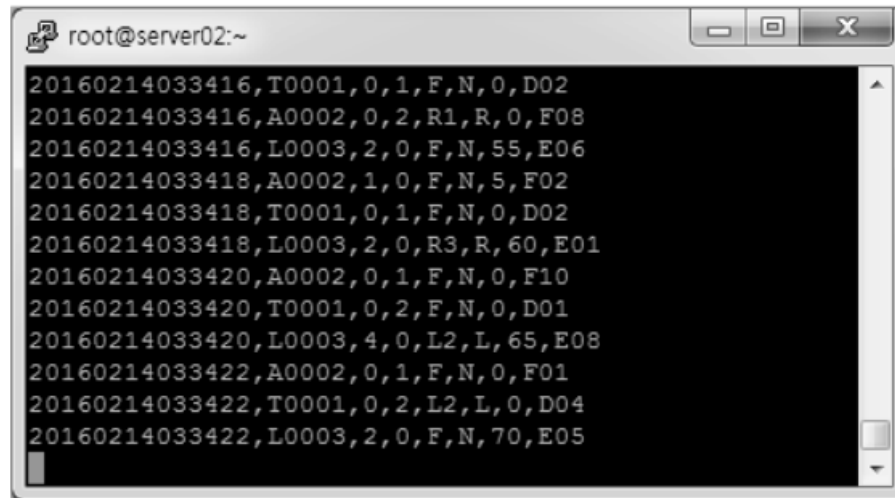
3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

수집 기능 점검 (2/3)

02. 이제 스마트카 운전자의 실시간 운전 정보인 DriverCarInfo가 정상적으로 수집되는지 확인한다. 앞서 실행했던 Kafka의 Consumer 콘솔창을 확인해 보자. 만약 창을 닫았다면 Server02에 SSH 접속 후, 아래의 명령어를 다시 실행한다.

```
kafka-console-consumer --bootstrap-server server02.hadoop.com:9092 --topic SmartCar-Topic  
--partition 0
```

아래와 같이 스마트카의 운전자 정보가 실시간으로 수집되고 있는지 확인할 수 있다.

A terminal window titled 'root@server02:~' showing the output of the Kafka console consumer. The output consists of 14 lines of comma-separated data representing driver information, including timestamps, IDs, and various attributes. The data is displayed in a black terminal window with white text. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

```
root@server02:~  
20160214033416,T0001,0,1,F,N,0,D02  
20160214033416,A0002,0,2,R1,R,0,F08  
20160214033416,L0003,2,0,F,N,55,E06  
20160214033418,A0002,1,0,F,N,5,F02  
20160214033418,T0001,0,1,F,N,0,D02  
20160214033418,L0003,2,0,R3,R,60,E01  
20160214033420,A0002,0,1,F,N,0,F10  
20160214033420,T0001,0,2,F,N,0,D01  
20160214033420,L0003,4,0,L2,L,65,E08  
20160214033422,A0002,0,1,F,N,0,F01  
20160214033422,T0001,0,2,L2,L,0,D04  
20160214033422,L0003,2,0,F,N,70,E05
```

그림 3.35 수집 기능 점검 2 - 카프카 Consumer로 실시간 수집 데이터 확인

3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

수집 기능 점검 (3/3)

03. 이제 백그라운드로 실행했던 스마트카 로그 시뮬레이터를 모두 종료한다.

```
$ ps -ef | grep smartcar.log
```

위 명령어로 조회된 두 자바 프로세스(CarLogMain, DriverLogMain)의 pid를 찾아 강제로 종료하자.

```
$ kill -9 [pid]
```


Tip _ 파일럿 환경의 로그 확인

빅데이터 시스템에서는 에코시스템들의 로그를 확인하는 것이 중요하다. 많은 소프트웨어가 설치되고 서로간의 의존성이 커서 다양한 문제점들을 로그를 통해 확인해야 하기 때문이다. 파일럿 환경의 로그를 점검하기 위해서는 아래의 경로들을 참고한다.

- Hadoop 에코시스템 서버들의 로그 위치: /var/log/디렉터리(cloudera, Hadoop, Oozie 등)
- Redis 서버 로그 위치: /var/log/redis_6379.log
- Storm 서버 로그 위치: /home/pilot-pjt/storm/logs/
- Zeppelin 서버 로그 위치: /home/pilot-pjt/zeppelin-0.8.2-bin-all/logs

Tip _ 파일럿 환경에서 HDFS 문제 발생

개인의 파일럿 프로젝트 환경은 가상머신으로 구성돼 있어 비정상적인 종료가 자주 발생할 수밖에 없다. 이때 HDFS상에 CORRUPT BLOCKS/FILES 같은 문제가 발생하거나 Safe 모드로 전환되어 빠져 나오지 못하는 경우가 자주 발생한다. 만약 파일럿 환경의 일부 기능 또는 설치 중에 문제가 발생한다면 HDFS의 파일/블록 깨짐 또는 Safe모드 전환 여부를 체크해야 한다.

- HDFS 파일 시스템 검사: `$ hdfs fsck /`
- HDFS에 Safe 모드 발생 후 빠져나오지 못할 경우
Safe 모드 강제 해제: `$ hdfs dfsadmin -safemode leave`
- HDFS에 CORRUPT BLOCKS/FILES 등이 발생해 복구가 불가능한 경우
손상된 파일 강제 삭제: `$ hdfs fsck / -delete`
손상된 파일을 `/lost + found` 디렉터리로 이동: `$ hdfs fsck / -move`

3.7 수집 파일럿 실행 5단계 - 수집 기능 테스트

실습