20220426

HBase shell

1. hbase shell을 입력하여 접속
2. create “smartcar_test_table”, “cf”로 테이블 생성
3. put 'smartcar_test_table', 'row-key1', 'cf:model', '20001’ 로 컬럼 생성
4. put 'smartcar_test_table', 'row-key1', 'cf:no' , ‘12345’ 로 컬럼 생성
5. get 'smartcar_test_table', 'row-key1’을 이용하여 생성한 컬럼 불러오기
6. disable 'smartcar_test_table’ 테이블 비활성화
7. drop 'smartcar_test_table’ 테이블 삭제

- HBase는 자원 소모가 높은 서버이므로 사용하지 않을 때는 일시 정지한다. 저사양 파일럿 환경에서는HBase 외에도 각 단계를 진행하면서 취사선택해야 하는 서비스들이 있다. 불편하지만 안정적인 파일럿환경을 유지하기 위해 미사용 서버는 중지시킨다.

레디스 설치

- yum install -y gcc* 설치
- yum install -y tcl 설치

** 설치가 되지 않을 때

echo "http://vault.centos.org/6.10/os/x86_64/" > /var/cache/yum/x86_64/6/base/mirrorlist.txt
echo "http://vault.centos.org/6.10/extras/x86_64/" > /var/cache/yum/x86_64/6/extras/mirrorlist.txt
echo "http://vault.centos.org/6.10/updates/x86_64/"/var/cache/yum/x86_64/6/updates/mirrorlist.txt
echo "http://vault.centos.org/6.10/sclo/x86_64/rh" > /var/cache/yum/x86_64/6/centos-sclo-rh/mirrorlist.txt
echo "http://vault.centos.org/6.10/sclo/x86_64/sclo" > /var/cache/yum/x86_64/6/centos-sclo-sclo/mirrorlist.txt 입력

레디스 5.0.7을 내려받아 빌드 및 설치

1. cd /home/pilot-pjt 로 이동
2. wget http://download.redis.io/releases/redis-5.0.7.tar.gz 로 설치 (파일 위치가 잘못 설치 되었을 때 mv 기존 경로 새로운 경로 로 이동)

3. tar -xvf redis-5.0.7.tar.gz 로 압축파일 풀기 (v : 푸는거 보여주기 f : 압축파일)
4. cd redis-5.0.7 로 이동
5. make 설치
6. make install로 오브젝트 파일 설치
7. cd /home/pilot-pjt/redis-5.0.7/utils 경로 이동
8. chmod 755 install_server.sh 로 변경 후에 ./install_server.sh를 실행
9. Please select the redis executable path [] 에 /usr/local/bin/redis-server 입력
10. vi /var/log/redis_6379.log 명령어로 레디스 서버 기동 여부 확인
11. service redis_6379 status 성공적으로 설치됐는지 점검

레디스 서비스를 시작/종료하는 명령은 다음과 같다.
시작: service redis_6379 start
종료: service redis_6379 stop

- vim /etc/redis/6379.conf 입력 후 /bind 127.0.0.1 치고 n을 이용하여 찾은다음 주석처리, /protected-mode yes 를 no로 변경 후 service redis_6379 restart로 재시작
    1. /usr/local/bin/redis-cli 접근
    2. set key:1 Hello!BigData 저장
    3. get key:1 로 조회
    4. del key:1 로 삭제
    5. quit 로 종료
- 경로가 /usr/local/bin 이 기본경로 아닌 경우

  vim ~/.bash_profile로 PATH설정

  source ~/.bash_profile 로 재정의

스톰 설치

- 스톰 설치
    1. cd /home/pliot-pjt/ 로 경로 이동
    2. wget http://archive.apache.org/dist/storm/apache-storm-1.2.3/apache-storm-1.2.3.tar.gz 입력하여 설치
    3. tar -xvf apache-storm-1.2.3.tar.gz 압축 파일 설치
    4. ln -s apache-storm-1.2.3 storm 심볼 생성

5. cd /home/pilot-pjt/storm/conf 경로 이동
6. vim storm.yaml 에서 설정

```
storm.zookeeper.servers:
 - "server02.hadoop.com"

storm.local.dir: "/home/pilot-pjt/storm/data"

nimbus.seeds: ["server02.hadoop.com"]

supervisor.slots.ports:
 - 6700

ui.port: 8088
```

- 스톰의 로그 레벨 수정(안해도 되지만, 성능상 문제로 수정)
  1. cd /home/pilot-pjt/strom/log4j2 로 경로 이동
  2. vim cluster.xml 을 열기
     a. :75를 입력하여 “level=info”를 모두 “level=error”로 변경
  3. vim worker.xml 을 열기
     a. :75를 입력하여 “level=info”를 모두 “level=error”로 변경
- 스톰 패스 경로 설정
  1. vim /root/.bash_profile 로 접근
  2. "/home/pilot-pit/storm/bin” 경로 추가
  3. source /root/.bash_profile 재시작
- JAVA 버전 확인
  - java -version (1.8.0버전이면 됨, 아니면 수정)
  - 아닐경우.
  - rm /usr/bin/java
  - rm /usr/bin/javac
  - ln -s /usr/java/jdk1.8.0_181-cloudera/bin/javac /usr/bin/javac
  - ln -s /usr/java/jdk1.8.0_181-cloudera/bin/java /usr/bin/java
- storm-numbus, storm-supervisor, storm-ui 를 파일질라를 이용하여 업로드
  - 업로드 경로 : /etc/rc.d/init.d
  - 권한 변경 → 실행 다 체크

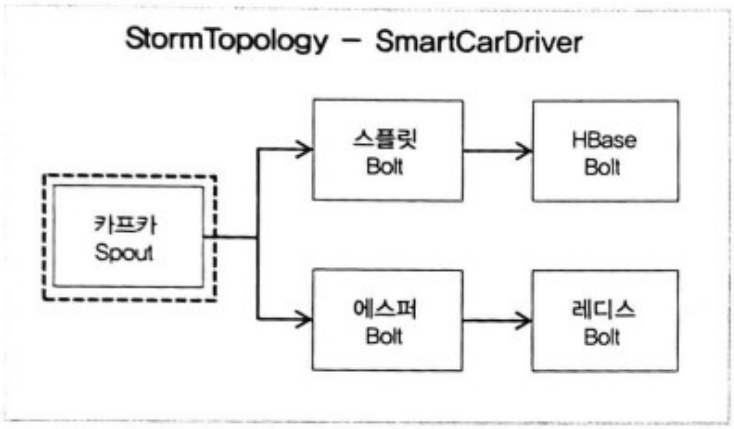
- 업로드 후 파일 권한 수정
  - chmod 755 /etc/init.d/storm-* (3개 파일이 동시 적용)
- Log 및 Pid 디렉터리 생성
  - mkdir /var/{log,run}/storm
- service/chkconfig 등록 명령을 각각 실행
  - service storm-nimbus start
  - service storm-supervisor start
  - service storm-ui start
  - ** 안될때는 로그 확인 .../var/log/storm에 있는 로그 확인
- chkconfig 를 이용한 실행
  - chkconfig storm-nimbus on
  - chkconfig storm-supervisor on
  - chkconfig storm-ui on
- chkconfig —list | grep storm-* 을 이용하여 2번 4번 5번이 on 되어있는지 확인
- 스톰이 정상적으로 구동 됐는지 확인
  - service storm-nimbus status
  - service storm-supervisor status
  - service storm-ui status
- 스톰 UI에 접속해 스톰의 중요 상태들을 모니터링 할 수 있다.
  - http://server02.hadoop.com:8088

3단계 - 실시간 적재 기능 구현

실시간 적재 기능 구현은 스톰의 Spout와 Bolt의 프로그램 구현 단계에 해당

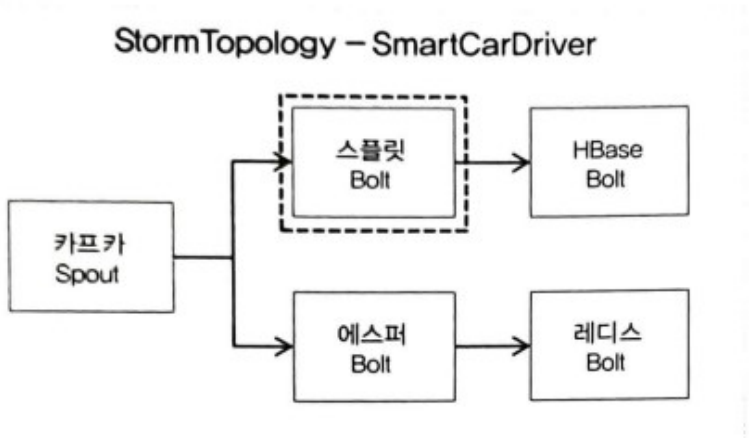
컴포넌트는 모두 자바 프로그램이며, 해당 프로그램을 통해 실시간 적재 기능을 구현한다.

kafka Spout 기능 구현



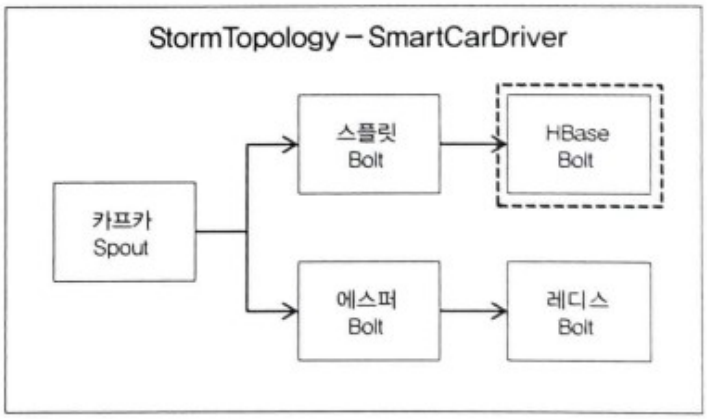
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4406f402-013a-4079-97cd-7705442d92c1/SmartCarDriverTopology.java

Split Bolt 기능 구현

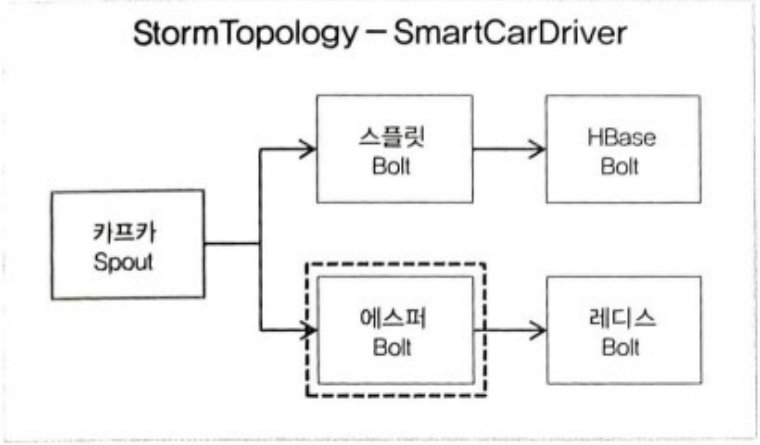


https://s3-us-west-2.amazonaws.com/secure.notion-static.com/79b518c0-05ab-4c50-8232-3e482c5a44c9/SplitBolt.java

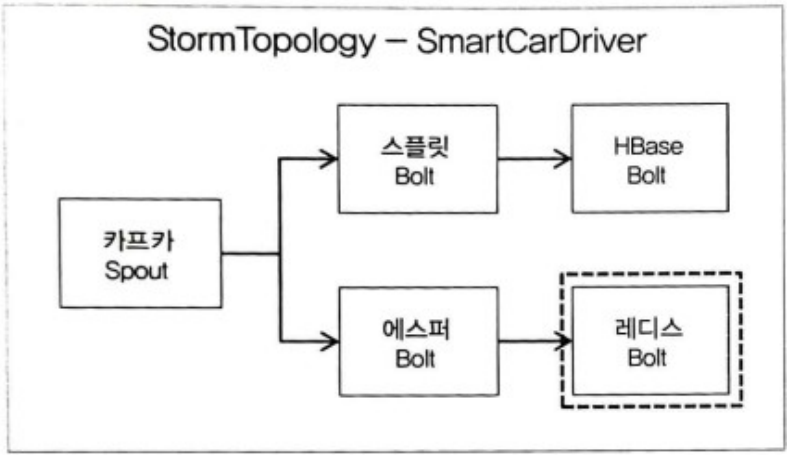
HBase Bolt 기능 구현



에스퍼 Bolt 기능 구현



레디스 Bolt 기능 구현



레디스 Bolt 구현의 핵심은 레디스 클라이언트 라이브러리인 **제디스(Jedis)**이다.

제디스는 제어하는 역할을 한다.

레디스 Bolt에서 레디스 서버에 적재한 과속 차량 정보는 스피드 데이터로부터 추출된 유용한정보다. 이처럼 가치 있는 실시간 정보는 주변 업무 시스템에서 곧바로 활용할 수 있어야 한다.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b2a43502-b2d1-4471-afd1-497f33708470/SmartCarDriverTopology.java

파일럿 컴포넌트의 시작/중지 명령

소프트웨어	시작 명령	종료 명령
클러스터 전체	CM 홈 → [Cluster 1] → [시작]	CM 홈 → [Cluster 1] → [정지]
Cloudera Management Service 전체	CM 홈 → [Cloudera Management Service] → [시작]	CM 홈 → [Cloudera Management Service] → [정지]
주키퍼	CM 홈 → [Zookeeper] → [시작]	CM 홈 → [Zookeeper] → [정지]
HDFS	CM 홈 → [HDFS] → [시작]	CM 홈 → [HDFS] → [정지]
YARN	CM 홈 → [YARN] → [시작]	CM 홈 → [YARN] → [정지]

소프트웨어	시작 명령	종료 명령
플럼	CM 홈 → [Flume] → [시작]	CM 홈 → [Flume] → [정지]
카프카	CM 홈 → [Kafka] → [시작]	CM 홈 → [Kafka] → [정지]
스톰	# Server02에 SSH로 접속한 후 다음 명령을 실행 service storm-nimbus start service storm-supervisor start service storm-ui start	# Server02에 SSH로 접속한 후 다음 명령을 실행 service storm-nimbus stop service storm-supervisor stop service storm-ui stop
레디스	Server02에 SSH로 접속한 후 다음 명령을 실행 service redis_6379 start	Server02에 SSH로 접속한 후 다음 명령을 실행 service redis_6379 stop
HBase	CM 홈 → [HBase] → [시작]	CM 홈 → [HBase] → [정지]
하이브	CM 홈 → [Hive] → [시작]	CM 홈 → [Hive] → [정지]
우지	CM 홈 → [Oozie] → [시작]	CM 홈 → [Oozie] → [정지]
휴	CM 홈 → [Hue] → [시작]	CM 홈 → [Hue] → [정지]
스파크	CM 홈 → [Spark] → [시작]	CM 홈 → [Spark] → [정지]
임팔라	CM 홈 → [Impala] → [시작]	CM 홈 → [Impala] → [정지]
제플린	# Server02에 SSH로 접속한 후 다음 명령을 실행 $ zeppelin-daemon.sh start	# Server02에 SSH로 접속한 후 다음 명령을 실행 $ zeppelin-daemon.sh stop