Chapter02. 하둡2 설치

목표
2.1 실행 모드 결정
2.2 리눅스 서버 준비
2.3 자바 설치
2.4 하둡 실행 계정 생성
2.5 SSH
2.5 하둡 다운로드
2.7 인코딩 방식 설정
2.8 프로토콜 버퍼 설치
2.9 하둡 실행
2.10 예제 실행
2.11 하둡 종료

목표

- 가상 분산 모드로 2대의 가상머신을 생성
  - 네임노드
  - 데이터 노드
- https://datacodingschool.tistory.com/30 (전체적인 과정에서 참조한 블로그)

2.1 실행 모드 결정

- 독립 실행 모드(Standalone mode)
  - 하둡의 기본 실행 모드. 하둡 환경설정 파일에 아무런 설정을 하지 않고 실행하면 로컬 장비에서만 실행되기 때문에 로컬 모드라고도 함. 하둡에서 제공하는 데몬을 구동하지 않기 때문에 분산 환경을 고려한 테스트는 불가능. 단순히 맵리듀스 프로그램을 개발하고, 해당 맵리듀스를 디버깅하는 용도로만 적합한 모드
- 가상 분산 모드(Pseudo-distributed mode) → 해당 방법으로 설치
  - 하나의 장비에 모든 하둡 환경설정을 하고, 하둡 서비스도 이 장비에서만 제공하는 방식. HDFS와 맵리듀스와 관련된 데몬을 하나의 장비에서만 실행. 주로 하둡을 처음 공부하면 이와 같은 방식으로 테스트 환경을 구성
- 완전 분산 모드(Fully distributed mode)
  - 여러 대의 장비에 하둡이 설치된 경우. 하둡으로 라이브 서비스를 하게 될 경우 이와 같은 방식으로 구성

2.2 리눅스 서버 준비

- 서비스용으로 하둡을 구축한다면 다양한 요소를 고려해야 함
  - I/O가 얼마나 빈번하게 발생하는지, 맵리듀스를 이용한 분석 작업이 CPU에 얼마나 부하를 주는지, 데이터 보관 위주로 하둡을 운영하는지 등
- 보조네임노드는 네임노드가 설치되는 서버와 동일한 사양의 서버에 설치하며, 데이터 노드 같은 다른 데몬을 설치하지 않음

    - 보조네임노드는 네임노드와 동일한 용량의 메모리를 요구하기 때문
    - 네임노드의 장애 발생에 대비하는 용도
- Window에서 Ubuntu 설치
    - https://blog.naver.com/skyshin0304/222079393598 (참고)
- apt-get update && upgrade

```
sudo apt-get update && sudo apt-get upgrade
```

- 기본적인 패키지 설치
    - wget 패키지 설치

    ```
    sudo apt-get install wget
    ```

    - C 컴파일러 설치

    ```
    sudo apt get install build-essential
    ```

    - make 패키지 설치

    ```
    sudo apt-get install make
    ```

2.3 자바 설치

- 하둡을 설치하려면 반드시 자바가 미리 설치돼 있어야 함. 하둡은 자바로 개발됐고, 데몬을 구동할 때 JAR 파일을 수정하기 때문에 반드시 자바가 필요
- 설치 과정
    1. Ubuntu에서 OpenJDK 8 설치
        - apt-get update && upgrade

        ```
        sudo apt-get update && sudo apt-get upgrade
        ```

        - openjdk 8 설치

        ```
        sudo apt-get install —맏openjdk-8-jdk
        ```

        - 잘 설치되었는지 확인

        ```
        java -version
        ```

        - https://codechacha.com/ko/ubuntu-install-open-jdk11/ (참고)

2. Ubuntu가 java를 인식하기 위해 /etc/profile 파일에 자바 경로 관련 환경변수를 등록
    - which java 명령으로 java 명령이 저장된 위치를 찾아주고, "readlink"로 명령이 가리키는 실제 위치를 확인
    - /etc/profile 파일을 엶

```
sudo vi /etc/profile
```

    - 다음 내용을 추가하고 저장
        - esc로 수정 모드 > a 클릭 > 다음 내용 입력 > esc로 수정 모드 종료 > : > wq > enter

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=$JAVA_HOME/lib/tools.jar
```

3. source 명령으로 변경사항을 저장

```
source /etc/profile
```

4. sudo apt-get upgrade로 apt-get 시스템 업그레이드

5. java -version 으로 잘 설치되었는지 확인

```
sudo java -version
```

- 해당 과정을 모든 기기에 반복
- https://m.blog.naver.com/PostView.naver?blogId=yuyyulee&logNo=222091223024&targetKeyword=&targetRecommendationCode=1 (참고)

2.4 하둡 실행 계정 생성

- root 계정으로 하둡을 실행하는 것은 좋지 않음
    - root 계저으로 사용하다가 실수라도 하게 되면 리눅스 시스템 자체를 사용할 수 없는 상태가 될 수도 있기 때문.
- root 계정 로그인 > 필요한 계정을 생성 > sudoers 파일에 사용자를 등록

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
sudo vi /etc/sudoers
```

📝 다음의 파일은 중요한 파일이여서 readonly(즉 읽는것만 가능하다고 나온다.)
이것을 해결하기 위해
":" + "q!" (강제종료로 나가서)

>> sudo bash
>> chattr -i /etc/sudoers
>> chmod u+w /etc/sudoers

다시 실행해서 수정
>> vi /etc/sudoers

#User privilege specification
root ALL=(ALL:ALL) ALL
hduser ALL=(ALL:ALL) ALL (추가해줌)

2.5 SSH

호스트 파일 수정

- SSH로 다른 서버에 접근할 때는 IP 호은 호스트명으로 접속할 수 있음.

📝 **SSH(Secure Shell)란?**
네트워크 상의 다른 컴퓨터에 로그인하거나 원격 시스템에서 명령을 실행하고 다른 시스템으로 파일을 복사할 수 있도록 해 주는 응용 프로그램 또는 그 프로토콜을 가리킨다. 기존의 rsh, rlogin, 텔넷 등을 대체하기 위해 설계되었으며, 강력한 인증 방법 및 안전하지 못한 네트워크에서 안전하게 통신을 할 수 있는 기능을 제공한다. 기본적으로는 22번 포트를 사용한다. SSH는 암호화 기법을 사용하기 때문에, 통신이 노출된다고 하더라도 이해할 수 없는 암호화된 문자로 보인다.

- 모든 방화벽 off

```
iptables -F
```

- 방화벽이 잘 off 되었는지 확인

```
iptables -L
```

- openssh server 설치

```
sudo apt-get update && sudo apt-get upgrade
apt-get install openssh-server
apt-get install openshh-client
apt-get install ssh
```

- 키를 생성해서, 앞으로 들어갈 때는 비밀번호 물어보지 않고 들어가게 설정

```
sudo su hduser -> cd
"""
혹시 안될 경우 추가로 해보기
ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key
ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key
"""
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

- ssh localhost가 잘 작동하는지 확인

```
ssh localhost
```

- 하둡 실행 계정은 하둡 클러스터를 구성하는 모든 서버에 동일하게 생성해야 함
- https://datacodingschool.tistory.com/30 (참고)
- root 계정으로 로그인해서 호스트 정보를 수정함으로써 IP 뿐만 아니라 호스트명을 이용해서도 하둡을 실행할 수 있도록 함(완전분산모드 일시에)

```
hadoop@DESKTOP-SJTS912:/home/leehyerim$ sudo vi /etc/hosts
192.168.56.120 DESKTOP-SJTS912_01
192.168.56.121 DESKTOP-SJTS912_02
192.168.56.122 DESKTOP-SJTS912_03
192.168.56.123 DESKTOP-SJTS912_04
```

2.5 하둡 다운로드

- 버전2.10.1 하둡 다운로드

```
wget https://archive.apache.org/dist/hadoop/core/hadoop-2.10.1/hadoop-2.10.1.tar.gz
```

- wget / unzip 패키지가 설치되어있지 않다면,

```
sudo apt-get install wget
sudo apt-get install unzip
```

- 다운로드 받은 파일의 압축 해제

```
tar -zxf hadoop-2.10.1.tar.gz
```

- 압축 푼 파일의 위치를 옮겨줌

```
su hduser
sudo mv 'hadoop-2.10.1' /usr/local/hadoop
```

- 소유권 변경

```
sudo chown hduser:hadoop -R /usr/local/hadoop
```

- 2개의 가상머신(네임노드, 데이터노드) 생성

```
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
```

- 소유권 변경

```
sudo chown hduser:hadoop -R /usr/local/hadoop_tmp
```

- 어느 경로에 있던 하둡을 실행하기 위해 환경설정 파일을 수정

  >> sudo vi .bashrc

  - .bashrc 가장 밑부분에 다음을 추가

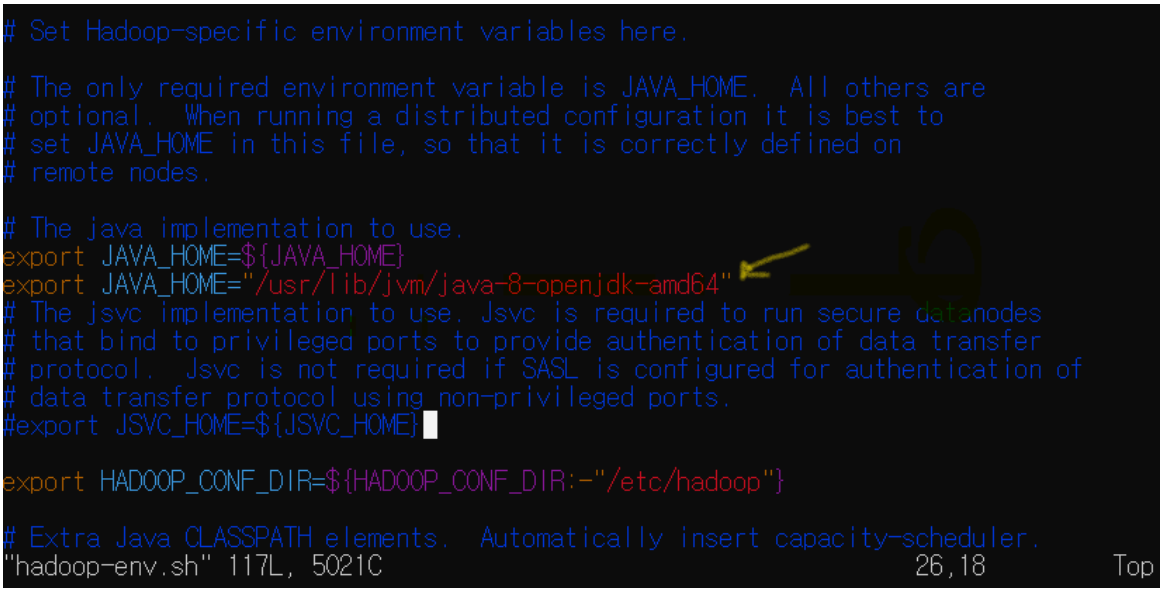
```
export PDSH_RCMD_TYPE=ssh

# -- HADOOP ENVIRONMENT VARIABLES START -- #
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARM_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export PATH=$PATH:/usr/local/hadoop/bin/
```

- 현재 디렉토리 변경

  >> cd /usr/local/hadoop/etc/hadoop

- hadoop 관련 파일 수정

  >> sudo vi hadoop-env.sh /비밀번호



>> sudo vi core-site.xml

>> sudo vi hdfs-site.xml

- dfs.namenode.name.dir : 파일 시스템 이미지를 저장할 로컬 파일 시스템 경로
- dfs.namenode.name.dir : 보조네임노드의 체크포인팅 데이터를 저장할 로컬 파일 시스템 경로
- dfs.datanode.name.dir : HDFS 데이터 블록을 저장할 로컬 파일 시스템 경로.

>> sudo vi yarn-site.xml

- yarn.nodemanager.aux-services : 얀은 노드매니저 간의 서비스 제어를 위해 AuxiliaryService를 제공. 이 속성은 AuxiliaryService로 사용할 서비스 명칭을 정하는 것. 여러 개의 서비스를 설정할 수 있으며, 구분자로 콤마(,) 사용
- yarn.nodemanager.aux-services.#.class : yarn.nodemanager.aux-services를 추가했다면 해당ㅇ 서비스를 구현한 클래스도 함께 설정해야 함. 해당 속성은 yarn.nodemanager.aux-services.서비스명.class이며, 속성값을 서비스를 구현한 클래스를 지정하면 됨. 이 경우 맵리듀스 셔플용으로 제공하는 ShuffleHandler를 설정
- yarn.nodemanager.local-dirs : 노드매니저가 애플리케이션을 실행할 때 필요한 파일을 저장하는 로컬 파일 시스템 경로
- yarn.resourcemanager.fs.state-store.uri : 리소스매니저의 상태 정보를 저장할 로컬 파일 시스템 경로
- yarn.resourcemanager.hostname : 리소스매니저의 호스트명을 정함
- yarn.web-proxy.address : 얀은 애플리케이션마스터에 대한 사용자의 접근을 제어하기 위해 웹 프록시 서버를 제공. 웹 프록시 서버는 리소스매니저가 실행되는 서버와 달느 서버에서 실행될 수 있으며, 포트는 관리자가 시스템 정책에 맞게 설정하면 됨.

```
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
                <value>org.apache.hadoop.mapred.ShuffleHandler</value>
        </property>
        <property>
                <name>yarn.nodemanager.local-dirs</name>
                <value>/usr/local/hadoop/data/yarn/nm-local-dir</value>
        </property>
        <property>
                <name>yarn.resourcemanager.fs.state-store-uri</name>
                <value>/usr/local/hadoop/data/yarn/system/rmstore</value>
        </property>
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>DESKTOP-SJTS912</value>
        </property>
        <property>
                <name>yarn.web-proxy.address</name>
                <value>0.0.0.0:8089</value>
        </property>
</configuration>
~
```

>> cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml

>> sudo vi mapred-site.xml

- 실행모드 : local , yarn
  - local mode : 하나의 JVM만으로 맵리듀스 잡을 실행. 디버깅이나 간단한 테스트 용도에 적합
  - yarn : 맵리듀스 잡을 얀으로 실행하는 것을 의미

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
~
```

- 가상 분산 모드의 경우 네임노드 HA를 구성할 수 없기 때문에, 보조네임노드를 실행해야 함. 하둡2는 master 파일이 생성되어 있지 않기 때무에, 생성해줌
  - master 수정

```
cd /usr/local/hadoop/etc/hadoop
vi masters
DESKTOP-SJTS912
```

- slaves 수정

```
cd /usr/local/hadoop/etc/hadoop
vi slaves
DESKTOP-SJTS912
```

2.7 인코딩 방식 설정

- 하둡은 인코딩 방식으로 UTF-8을 사용
  - 한글로 작성된 파일을 처리할 경우 글자가 깨짐
- echo 명령어로 서버의 인코딩 방식을 확인

```
root@DESKTOP-SJTS912:~# echo $LANG
C.UTF-8
```

- root 계정으로 로그인한 후 아래와 같이 인코딩 방식을 변경

```
root@DESKTOP-SJTS912:~# sudo vi /etc/sysconfig/i18n
LANG="ko_KR.UTF-8"
SUPPORTED="en_US.UTF-8:en_US:ko_KR.eucKR:ko_KR:ko"
SYSFONT="lat0-sun16"
SYSFONTACM="8859-15"
```

- i18n 편집이 끝나면 source 명령어를 이용해 수정한 i18n 파일을 시스템에 적용

```
source /etc/sysconfig/i18n
```

- locale 명렁어로 설정 확인

```
root@DESKTOP-SJTS912:~# locale
```

2.8 프로토콜 버퍼 설치

- 하둡2를 설치하기 위해선 프로토콜 버퍼가 설치돼 있어야 한다. 하둡2는 대부 데몬 간의 데이터 통신을 위해 프로토콜 버퍼를 적용했기 때문
- 프로토콜 버퍼는 구글에서 공개한 오픈소스 직렬화 라이브러리. 바이너리 데이터 방식을 지원하기 위한 프로토콜 버퍼. 프로토콜 버퍼는 데이터를 연속된 비트로 만들고, 이렇게 만들어진 비트를 해석해 원래의 데이터를 만들 수도 있음.
- IT기술의 발달로 이기종 서버 간의 데이터 통신 혹은 서로 다른 종류의 언어로 개발된 시스템 간의 통신이 빈번하게 발생. 이때 데이터를 전달하는 방식은 크게 텍스트 포맷(xml, json 등)을 이용하는 방법과 바이너리(Binary) 데이터를 이용하는 방법이 존재.
  - 텍스트 포맷을 이용하는 방식
    - 장점 : 데이터를 이해하기 쉽고 각 언어별로 여러 종류의 파서가 제공되고 있어 사용하기 편리.
    - 단점 : 데이터 자체의 크기가 크고, 이에 따라 파서의 성능이 떨어진다는 단점

  - Binary 데이터 방식
    - 장점 : 데이터의 크기가 작기 때문에 성능이 더 좋음
    - 단점 : Binary 코드를 만들고 이를 해독해야 하는 모듈을 만들어야 한다는 부담감
- 설치

```
su
cd /usr/local
wget https://github.com/google/protobuf/releases/download/v2.5.0/protobuf-2.5.0.tar.gz
tar xvfz protobuf-2.5.0.tar.gz
cd protobuf-2.5.0
./configure
make
make install
```

- 잘 설치되었는지 확인

```
protoc --version
```

- 오류 발생시

  >> protoc: error while loading shared libraries: libprotoc.so.8: cannot open shared object file: No such file or directory

```
ldconfig
```

```
protoc --version
```

2.9 하둡 실행

- 디렉토리 변경

  >> cd
- 파일 변경 반영

  >> source ~/.bashrc
- 디렉토리 변경

  >> cd /usr/local/hadoop_tmp/hdfs
- 데이터노드 데이터 삭제

  >> rm -rf /usr/local/hadoop_tmp/hdfs/datanode/*
- 네임노드 포맷

  >> hadoop namenode -format
- 데몬 실행

  >> start-dfs.sh

  >> start-yarn.sh

- jps로 잘 실행되었는지 확인

→ 이렇게 나오면 성공!

  - 만약 datanode가 실행이 안됐다면, namenode와 datanode의 cluserID를 일치시켜줌
- 맵리듀스 잡의 이력만 별도로 볼 수 있는 서버 실행

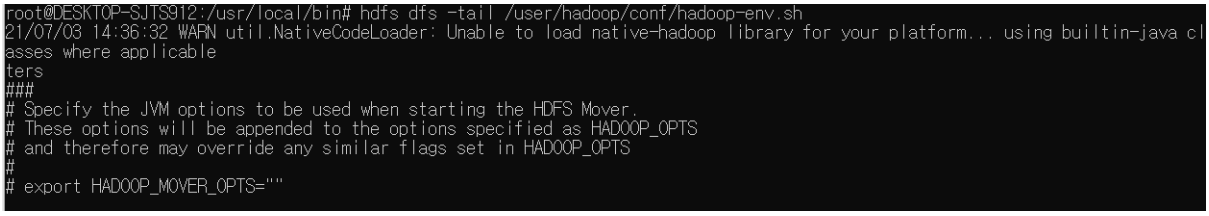
  >> mr-jobhistory-daemon.sh start historyserver
- 웹 프록시 서버 실행

  >> yarn-daemon.sh start proxyserver

2.10 예제 실행

- 하둡의 hadoop-env.sh를 HDFS에 저장

```
hdfs dfs -mkdir /user
hdfs dfs -mkdir /user/hadoop
hdfs dfs -mkdir /user/hadoop/conf
hdfs dfs -put /usr/local/hadoop/etc/hadoop/hadoop-env.sh /user/hadoop/conf
hdfs dfs -tail /user/hadoop/conf
```



2.11 하둡 종료

- 데몬 종료

  >> stop-all.sh
- 히스토리 서버 종료

  >> mr-jobhistory-daemon.sh stop historyserver