

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>



* 설치 OS : CentOS 7

* 계정 및 비번 nova/a1234567890
 root/a1234567890987654321

* HDD 가상화 : 각 서버별 8개 추가하고 Raid 10 구성함

* ~/다운로드 폴더에 다운로드 한 프로그램
JDK 8
Hadoop 3.0.3
Spark 2.3.1

* 각 호스트들을 고정아이피 설정되어 있음

```
# vi /etc/hosts
192.168.100.150 master
192.168.100.151 slave1
192.168.100.152 slave2
192.168.100.153 slave3 backup
```

* 방화벽 : 끄

* ssh 폴더 : 생성되어 있음(700권한)

* VMWare NAT 서브넷 주소 변경(주소 3번째 자리)

vmnetcfg.exe 파일을 C:\Program Files (x86)\VMware\VMware Player 폴더에 복사해
실행시키고 NAT 환경의 Subnet IP 세번째 자리를 100으로 수정해야 함

* VMWare에서 처음 실행할 때 *반드시* **[I Moved it]** 선택해야 함

* 한/영 키 전환 : 컨트롤+스페이스바

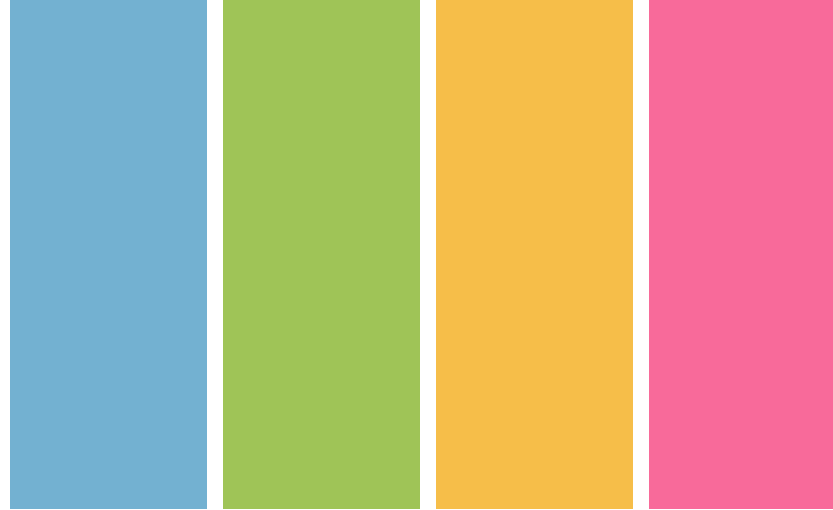
<https://bit.ly/33ISGQk>



하둡 설치 시작 파일

03 하둡 클러스터 구축

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



1. Hadoop의 종류와 구성요소

What is Apache Hadoop?

1. Hadoop 개요

- Apache Hadoop, High-Availability Distributed Object-Oriented Platform
- 대량의 자료를 처리할 수 있는 큰 컴퓨터 클러스터에서 동작하는 신뢰성 있고, 확장성 있는 **분산 컴퓨팅을 위한 오픈소스 프레임워크**
- 2005년 더그 커팅(Doug Cutting)과 마이크 캐퍼렐라(Mike Cafarella)가 개발
- 원래 너치의 분산 처리를 지원하기 위해 개발된 것으로, 아파치 루씬의 하부 프로젝트임
- 간단한 프로그래밍 모델을 사용하여 대용량 데이터의 분산 처리를 할 수 있는 프레임워크
- 분산 파일 시스템 **GFS**, 분산 처리 시스템 **MapReduce 소프트웨어구현체**
 - 아파치 Top-Level 프로젝트
 - 코어는 Java, C/C++, Python 등 지원
- 대용량 데이터 처리를 위한 플랫폼
 - 분산파일시스템(HDFS)
 - 분산병렬처리시스템(MapReduce)
 - 기반소프트웨어프레임워크(Core)



하둡 종류

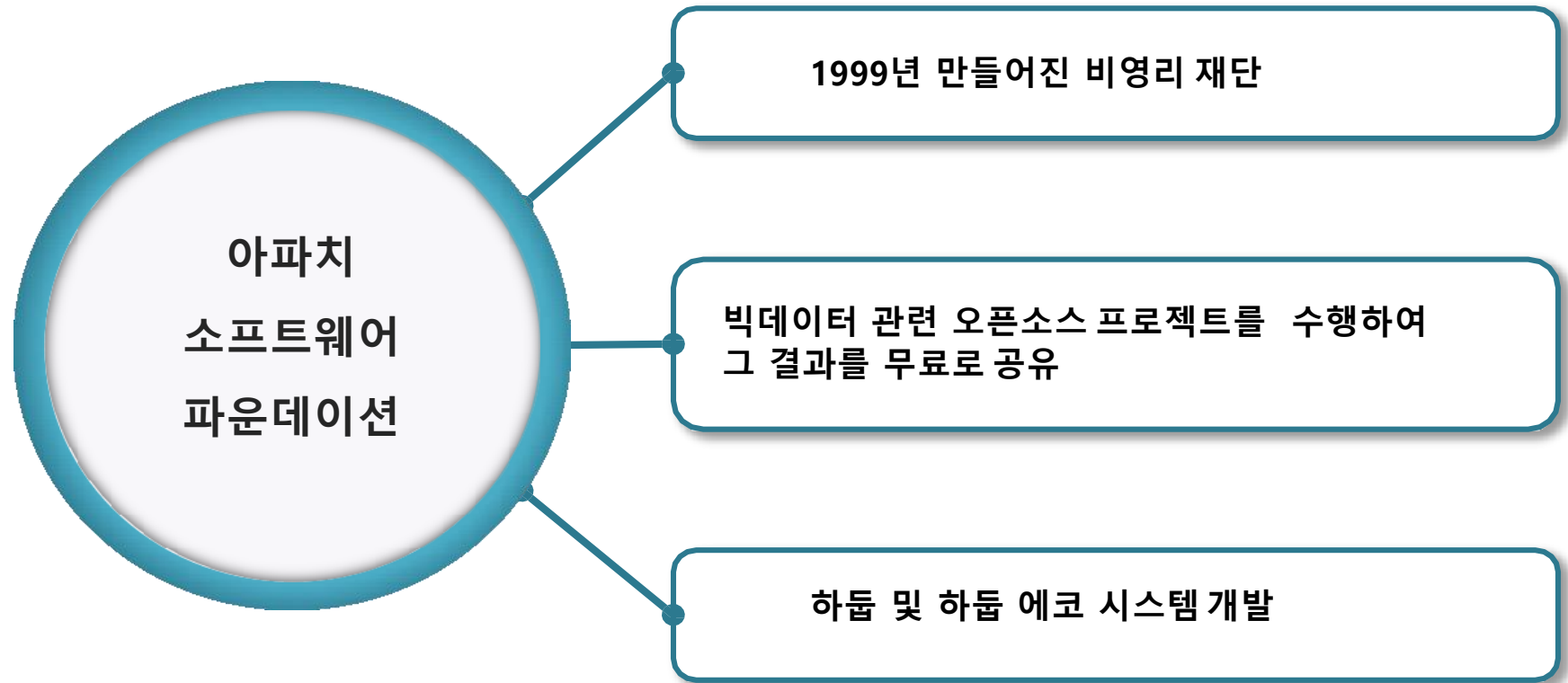
1. Hadoop 개요

- 표준 오픈소스 하둡 배포판 Apache Hadoop
 - 병렬 컴퓨팅을 실행시키는 하둡 맵리듀스(MapReduce)
 - 하둡 분산형 파일 시스템(HDFS: Hadoop Distributed File System)
 - 다른 하둡 모듈이 사용하는 유틸리티와 라이브러리 세트인 하둡 커먼(Hadoop Common)
- 벤더 배포판
 - 클라우데라(Cloudera), 맵알(MapR), 호트웍스(Hortonworks)의 배포판
 - IBM과 인텔(Intel), 피보탈 소프트웨어(Pivotal Software) 등의 하둡 배포판
- 벤더 배포판의 장점
 - 신뢰성 : 벤더들은 버그가 발견됐을 때 더 빨리 대응을 한다. 즉시 픽스와 패치를 배포해, 솔루션을 더 안정적으로 만든다.
 - 지원 : 기업 사명 달성에 아주 중요한 플랫폼 도입과 엔터프라이즈급 작업 처리가 가능하도록 기술 지원 서비스를 제공하는 회사들이 많다.
 - 완성도 : 특정 작업을 처리하기 위해, 다른 툴로 기능을 보강한 하둡 배포판이 아주 많다.

하둡 종류

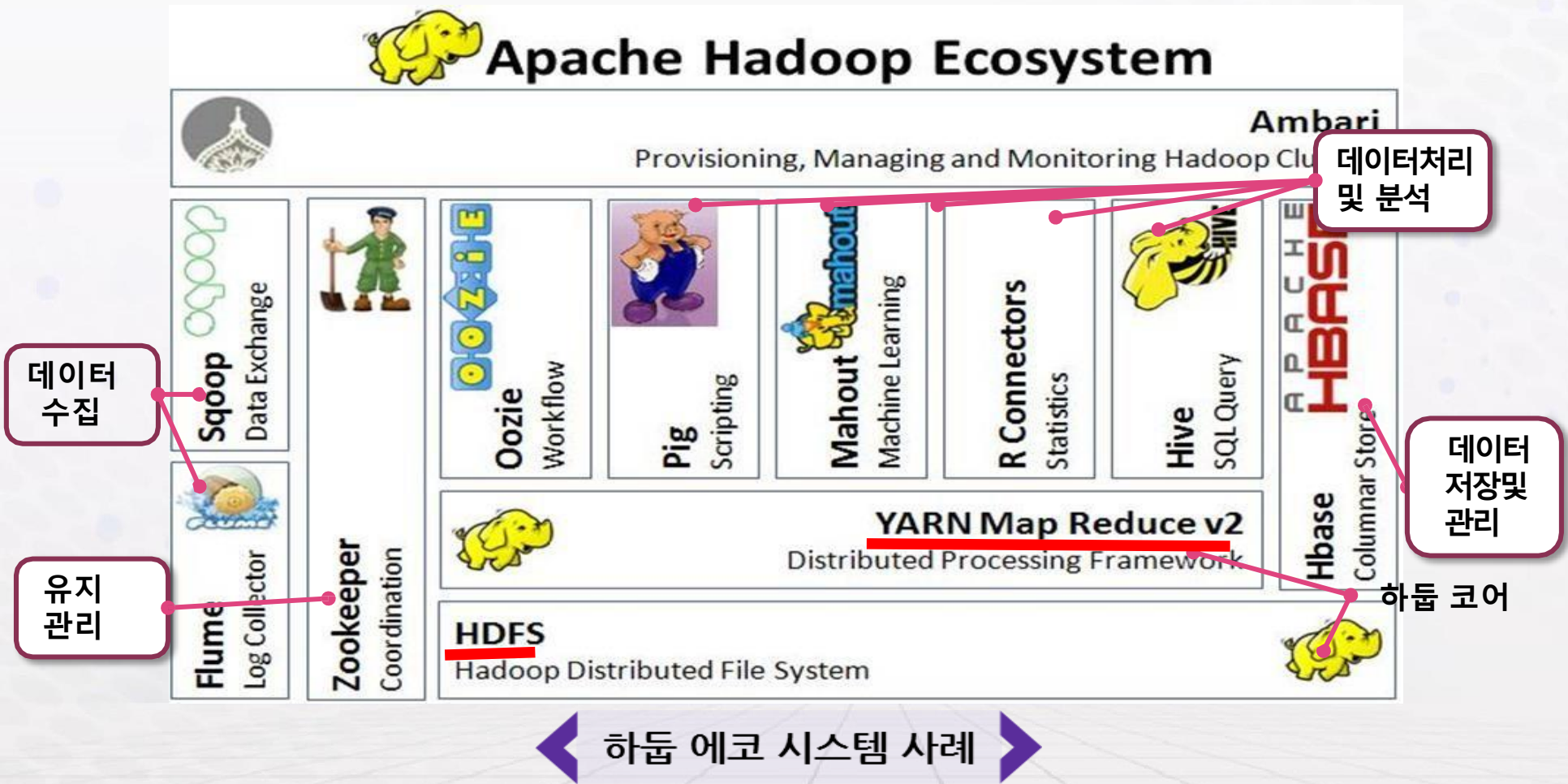
1. Hadoop 개요.

1 아파치 소프트웨어 파운데이션



하둡 종류

1. Hadoop 개요.



하둡 종류

1. Hadoop 개요.

2 클라우데라(Cloudera)



하둡 종류

1. Hadoop 개요.

3 호튼웍스(Hortonworks)



하둡 종류

1. Hadoop 개요.

4 마이크로소프트 애저 (Azure)

통합된 클라우드 가상머신 서비스 플랫폼

분석, 컴퓨팅, 데이터베이스, 모바일, 저장소, 웹 등
IT 인프라 서비스 제공

HDInsight 하둡 클러스터 서비스 제공

하둡 종류

1. Hadoop 개요.

5 아마존(Amazon Web Service)

통합된 클라우드 가상머신 서비스 플랫폼

분석, 컴퓨팅, 데이터베이스, 모바일, 저장소, 웹 등
IT 인프라 서비스 제공

빅데이터 처리를 위한 하둡 맵리듀스 프레임워크를
클라우드 서비스 형태로 제공

HDFS(Hadoop Distributed File System)

1. Hadoop 개요

- 어플리케이션 기반 파일시스템
- 파일의 분산 저장이 목적
- NameNodes와 DataNodes로 구성
 - Master NameNode : 파일시스템 이미지(fsimage)와 변경기록(edits)을 저장
 - Secondary NameNode : Master NameNode의 fsimage 파일과 edits 파일의 사본을 저장
 - DataNode : 데이터파일의 블록을 저장, 디폴트 블록의 크기는 128MB
- 저렴한 컴퓨터로 대용량 데이터를 저장할 수 있는 시스템
 - 네트워크 Raid와 같이 연결된 것 처럼 사용하는 하드디스크
 - Scale Out
- Block(Chunk) 단위로 파일관리 (저장/복제/삭제)
 - Default Size는 128M(134217728)
- 복제 기능을 통해 안전성/신뢰성을 보장
- 1대의 Master서버에 4000+이상의 DataNodes를 운영할 수 있음.
- API지원
 - 하둡 코어는 Python, Java, C/C++

Hadoop 1.x

1. Hadoop 개요

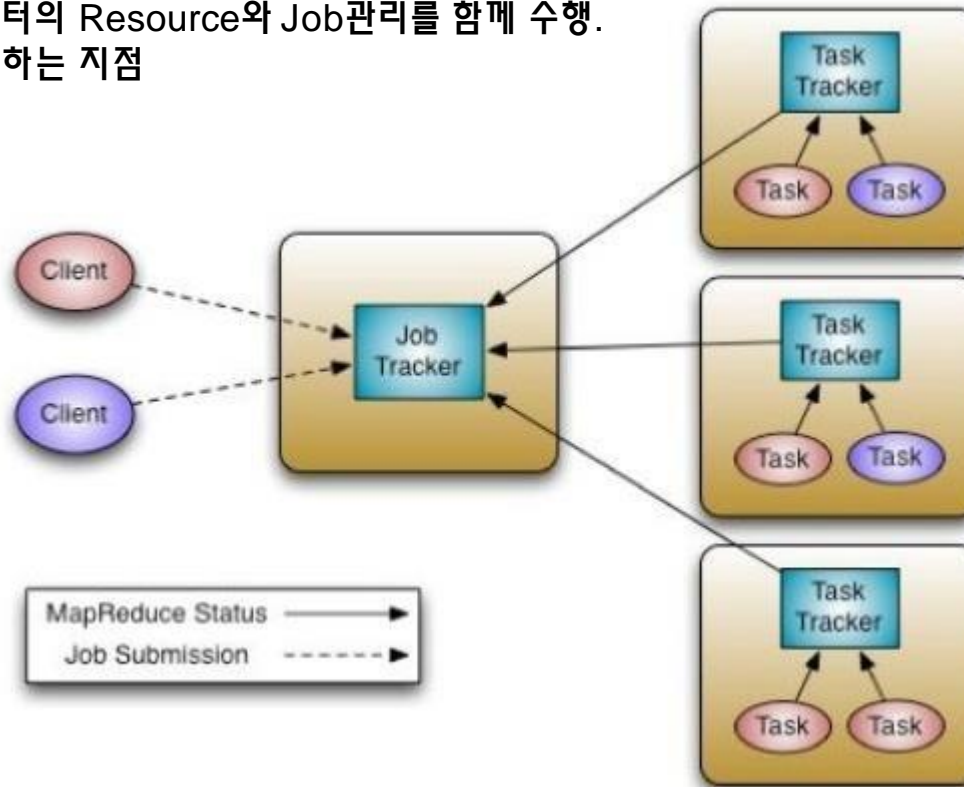
Master Node

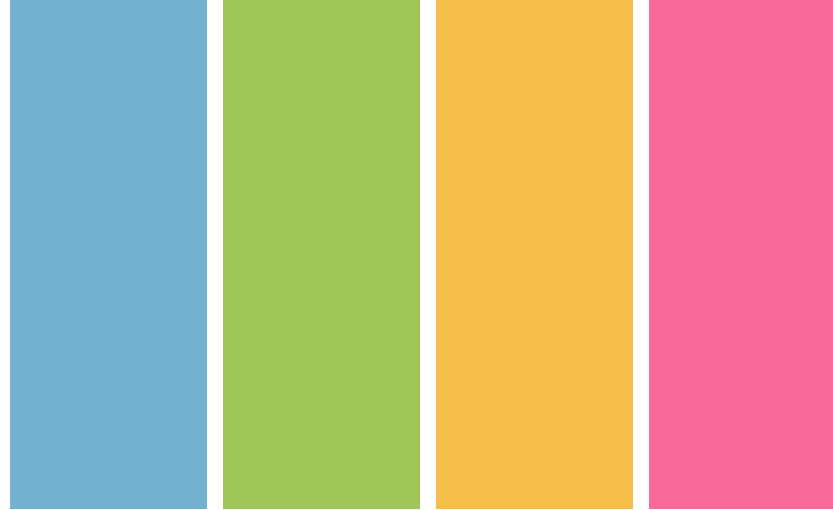
하둡 클러스터의 Resource와 Job관리를 함께 수행.
병목이 발생하는 지점

Data Node

한 노드에서 실행할 수 있는
Map과 Reduce Task의 개수 제한

M/R만 처리





2. 하둡 클러스터 구축을 위한 설정

Full Distributed Mode 하둡 클러스터 설치

하둡 설치를 위한 레이드 구성

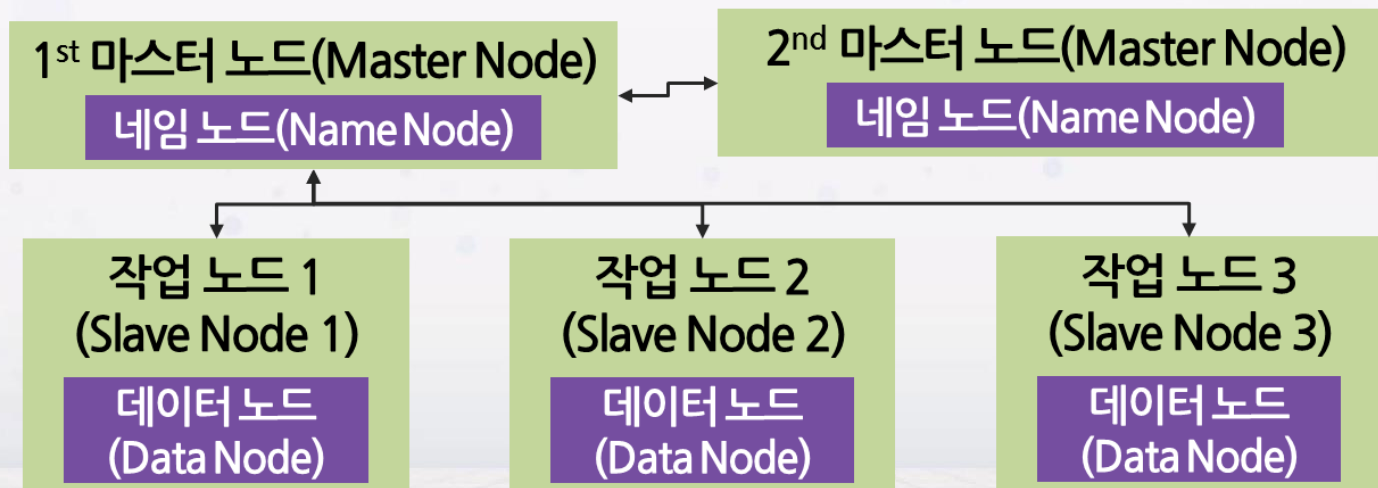
2. Full Distributed Mode 하둡 클러스터 설치

*

완전분산 모드 (Fully distributed Mode)

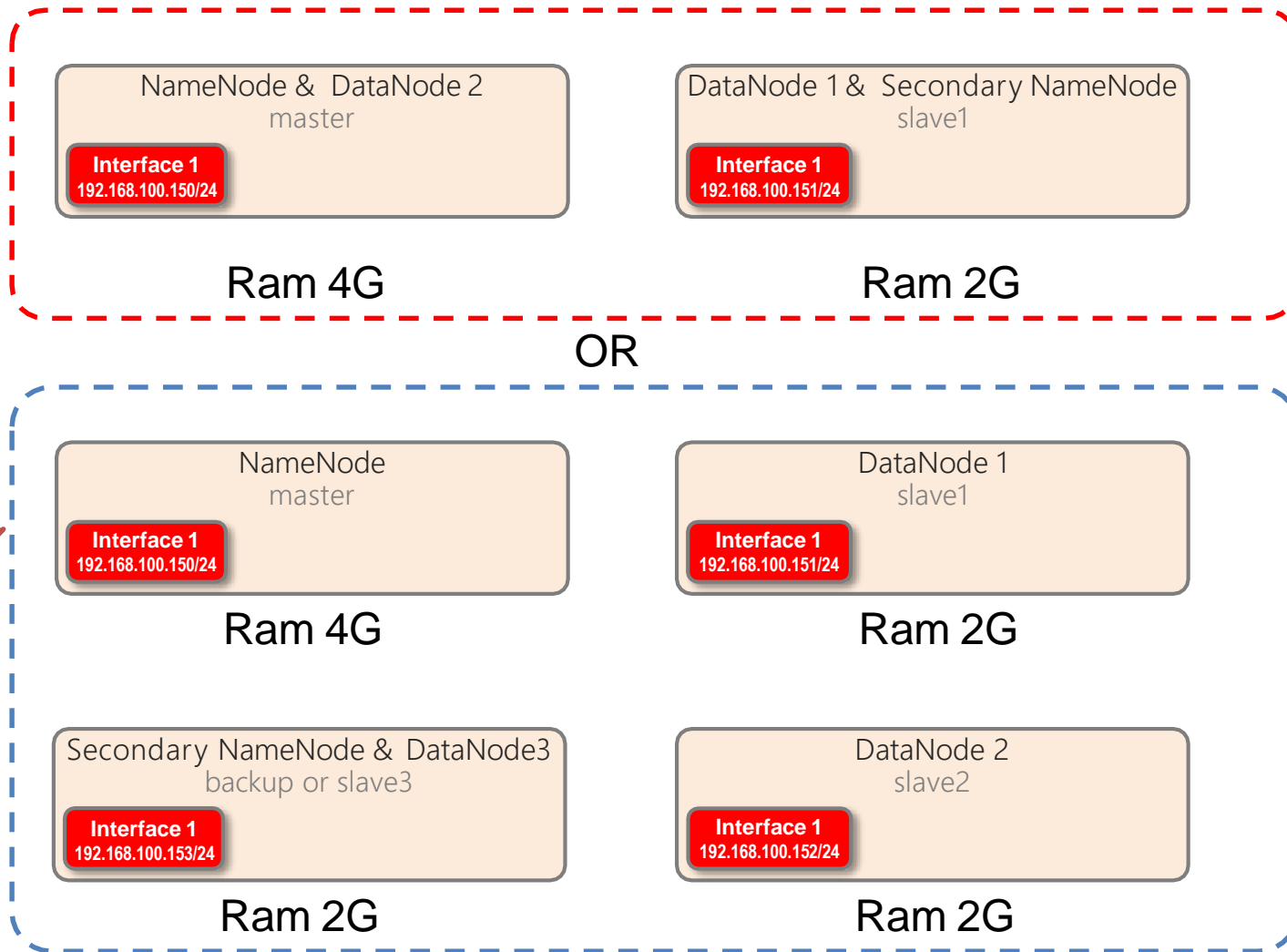


실제 빅데이터 분산 처리 시스템으로 동작하는 환경



PC 메모리가 8GB 또는 16GB라면

2. Full Distributed Mode 하둡 클러스터 설치



우리 과정에서의
실습 환경

모든 노드 : JDK 설치 및 Hadoop 설치

2. Full Distributed Mode 하둡 클러스터 설치

- 한대의 컴퓨터를 이용해 하둡을 설치하고 실행해 볼 수 있다. 여러 대의 컴퓨터를 이용하는 것에 비해 상대적으로 설치가 쉽다.
- 그러나 여러 대의 컴퓨터에 하둡을 설치하려면 모든 노드에 JDK와 하둡이 설치되어 있어야 하며, 한대의 컴퓨터를 이용해 하둡을 설치하는 것에 비해 설정이 어렵다.
- 가상머신 오픈시 “**I Moved It**” 클릭
- `hostname -I`를 통해 각 ip와 `hostname`을 확인
- JDK설치
 - java.sun.com -> Java SE -> JDK 다운로드(jdk-8u181-linux-x64.tar.gz)
\$ cd ~/Downloads
 - \$ cd ~
 - \$ tar -xf ~/다운로드/jdk-8u181-linux-x64.tar.gz
- 하둡 설치
 - hadoop.apache.org -> binary 다운로드(hadoop-3.0.3.tar.gz)
\$ cd ~/Downloads
\$ wget <http://apache.mirror.cdnetworks.com/hadoop/common/hadoop-3.0.3/hadoop-3.0.3.tar.gz>
 - \$ cd ~
 - \$ tar -xf ~/다운로드/hadoop-3.0.3.tar.gz

하둡 클러스터 구축 순서

2. 하둡 클러스터 구축을 위한 설정

- JDK 설치
- Hadoop 설치
- 환경변수 추가
- 네트워크 설정
- 방화벽 설정
- SSH 설정
- Hadoop 설정파일 수정(~/.hadoop 3.0.3/etc/Hadoop)
 - hadoop-env.sh
 - core-site.xml
 - hdfs-site.xml
 - mapred-site.xml
 - yarn-site.xml
 - workers
 - include.hosts
 - exclude.hosts

JDK 설치

- 다운로드
 - <http://java.sun.com> -> Java SE -> Java Platform (JDK)
 - Java SE 1.8 다운로드(Hadoop 2.8은 Java SE 1.7이상 있어야 함)
- 설치(사용자 홈 디렉토리(/home/nova 디렉토리에 설치)
 - \$ **cd ~**
 - \$ **tar -xvf ~/다운로드/jdk-8u181-linux-x64.tar.gz**

www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#); From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day hands-on workshops \(free\) and other events](#)
- [Java Magazine](#)

JDK 8u171 [checksum](#)
JDK 8u172 [checksum](#)

Java SE Development Kit 8u171

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.97 MB	jdk-8u171-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.89 MB	jdk-8u171-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.05 MB	jdk-8u171-linux-i586.rpm
Linux x86	184.88 MB	jdk-8u171-linux-i586.tar.gz
Linux x64	182.05 MB	jdk-8u171-linux-x64.rpm
Linux x64	182.05 MB	jdk-8u171-linux-x64.tar.gz
Mac OS X x64	247.84 MB	jdk-8u171-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.83 MB	jdk-8u171-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.19 MB	jdk-8u171-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.6 MB	jdk-8u171-solaris-x64.tar.Z
Solaris x64	97.05 MB	jdk-8u171-solaris-x64.tar.gz
Windows x86	199.1 MB	jdk-8u171-windows-i586.exe
Windows x64	207.27 MB	jdk-8u171-windows-x64.exe

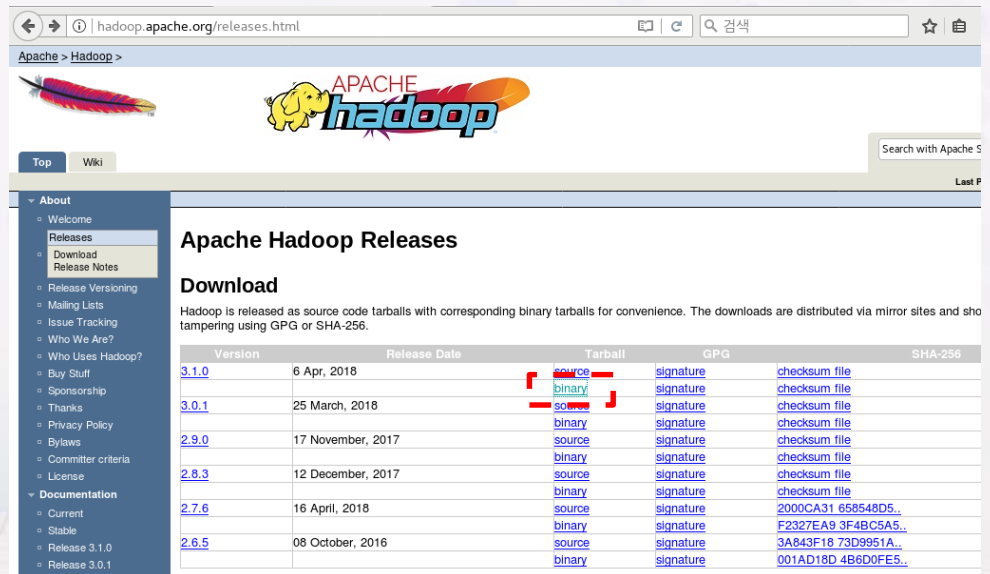
Hadoop 설치

- 다운로드

- <http://hadoop.apache.org/releases.html>
- binary -> hadoop-3.0.3.tar.gz
- 또는
- \$ wget <http://apache.mirror.cdnetworks.com/hadoop/common/hadoop-3.0.3/hadoop-3.0.3.tar.gz>

- 설치(압축 해제)

- \$ cd ~
- \$ tar -xvf ~/다운로드/hadoop-3.0.3.tar.gz



The screenshot shows the Apache Hadoop Releases page. The browser address bar displays 'hadoop.apache.org/releases.html'. The page features the Apache Hadoop logo and a navigation menu on the left. The main content area is titled 'Apache Hadoop Releases' and includes a 'Download' section. Below this, a table lists various Hadoop versions and their corresponding download links for source, binary, and signature files, along with GPG and SHA-256 checksums.

Version	Release Date	Tarball	GPB	SHA-256
3.1.0	6 Apr, 2018	source	signature	checksum file
3.0.1	25 March, 2018	binary	signature	checksum file
2.9.0	17 November, 2017	source	signature	checksum file
2.8.3	12 December, 2017	binary	signature	checksum file
2.7.6	16 April, 2018	source	signature	checksum file
2.6.5	08 October, 2016	binary	signature	checksum file

모든 노드 : .bashrc에 환경변수 설정

2. Full Distributed Mode 하둡 클러스터 설치

- 환경변수 설정(.bashrc 파일에 아래 내용 추가)
 - `$ vi ~/.bashrc`
`export JAVA_HOME=~/.jdk1.8.0_181`
`export PATH=$PATH:$JAVA_HOME/bin`
`export HADOOP_HOME=~/.hadoop-3.0.3`
`export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin`
- 환경변수 적용
 - `$ source ~/.bashrc`
 - source 명령은 명령을 실행한 터미널에만 환경변수가 적용된다.
 - .bashrc 파일을 수정하면 source 명령을 다시 실행해야 한다.
 - 리눅스를 재시작 하면 모든 터미널에 적용된다.
- 설치 확인
 - `$ java -version`
openjdk version "1.8.0_181"
OpenJDK Runtime Environment (build 1.8.0_181-b13)
OpenJDK 64-Bit Server VM (build 25.131-b12, mixed mode)
 - `$ hadoop version`
Hadoop 3.0.3
Source code repository <https://yjzhangal@git-wip-us.apache.org/repos/asf/hadoop.git> -r 37fd7d752db73d984dc31e0cdfd590d252f5e075
Compiled by yzhang on 2018-05-31T17:12Z
Compiled with protoc 2.5.0
From source with checksum 736cdcefa911261ad56d2d120bf1fa
This command was run using /home/nova/hadoop-3.0.3/share/hadoop/common/hadoop-common-3.0.3.jar

호스트명 설정

2. 하둡 클러스터 구축을 위한 설정

- 아이피 주소와 호스트명(도메인) 설정
- root 권한을 가진 사용자로 수정해야 한다.

```
# cat /etc/hosts
```

```
192.168.100.150 master
```

```
192.168.100.151 slave1
```

```
192.168.100.152 slave2
```

```
192.168.100.153 slave3 backup
```



아이피 주소는 다를 수 있습니다.

```
hostname -i  
ip addr show
```


하둡 환경 설정 파일

- 설정파일 위치
 - 1.x : `$HADOOP_HOME/conf`
 - 2.x, 3.x : `$HADOOP_HOME/etc/hadoop`
- `hadoop-env.sh`
 - JDK경로, 클래스 패스, 데몬 실행 옵션 등 설정
- `core-site.xml`
 - HDFS와 맵리듀스에서 공통적으로 사용할 환경정보 설정
- `hdfs-site.xml`
 - HDFS에서 사용할 환경 정보 설정
- `mapred-site.xml`
 - 맵리듀스에서 사용할 환경정보 설정
- `yarn-site.xml`
 - 맵리듀스 프레임워크에서 사용하는 셔플(shuffle) 서비스를 지정
- `include.hosts` & `exclude.hosts`
 - 노드 추가/제거에 사용하는 파일
- `slaves(2.x)` 또는 `workers(3.x)`
 - 데이터 노드 설정

모든 노드 : hadoop-env.sh 파일에 JAVA_HOME 설정

2. Full Distributed Mode 하둡 클러스터 설치

- \$ vi \$HADOOP_HOME/etc/hadoop/hadoop-env.sh

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

```
# Set Hadoop-specific environment variables here.
```

```
# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.
```

```
# The java implementation to use.
```

```
export JAVA_HOME=/home/nova/jdk1.8.0_181
```

```
# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol. Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}
```

```
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}
```

```
...
```

export
JAVA_HOME=\${JAVA_HOME}에
서 \${JAVA_HOME}을 JDK가 설치된
디렉토리로 설정한다.

3.x는 사용자의 환경변수를 읽어 들이므로
설정 안 해도 됨
(설정하면 이 설정이 우선 적용됨)

core-site.xml

- core-default.xml 파일의 설정을 오버라이드 하는 파일
- core-default.xml 파일의 위치는...
 - \$HADOOP_HOME/share/hadoop/common/hadoop-common-3.0.3.jar
- 필수 속성
 - fs.defaultFS
 - hdfs://master:9000
 - 모든 노드에 설정
- 주요 속성
 - hadoop.tmp.dir
 - /home/nova/hadoop-3.0.3/tmp
 - 모든 노드에 설정(디폴트 /tmp/hadoop-\${username})
 - fs.trash.interval
 - 휴지통에 보관할 시간(분), 디폴트 0(바로 영구 삭제함)
 - NameNode에 설정
 - fs.trash.checkpoint.interval
 - 휴지통 체크포인트 시간 간격(분), 디폴트 0(fs.trash.interval과 같은 값을 가짐)
 - NameNode에 설정

모든 노트 : core-site.xml

2. Full Distributed Mode 하둡 클러스터 설치

<http://asq.kr/p7OlvT5EwrSq9Ql>

- \$ vi \$HADOOP_HOME/etc/hadoop/core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software dis-
tributed under the License is distributed on an "AS IS" BASIS, WITHOUT
WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See th
e License for the specific language governing permissions and limitatio
ns under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</valu
e>
  </property>
</configuration>
```

master는 NameNode의
호스트 이름이다.
NameNode의 아이피
주소를 사용할 수 있다.

다른 호스트와 동기화

2. Full Distributed Mode 하둡 클러스터 설치

- Master에 설치되어 있는 **JDK, Hadoop**을 데이터 노드들에 동기화 시키세요.
- 동기화를 이용하면 모든 노드에 프로그램 설치 및 설정을 쉽게 할 수 있습니다.
- JDK 동기화

```
$ rsync -av ~/jdk1.8.0_181/ slave1:~/jdk1.8.0_181/
$ rsync -av ~/jdk1.8.0_181/ slave2:~/jdk1.8.0_181/
$ rsync -av ~/jdk1.8.0_181/ slave3:~/jdk1.8.0_181/
```

- .bashrc 파일 동기화

```
$ rsync -av ~/.bashrc slave1:~/.bashrc
$ rsync -av ~/.bashrc slave2:~/.bashrc
$ rsync -av ~/.bashrc slave3:~/.bashrc
```

- 하둡 동기화

```
$ rsync -av ~/hadoop-3.0.3/ slave1:~/hadoop-3.0.3/
$ rsync -av ~/hadoop-3.0.3/ slave2:~/hadoop-3.0.3/
$ rsync -av ~/hadoop-3.0.3/ slave3:~/hadoop-3.0.3/
```

```
# cat /etc/hosts
```

```
192.168.100.150 master
```

```
192.168.100.151 slave1
```

```
192.168.100.152 slave2
```

```
192.168.100.153 slave3backup
```

```
[nova@master hadoop]$ cd
[nova@master ~]$ rsync -a ~/jdk1.8.0_181/ slave1:~/jdk1.8.0_181/
nova@slave1's password:
[nova@master ~]$ rsync -a ~/jdk1.8.0_181/ slave2:~/jdk1.8.0_181/
nova@slave2's password:
[nova@master ~]$ rsync -a ~/jdk1.8.0_181/ slave3:~/jdk1.8.0_181/
nova@slave3's password:
[nova@master ~]$ rsync -a ~/hadoop-3.0.3/ slave1:~/hadoop-3.0.3/
nova@slave1's password:
[nova@master ~]$ rsync -a ~/hadoop-3.0.3/ slave2:~/hadoop-3.0.3/
nova@slave2's password:
[nova@master ~]$ rsync -a ~/hadoop-3.0.3/ slave3:~/hadoop-3.0.3/
nova@slave3's password:
[nova@master ~]$ rsync -a ~/.bashrc slave1:~/.bashrc
nova@slave1's password:
[nova@master ~]$ rsync -a ~/.bashrc slave2:~/.bashrc
nova@slave2's password:
[nova@master ~]$ rsync -a ~/.bashrc slave3:~/.bashrc
```

hdfs-site.xml

설명

- hdfs-default.xml 파일의 설정을 오버라이드 하는 파일
- hdfs-default.xml 파일의 위치는
 - \$HADOOP_HOME/share/hadoop/hdfs/hadoop-hdfs-3.0.3.jar
- 주요 속성
 - dfs.replication
 - 복제 수, 디폴트 3, NameNode에 설정
 - dfs.namenode.name.dir
 - fsimage(파일시스템이미지)가 저장될 경로, NameNode에 설정
 - dfs.namenode.edits.dir
 - edits(에디트 로그)가 저장될 경로, NameNode에 설정
 - dfs.namenode.secondary.http-address
 - 보조네임노드 주소와 포트번호 지정. 디폴트는 0.0.0.0:9860, NameNode에 설정(2.x포트는 50090)
 - dfs.datanode.data.dir
 - 데이터파일이 저장될 경로, DataNode에 설정
 - dfs.namenode.checkpoint.dir
 - fsimage, edits 사본이 저장될 경로(보조 네임노드), Secondary NameNode에 설정

NameNode : hdfs-site.xml

2. Full Distributed Mode 하둡 클러스터 설치

<http://asq.kr/p7OlvT5EwrSq9Ql>

- \$ vi \$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///dfs/name</value>
  </property>
  <property>
    <name>dfs.namenode.edits.dir</name>
    <value>file:///dfs/edits</value>
  </property>
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>backup:9868</value> <!-- hadoop 2.x는 50090 -->
  </property>
</configuration>
```

디폴트 복제 수

fsimage 파일이 저장 될 디렉토리

edits 파일이 저장될 디렉토리

보조 네임노드
지정

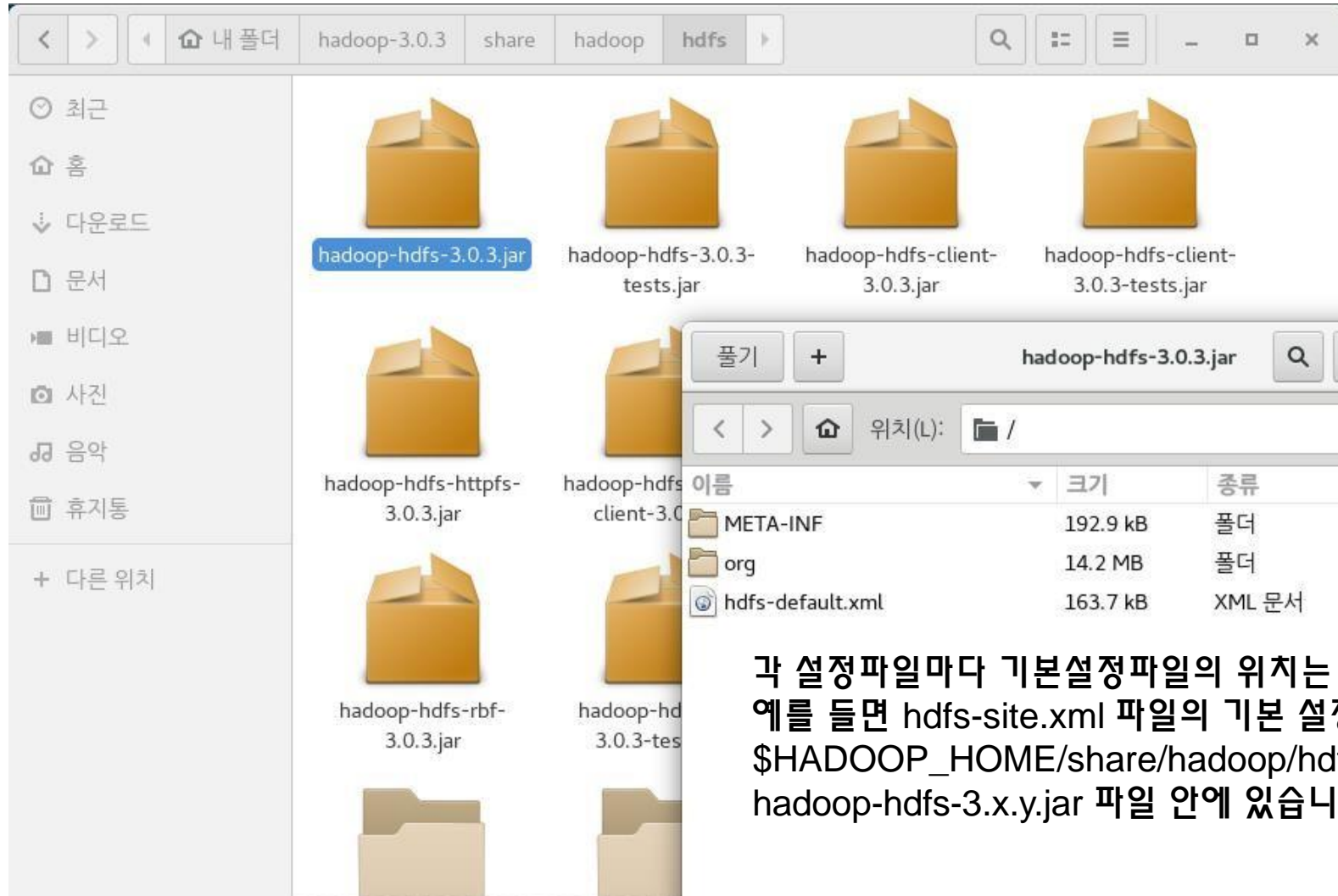
slave3를 Secondary NameNode로
사용하려면 slave3 이라고 입력한다.

* master 노드에 /dfs 폴더 아래에 name, edits 디렉토리가 있어야 합니다.

```
[root@master ~]$ su -
암호:
[root@master ~]# cd /dfs/ [ro
ot@master dfs]# mkdir name [r
oot@master dfs]# mkdir edits
[root@master dfs]# chown -R nova:nova /dfs/name/
[root@master dfs]# chown -R nova:nova /dfs/edits/
[root@master dfs]# ls -l
합계 28
drwxr-xr-x. 2 nova nova 4096 7월 31 12:35 edits
drwx----- 2 root root 16384 4월 23 16:43 lost+found
drwxr-xr-x. 2 nova nova 4096 7월 31 12:35 name
```

기본 설정은 어디에 있죠?

2. Full Distributed Mode 하둡 클러스터 설치



mapred-site.xml

설명

- mapred-default.xml 파일의 속성을 오버라이드 하는 파일
- mapred-default.xml 파일의 위치는
 - \$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.0.3.jar
- 필수 속성
 - mapreduce.framework.name
 - yarn
 - NameNode에 설정

NameNode : mapred-site.xml

2. Full Distributed Mode 하둡 클러스터 설치

- \$ cd \$HADOOP_HOME/etc/hadoop/
- \$ vi mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

맵리듀스
프레임워크 이름

yarn-site.xml

설명

- yarn-default.xml 파일의 속성을 오버라이드 하는 파일
- yarn-default.xml 파일의 위치
 - \$HADOOP_HOME/share/hadoop/yarn/hadoop-yarn-common-3.0.3.jar
- 주요 속성
 - yarn.resourcemanager.hostname
 - 디폴트는 0.0.0.0
 - NodeManager 노드에 설정
 - yarn.nodemanager.aux-services
 - 셔플 서비스 이름
 - mapreduce_shuffle
 - NodeManager 호스트를 별도로 운영할 경우 지정
 - ResourceManager 노드에 설정

NameNode : yarn-site.xml

<http://asq.kr/p7Olvt5EwrSq9Ql>

2. Full Distributed Mode 하둡 클러스터 설치

- \$ vi \$HADOOP_HOME/etc/hadoop/yarn-site.xml

```
<?xml version="1.0"?>
```

```
<configuration>
```

```
  <property>
```

```
    <name>yarn.nodemanager.aux-services</name>
```

```
    <value>mapreduce_shuffle</value>
```

```
  </property>
```

```
</configuration>
```

맵리듀스의 리듀스 서비스가
사용할 보조 서비스

NameNode : 데이터노드 호스트명 설정

2. Full Distributed Mode 하둡 클러스터 설치

<http://asq.kr/p7OlvT5EwrSq9QI>

- 하둡 2.x일 경우
 - \$ vi \$HADOOP_HOME/etc/hadoop/slaves
slave1
slave2
slave3
- 하둡 3.일 경우
 - \$ vi \$HADOOP_HOME/etc/hadoop/workers
slave1
slave2
slave3

localhost는
삭제하고 입력

localhost는
삭제하고 입력

NameNode : SSH 키 생성

2. Full Distributed Mode 하둡 클러스터 설치

- [nova@master ~]\$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nova/.ssh/id_rsa): [enter]
Enter passphrase (empty for no passphrase): [enter]
Enter same passphrase again: [enter]
Your identification has been saved in /home/nova/.ssh/id_rsa.
Your public key has been saved in /home/nova/.ssh/id_rsa.pub.
The key fingerprint is:
9b:38:7a:f8:41:aa:26:bf:1e:c3:57:3e:44:29:36:f3 nova@master
The key's randomart image is:
+--[RSA 2048]-----+

|
| .
| = o
| . *
| E S
| . * . o
| + o.* o
| . . = . . +
| == . . o .
+-----+

• \$ cd ~/.ssh/
• \$ cp id_rsa.pub authorized_keys
• \$ ls
authorized_keys id_rsa id_rsa.pub

키 지문과 이
미지는 다를 수
있습니다.

SSH 공개키 사본 복사

2. Full Distributed Mode 하둡 클러스터 설치

- Master NameNode에서 공개키 사본을 다른 노드에 복사

```
$ scp ~/.ssh/id_rsa.pub slave1:~/.ssh/authorized_keys
$ scp ~/.ssh/id_rsa.pub slave2:~/.ssh/authorized_keys
$ scp ~/.ssh/id_rsa.pub slave3:~/.ssh/authorized_keys
$ scp ~/.ssh/id_rsa.pub backup:~/.ssh/authorized_keys
```

파일을 복사할 때에는
대상 호스트의 비
밀번호를 묻습니다.

slave1~3 노드에 .ssh 디렉토리가
생성되어 있어야 합니다. 이 디
렉토리의 권한은 700여야 합니다.

```
[nova@master ~]$ scp ~/.ssh/id_rsa.pub slave1:~/.ssh/authorized_keys
nova@slave1's password:
id_rsa.pub                                100% 393   559.5KB/s   00:00
[nova@master ~]$ scp ~/.ssh/id_rsa.pub slave2:~/.ssh/authorized_keys
nova@slave2's password:
id_rsa.pub                                100% 393   312.1KB/s   00:00
[nova@master ~]$ scp ~/.ssh/id_rsa.pub slave3:~/.ssh/authorized_keys
nova@slave3's password:
id_rsa.pub                                100% 393   438.2KB/s   00:00
[nova@master ~]$
```

ssh 연결 확인

2. Full Distributed Mode 하둡 클러스터 설치

```
[nova@master ~]$ ssh slave1 date
2018. 08. 02. (목) 21:15:25 KST
[nova@master ~]$ ssh slave2 date
2018. 08. 02. (목) 21:15:28 KST
[nova@master ~]$ ssh slave3 date
2018. 08. 02. (목) 21:15:31 KST
[nova@master ~]$
```

비밀번호를 묻지
않고 현재 시간이
출력되어야 합니다.

```
The authenticity of host 'backup (192.168.100.153)' can't be established. E
CDSA key fingerprint is SHA256:0/cx51VwhfGqTHhWR8/3U2B2UPXvZkZhVhF0bocuxjg.
ECDSA key fingerprint is MD5:7d:f1:3b:f8:53:a6:14:1a:2d:2c:82:54:b2:cb:c1:b
3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'backup' (ECDSA) to the list of known hosts.
2018. 07. 31. (화) 12:51:06 KST
```

처음 한번은 known host로
추가할지 여부를 입력해야
합니다. 위와 같은 메시지가
출력되면 yes를 입력하세요.

DataNode : hdfs-site.xml

<http://asq.kr/p7OlvT5EwrSq9Ql>

2. Full Distributed Mode 하둡 클러스터 설치

- \$ vi \$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///dfs/data</value>
  </property>
</configuration>
```

데이터 블록이
저장될 디렉토리

* slave1~slave3 노드에 /dfs 폴더 아래에 data 디렉토리가 있어야 합니다.

```
[root@slave1 ~]$ su -
```

암호:

```
[root@slave1 ~]# cd /dfs/
```

```
[root@slave1 dfs]# mkdir data
```

```
[root@slave1 dfs]# chown -R nova:nova /dfs/data/
```

```
[root@slave1 dfs]# ll
```

합계 20

```
drwxr-xr-x. 2 nova nova 4096 7월 31 12:28 data
```

```
drwx----- . 2 root root 16384 4월 24 09:40 lost+found
```

DataNode : yarn-site.xml

2. Full Distributed Mode 하둡 클러스터 설치

<http://asq.kr/p7OlvT5EwrSq9Ql>

- \$ vi \$HADOOP_HOME/etc/hadoop/yarn-site.xml

```
<?xml version="1.0"?>
```

```
<configuration>
```

```
<!-- Site specific YARN configuration properties -->
```

```
<property>
```

```
<name>yarn.resourcemanager.hostname</name>
```

```
<value>master</value>
```

리소스 매니저
호스트 이름을 지정

```
</property>
```

```
</configuration>
```

* slave1 설정 후 slave2, slave3에 동기화 할 수 있습니다.

```
[nova@slave1 ~]$ rsync -a ~/hadoop-3.0.3/etc/hadoop/ slave2:~/hadoop-3.0.3/etc/hadoop/
The authenticity of host 'slave2 (192.168.100.152)' can't be established.
ECDSA key fingerprint is SHA256:R8R0dUf1z1WjMYt+BQNI9QYk1UVb81DvJfNJdydeDg.
ECDSA key fingerprint is MD5:13:1c:14:af:18:cd:02:36:2f:9d:89:a8:85:b7:c5:53.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave2,192.168.100.152' (ECDSA) to the list of known hosts.
nova@slave2's password:
[nova@slave1 ~]$ rsync -a ~/hadoop-3.0.3/etc/hadoop/ slave3:~/hadoop-3.0.3/etc/hadoop/
The authenticity of host 'slave3 (192.168.100.153)' can't be established. EC
DSA key fingerprint is SHA256:0/cx51VwhfGqTHhWR8/3U2B2UPXvZkZhVhF0bocuxjg. E
CDSA key fingerprint is MD5:7d:f1:3b:f8:53:a6:14:1a:2d:2c:82:54:b2:cb:c1:b3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave3,192.168.100.153' (ECDSA) to the list of known hosts.
nova@slave3's password:
[nova@slave1 ~]$
```

Secondary NameNode : hdfs-site.xml

2. Full Distributed Mode 하둡 클러스터 설치

- \$ vi \$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>dfs.namenode.checkpoint.dir</name>
    <value>file:///dfs/namesecondary</value>
  </property>
</configuration>
```

fsimage, edits 사본 파일이
저장될 디렉토리

slave3을 Secondary
NameNode로 사용
하려면 slave3의 설
정파일에 포함시켜야
합니다.

* backup노드에 /dfs 폴더 아래에 namesecondary 디렉토리가 있어야 합니다.

```
[root@backup ~]$ su -
```

암호:

```
[root@backup ~]# cd /dfs/
```

```
[root@backup dfs]# mkdir namesecondary
```

```
[root@backup dfs]# chown -R nova:nova /dfs/namesecondary/
```

```
[root@backup dfs]# ll
```

합계 20

```
drwxr-xr-x. 2 nova nova 4096 7월 31 12:28 data
```

```
drwx----- 2 root root 16384 4월 24 09:40 lost+found
```

```
drwxr-xr-x. 2 nova nova 4096 7월 31 12:39 namesecondary
```

NameNode 포맷

2. Full Distributed Mode 하둡 클러스터 설치

\$ hdfs namenode -format 포맷은 절대 1번만 해야지 두번 이상 하면 다시 처음부터 시작
포맷할 때 id가 생성되는데 다시 포맷하면 id가 바뀌어 기존의 data 못읽음

```
16/10/01 01:34:45 INFO namenode.NameNode: STARTUP_MSG:
WARNING: /home/nova/hadoop-3.0.3/logs does not exist. Creating.
2018-08-02 21:16:33,967 INFO namenode.NameNode: STARTUP_MSG:
***** STARTUP_MSG: Starting NameNode STARTUP_MSG: host = master/192.168.100.150
STARTUP_MSG: args = [-format] STARTUP_MSG: version = 3.0.3
STARTUP_MSG: classpath = /home/nova/hadoop-3.0.3/etc/hadoop:...
STARTUP_MSG: build = https://vishangal@git-wip-us.apache.org/repos/asf/hadoop.git -r 37fd7d752db73d984dc31e0cddf590d252f5e075; compiled by 'yizhang' on 2018-05-31T17:12Z STARTUP_MSG: java = 1.8.0_181

2018-08-02 21:16:33,975 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT] 2018-08-02 21:16:33,982 INFO namenode.NameNode: createNameNode[-format]
Formatting using clusterid: CID-6667faee-1c10-4f63-9ebc-334c8d05a7e7
2018-08-02 21:16:34,663 INFO namenode.FSEditLog: Edit logging is async:true 2018-08-02 21:16:34,678 INFO namenode.FSNamesystem: KeyProvider: null 2018-08-02 21:16:34,679 INFO namenode.FSNamesystem: fsLock is fair:true
2018-08-02 21:16:34,684 INFO namenode.FSNamesystem: Detailed lock hold time metrics enabled: false
2018-08-02 21:16:34,689 INFO namenode.FSNamesystem: fsOwner = nova(auth:SIMPLE) 2018-08-02 21:16:34,689 INFO namenode.FSNamesystem: supergroup = supergroup
2018-08-02 21:16:34,689 INFO namenode.FSNamesystem: isPermissionEnabled = true 2018-08-02 21:16:34,689 INFO namenode.FSNamesystem: HA Enabled: false
2018-08-02 21:16:34,729 INFO common.Util: dfs.datanode.fileio.profiling.sampling.percentage set to 0. Disabling file IO profiling
```

```
2018-08-02 21:16:34,740 INFO blockmanagement.DatanodeManager: dfs.block.invalidate.limit: configured=1000, counted=60, effected=1000
2018-08-02 21:16:34,740 INFO blockmanagement.DatanodeManager: dfs.namenode.registration.ip-hostname-check=true
2018-08-02 21:16:34,746 INFO blockmanagement.BlockManager: dfs.namenode.startup.delay.block.deletion.sec is set to 000:00:00.000
2018-08-02 21:16:34,746 INFO blockmanagement.BlockManager: The block deletion will start around 2018-8월 21:16:34 2
018-08-02 21:16:34,747 INFO util.GSet: Computing capacity for map BlocksMap
2018-08-02 21:16:34,748 INFO util.GSet: VM type = 64-bit
2018-08-02 21:16:34,749 INFO util.GSet: 2.0% max memory 839.5 MB = 16.8 MB
2018-08-02 21:16:34,749 INFO util.GSet: capacity = 2^21 = 2097152 entries
2018-08-02 21:16:34,824 INFO blockmanagement.BlockManager: dfs.block.access.token.enable = false
2018-08-02 21:16:34,828 INFO Configuration.deprecation: No unit for dfs.namenode.safemode.extension(30000) assumingMILLISECONDS
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManagerSafeMode: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManagerSafeMode: dfs.namenode.safemode.min.datanodes = 0
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManagerSafeMode: dfs.namenode.safemode.extension = 30000
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManager: defaultReplication = 2
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManager: maxReplication = 512
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManager: minReplication = 1
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManager: maxReplicationStreams = 2
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManager: redundancyRecheckInterval = 3000ms 2
2018-08-02 21:16:34,829 INFO blockmanagement.BlockManager: maxNumBlocksToLog = 1000
2018-08-02 21:16:34,885 INFO util.GSet: Computing capacity for map INodeMap
2018-08-02 21:16:34,885 INFO util.GSet: VM type = 64-bit
2018-08-02 21:16:34,886 INFO util.GSet: 2.0% max memory 839.5 MB = 8.4 MB
2018-08-02 21:16:34,886 INFO util.GSet: capacity = 2^20 = 1048576 entries 20
2018-08-02 21:16:34,887 INFO namenode.FSDirectory: ACLs enabled? false
2018-08-02 21:16:34,887 INFO namenode.FSDirectory: POSIX ACL inheritance enabled? true
2018-08-02 21:16:34,888 INFO namenode.FSDirectory: Capturing file state is occurring more than 10 times
2018-08-02 21:16:34,893 INFO snapshot.SnapshotManager: Loaded config captureOpenFiles: false, skipCaptureAccessTimeOnlyChange: false, snapshotDiffAllowSnapRootDescendant: true 2
018-08-02 21:16:34,896 INFO util.GSet: Computing capacity for map cachedBlocks
2018-08-02 21:16:34,896 INFO util.GSet: VM type = 64-bit
2018-08-02 21:16:34,897 INFO util.GSet: 0.25% max memory 839.5 MB = 2.1 MB
2018-08-02 21:16:34,907 INFO util.GSet: capacity = 2^18 = 262144 entries
2018-08-02 21:16:34,914 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.window.num.buckets = 10
2018-08-02 21:16:34,914 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.num.users = 10
2018-08-02 21:16:34,914 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
2018-08-02 21:16:34,917 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
2018-08-02 21:16:34,917 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis 2
018-08-02 21:16:34,918 INFO util.GSet: Computing capacity for map NameNodeRetryCache
2018-08-02 21:16:34,918 INFO util.GSet: VM type = 64-bit
2018-08-02 21:16:34,919 INFO util.GSet: 0.025000000000000000% max memory 839.5 MB = 257.9 KB
2018-08-02 21:16:34,919 INFO util.GSet: capacity = 2^15 = 32768 entries
2018-08-02 21:16:34,947 INFO namenode.FSImage: Allocated new FSImagePoolId: BP-2130602555-192.168.100.150-1533212194940
2018-08-02 21:16:34,975 INFO common.Storage: Storage directory /dfs/name has been successfully formatted.
2018-08-02 21:16:34,986 INFO common.Storage: Storage directory /dfs/editshas been successfully formatted.
2018-08-02 21:16:34,999 INFO namenode.FSImageFormatProtobuf: Saving image file /dfs/name/current/fsimage.ckpt_00000000000000000000 using no compression
2018-08-02 21:16:35,117 INFO namenode.FSImageFormatProtobuf: Image file /dfs/name/current/fsimage.ckpt_00000000000000000000 of size 389 bytes saved in 0 seconds . 2
018-08-02 21:16:35,126 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2018-08-02 21:16:35,126 INFO namenode.NameNode: SHUTDOWN_MSG:
G: Shutting down NameNode at master/192.168.100.150
/ [nova@master ~]$
```

포맷 하고 확인

```
[nova@master ~]$ cd /dfs/name/current/
```

```
[nova@master current]$ ls
```

```
fsimage_00000000000000000000 seen_txid
fsimage_00000000000000000000.md5 VERSION
```

클러스터 데몬 실행

2. Full Distributed Mode 하둡 클러스터 설치

데몬	시작 명령	종료 명령
모든 데몬	<code>start-all.sh</code>	<code>stop-all.sh</code>
파일시스템	<code>start-dfs.sh</code>	<code>stop-dfs.sh</code>
안	<code>start-yarn.sh</code>	<code>stop-yarn.sh</code>
네임노드	<code>hadoop-daemon.sh start namenode</code> <code>hdfs --daemon start namenode</code>	<code>hadoop-daemon.sh stop namenode</code> <code>hdfs --daemon stop namenode</code>
보조네임노드	<code>hadoop-daemon.sh start secondarynamenode</code> <code>hdfs --daemon start secondarynamenode</code>	<code>hadoop-daemon.sh stop secondarynamenode</code> <code>hdfs --daemon stop secondarynamenode</code>
모든 데이터노드	<code>hadoop-daemons.sh start datanode</code> <code>hdfs --workers --daemon start datanode</code>	<code>hadoop-daemons.sh stop datanode</code> <code>hdfs --workers --daemon stop datanode</code>
데이터노드	<code>hadoop-daemon.sh start datanode</code> <code>hdfs --daemon start datanode</code>	<code>hadoop-daemon.sh stop datanode</code> <code>hdfs --daemon stop datanode</code>
리소스 매니저	<code>yarn-daemon.sh start resourcemanager</code> <code>yarn --daemon start resourcemanager</code>	<code>yarn-daemon.sh stop resourcemanager</code> <code>yarn --daemon stop resourcemanager</code>
모든 노드 매니저	<code>yarn-daemons.sh start nodemanager</code> <code>yarn --workers --daemon start nodemanager</code>	<code>yarn-daemons.sh stop nodemanager</code> <code>yarn --workers --daemon stop nodemanager</code>
노드 매니저	<code>yarn-daemon.sh start nodemanager</code> <code>yarn --daemon start nodemanager</code>	<code>yarn-daemon.sh stop nodemanager</code> <code>yarn --daemon stop nodemanager</code>

클러스터 실행

2. Full Distributed Mode 하둡 클러스터 설치

- **\$ \$HADOOP_HOME/sbin/start-all.sh**

- yes/no 를 물어보면 yes 입력하세요.

```
[nova@master ~]$ start-all.sh
```

```
WARNING: Attempting to start all Apache Hadoop daemons as nova in 10 seconds.
```

```
WARNING: This is not a recommended production deployment configuration.
```

```
WARNING: Use CTRL-C to abort.
```

```
Starting namenodes on [master]
```

```
master: Warning: Permanently added 'master,192.168.100.150' (ECDSA) to the list of known hosts.
```

```
Starting datanodes
```

```
slave3: WARNING: /home/nova/hadoop-3.0.3/logs does not exist. Creating.
```

```
slave1: WARNING: /home/nova/hadoop-3.0.3/logs does not exist. Creating.
```

```
slave2: WARNING: /home/nova/hadoop-3.0.3/logs does not exist. Creating.
```

```
Starting secondary namenodes [backup]
```

```
backup: Warning: Permanently added 'backup' (ECDSA) to the list of known hosts.
```

```
Starting resourcemanager
```

```
Starting nodemanagers
```

하나씩 실행시키려면... had
oop-daemon.sh start ·hdf
s --daemon start ...

yarn-daemon.sh start ...
yarn --daemon start ...

```
[nova@master hadoop]$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as nova in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [master]
Starting datanodes
Starting secondary namenodes [backup]
Starting resourcemanager
Starting nodemanagers
[nova@master hadoop]$ █
```

두 번째부터 실행 시 로그

실행 확인

2. Full Distributed Mode 하둡 클러스터 설치

```
[nova@master ~]$ jps
21266 Jps
20916 ResourceManager
20534 NameNode
[nova@master ~]$ ssh slave1 jps
1960 Jps
1853 NodeManager
1742 DataNode
[nova@master ~]$ ssh slave2 jps
1955 Jps
1737 DataNode
1849 NodeManager
[nova@master ~]$ ssh slave3 jps
1713 DataNode
1813 SecondaryNameNode
2027 Jps
1902 NodeManager
[nova@master ~]$
```

<http://namenode-ip:50070> 으로 실행 확인(Hadoop 2.x)

<http://namenode-ip:9870> 으로 실행 확인(Hadoop 3.x)

Summary

Security is off.
Safemode is off.
1 files and directories, 0 blocks = 1 total filesystem object(s).
Heap Memory used 51.44 MB of 225 MB Heap Memory. Max Heap Memory is 839.5 MB.
Non Heap Memory used 45.83 MB of 46.96 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

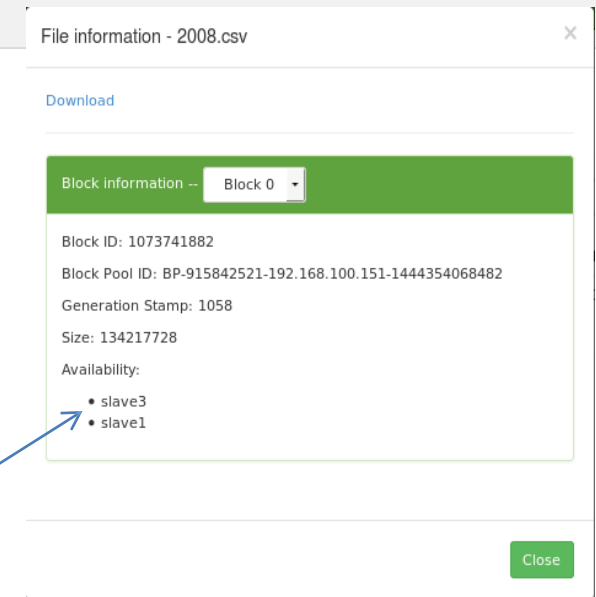
Configured Capacity:	117.64 GB
DFS Used:	72 KB (0%)
Non DFS Used:	144.16 MB
DFS Remaining:	111.45 GB (94.74%)
Block Pool Used:	72 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0

Live Nodes의 값이 3 이어야 함
(DataNode를 2대만 사용할 경우에는 2)

파일 업로드 후 데이터 노드에서 블록 확인

2. Full Distributed Mode 하둡 클러스터 설치

- 데이터 파일 다운로드
 - <https://bit.ly/33ISGQk> 접속 -> 2008.csv.bz2 파일 다운로드
- 압축파일 풀기(/home/nova/Downloads/2008.csv.bz2 압축파일을 풀어 놓는다.)
 - `$ bunzip2 ~/다운로드/2008.csv.bz2ls`
- 하둡 클러스터가 실행중인 상태라면...
 - `$ hdfs dfs -mkdir /airline/`
 - 하둡 클러스터에 /airline 디렉토리를 만듦
 - `$ hdfs dfs -put ~/다운로드/2008.csv /airline/`
 - /airline 디렉토리에 2008.csv 파일을 업로드
 - `$ hdfs dfs -ls /airline`
Found 1 items
-rw-r--r-- 2 hadoop supergroup 689413344 2016-11-09 04:28 /airline/2008.csv
- 업로드 한 파일은 어떤 데이터 노드에 있을까?
 - <http://namenode-ip:50070>
 - <http://namenode-ip:9870>
 - Utilities -> Browser the file system에서 파일 시스템을 웹에서 확인 가능
 - 파일의 블록이 몇 개이며 각 블록들은 어떤 데이터노드에 있는지 확인할 수 있다.
 - 데이터노드의 블록을 확인해 보자.
 - `$ ssh slave1`
 - `$ cd dfs/data/current/BP-xxx/current/finalized/subdir0/subdir0/`
 - `$ ls`



문제가 발생한다면?

2. Full Distributed Mode 하둡 클러스터 설치

- 로그 디렉토리는?
 - `$HADOOP_HOME/logs/hadoop-사용자-노드구분-호스트명.log` 파일을 보자.
 - 예: 네임노드가 실행이 안되면 로그파일 확인...
 - `cat $HADOOP_HOME/logs/hadoop-사용자명-namenode-master.log`
- 네임노드의 파일시스템 이미지는 어디에?
 - `dfs.namenode.name.dir` 속성에 지정된 디렉토리에 저장된다.
 - 이 예제는 `/dfs/name` 디렉토리에 저장된다.
- 데이터는 어디에?
 - `dfs.datanode.data.dir` 속성에 지정된 디렉토리에 저장된다.
 - 이 예제는 `/dfs/data` 디렉토리에 저장된다.
 - 네임노드 포맷 후 다시 포맷했을 경우 데이터노드가 실행이 안되면?
 - `/dfs/data/` 디렉토리 삭제 후 다시 실행
- 보조네임노드의 파일시스템이미지 사본이 저장되는 디렉토리는?
 - `dfs.namenode.checkpoint.dir` 속성에 지정된 디렉토리에 저장된다.
 - 이 예제는 `/dfs/namespacesecondary` 디렉토리에 저장된다.
- Error: JAVA_HOME is not set and could not be found
 - `vi $HADOOP_HOME/etc/hadoop/hadoop-env.sh`
`export JAVA_HOME=/home/nova/jdk1.8.0_181`

HDFS(파일시스템) 명령

2. Full Distributed Mode 하둡 클러스터 설치

- `hdfs dfs` -명령어 -옵션 명령행인자
 - ex) `hdfs dfs -mkdir -p /user/hadoop`
 - `hadoop fs` -명령어 명령행인자 : 1.x 명령
- 명령어

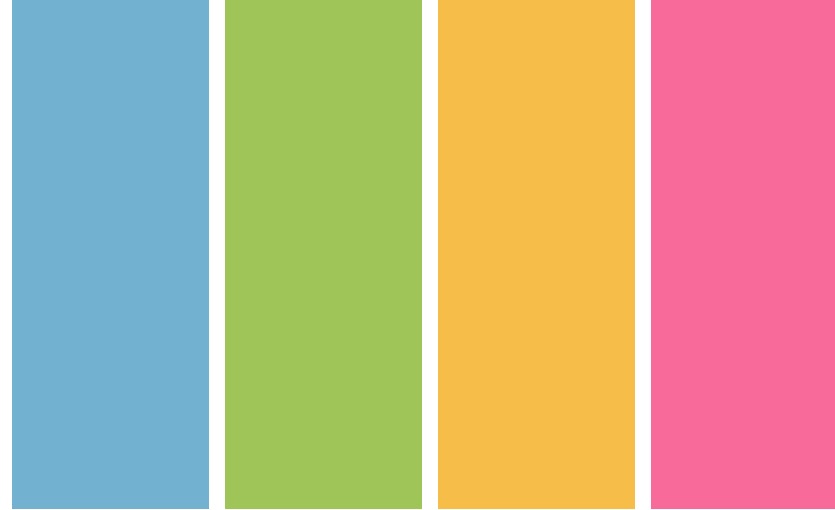
`hadoop fs -ls`

 - `ls [-d][-h][-R]` : 파일 또는 디렉토리 목록
 - `du [-s][-h]` : 파일 용량 확인
 - `cat, text` : 파일 내용 보기
 - `mkdir [-p]` : 디렉토리 생성
 - `put, get` : 파일 복사(로컬 <-> HDFS)
 - `getmerge [-nl]` : 병합해서 로컬에 저장(nl은 각 파일 끝에 개행문자 포함)
 - `cp, mv` : 파일 복사, 이동(HDFS <-> HDFS)
 - `rm [-R][-skipTrash]` : 파일 삭제, 디렉토리 삭제, 완전 삭제
 - `count [-q]` : 카운트 값 조회
 - `tail` : 파일의 마지막 내용 확인
 - `chmod, chown, chgrp` : 권한, 소유주, 그룹 변경
 - `touchz` : 0바이트 파일 생성
 - `stat [-R] <format>` : 통계 정보 조회
 - 포맷 : %b(바이트수) %F(파일인지디렉토리인지) %u(소유주) %g(그룹) %n(이름) %o(블록크기) %r(복제수) %y(날짜 및 시간) %Y(유닉스타임스탬프)
 - `setrep` : 복제 수 변경
 - `expunge` : 휴지통 비우기
 - `test -[edz]` : 파일 형식 확인(empty, zero, dir)

```
[nova@master ~]$ hdfs dfs -rm -R /airline
Deleted /airline
[nova@master ~]$ hdfs dfs -rm -R /user
Deleted /user
```

데이터 파일 : <https://bit.ly/33ISGQk> 접속 -> 2008.csv.bz2
\$ bunzip2 2008.csv.bz2

- 사용자의 홈디렉토리를 생성하세요.
 - 사용자 홈디렉토리에 airline 디렉토리를 생성하세요.
 - airline 디렉토리에 2008.csv 파일을 업로드 하세요.
 - 로컬의 2008.csv 파일을 삭제하세요.
 - HDFS의 2008.csv 파일을 로컬에 저장하세요.
 - airline 디렉토리를 삭제하세요.
 - 루트에 airline 디렉토리를 생성하세요.
 - /airline 디렉토리에 2008.csv 파일을 업로드 하세요.
 - 2008.csv 파일의 처음 5라인을 출력하세요.
 - 2008.csv 파일의 마지막 1KB를 출력하세요.
 - 2008.csv 파일의 통계 정보를 조회하세요.
 - 2008.csv 파일의 복제 데이터 개수를 변경하세요.
 - 2008.csv 파일의 복제 수를 확인하세요.
 - 2008.csv 파일의 복제 수를 1로 변경하세요.
- ```
hdfs dfs -mkdir -p /user/nova
hdfs dfs -mkdir airline
hdfs dfs -put 2008.csv airline/ rm 2008.csv
hdfs dfs -get airline/2008.csv
hdfs dfs -rm -R airline/
hdfs dfs -mkdir /airline
hdfs dfs -put 2008.csv /airline/
hdfs dfs -cat /airline/2008.csv | head -5
hdfs dfs -tail /airline/2008.csv
hdfs dfs -stat "%b %F %n %o %r %y"
hdfs dfs -setrep 1 /airline/2008.csv
hdfs dfs -stat %r /airline/2008.csv
hdfs dfs -setrep 2 /airline/2008.csv
```



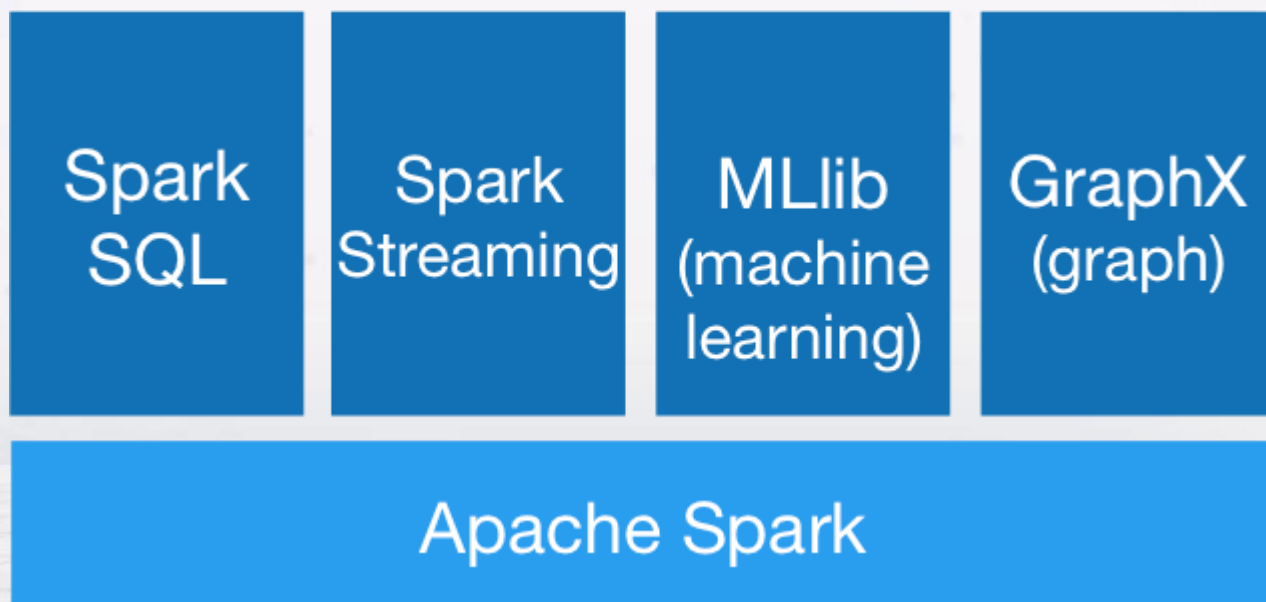
# Spark 다운로드 및 설치

---

# Spark의 개념과 주요 기능



- 범용적 목적의 분산 고성능 클러스터링 플랫폼 (General purpose high performance distributed platform) 스파크의 주요 기능
  - Map & Reduce (cf. Hadoop)
  - Streaming 데이터 핸들링 (cf. Apache Storm)
  - SQL 기반의 데이터 쿼리 (cf. Hadoop의 Hive)
  - 머신 러닝 라이브러리 (cf. Apache Mahout)
- 스칼라로 구현되었지만, 파이썬, 자바, R, 스칼라 등 다양한 언어를 지원하기 위한 SDK를 가지고 있음



<https://spark.apache.org/>

# Spark 다운로드



<http://spark.apache.org/downloads.html>

Download Libraries Documentation Examples Community Developers Apache Software Foundation

## Download Apache Spark™

1. Choose a Spark release: 2.3.1 (Jun 08 2018) ▼
2. Choose a package type: Pre-built for Apache Hadoop 2.7 and later ▼
3. Download Spark: spark-2.3.1-bin-hadoop2.7.tgz
4. Verify this release using the 2.3.1 signatures and checksums and project release KEYS.

Note: Starting version 2.0, Spark is built with Scala 2.10.4. To use the pre-built binaries, you must download the Spark source package and build with Scala 2.10 support.

## Link with Spark

Spark artifacts are hosted in Maven Central. You can add a Maven dependency with the following coordinates:

```
groupId: org.apache.spark
artifactId: spark-core_2.11
```

링크 클릭 후  
미러사이트의 링크  
주소를 복사해  
넣으세요.

## Latest News

Spark 2.3.1 released (Jun 08, 2018)

Spark+AI Summit (June 4-6th, 2018, San Francisco) agenda posted (Mar 01, 2018)

Spark 2.3.0 released (Feb 28, 2018)

Spark 2.2.1 released (Dec 01, 2017)

[Archive](#)



# Spark 다운로드

```
[nova@slave2 ~]$ wget http://apache.tt.co.kr/spark/spark-2.3.1/spark-2.3.1-bin-hadoop2.7.tgz
--2018-04-29 09:09:28-- http://apache.tt.co.kr/spark/spark-2.3.1/spark-2.3.1-bin-hadoop2.7.tgz
Resolving apache.tt.co.kr(apache.tt.co.kr)... 1.201.139.179
Connecting to apache.tt.co.kr(apache.tt.co.kr)|1.201.139.179|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 226128401 (216M) [application/x-gzip]
Saving to: 'spark-2.3.1-bin-hadoop2.7.tgz.1'

100%[=====>] 226,128,401 4.33MB/s in 52s

2018-04-29 09:10:21 (4.15 MB/s) - 'spark-2.3.1-bin-hadoop2.7.tgz.1' saved [225883783/225883783]

[nova@master Downloads]$ ls
spark-2.3.1-bin-hadoop2.7.tgz
```

# Spark 설치

```
[nova@master ~]$ cd 다운로드/
```

```
[nova@master 다운로드]$ ls
```

```
2007.csv.bz2 hadoop-3.0.3.tar.gz spark-2.3.1-bin-hadoop2.7.tgz
```

```
2008.csv.bz2 jdk-8u181-linux-x64.tar.gz
```

```
[nova@master 다운로드]$ cd
```

```
[nova@master ~]$ tar -xf ~/다운로드/spark-2.3.1-bin-hadoop2.7.tgz
```

```
[nova@master ~]$ mv spark-2.3.1-bin-hadoop2.7/ spark-2.3.1/
```

```
[nova@master ~]$ ls
```

```
hadoop-3.0.3 spark-2.3.1 다운로드 바탕화면 사진 음악
```

```
jdk1.8.0_181 공개 문서 비디오 서식
```

```
[nova@master ~]$
```



# Spark 환경 설정

```
[nova@master ~]$ cd spark-2.3.1/conf/
[nova@master conf]$ cp spark-env.sh.template spark-env.sh
[nova@master conf]$ vi spark-env.sh
```

```
#!/usr/bin/env bash
export SPARK_DIST_CLASSPATH=$(~/hadoop-3.0.3/bin/hadoop classpath)
export JAVA_HOME=~/jdk1.8.0_181
```

```
[nova@master conf]$ cp log4j.properties.template log4j.properties
[nova@master conf]$ vi log4j.properties
Set everything to be logged to the console log4j.root
log4j.rootCategory=WARN, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
```

INFO를 WARN으로 수정  
Spark 실행 시 너무 많은 로그가 나오기 때문에 로그 레벨을  
WARN으로 수정함  
로그 레벨은 FATAL, ERROR, WARN, INFO, DEBUG, TRACE 순서

# 하둡 실행 확인

Spark 실행 전 하둡 실행 확인. 만약 실행 중이 아니라면 start-all.sh로 하둡 클러스터와 yarn을 실행

```
[nova@master ~]$ jps 25857 Jps
```

```
25156 ResourceManager
```

```
24778 NameNode
```

```
[nova@master ~]$ ssh slave1 jps
```

```
3012 DataNode
```

```
3127 NodeManager
```

```
3257 Jps
```

```
[nova@master ~]$ ssh slave2 jps
```

```
2739 NodeManager
```

```
2869 Jps
```

```
2623 DataNode
```

```
[nova@master ~]$ ssh slave3 jps
```

```
2964 SecondaryNameNode
```

```
2853 DataNode
```

```
3046 NodeManager
```

```
3193 Jps
```

# Spark 실행

```
[nova@master conf]$ ~/spark-2.3.1/sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /home/nova/spark-
2.3.1/logs/spark-nova-org.apache.spark.deploy.master.Master-1-master.out lo
calhost: starting org.apache.spark.deploy.worker.Worker, logging to
/home/nova/spark-2.3.1/logs/spark-nova-org.apache.spark.deploy.worker.Worker-1-
master.out
[nova@master conf]$ jps
26803 Worker
26868 Jps
25156 ResourceManager
26644 Master
24778 NameNode
```

# pyspark 실행

## 2. Spark 다운로드 및 설치

```
[nova@master conf]$ ~/spark-2.3.1/bin/pyspark
Python 2.7.5 (default, Jul 13 2018, 13:06:57)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/nova/spark-2.3.1/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/nova/hadoop-3.0.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/08/02 23:00:07 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to

 / _/
 / _/
 / _/
 / _/
 / _/
 / _/
/_/_/

version 2.3.1
```

Spark에서 사용할 수 있는 언어는 Scala, Java, Python, R

```
Using Python version 2.7.5 (default, Jul 13 2018 13:06:57)
SparkSession available as 'spark'.
>>> quit()
[nova@localhost ~]$
```

# sparkR 실행

## 2. Spark 다운로드 및 설치

```
[nova@master conf]$ ~/spark-2.3.1/bin/sparkR
```

```
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)
```

R은 자유 소프트웨어이며, 어떠한 형태의 보증없이 배포됩니다.  
또한, 일정한 조건하에서 이것을 재배포 할 수 있습니다.  
배포와 관련된 상세한 내용은 'license()' 또는 'licence()'을 통하여 확인할 수 있습니다.

R은 많은 기여자들이 참여하는 공동프로젝트입니다.  
'contributors()'라고 입력하시면 이에 대한 더 많은 정보를 확인할 수 있습니다.  
그리고, R 또는 R 패키지들을 출판물에 인용하는 방법에 대해서는 'citation()'을 통해 확인하시길 부탁드립니다.

'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'help()'를 입력하시면 온라인 도움말을 이용하실 수 있습니다.  
또한, 'help.start()'의 입력을 통하여 HTML 브라우저에 의한 도움말을 이용하실 수 있습니다.  
R의 종료를 원하시면 'q()'을 입력해주세요.

```
Launching java with spark-submit command /home/nova/spark-2.3.1/bin/spark-submit "sparkr-shell"
/tmp/RtmpSDOhzn/backend/port69b7483c83ea
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/nova/spark-2.3.1/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/nova/hadoop-3.0.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/08/02 23:00:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

Welcome to



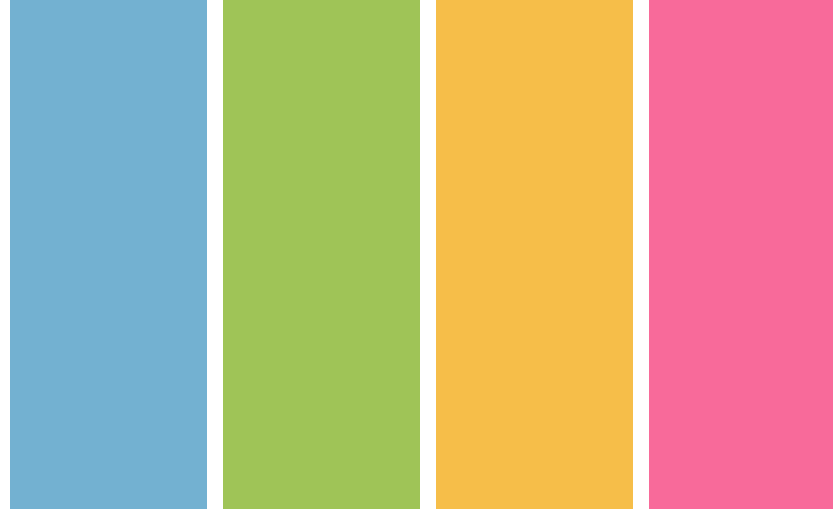
SparkSession available as 'spark'.

```
> q()
Save workspace image? [y/n/c]: y
[nova@master conf]$
```

R이 설치되어 있어야 합니다.  
설치되어 있지 않다면 root로 들어가서  
# yum install R -y

# spark-sql 실행

```
[nova@master conf]$ ~/spark-2.3.1/bin/spark-sql SL
F4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/nova/spark-2.3.1/jars/slf4j-log4j12
- 1.7.16.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/nova/hadoop-3.0.3/share/hadoop/common/lib/slf4j
- log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/08/02 23:02:05 WARN NativeCodeLoader: Unable to load native-hadoop library for you
r platform... using builtin-java classes where applicable
18/08/02 23:02:11 WARN ObjectStore: Version information not found in metastore. hive.
metastore.schema.verifcation is not enabled so recording the schema version 1.2.0 18/
08/02 23:02:11 WARN ObjectStore: Failed to get database default, returning
NoSuchObjectException
spark-sql> exit; [
nova@master conf]$
```



# Jupyter notebook

---



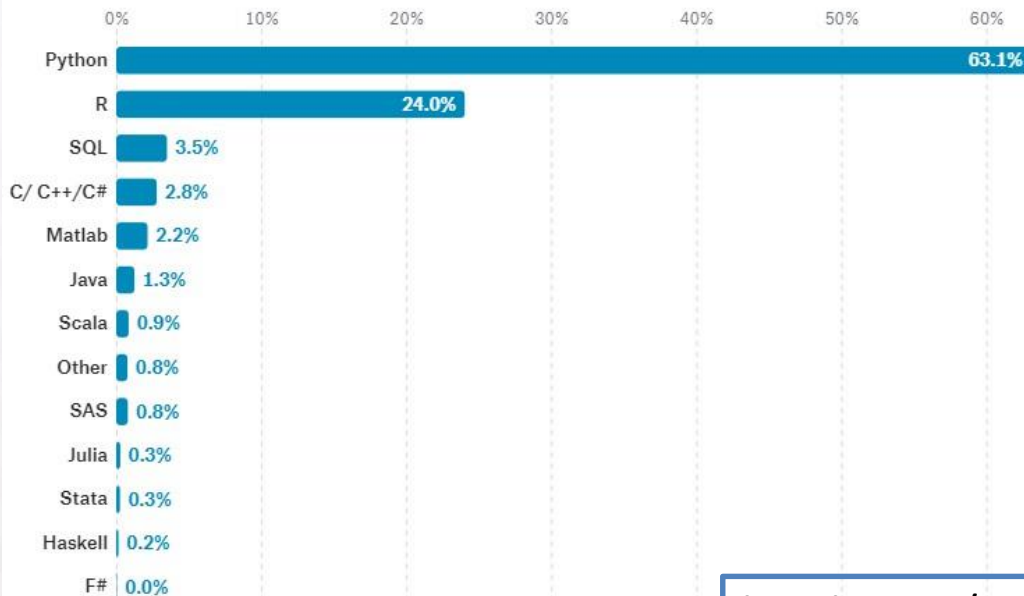
# 왜 pyspark를 사용해야 하나?

pyspark와 jupyter notebook

## What language would you recommend new data scientists learn first?

Everyone data scientist has an opinions on what language you should learn first. As it turns out, people who solely use Python or R feel like they made the right choice. But if you ask people that use **both** R and Python, they are twice as likely to recommend Python.

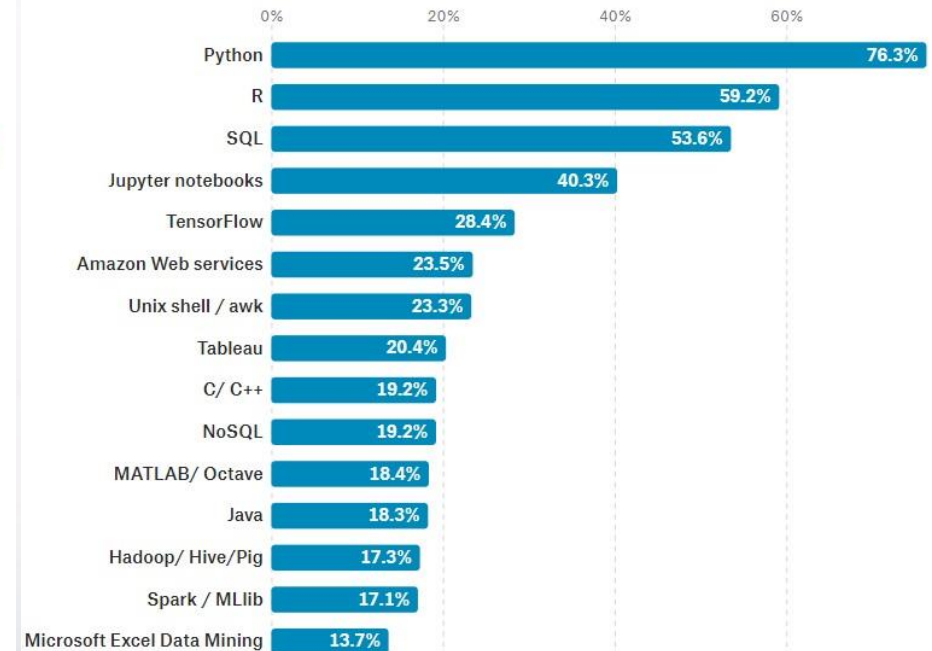
FILTER FROM USERS OF All Both Python R



## What tools are used at work?

Python was the most commonly used data analysis tool across employed data scientists overall, but more **Statisticians** are still loyal to R.

Company Size Industry Job Title



kaggle.com/surveys/2017

7,955 responses



# jupyter notebook 사용 환경변수 추가

```
$ vi .bashrc
```

```
export JAVA_HOME=~/.jdk1.8.0_181
```

```
export HADOOP_HOME=~/.hadoop-3.0.3
```

```
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

```
export PYSPARK_PYTHON=python3 ← pyspark에서 python3을 사용하려면 포함
```

```
export PYSPARK_DRIVER_PYTHON=jupyter
```

```
export PYSPARK_DRIVER_PYTHON_OPTS='notebook'
```

```
$ source .bashrc
```

# 하둡 실행 확인

```
[nova@master ~]$ jps
```

```
25857 Jps
```

```
25156 ResourceManager
```

```
24778 NameNode
```

```
[nova@master ~]$ ssh slave1 jps
```

```
3012 DataNode
```

```
3127 NodeManager
```

```
3257 Jps
```

```
[nova@master ~]$ ssh slave2 jps
```

```
2739 NodeManager
```

```
2869 Jps
```

```
2623 DataNode
```

```
[nova@master ~]$ ssh slave3 jps
```

```
2964 SecondaryNameNode
```

```
2853 DataNode
```

```
3046 NodeManager
```

```
3193 Jps
```

# Spark 실행

```
[nova@master conf]$ ~/spark-2.3.1/sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /home/nova/spark-
2.3.1/logs/spark-nova-org.apache.spark.deploy.master.Master-1-master.out lo
calhost: starting org.apache.spark.deploy.worker.Worker, logging to
/home/nova/spark-2.3.1/logs/spark-nova-org.apache.spark.deploy.worker.Worker-1-
master.out
[nova@master conf]$ jps
```

26803 Worker

26868 Jps

25156 ResourceManager

26644 Master

24778 NameNode

# pyspark 실행

```
nova@ubuntu:~$ ~/spark-2.3.1/bin/pyspark
```

```
[I 16:58:58.068 NotebookApp] Writing notebook server cookie secret to /run/user/1000/jupyter/notebook_cookie_secret
[I 16:58:58.515 NotebookApp] Serving notebooks from local directory: /home/nova
[I 16:58:58.515 NotebookApp] 0 active kernels
[I 16:58:58.515 NotebookApp] The Jupyter Notebook is running at:
[I 16:58:58.515 NotebookApp] http://localhost:8888/?token=495290cdc580c1fe9476b517decb4b8c2cb34ddb167d42a8
[I 16:58:58.515 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 16:58:58.519 NotebookApp]
```

Copy/paste this URL into your browser when you connect for the first time,  
to login with a token:

`http://localhost:8888/?token=495290cdc580c1fe9476b517decb4b8c2cb34ddb167d42a8`

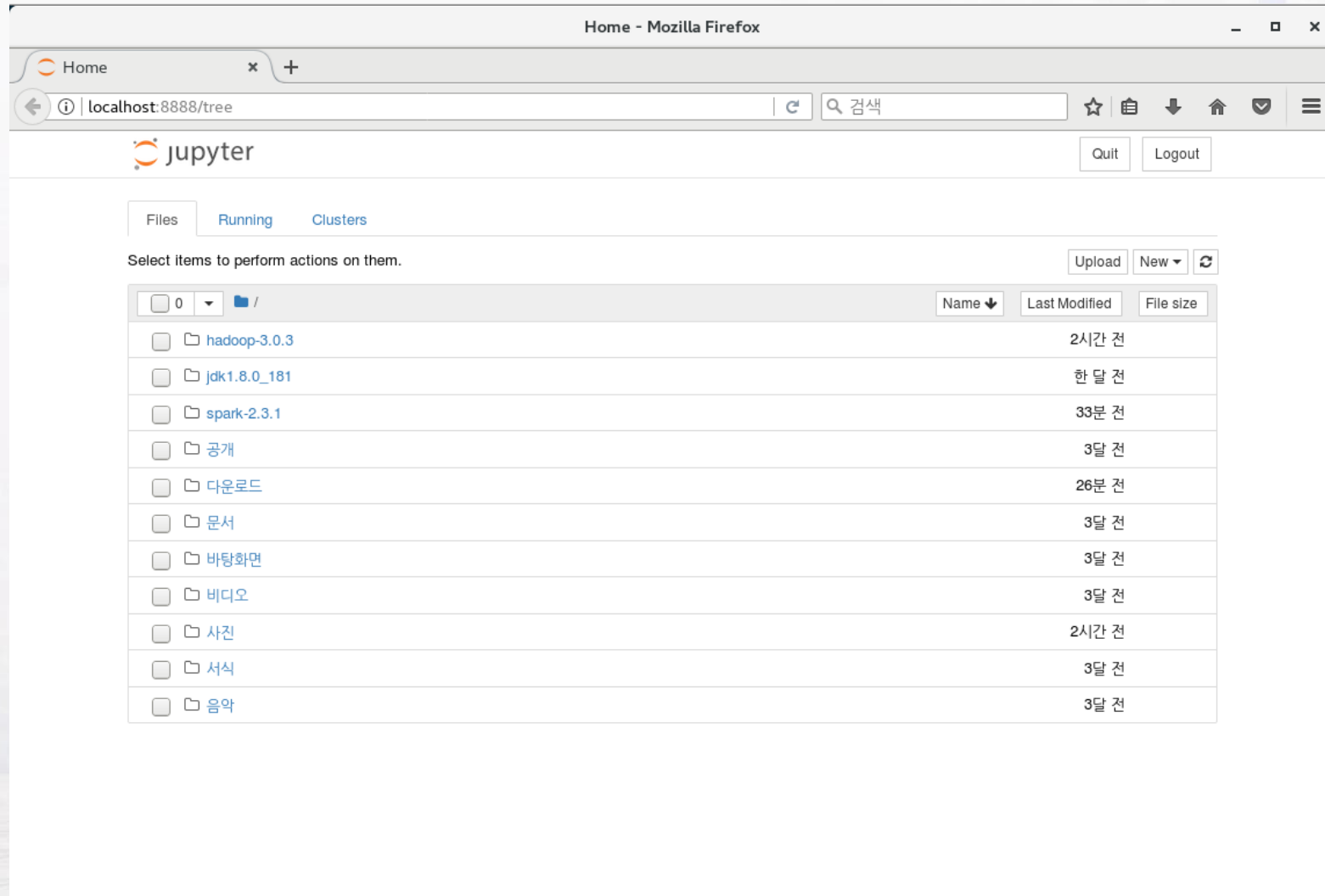
This tool has been deprecated, use 'gio open' instead.

See 'gio help open' for more info.

```
[I 16:59:01.002 NotebookApp] Accepting one-time-token-authenticated connection from 127.0.0.1
```

# jupyter notebook에서 실행된 pyspark

## 4. pyspark와 jupyter notebook



# 노트북 파일 만들기

## 4. pyspark와 jupyter notebook

The screenshot shows the Jupyter Notebook web interface in a browser. The address bar indicates the URL is `localhost:8888/tree`. The interface includes a top navigation bar with 'Home', 'Running', and 'Clusters' tabs. Below this, there are buttons for 'Quit' and 'Logout'. The main content area displays a file tree under the root directory '/'. A list of files and folders is shown, including 'hadoop-3.0.3', 'jdk1.8.0\_181', 'spark-2.3.1', '공개', '다운로드', '문서', '바탕화면', '비디오', '사진', '서식', and '음악'. A red dashed box highlights the 'New' button in the top right corner of the file list. A dropdown menu is open, showing options: 'Notebook', 'Python 2', 'Other:', 'Text File', 'Folder', and 'Terminal'. The 'Notebook' and 'Python 2' options are highlighted with a red dashed box.

Home x +

localhost:8888/tree | 검색

jupyter Quit Logout

Files Running Clusters

Select items to perform actions on them.

Upload New

Name

Notebook

Python 2

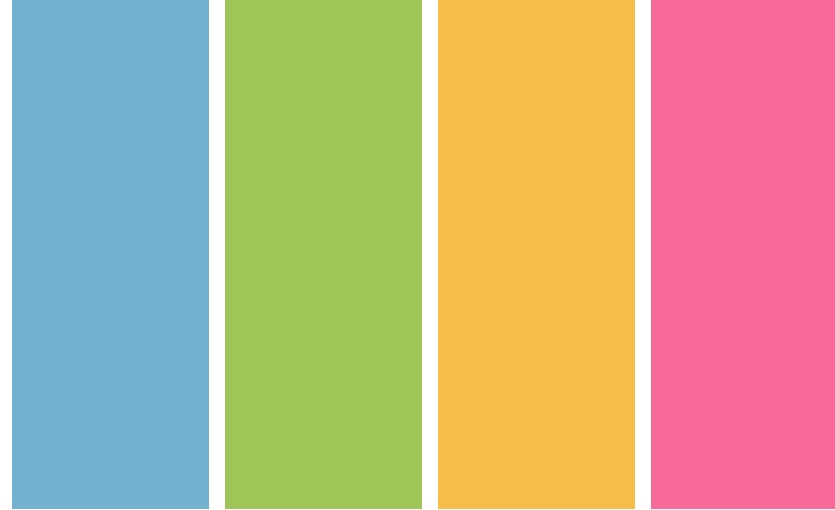
Other:

Text File

Folder

Terminal

| Name         | Modified |
|--------------|----------|
| hadoop-3.0.3 |          |
| jdk1.8.0_181 |          |
| spark-2.3.1  |          |
| 공개           |          |
| 다운로드         | 26분 전    |
| 문서           | 3달 전     |
| 바탕화면         | 3달 전     |
| 비디오          | 3달 전     |
| 사진           | 2시간 전    |
| 서식           | 3달 전     |
| 음악           | 3달 전     |



## 5. Spark를 이용한 데이터 분석

---

# HDFS에 파일 업로드

## 5. Spark를 이용한 데이터 분석

```
[nova@master ~]$ cd 다운로드/
[nova@master 다운로드]$ wget https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv
--2018-08-03 00:08:27-- https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.249
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.249|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 84199 (82K) [text/csv]
Saving to: 'winequality-red.csv'

100%[=====>] 84,199 147KB/s in 0.6s

2018-08-03 00:08:29 (147 KB/s) - 'winequality-red.csv' saved [84199/84199]

[nova@master 다운로드]$ hdfs dfs -mkdir -p /user/nova/wine-quality
[nova@master 다운로드]$ hdfs dfs -put winequality-red.csv /user/nova/wine-quality/
[nova@master 다운로드]$ hdfs dfs -ls /user/nova/wine-quality
Found 1 items
-rw-r--r-- 2 nova supergroup 84199 2018-08-03 00:08 /user/nova/wine-quality/winequality-red.csv
[nova@master 다운로드]$ hdfs dfs -ls wine-quality
Found 1 items
-rw-r--r-- 2 nova supergroup 84199 2018-08-03 00:08 wine-quality/winequality-red.csv
[nova@master 다운로드]$
```



# 와인 등급 분류하기

```
In [1]: import pandas as pd
import pydoop.hdfs as hd
```

pydoop은 하둡이 설치되어 있어야 함

```
In [2]: #redwine = np.loadtxt(fname='Data/winequality-red.csv', delimiter=',', skiprows=1)
with hd.open("/user/nova/wine-quality/winequality-red.csv") as f:
 redwine = pd.read_csv(f, sep=';')
```

```
In [3]: redwine.head()
```

Out[3]:

|   | fixed<br>acidity | volatile<br>acidity | citric<br>acid | residual<br>sugar | chlorides | free sulfur<br>dioxide | total sulfur<br>dioxide | density | pH   | sulphates | alcohol | quality |
|---|------------------|---------------------|----------------|-------------------|-----------|------------------------|-------------------------|---------|------|-----------|---------|---------|
| 0 | 7.4              | 0.70                | 0.00           | 1.9               | 0.076     | 11.0                   | 34.0                    | 0.9978  | 3.51 | 0.56      | 9.4     | 5       |
| 1 | 7.8              | 0.88                | 0.00           | 2.6               | 0.098     | 25.0                   | 67.0                    | 0.9968  | 3.20 | 0.68      | 9.8     | 5       |
| 2 | 7.8              | 0.76                | 0.04           | 2.3               | 0.092     | 15.0                   | 54.0                    | 0.9970  | 3.26 | 0.65      | 9.8     | 5       |
| 3 | 11.2             | 0.28                | 0.56           | 1.9               | 0.075     | 17.0                   | 60.0                    | 0.9980  | 3.16 | 0.58      | 9.8     | 6       |
| 4 | 7.4              | 0.70                | 0.00           | 1.9               | 0.076     | 11.0                   | 34.0                    | 0.9978  | 3.51 | 0.56      | 9.4     | 5       |

```
In [4]: redwine.shape
```

Out[4]: (1599, 12)

```
[root@master ~]# export JAVA_HOME=/home/nova/jdk1.8.0_181
[root@master ~]# export HADOOP_HOME=/home/nova/hadoop-3.0.3
[root@master ~]# pip install pydoop
[root@master ~]# pip install pandas
[root@master ~]# pip install sklearn
[root@master ~]# pip install scipy
```

# 와인 등급 분류하기

```
In [5]: import math
```

```
In [6]: math.floor(redwine.shape[0]*0.7)
```

```
Out[6]: 1119.0
```

```
In [7]: from pandas import Series, DataFrame
```

```
In [8]: redwine_dataframe = DataFrame(redwine,
 columns=["fixed acidity", "volatile acidity", "citric acid",
 "residual sugar", "chlorides", "free sulfur dioxide",
 "total sulfur dioxide", "density", "pH", "sulphates",
 "alcohol", "quality"])
```

```
In [9]: train = redwine_dataframe.sample(frac=0.7)
test = redwine_dataframe.loc[~redwine_dataframe.index.isin(train.index)]
```

```
In [10]: train.head()
```

```
Out[10]:
```

|     | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH   | sulphates | alcohol | quality |
|-----|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|------|-----------|---------|---------|
| 295 | 10.8          | 0.50             | 0.46        | 2.5            | 0.073     | 5.0                 | 27.0                 | 1.00010 | 3.05 | 0.64      | 9.5     | 5       |
| 322 | 7.8           | 0.62             | 0.05        | 2.3            | 0.079     | 6.0                 | 18.0                 | 0.99735 | 3.29 | 0.63      | 9.3     | 5       |

# 와인 등급 분류하기

```
In [10]: train.head()
```

```
Out[10]:
```

|      | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH   | sulphates | alcohol | quality |
|------|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|------|-----------|---------|---------|
| 295  | 10.8          | 0.50             | 0.46        | 2.5            | 0.073     | 5.0                 | 27.0                 | 1.00010 | 3.05 | 0.64      | 9.5     | 5       |
| 322  | 7.8           | 0.62             | 0.05        | 2.3            | 0.079     | 6.0                 | 18.0                 | 0.99735 | 3.29 | 0.63      | 9.3     | 5       |
| 778  | 8.3           | 0.43             | 0.30        | 3.4            | 0.079     | 7.0                 | 34.0                 | 0.99788 | 3.36 | 0.61      | 10.5    | 5       |
| 212  | 11.6          | 0.44             | 0.64        | 2.1            | 0.059     | 5.0                 | 15.0                 | 0.99800 | 3.21 | 0.67      | 10.2    | 6       |
| 1166 | 9.9           | 0.54             | 0.26        | 2.0            | 0.111     | 7.0                 | 60.0                 | 0.99709 | 2.94 | 0.98      | 10.2    | 5       |

```
In [11]: test.head()
```

```
Out[11]:
```

|    | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH   | sulphates | alcohol | quality |
|----|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|------|-----------|---------|---------|
| 2  | 7.8           | 0.76             | 0.04        | 2.3            | 0.092     | 15.0                | 54.0                 | 0.9970  | 3.26 | 0.65      | 9.8     | 5       |
| 10 | 6.7           | 0.58             | 0.08        | 1.8            | 0.097     | 15.0                | 65.0                 | 0.9959  | 3.28 | 0.54      | 9.2     | 5       |
| 18 | 7.4           | 0.59             | 0.08        | 4.4            | 0.086     | 6.0                 | 29.0                 | 0.9974  | 3.38 | 0.50      | 9.0     | 4       |
| 20 | 8.9           | 0.22             | 0.48        | 1.8            | 0.077     | 29.0                | 60.0                 | 0.9968  | 3.39 | 0.53      | 9.4     | 6       |
| 27 | 7.9           | 0.43             | 0.21        | 1.6            | 0.106     | 10.0                | 37.0                 | 0.9966  | 3.17 | 0.91      | 9.5     | 5       |

# 와인 등급 분류하기

```
In [12]: print (train.shape, test.shape)
```

```
((1119, 12), (480, 12))
```

```
In [13]: train_x = train.iloc[:, :-1]
train_y = train.iloc[:, -1]
test_x = test.iloc[:, :-1]
test_y = test.iloc[:, -1]
```

```
In [14]: from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(50,30))
mlp.fit(train_x, train_y)
print("Training score: %s" % mlp.score(train_x, train_y))
```

```
Training score: 0.571045576408
```

```
In [15]: pred = mlp.predict(test_x)
```

```
In [16]: confusion_matrix = pd.crosstab(test_y, pred,
 rownames=['True'],
 colnames=['Predicted'], margins=True)

type(confusion_matrix)
```

```
Out[16]: pandas.core.frame.DataFrame
```

# 와인 등급 분류하기

```
In [17]: print(confusion_matrix)
```

| Predicted | 5   | 6   | 7 | All |
|-----------|-----|-----|---|-----|
| True      |     |     |   |     |
| 3         | 3   | 0   | 0 | 3   |
| 4         | 13  | 1   | 0 | 14  |
| 5         | 154 | 39  | 0 | 193 |
| 6         | 101 | 96  | 1 | 198 |
| 7         | 15  | 48  | 4 | 67  |
| 8         | 0   | 5   | 0 | 5   |
| All       | 286 | 189 | 5 | 480 |

```
In [18]: confusion_matrix[2:]
```

Out[18]:

|      | Predicted | 5   | 6 | 7   | All |
|------|-----------|-----|---|-----|-----|
| True |           |     |   |     |     |
| 5    | 154       | 39  | 0 | 193 |     |
| 6    | 101       | 96  | 1 | 198 |     |
| 7    | 15        | 48  | 4 | 67  |     |
| 8    | 0         | 5   | 0 | 5   |     |
| All  | 286       | 189 | 5 | 480 |     |

# 와인 등급 분류하기

```
In [19]: print(confusion_matrix.iloc[2:5,0:3])
```

| Predicted | 5   | 6  | 7 |
|-----------|-----|----|---|
| True      |     |    |   |
| 5         | 154 | 39 | 0 |
| 6         | 101 | 96 | 1 |
| 7         | 15  | 48 | 4 |

```
In [20]: cm = confusion_matrix.as_matrix()
print(cm)
```

```
[[3 0 0 3]
 [13 1 0 14]
 [154 39 0 193]
 [101 96 1 198]
 [15 48 4 67]
 [0 5 0 5]
 [286 189 5 480]]
```

```
In [21]: cm_row = confusion_matrix.shape[0]
cm_col = confusion_matrix.shape[1]
accuracy = (cm[2][0] + cm[3][1] + cm[4][2])/float(cm[cm_row-1][cm_col-1])
print(accuracy)
```

```
0.529166666667
```

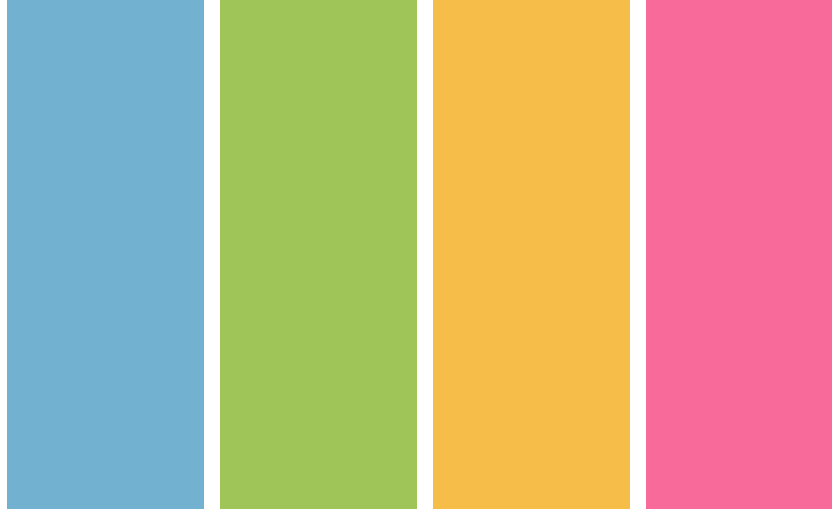
# References

- circa 1979 - **Stanford, MIT, CMU**, etc.: set/list operations in LISP, Prolog, etc., for parallel processing
  - <http://www-formal.stanford.edu/jmc/history/lisp/lisp.htm>
- circa 2004 - **Google**: MapReduce: Simplified Data Processing on Large Clusters  
Jeffrey Dean and Sanjay Ghemawat
  - <http://research.google.com/archive/mapreduce.html>
- circa 2006 - **Apache** Hadoop, originating from the Yahoo!'s Nutch Project Doug Cutting
  - <http://research.yahoo.com/files/cutting.pdf>
- circa 2008 - **Yahoo!**: web scale search indexing  
Hadoop Summit, HUG, etc.
  - <http://developer.yahoo.com/hadoop/>
- circa 2009 - **Amazon AWS**: Elastic MapReduce  
Hadoop modified for EC2/S3, plus support for Hive, Pig, Cascading, etc.
  - <http://aws.amazon.com/elasticmapreduce/>

# Spark References

- <http://spark.apache.org/docs/latest/programming-guide.html>
- <http://spark.apache.org/docs/latest/api/python/index.html>

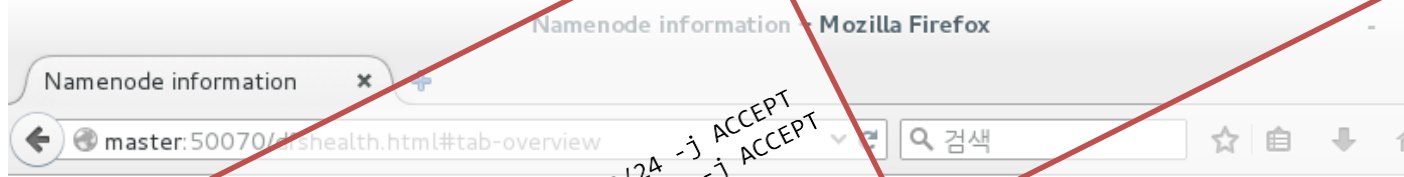




## 6. 하둡 클러스터 구성 관련 참고사항

---

# 방화벽 설정이 안 된 경우



CentOS 6  
 • 모든 노드 방화벽을 stop 시키거나  
 # service iptables stop  
 • 네트워크 CIDR 추가  
 # iptables -A INPUT -s 192.168.100.0/24 -d 192.168.100.0/24 -j ACCEPT  
 # iptables -A OUTPUT -s 192.168.100.0/24 -d 192.168.100.0/24 -j ACCEPT  
 # service iptables save  
 # service iptables restart

CentOS 7  
 • 모든 노드 방화벽을 stop 시키거나  
 # systemctl stop firewalld  
 # systemctl disable firewalld  
 • 네트워크 CIDR 추가(192.168.100. 대역에서 ssh 접근을 허용)  
 # firewall-cmd --permanent --zone=public --add-port=50010/tcp  
 # firewall-cmd --permanent --zone=public --add-source=192.168.100.0/24  
 # firewall-cmd --permanent --zone=public --add-rich-rule="rule family='ipv4' source address='192.168.100.0/24' port protocol='tcp' port='9000' accept"  
 # systemctl restart firewalld

| Configured Capacity:         |  |  |  |
|------------------------------|--|--|--|
| DFS Used:                    |  |  |  |
| Non-DFS Used:                |  |  |  |
| DFS Remaining:               |  |  |  |
| Block Pool Used:             |  |  |  |
| Block Pool Remaining:        |  |  |  |
| Nodes used:                  |  |  |  |
| Nodes in state:              |  |  |  |
| Nodes in state (Max/stdDev): |  |  |  |
| Decommissioning Nodes        |  |  |  |

# Primary NameNode(master)

```
{ $HADOOP_HOME/etc/hadoop/slaves or
workers}
$HADOOP_HOME/etc/hadoop/yarn-site.xml
yarn.nodemanager.aux-services mapreduce_shuffle

{ $HADOOP_HOME/etc/hadoop/mapred-site.xml}
mapreduce.framework.name yarn

{ $HADOOP_HOME/etc/hadoop/hdfs-site.xml}
dfs.replication dfs.namenode.name
enode.name.dir dfs.namenode.data.dir
menode.edits.dir dfs.namenode.edits.dir
dfs.namenode.secondary.http-address backup:50090 or backup:9868

{ $HADOOP_HOME/etc/hadoop/core-site.xml}
fs.defaultFS hdfs://master:9000

{ $HADOOP_HOME/etc/hadoop/hadoop-env.sh}
export JAVA_HOME=~/.jdk1.8.0_181

{ ~/.bashrc 또는 ~/.bash_profile}
export JAVA_HOME=~/.jdk1.8.0_181
export HADOOP_HOME=~/.hadoop-3.0.3
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

hadoop-3.0.3{ ~/.hadoop-3.0.3/ }

jdk1.8.0\_181{ ~/.jdk1.8.0\_181/ }

# DataNode(slave)

```
{ $HADOOP_HOME/etc/hadoop/hdfs-site.xml }
dfs.datanode.data.dir /dfs/data/

{ $HADOOP_HOME/etc/hadoop/yarn-site.xml }
yarn.resourcemanager.hostname master

{ $HADOOP_HOME/etc/hadoop/core-site.xml }
fs.defaultFS hdfs://master:9000

{ $HADOOP_HOME/etc/hadoop/hadoop-env.sh }
export JAVA_HOME=~/.jdk1.8.0_181

{ ~/.bashrc 또는 ~/.bash_profile }
export JAVA_HOME=~/.jdk1.8.0_181 exp
ort HADOOP_HOME=~/.hadoop-3.0.3
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

hadoop-3.0.3{~/.hadoop-3.0.3/}

jdk1.8.0\_181{~/.jdk1.8.0\_181/}

# Secondary NameNode(backup)

```
{ $HADOOP_HOME/etc/hadoop/hdfs-site.xml }
 dfs.namenode.checkpoint.dir /dfs/namesecondary/

{ $HADOOP_HOME/etc/hadoop/core-site.xml }
 fs.defaultFS hdfs://master:9000

{ $HADOOP_HOME/etc/hadoop/hadoop-env.sh }
 export JAVA_HOME=~/.jdk1.8.0_181

{ ~/.bashrc 또는 ~/.bash_profile }
 export JAVA_HOME=~/.jdk1.8.0_181
 export HADOOP_HOME=~/.hadoop-3.0.3
 export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

hadoop-3.0.3{ ~/.hadoop-3.0.3/ }

jdk1.8.0\_181{ ~/.jdk1.8.0\_181/ }

# 웹 인터페이스 서비스(Hadoop 2.x)

- NameNode
  - <http://namenode-ip:50070/>
- DataNode
  - <http://datanode-ip:5987/>
  - 방화벽 설정은 데이터 노드에 설정해야 함
- Cluster Metrics
  - <http://namenode-ip:8088/>
- Job History
  - <http://namenode-ip:19888/>
  - historyserver 가 실행되어 있어야 함
    - `$ mr-jobhistory-daemon.sh start historyserver`
- NodeManager Information
  - <http://nodemanager-ip:8042/>

# 웹 인터페이스 서비스(Hadoop 3.x)

- NameNode
  - `http://namenode-ip:9870/`
- DataNode
  - `http://datanode-ip:5987/`
  - 방화벽 설정은 데이터 노드에 설정해야 함
- Cluster Metrics
  - `http://namenode-ip:8088/`
- Job History
  - `http://namenode-ip:19888/`
  - historyserver 가 실행되어 있어야 함
    - `$ mapred --daemon start historyserver`
- NodeManager Information
  - `http://nodemanager-ip:8042/`