

# Hadoop 설치

on CentOS 7

# CentOS 기본 설치

## : 실습

- ▶ 다음과 같이 VirtualBox에 CentOS 가상 머신을 만들어 봅니다
  - ▶ 이름 : CentOS 7
  - ▶ 종류 : Linux
  - ▶ 버전 : CentOS 7 (64-bit)
  - ▶ 메모리 : 4096MB(4GB)
  - ▶ 하드디스크 : 128GB(VDI, 동적 할당)
- ▶ 네트워크 어댑터 2개를 설정합니다.
  - ▶ 어댑터 1: NAT
  - ▶ 어댑터 2: 호스트 전용 어댑터
    - ▶ 고급: 무작위 모드 - 모두 허용
- ▶ 설치 디스크를 넣고 가상머신을 시동하여 설치 작업을 진행합니다.

# CentOS 기본 설치

- ▶ 설치가 끝난 후 기본적인 업데이트 작업과 기본적인 패키지 설치 작업을 진행합니다.

```
$ yum update
$ yum upgrade
$ yum install rdate net-tools gcc make wget gcc-c++ bind-utils psmisc
```

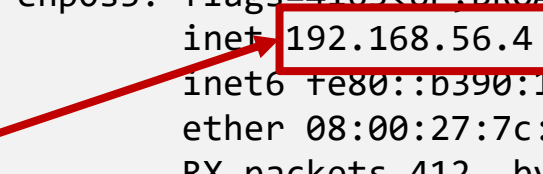
- ▶ 가상 머신의 네트워크를 확인해 봅니다.

```
$ ifconfig
```

- ▶ SSH 클라이언트로 접속해 봅니다.

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.56.4 netmask 255.255.255.0 broadcast 192.168.56.255
  inet6 fe80::b390:11a7:7860:1f84 prefixlen 64 scopeid 0x20<link>
  ether 08:00:27:7c:91:83 txqueuelen 1000 (Ethernet)
  RX packets 412 bytes 46078 (46.0 KB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 129 bytes 18329 (18.3 KB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

...
```



# CentOS 기본 설치

## : OpenJDK 설치

- ▶ JDK 설치(여기서는 OpenJDK 8을 설치합니다)

```
$ yum list java*jdk-devel # 설치할 수 있는 jdk 확인  
$ yum install java-1.8.0-openjdk-devel
```

- ▶ 손쉬운 접근을 위해 심볼릭 링크 설정

```
$ cd /usr/lib/jvm  
$ ln -s java-1.8.0-openjdk-1.8.0.272.b10-1.e17_9.x86_64/ jdk
```

- ▶ 환경 변수에 등록 (/etc/profile)

```
export JAVA_HOME=/usr/lib/jvm/jdk
```

- ▶ 쉘에서 환경변수 적용 및 확인(/etc/profile 에 했을 경우)

```
$ source /etc/profile  
$ echo $JAVA_HOME
```

# CentOS 기본 설치

## : protobuf 설치

### ▶ protobuf

- ▶ 구글이 작성
- ▶ C로 개발
- ▶ 이기종 언어들 간의 문자 체계가 깨지는 것을 방지하는 역할

```
$ yum -y install protobuf protobuf-devel  
$ # 설치 확인  
$ protoc --version
```

# Hadoop 설치

- ▶ 서버상에서 구동되는 대부분의 프로그램들은 **root** 계정으로 설치하는 것을 권장하지 않음
  - ▶ 하둡 운용을 위한 별도의 그룹과 계정을 생성하여 설치 및 운용

- ▶ Hadoop 실행 그룹 생성

```
$ groupadd hadoop
```

- ▶ Hadoop 계정을 생성 및 그룹에 등록

```
$ useradd -g hadoop -G wheel -m hadoop  
$ # hadoop 사용자를 wheel 그룹에 포함
```

- ▶ 이하, **hadoop** 계정으로 로그인하여 진행

# Hadoop 설치

- ▶ 독자 모드(Standalone mode)
  - ▶ 하둡의 기본 모드로 **HDFS**를 사용하지 않음. 다른 노드와 통신할 필요 없음
  - ▶ 테스트 및 디버깅 용도로 사용하는 모드. 일명 로컬 모드
- ▶ 가상 분산 모드(Pseudo-Distributed mode, Single Node Cluster mode)
  - ▶ 단일 노드에서 클러스터를 구성
  - ▶ 한 대의 컴퓨터에 모든 노드를 설치하여 노드(혹은 데몬)간 통신을 통해 **HDFS**를 사용
  - ▶ 실제 운영에 앞서 실제 **HDFS** 내에서의 구동을 확인하기 위한 모드
- ▶ 멀티 분산 모드(Multi-Distributed mode, Full-Distributed mode)
  - ▶ 두 대 이상의 노드를 클러스터로 묶어 구성
  - ▶ 여러 컴퓨터에 각각의 노드들을 설치하여 노드(혹은 데몬)간 통신을 통해 **HDFS**를 사용
  - ▶ 실제 운영에서 사용하는 모드
- ▶ 본 실습에서는 가상 분산 모드로 **HDFS**를 구성해 볼 예정

# Hadoop 설치

## : Download and Install

- ▶ <http://hadoop.apache.org>에서 링크 복사 후 다운로드

```
$ wget http://mirror.apache-kr.org/hadoop/common/hadoop-2.9.2/hadoop-2.9.2.tar.gz
```

- ▶ 압축 해제

```
$ tar zxvf hadoop-2.9.2.tar.gz
```

- ▶ 실제 실행할 위치로 이동

```
$ sudo mv hadoop-2.9.2 /usr/local/hadoop
```

```
# Hadoop-2.9.2 디렉토리를 /usr/local/Hadoop 디렉토리로 이동
```



# Hadoop 설치

## : 기본 설정

- ▶ /usr/local/hadoop/etc/hadoop으로 이동하여 hadoop-env.sh 파일을 수정

```
export JAVA_HOME=/usr/lib/jvm/jdk
```

- ▶ .bashrc 수정(HADOOP\_HOME 정보 설정)

```
export HADOOP_HOME=/usr/local/hadoop ← 하둡이 설치된 실제 경로
PATH=$PATH:$HADOOP_HOME/bin
PATH=$PATH:$HADOOP_HOME/sbin
export PATH
```

- ▶ 설정 적용 및 하둡 실행 확인

```
$ source ~/.bashrc
$ echo $HADOOP_HOME
$ hadoop version
```

# Hadoop 설치

## : 네임 노드 설정

- ▶ /usr/local/hadoop/etc/hadoop으로 이동하여 core-site.xml 파일을 수정

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/hadoop/temp</value>
  </property>
</configuration>
```

← 가급적 설정해 주도록 합니다.

# Hadoop 설치

## : 하둡 파일 시스템 설정

- ▶ /usr/local/hadoop/etc/hadoop으로 이동하여 hdfs-site.xml 파일을 수정

```
<configuration>
  <property>
    <name>dfs.replication</name> ← 복제본의 개수 설정
    <value>1</value>               : 현재 실행 모드는 Pseudo Distributed Mode이므로
                                   1이어야 합니다.
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/usr/local/hadoop/data/dfs/datanode</value> ← 데이터 노드의 저장 위치
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/usr/local/hadoop/data/dfs/namenode</value> ← 네임 노드의 저장 위치
  </property>
</configuration>
```

# Hadoop 설치

## : MapReduce 설정

- ▶ /usr/local/hadoop/etc/hadoop으로 이동하여 mapred-site.xml 파일을 생성(템플릿 복사)

```
$ cp mapred-site.xml.template mapred-site.xml
```

- ▶ mapred-site.xml 파일을 수정

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

# Hadoop 설치

## : YARN 설정

- ▶ /usr/local/hadoop/etc/hadoop으로 이동하여 yarn-site.xml 파일을 수정

```
<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>
</configuration>
```

- ▶ Resource Manager와 Node Manager에 대한 설정을 잡는 과정

# Hadoop 설치

## : SSH 설정

- ▶ 현재 설치하는 방법은 가상 분산모드이므로 실제 컴퓨터가 1대일지라도 네임 노드와 데이터 노드는 네트워크를 이용하여 서로 통신을 주고 받는다
  - ▶ 서버에 로그인할 때 계정과 암호를 입력하는 절차를 생략하기 위해 상호 신뢰를 위한 키를 교환하여야 한다

```
$ cd ~/.ssh  
$ ssh-keygen -t rsa  
# $ cat id_rsa.pub >> authorized_keys  
$ ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub hadoop@192.168.56.100
```

- ▶ 로그인 없이 서버 접속이 가능한지 테스트

```
$ ssh localhost
```

# Hadoop 설치

## : HDFS 포맷 및 실행

- ▶ HDFS를 사용하기 위해 포맷 작업을 수행

```
$ hadoop namenode -format
```

- ▶ 하둡 데몬 실행

```
$ start-dfs.sh  
# 노드 연결을 위해 yes 입력(처음 실행시에만 요구함)  
$ start-yarn.sh
```

- ▶ 데몬 실행 확인

```
$ jps  
11991 DataNode  
12809 Jps  
12618 ResourceManager  
11836 NameNode  
12780 NodeManager  
12174 SecondaryNameNode
```

# Hadoop 설치

## : Web Interface의 활용

- ▶ 외부에서 Web Interface에 접근하기 위한 방화벽 설정

```
$ sudo firewall-cmd --permanent --zone=public --add-port=50070/tcp
$ sudo firewall-cmd --permanent --zone=public --add-port=50075/tcp
$ sudo firewall-cmd --permanent --zone=public --add-port=8042/tcp
$ sudo firewall-cmd --permanent --zone=public --add-port=8088/tcp

$ sudo systemctl restart firewallld
```

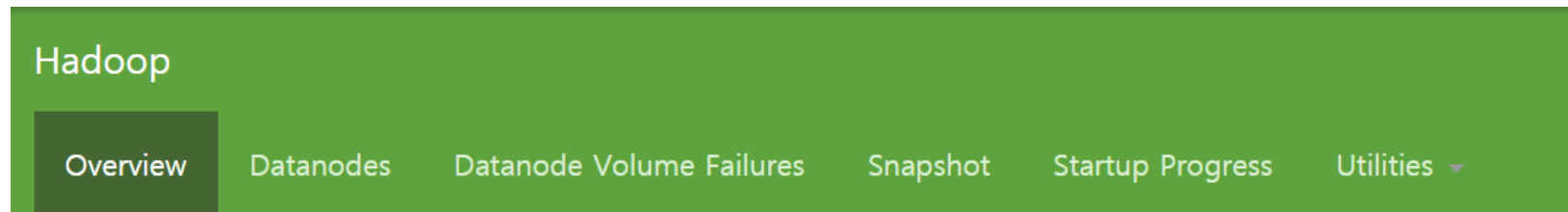


# Hadoop 설치

## : Web Interface의 활용

### ▶ Hadoop을 활용하기 위한 Web Interface

```
http://localhost:50070 # HDFS 네임노드 확인  
http://localhost:8042 # YARN의 노드 매니저 웹 인터페이스  
http://localhost:8088 # YARN 리소스 매니저의 웹 인터페이스(구 Job Tracker)
```



## Overview 'localhost:9000' (active)

Started:	Tue Nov 05 17:15:59 +0900 2019
Version:	2.8.5, r0b8464d75227fcee2c6e7f2410377b3d53d3d5f8
Compiled:	Mon Sep 10 12:32:00 +0900 2018 by jdu from branch-2.8.5

# HDFS Commands

# HDFS Commands

- ▶ 하둡은 사용자가 **HDFS**를 쉽게 제어할 수 있도록 쉘 명령어를 제공
  - ▶ 대부분의 명령어들은 유닉스/리눅스의 파일 관리 명령어들과 사용법이 유사

```
$ hdfs dfs -cmd [args]
```

- ▶ 사용법을 확인하려면

```
$ hdfs dfs --help
```

- ▶ 파일 목록 보기

```
$ hdfs dfs -ls [디렉터리|파일...]  
$ hdfs dfs -ls -R [디렉터리] # Recursive
```

# HDFS Commands

## ▶ 디렉터리 관리

### ▶ 생성

```
$ hdfs dfs -mkdir [디렉터리]
```

### ▶ 삭제

```
$ hdfs dfs -rm -R [디렉터리]
```

## ▶ 파일 용량 확인

```
$ hdfs dfs -du [디렉터리|파일...]
```

```
$ hdfs dfs -dus [디렉터리|파일...] # 전체 합계 용량만 출력
```

# HDFS Commands

## ▶ 파일 관리

### ▶ 복사 : local -> HDFS

```
$ hdfs dfs -copyFromLocal [로컬 디렉터리|파일] [HDFS 디렉터리|파일]  
$ hdfs dfs -put [로컬 디렉터리|파일] [HDFS 디렉터리|파일]
```

### ▶ 복사 : HDFS -> local

```
$ hdfs dfs -copyToLocal [소스 디렉터리|파일] [로컬 디렉터리|파일]  
$ hdfs dfs -get [소스 디렉터리|파일] [로컬 디렉터리|파일]
```

### ▶ 복사 : HDFS 내부

```
$ hdfs dfs -cp [소스 디렉터리|파일] [목적지 디렉터리|파일]
```

# HDFS Commands

## ▶ 파일 관리

### ▶ 이동

```
$ hdfs dfs -mv [소스 디렉터리|파일] [목적지 디렉터리|파일]
```

### ▶ 이동 : local -> HDFS

```
$ hdfs dfs -moveFromLocal [로컬 디렉터리|파일] [HDFS 디렉터리|파일]
```

### ▶ 이동 : HDFS -> local

```
$ hdfs dfs -moveToLocal [HDFS 디렉터리|파일] [로컬 디렉터리|파일]
```

### ▶ 삭제

```
$ hdfs dfs -rm [파일]
```

```
$ hdfs dfs -rm -R [디렉터리]
```

# HDFS Commands

## ▶ 파일 관리

- ▶ 카운트 조회: 지정 경로에 대한 전체 디렉터리 개수, 전체 파일 개수, 전체 파일 크기

```
$ hdfs dfs -count [디렉터리|파일]
```

## ▶ 파일 내용 보기

```
$ hdfs dfs -cat [파일]
```

```
$ hdfs dfs -text [파일] # 압축된 파일도 텍스트 형태로 출력
```

## ▶ 파일의 마지막 내용 보기

```
$ hdfs dfs -tail [파일]
```

# HDFS Commands

## ▶ 파일 관리

### ▶ 권한 변경

```
$ hdfs dfs -chmod <-R> [권한 모드...] [디렉터리|파일]
```

### ▶ 소유자 변경

```
$ hdfs dfs -chown <-R> [변경사용자명:변경그룹명] [디렉터리|파일]
```

### ▶ 소유권 그룹의 변경

```
$ hdfs dfs -chgrp <-R> [변경그룹명] [파일]
```

### ▶ 0바이트 파일 생성

```
$ hdfs dfs -touchz [파일]
```



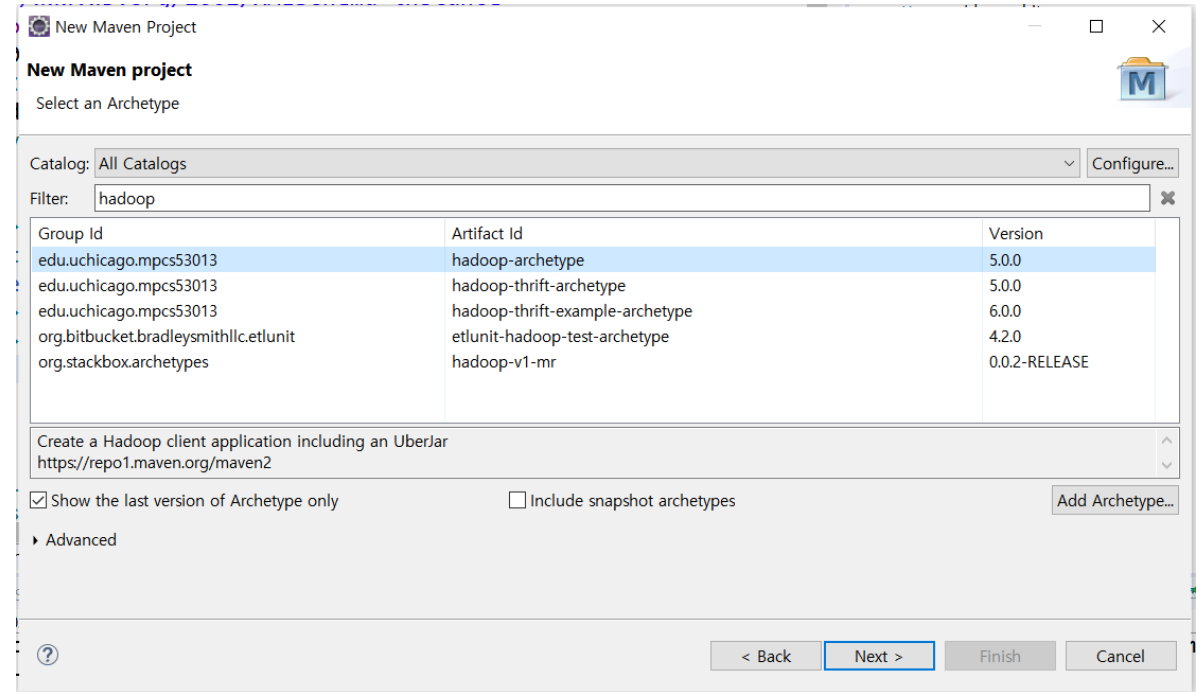
# HDFS IO Programming

# HDFS Java IO Programming

- ▶ Maven Project 생성
  - ▶ Archetype : hadoop-archetype
- ▶ Project Info
  - ▶ Group Id: com.bit
  - ▶ Artifact Id: myhadoop
  - ▶ version: 0.0.1
  - ▶ class package: myhadoop

- ▶ Dependency 정보 변경

```
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-client</artifactId>
  <!-- Client 버전은 서버의 Hadoop 버전과 일치 -->
  <version>2.9.2</version>
</dependency>
```



# HDFS Java IO Programming

## ▶ Maven Project 생성

### ▶ compiler plugin 정보 변경

```
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.3</version>
  <!-- Java Version Config -->
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
  </configuration>
</plugin>
```

### ▶ Maven > Update Project ...

# HDFS Java IO Programming

- ▶ [실습] 경로와 메시지를 전달받아 저장하는 HDFS IO Programming을 개발해 봅시다.
- ▶ org.apache.hadoop.fs 패키지로 로컬이나 HDFS 파일을 제어할 수 있음
  - ▶ Step 1. Configuration 객체 생성
    - ▶ 이 객체를 이용하여 하둡의 각종 설정 파일에 설정된 값을 조회하거나 변경할 수 있음
  - ▶ Step 2. FileSystem 객체를 획득

```
Configuration conf = new Configuration();  
FileSystem hdfs = FileSystem.get(conf);
```

- ▶ Step 3. 작업 대상 경로(디렉터리 or 파일) 객체 생성

```
Path path = new Path("/path/to/source");
```

# HDFS Java IO Programming

- ▶ Step 4. Stream 객체 생성, 사용 후 닫기

```
FSDataOutputStream os = hdfs.create(path);  
os.writeUTF("Message");  
os.close();
```

- ▶ Step 5. Build and make .jar

- ▶ Run As > Maven Build > Goal : clean install

- ▶ Hadoop 서버에서 실행해 봅니다.

- ▶ [추가 과제] 커맨드 라인에서 파일 경로를 입력 받아 내용을 출력하는 프로그램을 작성해 보시다.

# HDFS Python IO Programming

- ▶ Python에서 HDFS를 사용하는 다수의 모듈이 존재

- ▶ hdfs 모듈을 이용하여 HDFS에 접근해 봅시다.

- ▶ 모듈의 설치

```
pip install hdfs # PIP의 경우  
conda install -c conda-forge python-hdfs # conda의 경우
```

- ▶ 모듈 임포트 및 초기화

```
from hdfs import InsecureClient  
  
client = InsecureClient("http://192.168.56.100:50070", user="hadoop")  
client.status
```

# HDFS Python IO Programming

## ▶ hdfs 모듈

### ▶ 파일 목록 불러오기

```
files = client.list(path) # hdfs 내의 지정한 디렉터리 내 파일 이름의 목록을 반환

if len(files) == 0:
    print("File Not Founds")
else:
    for file in files:
        print("File", file)
```

### ▶ 디렉터리 생성

```
client.makedirs("/path/to/target") # 디렉터리를 재귀적으로 생성
```

# HDFS Python IO Programming

- ▶ hdfs 모듈

- ▶ 파일에 내용 기록

```
with client.write(path, encoding="UTF-8") as writer:  
    writer.write(text)
```

- ▶ 파일 내용 읽어오기

```
with client.read("source/filename", encoding="utf-8") as reader:  
    data = reader.read()  
  
    print("Hadoop Read:", data)
```