묻지마 Data Science Tools 설치하기

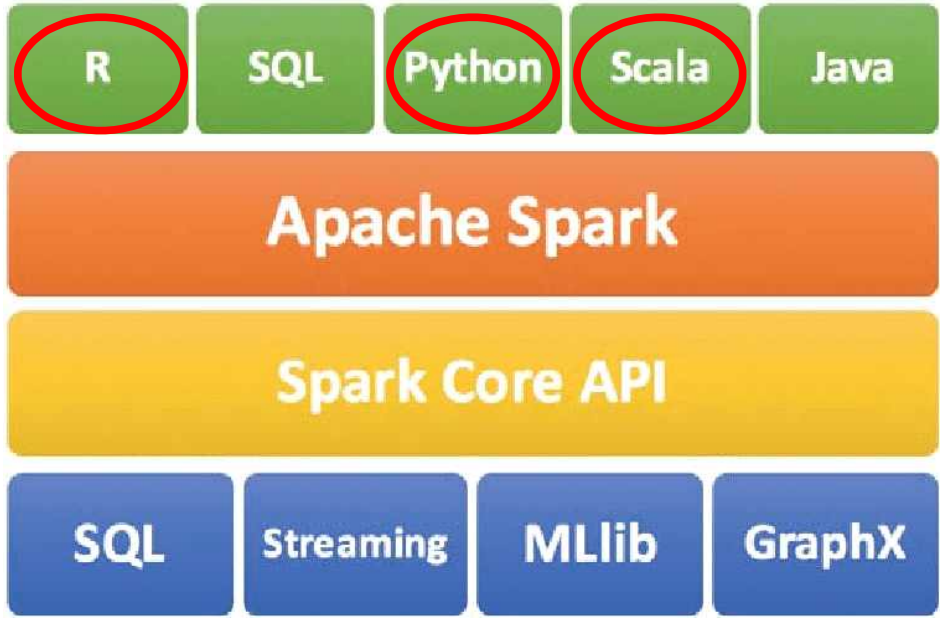
숙명여대 여인권

inkwon.yeo@gmail.com

2019/03/28

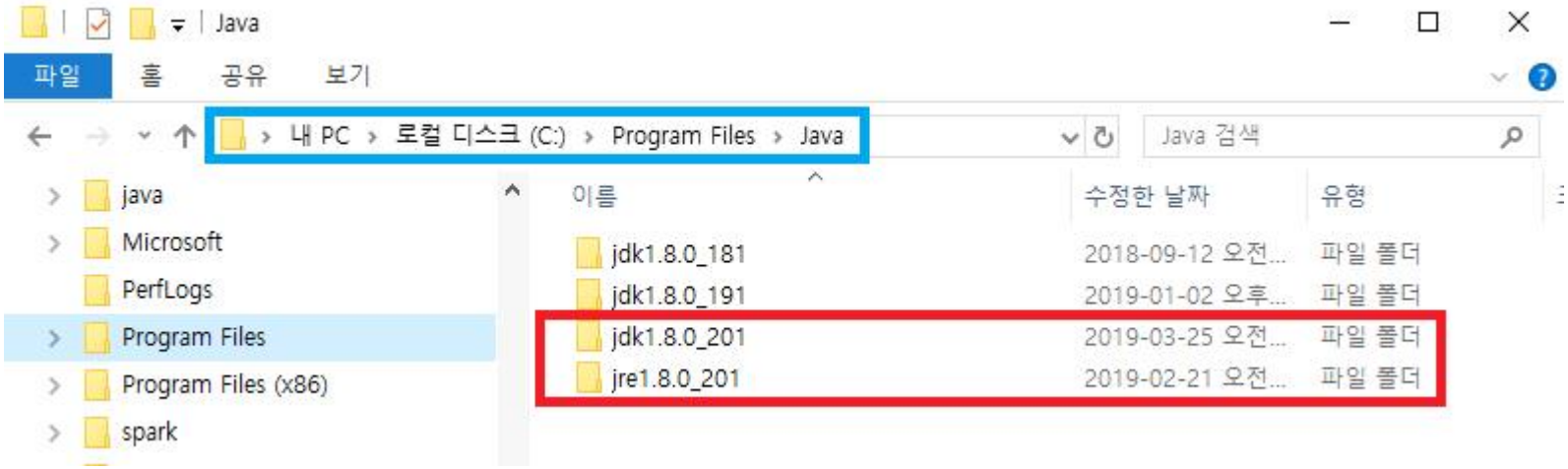
□ 설치프로그램

- JAVA, R, Python, Scala
- Hadoop
- Spark
- H2O, Sparkling Water

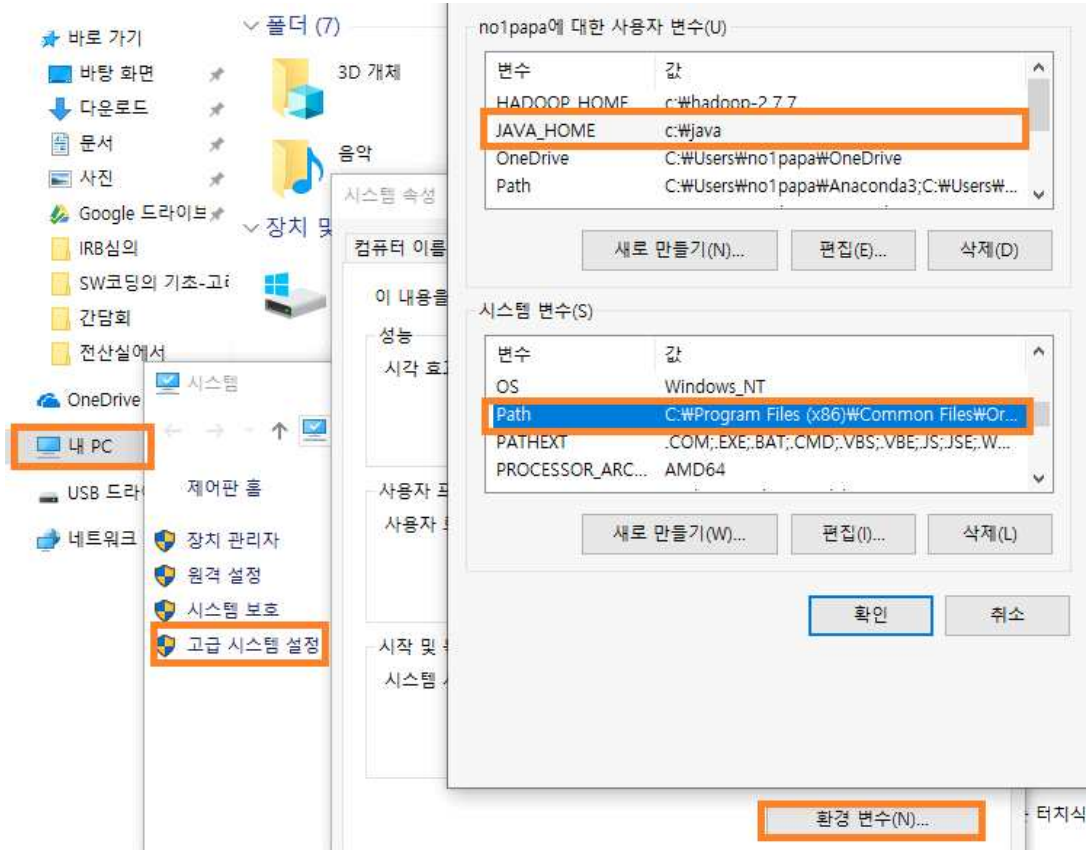


□ Java

- Windows용 JDK, JRE 다운로드 후 설치
  - Java SE Development Kit 8 - Downloads – Oracle
  - Java Runtime Environment - Oracle
  - 설치확인: C:₩Program Files₩Java₩jdkX.X.X_X
- C:₩java 폴더 생성 후 C:₩Program Files₩Java₩jdkX.X.X_X 내용 복사



- 고급시스템설정 ⇒ 환경변수
  - 사용자 변수(U) ▶ 새로 만들기 ⇒ 변수: JAVA_HOME, 값: C:₩java
  - 시스템 변수(S) ▶ path 편집 ⇒ C:₩java₩bin; C:₩java₩jre₩bin 추가



- java 버전 확인

```
> java -version
> javac -version
```

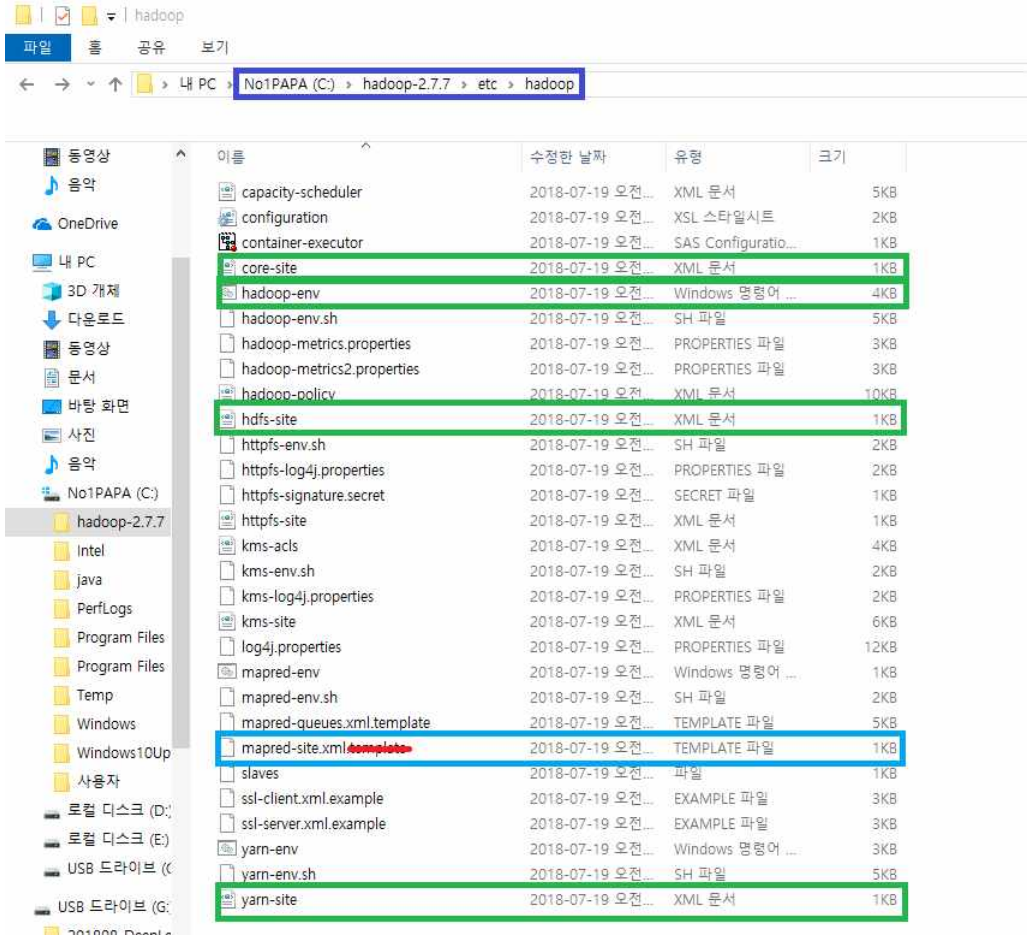
□ Hadoop

○ 하둡 설치

- 하둡 다운로드: https://hadoop.apache.org/releases.html
- Hadoop 2.X.X.tar.gz 압축을 풀고 c:₩hadoop-2.X.X로 이동(복사)
- 고급시스템설정 ⇒ 환경변수
  - 사용자 변수(U) > 새로 만들기
    ⇒ 변수: HADOOP_HOME, 값: C:₩hadoop-2.X.X
  - 시스템 변수(S) > path 편집
    ⇒ C:₩hadoop-2.X.X₩bin 추가

○ 하둡 설정변경

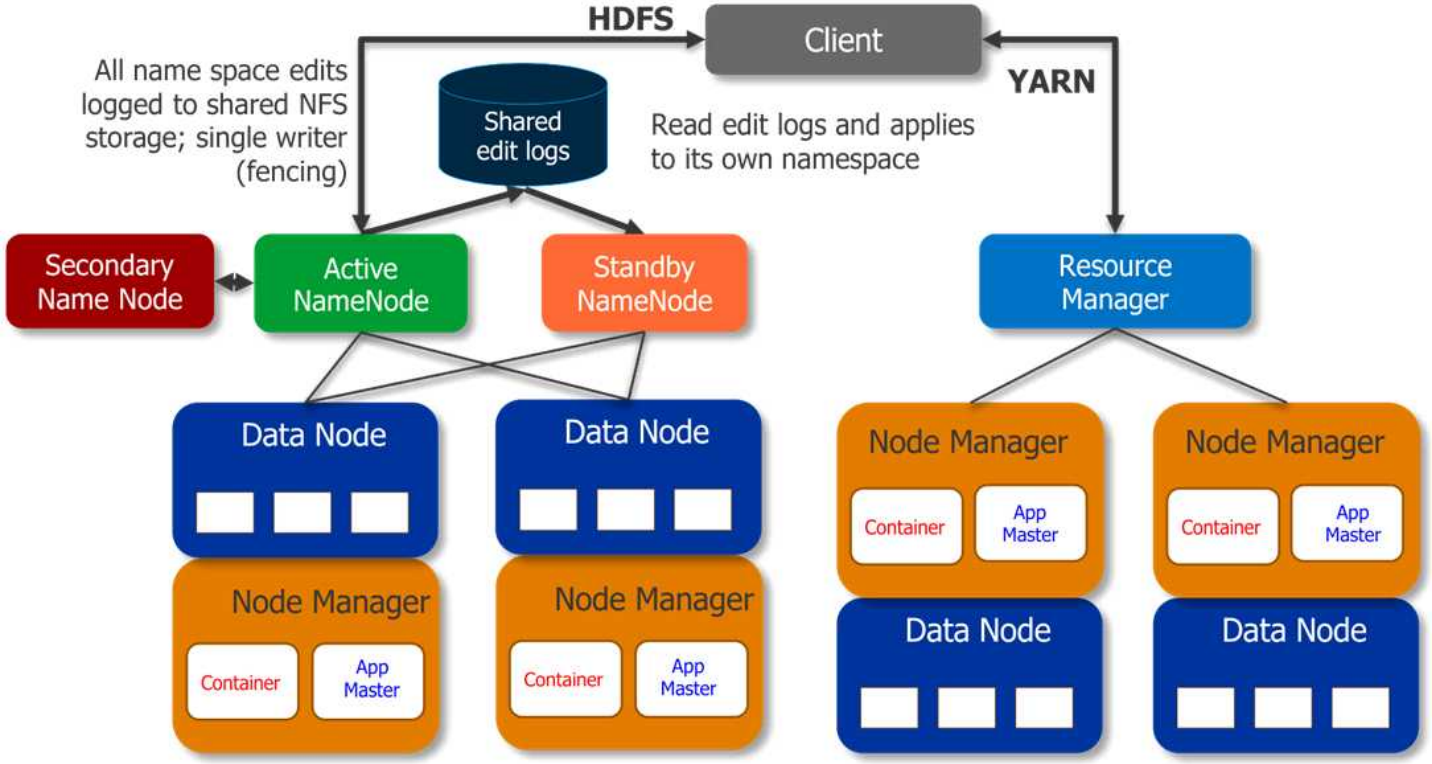
- C:₩hadoop-2.X.X₩etc₩hadoop로 이동



- hadoop-env.cmd 수정

set JAVA_HOME=%JAVA_HOME% 을 **set JAVA_HOME=c:₩java** 로
set HADOOP_PREFIX=c:₩hadoop-2.X.X
set HADOOP_CONF_DIR=%HADOOP_PREFIX%₩etc₩hadoop
set YARN_CONF_DIR=%HADOOP_CONF_DIR%
set PATH=%PATH%;%HADOOP_PREFIX%₩bin

○ 하둡 2.x 시스템



출처: https://bigdatainnovation.blogspot.com/2014/09/hadoop-10-vs-hadoop-20.html

◦ HDFS: Storage framework, MapReduce: Computation framework

- core-site.xml 수정: HDFS와 맵리듀스에서 공통적으로 사용되는 IO 설정 같은 하둡 코어 환경정보 설정
  - 네임노드 설정

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

- d:₩hadoop₩data 폴더 생성
  - namenode와 datanode를 생성할 위치

- hdfs-site.xml 수정: 파일복제 옵션

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/d:/hadoop/data/namenode</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/d:/hadoop/data/datanode</value>
    </property>
</configuration>
```

- mapred-site.xml 수정 (templete 확장자 제거)
  - 맵리듀스 설정 관리

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

  - 추가옵션
    - mapreduce.map.memory.mb : map작업시 활용하는 메모리
    - mapreduce.reduce.memory.mb : reduce작업시 활용하는 메모리

- yarn-site.xml 수정
  - yarn: 맵리듀스기반 하둡의 문제(리소스 할당문제나 Spark 등의 새로운 플랫폼 출현)를 해결하기 위한 하둡의 서브 프로젝트

```
<configuration>
<!-- Site specific YARN configuration properties -->
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.auxservices.mapreduce_shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
```

```
    <property>
        <name>yarn.application.classpath</name>
        <value>
            %HADOOP_HOME%₩etc₩hadoop,
            %HADOOP_HOME%₩share₩hadoop₩common₩*,
            %HADOOP_HOME%₩share₩hadoop₩common₩lib₩*,
            %HADOOP_HOME%₩share₩hadoop₩mapreduce₩*,
            %HADOOP_HOME%₩share₩hadoop₩mapreduce₩lib₩*,
            %HADOOP_HOME%₩share₩hadoop₩hdfs₩*,
            %HADOOP_HOME%₩share₩hadoop₩hdfs₩lib₩*,
            %HADOOP_HOME%₩share₩hadoop₩yarn₩*,
            %HADOOP_HOME%₩share₩hadoop₩yarn₩lib₩*
        </value>
    </property>
</configuration>
```

○ 하둡포멧

- https://github.com/Statfunny/Statistical-Computing/hadooponwindows.zip 다운로드
- hadooponwindows-master 압축풀고 bin 폴더에 있는 내용을 C:₩hadoop-2.X.X의 bin폴더에 덮어쓰기
- 네임노드 포맷

```
> cd c:₩hadoop-2.X.X₩bin
> hdfs namenode -format
```

  - d:₩hadoop₩data 폴더에 namenode 생성 확인

- **HDFS 실행**
  - Namenode와 Datanode 시작

```
> cd %HADOOP_HOME%₩sbin
> start-dfs.cmd  # 하둡 클러스터 시작
> jps            # 하둡 데몬(프로세스) 확인
```

  - d:₩hadoop₩data 폴더에 datanode 생성
  - Crome 실행후 localhost:50070 접속

- 맵리듀스 실행
  - ResourceManger와 NodeManger 시작

```
> start-yarn.cmd
> jps
```

  - Crome 실행후 localhost:8088접속, localhost:8042접속

- 프로그램 실행 & 종료(사용하지 않음)

```
> start-all.cmd
> stop-all.cmd
```

◉ 예제】 Data upload

```
> hdfs dfs –mkdir -p input_dir        # hdfs dfs 대신 hadoop fs 사용 가능
> hdfs dfs -put data.txt input_dir
  # localhost: 50070 > utilities > Browse the file system
  # hdfs dfs -copyFromLocal data.txt input_dir
> hdfs dfs -ls input_dir
> hdfs dfs -cat input_dir/data.txt     # /user/xxx/input_dir/data.txt 사용가능
> hdfs dfs -rm -r input_dir
```

○ 기타 주요 하둡 명령어

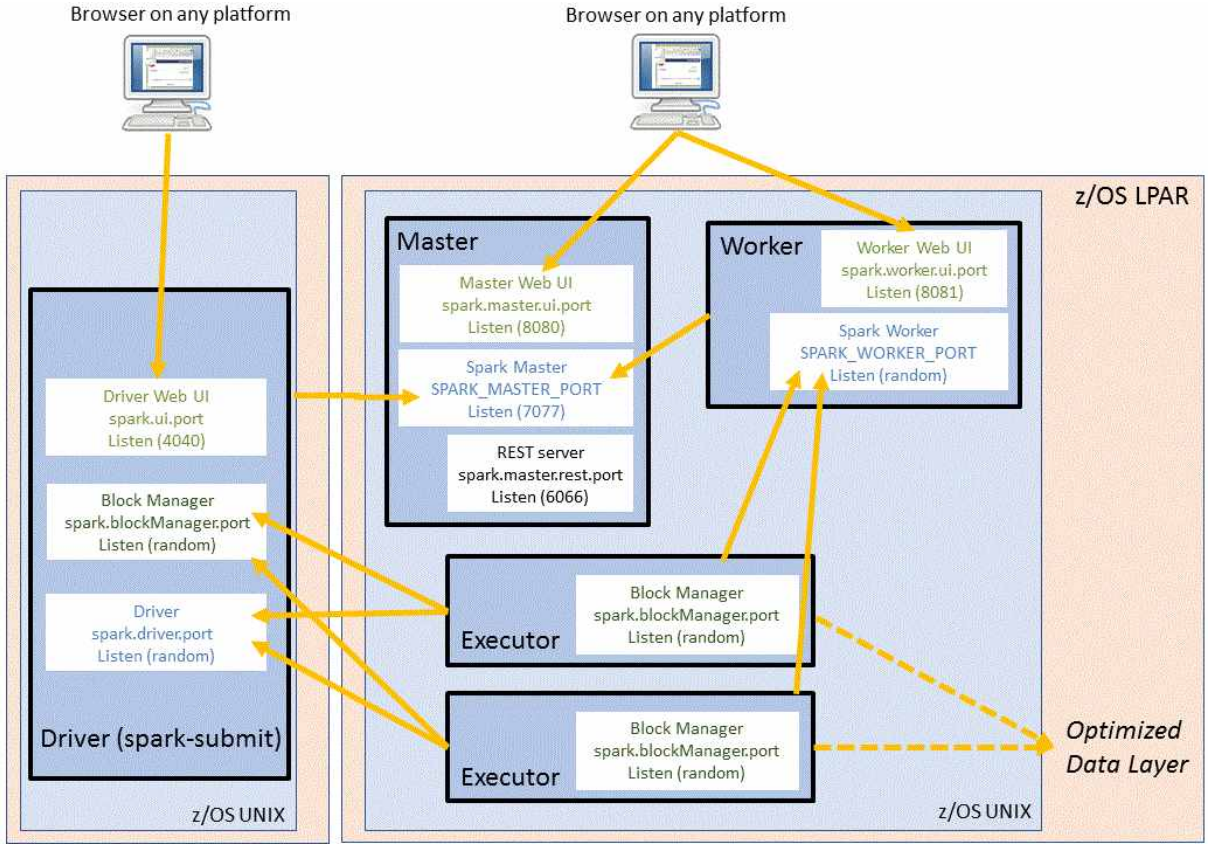
- hdfs dfs -put -f : 파일존재 시 덮어쓰기
- hdfs dfs -get input_dir/data.txt workspace/
  - HDFS 파일시스템 파일을 로컬파일시스템으로 복사(-copyToLocal)
- hdfs dfs -cp input_dir/data.txt output_dir/
  - HDFS 파일시스템 파일을 HDFS 파일시스템으로 복사
- hdfs dfs -mv input_dir/data.txt output_dir/
  - HDFS 파일시스템 파일을 HDFS 파일시스템으로 이동(잘라 붙이기)
- hdfs dfs -getmerge input_dir
  - HDFS 파일시스템 폴더의 파일들을 로컬파일시스템의 파일하나로 병합
  - Mapreduce job output file의 콘텐츠를 읽을 때 사용
- hdfs dfs -appendToFile file1.txt file2.txt input_dir/data.txt
  - 모든 로컬파일을 HDFS 파일시스템의 지정파일에 추가

□ Scala

- Java 바이트코드 사용 ⇨ JVM에서 실행할 수 있으며 Java API 그대로 사용
- Spark는 Scala로 작성
- Scala 다운로드 & 실행: https://www.scala-lang.org/
- 고급시스템설정 ⇒ 환경변수
  - ○ 사용자 변수(U) > 새로 만들기
    ⇒ 변수: SCALA_HOME, 값: C:₩Program Files (x86)₩scala
  - ○ 시스템 변수(S) > path 편집
    ⇒ C:₩program Files (x86)₩scala₩bin
- scala 실행 & 종료

```
> scala
> sys.exit
```

□ Spark



출처: IBM Knowledge Center 홈페이지
Network ports used in a typical Apache Spark environment

- http://spark.apache.org/downloads.html에서 하둡버전에 맞는 압축파일 다운로드
- c:₩spark폴더 생성 후 여기에 압축파일 풀기
- 고급시스템설정 ⇒ 환경변수
  - 사용자 변수(U) > 새로 만들기
    ⇒ 변수: SPARK_HOME, 값: C:₩spark₩spark-X.X.X-bin-hadoopX.X
  - 시스템 변수(S) > path 편집
    ⇒ C:₩spark₩spark-X.X.X-bin-hadoopX.X₩bin

```
> spark-shell          # scala, > pyspark # python, > sparkR # R
```

- Spark UI: Crome 실행 후 localhost:4040
  - sparkR, spark-shell, pyspark 모두 같음

○ 스파크 Standalone 클러스터 구축

- Windows에서는 %SPARK_HOME%₩sbin 지원하지 않음
  - start-master.sh, start-slaves.sh 사용 못함
- /conf/spark-env.sh.~~template~~

```
set JAV_HOME=c:₩java
set HADOOP_PREFIX=c:₩hadoop-2.7.7
set HADOOP_CONF_DIR=%HADOOP_PREFIX%/etc/hadoop
set YARN_CONF_DIR=%HADOOP_CONF_DIR%
set SPARK_YARN_QUEUE=dev
set SPARK_WORKER_INSTANCE=3
```

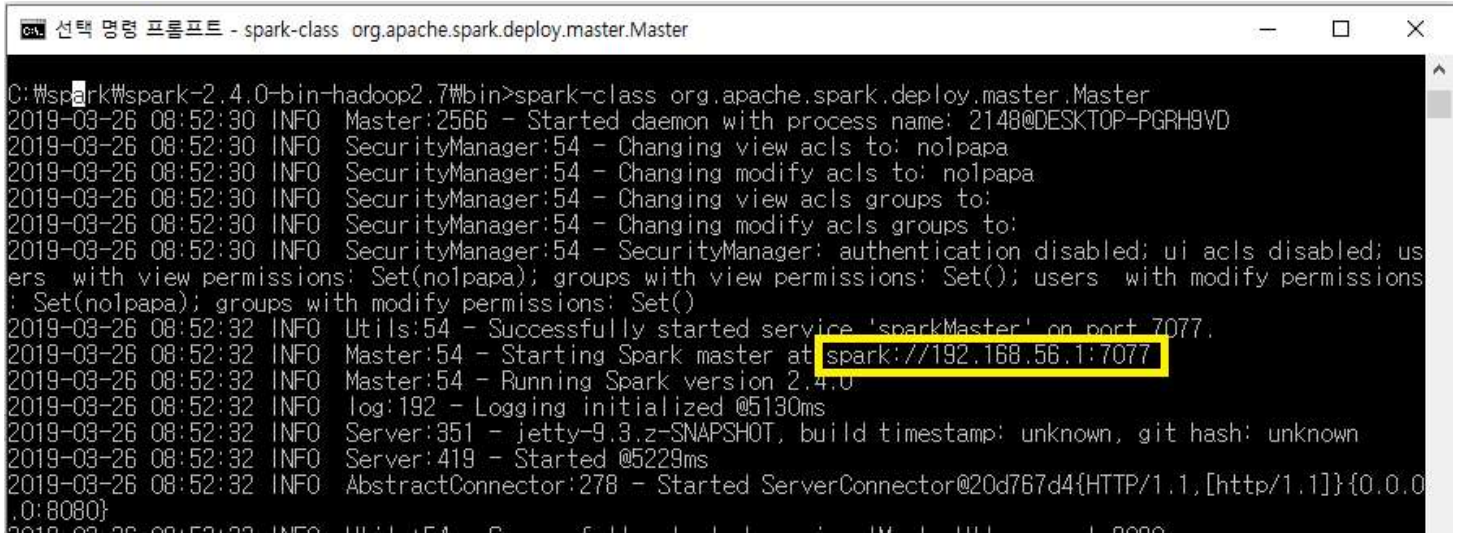
- /conf/spark-defaults.conf.~~template~~

```
spark.executor.instances=1
spark.executor.cores=3
spark.executor.memory=4g
spark.driver.cores=1
spark.driver.memory=4g
```

- %SPARK_HOME%₩examples₩src₩main에 java, scala, python, R 예제 참조
  - 예제: 원주율

```
> spark-submit --master yarn --deploy-mode cluster
  $SPARK_HOME/examples/src/main/python/pi.py 10
```

- /bin spark-class org.apache.spark.deploy.master.Master



  - localhost:8080

- spark-class org.apache.spark.deploy.worker.Worker spark://ip:port –m 516M –c 1
  - -m: worker가 사용할 메모리, –c: 워커가 사용할 코어의 수
  - localhost:8081

- spark-shell --master spark://ip:port

□ R

○ R & RStudio 설치

- R: https://www.r-project.org
- RStudio: https://www.rstudio.com/
- 고급시스템설정 ⇒ 환경변수
  - ○ 시스템 변수(S) > path 편집
    ⇒ C:₩Program Files₩R₩R-X.X.X₩bin₩x64

○ 업데이트

- R

```
> version
> if(!require(installr)) {install.packages("installr"); require(installr)}
> updateR()
# package 업데이트
> update.packages(ask=FALSE)
```

- RStudio 업데이트: Help > Check for Updates

○ RStudio에서의 SparkR

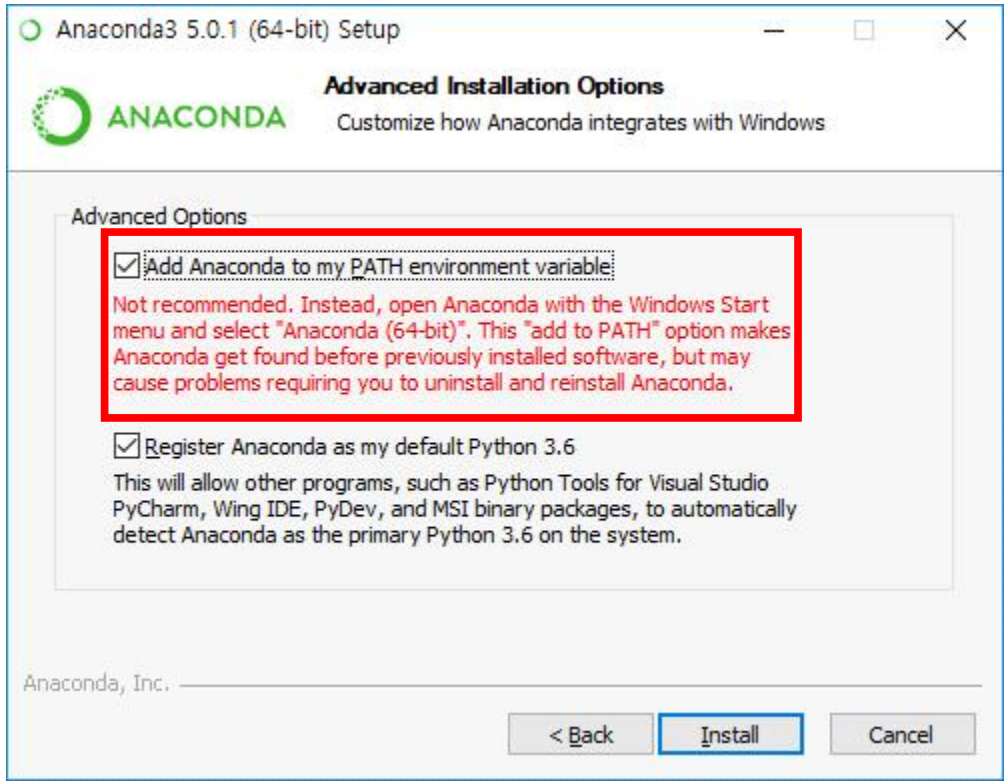
```
# Library Paths 설정
# Sys.setenv(SPARK_HOME = "C:/spark/spark-X.X.X-bin-hadoopX.X")
> .libPaths(c(file.path(Sys.getenv("SPARK_HOME"),"R","lib"),.libPaths()))
# SparkR library 로드
> library(SparkR)
# SparkR Context 초기화
> sc <- sparkR.session(master="local")
# Spark UI: Crome 실행후  localhost:4040
# 예제: sparkR.R 참고
# SparkR 종료
> sparkR.stop()
```

□ Python

○ Anaconda

- **다운로드**
  - https://www.anaconda.com/download/
    - 최신버전 설치 후 원하는 버전 환경 만든 후 다운그레이드
    - Ex) conda install python=3.6
  - https://repo.anaconda.com/archive/에서 원하는 버전 다운로드 후 설치

- 설치 시 주의할 점: path설정



- **Anaconda upgrade**

```
# 버전 확인
> conda list anaconda
# 설치된 anaconda information
> conda info
> conda update -n base conda
> conda update anaconda
# 모든 것(anaconda-navigator, ...) 업데이트
> conda update --all
```

- **Python packages**

```
# 버전 확인
> pip freeze
# 설치 & 업그레이드
> pip install Packagenames
> pip install --upgrade Packagenames
# 제거
> pip uninstall –y Packagename      # -y, --yes
> pip uninstall –r Packagename      # -r, --requirement
# 다운그레이드(버전 지정 x.x.x)
> pip uninstall Packagename
> pip install --upgrade Packagename=x.x.x
# Python에서 spark 패키지 설치
> pip install pyspark
```

○ Conda 환경

- conda 가상환경 만들기 및 제거

```
# conda 가상환경 생성
> conda create -n tensorflow python=3.6
# conda 가상환경 제거
> conda info --envs
> conda remove --name tensorflow --all
```

○ Tensorflow 설치

- **GPU의 경우(Mac OS 제외)**
  - NDIVIA 그래픽 드라이버 설치
    - https://www.nvidia.co.kr/Download/Find.aspx?lang=kr
  - Tensorflow에 맞는 CUDA 버전 다운로드 후 설치(Visual Studio 설치)
    - https://developer.nvidia.com/cuda-downloads
  - 설치된 CUDA에 맞는 cudnn 압축파일 다운로드 후 풀고 CUDA에 덮어쓰기
    - https://developer.nvidia.com/cudnn
    - Program Files > NVIDIA GPU Computing Toolkit > CUDA

- **tensorflow 설치**

```
> pip install tensorflow-gpu
# 주요 packages
> pip install Theano keras
# 버전 확인
> python
>>>import tensorflow as tf
>>>tf.VERSION
```

필요에 따라 프로그램 다운그레이드 필요

예: setuptools downgrade

> pip install setuptools == ****

○ Conda 환경

```
> activate tensorflow
> source activate tensorflow    # server: Linux, macOS인경우
(tensorflow)> conda install tensorflow-gpu          # GPU
(tensorflow)> conda install tensorflow              # CPU
(tensorflow)> conda install tensorflow-gpu==1.12.0  # version 지정
(tensorflow)> pip install --upgrade tensorflow-gpu  # upgrade
# 주요 packages
(tensorflow)> conda install Theano keras jupyter matplotlib
# Conda 환경 종료
(tensorflow)> conda deactivate
```

□ H2O

- hadoop이나 cloud computing 시스템 상에 데이터 분석에 사용
- (R, Python, Java)로 작성 & (...)+scala와 인터페이스
- 다운로드 : https://www.h2o.ai/download/#h2o
- 압축 풀고 해당 폴더에서

> java –jar h2o.jar

  - GUI: Crome(IE, Safari, Firefox)에서 localhost:54321

- **R에서 설치**

```
# 이전에 설치된 H2O 패키지 삭제
if ("package:h2o" %in% search()) { detach("package:h2o", unload=TRUE) }
if ("h2o" %in% rownames(installed.packages())) { remove.packages("h2o") }
# H2O와 종속된 패키지 설치
pkgs <- c("RCurl","jsonlite")
for (pkg in pkgs) {
if (! (pkg %in% rownames(installed.packages()))) { install.packages(pkg) }
}
# H2O 패키지 설치 및 H2O cluster 초기화
install.packages("h2o", type="source",
    repos="http://h2o-release.s3.amazonaws.com/h2o/rel-xu/1/R")
library(h2o)
h2o.init()
```

- **Python에서의 설치**

```
# 사전 패키지 설치
> pip install requests tabulate "colorama>=0.3.8" future
# H2O Python 설치
> pip install
http://h2o-release.s3.amazonaws.com/h2o/rel-xu/1/Python/h2o-3.22.1.1-py2.py3-none-any.whl
# H2O module 제거
> pip uninstall h2o
```

○ Sparkling Water

- H2O의 머신러닝 + Spark
  - Spark SQL ⇨ H2O ⇨ Spark
- 다운로드 : https://www.h2o.ai/download/
- 압축 풀고 해당 폴더에서

```
> cd sparkling-water-2.4.2₩bin
> sparkling-shell --conf "spark.executor.memory=1g"
```

  - scala가 실행됨

```
> import org.apache.spark.h2o._
> val h2oContext = H2OContext.getOrCreate(spark)
> import h2oContext._
```

  - GUI: Crome(IE, Safari, Firefox)에서 localhost:54323

Thank You for Listening

묻지마 시리즈 2탄(AWS?) Coming Soon!!!