KSB 툴박스 설치하기(nvidia-docker버전)

KSB 프레임워크를 쉽게 사용하기 위한 KSB 툴박스 1.0 nvidia-docker 버전 설치 방법을 설명합니다.

설치 요구사항

- Ubuntu 16.04가 설치된 메모리 16GB 이상 데스트탑이나 노트북이 요구됩니다.
- KSB 툴박스는 Hadoop, hbase, kafka, zookeeper, spark 등 빅데이터 처리 프레임워크과 연동합니다. 따라서 다양한 워크플로우 예제를 실행하기 위해서는 메모리 16GB 이상이 요구됩니다. (메모리 8GB 사양 PC에서는 제한적으로 워크플로우 예제 실행이 가능합니다.)

사용시 유의사항

- 제한된 GPU 카드에 대해서만 테스트 되어 있습니다.(Titanx, 1080Ti). 따라서 저사양 GPU 카드나 최신 GPU 카드를 사용하는 경우 동작하지 않을 수 있습니다.
- 현재 KSB 툴박스 nvidia-docker 버전은 CUDA 9.0, CUDNN 7.3.1 이 포함되어 있습니다.

```
cat /usr/local/cuda/version.txt
CUDA Version 9.0.176
cat /usr/include/cudnn.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR 7
#define CUDNN_MINOR 3
#define CUDNN_PATCHLEVEL 1
```

설치 순서

Host PC에 KSB 툴박스를 설치하기 위해서 아래와 같은 절차로 설치합니다.

- [Host PC] csle 사용자 계정(권한: administrator) 생성하기

- [Host PC] hostname을 csle1으로 변경하기
- [Host PC] NVIDIA 그래픽카드 드라이버 설치하기
- [Host PC] java-8-oracle 설치하기
- [Host PC] KSB 툴박스 관련 프로그램 설치하기
- [Host PC] SSH port 변경 및 root 로그인 가능하도도록 SSHD config 수정하기
- [Host PC] SSH config 수정하기
- [Host PC] Docker 설치하기
- [Host PC] nvidia-docker 1.0.1 설치하기
- [Host PC] /etc/hosts 수정하기
- [Host PC] bashrc 설정 추가하기
- [Host PC] 크롬 설치하기
- [Host PC] KSB 툴박스 설치 및 docker image 다운받기
- [Host PC] KSB 툴박스 docker 이미지내 SSH 키를 host pc에 복사하기
- [Host PC] SSH 재시작하기
- [Host PC] ksb-csle 폴더 권한 변경하기
- [Host PC] startDockerCsle.sh, stopDockerCsle.sh 추가하기

[Host PC] csle 사용자 계정(권한 : administrator) 생성하기

Host PC에 기존 Ubuntu 사용자와 설치환경을 분리하기 위해 아래의 명령으로 csle 계정을 추가 생성합니다.

```
sudo adduser csle
    Adding user 'csle' ...
    Adding new group 'csle' (1001) ...
    Adding new user 'csle' (1001) with group 'csle' ...
    Creating home directory '/home/csle' ...
    Copying files from '/etc/skel' ...
    새 UNIX 암호 입력:
    새 UNIX 암호 재입력:
    passwd: password updated successfully
    Changing the user information for csle
    Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
    Is the information correct? [Y/n] Y
```

/etc/sudoers를 열어 csle 계정에 administrator 권한을 추가합니다.
"경고: 읽기 전용 파일을 고치고 있습니다" 경고는 무시하고 저장하고 나옵니다.

```
sudo vi /etc/sudoers
    # User privilege specification
    root    ALL=(ALL:ALL) ALL
    # 맨 아래줄에 내용을 추가합니다.
    #includedir /etc/sudoers.d
    csle    ALL=(ALL) NOPASSWD: ALL
```

[Host PC] hostname을 csle1으로 변경하기

docker 컨테이너가 호스트 pc의 네트워크를 동일하게 사용하는 host 모드로 동작하기 위해 docker 컨테이너와 호스트 pc의 호스트네임을 동일하게 합니다.

```
sudo vi /etc/hostname
csle1
```

PC를 재부팅하여 csle 계정으로 로그인합니다.

[Host PC] NVIDIA 그래픽카드 드라이버 설치하기

```
sudo add-apt-repository ppa:graphics-drivers
sudo apt-get update

왼쪽 우분투 "시스템설정" 아이콘 클릭 --> "자세히 보기" 아이콘 클릭 -->
하단의 "업데이트 설치" 버튼 클릭

왼쪽 우분투 "시스템설정" 아이콘 클릭 --> "소프트웨어&업데이터" 아이콘 클릭 -->
상단 메뉴에서 "추가 드라이버" 메뉴 선택 -->
NVIDIA 드라이버 사용 선택(예:NVIDIA binary driver-version 396.54(독점,확인함)) -->
"바뀐 내용 적용" 후 컴퓨터 재부팅
```

```
csle@csle1:~$ nvidia-smi
Sat Oct 24 13:17:41 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.54                 Driver Version: 396.54                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name       Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  TITAN X (Pascal)   Off  | 00000000:01:00.0  On |                  N/A |
| 25%   40C    P8   17W / 250W |   1186MiB / 12190MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU      PID   Type   Process name                              Usage      |
|=============================================================================|
|    0     1021      G   /usr/lib/jvm/java-8-oracle/bin/java          632MiB |
+-----------------------------------------------------------------------------+
```

PC를 재부팅합니다.

[Host PC] java-8-oracle 설치하기

Host PC에 Ubuntu 16.04를 처음 설치한 경우 java 8을 설치합니다. 아래의 내용을 터미널에 복사하여 설정가능합니다.

```
sudo apt-get update && \
sudo apt-get install -y python-software-properties && \
sudo add-apt-repository -y ppa:webupd8team/java && \
sudo apt-get update && \
echo "oracle-java8-installer \
shared/accepted-oracle-license-v1-1 select true" \
| sudo debconf-set-selections && \
sudo apt-get install -y oracle-java8-installer
```

터미널을 다시 시작한 후, 아래의 명령으로 java 설치 여부를 확인합니다.

```
csle@csle1:~$ java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
```

가끔 오라클에서 jdk 버전을 업데이트 하면 아래와 같은 에러가 발생하며 jdk가 설치되지 않는 경우가 있습니다.

```
접속 download.oracle.com (download.oracle.com)|23.35.220.157|:80... 접속됨.
HTTP request sent, awaiting response... 404 Not Found
2018-10-17 11:08:56 ERROR 404: Not Found.

download failed
Oracle JDK 8 is NOT installed.
dpkg: error processing package oracle-java8-installer (--configure):
 설치한 post-installation 스크립트 하위 프로세스가 오류 1번을 리턴했습니다
처리하는데 오류가 발생했습니다:
 oracle-java8-installer
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

해결방법은 오라클 사이트에 방문하여 최신 버전(2018.10.21. 기준 : 8u191)을 jdk-8u191-linux-x64.tar.gz 파일을 다운받습니다.
https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

그리고, 아래의 스크립트에서 다운받은 파일의 위치와 버전 부분만을 수정한 후, 터미널에 복사하면 직접 설치가능합니다.

```
sudo mkdir /usr/lib/jvm
cd /usr/lib/jvm
sudo tar -xvzf /home/csle/jdk-8u191-linux-x64.tar.gz  # 수정 필요
sudo mv jdk1.8.0_191/ java-8-oracle/  # 수정 필요

sudo truncate -s0 /etc/environment
echo "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:\
/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/jvm/java-8-oracle/bin:\
/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin" |\
sudo tee -a /etc/environment >/dev/null && \
echo "J2SDKDIR=/usr/lib/jvm/java-8-oracle" | \
sudo tee -a /etc/environment >/dev/null && \
echo "J2REDIR=/usr/lib/jvm/java-8-oracle/jre*" | \
sudo tee -a /etc/environment >/dev/null && \
echo "JAVA_HOME=/usr/lib/jvm/java-8-oracle" | \
sudo tee -a /etc/environment >/dev/null && \
echo "DERBY_HOME=/usr/lib/jvm/java-8-oracle/db" | \
sudo tee -a /etc/environment >/dev/null

sudo update-alternatives --install \
"/usr/bin/java" "java" "/usr/lib/jvm/java-8-oracle/bin/java" 0
sudo update-alternatives --install \
"/usr/bin/javac" "javac" "/usr/lib/jvm/java-8-oracle/bin/javac" 0
sudo update-alternatives --set java /usr/lib/jvm/java-8-oracle/bin/java
sudo update-alternatives --set javac /usr/lib/jvm/java-8-oracle/bin/javac
update-alternatives --list java
update-alternatives --list javac
```

[Host PC] KSB 툴박스 관련 프로그램 설치하기

Host PC에 아래의 명령으로 KSB 툴박스 관련 프로그램을 설치합니다. 아래의 내용을 터미널에 복사하여 설정가능합니다.

```
sudo apt-get update && \
sudo apt-get install -y --no-install-recommends apt-utils curl bc jq && \
sudo apt-get install -y ssh locales wget git vim rsync locales \
filezilla python3-pip && \
sudo apt-get install -y net-tools && \
pip3 install kafka-python
```

[Host PC] SSH port 변경 및 root 로그인 가능하도록

SSHD config 수정하기

아래의 명령을 수행하여 포트 정보를 2243로 수정하고, root 로그인을 허용합니다.

```
sudo sed -ri 's/^Port 22/Port 2243/g' /etc/ssh/sshd_config
sudo sed -ri \
's/^PermitRootLogin prohibit-password/PermitRootLogin yes/g' \
/etc/ssh/sshd_config
```

[Host PC] SSH config 수정하기 (known_hosts에 호스트 저장 질문을 하지 않도록 설정)

아래의 명령을 수행하여 ssh_config의 기존 내용을 모두 삭제하고 설정을 추가합니다. 아래의 내용을 터미널에 복사하여 설정가능합니다.

```
sudo truncate -s0 /etc/ssh/ssh_config
echo "Host localhost" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "StrictHostKeyChecking no" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "Host 0.0.0.0" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "StrictHostKeyChecking no" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "Host 127.0.0.1" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "StrictHostKeyChecking no" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "Host csle*" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "StrictHostKeyChecking no" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "UserKnownHostsFile=/dev/null" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "Host master" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "StrictHostKeyChecking no" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
echo "UserKnownHostsFile=/dev/null" \
| sudo tee -a /etc/ssh/ssh_config >/dev/null
sudo service ssh restart
```

[Host PC] Docker 설치하기

Host PC에 docker-ce 프로그램을 설치하기 위해 아래 명령어를 순차적으로 입력합니다. 아래의 내용을 터미널에 복사하여 설정가능합니다.
현재 최신 버전은 18.06.1~ce~3-0~ubuntu 입니다.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
sudo apt-key add - && \
sudo add-apt-repository \
"deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" && \
sudo apt-get update && \
apt-cache policy docker-ce && \
sudo apt-get install -y docker-ce && \
sudo systemctl status docker
```

[참고사항] 기존 docker-engine 삭제 방법

```
sudo apt-get purge -y docker docker-engine docker.io
```

Docker 명령시 root 권한이 필요합니다. root가 아닌 사용자가 sudo 없이 사용하려면 해당 사용자를 docker 그룹에 추가합니다.

```
sudo usermod -aG docker csle
# 사용자가 로그인 중이라면 다시 로그인 후 권한이 적용됩니다.
# 단, 현재 터미널에서만 적용됩니다.
sudo su csle
PC 재부팅
```

PC 재부팅을 통해서 sudo 명령없이 docker 명령을 사용할 수 있습니다.

[Host PC] nvidia-docker 2.0 설치하기

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
sudo apt-key add - && \
curl -s -L \
https://nvidia.github.io/nvidia-docker/\
ubuntu16.04/amd64/nvidia-docker.list | \
sudo tee /etc/apt/sources.list.d/nvidia-docker.list && \
sudo apt-get update && \
sudo apt-get install -y nvidia-docker2 && \
sudo pkill -SIGHUP dockerd
```

아래의 질의에 대해서 Y를 누르면 설치가 완료됩니다.

```
Configuration file '/etc/docker/daemon.json'
 ==> File on system created by you or by a script.
 ==> File also in package provided by package maintainer.
   어떻게 하시겠습니까? 다음 중에 하나를 선택할 수 있습니다:
    Y 또는 I : 패키지 관리자의 버전을 설치합니다
    N 또는 O : 현재 설치된 버전을 유지합니다
       D     : 버전 간의 차이점을 표시합니다
       Z     : 프로세스를 백그라운드로 하고 상황을 알아봅니다
기본값으로 현재 버전을 그대로 유지합니다.
*** daemon.json (Y/I/N/O/D/Z) [기본값=N] ? Y
```

```
csle@csle1:~$ nvidia-docker version
NVIDIA Docker: 2.0.3
Client:
 Version:           18.06.1-ce
 API version:       1.38
 Go version:        go1.10.3
 Git commit:        e68fc7a
 Built:             Tue Aug 21 17:24:56 2018
 OS/Arch:           linux/amd64
 Experimental:      false
```

```
Server:
 Engine:
  Version:          18.06.1-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.3
  Git commit:       e68fc7a
  Built:            Tue Aug 21 17:23:21 2018
  OS/Arch:          linux/amd64
  Experimental:     false
```

[참고사항] 기존 nvidia-docker 1.0 버전을 삭제 방법

```
docker volume ls -q -f driver=nvidia-docker | \
xargs -r -I{} -n1 docker ps -q -a -f volume={} | xargs -r docker rm -f
sudo apt-get purge nvidia-docker
```

[Host PC] /etc/hosts 수정하기

KSB 툴박스의 docker 컨테이너 주소에 쉽게 접근하기 위해 아래와 같이 /etc/hosts 내용을 수정합니다. (host pc의 IP는 192.168.0.5로 가정합니다. 자신의 PC IP에 맞게 설정합니다.)

```
127.0.0.1       localhost
# 아래 주석처리. Hdfs 연동시 문제가 생김.
#127.0.1.1      csle1

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# 아래의 내용 추가
192.168.0.5     csle1 master
```

[Host PC] bashrc 설정 추가하기

csle 계정의 bashrc 파일을 업데이트 합니다. 아래의 내용을 터미널에 복사하여 설정가능합니다.

```
cat <<EOT >> ~/.bashrc
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export PATH=\$PATH:\$JAVA_HOME/bin
export KSB_HOME=/home/csle/ksb-csle
export PYTHONPATH=/home/csle/ksb-csle/pyML/:\$PYTHONPATH
export PYTHONPATH=./:/home/csle/ksb-csle/ksblib:\$PYTHONPATH
EOT
source ~/.bashrc
```

[Host PC] 크롬 설치하기 (optional)

KSB 툴박스의 웹툴킷을 안정적으로 사용하기 위해서 크롬 브라우저를 설치합니다. 아래의 내용을 터미널에 복사하여 설정가능합니다.

```
cd && \
wget \
https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo apt-get update && \
sudo apt-get install -y libxss1 libappindicator1 libindicator7 && \
sudo dpkg -i google-chrome-stable_current_amd64.deb
```

[Host PC] KSB 프레임워크 설치 및 docker image 다운받기

다운로드 페이지로부터 KSB 툴박스(docker image 별도) 파일을 다운로드합니다.

- ksb_toolbox_v1.tar.gz : KSB 툴박스 파일

csle 사용자 계정의 home에 ksb_toolbox_v1.tar.gz 압축을 해제합니다. 결과적으로 /home/csle/ksb-csle 폴더가 생성됩니다.

```
tar zxvf ksb_toolbox_v1.tar.gz -C /home/csle/
```

/home/csle/ksb-csle 폴더는 다음의 하위 폴더들로 구성됩니다.

- /bin : KSB 프레임워크 shell 스크립트 파일 저장 폴더
- /components : tensorflow train 예제 프로그램 저장 폴더

- /conf : KSB 프레임워크 환경설정 파일 저장 폴더
- /docker : KSB 툴박스 docker 컨테이너 실행 스크립트 저장 폴더
- /examples : CLI(Command Line Interface)를 통해 워크플로우 생성 및 submit 할 수 있는 프로젝트 폴더
- /jars : KSB 프레임워크 1.0 버전의 jar 파일 저장 폴더
- /ksblib : python 전처리 라이브러리 폴더
- /kubernetes : 쿠버네티스 환경설정 폴더
- /logs : log 저장 폴더
- /pyML : autoML python 프로젝트 폴더
- /tools: 예제 테스트를 위한 프로그램(jmeter, hadoop, kafka, maven, .ssh, webToolkit_db)

Host PC에서 다음 명령으로 KSB 툴박스 docker image를 docker hub로부터 pull 합니다.

```
docker pull ksbframework/ksb_toolbox_nvidia:1.0.0
```

Host PC에서 다음 명령으로 KSB 툴박스 docker image를 확인합니다.
(참고사항: 현재 docker image 버전은 1.0.0 버전이며, 추후 TAG 정보 및 SIZE 정보는 배포 버전에 따라 다를 수 있습니다)

```
csle@csle1:~/ksb-csle$ docker images
REPOSITORY                         TAG    IMAGE ID      CREATED     SIZE
ksbframework/ksb_toolbox_nvidia  1.0.0  fc79c32f3fee  7 hours ago 13.3GB
```

[Host PC] KSB 툴박스 docker 이미지내 SSH 키를 host pc에 복사하기

KSB 툴박스 docker 컨테이너의 인증키를 공유하기 위해 ksb-csle/tools 폴더에 있는 .ssh 폴더를 csle 홈에 카피합니다.

```
cp -r ~/ksb-csle/tools/.ssh/ /home/csle/
```

[Host PC] SSH 재시작하기

```
sudo service ssh restart
```

[Host PC] ksb.conf 수정하기

KSB 툴박스 nvidia 이미지 버전은 tensorflow 1.6 gpu 버전과 cpu 버전이 설치되어있습니다. 따라서, 아래와 같이 ksb.conf 파일을 수정합니다.

```
tensorflow {
    enable = "true"
#       python_path = "/anaconda3/envs/tensorflow/bin/python"
    python_path = "/anaconda3/envs/tensorflow16_gpu/bin/python"
    python_code_project_path = "/analysis"
}
```

[Host PC] ksb-csle 폴더 권한 변경하기

KSB 툴박스 docker 컨테이너에서 Host PC의 ksb-csle 폴더를 엑세스 할 수 있도록 폴더 권한을 변경합니다.

```
sudo chmod 777 -R /home/csle/ksb-csle/
```

[Host PC] startDockerCsle.sh stopDockerCsle.sh 추가하기

아래의 스크립트를 복사하여 터미널에서 실행하면 startDockerCsle.sh 파일이 생성됩니다.

```
mkdir -p /home/csle/ksb-csle/docker/1.0-nvidia
cat <<EOT >> /home/csle/ksb-csle/docker/1.0-nvidia/startDockerCsle.sh
#!/bin/bash
sudo service postgresql stop

nvidia-docker network rm csle_cluster csle_standalone

nvidia-docker rm -f csle1
echo "start csle1 slave container..."
nvidia-docker run --rm -itd \\
                --net=host \\
                -v /home/csle/ksb-csle:/home/csle/ksb-csle \\
                -v /etc/localtime:/etc/localtime:ro \\
                -v /etc/timezone:/etc/timezone \\
                --user=csle \\
                --name=csle1 \\
                --hostname=csle1\\
                ksbframework/ksb_toolbox_nvidia:1.0.0 bash

nvidia-docker exec -it csle1 bash
EOT
chmod +x /home/csle/ksb-csle/docker/1.0-nvidia/startDockerCsle.sh
cat <<EOT >> /home/csle/ksb-csle/docker/1.0-nvidia/stopDockerCsle.sh
#!/bin/bash
nvidia-docker rm -f csle1
EOT
chmod +x /home/csle/ksb-csle/docker/1.0-nvidia/stopDockerCsle.sh
```

KSB 인공지능 프레임워크 실행하기 위해 KSB 인공지능 프레임워크 실행하기 페이지로 이동하여 KSB 프레임워크를 구동합니다.
단, nvidia-docker 버전 KSB 프레임워크 구동시 기존 cpu버전 툴박스와의 차이점은 위에서 생성한 /home/csle/ksb-csle/docker/1.0-nvidia/startDockerCsle.sh을 실행하여 KSB 툴박스 docker 컨테이너를 실행하고 접속하는 것일 뿐 모두 동일합니다.

```
csle@csle1:~/ksb-csle/docker/1.0-nvidia$ ./startDockerCsle.sh
Error response from daemon: network csle_cluster not found
Error response from daemon: network csle_standalone not found
csle1
start csle1 slave container...
4918997cd2dac93fe0b52f3df3a45a46db6df1f3f052854061696695451d2f22
csle@csle1:/$
```

이상으로 nvidia-docker 버전 KSB 툴박스를 구동하기 위해 준비가 완료되었습니다.

KSB 인공지능 프레임워크 실행하기 페이지로 이동하여 KSB 프레임워크를 구동하도록 하겠습니다.