

posco

포항 AI서버 렌탈 서비스 표준환경 구성 매뉴얼



최의규 (Syper)

2022-03-30 16:26:21

내용

1	AI과제 환경 요건 분석.....	4
1.1	과제환경 검토사항.....	4
1.1.1	과제환경 자원 제한 설정.....	4
1.1.2	과제환경 자원 할당 기준.....	4
1.1.3	과제(컨테이너) 이미지 크기	4
1.1.4	과제 특성	4
1.1.5	자원 증설 시점	4
1.2	AI과제 환경 검토 양식	5
2	Container Image 제작	9
2.1	참고사항	19
2.1.1	표준 TASK 이미지 제작	26
2.1.2	AI-Model 이미지 제작	9
3	Private Registry에 Image Upload 및 이미지 관리 ..오류! 책갈피가 정의되어 있지 않습니다.	
3.1	Image push.....오류! 책갈피가 정의되어 있지 않습니다.	
3.1.1	Image push 방법 오류! 책갈피가 정의되어 있지 않습니다.	
4	NAS 内 데이터 저장을 위한 디렉토리 생성	19
4.1	디렉토리 생성	오류! 책갈피가 정의되어 있지 않습니다.
5	Chart 신규 구성	19
5.1	Chart.yaml 수정	20
5.2	value.yaml 수정	21
6	Chart 신규 등록 및 컨테이너 기동 ..오류! 책갈피가 정의되어 있지 않습니다.	
6.1	Chart 신규 등록	25
6.2	신규 등록된 chart로 pod 및 컨테이너 기동 ..오류! 책갈피가 정의되어 있지 않습니다.	



최의규 (syper)

2022-03-30 16:26:21

1 AI과제 환경 요건 분석

1.1 과제환경 검토 사항

1.1.1 과제환경 자원 제한 설정

과제 환경 제공 시 CPU, Memory, GPU 자원 제한을 할 수 있지만, 제한 값 이상을 사용하게 되면 오버헤드가 꽤 발생하므로, 이 부분을 고려해야 합니다. 일반적으로는 특정 과제환경이 자원을 과도하게 사용하거나 제한이 꼭 필요한 경우에만 사용하는 것을 권고드립니다.

1.1.2 과제환경 자원 할당 기준

과제환경에 대한 자원 할당 관련하여 명확한 기준은 존재하지 않으며, 과제 별로 편차가 존재합니다. PosFrame AI에서는 제공하는 과제 환경에는 기본적으로 CPU 6 Core, Memory 48GB를 산정하여 Limit 설정하고 있으나, 일부러 다소 여유있게 제공하고 있습니다.

운영 경험으로 봤을 때, 영상 과제의 경우 사용량이 높아서 보통 Memory를 약 40GB 사용하는 경우도 있고, 일반적인 text 과제인 경우에는 cpu 1Core 이하, Memory도 5G 이하를 사용하는 경우도 있었습니다. 아래 내용은 대략적인 산정 기준으로 참고하시기 바랍니다.

- text 분석 과제 : CPU 약 1-2 Core 이내, Memory는 약 10G 이내
- 영상분석과제 : CPU 약 3-4Core 이내, Memory는 최대 약 40G 이내

1.1.3 과제(컨테이너) 이미지 크기

컨테이너 이미지의 크기는 되도록 10G 이내, 크더라도 15G 이내가 적당합니다. 하지만 영상이나 음성 과제처럼 라이브러리가 많고 용량이 큰 특별한 경우에는 이미지 크기 2-30G 이상 감수해야 합니다. 단, 이 경우에는 이미지가 다운로드되지 않은 서버에서 이미지를 Pull할 경우, 상당히 많은 시간이 소요된다는 점을 고려해야 합니다. 이미지 Pull 시간이 최대 10분 정도 걸리는 경우도 있습니다.

1.1.4 과제 특성

컨테이너는 서버 장애 시 컨테이너가 다른 노드에서 정상적으로 기동될 수 있도록, 가용성을 위해 일반적으로 NAS 스토리지를 사용합니다. 따라서 트랜잭션이 아주 많거나 대용량의 미션 크리티컬한 데 이터베이스, 0.5초 이내의 아주 빠른 실시간 응답이 필요한 경우, 과제 환경 사이즈가 과도하게 큰 경우에는 적합하지 않으며, 이 경우 상세한 검토가 필요합니다.

1.1.5 자원 증설 시점

클러스터 전체의 사용량이 50-60% 수준 또는 초과하는 경우, GPU 자원이 부족한 경우 증설 고려

1.2 Anaconda – Python 버전 매트릭스

참고사이트 : <https://docs.anaconda.com/anaconda/reference/release-notes/>

Anaconda	Release date	Conda	Python (Support)
4.3.0	2017-01-31	conda from 4.2.9 to 4.3.8	Python 3.6.0
4.3.0.1	2017-02-03	conda from 4.3.8 to 4.3.14	
4.4.0	2017-05-31	conda from 4.3.14 to 4.3.21	python 3.5 from 3.5.2 to 3.5.3 python 3.6 from 3.6.0 to 3.6.1
5.0.0.1	2017-10-02	conda 4.3.21 -> 4.3.27	python 3.5.3 -> 3.5.4 python 3.6.1 -> 3.6.2
5.1.0	2018-02-15	conda 4.3.27 -> 4.4.10	python 3.6.3 -> 3.6.4
5.2.0	2018-05-30	conda 4.5.4	python 3.6.4 -> 3.6.5 (Not Work)
5.3.1(5.3.0)	2018-11-19		python 3.6.5 -> 3.7.0
2018.12	2018-12-21	conda 4.5.12	python 3.7.0 -> 3.7.1
2019.03	2019-04-04	conda 4.6	python 3.7.1 -> 3.7.3
2019.07	2019-07-24	Conda 4.7.10	
2019.10	2019-10-15		python 3.7.3 -> 3.7.4
2020.02	2020-03-11		python 3.7.4 -> 3.7.6
2020.07	2020-07-23		python 3.8.1 -> 3.8.3
2020.11	2020-11-19		python 3.8.3 -> 3.8.5

※ python 버전에 따른 Anaconda 버전 권고

python 3.6 : Anaconda 5.1.0 버전

python 3.7 : Anaconda 2020.02 버전

python 3.8 : Anaconda 2020.11 버전

1.3 Tensorflow – CUDA 버전 매트릭스

버전	Python 버전	컴파일러	빌드 도구	cuDNN	CUDA	검증 확인
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8	11	
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1	
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1	
tensorflow-2.1.0	2.7, 3.5-3.7	GCC 7.3.1	Bazel 0.27.1	7.6	10.1	
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10	
tensorflow_gpu-1.15.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10	
tensorflow_gpu-1.14.0	2.7, 3.3-3.7	GCC 4.8	Bazel 0.24.1	7.4	10	
tensorflow_gpu-1.13.1	2.7, 3.3-3.7	GCC 4.8	Bazel 0.19.2	7.4	10	python 3.6
tensorflow_gpu-1.12.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9	
tensorflow_gpu-1.11.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9	
tensorflow_gpu-1.10.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.15.0	7	9	
tensorflow_gpu-1.9.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.11.0	7	9	
tensorflow_gpu-1.8.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.10.0	7	9	
tensorflow_gpu-1.7.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0	7	9	
tensorflow_gpu-1.6.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.9.0	7	9	
tensorflow_gpu-1.5.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.8.0	7	9	
tensorflow_gpu-1.4.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.5.4	6	8	
tensorflow_gpu-1.3.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	6	8	
tensorflow_gpu-1.2.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.5	5.1	8	
tensorflow_gpu-1.1.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8	
tensorflow_gpu-1.0.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.4.2	5.1	8	

※ 참고 사항

현재 python 3.6 기준으로 Anaconda 패키지 설치 후, 기존에 이미지 빌드를 하기 위해 사용하던 conda 명령어를 이용한 컨테이너 이미지 빌드가 정상적으로 되지 않습니다. (dependency 체크에서 진행되지 않음)

이미지 빌드를 위해 python 3.6 버전의 경우 conda가 아닌 pip를 통해서 tensorflow를 설치하는 것으로 변경하였으며, pip를 이용하여 tensorflow 1.13.1 버전의 경우 정상 동작을 확인하였으나, 1.14.0 ~ 부터는 호환성 문제로 정상적으로 이미지 빌드가 되지 않는 것을 확인하였습니다.

따라서 3.6 버전의 이미지가 필요한 경우 tensorflow 1.13.1 버전을 활용하시거나, 또는 기존에 제작된 이미지를 활용하시기 바랍니다.

1.4 Pytorch - CUDA 버전 매트릭스 (참고 사이트)

<https://pytorch.org/get-started/previous-versions/>

1.5 Image Build 시 주요 에러 (업데이트 예정)

1.5.1 *ERROR: Cannot uninstall 'wrapt'. It is a distutils installed project and thus we cannot accurately determine which files belong to it which would lead to only a partial uninstall.*

cuda, python, tensorflow 그리고 기타 의존성이 있는 라이브러리 간 호환성이 존재하는데, 분석환경 구성을 위해서는 이것들 간의 호환성이 맞아야 합니다. 이 에러는 python과 tensorflow 버전 간 호환성 문제일 가능성이 높으며, 일반적으로 python 버전에 맞는 적절한 tensorflow 버전을 지정하여 해결할 수 있습니다.

1.5.2 *ImportError: libGL.so.1: cannot open shared object file: No such file or directory*

Import 시에 위와 같은 에러가 발생한다면, 컨테이너에 필요한 라이브러리가 없거나 버전이 맞지 않아 다른 이름으로 존재하는 경우입니다. 일반적으로는 mesa-libGL-devel 패키지를 추가 설치하여 해결할 수 있습니다.

1.5.3 *ImportError: libcublas.so.10.0: cannot open shared object file: No such file or directory*

이 에러는 tensorflow에서 참조하는 cuda 라이브러리 버전이 맞지 않거나 라이브러리를 찾지 못하는 경우입니다. 예를 들어, 10.2버전 cuda 이미지에서 tensorflow 1.13.1 버전을 설치한 경우 tensorflow에서는 cuda 10.0.0 라이브러리를 찾기 때문에 발생하는 에러입니다. 따라서 Tensorflow-CUDA 버전 매트릭스를 참조하여, Dockerfile의 base image를 적절한 버전의 CUDA 이미지로 지정해야 합니다.

1.5.4 *ImportError: libcuda.so.1: cannot open shared object file: No such file or directory*

이 에러는 tensorflow에서 참조하는 cuda 라이브러리가 설치되어 있지 않거나 찾지 못하는 경우입니다. 예를 들어, 10.0.0 버전의 cuda 라이브러리에서는 cuda.so.1 라이브러리가 /usr/local/cuda/compat에 존재하는데, 해당 경로가 LD_LIBRARY_PATH에 지정되어 있지 않으므로 에러가 발생합니다. 따라서 라이브러리가 설치가 안된 경우에는 tensorflow-CUDA 버전 매트릭스를 참조하여 cuda 라이브러리를 설치하거나, 이미 설치된 Base Image를 사용하시기 바랍니다. 만일 라이브러리가 설치되어 있으나 찾지 못하는 경우에는 tensorflow에서 참조하는 cuda 라이브러리 경로를 LD_LIBRARY_PATH에 지정하여 해결할 수 있습니다. (python36-gpu Dockerfile 참조)

1.6 AI과제 환경 접수 샘플 양식 (참고)

아래의 샘플 양식을 참고하여, 필요한 과제환경 정보를 확인 (예제로 과제 성격에 따라 조사 필요)

항목	조사 내용 (예시)	설명
과제명 (영문/한글)	한글 : XX 공장 xx 라인의 ??? 개선을 위한 AI 모델 적용 영문 :	영문명은 과제명 기준으로 작성
AI 모델 수량	??? 모델, ??? 모델	과제별 수행되는 AI 모델명 및 수량
AI 환경 (기본)	- Anaconda 버전, Python 버전	표준 개발 버전
AI 환경 (Framework)	- Tensorflow(cpu type) / Tensorflow-gpu 1.15.0 - Pytorch 1.18.0	ML/DL Framework
AI 환경 (추가 Library)	- theano 1.0.4 등	표준 AI python/R 환경기준으로 과제 환경에 필요한 Package 명 및 버전
AI 환경 (R)	R 3.6. zoo 1.4, car 2.0	
필요 자원 (CPU,Memory, GPU)	CPU : 6Core	과제에서 예상되는 CPU, Memory, GPU 필요 수량
	Memory : 32GB	
	GPU : 1EA	
DATA 저장 용량	DB Data: 50GB 이내 AI-model Source: 20GB 이내	AI-model : 20GB , DB : 50GB 용량 제한
GPU 사용 기간	개발기간만 사용 ('20.10.1~12.E)	
과제원 정보	홍길동 : abcdefg (직번 : 000000)	과제원 정보

2 [AI-model 이미지] 환경 구성

2.1 AI-Model 개발환경 이미지 제작

Python이나 Tensorflow 등 S/W 버전 상의 문제로 기존에 제공된 AI 모델 개발 환경을 추가로 제작해야 하는 경우 사용. (기존 Image를 사용할 경우에는 skip)

기존 제작된 AI-model의 Dockerfile을 복사하여 수정하면 쉽게 업데이트 가능합니다.

※ Image build 작업은 인터넷이 되는 서버에서 패키지 설치 필요 : 172.16.185.32 서버에 접속

2.1.1 Python 기반 이미지 빌드 (신규 이미지 생성)

기존 제작된 AI-model 환경구성 폴더에 접속 후, 유사한 과제환경을 확인하고 아래와 같이 새로운 과제환경 이름으로 복사합니다.

```
# cd /build/standard/analysis
# mkdir /build/과제명
# cp -r /build/standard/ /build/과제명/
# cd /build/과제명/analysis/
```

※ 기존 제작 AI-model 과제환경 태입(정보)

디렉토리	내용
python36	Python3.6기반 패키지 설치 파일
python36-gpu	Python3.6기반 패키지 + tensorflow (1.14.0) 설치
python37-gpu	Python3.7기반 패키지 + tensorflow (1.15.0) 설치
python38-gpu	Python3.7기반 패키지 + tensorflow (1.15.0) 설치
python36-gpu-nanomsg	nanomsg 라이브러리 추가 설치 (EGL용)
R-Studio	R 3.6.0 + R 패키지

복사한 과제환경 폴더에 접속하여 Dockerfile을 수정합니다.

2.1.1.1 Dockerfile 수정

※ 참고로 python36-gpu 디렉토리의 Dockerfile기준으로 설명

- 디렉토리 내 파일 구조

파일명	내용
Dockerfile	Docker Image build 설정 파일
additional_packages.txt	Anaconda와 기본 Framework(tensorflow, pytorch) 외에 추가로 설치할 python 패키지 리스트 예시) theano 1.0.4 설치
fix-permissions	빌드 시 디렉토리 권한/소유권 설정 스크립트
jupyter_notebook_config.py	Jupyternotebook 환경설정 파일
start-notebook.sh	최초 호출 스크립트로 JupyterLAB 사용여부 설정 스크립트 : start-singleuser.sh 호출
start-singleuser.sh	Jupyter notebook을 singleuser로 실행하는 스크립트 : start.sh를 single-user 모드로 호출
start.sh	Jupyter notebook 실행 스크립트

- Dockerfile 작성 예제 (Anaconda 2020.02, python3.7(python37-gpu), R 3.6)

※ 참고사항

- Base Image의 정확한 이름 및 버전은 <http://hub.docker.com> 에 접속하여 nvidia/cuda로 검색
- Tensorflow, Pytorch 등 Python의 라이브러리 이름 및 버전은 <https://pypi.org> 에 접속하여 검색
또는 conda가 설치된 경우 conda search [패키지명]으로 검색 ex) [conda search tensorflow](#)

vi Dockerfile **※ 업무 요건에 맞게 앞부분 설정 부분의 붉은 색 라인 수정 필요**

```
#####
# Configuration Begin #####
#####

# Base Container Image
FROM nvidia/cuda:10.2-cudnn7-devel-centos7      # 최초 이미지 빌드 시에 사용할 base 이미지 지정

# Anaconda & Python version config
ARG ANACONDA_VERSION=2020.02                      # Anaconda 버전 지정

# Tensorflow version config - comment below if you don't want to install tensorflow
ARG TENSORFLOW_VERSION=1.15.0                         # Tensorflow 버전 지정, 미 설치 시 주석 처리

# Pytorch version config - comment below if you don't want to install pytorch
ARG PYTORCH_VERSION=1.8.0                            # Pytorch 버전 지정, 미 설치 시 주석 처리

# Specify framework type. You can set type 'gpu' or 'cpu'
# type 'gpu' will install tensorflow-gpu and cudatoolkit
ARG ML_TYPE=gpu                                      # tensorflow-gpu 설치 시 gpu, cpu용 tensorflow는 cpu 지정
```

```
#####
# Configuration End #####
#####

LABEL maintainer="CloudHUB <support@brickcloud.co.kr>

ARG NB_USER="posco"
ARG NB_UID="1000"
ARG NB_GID="100"

# Fix DL4006
SHELL ["/bin/bash", "-o", "pipefail", "-c"]

USER root

# Install OS dependencies
RUN yum-config-manager --quiet --disable cuda nvidia-ml > /dev/null && \
    yum install -y bzip2 wget sudo ca-certificates && \
    yum update -y && \
    yum clean all && rm -rf /var/cache/yum/* /tmp/* /root/.cache

# Configure Container Environment
ENV CONDA_DIR=/opt/conda \
    SHELL=/bin/bash \
    NB_USER=$NB_USER \
    NB_UID=$NB_UID \
    NB_GID=$NB_GID \
    LC_ALL=en_US.UTF8 \
    LANG=en_US.UTF8 \
    LANGUAGE=en_US.UTF8 \
    PIP_USE_FEATURE=2020-resolver
ENV PATH=$CONDA_DIR/bin:$PATH \
    HOME=/home/$NB_USER
ENV TZ=Asia/Seoul

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && \
    echo $TZ > /etc/timezone

# Copy a permission script
COPY fix-permissions /usr/local/bin/fix-permissions
RUN chmod a+r /usr/local/bin/fix-permissions

# Enable prompt color in the skeleton .bashrc before creating the default NB_USER
RUN sed -i 's/^#force_color_prompt=yes/force_color_prompt=yes/' /etc/skel/.bashrc
```

```

# Create NB_USER
RUN echo "auth requisite pam_deny.so" >> /etc/pam.d/su && \
    sed -i.bak -e 's/^%admin/#%admin/' /etc/sudoers && \
    sed -i.bak -e 's/^%sudo/#%sudo/' /etc/sudoers && \
    useradd -m -s /bin/bash -N -u $NB_UID $NB_USER && \
    mkdir -p $CONDA_DIR && \
    chown $NB_USER:$NB_GID $CONDA_DIR && \
    chmod g+w /etc/passwd && \
    fix-permissions $HOME && \
    fix-permissions $CONDA_DIR

USER $NB_UID
WORKDIR $HOME

# Setup work directory for backward-compatibility
RUN mkdir /home/$NB_USER/work && \
    fix-permissions /home/$NB_USER

# Install Anaconda
WORKDIR /tmp

RUN curl -k -o /tmp/Anaconda3-${ANACONDA_VERSION}-Linux-x86_64.sh \
https://repo.anaconda.com/archive/Anaconda3-${ANACONDA_VERSION}-Linux-x86_64.sh && \
/bin/bash Anaconda3-${ANACONDA_VERSION}-Linux-x86_64.sh -f -b -p $CONDA_DIR && \
rm Anaconda3-${ANACONDA_VERSION}-Linux-x86_64.sh && \
conda config --set ssl_verify false && \
conda config --system --prepend channels conda-forge && \
conda config --system --set auto_update_conda false && \
conda config --system --set show_channel_urls true && \
conda config --system --set channel_priority flexible && \
#conda config --system --set channel_priority strict && \
if $PYTHON_VERSION; then conda install --quiet --yes python=$PYTHON_VERSION; fi && \
conda list python | grep '^python' | tr -s ' ' | cut -d '!' -f 1,2 | sed 's/$.*/' >> $CONDA_DIR/conda- \
meta/pinned && \
conda install --update-deps --quiet --yes conda && \
#conda install --quiet --yes pip && \
pip install --upgrade --no-cache-dir pip && \
conda update --all --quiet --yes && \
conda clean --yes --all && \
rm -rf /home/$NB_USER/.cache && \
fix-permissions $CONDA_DIR && \
fix-permissions /home/$NB_USER

# Install Tini

```

```
RUN conda config --set ssl_verify false && \
    conda install --quiet --yes 'tini=0.18.0' && \
    conda list tini | grep tini | tr -s ' ' | cut -d ' ' -f 1,2 >> $CONDA_DIR/conda-meta/pinned && \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

# Install Tensorflow(gpu type)
RUN if [ "$ML_TYPE" = "gpu" -a $TENSORFLOW_VERSION ]; then \
    conda config --set ssl_verify false && \
    conda install --quiet --yes tensorflow-gpu=$TENSORFLOW_VERSION && \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER; \
fi

# Install Tensorflow(cpu type)
RUN if [ "$ML_TYPE" = "cpu" -a $TENSORFLOW_VERSION ]; then \
    conda config --set ssl_verify false && \
    conda install --quiet --yes tensorflow=$TENSORFLOW_VERSION && \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER; \
fi

# Install Pytorch(gpu type)
RUN if [ "$ML_TYPE" = "gpu" -a $PYTORCH_VERSION ]; then \
    conda config --set ssl_verify false && \
    conda install --quiet --yes pytorch=$PYTORCH_VERSION torchvision cudatoolkit -c pytorch -c conda-forge \
&& \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER; \
fi

# Install Pytorch cpu-type
RUN if [ "$ML_TYPE" = "cpu" -a $PYTORCH_VERSION ]; then \
    conda config --set ssl_verify false && \
    conda install --quiet --yes pytorch=$PYTORCH_VERSION torchvision -c pytorch -c conda-forge && \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER; \
fi
```

```

# Install Additional Libraries
COPY additional_packages.txt /tmp/

RUN if grep -E -v '^$|^#' additional_packages.txt; then \
    conda config --set ssl_verify false && \
    conda install --quiet --yes --file additional_packages.txt && \
    conda clean --yes --all && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER; \
fi

# Configure Jupyter
RUN conda config --set ssl_verify false && \
    conda clean --yes --all && \
    jupyter notebook --generate-config && \
    rm -rf $CONDA_DIR/share/jupyter/lab/staging && \
    rm -rf /home/$NB_USER/.cache/yarn && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

EXPOSE 8888

# Configure container startup
ENTRYPOINT ["tini", "-g", "--"]
CMD ["start-notebook.sh"]

# Copy local files as late as possible to avoid cache busting
COPY start.sh start-notebook.sh start-singleuser.sh /usr/local/bin/
COPY jupyter_notebook_config.py /etc/jupyter/

# Fix permissions on /etc/jupyter as root
USER root
RUN fix-permissions /etc/jupyter/

USER $NB_USER
ADD --chown=1000:100 test $HOME/test

WORKDIR $HOME

```

2.1.1.2 Image build 및 Image 조회

이미지 빌드 시에는 반드시 앞부분에 posco.po/hub/를 붙여서 이미지를 제작해야 한다. posco.po/hub는 이미지 저장소의 위치(URL)을 나타낸다.

```
# docker build -t posco.po/hub/imagename:tag .
```

예시) # docker build -t posco.po/hub/test-standard:py36 .

- image build가 정상적으로 완료시 아래와 같이 나타남.

```
Successfully built 7ad1f46cff01
Successfully tagged posco.po/hub/test-standard:py36
# docker images | grep "imagename"      # 빌드된 이미지 확인
[root@poscoai01 python36]# docker images |grep test-standard
posco.po/hub/test-standard                                     py36
7ad1f46cff01... 14 minutes ago    3.34GB
```

2.1.1.3 이미지 업로드

이미지를 해당 서버에서 빌드한 이후에는, 아래와 같이 공식 이미지 저장소로 업로드를 해야 합니다.

※ 참고로 이미지 이름 앞부분 posco.po는 이미지 저장소 위치(URL)를 나타내며, 이미지 push 시에 자동적으로 해당 도메인을 검색하여 push하게 됩니다.

```
# docker push posco.po/hub/imagename:tag
```

예시) # docker push posco.po/hub/test-standard:py36

```
# docker push posco/hub/test-standard:py36
```

```
[root@poscoai01 python36]# docker push posco.po/hub/test-standard
The push refers to repository [posco.po/hub/test-standard]
c691d8661aa0: Pushed
392dabb4d08e: Pushed
8bf5c864b187: Pushed
73a8aa314a49: Pushed
57c9a40b002c: Pushed
b0a536cba1d3: Pushed
2737b6cc390b: Pushed
ae8fe8fc5775: Pushed
f73d557c1c45: Pushed
8d2138c5b050: Pushed
d2888f0b05aa: Pushed
5be5e72e510a: Pushed
1ed4cb6297d1: Pushed
7fb28e80062d: Pushed
c3a5c094cf41: Pushed
174f56854903: Pushed
py36: digest: sha256:3845dc3a3a98cf7b114786997cbe0e2a551c75c750bb8eb8affac2ed7dae858c size: 3868
```

2.1.2 R Image build (신규 이미지 생성)

기존 제작된 AI-model 환경구성 폴더에 접속 후, 유사한 과제환경을 확인하고 아래와 같이 새로운 과제환경 이름으로 복사합니다.

```
# cd /build/standard/analysis
# mkdir /build/과제명
```

```
# cp -r /build/standard/ /build/과제명/
# cd /build/과제명/analysis/
```

※ 기존 제작 AI-model 과제환경 탑입(정보)

디렉토리	내용
Python36	Python3.6기반 패키지 설치 파일
Python36-gpu	Python3.6기반 패키지 + tensorflow 패키지설치
Python36-gpu-nanomsg	nanomsg 모듈 (EGL용)
R-Studio	R 3.6.0 + R 패키지

2.1.2.1 Dockerfile 수정

- 디렉토리 내 파일 구조

파일명	내용
Rprofile.site	R의 Repository Site설정 파일 (CRAN URL)
disable_auth_rserver.conf	익명 로그인 방지 설정
pam-helper.sh	Rstudio 사용자 인증 관련 스크립트
userconfig.service	사용자계정 생성하는 스크립트 실행 시스템 서비스
userconfig.sh	사용자 계정 생성 스크립트

복사한 과제환경 폴더에 접속하여 Dockerfile을 수정합니다.

```
# vi Dockerfile
```

```
FROM centos:centos7

# Install R & Rstudio Configuration
ARG R_VERSION=3.6.0
ARG RSTUDIO_VERSION=1.3.1073
ARG PANDOC_TEMPLATES_VERSION=2.10
ARG RUSER=rstudio

ENV R_VERSION=${R_VERSION:-3.6.0} #
```

```

RSTUDIO_VERSION=${RSTUDIO_VERSION:-1.3.1073} &&
R_USER=rstudio
ENV PANDOC_TEMPLATES_VERSION=${PANDOC_TEMPLATES_VERSION:-2.10}
LC_ALL=en_US.UTF-8 &&
LANG=en_US.UTF-8

ENV TZ=Asia/Seoul

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && &&
echo $TZ > /etc/timezone

USER root

# Install OS dependency
RUN yum install -y &&
    bash-completion file bzip2 wget curl sudo net-tools make gcc git perl initscripts zip &&
    unzip pandoc zlib-devel ca-certificates && &&
    yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm && &&
    yum install -y R && &&
    R -e "install.packages('zoo', repos='http://cran.rstudio.com/')" && &&
    ## 기본 R 패키지 이외 추가 설치해야 될 패키지 설치
    yum install -y https://download2.rstudio.org/server/centos6/x86_64/rstudio-server-rhel-
${RSTUDIO_VERSION}-x86_64.rpm && &&
    yum update -y && clean all && &&
    rm -fr /tmp/* /root/.cache /var/cache/yum/*

COPY userconfig.sh /etc/init.d/userconfig
COPY userconfig.service /usr/lib/systemd/system/userconfig.service

# Configure RStudio-Server Environment
RUN ln -s /usr/lib/rstudio-server/bin/pandoc/pandoc /usr/local/bin && &&
ln -s /usr/lib/rstudio-server/bin/pandoc/pandoc-citeproc /usr/local/bin && &&
# git clone --recursive --branch ${PANDOC_TEMPLATES_VERSION} https://github.com/jgm/pandoc-templates
&& &&
mkdir -p /opt/pandoc/templates && &&
cp -r pandoc-templates/* /opt/pandoc/templates && rm -rf pandoc-templates* && &&
mkdir /root/.pandoc && ln -s /opt/pandoc/templates /root/.pandoc/templates && &&
mkdir -p /etc/R && &&
echo "PATH=${PATH}" >> /usr/lib64/R/etc/Renviron && &&
# groupadd -g 1000 rstudio && &&
# useradd -m -s /bin/bash -u 1000 -g 1000 $R_USER && &&
# echo "$R_USER:$R_USER" | chpasswd && &&
# usermod -G wheel $R_USER && &&
echo 'rsession-which-r=/usr/bin/R' >> /etc/rstudio/rserver.conf && &&
echo 'lock-type=advisory' >> /etc/rstudio/file-locks && &&

```

```

systemctl enable userconfig.service && \
rm -fr /tmp/* /root/.cache /var/cache/yum/*

COPY disable_auth_rserver.conf /etc/rstudio/disable_auth_rserver.conf
COPY pam-helper.sh /usr/lib/rstudio-server/bin/pam-helper
COPY Rprofile.site /usr/lib64/R/etc/Rprofile.site

EXPOSE 8787

CMD ["init"]

```

2.1.2.2 Image build 및 Image 조회

```
# docker build -t posco.po/hub/imagename:tag .
```

예시) # docker build -t posco.po/hub/test-standard:py36 .

- image build가 정상적으로 완료시 아래와 같이 나타남.

```
Successfully built 7ad1f46cff01
Successfully tagged posco.po/hub/test-standard:py36
```

docker images | grep "imagename" # 빌드된 이미지 확인

```
[root@poscoai01 python36]# docker images |grep test-standard
posco.po/hub/test-standard                                     py36
7ad1f46cff01          14 minutes ago      3.34GB
```

2.1.2.3 이미지 업로드

이미지를 해당 서버에서 빌드한 이후에는, 아래와 같이 공식 이미지 저장소로 업로드를 해야 합니다.

※ 참고로 이미지 이름 앞부분 posco.po는 이미지 저장소 위치(URL)를 나타내며, 이미지 push 시에 자동적으로 해당 도메인을 검색하여 push하게 됩니다.

```
# docker push posco.po/hub/imagename:tag
```

예시) # docker push posco.po/hub/test-standard:py36

3—~~NAS 내 데이터 저장을 위한 디렉토리 구성(제외)~~

※ 차트 실행 시 자동으로 관련 폴더가 생성되도록 수정하여, NAS 디렉토리 구성은 필요하지 않으며, 이 단계는 skip하시기 바랍니다.



4 [Chart] 환경 구성

Chart는 AI 개발환경을 서비스로 실행할 때 간편하게 실행할 수 있도록 하는 기능입니다. 실제 개발환경을 서비스로 제공하기 위해서는 여러 개의 컨테이너와 부가적인 기능들(오브젝트)이 필요한데, 이런 다수의 컨테이너와 오브젝트들을 한번에 실행할 수 있도록 만들어 실행할 수 있습니다.

※ Namespace Name 및 Chart Name 생성 기준

- 알파벳 소문자를 첫글자로 생성해야 하며,
- 알파벳 소문자, 숫자, -, 소수점(.)의 조합으로 구성 가능
ex) egl2-zn, kr1.abce, sts1.abcd-coater

- 작업 서버 : Control 서버 (172.16.185.23)

4.1 Chart 복사

Control 서버의 /control/install/appcharts/ 폴더(기존 제작된 Chart 구성 폴더)에 접속 후, 표준 프로젝트나 구성하려는 과제와 유사한 과제 환경을 확인하고 아래와 같이 새로운 과제환경 이름으로 복사합니다.

※ 표준 과제환경 샘플 차트 : /control/install/appcharts/project

```
# pwd  
/control/install/appcharts  
# ls -l  
...  
project  
...  
  
# cp -a project ai-test  
# cd ai-test
```

4.2 Chart 설정

/control/install/appcharts 아래 새로 복사한 차트 디렉토리로 이동하여, 새로운 과제환경에 맞게 차트를 수정합니다.

4.2.1 Chart.yaml 설정

Chart.yaml은 차트의 기본 정보와 같으며, 차트의 속성이나 이름 등이 중복되지 않도록 변경해야 합니다. 환경에 맞게 적절하게 수정하여야 하며, 중복되지 않도록 name, description은 필수적으로 수정하셔야 합니다.

- name , description 내용 수정

```
apiVersion: v2
name: ai-test
description: Test Production Environment. A Helm chart for AI/DL Edge Service
  Provides task(tcp,edit), database(mysql), python, etc
icon: http://kubeapps-logo.kubeapps.svc/icon/d1.png
version: 0.1.0
appVersion: 1.0
maintainers:
- name: ickyu.choi
  email: syper@poscoict.com
```

4.2.2 value.yaml 설정

Chart로 과제환경을 실행할 때, 과제환경의 설정 정보를 등록하는 부분입니다. 실제 과제가 실행될 때 사용되는 컨테이너 이미지명, 저장경로 디렉토리, IP 등 컨테이너 기동을 위해 필요한 정보들을 과제환경에 맞게 수정해 주어야만 합니다.

vi values.yaml (※ 붉은색은 과제 성격에 맞게 수정 필요)

```
# Default values for project-standard.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

project: example # 과제명으로 수정

# configuration for tcp/edit task
task:
  enabled: false # task 컨테이너 기동 여부 설정 (true: 기동, false: 기동X)

  tcpTask:
    image: "posco.po/hub/tcp-task:std-1.0" # tcp-task 컨테이너 이미지 명

    port: 9091 # tcp task 컨테이너의 Listen 포트

  # for resources configuration
```

```
resources: {}  
# limits: : # llimits는 최대 제한 자원 설정  
#   cpu: 1  
#   memory: 1Gi  
  
service:  
port: 9091 # tcp task 컨테이너의 외부 오픈 포트 변경 시 수정  
  
# task ingress type is LoadBalancer  
loadBalancerIP: "172.16.185.53" # IP는 172.18.185.53을 이용한다  
  
editTask:  
image: "posco.po/hub/edit-task:std-1.1" # edit-task 컨테이너 이미지 명  
  
# for resources configuration  
resources: {}  
# limits: : # llimits는 최대 제한 자원 설정  
#   cpu: 1  
#   memory: 1Gi  
  
## configuration for task volume  
## once the value is set and container is up and running, it should not be modified  
persistence:  
type: nfs  
server: 172.16.185.34  
  
size: 10Gi # task 영역 저장공간 용량 제한 (NAS)  
  
# configuration for mysql database  
# default mysql port is 3306  
b:  
image: "posco.po/hub/mysql:5.7.30" # mysql db 컨테이너 이미지 명  
  
# for resources configuration  
resources: {}  
# limits:  
#   cpu: 1  
#   memory: 1Gi  
  
config: # mysql dataset config  
rootUser: root  
rootPassword: "Passw0rd" # mysql 계정 패스워드  
dbName: KR_AI # mysql 데이터베이스 명
```

```

## configuration for database volume
## once the value is set and container is up and running, it should not be modified
## default mount point in container is /var/lib/mysql
persistence:
  type: nfs
  server: 172.16.185.34

  size: 50Gi                                # db 영역 저장공간 용량 제한 (NAS)

workbench:
  ingress:
    hostname: workbench-kr1aisys.paics.posco.co.kr  # mysql workbench 외부 접속 도메인 설정

## configuration for analysis program
analysis:
  enabled: true                             # analysis 전체컨테이너 기동 여부 설정 (true: 기동, false: 기동X)

## configuration for model program
## unless the model program script is specified in the container image,
## it must be specified with command option below
model:
  enabled: false                           # 개발된 머신러닝/딥러닝 모델 프로그램 설정
  image: "posco.po/hub/anls-python:1.0-nanomsg"  # 모델 프로그램 컨테이너 이미지 명

## script to start model program
# command: [ "python", "taskAIManager.py" ]

resources: {}
# limits:
#   cpu: 1
#   memory: 1Gi
#   nvidia.com/gpu: 1                         # GPU 할당 시 입력. 정수로 입력

## configuration for jupyter notebook
## default jupyter notebook port is 8888
jupyter:
  enabled: true                            # jupyter 컨테이너 기동 여부 설정 (true: 기동, false: 기동X)
  image: "posco.po/hub/anls-python:1.0"      # jupyter notebook 컨테이너 이미지 명

resources: {}
# limits:
#   cpu: 1
#   memory: 1Gi
#   nvidia.com/gpu: 1                         # GPU 할당 시 입력. 정수로 입력

```

```

## jupyter notebook password
password: "Pcinfra1!" # jupyter-notebook 로그인 패스워드

ingress:
hostname: jupyter-kr1aisys.paics.posco.co.kr # jupyter notebook 외부 접속 도메인 설정

## configuration for rstudio
## default rstudio port is 8787
rstudio:
enabled: false # rstudio 컨테이너 기동 여부 설정 (true: 기동, false: 기동X)
image: "posco.po/hub/anls-r:1.0" # jupyter notebook 컨테이너 이미지 명

resources: {}
# limits:
#   cpu: 1
#   memory: 1Gi

# configuration for rstudio user
user: |
  posco:Pcinfra1!
  user1:Passw0rd

ingress:
hostname: rstudio-kr1aisys.paics.posco.co.kr # rstudio 외부 접속 도메인 설정

## configuration for analysis volume
## once the value is set and container is up and running, it should not be modified
persistence:
type: nfs
server: 172.16.185.34

size: 20Gi # analysis 영역 저장공간 용량 제한 (NAS)

## default mount point in container is /home/posco
## if you want to change the mount point in container, please modify the mountPath value below
# mountPath: /home/posco

nodeSelector: {}
# nodeSelector:
#   type: nvidiagpu # 컨테이너가 기동될 node 지정 설정

```

4.3 Chart 등록

Chart의 수정이 완료되면, 수정한 Chart를 Chart저장소에 등록해야 합니다. 차트 디렉토리(예시: ai-test)에서 차트를 push하여 업로드합니다. 포항 AI 렌탈 클라우드의 공식 차트 저장소 이름은 posco입니다.

※ 차트 저장소 이름 : posco

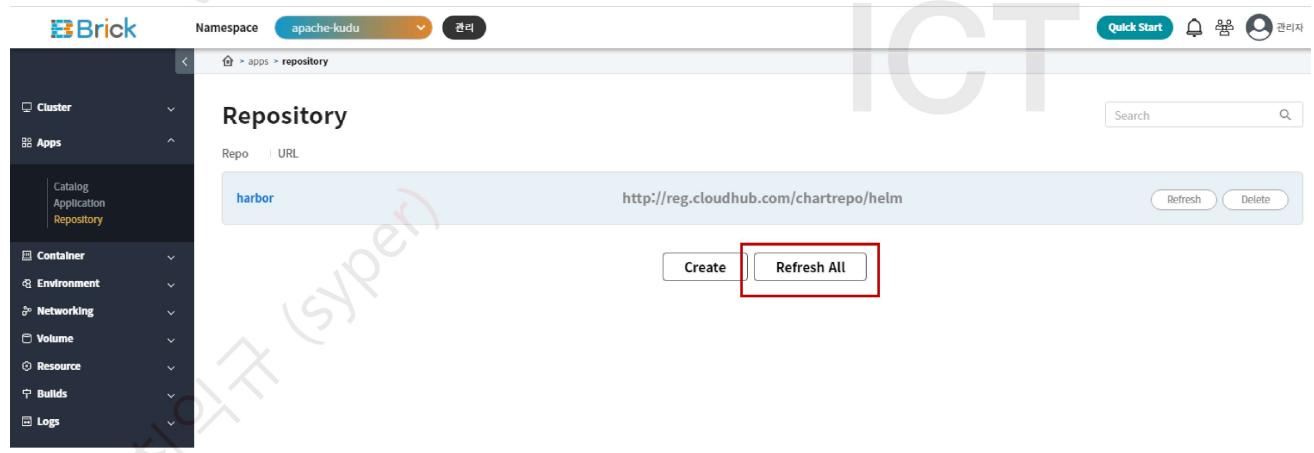
```
# helm push . posco          # 명령 실행 디렉토리가 해당 차트 디렉토리 안 일 경우
```

```
# helm push ./ai-test posco    # 명령 실행 디렉토리가 해당 차트 밖인 경우
```

```
[root@plpaismss01 appcharts]# helm push ./ai-test/ posco
Pushing ai-test-0.1.0.tgz to posco...
Done.
```

4.4 Chart 갱신 및 업로드 확인

업로드한 차트는 약 10분 이후 자동으로 갱신됩니다. 만일 바로 갱신하여 확인하고 싶은 경우 관리화면 – Apps – Repository 화면에서 [Refresh All] 메뉴를 한 번 클릭하시기 바랍니다.



이후 관리화면 – Apps – Catalog 화면에서 아래와 같이 신규 등록된 차트(ai-test)가 보이면 정상입니다.

The screenshot shows the CloudHub management interface. On the left, a dark sidebar menu includes 'Cluster', 'Apps' (which is expanded to show 'Catalog'), 'Container', 'Environment', 'Networking', 'Volume', 'Resource', and 'Logs'. The main area is titled 'Catalog' and displays two chart entries:

- ai-test**: Test Production Environment. A Helm chart for AI/DL Edge Service. Provides task(tcp,edit), database(mysql), python, etc. Version 1 is listed under 'cloudhub'.
- egl2coater**: A Helm chart for EGL Service. Provides task(tcp,edit), database(mysql), python, etc. Version 1 is listed under 'cloudhub'.

4.5 Chart 삭제

업로드한 Chart를 카탈로그에서 아예 삭제하고자 하는 경우, 관리화면 – Apps – Catalog에서 먼저 삭제하고자 하는 차트를 선택하여 진입한 후, 아래에 [Delete]를 클릭하여 차트를 삭제할 수 있습니다.

This screenshot shows the Catalog page with the 'egl2coater' chart selected. The chart details are displayed, including its name ('harbor/egl2-zn'), version ('1 - harbor'), and a brief description ('A Helm chart for EGL Service. Provides task(tcp,edit), database(mysql), python, etc'). Below the chart details, there are dropdown menus for 'Chart Versions' (set to '1 - 2020-10-12') and 'App Version' (set to '1'). A 'Deploy' button is located on the right. At the bottom of the chart card, a 'Delete' button is highlighted with a red dashed box.

4.6 Chart 기동

- ① 신규 등록된 Chart 클릭 (ai-test) → Deploy 클릭

This screenshot shows the Catalog page with the 'ai-test' chart selected. The chart details are displayed, including its name ('clouduhub/ai-test'), version ('1 - clouduhub'), and a brief description ('Test Production Environment. A Helm chart for AI/DL Edge Service. Provides task(tcp,edit), database(mysql), python, etc'). Below the chart details, there are dropdown menus for 'Chart Versions' (set to '0.1.0 - 2020-11-10') and 'App Version' (set to '1'). A 'Deploy' button is located on the right. At the bottom of the chart card, a 'Delete' button is highlighted with a red dashed box.

- ② Name 및 Namespace 선택 후 "Submit" 클릭

Deploy

cloudhub/ai-test

Name	ai-test
Namespace	sts1
Version	0.1.0

```

1 analysis:
2   enabled: true
3 jupyter:
4   enabled: true
5   image:
6     repository: posco.po/hub/anls-python
7     tag: "1.0"
8   ingress:
9     enabled: true
10    hostname: jupyter-test.paics.posco.co.kr
11    password: Pcinfra1!
12    port: 8888
13   service:
14     port: 8888
15     type: ClusterIP
16   model:
17     enabled: false
18     image:
19       repository: posco.po/hub/anls-python
20       tag: "1.0"

```

Note : Only comments from the original chart values will be preserved.

Restore Chart Defaults **Submit**

- ③ 정상 Deploy된 결과 확인

Application Detail

ai-test

Test Production Environment. A Helm chart for AI/DL Edge Service. Provides task(tcp,edit), database(mysql), python, etc

App Version: 1
Chart Version: 0.1.0
 Up to date

Unknown

Access URLs

RESOURCE	TYPE	URL
No data available		

Notes

```

1. You get the Jupyter Application URL by running these commands:
http://jupyter-test.paics.posco.co.kr

2. You get the Rstudio Application URL by running these commands:
http://rstudio-test.paics.posco.co.kr

3. You get the MySQL Application URL by running these commands:

```

- ④ 생성된 Namespace 선택 후 Container → Pod 클릭

⇒ 기동된 Pod 상태 확인

Name	NodeName	OwnerRef	CreatedAt
ai-test-ai-test-analysis-58fdc66948-7dpgs	plpaisws02	Deployment / ai-test-ai-test-analysis	2020-11-10 16:13:18
ai-test-ai-test-db-fc86dd64-vgbz2	plpaisws03	Deployment / ai-test-ai-test-db	2020-11-10 16:13:18
ai-test-ai-test-task-77ccfdff59-ds8bz	plpaisws02	Deployment / ai-test-ai-test-task	2020-11-10 16:13:18

⇒ 컨테이너 상태 확인

Containers

Name edit Image posco.po/hub/edit-task/std-1.0 State terminated Ready false StartedAt RestartCount 23	Name tcp Image posco.po/hub/tcp-task/std-1.0 State running Ready true StartedAt 2020-11-09T18:53:09.000+09:00 RestartCount 0	Name workbench Image posco.po/hub/linuxserver/mysql-workbench:version-8.0.22 State running Ready true StartedAt 2020-11-09T18:53:10.000+09:00 RestartCount 0
--	---	---

⑤ 컨테이너 접속하여 source file 및 Directory 상태 등 컨테이너 점검

ai-test-ai-test-task-77ccfdff59-ds8bz Running

tcp
edit
workbench

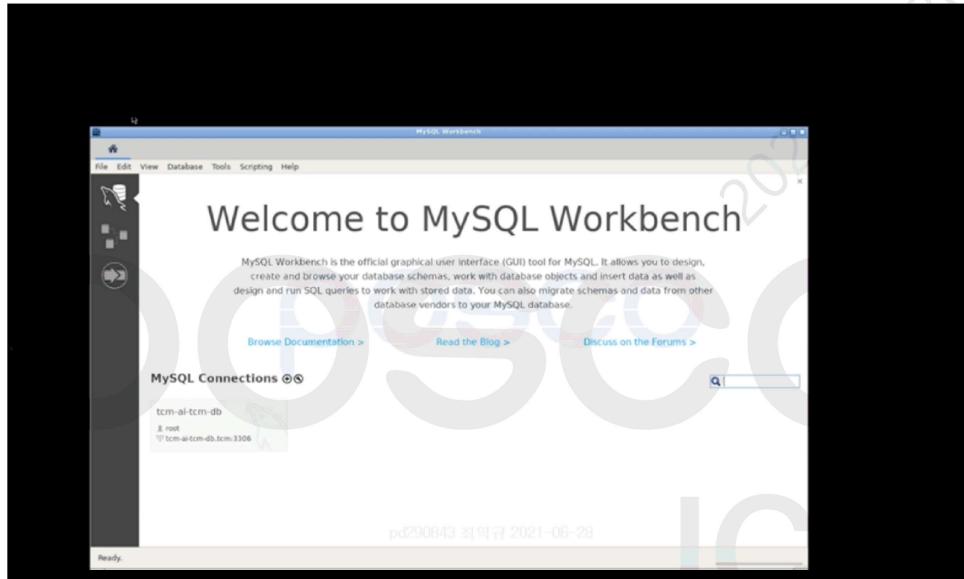
5 사용 방법

5.1 MySQL Workbench

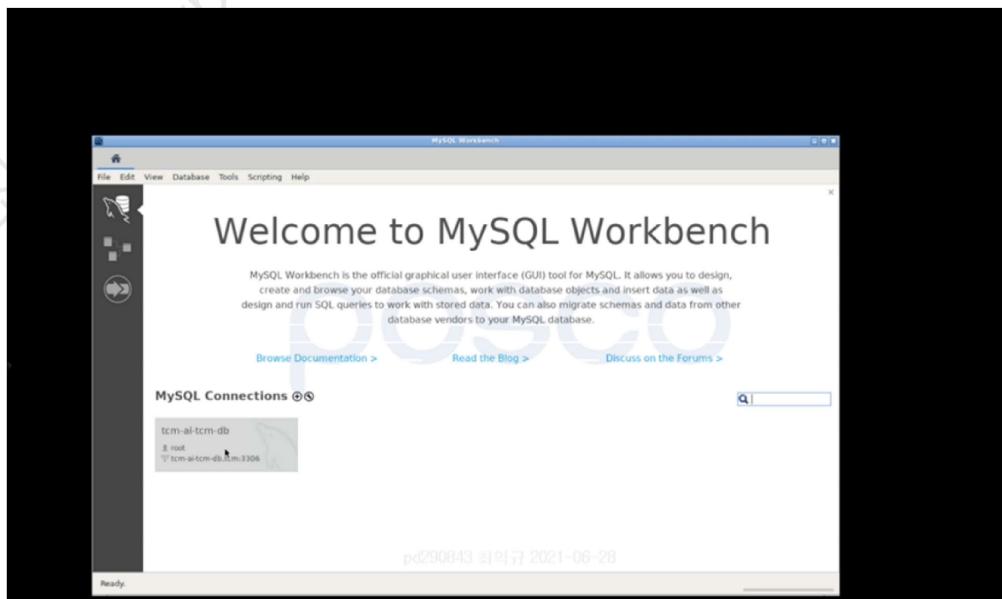
5.1.1 MySQL Workbench 접속

MySQL Workbench는 Chart를 구성할 때 values.yaml에 지정한 workbench ingress host도메인으로 브라우저를 통해 접속할 수 있습니다.

예를 들어, Chart의 values.yaml 파일에 db.workbench.ingress.host에 workbench-example.paics.posco.co.kr 도메인으로 지정했다면, <http://workbench-example.paics.posco.co.kr> 로 브라우저를 통해 접속하시면 아래와 같은 Workbench 화면에 접속하실 수 있습니다.



과제의 MySQL DB와 자동으로 연결할 수 있도록 구성정보가 등록되어 있을 것입니다. 선택하여 DB 암호를 입력하여 DB 데이터에 접근할 수 있습니다.



간혹 브라우저의 화면이 검은색으로 표시되는 경우에는, 아래와 같이 화면에 마우스 오른쪽 버튼을 클릭하여 “exit”를 클릭한 후 브라우저 새로고침을 해보시기 바랍니다.



그래도 제대로 표시되지 않는다면, 화면에 마우스 오른쪽 버튼을 클릭하여 “Restart”를 클릭하여 컨테이너를 재기동하시기 바랍니다.



6 표준 TASK 이미지 제작 참고

- 표준 Task 이미지는 기존 제작된 이미지를 사용해도 되며, 버전이 변경된 경우 업그레이드 필요합니다.
- Image 제작을 위한 Dockerfile 위치

. 원본 : 172.16.185.25:/ build/standard (*DB Dockerfile는 tcp-task에 위치)

```
[root@plpaisdr01 standard]# pwd
/build/standard
[root@plpaisdr01 standard]# ls -al
total 0
drwxr-xr-x 7 root root 86 Nov 17 13:29 .
drwxr-xr-x 5 root root 53 Nov 10 14:35 ..
drwxr-xr-x 6 root root 92 Nov 23 14:15 analysis
drwxr-xr-x 5 root root 56 Nov 17 13:27 backup
drwxr-xr-x 3 root root 36 Nov 17 13:25 edit-task
drwxr-xr-x 2 root root 44 Nov 17 13:29 task-base
drwxr-xr-x 3 root root 36 Nov 10 15:06 tcp-task
```

- 1) 기 제작된 표준 TASK Image 정보

구분	Image Name	Image Tag
Tcp-Task	posco.po/hub/tcp-task	std-1.0
Edit-Task	posco.po/hub/edit-task	std-1.0
DB	posco.po/hub/mysql	5.7.30

※ Ex) Image 조회 방법

- Registry서버 접속 : 172.16.185.25 (plpaisdr01)
- # docker images | grep task

```
[root@plpaisdr01 build]# docker images |grep task
posco.po/hub/eg[2-a]task          1.0           9e002925b0a8    11 days ago   3.06GB
posco.po/hub/eg[2-a]task          <none>        5b579da15278   6 weeks ago   3.06GB
posco.po/hub/edit-task           1.0           9c9beba6f289   2 months ago  406MB
posco.po/hub/tcp-task            1.0           87153df5fb91   2 months ago  406MB
task-base                         centos7      106a39a1e845   2 months ago  406MB
```

- 2) 표준 Task Image 업그레이드

1. Image build 작업은 인터넷이 되는 서버에서 패키지 설치 필요 : 172.16.185.32 서버에 접속
2. # cd /build/standard/
3. # mkdir /build/과제명
4. # cp -r /build/standard/ /build/과제명/
5. # cd /build/과제명/
6. Tcp-task, edit-task, DB 중 변경이 필요한 폴더 접속하여 Dockerfile 수정
7. # vi Dockerfile

```
FROM posco.po/hub/task-base:centos7
```

```

RUN yum install -y tcpdump strace lsof &&
  && yum clean all &&
  && rm -rf /var/cache/yum/*

# Configure environment
ENV LC_ALL=en_US.UTF8 &&
  LANG=en_US.UTF8 &&
  LANGUAGE=en_US.UTF8 &&
  TZ=Asia/Seoul

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && &&
  echo $TZ > /etc/timezone

ADD task /task

RUN cd /task/src && make

ENV PATH=$PATH:/task
WORKDIR /task

#ENTRYPOINT [ "/edit_task" ]
CMD [ "/task/edit_task" ]

```

8. Dockerfile 수정후 image build 진행

docker build -t posco.po/hub/edit-task:std-1.1 .

```

Sending build context to Docker daemon 107kB
Step 1/9 : FROM posco.po/hub/task-base:centos7
--> 106a39a1e845
Step 2/9 : RUN yum install -y tcpdump strace lsof && yum clean all && rm -rf
/var/cache/yum/*
--> Using cache
--> 243f2733f892
Step 3/9 : ENV LC_ALL=en_US.UTF8 LANG=en_US.UTF8 LANGUAGE=en_US.UTF8
TZ=Asia/Seoul
--> Using cache
--> 3c09ebbd0a94
Step 4/9 : RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ >
/etc/timezone
--> Using cache

```

```
---> 70a468365b84
Step 5/9 : ADD task /task
---> Using cache
---> ee8a66e88696
Step 6/9 : RUN cd /task/src && make
---> Using cache
---> 2961101ea0f1
Step 7/9 : ENV PATH=$PATH:/task
---> Using cache
---> 7a61f34b5d50
Step 8/9 : WORKDIR /task
---> Using cache
---> 38125a4dacae
Step 9/9 : CMD [ "/task/edit_task" ]
---> Using cache
---> 3df88c04fec6
Successfully built 3df88c04fec6
Successfully tagged posco.po/hub/edit-task:std-1.1 → Image build 정상 완료
```

9. posco.po/hub/edit-task:std-1.1 이미지 생성 확인

```
[root@poscoai01 edit-task-1110]# docker images |grep edit-task
posco.po/hub/edit-task
    3df88c04fec6      3 weeks ago      433MB
```

std-1.1

10. 나머지 tcp-edit와 DB도 동일하게 이미지 업데이트 진행