

2022-05-24 09:55:02



AI 연구개발용 PAAS 솔루션 사용 가이드

내용

1	AI과제 환경 요건 분석.....	4
1.1	과제환경 검토 사항	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	5
1.2	Anaconda – Python 버전 매트릭스.....	5
1.3	Tensorflow – CUDA 버전 매트릭스	6
1.4	Pytorch - CUDA 버전 매트릭스 (참고 사이트).....	7
1.5	Image Build 시 주요 에러 (업데이트 예정)	7
1.5.1	<i>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.</i>	7
1.5.2	<i>ImportError: libGL.so.1: cannot open shared object file: No such file or directory</i>	7
1.5.3	<i>ImportError: libcublas.so.10.0: cannot open shared object file: No such file or directory</i>	7
1.5.4	<i>ImportError: libcuda.so.1: cannot open shared object file: No such file or directory</i>	7
1.6	AI과제 환경 접수 샘플 양식 (참고)	8
2	[AI 개발 환경] 구성	9
2.1	AI 개발환경 이미지 제작	9
2.1.1	신규 이미지 생성	9
3	[Chart] 환경 구성	17
3.1	Chart 복사	17
3.2	Chart 수정	18

3.2.1 Chart.yaml 수정	18
3.2.2 value.yaml 수정	18
3.3Chart 등록	21
3.4Chart 갱신 및 업로드 확인	22
3.5Chart 삭제	23
3.6Chart 기동	23



최의규 (Syper)

1 AI과제 환경 요건 분석

1.1 과제환경 검토 사항

1.1.1 과제환경 자원 제한 설정

컨테이너 기반의 과제 환경 제공 시 CPU, Memory, GPU 등의 자원을 제한할 수는 있지만, 컨테이너 내에서 제한 값 이상으로 자원을 요구하게되는 경우, 시스템에서는 해당 컨테이너의 자원을 제한하는 데 오버헤드가 발생하게 됩니다. 이러한 부분을 고려하여 특정 과제 환경이 자원을 과도하게 사용하거나 제한이 꼭 필요한 경우에 사용하는 것이 적절하며, 자원을 제한해야 하는 경우에는 반드시 해당 컨테이너가 사용하는 자원을 확인한 후, 해당 사용량보다는 여유를 두고 제한을 하는 것이 좋습니다. 참고로 볼륨 관련하여 용량을 설정하더라도, NAS 스토리지의 경우에는 실제 스토리지의 용량을 제한하는 컨트롤러가 없으므로 제한이 적용되지 않습니다. 스토리지 용량을 별로 제한하려면 동적 볼륨 컨트롤러인 Provisioner 등을 활용해야만 가능합니다.

1.1.2 과제환경 자원 할당 기준

자원 할당 관련하여 명확히 제시되는 기준은 존재하지 않으며, 실제 컨테이너 내에서 동작하는 어플리케이션들의 사용량에 따라 설정해야 합니다.

영상 이미지 과제의 경우 사용량이 높아서 보통 Memory를 약 40GB 사용하는 경우도 있으며, 일반적인 text 과제인 경우에는 cpu 1Core 이하, Memory도 5G 이하를 사용하는 경우도 있습니다. 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 이상의 용량을 감수해야 합니다. 이 경우에는 이미지가 사전에 다운로드되지 않은 서버에서 이미지를 Pull할 경우, 상당히 많은 시간이 소요된다는 점을 고려해야 하며, 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, 1초 이내의 아주 빠른 실시간 응답이 필요한 경우, 과제 환경 사이즈가 과도하게 큰 경우에는 적합하지 않으며, 이 경우 상세한 테스트 및 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
2021.05	2021-05-13		python 3.8.5 -> 3.9.4
2021-11	2021-11-17		python 3.9.4 -> 3.9.7

※ 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

참고사이트 : <https://www.tensorflow.org/install/source#gpu>

버전	Python 버전	컴파일러	빌드 도구	cuDNN	CUDA	검증 확인
tensorflow-2.7.0	3.7~3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2	
tensorflow-2.6.0	3.6~3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2	
tensorflow-2.5.0	3.6~3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2	
tensorflow-2.4.0	3.6~3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0	
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	10.0	
tensorflow_gpu-1.15.0	2.7, 3.3~3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0	
tensorflow_gpu-1.14.0	2.7, 3.3~3.7	GCC 4.8	Bazel 0.24.1	7.4	10.0	
tensorflow_gpu-1.13.1	2.7, 3.3~3.7	GCC 4.8	Bazel 0.19.2	7.4	10.0	
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 요청/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 Memory : 32GB GPU : 1EA	과제에서 예상되는 CPU, Memory, GPU 필요 수량
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 개발 환경] 구성

2.1 AI 개발환경 이미지 제작

기존에 제작된 이미지가 아닌 새로운 이미지를 제작하는 경우에 사용합니다. 예를 들어 기존 이미지에 추가적인 작업이 필요한 경우나, Python이나 Tensorflow 등 S/W 버전 상의 문제로 기존에 제공된 개발 환경이 아닌 새로운 이미지를 제작해야 하는 경우 사용합니다.. (기존 Image를 사용할 경우에는 skip)

기존 템플릿 Dockerfile을 복사하여 수정하면 쉽게 업데이트 가능합니다.

※ 컨테이너 이미지 빌드는 인터넷이 되는 서버에서 진행해야 합니다. (이미지 빌드 서버 참고)

2.1.1 신규 이미지 생성

기존 제작된 개발환경 템플릿 폴더에서 유사한 과제 환경을 확인하고 복사합니다.

```
# cd /build/standard
# ls -la
total 4
drwxr-xr-x. 8 root root 4096 Mar 22 17:43 .
drwxr-xr-x. 7 root root 100 Mar 30 16:28 ..
drwxr-xr-x. 2 root root 149 Mar 22 11:38 1.tensorflow2.6-py38-cuda11.2-cudnn8.1
drwxr-xr-x. 2 root root 149 Mar 23 09:13 2.tensorflow1.15-py36-cuda10-cudnn7.4
drwxr-xr-x. 4 root root 250 Mar 22 11:43 3.tensorflow2.6-py38-cuda11.2-cudnn8.1-vnc
drwxr-xr-x. 4 root root 228 Mar 22 14:22 3.tensorflow2.6-py38-cuda11.2-cudnn8.1-xrdp
drwxr-xr-x. 4 root root 223 Mar 23 09:13 4.tensorflow1.15-py36-cuda10-cudnn7.4-vnc
drwxr-xr-x. 4 root root 228 Mar 23 09:14 4.tensorflow1.15-py36-cuda10-cudnn7.4-xrdp

# mkdir /build/custom/[new-name]
# cp -r 1.tensorflow2.6-py38-cuda11.2-cudnn8.1 /build/custom/[new-name]/
# cd /build/custom/[new-name]
```

복사한 과제환경 폴더에 접속하여 Dockerfile을 수정합니다.

2.1.1.1 Dockerfile 수정

※ 1.tensorflow2.6-py38-cuda11.2-cudnn8.1 기준 설명

- 디렉토리 내 파일 구조

파일명	내용
Dockerfile	Docker Image build 설정 파일
fix-permissions	빌드 시 디렉토리 권한/소유권 설정 스크립트

jupyter_notebook_config.py	Jupyter notebook 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
xfce4-config	VNC 원도우 화면 설정 파일
test	Tensorboard용 테스트 파일

- Dockerfile 작성 예제

※ 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 conda로 설치하는 경우에는 conda search [패키지명]으로 python 버전에 맞는 패키지 버전 검색
ex) [conda search tensorflow](#)

vi Dockerfile

```
# Copyright (c) Cloudhub.

# Ubuntu 18.04 (bionic)
# https://hub.docker.com/_/ubuntu/?tab=tags&name=bionic
# Reference : https://hub.docker.com/r/jupyter/base-notebook/dockerfile
# OS/ARCH: linux/amd64

##### Configuration #####
# Base Container Image
FROM nvidia/cuda:11.2.0-cudnn8-devel-ubuntu18.04

# Python Version Configuration(Anaconda, Miniconda)
ARG ANACONDA_VERSION=5.1.0
ARG MINICONDA_VERSION=4.11.0
ARG CONDA_VERSION=4.11.0

# Tensorflow version config - comment below if you don't want to install tensorflow
ARG TENSORFLOW_ENABLE="yes"
ARG TENSORFLOW_VERSION=2.6.0
```

```
# Pytorch version config - comment below if you don't want to install pytorch
ARG PYTORCH_ENABLE="yes"
ARG TORCHVISION=0.11.3

#####
# Configuration #####
#####

LABEL maintainer="clouithub <support@brickcloud.co.kr>"
ARG NB_USER="poscoict"
ARG NB_UID="1000"
ARG NB_GID="100"

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

USER root

# Install all OS dependencies for notebook server that starts but lacks all
# features (e.g., download as all possible file formats)
ENV DEBIAN_FRONTEND noninteractive

# For NVIDIA Repository Certificates Issues
RUN echo "deb [by-hash=no] http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 /"
> /etc/apt/sources.list.d/cuda.list && \
  echo "deb [by-hash=no] http://developer.download.nvidia.com/compute/machine-
learning/repos/ubuntu1804/x86_64 /" > /etc/apt/sources.list.d/nvidia-ml.list

RUN apt-get update \
&& apt-get install -yq --no-install-recommends \
  apt-utils \
  build-essential \
  wget \
  bzip2 \
  ca-certificates \
  vim \
  tzdata \
  supervisor \
  openssh-server \
  gnupg \
  git \
  sudo \
  locales \
  fonts-liberation \
  pandoc
```

```
run-one ¶
&& apt-get clean && rm -rf /var/lib/apt/lists/* ¶
&& echo "en_US.UTF-8 UTF-8" > /etc/locale.gen && locale-gen

# Configure environment
ENV CONDA_DIR=/opt/conda ¶
SHELL=/bin/bash ¶
NB_USER=$NB_USER ¶
NB_UID=$NB_UID ¶
NB_GID=$NB_GID ¶
LC_ALL=en_US.UTF-8 ¶
LANG=en_US.UTF-8 ¶
LANGUAGE=en_US.UTF-8
ENV PATH=$CONDA_DIR/bin:$PATH ¶
HOME=/home/$NB_USER ¶
TZ=Asia/Seoul

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime

# Copy a script that we will use to correct permissions after running certain commands
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 with name jovyan user with UID=1000 and in the 'users' group
# and make sure these dirs are writable by the `users` group.
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 && ¶
    usermod -aG sudo $NB_USER && ¶
    echo "%sudo ALL=NOPASSWD: ALL" >> /etc/sudoers && ¶
    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
ARG PYTHON_VERSION=default
```

```

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

ENV MINICONDA_VERSION=4.11.0 \
    CONDA_VERSION=4.11.0

# Install Anaconda
WORKDIR /tmp
RUN wget --quiet https://repo.anaconda.com/miniconda/Miniconda3-py38_${MINICONDA_VERSION}-Linux-x86_64.sh && \
    /bin/bash Miniconda3-py38_${MINICONDA_VERSION}-Linux-x86_64.sh -f -b -p $CONDA_DIR && \
    rm -f Miniconda3-py38_${MINICONDA_VERSION}-Linux-x86_64.sh && \
    echo "conda ${CONDA_VERSION}" >> $CONDA_DIR/conda-meta/pinned && \
    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 strict && \
    if [ ! $PYTHON_VERSION = 'default' ]; then conda install --yes python=$PYTHON_VERSION; fi && \
    conda list python | grep '^python' | tr -s ' ' | cut -d ' ' -f 1,2 | sed 's/$.*/' >> $CONDA_DIR/conda-meta/pinned && \
    conda install --quiet --yes conda && \
    conda install --quiet --yes pip && \
    conda update --all --quiet --yes && \
    conda clean --all -f -y && \
    rm -rf /home/$NB_USER/.cache/yarn && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

#RUN sed -i -e 's/self.verify = False/self.verify = True/' /opt/conda/lib/python3.6/site-packages/pip/_vendor/requests/sessions.py

# Install Tini
RUN 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 --all -f -y && \
    fix-permissions $CONDA_DIR && \
    fix-permissions /home/$NB_USER

# Install Jupyter
RUN conda install --quiet --yes \
    'notebook=6.0.3' \
    'jupyterhub=1.1.0' \

```

```
'jupyterlab=2.1.5' && ¶
conda clean --all -f -y && ¶
npm cache clean --force && ¶
jupyter notebook --generate-config && ¶
jupyter labextension install jupyterlab_tensorboard && ¶
rm -rf $CONDA_DIR/share/jupyter/lab/staging && ¶
rm -rf /home/$NB_USER/.cache/yarn && ¶
fix-permissions $CONDA_DIR && ¶
fix-permissions /home/$NB_USER

# Install Tensorflow
RUN 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

# Install Pytorch
RUN if [ "$PYTORCH_ENABLE" = "yes" ]; then ¶
    pip3 install --no-cache-dir torch==1.10.2+cu113 torchvision==0.11.3+cu113 torchaudio==0.10.2+cu113 -f \
https://download.pytorch.org/wheel/cu113/torch_stable.html && ¶
    fix-permissions $CONDA_DIR && ¶
    fix-permissions /home/$NB_USER; ¶
fi

# Install Other packages
RUN conda config --set ssl_verify false && ¶
    conda install --quiet --yes ¶
    pillow ¶
    pandas ¶
    scikit-learn ¶
    matplotlib && ¶
    conda clean --yes --all && ¶
    fix-permissions $CONDA_DIR && ¶
    fix-permissions /home/$NB_USER

EXPOSE 8888 22

# 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/
# Switch back to jovyan to avoid accidental container runs as root
USER $NB_UID
WORKDIR $HOME
```

2.1.1.2 Image build 및 Image 조회

이미지를 빌드 후에 클러스터 환경에서 사용하기 위해서는 이미지 저장소 서버에 이미지를 업로드(PUSH)해야만 하며, 이미지 업로드를 위해서는 이미지명의 앞부분 Repository가 이미지 저장소의 주소에 맞게 변경이 되어 있어야만 가능합니다. 따라서 이미지 빌드 시에 이름을 지정하시거나, 또는 빌드 이후에 명령을 통해 반드시 주소를 변경해 주어야 합니다.

※ 컨테이너 이미지 명 형식 : [REPOSITORY]:[TAG]

이미지 빌드 시 사전에 구성한 이미지 저장소 정보를 REPOSITORY로, 그리고 이미지의 특성에 맞는 정보를 TAG로 입력하여 빌드합니다. registry.poscoict.com/library는 이미지 저장소의 위치(REPOSITORY)를 나타냅니다.

```
# docker build -t registry.poscoict.com/library/tensorflow1.15-py36 .
...
Successfully built b7ab94c44a52
Successfully tagged registry.poscoict.com/library/tensorflow1.15-py36

# docker images | grep tensorflow1.15
registry.poscoict.com/library/tensorflow1.15-py36      cuda10-cudnn7          183faa0234b2  13 days ago
ago 8.8GB
tensorflow1.15-py36-xrdp      cuda10-cudnn7          b7ab94c44a52  7 days ago  10.6GB
tensorflow1.15-py36-novnc     cuda10-cudnn7          b83c95669c13  7 days ago  10.6GB
```

2.1.1.3 이미지 업로드

이미지를 해당 서버에서 빌드한 이후에는, 아래와 같이 공식 이미지 저장소로 업로드를 해야 합니다. 만일 이미지명에 REPOSITORY가 제대로 설정되어 있지 않으면, tag 옵션을 사용해 이름을 변경한 후 업로드하기 바랍니다.

```
# docker tag tensorflow1.15-py36-xrdp: cuda10-cudnn7 registry.poscoict.com/library/tensorflow1.15-py36-xrdp: cuda10-cudnn7
```

```
# docker push registry.poscoict.com/library/tensorflow1.15-py36-xrdp: cuda10-cudnn7
```



최의규 (syper)

2022-05-24 09:55:02

3 [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, -(Hyphens)조합으로 구성 가능
ex) abc1 abc-name, a123-nameegl2-zn, kr1.abce, sts1.abcd-coater

- 작업 서버 정보는 구성현황 정보 참조

3.1 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/standard

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

3.2 Chart 설정

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

3.2.1 Chart.yaml

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

3.2.2 value.yaml

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

# !Important: If you don't need to set a value,
# just use curly braces instead of commenting it out.

global:
  # Access VIP
  accessIp: 203.238.192.132

  # Hostname of Container, It does not mean container name
  hostname: standard          # 컨테이너 호스트명

extraHosts: {}
  # 192.168.0.1: test1.testdomain.com
  # 192.168.0.2: test2.testdomain.com

password: "Passw0rd!"          # 어플리케이션 접속 암호

image: registry.poscoict.com/library/tensorflow2.6-py38-novnc:cuda11.2-cudnn8  # 컨테이너 이미지명
```

```

# image: registry.poscoict.com/library/tensorflow1.15-py36-novnc:cuda10-cudnn7

# !important: To use a mig gpu device, use nvidia.com/mig-3g.40gb or nvidia.com/mig-2g.20gb
# instead of the nvidia.com/gpu option.

resources: {}                                # 컨테이너 자원 제한 설정

  # limits:
  #   cpu: 1
  #   memory: 1Gi
  #   nvidia.com/gpu: 1                         # Non-MIG GPU, 정수 입력
  #   nvidia.com/mig-3g.40gb: 1                  # 40GB MIG GPU
  #   nvidia.com/mig-2g.20gb: 1                  # 20GB MIG GPU

# Choose which services to enable.

# Supported Service is jupyter notebook, vnc(xrdp), tensorboard, sshd
# you also should configure supervisor configuration

service:                                     # 기동 서비스 지정

  notebook:
    enabled: true                            # jupyter notebook 기동 여부 설정
    ingress:
      hostname: standard-notebook.aidevs.poscoict.com # jupyter notebook 외부 접속 도메인

  webvnc:
    enabled: true
    ingress:
      hostname: standard-vnc.aidevs.poscoict.com

  tensorboard:
    enabled: false
    ingress:
      hostname: standard-tensorboard.aidevs.poscoict.com

  sshd:
    enabled: true
    port: 31234

# Volume Configuration
# !important: Changing the value has no effect while the container is running.

persistence:
  data1:
    #!important: specify the type as "nfs" or "none"(software-defined storage).
    # Specify "nfs" to create volume on nas, "none" on ceph storage.
    type: none
    server: 203.238.192.170
    # To use a different path, set the path value below
    # path: /volume1/repo

```

```

size: 1000Gi

# mountPath value means mountpoint in container
# The default value is the project name.
# Specify a value to set a different path
# mountPath: /standard

# Setting subPath will create the directory on the volume path
# eg, value "standard" creates the /volume1/repo/standard directory.
# Without a value, no directory is created.
subPath: standard

data2: {}
  # type: nfs
  # server: 203.238.192.170
  # size: 1000Gi
  # mountPath: /standard2
  # subPath: standard2

nodeSelector: {}
  # type: nvidiagpu
  # 컨테이너를 기동할 node의 label 지정

```

3.2.3 files/svc.conf

컨테이너에서 여러 어플리케이션을 기동하고 관리하기 위해 supervisor를 사용하며, supervisor의 어플리케이션 설정 configuration입니다. 실제 컨테이너에 올릴 어플리케이션 정보를 설정하면 됩니다. 사용할 서비스에 대해서 autostart=true로, 사용하지 않을 서비스에 대해서는 autostart=false로 지정하고, Values.yaml 의 서비스 내용에 맞게 수정해 주시면 됩니다.

```
# vi files/svc.conf
```

```

[program:notebook]
autostart=true
startretries=3
command=start-notebook.sh --config=/etc/jupyter/jupyter_notebook_config.py
process_name=%(program_name)s
numprocs=1
umask=022

[program:tensorboard]
autostart=false
startretries=3

```

```

user=poscoict
command=tensorboard --logdir=/home/poscoict/work --host 0.0.0.0
process_name=%(program_name)s
numprocs=1
umask=022

[program:webvnc]
autostart=true
startretries=3
user=poscoict
directory=/home/poscoict/vnc_startup
environment=PATH="%{ENV_PATH}s",NO_VNC_HOME="%{ENV_NO_VNC_HOME}s",
NO_VNC_PORT="%{ENV_NO_VNC_PORT}s",VNC_PW="%{ENV_VNC_PW}s",
VNC_BLACKLIST_THRESHOLD="%{ENV_VNC_BLACKLIST_THRESHOLD}s",VNC_BLACKLIST_TIMEOUT="%{ENV_VN
C_BLACKLIST_TIMEOUT}s",
VNC_COL_DEPTH="%{ENV_VNC_COL_DEPTH}s",VNC_PORT="%{ENV_VNC_PORT}s",VNC_RESOLUTION="%{ENV_
VNC_RESOLUTION}s",
VNC_STARTUPDIR=%{ENV_VNC_STARTUPDIR}s,VNC_CONFDIR=%{ENV_VNC_CONFDIR}s,VNC_PW="%{ENV_VNC
_PW}s"
command=/home/poscoict/vnc_startup/startup.sh
process_name=%(program_name)s
numprocs=1
umask=022

[program:sshd]
autostart=true
startretries=0
command=sh -c "echo poscoict:%{ENV_PASSWORD}s | chpasswd; /etc/init.d/ssh start"
startsec=0

```

3.3 Chart 등록

Chart의 수정이 완료되면, 수정한 Chart를 Chart저장소에 등록해야 합니다. 차트 디렉토리(예시: standard-test)에서 차트를 push하여 업로드합니다. 공식 차트 저장소 이름은 poscoict입니다.

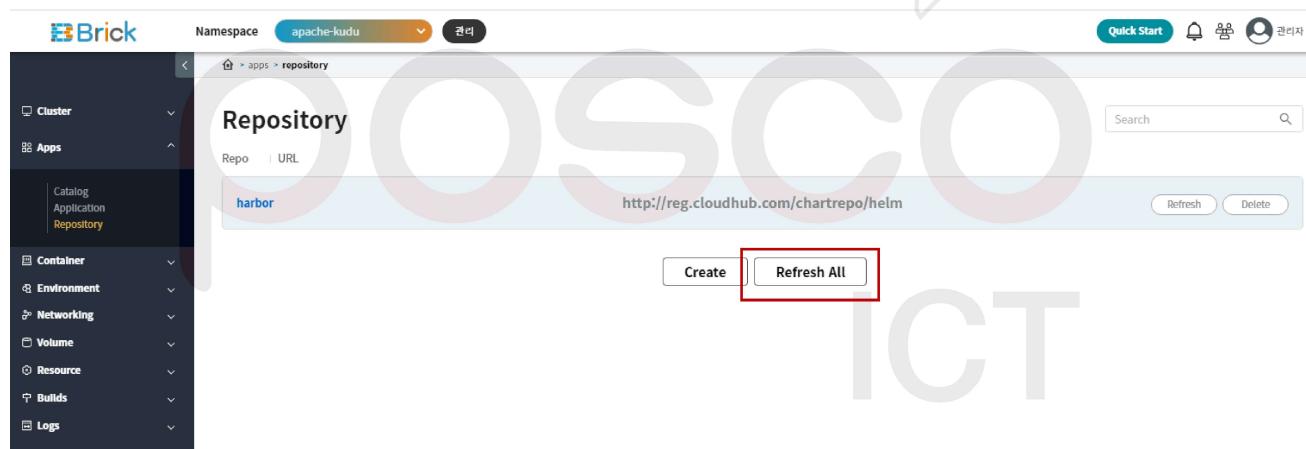
※ 차트 저장소 이름 : **poscoict**

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

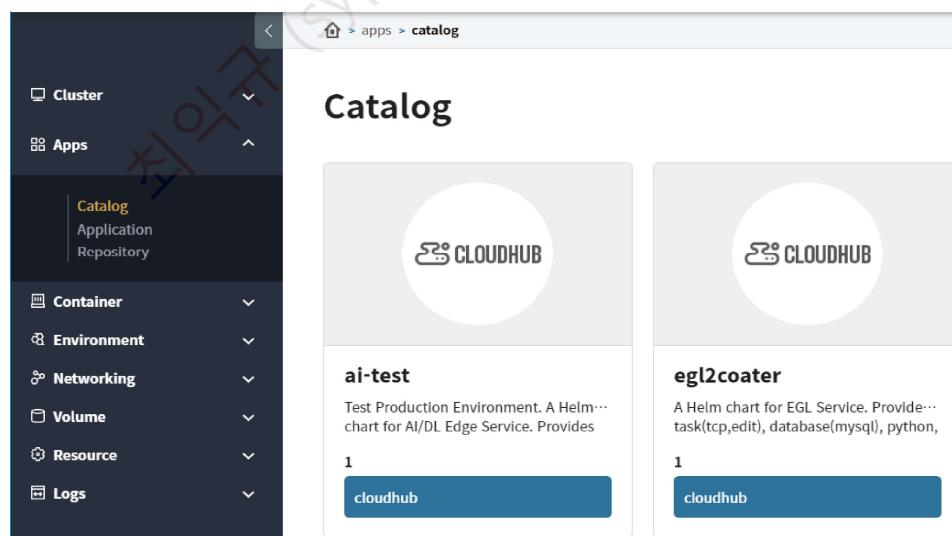
# helm push ./standard-test poscoict # 명령 실행 디렉토리가 해당 차트 밖인 경우, Chart.yaml 위치
```

3.4 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.

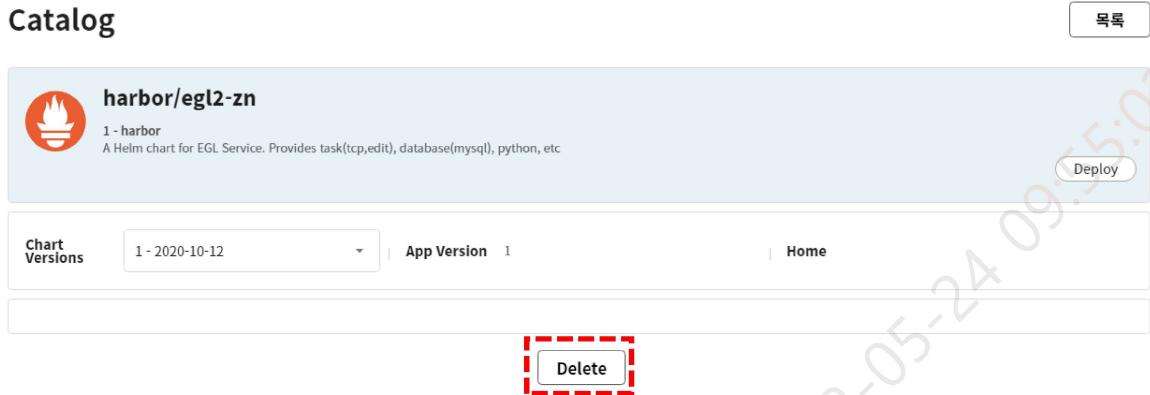


이후 관리화면 – Apps –Catalog 화면에서 아래와 같이 신규 등록된 차트(ai-test)가 보이면 정상입니다.



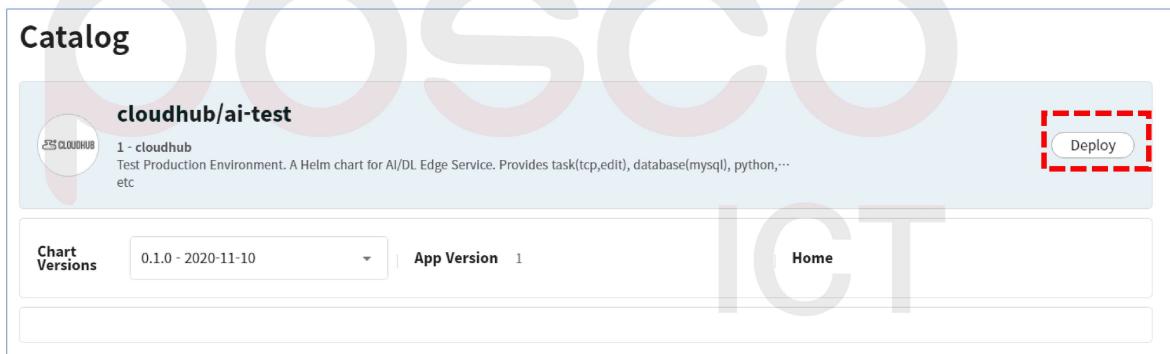
3.5 Chart 삭제

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.



3.6 Chart 기동

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



- 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