딥러닝, 빅데이터 연구를 위한 Docker

작성자: 경상국립대학교 컴퓨터과학과 조교 이자룡

목차

0. Docker 소개

하나의 GPU 서버를 딥러닝 목적으로 여러 연구자가 사용하기 위해 세팅하는 것은 매우 어려운 일입니다. 각각의 연구자가 원하는 개발환경은 서로 상이하며, 연구자 한 명에게 부여할 수 있는 자원 역시도 한정되어야 하기 때문입니다.

이러한 환경을 구성하기 위해, 다음의 전통적인 방식들이 있었습니다.

1. 하나의 컴퓨터에 여러 명의 사용자
리눅스 환경은 기본적으로 여러 명의 사용자를 가정합니다. 만약 딥러닝 연구를 하길 원하는 연구자가 있다면 그 수에 맞게 계정을 생성하여 각각의 계정에 관리자 권한을 부여해야 하는데, 이는 종합적으로 상당히 위험할 수 있습니다. 각각의 관리자가 의존성과 버전을 생각하지 않고 패키지를 설치할 시 타 연구자에게 심각한 영향을 초래하거나, 실수 혹은 고의로 타 연구자의 결과물에 접근할 수 있기 때문입니다.

2. 가상환경
 전통적으로 하나의 운영체제 위에 여러 개의 가상머신을 사용하는 방식이 보편적이었으나, 여러 개의 운영체제를 가동한다는 것은 그 자체만으로도 심각한 자원의 낭비를 초래할 수밖에 없으며, 서버 관리자는 각각의 가상환경에 대한 CPU, 메모리 및 GPU 자원을 배분하는데 있어 세심한 주의를 기울여야 합니다.

이러한 문제를 해결하기 위해, 새로 설치된 GPU 서버는 Docker를 사용하여 운영됩니다. Docker를 사용할 경우 다음과 같은 장점이 있습니다.
  1. 가상환경처럼 운영체제 위에 운영체제를 올리는 방식이 아니라, 실행 환경만 독립적으로 작동합니다. 그렇기 때문에 서버의 자원을 상당 부분 절약할 수 있으며, 로컬 프로그램을 실행시키는 것과 동일한 성능을 보입니다.
  2. 관리자가 아닌 일반 사용자로 접속해도 완벽하게 분리된 개발환경을 구성할 수 있습니다. Docker의 가상 작업공간은 컨테이너 단위로 나뉘는데, 각각의 컨테이너는 다른 컨테이너에 영향을 미치지 않습니다. 또한 컨테이너는 독립된 역할을 수행하기 때문에 연구자는 다른 컨테이너 혹은 서버 컴퓨터의 패키지 의존성과 버전 문제를 고려할 필요가 없습니다.
  3. 사용자의 개발환경은 하나의 이미지로 저장되어 Docker Hub 라는 사이트에 업로드할 수 있습니다. 사용자는 Docker Hub Repository 에 자신의 이미지를 가지고 있기 때문에, 매번 서버 환경을 다시 구성할 필요가 없습니다. 사용자는 필요할 때마다 Docker Hub 에 접속해 마치 클라우드를 이용하는 것처럼 내려받아 사용하기만 하면 됩니다.

4. Docker 에서 이미지를 작동시키면 컨테이너라는 형태가 되며, 이것은 곧 하나의 프로세스를 의미합니다. 서버 컴퓨터에 있는 자원 및 프로그램을 Docker 컨테이너 안에서 사용할 수 있습니다. 이것은 곧 GPU 자원도 컨테이너 안에서 사용 가능하다는 의미입니다.

5. 컨테이너 사용 중 문제가 생겼을 땐 복잡한 작업을 거칠 필요 없이 컨테이너를 중지시키고, 문제를 파악한 다음 새 컨테이너를 실행시키기만 하면 됩니다. 컨테이너만 작업하기 때문에 실제 컴퓨터나 다른 작업자에게 전혀 영향을 끼치지 않습니다.

이러한 장점들을 종합적으로 고려하여 학과의 공용 GPU 서버에 Docker engine 을 설치해 운용하기로 하였습니다. 이 사용설명서에서 최종적으로, Docker PyTorch 이미지를 사용해 간단한 예제를 실습해보겠습니다.

1. 공용 GPU 서버 접속

절차는 다음과 같습니다.

1. 서버 관리자인 실습 조교에게 문의하여 공용 GPU 서버 계정 요청
2. 관리자는 Ubuntu 관리자 계정이 아닌 일반 계정 하나를 생성해, 타 연구자의 이미지 및 컨테이너에 접근할 수 없도록 적절한 보안작업 진행
3. 작업이 완료되면 ssh 접속 후 계정 비밀번호 변경 후 사용

2. 기본적인 Docker 명령어 - 조회 및 삭제

```
$ docker images       현재 보유한 이미지 목록 출력
결과:
REPOSITORY   TAG          IMAGE ID       CREATED        SIZE
lunchbox     latest       cc719e392903   20 hours ago   472MB
php          8.0-apache   aa79271a3ea6   7 days ago     472MB

현재 사용자는 php:8.0-apache 와 lunchbox:latest 이미지를 보유 중입니다.
REPOSITORY          이미지 이름
TAG                 이미지의 버전. 이미지별로 다양한 목적에 맞게 변형한 TAG를 제공합니다.
                    lunchbox:latest 이미지의 latest는 가장 최신 버전임을 의미합니다.
                              docker hub 에서 별다른 태그 없이 이미지를 pull 하면
                              기본적으로 latest 버전을 가져옵니다.
                    php:8.0-apache 이미지는 php8.0 버전이며, apache 를 포함하고 있습니다.
IMAGE ID            해당 이미지의 고유한 ID 입니다.
                    Docker hub 에 등록된 모든 이미지는 각기 다른 ID를 보유하고 있습니다.
CREATED             해당 이미지가 생성된 시점입니다.
                    여기선 해당 이미지가 최종 업데이트된 시간을 의미합니다.
SIZE                해당 이미지의 용량입니다.
```

$ docker ps -a　　현재 보유한 모든 컨테이너 목록 출력

결과:

```
CONTAINER ID   IMAGE     COMMAND                  CREATED                  STATUS                      PORTS                                     NAMES
daf5a4ffad79   lunchbox  "docker-php-entrypoi…"   About a minute ago       Up About a minute           0.0.0.0:1993->80/tcp, :::1993->80/tcp     lunchbox1
7471695662e4   lunchbox  "docker-php-entrypoi…"   3 seconds ago            Exited (0) 1 second ago                                               lunchbox2
```

이미지를 사용하려면 컨테이너로 바꿔야 합니다. 사용자가 보유하고 있는 모든 컨테이너 목록입니다.

CONTAINER ID　　해당 컨테이너의 고유한 ID입니다.
하나의 이미지에서 만들어진 컨테이너는 각각 다른 ID를 가집니다.
이 아이디는 서버 컴퓨터 안에서만 중복 없이 유효합니다.

IMAGE　　컨테이너에 해당하는 이미지를 의미합니다.

CREATED　　해당 컨테이너가 생성된 시점입니다.
여기선 해당 컨테이너가 서버 컴퓨터에 생성된 시간을 의미합니다.

STATUS　　해당 컨테이너의 상태를 말합니다.
lunchbox1 컨테이너는 Up, 즉 약 1분 전부터 실행 중입니다.
lunchbox2 컨테이너는 1초 전에 종료되었습니다.

PORT　　해당 컨테이너에 지정된 포트 번호를 말합니다.
웹 개발 시에 Docker 를 사용하게 되면 기본적으로 3개의 포트가 필요합니다.
예를 들면 다음과 같습니다.

Client (Frontend)	Apache2.0	Port 8080
Server (Backend)	Java8.0	Port 5000
Database	MySQL8.0	Port 3306

이것은 곧, 세 개의 컨테이너로 나눠서 서비스해야 함을 의미합니다.
Docker의 -p 명령을 이용해 서버 컴퓨터의 port 번호와 컨테이너의 port 번호를 연결합니다.
* 딥러닝, 빅데이터 연구를 위해 http 서버, api, database 컨테이너를
분리할 필요가 있을 수 있습니다.
이 경우엔 반드시, 실습 조교에게 이용 가능한 포트를 지정받아 사용하시기 바랍니다.

NAME　　해당 컨테이너를 식별하는 이름입니다.
사용자가 특별히 지정하지 않을 시, 영어사전에서 두 개의 단어를 임의로 조합해 만듭니다.

$ docker ps　　현재 '실행중인' 컨테이너 목록 출력

결과:

```
CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS             PORTS                                     NAMES
daf5a4ffad79   lunchbox  "docker-php-entrypoi…"   About an hour ago   Up About an hour   0.0.0.0:1993->80/tcp, :::1993->80/tcp     lunchbox1
```

docker ps -a 명령은 사용자가 보유하고 있는 모든 컨테이너를 출력하지만,
docker ps 명령은 현재 실행 중인 컨테이너만 출력합니다.

$ docker stop 컨테이너이름　　현재 실행 중인 컨테이너 중지

ex) $ docker stop lunchbox1　　실행 중인 lunchbox1 컨테이너 중지

결과:

```
lunchbox1                                                  성공 시 중지한 컨테이너 이름 반환
Error response from daemon: No such container: lunchbox1   실패 시 에러
```

$ docker start 컨테이너이름　　컨테이너 실행

$ docker restart 컨테이너이름　　컨테이너 재실행

$ docker rm 컨테이너 이름 — 컨테이너 삭제

반드시 stop 한 후에 실행해야 에러 발생 안 합니다.

$ docker rmi 이미지이름:태그 — 이미지 삭제

만약 해당 이미지에서 비롯된 컨테이너를 보유하고 있다면, 삭제 명령이 실행되지 않습니다.

ex) $ docker rmi php:8.0-apache — php:8.0-apache 이미지 삭제
반드시 태그까지 함께 써줘야 삭제가 진행됩니다.
보유하고 있는 여러 php 태그가 있을 수 있기 때문입니다.

결과:

```
Untagged: php:8.0-apache
Untagged: php@sha256:be7c3c1cef182a0f8dfeaf2208a5ce39649d6efc9ebbf5a325a4d6291e00defe
```

$ docker pull 이미지이름:태그 — Docker hub 에서 이미지 받아오기

ex) $ docker pull node — node:latest 이미지를 docker hub 에서 받아오기
즉, 아무 태그도 안 쓰면 latest 를 기본으로 받아옴

$ docker pull node:alpine3.14 — node:alpine3.14 를 docker hub 에서 받아오기

$ docker run -it --name 원하는이름 이미지이름:태그 — 이미지를 컨테이너로 만들어 포그라운드에서 실행

ex) $ docker run -it --name testNode node:alpine3.14 — node:alpine3.14 이미지를
-it 옵션을 이용해 포그라운드에서 실행
컨테이너이름은 testNode

* 이미지가 서버 컴퓨터에 존재하지 않을 경우, 자동으로 pull 작업이 먼저 실행됨.
** 만약 백그라운드 실행을 원한다면 — -d 플래그 사용
만약 컨테이너 종료와 함께 컨테이너를 자동삭제하고싶으면 — --rm 플래그 사용