Docker 가상화 환경 구성과 Github 개발 관리

https://bit.ly/docker-github

박동희

Docker 가상환경

- Docker 소개
- Docker 사용하는 이유?
- Docker 동작 원리 및 개념
- 실습

Docker 소개

- 배경 기술: 리눅스 컨테이너 LXC. 가상화 기술로 리눅스 커널에서 지원. 유저에게 독립된 리소스 환경(네임스페이스)과 관리 기능(cgroups)을 제공
- Docker의 시작: 리눅스 컨테이너 관리 도구로 프로젝트 시작. PyCon 2013 에서 Solomon Hykes가 Docker 프로젝트 발표. 2014년 v1.0릴리즈 하면서 본격적인 사용 시작
- 유명해진 이유: 마이크로 서비스 구축을 위해 컨터이너 기반의 인프라 도입

Docker를 사용하는 이유?

개발자에게 특히 유용한 이유

- 다양한 환경에 대한 호환성 문제 해결(OS, 프로그래밍 언어, 라이브러리 버전 의존성 해결)
- 하나의 컴퓨터에 여러개의 독립적인 환경 제공
- 개발환경을 그대로 배포 환경에 사용

Docker 동작 원리 및 개념

- **이미지**: Docker 이미지는 어플리케이션과 그 모든 의존성을 포함하는 경량의, 이동 가능한, 자체 충분한 소프트웨어 패키지. 이미지는 불변(immutable)이며 컨테이너를 시작하는 데 사용.
- **컨테이너**: 이미지를 기반으로 실행되는 런타임 인스턴스. 각 컨테이너는 분리된 환경에서 실행되며, 운영 체제의 나머지 부분과 격리됨.
- **도커 데몬(Docker Daemon)**: 도커 컨테이너를 생성, 실행, 모니터링하는 주요 서비스.
- **도커 클라이언트(Docker Client)**: 사용자가 도커 데몬과 소통할 수 있게 하는 인터페이스. 커맨드 라인 인터페이스(CLI)를 통해 도커 데몬에 명령을 보낼 수 있음.
- **도커 레지스트리(Docker Registry)**: 도커 이미지를 저장하는 곳으. Docker Hub, Github Package Registry

실습

- docker engine 설치
- NVIDIA Container Toolkit 설치
- docker desktop 설치

docker engine 설치

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh ./get-docker.sh --dry-run
```

유저 계정에서 docker를 사용하려면 `docker` 그룹에 유저 추가 필요

```
sudo usermod -aG docker $USER
```

nvidia container toolkit 설치 확인

```
sudo docker run --rm --runtime=nvidia --gpus all nvidia/cuda:11.4.3-devel-ubuntu20.04 nvidia-smi
```

Docker 환경구성

- Docker Client 자주 사용하는 명령
- Dockerfile
- Docker 사용 실습

Docker Client 명령 #1

- 생성된 도커 이미지 리스트

```
docker images
```

- 현재 실행된 도커 컨테이너 리스트

```
docker ps
```

- 도커 이미지 삭제

```
docker rmi '이미지이름 또는 ID'
```

- 도커 컨테이너 삭제

```
docker rm '컨테이너 이름 또는 컨테이너ID'
```

Docker Client 명령 #2

- 도커 이미지 빌드

```
docker build . -t '이미지 이름'
```

- 도커 이미지 실행하여 컨테이너 생성

```
docker run '이미지 이름' '실행 프로그램'
```

- 실행되어 있는 도커 컨테이너에 접속

```
docker attach '컨테이너 이름 또는 컨테이너ID'
```

- 실행되어 있는 도커 컨테이너 내부의 프로그램 실행

```
docker exec -it '컨테이너 이름 또는 컨테이너ID' '실행 프로그램'
```

- Docker CLI Cheat Sheet: https://docs.docker.com/get-started/docker_cheatsheet.pdf

Dockerfile

- Docker 이미지를 만든는 명세파일
- Dockerfile 주요 명령: https://docs.docker.com/reference/dockerfile/

예시

Dockerfile

```
FROM alpine
CMD ["echo", "Hello World!"]
```

Dockerfile 로 부터 이미지 생성, 컨테이너 실행

```
docker build -t hello .
docker run --rm hello
```

실습: Docker `hello-world`

실습: https://github.com/docker-library/hello-world

```
docker run hello-world
```

컨테이너 삭제

```
docker ps -a
docker ps -aq --filter ancestor=hello-world

docker rm '컨테이너 id'

# 이미지와 연결된 컨테이너 삭제
docker rm -f $(docker ps -aq --filter ancestor=hello-world)
```

이미지 삭제

```
docker images
docker rmi hello-world
```

Python 환경구성

- virtualenv: 하나의 파이썬 버전으로 여러 가상 환경 생성
- pyenv: 여러개의 파이썬 버전설치, 로컬 및 글로벌 환경 설정

실습: Dockerfile 읽기

torch 2.0.0+cu117 가상 환경 구성

```
- python : 3.8.8
- cuda : 11.4
- torch : 2.0.0
- gpu : NVIDIA RTX A6000
```

- 다운로드: https://github.com/donghee/pytorch-containers/tree/main/torch-2.0.0

Dockerfile 빌드

torch 2.0.0+cu117 이미지 빌드

```
docker build -t ghcr.io/donghee/torch-2.0.0:cuda11.7 .
```

torch 2.0.0+cu117 컨테이너 실행

```
docker run -it --privileged --rm --gpus all ghcr.io/donghee/torch-2.0.0:cuda11.7 bash
```

가상환경 설치 버전 확인

python shell에서 pytorch 버전 확인, cuda version 확인

```
import torch
print(torch.__version__)
print(torch.version.cuda)
```

실습: torch 1.13.1 가상 환경 구성

torch 2.0.0 Dockerfile을 참고하여 torch 1.13.1 가상 환경 구성하기

```
- python : 3.10.13
- cuda : 11.7
- torch : 1.13.1
```

참고: https://github.com/donghee/pytorch-containers/tree/main/torch-1.13.1

Git

- Git 소개
- Github
- Git Flow
- Git 사용법
- Github 이슈관리, 문서화 (markdown)
- Github Packages Container Registry
- 실습

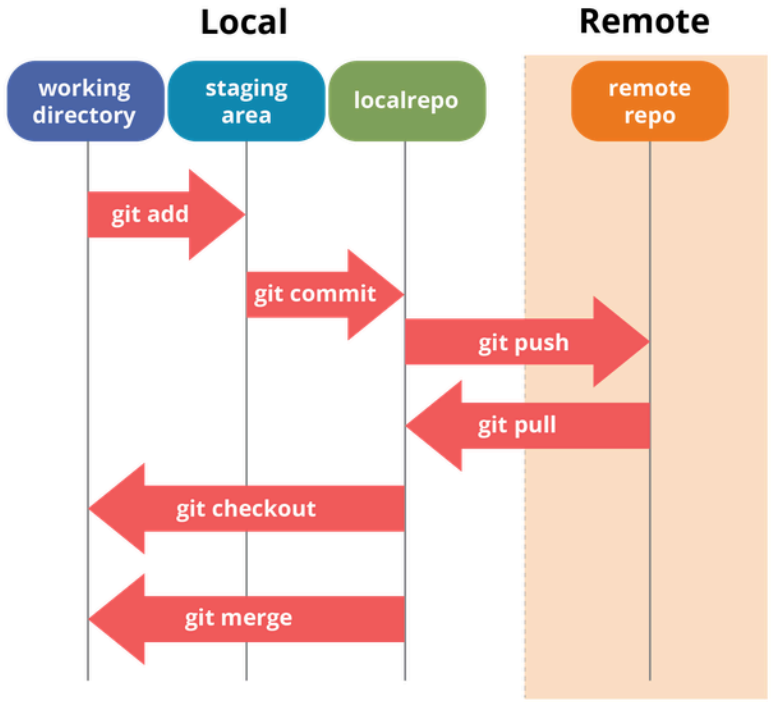
Git 소개

- 리누스 토발즈가 2005년 Git을 개발
  - 리눅스 커널 개발의 효율적인 코드 관리와 협업을 지원하기 위해 개발
  - 높은 성능의 분산 버전 관리 시스템 필요. 기존에 사용하는 버전 관리 시스템의 라이센스 문제가 있었음.
- 분산 버전 관리 시스템: 독립적인 작업 가능
- 브랜치와 병합기능이 탁월

Github

- **소프트웨어 개발 협업 플랫폼**: Git 버전 관리 시스템을 이용하여 코드 호스팅과 버전 관리 기능
- **프로젝트 관리**: 이슈 트래킹, 코드 리뷰, 프로젝트 관리 도구를 제공. 오픈소스 코드의 경우 무료 사용 가능

Git Flow: 주요 개념



- branch, commit, log
- local, remote repo
- merge, rebase

Git 사용법

GIT CHEAT SHEET

```
git init
git clone ‘저장소 주소’
git pull
git add ‘파일 이름’
git commit -m '커밋 메시지'
git push
git branch ‘브랜치 이름’
git checkout ‘이동할 브랜치 이름’
git rebase
```

Github 이슈 관리

- 이슈 관리를 위한 게시판
- 커밋 과 연동 가능

https://github.com/donghee/pytorch-containers/issues

Github를 이용한 문서화

Github Markdown

- README.md
- Github pages: 웹호스팅 가능
- https://github.com/sadtalker/sadtalker.github.io Github pages를 이용한 문서화 예시

Github Packages Container registry

`GitHub Packages`의 Container registry 를 이용하여 Docker 이미지 배포

- https://ghcr.io
- Working with the Container registry

예시:

- https://github.com/donghee/wearable_robot_eval/pkgs/container/wearable_robot_eval

Git 실습

- github에 새로운 코드 저장소 만들어서 README.md 파일을 추가 해보자.
  - 힌트: github 가입, 새로운 저장소 추가, ssh 키추가. README.md 수정, 커밋, push
- 저장소에 Dockerfile 올리기
  - 'Dockerfile 추가' 라고 하는 새로운 이슈 추가
  - Dockerfile 을 커밋하고, 이슈와 연결
    - 힌트: 커밋 메시지 'Closes #1, Add Dockerfile'
- Docker image git registrey에 올리기
- 참고: https://www.youtube.com/watch?v=RGOj5yH7evk

더 해보면 좋은 도구

- https://huggingface.co/
- vscode remote development 플러그인
- docker desktop