docker

Requirements

- 시스템과 인프라 기초 지식
  - 시스템 기반의 구성 요소; 기능 요구사항(functional requirement), 비기능 요구사항(non-functional requirement): 신뢰성, 확장성, 운용성, 보안 등, 하드웨어, 네트워크, OS, 미들웨어
  - 클라우드와 온프레미스(on-premises)
  - 시스템 기반의 구축/운용 흐름
- 하드웨어와 네트워크 기초 지식
  - 서버 장비
  - 네트워크 주소
  - OSI 참조 모델과 통신 프로토콜
  - 방화벽
  - 라우터/레이어3 스위치
- Linux 기초 지식
  - 파일 시스템
  - 디렉토리 구성
  - 보안 기능
- 미들웨어 기초 지식
  - 웹서버/웹 애플리케이션 서버
  - 데이터베이스 서버
  - 시스템 감시 툴
- 인프라 구성 관리 기초 지식
  - 인프라 구성 관리
  - 지속적 인티그레이션/지속적 딜리버리

컨테이너 기술과 Docker 개요

- 컨테이너
- Docker
- Docker 기능;
  - Docker 이미지를 만드는 기능(Build)
  - Docker 이미지를 공유하는 기능 (Ship)
  - Docker 컨테이너를 작동시키는 기능 (Run)
- Docker의 작동 구조
  - 컨테이너를 구획하는 장치 (namespace)
  - 릴리스 관리 장치 (cgroups)
  - 네트워크 구성(가상 브리지/가상 NIC)
  - Docker 이미지의 데이터 관리 장치

Getting started docker

- 설치

- 작동 확인
  - hello world
  - 버전 확인 (docker version)
  - 실행 환경 확인 (docker system info)
  - 디스크 이용 상황 (docker system df)
- nginx 동작 예제; docker 이미지 다운로드 → nginx 작동 → nginx 작동 확인 → nginx 기동 정지

Commands

이미지 조작

Docker Hub

- https://hub.docker.com

이미지 다운로드(docker image pull)

docker image pull

```
$ docker image pull [옵션] 이미지명[:태그명]
```

```
$ docker image pull centos:7  # CentOS의 이미지 취득
$ docker image pull -a centos  # CentOS의 모든 태그 이미지 취득
$ docker image pull gcr.io.tensorflow/tensorflow  # TensorFlow의 URL을 지정하여 이미지 취득
```

이미지 목록 표시(docker image ls)

docker image ls

```
$ docker image ls [옵션] [리포지토리명]
```

옵션	설명
-all, -a	모든 이미지를 표시
–digests	다이제스트를 표시할지 말지
–no-trunc	결과를 모두 표시
–quiet, -q	Docker 이미지 ID만 표시

```
$ docker image ls
```

결과

항목	설명
REPOSITORY	이미지 이름
TAG	이미지 태그명
IMAGE ID	이미지 ID
CREATED	작성일
SIZE	이미지 크기

이미지 상세 정보 확인(docker image inspect)

이미지 태그 설정(docker image tag)

이미지 검색(docker search)

이미지 삭제(docker image rm)

Docker Hub에 로그인(docker login)

이미지 업로드(docker image push)

Docker Hub에서 로그아웃(docker logout)

컨테이너 생성/시작/정지

Docker 컨테이너의 라이프 사이클

컨테이너 생성 및 시작(docker container run)

컨테이너의 백그라운드 실행(docker container run)

컨테이너의 네트워크 설정(docker container run)

자원을 지정하여 컨테이너 생성 및 실행(docker container run)

컨테이너를 생성 및 시작하는 환경을 지정(docker container run)

가동 컨테이너 목록 표시(docker container ls)

컨테이너 가동 확인(docker container stats)

컨테이너 시작(docker container start)

컨테이너 정지(docker container stop)

컨테이너 재시작(docker container restart)

컨테이너 삭제(docker container rm)

컨테이너 중단/재개(docker container pause/docker container unpause)

컨테이너 네트워크

네트워크 목록 표시(docker network ls)

네트워크 작성(docker network create)

네트워크 연결(docker network connect/docker network disconnect)

네트워크 상세 정보 확인(docker network inspect)

네트워크 삭제(docker network rm)

가동중인 컨테이너 조작

가동 컨테이너 연결(docker container attach)

가동 컨테이너에서 프로세스 실행(docker container exec)

가동 컨테이너의 프로세스 확인(docker container top)

가동 컨테이너의 포트 전송 확인(docker container port)

컨테이너 이름 변경(docker container rename)

컨테이너 안의 파일을 복사(docker container cp)

컨테이너 조작의 차분 확인(docker container diff)

이미지 생성

컨테이너로부터 이미지 작성(docker container commit)

컨테이너를 tar 파일로 출력(docker container export)

tar 파일로부터 이미지 작성(docker image import)

이미지 저장(docker image save)

이미지 읽어 들이기(docker image load)

불필요한 이미지/컨테이너를 일괄 삭제(docker system prune)

Dockerfile을 사용한 코드에 의한 서버 구축

Dockerfile을 사용한 구성 관리

Dockerfile이란?

Dockerfile의 기본 구문

Dockerfile 작성

Dockerfile의 빌드와 이미지 레이어

Dockerfile로부터 Docker 이미지 만들기

Docker 이미지의 레이어 구조

멀티스테이지 빌드를 사용한 애플리케이션 개발

Dockerfile 만들기

Docker 이미지의 빌드

Docker 컨테이너의 시작

명령 및 데몬 실행

명령 실행(RUN 실행)

데몬 실행(CMD 명령)

데몬 실행(ENTRYPOINT 명령)

빌드 완료 후에 실행되는 명령(ONBUILD 명령)

시스템 콜 시그널의 설정(STOPSIGNAL 명령)

컨테이너의 헬스 체크 명령(HEALTHCHECK 명령)

환경 및 네트워크 설정

환경변수 설정(ENV 명령)

작업 디렉토리 지정(WORKDIR 명령)

사용자 지정(USER 명령)

라벨 지정(LABEL 명령)

포트 설정(EXPOSE 명령)

Dockerfile 내 변수의 설정**(ARG 명령)**

기본 쉘 설정(SHELL 명령)

파일 설정

파일 및 디렉토리 추가(ADD 명령)

파일 복사(COPY 명령)

볼륨 마운트(VOLUME 명령)

Docker 이미지 공개

Docker 이미지의 자동 생성 및 공개

Automated Build의 흐름

GitHub에 공개하기

Docker Hub의 링크 설정

Dockerfile의 빌드

Docker 이미지 확인

Docker Registry를 사용한 프라이빗 레지스트리 구축

로컬 환경에 Docker 레지스트리 구축하기

Docker 이미지 업로드

Docker 이미지의 다운로드와 작동 확인

클라우드 서비스를 사용한 프라이빗 레지스트리 구축

Google Container Registry 준비하기

Docker 이미지의 업로드

Docker 이미지의 다운로드와 작동 확인

여러 컨테이너의 운용 관리

여러 컨테이너 관리의 개요

웹 3계층 시스템 아키텍처

영구 데이터의 관리

Docker Compose

웹 애플리케이션을 로컬에서 움직여 보자

Compose 구성 파일의 작성

여러 Docker 컨테이너 시작

여러 Docker 컨테이너 정지

Docker Compose를 사용한 여러 컨테이너의 구성 관리

docker-compose.yml의 개요

이미지 지정(image)

이미지 빌드(build)

컨테이너 안에서 작동하는 명령 지정(command/entrypoint)

컨테이너 간 연결(links)

컨테이너 간 통신(ports/expose)

서비스의 의존관계 정의(depends_on)

컨테이너 환경변수 지정(environment/env_file)

컨테이너 정보 설정(container_name/labels)

컨테이너 데이터 관리(volumes/volumes_from)

Docker Compose를 사용한 여러 컨테이너의 운용

Docker Compose의 버전 확인

Docker COmpose의 기본 명령

여러 컨테이너의 생성(up)

여러 컨테이너 확인(ps/logs)

컨테이너에서 명령 실행(run)

여러 컨테이너 시작/정지/재시작(start/stop/restart)

여러 컨테이너 일시 정시/재개(pause/unpause)

서비스의 구성 확인(port/config)

여러 컨테이너 강제 정지/삭제(kill/rm)

여러 리소스의 일괄 삭제(down)

멀티호스트 환경에서 Docker 실행 환경 구축

멀티호스트 환경에서 컨테이너 관리의 개요

멀티호스트 환경과 클러스터링

Docker Machine이란?

웹 애플리케이션을 서비스 공개해 보자

Docker 실행 환경 작성

웹 애플리케이션 전개

Docker 실행 환경 삭제

Docker Machine을 사용한 실행 환경 구축

Docker Machine의 기본 명령

실행 환경 작성(create)

실행 환경 목록 표시(ls/status/url)

실행 환경에 대한 SSH 연결(ssh)

실행 환경 시작/정지/재시작(start/stop/restart)

실행 환경으로부터 파일 다운로드(scp)

실행 환경 삭제(rm/kill)

실행 환경 정보 확인(ip/inspect)

클라우드를 사용한 Docker 실행 환경 구축

클라우드 환경에서 Docker 오케스트레이션하기

분산 환경에서의 컨테이너 운용 관리

퍼블릭 클라우드가 제공하는 매니지드 서비스

Google Cloud Platform의 컨테이너 관련 서비스

Kubernetes의 개요

Kubernetes의 서버 구성

애플리케이션 구성 관리(Pod, ReplicaSet, Deployment)

네트워크 관리(Service)

Label을 사용한 리소스 식별

Kubernetes의 구조

GCP를 사용한 Docker 애플리케이션 개발

애플리케이션 개발 흐름

소스코드 관리(Cloud Source Repositories)

Docker 이미지 빌드(Cloud Container Builder)

GCP를 사용한 Docker 애플리케이션 실행 환경 구축

Kubernetes 클러스터 구축

애플리케이션의 설정 정보 관리(ConfigMap, Secrets)

앱의 전개(Deployment)

서비스 공개(Service)

앱의 버전업(Blue-Green Deployment)

배치 잡 실행(CronJob)

클라우드를 사용한 Docker 실행 환경의 운용 관리

시스템 운용의 기초 지식

가용성 관리

수용성(Capacity) 관리

시스템 감시

GKE를 사용한 Docker 실행 환경의 운용

Kubernetes의 스테이터스 확인

Kubernetes의 Pod 관리

Kubernetes의 노드 관리

Kubernetes의 리소스 작성/삭제/변경

Kubernetes의 업그레이드/다운그레이드

Stackdriver에서 로그 확인

node docker image

```
$ docker exec -it node bash
```

nginx-php-fpm docker image

```
richarvey/nginx-php-fpm



$ docker run --name ngx-php -d richarvey/nginx-php-fpm

$ docker exec -e 'DOMAIN=theta5912.net' -e 'GIT_EMAIL=alex@theta5912.net' -e
'WEBROOT=/var/www/html' -t  ngx-php /usr/bin/letsencrypt-setup

$ docker exec -t -i ngx-php /bin/bash
```

```
$ docker exec -e 'DOMAIN=theta5912.net'
$ docker exec -e 'GIT_EMAIL=alex@theta5912.net'
$ docker exec -t ngx-php /usr/bin/letsencrypt-setup (90days)

$ docker exec -t ngx-php /usr/bin/letsencrypt-renew
$ docker exec -e 'DOMAIN=theta5912.net' -t ngx-php /usr/bin/letsencrypt-
renew

$ docker start ngx-php

$ docker commit -a "Alex Levine<alex@theta5912.net>" -m "update dokuwiki,
December 29, 2017. Friday" ngx-php

---
setting the timezone
# apk add tzdata
# ls /usr/share/zoneinfo

# cp /usr/share/zoneinfo/Asia/Seoul /etc/localtime
# echo "Asia/Seoul" > /etc/timezone
# date

# apk del tzdata

in dokuwiki
dokuwiki/inc/init.php 88
date_default_timezone_set("Asia/Seoul");

---
# apk update
# apk upgrade
# rm -rf /var/cache/apk/*


---
# export DOMAIN=theta5912.net
# export GIT_EMAIL=alex@theta5912.net
# export WEBROOT=/var/www/html
# /usr/bin/letsencrypt-setup


---
# wget http://download.dokuwiki.org/src/dokuwiki/dokuwiki-stable.tgz
# tar xvf dokuwiki-stable.tgz --strip 1


---
cp

host -> container
$ docker cp /path/foo.txt mycontainer:/path/foo.txt
```

```
container -> host
$ docker cp mycontainer:/path/foo.txt /path/foo.txt

---
$ docker run -i -t ---name <container name> -v <host directory>
```

Google Cloud Platform 사용법

A.1 계정 등록

[1] 등록 시작 [2] 계정 정보 등록

A.2 프로젝트 작성과 삭제

[1] 프로젝트 작성 [2] 프로젝트명 설정 [3] 프로젝트 삭제

A.3 Cloud Console 사용법

툴과 서비스 대시보드

A.4 Cloud Shell 사용법

A.5 Cloud SDK 설치하기