Docker

도커 이미지

- 컨테이너를 생성할 때 필요한 요소
- 가상 머신의 iso 파일과 비슷한 무엇인가
- 사용할 때는 읽기 전용으로 사용되며
- 도커 명령어로 내려받을 수 있음
- 이름 형식 : [저장소 이름]/[이미지 이름]:[태그]의 형태
  - bitnami/pytorch:2.0.1
  - bitnami/pytorch:latest
  - ubuntu:22.04

도커 허브

- https://hub.docker.com/

도커 컨테이너 생성/시작

- 도커 엔진 버전 확인
  - docker –v
- 도커를 쓸 수 있는 상황인가?
  - docker –ps
- 위에거는 틀렸고, 도커 데몬이 돌고 있는지 확인을 해야 합니다.
- 컨테이너 실행
  - docker run
  - docker run –i –t ubuntu:22.04
    - 종료 : exit
    - 나오기 : CTRL P, Q

도커 컨테이너 생성/시작 (2)

- 이미지 가져오기
  - docker pull
    - docker pull bitnami/pytorch
      - 기본으로 :latest tag를 가지고 오게 됨
  - 이미지 확인
    - docker images
- 컨테이너 생성
  - docker create
    - docker create -i -t --name mytorch bitnami/pytorch:latest
  - 이름을 생략하면?
    - 형용사_명사 형태로 이름은 자동 부여

도커 컨테이너 생성/시작 (3)

- 생성된 컨테이너 시작
  - docker start 이름
  - docker attach 이름
- 컨테이너 확인
  - docker ps
    - docker ps -a
    - docker ps -q
  - container id : 컨테이너 식별자
  - command : 컨테이너가 시작될 때 실행할 명령, 지정 가능
    - docker run -i -t ubuntu echo hello world
  - 이름 : 변경 가능 (중복 금지)
    - docker rename oldname newname

도커 컨테이너 삭제

- 삭제
  - docker rm 이름
    - docker rm mytorch
  - 실행중인 컨테이너는 삭제할 수 없으므로 정지
    - docker stop
    - docker rm -f : 강제삭제
  - 모두삭제
    - docker container prune
    - 실행중인 컨테이너는 삭제되지 않음
    - docker rm $(docker ps -a -q) *windows에서는 안 될 수도 있음
- 이미지 삭제 : docker rmi

컨테이너를 노출시키자

- 컨테이너의 네트워크 확인
  - docker run -i -t ubuntu 후 ifconfig
    - apt-get update / apt-get install net-tools
  - 컨터이너 내의 eth0을 동작하게 만드려면
    - 호스트의 IP/PORT를 컨테이너 eth0의 IP/PORT와 바인딩
- 포트 바인딩
  - -p 옵션 : -p 호스트:컨테이너
    - -p 80:80 → 호스트의 80번 포트는 컨테이너의 80번 포트로 바인딩 되어 사용
    - -p 192.168.35.253:8808:80 → 호스트의 192.168.35.253:8808은 컨테이너의 80번 포트로 바인딩 되어 사용

컨테이너를 노출시키자 (2)

- 실습
  - docker run -i -t -p 80:80 --name web ubuntu
  - 웹서버 설치 & 실행
    - apt-get update
    - apt-get install nginx
    - service nginx start

컨테이너 APP 구축

- 내가 mysql이 필요한데...
  - docker 컨테이너로 해 볼 수 있을까?
  - docker hub의 mysql을 방문
  - How to use this image...
- docker run의 -e option
  - 환경변수 설정
  - docker run -i -t -e username=edberg ubuntu
    - echo $username

컨테이너 APP 구축 (2)

- docker run -e MYSQL_ROOT_PASSWORD=... -e MYSQL_DATABASE=... mysql:latest
- docker run -d -e MYSQL_ROOT_PASSWORD=... -e MYSQL_DATABASE=... mysql:latest → detached
- 동작중인 docker의 무엇인가를 실행 : docker exec -i -t
  - docker exec -i -t name /bin/bash
  - mysql -u root -p, use databases

컨테이너 APP 구축 (3)

- 외부에서는 이 mysql을 어떻게 사용하지?
  - port mapping
    - --p 3306:3306
  - 컨테이너 끼리라면?
    - --link : --link 컨테이너이름:호스트이름
  - docker run -d --name mydb -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=testdb mysql:latest
  - docker run –i -t --link mydb:mysqlhost Ubuntu
    - apt-get update && apt-get install mysql-client
    - mysql –h mysqlhost -u root -p

컨테이너 APP 구축 (4)

- wordpress 블로그를 뛰워 봅시다.
  - step1. mysql 컨테이너 실행
    - 환경 변수로 설정해야 할 것
      - MYSQL_ROOT_PASSWORD
      - MYSQL_DATABASE
  - step2. wordpress 컨테이너 실행
    - 환경 변수로 설정해야 할 것
      - WORDPRESS_DB_HOST
      - WORDPRESS_DB_USER
      - WORDPRESS_DB_PASSWORD
      - WORDPRESS_DB_NAME
    - 그리고 LINK –p 8888:80

- container 1 : mysql image로 mydb container. 외부로 노출할 필요가 없어요. 내부에서 컨테이너 끼리만 통신을 하니까.
- 설정을 요구하는것
- -e MYSQL_ROOT_PASSWORD=password1234
- wordpress를 위해서 필요한 설정
- -e MYSQL_DATABASE=testdb
- --name mydb

- container 2 : wordpress image로 wp container. 80번 포트를 외부로 노출. 왜냐하면 외부에서 블로깅을 해야 하니까. --name wp
- 설정을 요구하는 것
- -e WORDPRESS_DB_HOST=db
- -e WORDPRESS_DB_USER=root
- -e WORDPRESS_DB_PASSWORD=password1234
- -e WORDPRESS_DB_NAME=testdb
- --link mydb:db
- -p 8888:80 (누가 8888번 포트로 호스트에 접근하면, 80번 컨테이너 포트로 바인딩)

- docker run –d -e MYSQL_ROOT_PASSWORD=password1234 -e MYSQL_DATABASE=testdb --name mydb mysql:latest

- docker run –d -e WORDPRESS_DB_HOST=db -e WORDPRESS_DB_USER=root -e WORDPRESS_DB_PASSWORD=password1234 -e WORDPRESS_DB_NAME=testdb --name wp -p 8888:80 --link mydb:db wordpress:latest

- docker container의 로그를 볼 수 있는 가장 쉬운 방법
  - docker logs container_id (혹은 이름)

데이터 저장

- 데이터는 유지될 것인가?
  - 생성된 이미지는 변경되지 않으며
  - 컨테이너는 휘발성
- Persistent 데이터를 사용하려면
  - 호스트 볼륨 공유
  - 볼륨 컨테이너
  - 도커 볼륨

컨테이너로 파일 copy

- docker cp filepath(local) container_name:path(in container)
  - docker cp cp.html web:/var/www/html
  - 호스트 → 컨테이너

- 반대방형 : 컨테이너에서 파일 copy
  - docker cp container_name:path(in container) path(local)
  - 컨테이너 → 호스트

호스트 볼륨 공유

- -v [호스트디렉토리]:[컨테이너디렉토리]
  - docker run -i -t -v ./temp/docker:/root ubuntu
    - 윈도우라면 -v C:/temp/docker:/root 이런 식으로
  - 앞의 mysql에 -v /mydir:/var/lib/mysql 을 추가한다면?
- 컨테이너를 삭제하더라도 데이터는 유지됨
- 새로운 컨테이너에 기존의 호스트 디렉토리를 공유하면?
  - 컨테이너의 디렉토리는 호스트 디렉토리로 overwrite

볼륨 컨테이너

- --volumes-from : 다른 컨테이너와 볼륨을 공유
  - docker run -i -t --name u1 -v /temp/docker:/root ubuntu
  - CTRL-P, Q 후
  - docker run -i -t --volumes-from u1 ubuntu

도커 볼륨 (1)

- 도커 자체에서 제공하는 볼륨 기능
- 볼륨 생성
  - docker volume create --name tutorial
- 볼륨 확인
  - docker volume ls
- 도커 볼륨 마운트 : -v
  - docker run -i -t -v tutorial:/root/ ubuntu
- 그냥 쓰는 -v : 볼륨을 자동으로 생성
  - docker run -i -t -v /root Ubuntu
  - docker container inspect containe이름 : 컨테이너 각종 정보 조사, 특히 볼륨도 (Mounts/Source)

도커 볼륨 (2)

- 도커 볼륨 삭제
  - docker volume rm 볼륨명
    - 실행중인 컨테이너에 연결된 경우는 삭제 안됨
  - docker volume prune
    - 모든 볼륨을 삭제하되, 단 한번도 파일이 담궈지지 않은 볼륨에 한하여
- --mount 옵션도 유사
  - --mount type=volume,source=tutorial,target=/root
  - --mount type=bind,source=C:/Temp/docker,target=/root
- -v는 여러개 쓸 수 있으며, 디렉토리 뿐 아니라 파일 단위로도 가능