도커 볼륨을 이용한 퍼시스턴스 스토리지

최 혁

기록 가능 레이어

- 컨테이너의 파일 시스템은 단일 디스크다. 이 디스크는 도커가 이미지 레이어와 기록 가능 레이어를 합쳐 만들고 컨테이너에 전달한 가상 파일 시스템이다.
- 기록 가능 레이어는 컨테이너와 같은 생애주기를 갖는다.
- 기록 가능 레이어는 컨테이너마다 다르다.
- 기록 기능 레이어는 copy-on-write 방법으로 읽기 전용 레이어의 파일을 수정할 수 있다.

컨테이너에서 이미지 레이어에 포함된 파일 수정하면?

`도커가 파일을 쓰기 가능 레이어로 복사 -> 쓰기 가능 레이어에서 파일 수정`

도커 볼륨을 사용하는 컨테이너 실행하기

- 도커 볼륨: 도커에서 스토리지를 다루는 단위(컨테이너를 위한 USB)
- 볼륨은 컨테이너와 독립적으로 존재하며 별도의 생명주기를 갖는다.
- 볼륨을 컨테이너에 연결하면 컨테이너 파일 시스템의 한 디렉터리가 된다.

컨테이너에서 볼륨을 사용하는 방법

1. 수동으로 직접 볼륨을 생성해 컨테이너에 연결
   `docker volume create [name]`
2. Dockerfile 스크립트에서 VOLUME instruction 문법 사용
   `VOLUME <target-directory>`

두 컨테이너가 한 볼륨을 공유하는 방법

```
# 이 컨테이너 실행 시 볼륨 생성
docker container run --name todo2 -d diamol/ch06-todo-list
docker container exec todo2 ls /data

# 이 컨테이너는 todo1의 볼륨을 공유한다.
docker container run -d --name t3 --volumes-from todo1 diamol/ch06-todo-list
docker container exec t3 ls /data
```

- 볼륨은 컨테이너 간 파일 공유보다는 업데이트 간 상태를 보존하기 위한 용도로 사용해야 한다.
- 따라서 이미지에서 정의하는 것보다는 명시적으로 관리하는 편이 더 낫다.
- 볼륨에 이름을 붙여 생성하고 업데이트 시 다른 컨테이너로 옮겨 연결하면 된다.

수동으로 볼륨 생성하여 연결

```
# 복사 대상 경로를 환경 변수로 정의한다.
target='/data'

# 데이터를 저장할 볼륨 생성
docker volume create todo-list

# 볼륨을 연결해 v1 애플리케이션을 실행
docker container run -d -p 8011:80 -v todo-list:$target --name todo-v1
diamol/ch06-todo-list

# v1 애플리케이션 삭제
docker container rm -f todo-v1

docker container run -d -p 8011:80 -v todo-list:$target --name todo-v2
diamol/ch06-todo-list:v2
```

볼륨 정리

- Dockerfile 스크립트의 VOLUME과 docker container 명령의 --volume 블래그는 별개 기능
- VOLUME instruction을 사용해 빌드된 이미지로 컨테이너를 만들 때 볼륨을 지정하지 않으면 항상 새로운 볼륨이 함께 생성됨
- 이미지를 만드는 입장에서 VOLUME 인스트럭션을 이미지 정의에 포함시켜 두는 것이 좋다.
- 사용자 입장에서는 별도의 이름을 붙여 만든 볼륨을 사용하는 것이 좋다.

파일 시스템 마운트를 사용하는 컨테이너 실행하기

- 바인드 마운트(bind mount)는 호스트 컴퓨터 파일 시스템의 디렉터리를 컨테이너 파일 시스템의 디렉터리로 만든다.
- 볼륨과 같이 컨테이너 입장에서는 평범한 디렉터리이나, 도커를 사용하는 입장에서는 컨테이너가 호스트 컴퓨터의 파일에 직접 접근할 수 있고, 그 반대도 가능하다.
- SSD, 네트워크 분산 스토리지까지 사용 가능하다.
- 바인드 마운트는 양방향으로 동작한다.
    - 컨테이너에서 만든 파일을 호스트 컴퓨터에서 수정 가능
    - 호스트에서 만든 파일도 컨테이너에서 수정 가능

```
source="$(pwd)/databases" && target='/data'

mkdir ./databases

# readonly 옵션을 주어 디렉터리를 읽기 전용으로 연결할 수도 있다.
docker container run --mount type=bind,source=$source,target=$target -d -p 8012:80
diamol/ch06-todo-list

curl http://localhost:8012

ls ./databases
```

파일 시스템 마운트의 한계점

- 컨테이너의 마운트 대상 디렉터리가 존재하고, 안에 파일이 있다면 원래 디렉터리의 내용은 숨겨지고 바인드 마운트의 원본 디렉터리가 이를 대체한다.
- 호스트 컴퓨터의 파일이 컨테이너에 이미 존재하는 디렉터리로 마운트하면 두 파일이 합쳐진다.
- 분산 파일 시스템을 마운트하면 지원하지 않는 동작이 있을 수 있다.(로컬 컴퓨터 운영체제의 파일 시스템과 분산 파일 시스템이 다른 경우가 많기 때문)

컨테이너의 파일 시스템은 어떻게 만들어지는가?

- `유니언 파일 시스템` : 도커가 다양한 출처로부터 모아 만든 단일 가상 디스크로 구성된 파일 시스템
- 컨테이너에서 실행되는 애플리케이션의 입장에서는 단일 디스크만을 볼 수 있지만, 컨테이너나 이미지를 생성해 사용하는 사용자는 여러 개의 이미지 레이어, 하나 이상의 볼륨 마운트나 바인드 마운트를 컨테이너에 연결할 수 있다.(기록 가능 레이어는 하나밖에 가질 수 없다)

best practice

- 기록 가능 레이어: 비용이 비싼 계산이나 네트워크를 통해 저장해야 하는 데이터의 캐싱 등 단기 저장에 적합하다.(컨테이너가 삭제되면 데이터가 유실되기 때문)
- 로컬 바인드 마운트: 호스트 컴퓨터와 컨테이너 간 데이터를 공유하기 위해 사용한다. 개발자의 로컬 컴퓨터에서 컨테이너로 소스 코드를 전달하기 위해 사용하면 로컬 컴퓨터에서 수정한 내용이 이미지 빌드 없이도 즉시 컨테이너로 전달될 수 있다.
- 분산 바인드 마운트: 네트워크 스토리지와 컨테이너 간에 데이터를 공유하기 위해 사용한다. 읽기 전용으로 설정 파일을 전달하거나 공유 캐시로 활용할 수 있으며 읽기 쓰기 가능으로 데이터를 저장해 동일 네트워크상의 모든 컨테이너나 컴퓨터와 데이터를 공유하는데 적합하다.
- 볼륨 마운트: 컨테이너와 도커 객체인 볼륨 간에 데이터를 공유하기 위해 사용된다. 컨테이너를 교체하는 방식으로 애플리케이션을 업데이트해도, 이전 버전 컨테이너의 데이터를 그대로 유지할 수 있다.
- 이미지 레이어: 이미지 레이어는 컨테이너의 초기 파일 시스템을 구성한다. 레이어는 읽기 전용이며 여러 컨테이너가 공유한다.

연습문제 풀이

```
# 이전 실습 삭제
docker rm -f $(docker ps -aq)

# diamol/ch06-lab 이미지로 컨테이너를 실행해 현재 등록된 할 일 확인
docker container run -d -p 80:80 diamol/ch06-lab

# (덮어씌울) 볼륨 생성
docker volume create ch06-lab

# 변수 설정
configSource="$(pwd)/solution"
configTarget='/app/config'
dataTarget='/new-data'

# 마운트와 볼륨 설정하여 실행
docker container run -d -p 81:80 --mount type=bind,source=$configSource,
target=$configTarget,readonly --volume ch06-lab:$dataTarget diamol/ch06-lab
```