

Contents

- Container layer(R/W layer)
- Volume을 이용하여 Todo app. 실행하기

Container layer(R/W layer)

앞서 우리는 도커가 사용하는 레이어 파일시스템에 대해 알아보았습니다.

컨테이너가 실행될 때마다 새로운 레이어(Container layer, R/W layer)가 생성되고, 컨테이너가 삭제될 경우 그 레이어의 내용은 사라지게 됩니다.

직접 한번 컨테이너를 생성해서 볼까요?

먼저 ubuntu 컨테이너를 하나 실행합니다.

```
ubuntu@ip-10-0-1-14:~$ docker run --name my-ubuntu --detach ubuntu bash -c "echo 'Hello Docker...' > /test.txt && tail -f /dev/null"
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
405f018f9d1d: Pull complete
Digest: sha256:b6b83d3c331794420340093eb706a6f152d9c1fa51b262d9bf34594887c2c7ac
Status: Downloaded newer image for ubuntu:latest
6f4a471389915ed1d2b47d7814899968a3f05aab34de2b62721a0ee694a38f70
```

명령어 : `docker run --name my-ubuntu --detach ubuntu bash -c "echo 'Hello Docker...' > /test.txt && tail -f /dev/null"`

"Hello Docker..."라는 문자열을 담은 txt파일(/test.txt)을 하나 만들고, 컨테이너를 running상태로 두기 위해서 `tail` 명령을 실행했습니다.

Docker & Kubernetes - [Hands-on] 03. Docker volumes

이제 우리가 만든 파일을 한 번 볼까요?

```
ubuntu@ip-10-0-1-14:~$ docker exec my-ubuntu cat /test.txt  
Hello Docker...
```

명령어 : `docker exec my-ubuntu cat /test.txt`

우리가 적어놓은 테스트문구("Hello Docker...")가 보일거예요.

컨테이너가 실행되는 동안 처리된 내용이기 때문에 Container layer에 이 내용이 기록되게 됩니다.

이제 같은 ubuntu 이미지를 이용해서 새로운 컨테이너를 실행하고, test.txt 파일이 있나 살펴봅시다.

```
ubuntu@ip-10-0-1-14:~$ docker run -it ubuntu ls /  
bin dev home lib32 libx32 mnt proc run srv tmp var  
boot etc lib lib64 media opt root sbin sys usr
```

명령어 : `docker run -it ubuntu ls /`

당연히 없겠죠... 왜 그럴까요? (◑,◑)

우리가 앞에서 알아본 레이어의 개념을 잘 떠올려 보세요.

Volume을 이용하여 Todo app. 실행하기

이제 도커 볼륨(Volume)을 이용해서 데이터를 유지하는 방법을 알아보겠습니다.

우리 샘플 애플리케이션(Todo List Manager)은 SQLite database를 사용하고 있습니다.

데이터는 `/etc/todos/todo.db`에 파일로 저장이 되고 있구요.

이제 도커 볼륨을 이용해서 데이터가 저장되는 위치를 host 머신의 경로로 바꿔보겠습니다.

먼저 도커 볼륨을 하나 생성합니다.

```
ubuntu@ip-10-0-1-14:~$ docker volume create todo-db  
todo-db
```

명령어 : `docker volume create todo-db`

생성된 볼륨을 확인하려면 아래 명령어를 사용하면 됩니다.

```
ubuntu@ip-10-0-1-14:~$ docker volume list  
DRIVER      VOLUME NAME  
local      todo-db
```

명령어 : `docker volume list` 또는 `docker volume ls`

Docker & Kubernetes - [Hands-on] 03. Docker volumes

그리고, 볼륨의 더 자세한 정보를 알아보려면 아래 명령어를 사용하면 됩니다.

```
ubuntu@ip-10-0-1-14:~$ docker volume inspect todo-db
[
  {
    "CreatedAt": "2022-06-26T08:49:50Z",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/todo-db/_data",
    "Name": "todo-db",
    "Options": {},
    "Scope": "local"
  }
]
```

명령어 : `docker volume inspect todo-db`

Mountpoint가 바로 실제 데이터가 저장되는 Host 머신의 위치입니다.

Docker & Kubernetes - [Hands-on] 03. Docker volumes

이제 방금 생성한 볼륨을 우리 애플리케이션의 데이터 저장경로로 마운트해서 실행해 보겠습니다.

- Private repository의 이미지를 사용할 경우 로그인(`docker login -u [USER-NAME]`)이 필요합니다.

```
ubuntu@ip-10-0-1-14:~$ docker run --detach --publish 3000:3000 --volume todo-db:/etc/todos --name my-todo-manager rogallo/todo-app:1.0.0
Unable to find image 'rogallo/todo-app:1.0.0' locally
1.0.0: Pulling from rogallo/todo-app
ddad3d7c1e96: Already exists
de915e575d22: Already exists
7150aa69525b: Already exists
d7aa47be044e: Already exists
e998ae3a37ac: Pull complete
2b5756c5faea: Pull complete
622ae33e24a6: Pull complete
Digest: sha256:146bff86564f7937dad94f018a5e801ad6cf7e0fc03be810e02ead6376fa3b05
Status: Downloaded newer image for rogallo/todo-app:1.0.0
116cbd4d4c201e480ef4f935976ea541ab7d1f83a0b42c801f84b06c7b69cd33
```

명령어 : `docker run --detach --publish 3000:3000 --volume todo-db:/etc/todos --name my-todo-manager [USER-NAME]/todo-app:1.0.0`
[USER-NAME]에는 여러분의 정보로 채워넣어 주세요.

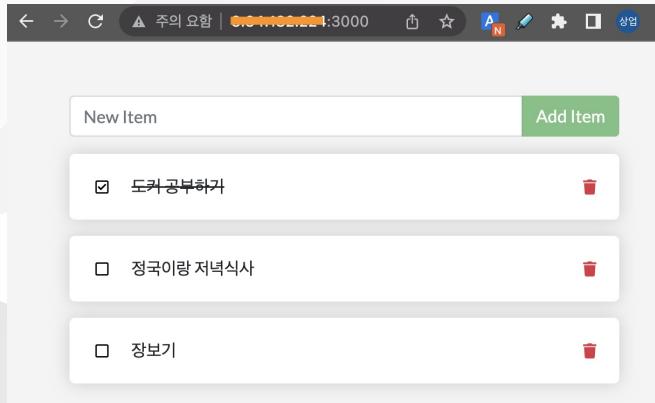
`--volume todo-db:/etc/todos`에서 콜론(:)을 구분자로 사용해서 첫 번째로는 volume의 이름을,
두 번째로는 마운트할 컨테이너의 경로를 적어줍니다.

또는 `--mount source=todo-db,target=/etc/todos`로 옵션을 설정해도 됩니다.

Docker & Kubernetes - [Hands-on] 03. Docker volumes

이제 실행된 애플리케이션에 접속하고 오늘 할 일을 몇 개 적어볼까요?

- AWS EC2인 경우 인스턴스의 Public IPv4 address로 접속하면 됩니다. (e.g. <http://IP:3000/>)
- Security group의 Inbound rule에 3000번 포트에 대한 규칙이 있어야 합니다.



그리고, 컨테이너를 멈추고 삭제합니다.

```
ubuntu@ip-10-0-1-14:~$ docker stop my-todo-manager
my-todo-manager
ubuntu@ip-10-0-1-14:~$ docker rm my-todo-manager
my-todo-manager
```

명령어 : `docker stop my-todo-manager` , `docker rm my-todo-manager`

- 컨테이너는 생성할때 `--name` 옵션으로 이름을 정하면, 이후에 이 이름을 이용할 수 있습니다.

Docker & Kubernetes - [Hands-on] 03. Docker volumes

이전 같으면(Volume을 사용하지 않았을 때는) 방금 저장한 할 일이 모두 사라지고 없겠죠?

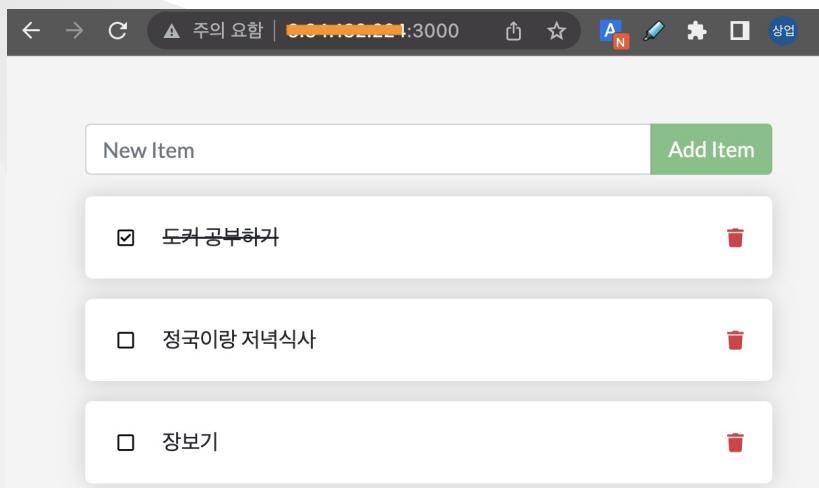
이제 다시한번 같은 명령어로 우리 애플리케이션을 실행해 볼까요?

```
ubuntu@ip-10-0-1-14:~$ docker run --detach --publish 3000:3000 --volume todo-db:/etc/todos --name my-todo-manager rogallo/todo-app:1.0.0  
6ace1a017e3478fec5352ebc7fff13c566f299bbd2a95406dd735fac340e4af6
```

명령어 : `docker run --detach --publish 3000:3000 --volume todo-db:/etc/todos --name my-todo-manager [USER-NAME]/todo-app:1.0.0`
[USER-NAME]에는 여러분의 정보로 채워넣어 주세요.

그리고 다시 우리 애플리케이션으로 접속해보세요. (e.g. <http://IP:3000/>)

어떤가요? 오늘 할 일 목록이 그대로 남아있나요? 정국이와 저녁식사도 장보기도 잊지않고 할 수 있게 되었습니다.



이제, 좀 제대로 된 애플리케이션이 된 것 같네요.... (ಠ_ಠ)