제 목	02장 Docker CLI 익히기	
상세내용	현업에서 자주 사용하는 Docker CLI 익히기	

1. 이미지(Image) 다운로드

[최신 버전(latest) 이미지 다운로드]

```
# docker pull 이미지명
$ docker pull nginx     # docker pull nginx:latest와 동일하게 작동
```

```
✘ jaeseong ~ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
24c63b8dcb66: Pull complete
ac894f1d1dfb: Pull complete
2572d4eb2260: Pull complete
0ac3805c647c: Pull complete
da20f09652a8: Pull complete
2de21a3abd85: Pull complete
77cea143f3c3: Pull complete
Digest: sha256:a484819eb60211f5299034ac80f6a681b06f89e65866ce91f356ed7c72af059c
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

이미지를 다운로드 할 때 **Dockerhub**이라는 곳에서 이미지를 다운 받는다.

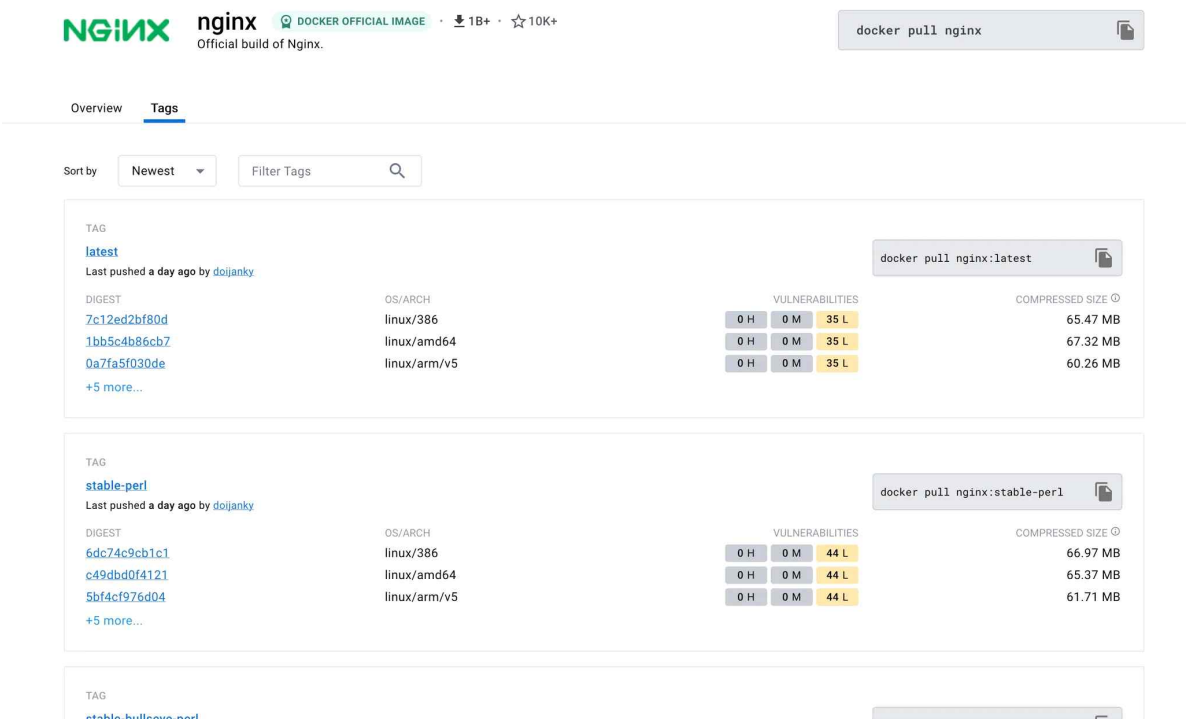
Github은 사람들이 올려놓은 다양한 코드들이 저장되어 있어서 clone, pull을 받아서 사용할 수 있다. Dockerhub도 마찬가지로 사람들이 올려놓은 이미지들이 저장되어 있어서 pull을 통해 다운받아서 사용할 수 있다.

Dockerhub은 Github처럼 **이미지를 저장 및 다운받을 수 있는 저장소 역할**을 하고 있다.

[특정 버전 이미지 다운로드]

```
# docker pull 이미지명:태그명
$ docker pull nginx:stable-perl
```

- **특정 버전을 나타내는 이름**을 **태그명**이라고 한다. **태그명**은 dockerhub에서 확인할 수 있다.



- https://hub.docker.com/_/nginx

2. 이미지(Image) 조회 / 삭제

✓ 다운받은 모든 이미지 조회

```
$ docker image ls
```

- `ls` : list의 약자
- `REPOSITORY` : 이미지 이름(이미지명)
- `TAG` : 이미지 태그명
- `IMAGE ID` : 이미지 ID
- `CREATED` : 이미지가 생성된 날짜 (다운받은 날짜 X)
- `SIZE` : 이미지 크기

✓ 이미지 삭제

[특정 이미지 삭제]

```
$ docker image rm [이미지 ID 또는 이미지명]
```

- rm : remove의 약자
- **이미지 ID**를 입력할 때 전체 ID를 다 입력하지 않고 ID의 일부만 입력해도 된다.

  (단, ID의 일부만 입력했을 때, 입력한 ID의 일부를 가진 이미지가 단 1개여야 한다.)
- 컨테이너에서 사용하고 있지 않은 이미지만 삭제가 가능하다

[중지된 컨테이너에서 사용하고 있는 이미지 강제 삭제하기]

```
$ docker image rm -f [이미지 ID 또는 이미지명]
```

- 실행 중인 컨테이너에서 사용하고 있는 이미지는 강제로 삭제할 수 없다.

[전체 이미지 삭제]

```
# 컨테이너에서 사용하고 있지 않은 이미지만 전체 삭제
$ docker image rm $(docker images -q)
```

```
# 컨테이너에서 사용하고 있는 이미지를 포함해서 전체 이미지 삭제
$ docker image rm -f $(docker images -q)
```

- docker images -q : 시스템에 있는 모든 이미지의 ID를 반환한다. 여기서 **-q** 옵션은 quite를 의미하며, 상세 정보 대신에 각 이미지의 고유한 ID만 표시하도록 지시한다.

3. 컨테이너(Container) 생성 / 실행 - 1

✓ 컨테이너 생성

> 이미지를 바탕으로 컨테이너를 **생성**한다. 이 때, 컨테이너를 **실행시키지는 않는다**.

(컨테이너를 실행하지 않고 생성만 하는 경우가 잘 없어서, 이 명령어는 잘 사용하지 않는다.)

```
# docker create 이미지명[:태그명]
$ docker create nginx

$ docker ps -a # 모든 컨테이너 조회
```

- 로컬 환경에 다운받은 이미지가 없다면 Dockerhub으로부터 이미지를 다운(**docker pull**)받아서 컨테이너를 생성한다.

```
jaeseong ~ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS    PORTS     NAMES
f76af5889cdb   nginx:latest   "/docker-entrypoint.…"   2 minutes ago   Created             elegant_mclaren
```

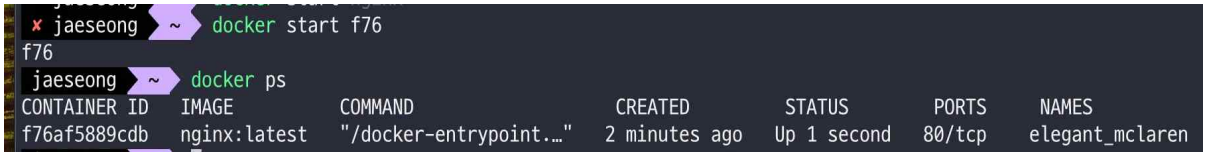
✓ 컨테이너 실행

> 정지되어 있는 컨테이너를 실행시킨다.

```
# docker start 컨테이너명[또는 컨테이너 ID]
$ docker start 컨테이너명[또는 컨테이너 ID]
```

```
$ docker ps # 실행중인 컨테이너 조회
```

```
# Nginx 컨테이너 중단 후 삭제하기
$ docker ps # 실행 중인 컨테이너 조회
$ docker stop {nginx를 실행시킨 Contnainer ID} # 컨테이너 중단
$ docker rm {nginx를 실행시킨 Contnainer ID} # 컨테이너 삭제
$ docker image rm nginx # Nginx 이미지 삭제
```



4. 컨테이너(Container) 생성 / 실행 - 2

✓ 컨테이너 생성 + 실행

> 이미지를 바탕으로 컨테이너를 생성한 뒤, 컨테이너를 실행까지 시킨다.

(처음에 이미지를 바탕으로 컨테이너를 실행시키고 싶을 때, 이 명령어를 자주 사용한다.)

```
# docker run 이미지명[:태그명]
$ docker run nginx # 포그라운드에서 실행 (추가적인 명령어 조작을 할 수가 없음)

# Ctrl + C로 종료할 수 있음
```

- 로컬 환경에 다운받은 이미지가 없다면 Dockerhub으로부터 이미지를 다운(docker pull)받아서 실행시킨다.
- Dockerhub으로부터 **새롭게 갱신된 이미지를 다운** 받고 싶다면 docker pull 명령어를 활용해야 한다.

[컨테이너를 백그라운드에서 실행시키기]

포그라운드(foreground)와 백그라운드(background)의 차이를 모르는 분들을 위해 간단히 정리하고 가자.

포그라운드는 **내가 실행시킨 프로그램의 내용이 화면에서 실행되고 출력되는 상태**를 뜻한다. 그러다보니 포그라운드 상태에서는 다른 프로그램을 조작할 수가 없다.

백그라운드는 **내가 실행시킨 프로그램이 컴퓨터 내부적으로 실행되는 상태**를 의미한다. 그래서 프로그램이 어떻게 실행되고 있는 지에 대한 정보를 화면에서 확인할 수 없다. 이런 특성 때문에 다른 명령어를 추가로 입력할 수도 있고, 새로운 프로그램을 조작할 수도 있다.

```
# docker run -d 이미지명[:태그명]
$ docker run -d nginx

# Nginx 컨테이너 중단 후 삭제하기
$ docker ps    # 실행 중인 컨테이너 조회
$ docker stop {nginx를 실행시킨 Contnainer ID}    # 컨테이너 중단
$ docker rm {nginx를 실행시킨 Contnainer ID}     # 컨테이너 삭제
$ docker image rm nginx     # Nginx 이미지 삭제
```

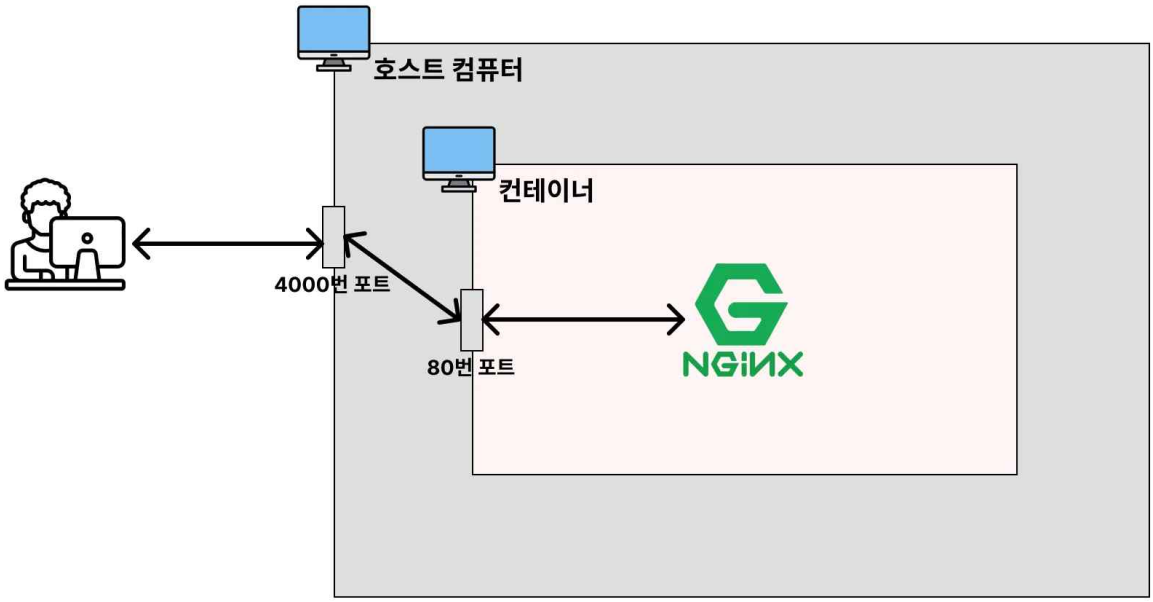
[컨테이너에 이름 붙여서 생성 및 실행하기]

```
# docker run -d --name [컨테이너 이름] 이미지명[:태그명]
$ docker run -d --name my-web-server nginx

# Nginx 컨테이너 중단 후 삭제하기
$ docker ps # 실행 중인 컨테이너 조회
$ docker stop {nginx를 실행시킨 Contnainer ID} # 컨테이너 중단
$ docker rm {nginx를 실행시킨 Contnainer ID} # 컨테이너 삭제
$ docker image rm nginx # Nginx 이미지 삭제
```

[호스트의 포트와 컨테이너의 포트를 연결하기]

```
# docker run -d -p [호스트 포트]:[컨테이너 포트] 이미지명[:태그명]
$ docker run -d -p 4000:80 nginx
```



- **docker run -p 4000:80** 라고 명령어를 입력하게 되면, 도커를 실행하는 호스트의 4000번 포트를 컨테이너의 80번 포트로 연결하도록 설정한다.

5. 컨테이너(Container) 조회 / 중지 / 삭제

✓ 컨테이너 조회

[실행 중인 컨테이너들만 조회]

```
$ docker ps
```

- ps : process status의 약자

[모든 컨테이너 조회 (작동 중인 컨테이너 + 작동을 멈춘 컨테이너)]

```
$ docker ps -a
```

- -a : all의 약자

✓ 컨테이너 중지

```
$ docker stop 컨테이너명[또는 컨테이너 ID]
$ docker kill 컨테이너명[또는 컨테이너 ID]
```

- 집에 있는 컴퓨터로 비유하자면 **stop**은 시스템 종료 버튼을 통해 정상적으로 컴퓨터를 종료하는 걸 의미하고, **kill**은 본체 버튼을 눌러 무식하게 종료하는 걸 의미한다.

✓ 컨테이너 삭제

[중지되어 있는 특정 컨테이너 삭제]

```
$ docker rm 컨테이너명[또는 컨테이너 ID]
```

- 실행 중인 컨테이너는 중지한 후에만 삭제가 가능하다.

[실행되고 있는 특정 컨테이너 삭제]

```
$ docker rm -f 컨테이너명[또는 컨테이너 ID]
```

[중지되어 있는 모든 컨테이너 삭제]

```
$ docker rm $(docker ps -qa)
```

[실행되고 있는 모든 컨테이너 삭제]

```
$ docker rm -f $(docker ps -qa)
```

6. 컨테이너(Container) 로그 조회

컨테이너를 실행시키고나서 실행시킨 컨테이너가 잘 실행되고 있는 지, 에러가 발생한 건 아닌 지 로그를 확인할 수 있어야 한다. 디버깅할 때 필수로 확인해야 하는 게 로그다. 지금부터 컨테이너에서 발생한 로그는 어떻게 확인하는 지 알아보자.

✓ 컨테이너(Container) 로그 조회

[실행되고 있는 특정 컨테이너 삭제]

```
# docker logs [컨테이너 ID 또는 컨테이너명]

$ docker run -d nginx
$ docker logs [nginx가 실행되고 있는 컨테이너 ID]
```

[최근 로그 10줄만 조회]

```
# dokcer logs --tail [로그 끝부터 표시할 줄 수] [컨테이너 ID 또는 컨테이너명]
$ dokcer logs --tail 10 [컨테이너 ID 또는 컨테이너명]
```

[기존 로그 조회 + 생성되는 로그를 실시간으로 보고 싶은 경우]

```
# docker logs -f [컨테이너 ID 또는 컨테이너명]

# Nginx의 컨테이너에 실시간으로 쌓이는 로그 확인하기
$ docker run -d -p 80:80 nginx
$ docker logs -f
```

- **-f** : follow의 약어

[기존 로그는 조회하지 않기 + 생성되는 로그를 실시간으로 보고 싶은 경우]

```
$ docker logs --tail 0 -f [컨테이너 ID 또는 컨테이너명]
```

7. 실행중인 컨테이너 내부에 접속하기 (exec -it)

✓ 컨테이너 개념 다시 짚어보기

> 위 설명에서 **컨테이너**는 **미니 컴퓨터**라고 표현했다. 즉, 호스트 컴퓨터 안에 다른 새로운 컴퓨터가 여러개 있는 것과 같다. 따라서 각각의 컨테이너는 자기만의 컴퓨터 공간(OS, 저장 공간, 프로그램 등)을 가지고 있다.

✓ 실행 중인 컨테이너 내부에 접속하기

```
# docker exec -it 컨테이너명[또는 컨테이너 ID] bash

$ docker run -d nginx
$ docker exec -it [Nginx가 실행되고 있는 컨테이너 ID] bash
$ ls # 컨테이너 내부 파일 조회
$ cd /etc/nginx
$ cat nginx.conf
```

- 컨테이너 내부에서 나오려면 Ctrl + D 또는 exit 을 입력하면 된다.
- bash : 쉘(Shell)의 일종
- -it : -it 옵션을 사용해야 명령어를 입력하고 결과를 확인할 수 있다. `-it` 옵션을 적지 않으면 명령어를 1번만 실행시키고 종료되어 버린다. 즉, `-it` 옵션을 적어야 계속해서 명령어를 입력할 수 있다.

[실습] Docker 전체 흐름 다시 느껴보기 (Nginx 설치 및 실행)

✓ Docker를 활용해 Nginx 실행시키기

1. Nginx 이미지 다운로드

```
$ docker pull nginx
```

2. 다운로드 된 이미지 확인하기

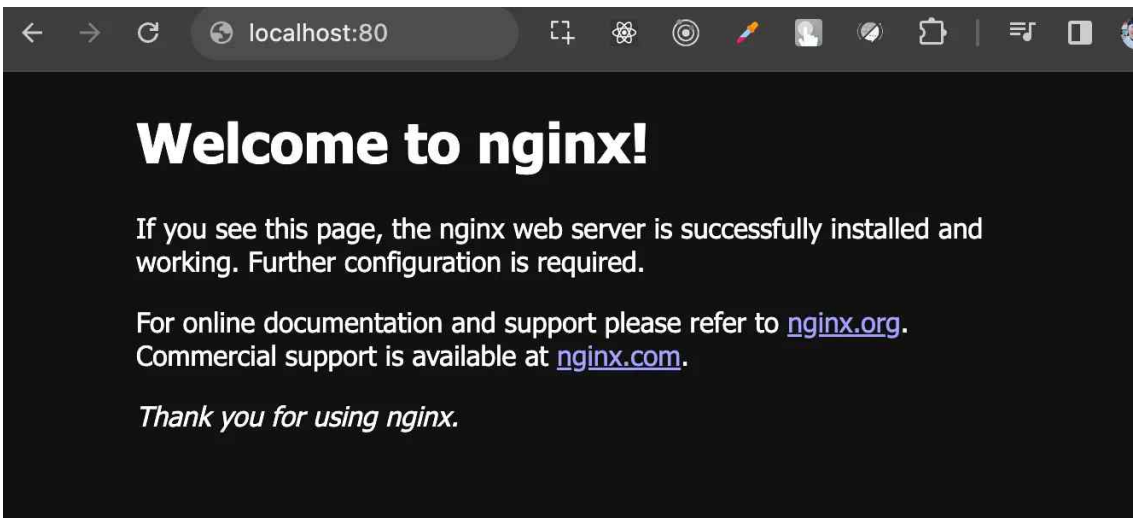
```
$ docker image ls
```

- ls : list의 약자

3. 이미지를 컨테이너에 올려 Nginx 서버 실행시키기

```
$ docker run --name webserver -d -p 80:80 nginx
```

4. Nginx 서버가 잘 실행되는 지 확인하기



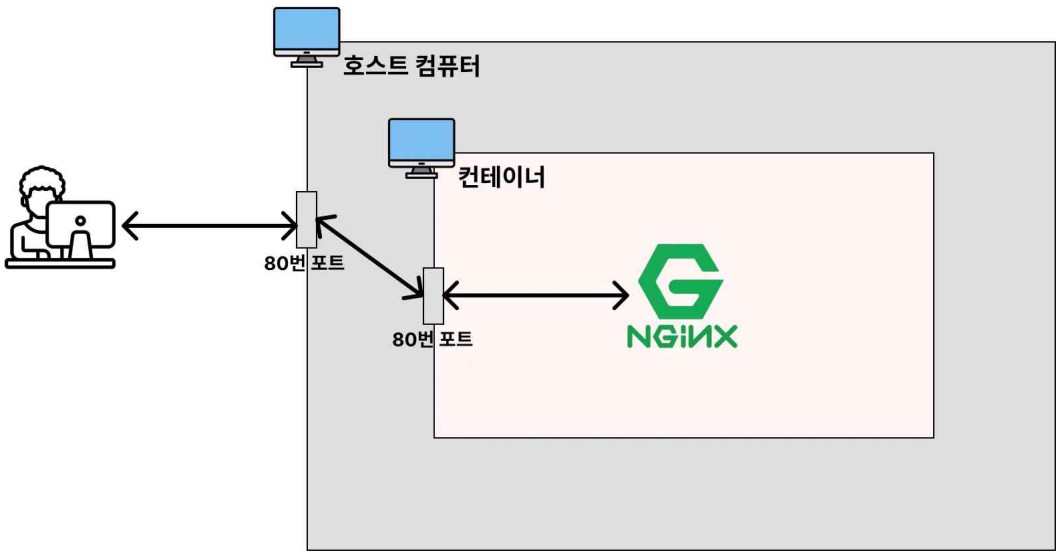
5. 실행되고 있는 모든 컨테이너 상태 확인하기

```
$ docker ps
```

6. 특정 컨테이너 정지

```
$ docker stop webserver
```

✓ 그림으로 이해하기



[실습] Docker로 Redis 실행시켜보기

✓ Docker로 Redis 실행시켜보기

1. Redis 이미지를 바탕으로 컨테이너 실행시키기

```
$ docker run -d -p 6379:6379 redis
```

- 로컬 환경에 redis 이미지가 없으면 Dockerhub으로부터 Redis 이미지를 자동으로 다운받는다.

2. Redis 이미지를 바탕으로 컨테이너 실행시키기

```
$ docker image ls
```

3. 컨테이너가 잘 실행되고 있는 지 체크

```
$ docker ps
```

4. 컨테이너 실행시킬 때 에러 없이 잘 실행됐는 지 로그 체크

```
$ docker logs [컨테이너 ID 또는 컨테이너명]
```

5. Redis 컨테이너에 접속

```
$ docker exec -it [컨테이너 ID 또는 컨테이너명] bash
```

6. 컨테이너에서 redis 사용해보기

```
$ redis-cli
```

127.0.0.1:6379> set 1 jscode

127.0.0.1:6379> get 1

```
root@a47e133c7583:/# mongosh
Current Mongosh Log ID: 64b2a17e78aabb1cbc71c66f
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.9.1
Using MongoDB:          6.0.6
Using Mongosh:          1.9.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-07-15T13:34:51.978+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2023-07-15T13:34:52.660+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2023-07-15T13:34:52.660+00:00: vm.max_map_count is too low
------

test>
```

```
root@04fbae116aca:/data# redis-cli
127.0.0.1:6379> set 1 jscode
OK
127.0.0.1:6379> get 1
"jscode"
```

✓ 그림으로 이해하기