

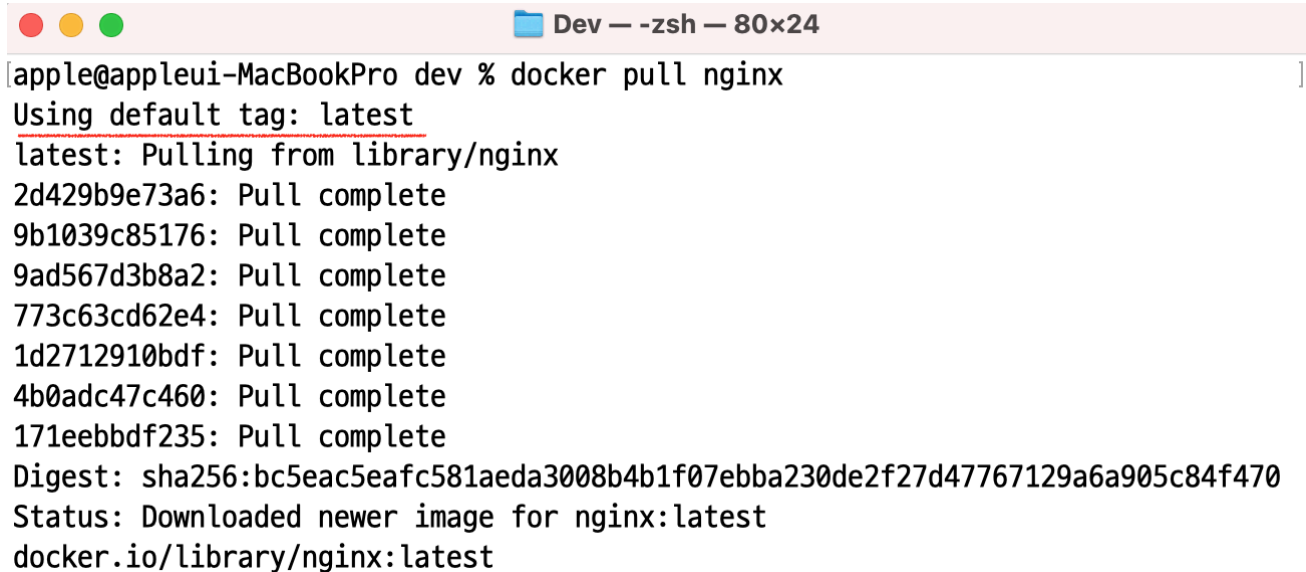
현업에서 자주 사용하는 **Docker CLI** 익히기

1. 이미지(Image) 다운로드

✅ 이미지 다운로드

[최신 버전(**latest**) 이미지 다운로드]

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



```
apple@appleui-MacBookPro dev % docker pull nginx  
Using default tag: latest  
latest: Pulling from library/nginx  
2d429b9e73a6: Pull complete  
9b1039c85176: Pull complete  
9ad567d3b8a2: Pull complete  
773c63cd62e4: Pull complete  
1d2712910bdf: Pull complete  
4b0adc47c460: Pull complete  
171eebbdf235: Pull complete  
Digest: sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470  
Status: Downloaded newer image for nginx:latest  
docker.io/library/nginx:latest
```

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

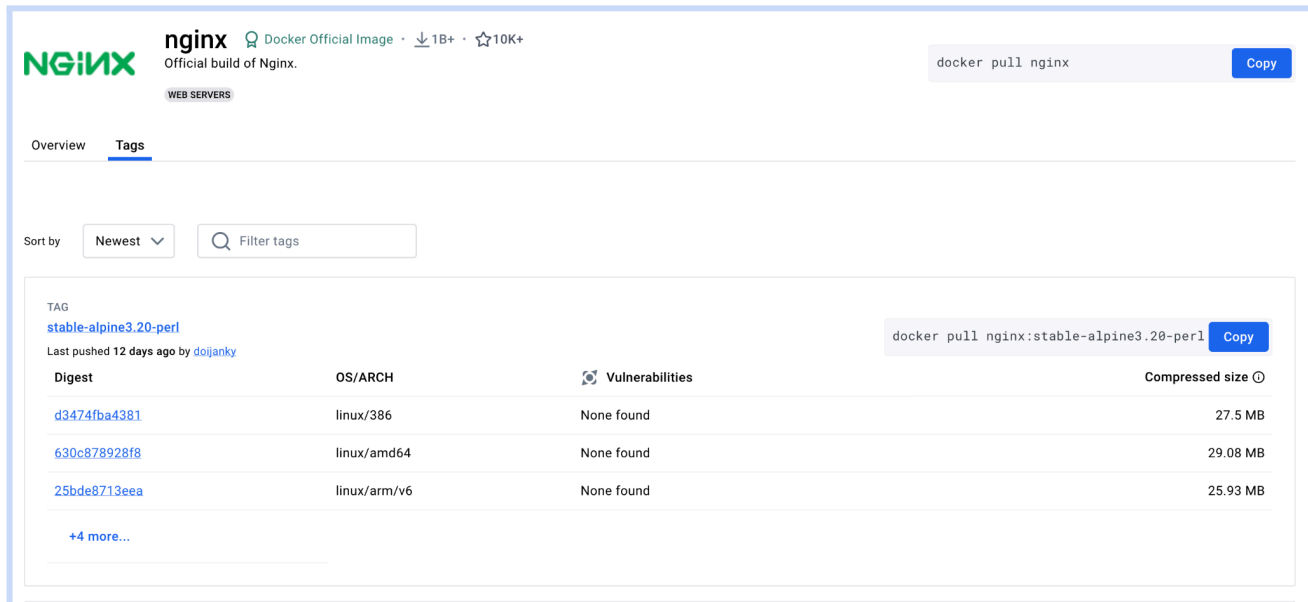
Github은 사람들이 올려놓은 다양한 코드들이 저장되어 있어서 **clone**, **pull**을 받아서 사용할 수 있다. **Dockerhub**도 마찬가지로 사람들이 올려놓은 이미지들이 저장되어 있어서 **pull**을 통해 다운받아서 사용할 수 있다.

Dockerhub은 Github처럼 이미지를 저장 및 다운받을 수 있는 저장소 역할을 하고 있다.

[특정 버전 이미지 다운로드]

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

- 특정 버전을 나타내는 이름을 **태그명**이라고 한다. **태그명**은 **dockerhub**에서 확인할 수 있다.



[nginx - Official Image | Docker Hub](#)

2. 이미지(Image) 조회 / 삭제

✓ 다운받은 모든 이미지 조회

```
$ docker image ls
```

```
apple@appleui-MacBookPro dev % ls
Android          flutter
apache-tomcat-10.1.11  lombok.jar
apple@appleui-MacBookPro dev % docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	60c8a892f36f	7 weeks ago	192MB
mariadb	latest	bc6434c28e9a	9 months ago	405MB
gvenzl/oracle-xe	21-full	376889d4ee76	12 months ago	6.7GB
alpine/git	latest	241890ad72b1	2 years ago	38.2MB
jaspeen/oracle-xe-11g	latest	52fbd1fe2d7a	9 years ago	792MB

- **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
```

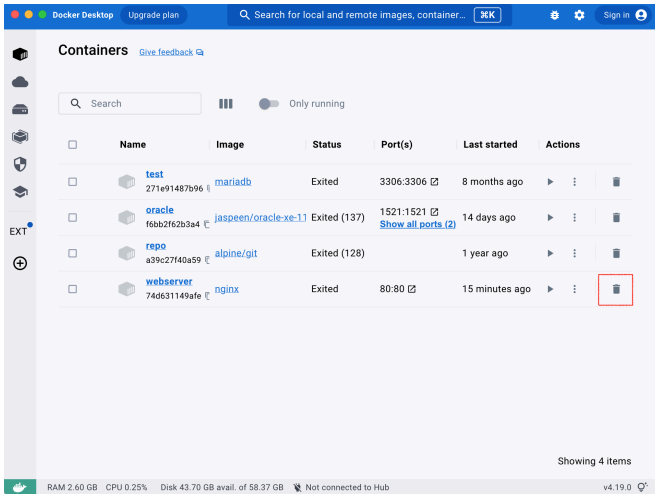
```
$ docker image rm $(docker images -q)
```

```
# 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 **quite**를 의미하며, 상세 정보 대신에 각 이미지의 고유한 ID만 표시하도록 지시한다.

데스크탑에서 삭제하기 - 이미지 리스트에는 남아 있음.



3. 컨테이너(Container) 생성 / 실행 - 1

✓ 컨테이너 생성

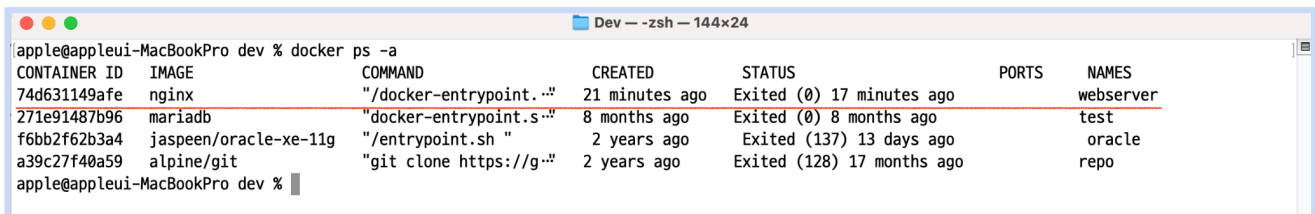
이미지를 바탕으로 컨테이너를 생성한다. 이 때, 컨테이너를 실행시키지는 않는다. (컨테이너를 실행하지 않고 생성만 하는 경우가 잘 없어서, 이 명령어는 잘 사용하지 않는다.)

```
# docker create 이미지명[:태그명]
```

```
$ docker create nginx
```

```
$ docker ps -a # 모든 컨테이너 조회
```

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



✓ 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 Container ID} # 컨테이너 중단

\$ docker rm {nginx를 실행시킨 Container ID} # 컨테이너 삭제

\$ docker image rm nginx # Nginx 이미지 삭제

```
apple@appleui-MacBookPro dev % docker start 74d
74d
apple@appleui-MacBookPro dev % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                NAMES
74d631149afe   nginx     "/docker-entrypoint. ..." 25 minutes ago Up 7 seconds    0.0.0.0:80->80/tcp    webserver
apple@appleui-MacBookPro dev %
apple@appleui-MacBookPro dev %
apple@appleui-MacBookPro dev %
```

삭제 확인

```
apple@appleui-MacBookPro dev % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                NAMES
74d631149afe   nginx     "/docker-entrypoint. ..." 26 minutes ago Up About a minute    0.0.0.0:80->80/tcp    webserver
apple@appleui-MacBookPro dev % docker stop 74d
74d
apple@appleui-MacBookPro dev % docker rm 74d
74d
apple@appleui-MacBookPro dev % docker image rm nginx
Untagged: nginx:latest
Untagged: nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
Deleted: sha256:60c8a892f36faf6c9215464005ee6fb8cf0585f70b113c0b030f6cb497a41876
Deleted: sha256:47984982982b32672d3b0cc6ebc1016e70916a8347c79765dc2ba09ed9afc97c
Deleted: sha256:f8fffe24ebb396c3e1721168923665f594d6b0ec1270700f642155fb51179cb
Deleted: sha256:ceff183e9da02c76af52712096cbe7e26e01909f827f18141058afb4f7e32db
Deleted: sha256:01c22c5216c94ae4a6285e21b0ccb6bb786d437aa7eb7d3e2de8a454115d17a8
Deleted: sha256:9a980991ece0116dad7650d5af48faa2f693f9277bfd99f4fb3c8c2ce0b4e27d
Deleted: sha256:d775439dbfb804d168b7ab8501c32013896d40d66b14944d2429778d995c7fe4
Deleted: sha256:c3548211b8264f8bfa47a6727043a64f1791b82ac965a284a7ea187e971a95e2
apple@appleui-MacBookPro dev % docker image ls
REPOSITORY      TAG       IMAGE ID       CREATED        SIZE
mariadb         latest    bc6434c28e9a   9 months ago   405MB
gvenzl/oracle-xe 21-full   376889d4ee76   12 months ago   6.7GB
alpine/git       latest    241890ad72b1   2 years ago    38.2MB
jaspeen/oracle-xe-11g latest    52fbd1fe2d7a   9 years ago    792MB
apple@appleui-MacBookPro dev %
```

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 상태를 뜻한다. 그러다보니 포그라운드 상태에서는 다른 프로그램을 조작할 수가 없다.

백그라운드는 내가 실행시킨 프로그램이 컴퓨터 내부적으로 실행되는 상태를 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 Container ID} # 컨테이너 중단  
$ docker rm {nginx를 실행시킨 Container ID} # 컨테이너 삭제  
$ docker image rm nginx # Nginx 이미지 삭제
```

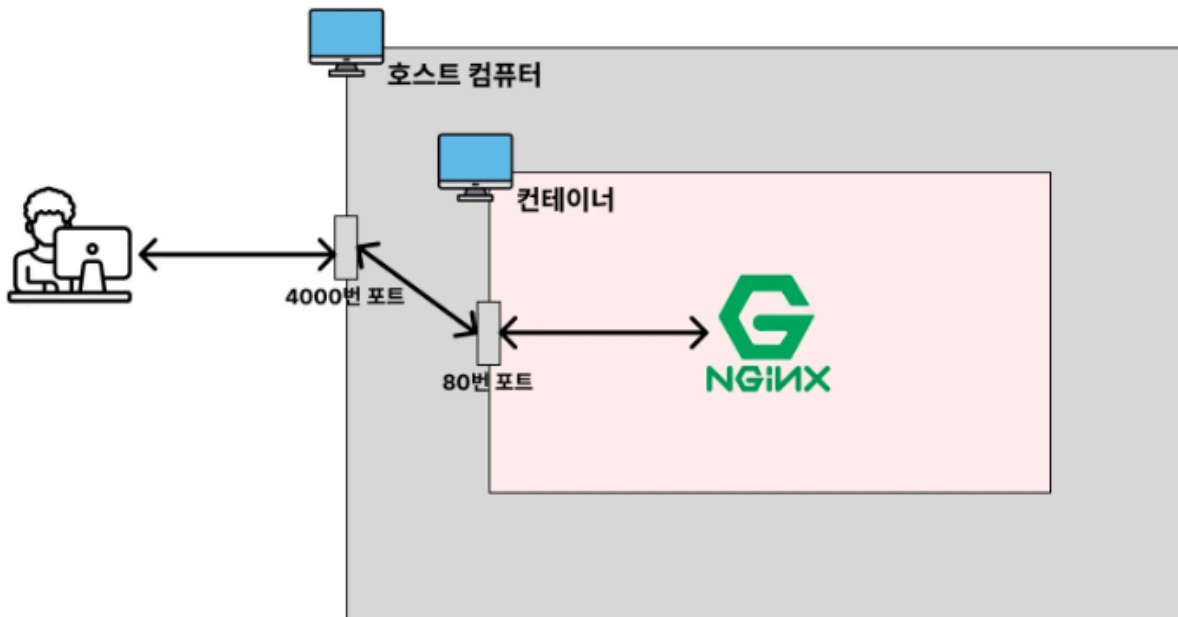
[컨테이너에 이름 붙여서 생성 및 실행하기]

```
# 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 Container ID} # 컨테이너 중단  
$ docker rm {nginx를 실행시킨 Container ID} # 컨테이너 삭제  
$ docker image rm nginx # Nginx 이미지 삭제
```

[호스트의 포트와 컨테이너의 포트를 연결하기]

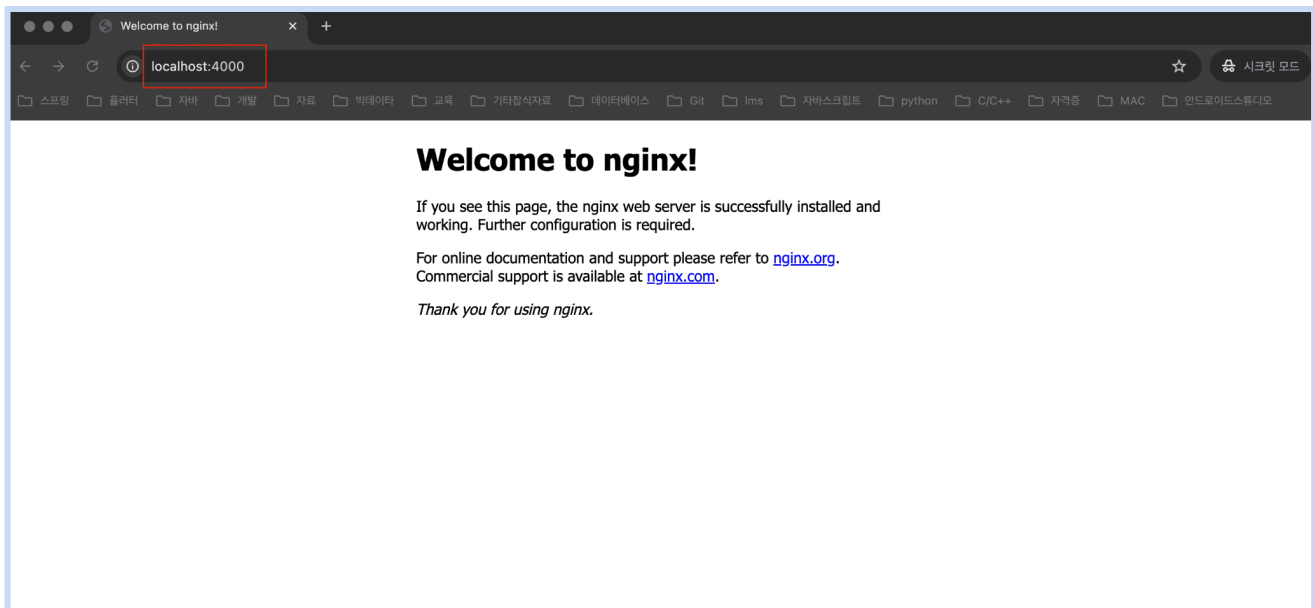
```
# docker run -d -p [호스트 포트]:[컨테이너 포트] 이미지명[:태그명]  
$ docker run -d -p 4000:80 nginx
```

```
apple@appleui-MacBookPro Docker % docker ps  
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS               NAMES  
79824c64d21b   nginx    "/docker-entrypoint. ..."  14 seconds ago Up 10 seconds    0.0.0.0:4000->80/tcp    goofy_gauss  
apple@appleui-MacBookPro Docker %
```



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

웹브라우저에서 실행해 보기 크롬 시크릿으로 실행 할 것. Nginx가 실행이 안되고 있는데 되는 것처럼 나올수 있으므로...



이렇게 외부에서도 들어갈 수 있음.

5. 컨테이너(Container) 조회 / 중지 / 삭제

✓ 컨테이너 조회

[실행 중인 컨테이너들만 조회]

```
$ docker ps
```

- **ps** : process status(정리)의 약자

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

```
$ docker ps -a
```

- **-a** : all의 약자

```
apple@appleui-MacBookPro Docker % docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
c77d4b583200	gvenzl/oracle-xe:21-full	"container-entrypoint..."	34 minutes ago	Exited (143) 31 minutes ago	
271e91487b96	mariadb	"docker-entrypoint.s..."	8 months ago	Exited (0) 8 months ago	
f6bb2f62b3a4	jaspeen/oracle-xe-11g	"/entrypoint.sh "	2 years ago	Exited (137) 2 weeks ago	
a39c27f40a59	alpine/git	"git clone https://g..."	2 years ago	Exited (128) 18 months ago	

✓ 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.

```
apple@appleui-MacBookPro Docker % docker run -d nginx
6a9539221428c380ccf5cbc74ba3f0ab581c2b0fbd06456d481340baa2b67bb5
apple@appleui-MacBookPro Docker % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
6a9539221428   nginx     "/docker-entrypoint...." 8 seconds ago  Up 7 seconds  80/tcp       exciting_ride
apple@appleui-MacBookPro Docker % docker rm 6a9
Error response from daemon: You cannot remove a running container 6a9539221428c380ccf5cbc74ba3f0ab581c2b0fbd06456d481340baa2b67bb5. Stop the container before attempting removal or force remove
```

[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)
```

여러 컨테이너 삭제 할때

```
$ docker rm -f 컨테이너ID 컨테이너ID
```

```
apple@appleui-MacBookPro Docker % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
8f96b721b63b   nginx     "/docker-entrypoint...." 8 minutes ago  Exited (0) 12 second
s ago         pensive_napier
051c39034975   nginx     "/docker-entrypoint...." 21 minutes ago  Up 21 minutes
80/tcp       infallible_gould
c77d4b583200   gvenzl/oracle-xe:21-full "container-entrypoint..." About an hour ago  Exited (143) About a
n hour ago   oracle_xe_21
271e91487b96   mariadb   "docker-entrypoint.s..." 8 months ago  Exited (0) 8 months
ago         test
f6bb2f62b3a4   jaspeen/oracle-xe-11g "/entrypoint.sh "      2 years ago  Exited (137) 2 week
s ago       oracle
a39c27f40a59   alpine/git "git clone https://g..." 2 years ago  Exited (128) 18 mont
hs ago      repo
apple@appleui-MacBookPro Docker % docker rm -f 8f9 051c
8f9
051c
apple@appleui-MacBookPro Docker % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
c77d4b583200   gvenzl/oracle-xe:21-full "container-entrypoint..." About an hour ago  Exited (143) About a
n hour ago   oracle_xe_21
271e91487b96   mariadb   "docker-entrypoint.s..." 8 months ago  Exited (0) 8 months
ago         test
f6bb2f62b3a4   jaspeen/oracle-xe-11g "/entrypoint.sh "      2 years ago  Exited (137) 2 week
s ago       oracle
a39c27f40a59   alpine/git "git clone https://g..." 2 years ago  Exited (128) 18 mont
hs ago      repo
```

6. 컨테이너(Container) 로그 조회

 컨테이너를 실행시키고 나서 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

[특정 컨테이너의 모든 로그 조회]

```
# docker logs [컨테이너 ID 또는 컨테이너명]

$ docker run -d nginx

$ docker logs [nginx가 실행되고 있는 컨테이너 ID]
```

```
apple@appleui-MacBookPro Docker % docker logs 051
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/11/28 07:44:34 [notice] 1#1: using the "epoll" event method
2024/11/28 07:44:34 [notice] 1#1: nginx/1.27.3
2024/11/28 07:44:34 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/11/28 07:44:34 [notice] 1#1: OS: Linux 5.15.49-linuxkit
2024/11/28 07:44:34 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/11/28 07:44:34 [notice] 1#1: start worker processes
2024/11/28 07:44:34 [notice] 1#1: start worker process 29
2024/11/28 07:44:34 [notice] 1#1: start worker process 30
2024/11/28 07:44:34 [notice] 1#1: start worker process 31
2024/11/28 07:44:34 [notice] 1#1: start worker process 32
```

[최근 로그 10줄만 조회]

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

$ docker logs --tail 10 [컨테이너 ID 또는 컨테이너명]
```

[기존 로그 조회 + 생성되는 로그를 실시간으로 보고 싶은 경우]

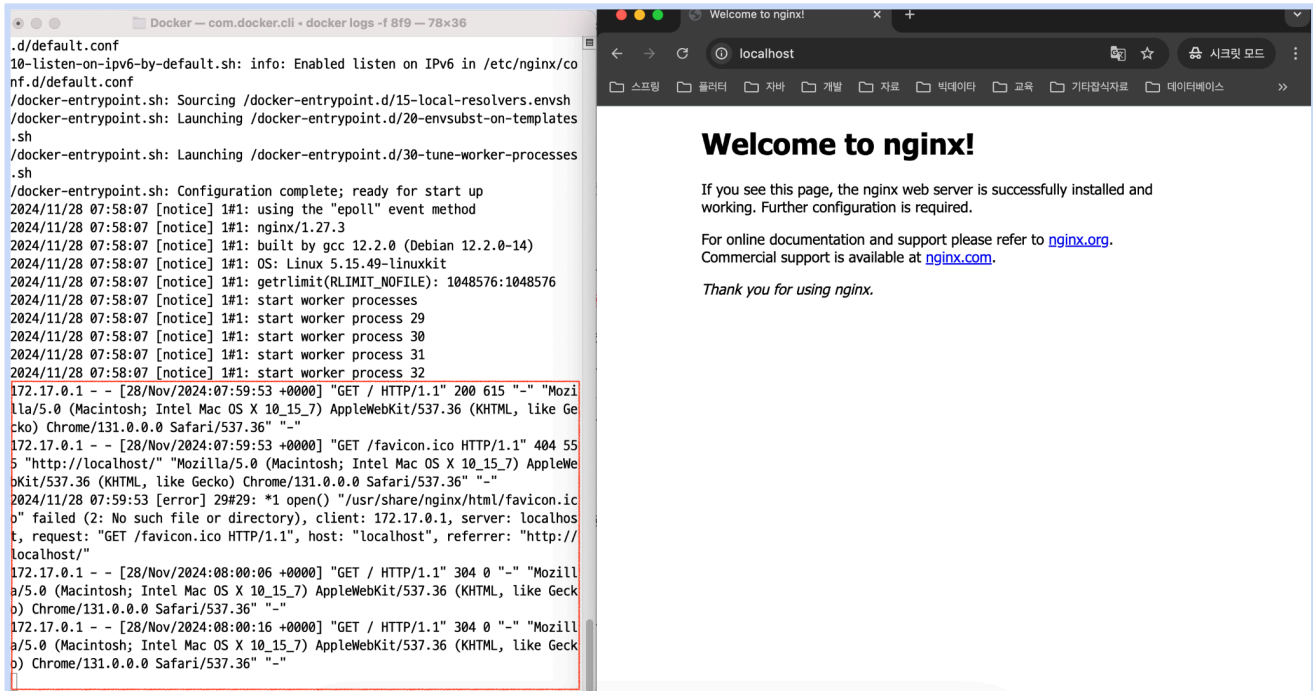
```
# docker logs -f [컨테이너 ID 또는 컨테이너명]
```

Nginx의 컨테이너에 실시간으로 쌓이는 로그 확인하기

\$ docker run -d -p 80:80 nginx

\$ docker logs -f 컨테이너 ID

- **-f** : follow의 약어



새로 고침 할때마다 로그가 생기는 것을 볼수 있다.

[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

컨테이너는 미니 컴퓨터라고 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
```

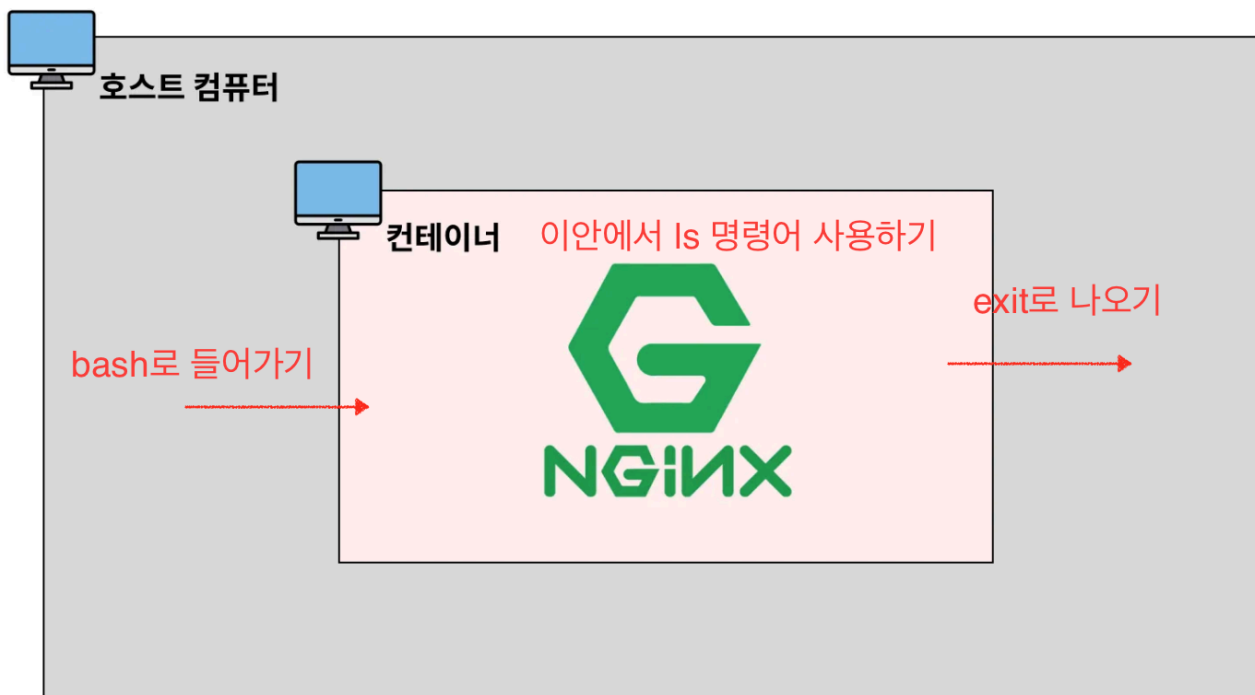
```
$ docker exec -it [Nginx가 실행되고 있는 컨테이너 ID] bash
```

```
$ ls # 컨테이너 내부 파일 조회
```

```
$ cd /etc/nginx
```

```
$ cat nginx.conf
```

- 컨테이너 내부에서 나오려면 **Ctrl + D** 또는 **exit**을 입력하면 된다.
- **bash** : 셸(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] Docker로 Redis 실행시켜보기

✓ Docker로 Redis 실행시켜보기

1. Redis 이미지를 바탕으로 컨테이너 실행시키기

[redis - Official Image | Docker Hub](#)

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

로컬 환경에 redis 이미지가 없으면 Dockerhub으로부터 Redis 이미지를 자동으로 다운받는다.

2. 다운로드 된 이미지 확인하기

```
$ docker image ls
```

- **ls** : list의 약자

3. 컨테이너가 잘 실행되고 있는 지 체크

```
$ docker ps
```

```
apple@appleui-MacBookPro Docker % docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
f04bd45ff6e8	redis	"docker-entrypoint.s..."	6 seconds ago	Up 5 seconds	0.0.0.0:6379->6379/tcp

```
apple@appleui-MacBookPro Docker % docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
postgres	latest	80cbdc6c3301	6 days ago	435MB
my-node-server	latest	6a4d9af17bce	7 days ago	1.12GB
mongo	latest	df3f01eba940	4 weeks ago	854MB
my-server	latest	85e7d404c34e	6 weeks ago	78.1MB
mysql	latest	56a8c14e1404	6 weeks ago	603MB
redis	latest	6c199afc1dae	7 weeks ago	117MB

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
```

```
apple@appleui-MacBookPro dev % docker exec -it b44 bash
root@b44bfc1f1efd:/data# redis-cli
127.0.0.1:6379> set 1 jscore
OK
127.0.0.1:6379> get 1
"jscore"
127.0.0.1:6379> 
```

bash끝내기 Exit

✓ 그림으로 이해하기

