# 그림과 실습으로 배우는 도커 & 쿠버네티스

## Chapter 4. 컨테이너를 실행해 보자

### 01. 도커 엔진 시작하기 / 종료하기

> 1) 도커 엔진 시작 → Docker Desktop 을 클릭
> 2) 도커 엔진 종료 → 화면 오른쪽 아래 도커 데스크톱 아이콘 클릭한 후 Quit Docker Desktop 선택(윈도우), 화면 오른쪽 위 도커(고래모양) 클릭한 후 Quit Docker Desktop 선택(macOs)
> 3) 자동 실행 설정 → 도커 테스크톱 아이콘 클릭한 후 Start Docker Desktop when you log in 에 체크(체크를 해제하면 비활성화)

### 02. 컨테이너의 기본적인 사용 방법

> 컨테이너를 다루는 모든 명령은 ‘docker 명령어’로 시작

```
docker ~

docker 커맨드 대상
docker 상위커맨드 하위커맨드 대상
ex) docker container run penguin

docker 상위커맨드 하위커맨드 옵션 대상 인자
ex) docker container run -d penguin --mode=1
-d 는 '백그라운드로 실행하라', --mode=1 은 '모드 1로 실행하라'
```

> docker 명령어의 기본적인 명령어의 형태는
> docker 상위커맨드 하위커맨드 (옵션) 대상 (인자)
> 여기서 상위커맨드는 ‘무엇을’, 하위커맨드는 ‘어떻게’ 할 것인지 지정하

> 는 부분
> 대상은 '구체적인 이름을 지정'

> 간단한 명령어를 사용하는 예시 → docker 의 버전을 알아보는 명령어

```
docker version
```

## 03. 컨테이너의 생성과 삭제, 실행, 정지

> 1) 도커 이미지를 내려받기 → docker image pull 대상
> 2) 도커 컨테이너를 생성 → docker container create 대상
> 3) 도커 컨테이너를 실행 → docker container start 대상
> 1, 2, 3 기능을 한꺼번에 수행 → docker container run 대상

> 도커 컨테이너를 삭제하려면 동작 중인 컨테이너를 그대로 삭제할 수 없음
> 도커 컨테이너를 정지시키고 삭제해야 함
> 1) 도커 컨테이너 정지 → docker container stop 대상
> 2) 도커 컨테이너 삭제 → docker container rm 대상

> 도커 컨테이너를 생성하고 실행하는 커맨드
> docker container run (옵션) 이미지(대상) (인자)

> 도커 컨테이너의 목록을 출력하는 커맨드
> docker container ls = docker ps = docker container ps

```
docker ps -a
-a 옵션은 현재 존재하는 컨테이너(정지 상태의 컨테이너 포함) 목록을 출력
```

> [실습] 컨테이너를 생성하고, 실행, 상태확인, 종료, 삭제해 보자

```
docker container run --name example -d httpd
--name example : example 이름으로 컨테이너를 생성
-d : 백그라운드로 실행
httpd : 아파치의 이미지 이름, 버전을 지정하지 않았으므로 가장 최신 버전(l
```

```
dogyeonglog@dogyeonglogs-MacBook-Air ~ % docker container run --name example -d httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
8dffd7e864fd: Download complete
4f4fb700ef54: Download complete
60e4ed7f50dd: Download complete
e219b11acd47: Download complete
1cfbb68d32bc: Download complete
7ce705000c39: Download complete
Digest: sha256:4c7788695c832bf415a662dfb5160f1895e65fc65c025e85f436ee2c9e7d7f3e
Status: Downloaded newer image for httpd:latest
a011d60e6a4d39ff282342cdcf4c76cc2dc281395e654709355576bee18e99bf
```

> 처음 실행하면 이미지를 내려받게 되므로 실행에 조금 시간이 걸림
> Unable to find image ‘httpd:latest’ locally 는 해당 이미지가 현재 컴퓨터에 없다는 뜻이고 조금 기다리면 해당 이미지를 다운받아 정상적으로 컨테이너가 실행됨

```
docker container ps
ps 커맨드를 사용해 컨테이너가 실행 중인지 확인
docker container ps -a
실행 중인 컨테이너 포함 및 정지 상태의 컨테이너 모두 확인
```

```
dogyeonglog@dogyeonglogs-MacBook-Air ~ % docker container ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS
 NAMES
a011d60e6a4d   httpd     "httpd-foreground"   3 minutes ago   Up 3 minutes   80/tcp
 example
```

```
docker container stop example
example 이름의 도커 컨테이너를 종료
```

> 해당 컨테이너 종료를 확인

```
dogyeonglog@dogyeonglogs-MacBook-Air ~ % docker container ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
dogyeonglog@dogyeonglogs-MacBook-Air ~ % docker container ps -a
CONTAINER ID   IMAGE                         COMMAND              CREATED         STATUS                     PORTS     NAMES
a011d60e6a4d   httpd                         "httpd-foreground"   11 minutes ago  Exited (0) 25 seconds ago            example
```

> 해당 컨테이너 삭제

```
docker container rm example
```

## 04. 컨테이너의 통신

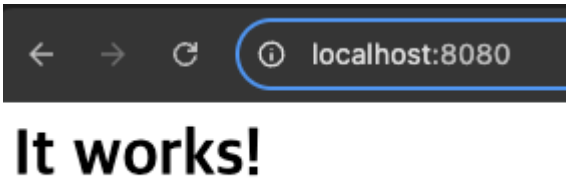
> 아파치는 웹 서버 기능을 제공하는 소프트웨어
> 쉽게 설명해서 아파치가 동작 중인 서버에 파일을 두면 이 파일을 웹 사이트 형태로 볼 수 있음
> 앞의 방법으로 컨테이너를 생성해서 실행하면 웹 사이트의 동작을 확인할 수 없음
> 왜냐하면 컨테이너가 컨테이너 외부에서 접근이 불가능한 상태로 실행되기 때문(컨테이너의 독립성)
> 따라서 옵션을 넣은 생성 명령을 통해 설정 ⇒ 이를 위해 포트를 설정

> 포트란 통신 내용이 드나드는 통로를 의미
> 아파치는 서버에서 정해둔 포트(80번 포트)에서 웹 사이트에 대한 접근을 기다리다가 사용자가 이 프로트를 통해 접근해 오면 요청에 따라 웹 사이트의 페이지를 제공
> 하지만 컨테이너 속에서 실행 중인 아파치는 외부와 직접 연결되지 않았기 때문에 외부에서 접근할 수 없음
> 그래서 컨테이너를 실행 중인 물리적 컴퓨터가 외부의 접근을 대신 받아 전달해주도록 함
> 좀 더 구체적으로 설명하면 컨테이너를 실행 중인 컴퓨터(호스트)의 8080번 포트(이 포트는 다른 소프트웨어가 사용하는 포트와 겹치지 않는 한 임의의 수자를 사용해도 됨)와 컨테이너의 80번 포트를 연결
> 이러한 설정이 -p 옵션이며, 그 뒤로 '호스트의 포트 번호'와 '컨테이너의 포트 번호'를 콜론으로 연결해 함께 기재
> -p 호스트_포트_번호:컨테이너_포트_번호
> ex) -p 8080:80

> 컨테이너를 사용하면 여러 개의 웹 서버를 함께 실행할 수도 있음
> 이러한 경우 호스트 포트 번호를 모두 같은 것으로 사용하면 어떤 컨테이너로 가야 할 요청인지 구분할 없음⇒ 컨테이너 A에서는 호스트 포트 8080, 컨테이너 B에는 호스트 포트 8081과 같은 식으로 호스트의 포트 번호를 겹치지 않게 설정
> ⇒ 꼭 여러 컨테이너로 연결되는 포트를 같게 설정하고 싶다면 리버스 프락시로 서버이름을 통해 구별하도록 구성한다

```
docker container run --name example -d -p 8080:80 httpd
물리적 컴퓨터(호스트)의 8080번 포트에서 접속시 해당 80 포트에 통신할 수 있
```

> 해당 컨테이너로 접속해보면(localhost:8080으로 접속한 결과)



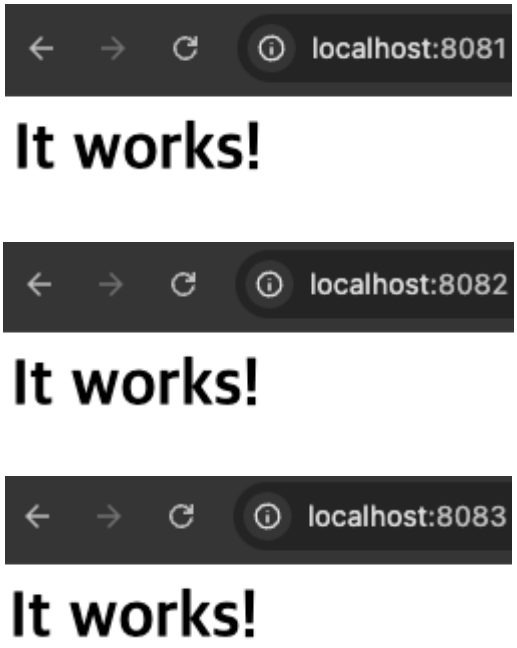

## 05. 컨테이너 생성에 익숙해지기

> 다양한 유형의 컨테이너를 생성해보며 연습해보자

> [실습] 아파치 컨테이너를 여러 개 실행하기

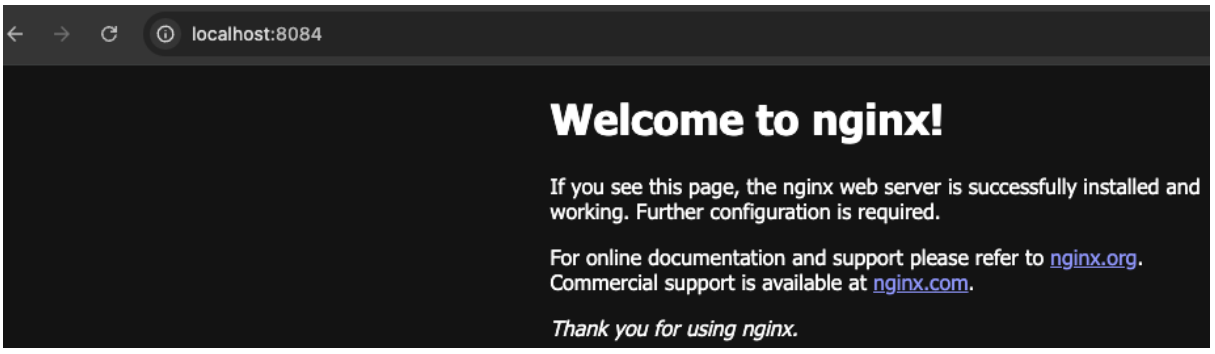
```
docker container run --name example1 -d -p 8081:80 httpd
docker container run --name example2 -d -p 8082:80 httpd
docker container run --name example3 -d -p 8083:80 httpd
```

```
dogyeonglog@dogyeonglogs-MacBook-Air ~ % docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED          STATUS          PORTS                   NAMES
62968627e139   httpd     "httpd-foreground"   2 seconds ago    Up 1 second     0.0.0.0:8083->80/tcp    example3
80a1d5594615   httpd     "httpd-foreground"   9 seconds ago    Up 9 seconds    0.0.0.0:8082->80/tcp    example2
0c16dc355af9   httpd     "httpd-foreground"   17 seconds ago   Up 16 seconds   0.0.0.0:8081->80/tcp    example1
```




> [실습] Nginx 컨테이너 실행하기

```
docker run --name nginx -d -p 8084:80 nginx
docker ps
```




> [실습] MySQL 컨테이너 실행하기

> 데이터베이스 서버용 컨테이너의 경우 기본적으로 루트 패스워드를 반드시 지정해야 한다

```
docker run --name mysql -dit -e MYSQL_ROOT_PASSWORD=exmple my
-dit : 백그라운드에서 실행 및 키보드를 통해 컨테이너 내부의 파일 시스템을
```

```
-e MYSQL_ROOT_PASSWORD : MySQL 의 로트 패스워들 지정
```

## 06. 이미지 삭제

> 컨테이너를 여러 번 만들다 보면 컨테이너를 삭제해도 이미지는 그대로 남아 쌓임
> 이미지가 늘어나면 스토리지 용량을 압박하게 되므로 필요없어진 이미지는 그때그때 삭제하도록 함
> 이미지를 삭제할 때는 이미지 ID 또는 이미지 이름으로 지정(컨테이너와 거의 비슷)
> 그리고 해당 이미지로 실행한 컨테이너가 남아 있으면 이미지를 삭제할 수 없으므로 docker ps -a 와 같이 컨테이너 목록을 출력해 확인하고 컨테이너를 먼저 종료 및 삭제한 다음, 이미지를 삭제

> 단일 이미지 삭제의 경우 ⇒ docker image rm 대상
> 여러 개의 이미지를 삭제하는 경우 ⇒ docker image rm 대상1 대상2 대상3 ...

> 이미지를 삭제하려면 이미지 이름 또는 이미지 ID 를 알아야 한다
> docker image ls → ps 와 달리 -a 옵션은 사용할 수 없음(이미지는 컨테이너와 달리 '실행 중', '종료' 등의 상태를 가질 수 없기 때문)

```
docker image ls
```

> 도커 이미지 삭제 시
> 1) 해당 컨테이너 종료
> 2) 해당 컨테이너 삭제
> 3) 이미지 삭제

```
docker image rm httpd nginx mysql
```