Lab4. Docker Container 생성 및 배포

1. Ubuntu기반 git 설치의 이미지 생성하기

```
$ mkdir demo
$ cd demo
$ docker system prune -a     <---Simply run to remove any stopped containers.
```

1)Dockerfile 생성

```
$ vim Dockerfile

    FROM ubuntu:latest

    RUN apt-get update
    RUN apt-get install -y git
```

2)Image Build

```
$ docker build -t ubuntu:git-dockerfile .
$ docker images
```

3)Container 생성하기

```
$ docker run -it --name git3 ubuntu:git-dockerfile bash
/# git --version
git version 2.34.1
```

2. Lab

1)Dockerfile 작성하기

```
$ mkdir sample
$ cd sample
$ vim dockerfile
    FROM centos:7
    COPY name.dat .
    CMD  cat ./name.dat

$ cat > name.dat
Hello, World
Ctrl + Z
$ cat name.dat
```

2)Dockerfile 빌드하기

```
$ docker build -t {{dockerhub 계정}}/dockerfiledemo:v1 .
$ docker images
```

3)Container 실행하기

```
$ docker run {{dockerhub 계정}}/dockerfiledemo:v1
Hello, World

$ docker ps -a
```

4)Dockerfile 수정

```
$ vim dockerfile
    FROM centos:7
    COPY name.dat .
    CMD  while true;  do sleep 3; cat ./name.dat;  done;

$ docker build -t {{dockerhub 계정}}/dockerfiledemo:v2 .
$ docker run {{dockerhub 계정}}/dockerfiledemo:v2
-3초마다 Hello, World 출력
```

5)또 다른 세션에서

```
$ docker ps -a
$ docker exec -it {{ContainerID}} bash
/# ls
/# cat name.dat
/# vi name.dat
Hello, Docker World!!!

/#exit
```

6)원래의 세션에서도 변경된 텍스트 출력확인

```
Hello, Docker World!!!
$ docker stop {{ContainerID}}
```

3. Lab

1)Dockerfile 생성하기

```
$ mkdir hellojs
$ cd hellojs
$ vim hello.js

    const http = require('http');

    const server = http.createServer();

    server.addListener('request', function(request, response) {
        console.log('requested...');
        response.writeHead(200, {'Content-Type' : 'text/plain'});
        response.write('Hello, nodejs!!!');
        response.end();
    });

    server.addListener('connection', function(socket){
        console.log('connected...');
    });

    server.listen(8888);

$ vi dockerfile
    FROM node:18     <---Docker Hub에서 검색해서 버전확인
    COPY hello.js /
    CMD ["node", "/hello.js"]

$ docker build -t hellojs:latest  .

$ docker images

$ docker run -d -p 8080:8888 --name web  hellojs
$ curl localhost:8080
```

2)Ubuntu 기반의 Web Server Container 만들기

-DockerHub에서 'httpd'로 검색

```
    $ mkdir webserver
    $ cd webserver
    $ nano dockerfile

        FROM ubuntu:latest
        LABEL maintainer="instructor <javaexpert@nate.com>"
```

```
#  Install Apache2
RUN apt update ₩
      && apt install -y apache2
RUN echo "<body><h1>Hello Apache2</h1></body>" > /var/www/html/index.html

EXPOSE 80
CMD ["/usr/sbin/apache2ctl", "-DFOREGROUND"]
```

```
$ docker build -t webserver:v1 .
$ docker image ls

$ docker run -d -p 80:80 --name web webserver:v1
$ curl localhost:80

$ docker rm -f web
$ docker ps -a
$ docker images
```

3)Container Image 배포하기

```
$ docker login
Username :
Password :

Login Succeeded
$ docker images

$ docker tag webserver:v1 {{dockerhub 계정}}/webserver:v1
$ docker images

$ docker push {{dockerhub 계정}}/webserver:v1
```

DockerHub/{{dockerhub 계정/repositories에서 확인할 것

```
$ cd ..
$ cd hellojs

$ docker tag hellojs {{dockerhub 계정}}/hellojs
$ docker images

$ docker push {{dockerhub 계정}}}/hellojs
```

DockerHub/{{dockerhub 계정}}/repositories에서 확인할 것