```
Lab4. Docker Container 생성 및 배포
 1
 2
    1. Ubuntu기반 qit 설치의 이미지 생성하기
       $ mkdir demo
 4
 5
       $ cd demo
 6
       $ docker system prune -a <---Simply run to remove any stopped containers.
 7
 8
       1)Dockerfile 생성
 9
          $ vim Dockerfile
10
11
             FROM ubuntu:latest
12
13
             RUN apt-get update
             RUN apt-get install -y git
14
15
16
       2)Image Build
          $ docker build -t ubuntu:git-dockerfile .
17
18
          $ docker images
19
       3)Container 생성하기
20
21
          $ docker run -it --name git3 ubuntu:git-dockerfile bash
          /# git --version
22
23
          git version 2.34.1
24
25
26
   2. Lab
       1)Dockerfile 작성하기
27
28
          $ mkdir sample
29
          $ cd sample
          $ vim dockerfile
30
             FROM centos:7
31
32
             COPY name.dat.
33
             CMD cat ./name.dat
34
35
          $ cat > name.dat
36
          Hello, World
          Ctrl + Z
37
          $ cat name.dat
38
39
40
       2)Dockerfile 빌드하기
41
42
          $ docker build -t {{dockerhub 계정}}/dockerfiledemo:v1.
43
          $ docker images
44
45
       3)Container 실행하기
46
          $ docker run {{dockerhub 계정}}/dockerfiledemo:v1
47
48
          Hello, World
49
50
          $ docker ps -a
51
52
53
       4)Dockerfile 수정
          $ vim dockerfile
54
55
             FROM centos:7
56
             COPY name.dat .
57
             CMD while true; do sleep 3; cat ./name.dat; done;
58
          $ docker build -t {{dockerhub 계정}}/dockerfiledemo:v2.
59
          $ docker run {{dockerhub 계정}}/dockerfiledemo:v2
60
          -3초마다 Hello, World 출력
61
```

```
62
 63
        5)또 다른 세션에서
 64
           $ docker ps -a
 65
           $ docker exec -it {{ContainerID}} bash
           /# Is
 66
 67
           /# cat name.dat
 68
           /# vi name.dat
 69
           Hello, Docker World!!!
 70
 71
           /#exit
 72
        6)원래의 세션에서도 변경된 텍스트 출력확인
 73
 74
           Hello, Docker World!!!
 75
           $ docker stop {{ContainerID}}
 76
 77
 78
     3. Lab
 79
        1)Dockerfile 생성하기
 80
           $ mkdir hellojs
           $ cd hellojs
 81
 82
           $ vim hello.js
 83
 84
              const http = require('http');
 85
 86
              const server = http.createServer();
 87
              server.addListener('request', function(request, response) {
 88
 89
                 console.log('requested...');
 90
                 response.writeHead(200, {'Content-Type': 'text/plain'});
 91
                 response.write('Hello, nodejs!!!');
 92
                 response.end();
 93
              });
 94
 95
              server.addListener('connection', function(socket){
 96
                 console.log('connected...');
 97
              });
 98
 99
              server.listen(8888);
100
101
           $ vi dockerfile
                                <---Docker Hub에서 검색해서 버전확인
              FROM node:18
102
103
              COPY hello.js /
104
              CMD ["node", "/hello.js"]
105
           $ docker build -t hellojs:latest .
106
107
           $ docker images
108
109
           $ docker run -d -p 8080:8888 --name web hellojs
110
           $ curl localhost:8080
111
112
113
        2)Ubuntu 기반의 Web Server Container 만들기
114
           -DockerHub에서 'httpd'로 검색
115
116
              $ mkdir webserver
117
              $ cd webserver
118
              $ nano dockerfile
119
120
121
                 FROM ubuntu:latest
122
                 LABEL maintainer="instructor <javaexpert@nate.com>"
```

```
123
124
                # Install Apache2
                RUN apt update ₩
125
126
                     && apt install -y apache2
127
                RUN echo "<body><h1>Hello Apache2</h1></body>" > /var/www/html/index.html
128
129
                EXPOSE 80
                CMD ["/usr/sbin/apache2ctl", "-DFOREGROUND"]
130
131
132
             $ docker build -t webserver:v1.
133
             $ docker image Is
134
135
             $ docker run -d -p 80:80 --name web webserver:v1
             $ curl localhost:80
136
137
             $ docker rm -f web
138
139
             $ docker ps -a
140
             $ docker images
141
142
143
        3)Container Image 배포하기
144
           $ docker login
145
           Username:
146
           Password:
147
           Login Succeeded
148
149
           $ docker images
150
151
           $ docker tag webserver:v1 {{dockerhub 계정}}/webserver:v1
152
           $ docker images
153
           $ docker push {{dockerhub 계정}}/webserver:v1
154
155
           DockerHub/{{dockerhub 계정/repositories에서 확인할 것
156
157
158
           $ cd ..
           $ cd hellojs
159
160
161
           $ docker tag hellojs {{dockerhub 계정}}/hellojs
162
           $ docker images
163
164
           $ docker push {{dockerhub 계정}}}/hellojs
165
           DockerHub/{{dockerhub 계정}}/repositories에서 확인할 것
166
```