```
1
    Lab. Nodejs Web Application Container
 2
    1. <a href="https://www.fastify.io/docs/latest/Guides/Getting-Started/">https://www.fastify.io/docs/latest/Guides/Getting-Started/</a> 접속
 4
 5
    2. 사전 Test
 7
        $ mkdir demo
 8
        $ cd demo
 9
        $ sudo apt install npm
10
        $ wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
        $ export NVM_DIR="$HOME/.nvm"
11
        $ [ -s "$NVM_DIR/nvm.sh" ] && ₩. "$NVM_DIR/nvm.sh" # This loads nvm
12
13
        $ [ -s "$NVM_DIR/bash_completion" ] && ₩. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
14
        $ nvm install node
15
        $ npm init
16
        $ npm i fastify --save
17
18
        $ vim app.js
19
           // Require the framework and instantiate it
20
21
           // CommonJs
           const fastify = require('fastify')({
22
23
            logger: true
24
25
           // Declare a route
26
27
           fastify.get('/', function (request, reply) {
28
            reply.send({ hello: 'world' })
29
           })
30
31
           // Run the server!
32
           fastify.listen(3000, '0.0.0.0', function (err, address) {
33
            if (err) {
34
              fastify.log.error(err)
35
              process.exit(1)
36
37
            fastify.log.info(`server listening on ${address}`)
38
           })
39
40
41
        $ node app.js
42
        -다른 세션에서 결과 확인
43
44
           $ curl localhost:3000
45
           {"hello":"world"}
46
47
48
    3. Docker Image 생성하기
49
        1)Dockerfile 생성하기
50
51
           $ vim Dockerfile
52
53
              FROM
                           node:14
54
              WORKDIR
                           /app
55
              COPY
                           package.json .
                           npm install
56
              RUN
57
              COPY
58
              EXPOSE
                           3000
59
60
              ENV
                           ADDRESS=0.0.0.0 PORT=3000
61
              CMD
                           ["node", "app.js"]
```

```
63
        2).dockerignore 파일 생성
 64
 65
            $ vim .dockerignore
              node_modules/*
 66
 67
 68
 69
        3)Docker Image 생성
 70
            $ docker build -t myweb .
 71
            $ docker images
 72
 73
            $ docker run -d -p 3000:3000 myweb
 74
            -Web Browser 확인
 75
           {"hello":"world"}
 76
 77
 78
            $ docker stop {{CONTAINER ID}}
 79
 80
        4)Source Code 수정
 81
 82
            -app.js
 83
              // Require the framework and instantiate it
 84
 85
               //CommonJs
 86
               const fastify = require('fastify')({
                logger: true
 87
 88
              })
 89
 90
              // Declare a route
 91
               fastify.get('/', function (request, reply) {
                                                                <---여기 수정
 92
                reply.send({ hello: 'docker world' })
 93
              })
 94
 95
              // Run the server!
 96
               fastify.listen(3000, '0.0.0.0', function (err, address) {
 97
                if (err) {
                 fastify.log.error(err)
 98
 99
                  process.exit(1)
100
101
                fastify.log.info('server listening on ${address}')
102
              })
103
104
105
        5)Build Again
           $ sudo docker build -t myweb .
106
107
           $ sudo docker images
108
109
            $ docker run -d -p 3001:3000 --name myweb myweb
110
            -Web Browser 확인
111
           {"hello":"docker world"}
112
113
114
115
     4. Dockerfile 최적화
116
        1)Dockerfile 수정
117
           FROM node:18 <----여기만 수정
118
119
120
        2)Docker Image build
121
122
            $ sudo docker build -t myweb.
```

```
124
        3)한번 build 하면 cache에 남아있기 때문에 소스코드가 변경되지 않으면 Cache 그냥 사용한다. 그래서 빨리 끝난다.
125
126
127
          $ sudo docker build -t myweb .
          $ sudo docker build -t myweb .
128
129
          $ sudo docker build -t myweb .
130
131
132
          ...
133
                                                  <---여기 주목
134
          => CACHED [2/5] WORKDIR /app
135
          => CACHED [3/5] COPY package.json .
          => CACHED [4/5] RUN npm install
136
          => CACHED [5/5] COPY . .
137
138
139
140
        4)Docker가 build할 때 코드가 바뀌지 않았다면 cache를 사용하고 코드가 수정됐다면 cache를 사용하지 않는다.
        5)Source Code 수정
141
142
          -app.js
143
             // Require the framework and instantiate it
144
145
             //CommonJs
             const fastify = require('fastify')({
146
              logger: true
147
148
             })
149
150
             // Declare a route
151
             fastify.get('/', function (request, reply) {
                                                   <---여기 수정
152
              reply.send({ hello: 'world' })
153
             })
154
155
             // Run the server!
             fastify.listen(3000, '0.0.0.0', function (err, address) {
156
157
              if (err) {
                fastify.log.error(err)
158
159
                process.exit(1)
160
161
              fastify.log.info(`server listening on ${address}`)
162
             })
163
164
        6)다시 build 해본다.
165
166
          •••
167
          •••
168
        7)build한 용량이 너무 크면 alpine 버전을 사용할 것
169
          -현재 하나의 이미지 용량이 거의 1.11GB
170
171
                         node:18-alpine
                                           <----여기만 수정
172
             FROM
173
174
          -거의 1/10으로 줄어듦 --> 147MB
```

123