

## 1 Lab. Nodejs Web Application Container

1. <https://www.fastify.io/docs/latest/Getting-Started/> 접속

### 2. 사전 Test

```
$ mkdir demo
$ cd demo
$ apt install npm
$ npm init
$ npm i fastify --save

$ vim app.js
// Require the framework and instantiate it
const fastify = require('fastify')({
  logger: true
})

// Declare a route
fastify.get('/', function (request, reply) {
  reply.send({ hello: 'world' })
})

// Run the server!
fastify.listen(3000, '0.0.0.0', function (err, address) {
  if (err) {
    fastify.log.error(err)
    process.exit(1)
  }
  fastify.log.info(`server listening on ${address}`)
})

$ node app.js
-Web Browser에서 확인
--http://{IP}:3000
```

### 3. Docker Image 생성하기

#### 1) app.js 수정

```
// Require the framework and instantiate it
const fastify = require('fastify')({
  logger: true
})

// Declare a route
fastify.get('/', function (request, reply) {
  reply.send({ hello: 'world' })
})

// Run the server!
fastify.listen(3000, '0.0.0.0', function (err, address) { <-----여기 수정
  if (err) {
    fastify.log.error(err)
    process.exit(1)
  }
  fastify.log.info(`server listening on ${address}`)
})
```

#### 2) Dockerfile 생성하기

```

61 # 1. nodejs 설치
62 FROM ubuntu:20.04
63 RUN apt-get update
64 RUN DEBIAN_FRONTEND=noninteractive apt-get -y install nodejs npm
65
66 # 2. Source File 복사
67 COPY . /usr/src/app
68
69 # 3. Nodejs Packages 설치
70 WORKDIR /usr/src/app
71 RUN npm install
72
73 # 4. Web Server 실행
74 EXPOSE 3000
75 CMD node app.js
76
77 3).dockerignore 파일 생성
78 node_modules/*
79
80 4) Docker Image 생성
81 $ docker build -t myweb .
82 $ docker images
83
84 $ docker run --rm -d -p 3000:3000 myweb
85
86 -Web Browser 확인
87
88 5) Source Code 수정
89 -app.js
90 // Require the framework and instantiate it
91 const fastify = require('fastify')({
92   logger: true
93 })
94
95 // Declare a route
96 fastify.get('/', function (request, reply) {
97   reply.send({ hello: 'docker world' })
98 })
99
100 // Run the server!
101 fastify.listen(3000, '0.0.0.0', function (err, address) {
102   if (err) {
103     fastify.log.error(err)
104     process.exit(1)
105   }
106   fastify.log.info(`server listening on ${address}`)
107 })
108
109 6)재 Build
110 $ docker build -t myweb .
111 $ docker images
112
113 $ docker run --rm -d -p 3000:3000 myweb
114
115 -Web Browser 확인
116
117
118 4. Dockerfile 최적화
119 1) Dockerfile 수정
120 # 1. nodejs 설치

```

<---여기 수정

121 FROM node:12 <---이미 Node가 설치되어 있는 이미지

122

123 # 2. Source File 복사

124 COPY . /usr/src/app

125

126 # 3. Nodejs Packages 설치

127 WORKDIR /usr/src/app

128 RUN npm install

129

130 # 4. Web Server 실행

131 EXPOSE 3000

132 CMD node app.js

133

134

135 2)한번 build 하면 cache에 남아있기 때문에 소스코드가 변경되지 않으면 Cache 그냥 사용한다. 그래서 빨리 끝난다.

136 3)지금 느려지는 부분은 바로 Nodejs Packages 설치의 npm install 부분이다.

137 4)이것을 최적화할 수 있는데, 먼저 패키지 설치하고 소스코드 복사하면 그만큼 속도가 더 빨라진다.

138

139 # 1. nodejs 설치

140 FROM node:12

141

142 # 2. 패키지 우선 복사

143 COPY ./package\* /usr/src/app/

144 WORKDIR /usr/src/app

145 RUN npm install

146 COPY . /usr/src/app

147

148 # 3. 소스코드 복사

149 COPY . /usr/src/app

150

151 # 4. Web Server 실행

152 EXPOSE 3000

153 CMD node app.js

154

155 5)npm install까지 cache되어서 빌드가 빨라짐

156

157 6)용량이 너무 크면 alpine 버전을 사용할 것

158 -현재 하나의 이미지 용량이 거의 924MB

159 # 1. nodejs 설치

160 FROM node:12-alpine

161

162 -거의 1/10으로 줄어듦.