

```

1 Lab. Container Network
2
3 1. Container Network 사용하기
4 1)docker0 사용 확인하기
5 $ ip addr
6 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
7 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
8 inet 127.0.0.1/8 scope host lo
9 valid_lft forever preferred_lft forever
10 inet6 ::1/128 scope host
11 valid_lft forever preferred_lft forever
12 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
13 link/ether 02:6b:81:64:e1:32 brd ff:ff:ff:ff:ff:ff
14 inet 10.0.10.23/24 metric 100 brd 10.0.10.255 scope global dynamic eth0
15 valid_lft 1923sec preferred_lft 1923sec
16 inet6 fe80::6b:81ff:fe64:e132/64 scope link
17 valid_lft forever preferred_lft forever
18 3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
19 link/ether 02:42:69:d8:e7:3d brd ff:ff:ff:ff:ff:ff
20 inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
21 valid_lft forever preferred_lft forever
22 inet6 fe80::42:69ff:fed8:e73d/64 scope link
23 valid_lft forever preferred_lft forever
24
25 $ sudo brctl show
26 bridge name bridge id STP enabled interfaces
27 docker0 8000.024269d8e73d no
28
29 $ sudo docker run --name busybox -it busybox
30 /# ifconfig
31 eth0 Link encap:Ethernet HWaddr 02:42:AC:11:00:02
32 inet addr:172.17.0.2 Bcast:172.17.255.255 Mask:255.255.0.0
33 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
34 RX packets:10 errors:0 dropped:0 overruns:0 frame:0
35 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
36 collisions:0 txqueuelen:0
37 RX bytes:876 (876.0 B) TX bytes:0 (0.0 B)
38
39 lo Link encap:Local Loopback
40 inet addr:127.0.0.1 Mask:255.0.0.0
41 UP LOOPBACK RUNNING MTU:65536 Metric:1
42 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
43 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
44 collisions:0 txqueuelen:1000
45 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
46
47 /# ping -c 4 8.8.8.8 <----외부 통신 가능 확인
48
49
50 2)자동으로 172.17.0.x의 IP Address 부여 확인하기
51 -다른 세션을 열어서
52 $ sudo docker run --name busybox1 -it busybox
53 /# ifconfig
54 eth0 Link encap:Ethernet HWaddr 02:42:AC:11:00:02
55 inet addr:172.17.0.3 Bcast:172.17.255.255 Mask:255.255.0.0
56 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
57 RX packets:9 errors:0 dropped:0 overruns:0 frame:0
58 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
59 collisions:0 txqueuelen:0
60 RX bytes:806 (806.0 B) TX bytes:0 (0.0 B)
61
62
63 -또 다른 세션을 열어서
64 $ sudo docker run -d -p 80:80 --name web nginx
65 $ sudo docker inspect web
66 $ curl 172.17.0.4
67 $ sudo iptables -t nat -L -v
68 Chain PREROUTING (policy ACCEPT 1 packets, 84 bytes)
69 pkts bytes target prot opt in out source destination
70 815 42348 DOCKER all -- any any anywhere anywhere ADDRTYPE match dst-type LOCAL
71
72 Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
73 pkts bytes target prot opt in out source destination
74
75 Chain OUTPUT (policy ACCEPT 9 packets, 1174 bytes)
76 pkts bytes target prot opt in out source destination
77 0 0 DOCKER all -- any any anywhere !localhost/8 ADDRTYPE match dst-type LOCAL
78
79 Chain POSTROUTING (policy ACCEPT 9 packets, 1174 bytes)
80 pkts bytes target prot opt in out source destination
81 1513 93953 MASQUERADE all -- any !docker0 172.17.0.0/16 anywhere
82 0 0 MASQUERADE tcp -- any any 172.17.0.4 172.17.0.4 tcp dpt:http
83
84 Chain DOCKER (2 references)

```

```

85         pkts bytes target   prot opt in     out     source            destination
86         0    0 RETURN  all  --  docker0 any    anywhere          anywhere
87         0    0 DNAT    tcp  --  !docker0 any    anywhere          anywhere
88
89
90
91 2. Port-Forwarding
92 1)host의 port와 container의 port 지정해서 연결하기
93   $ sudo docker run -p 80:80 -d --name web1 nginx
94
95   $ sudo docker ps
96   CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
97   NAMES
98   05c359f8bcd6   nginx    "/docker-entrypoint...." About a minute ago Up About a minute  0.0.0.0:80->80/tcp,
99   :::80->80/tcp   web1
100
101   $ curl localhost:80
102
103 2)host의 port를 랜덤으로 연결하기
104   $ sudo docker run -p 80 -d --name web2 nginx
105   $ sudo docker ps
106   CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
107   NAMES
108   8e4372270de9   nginx    "/docker-entrypoint...." 6 seconds ago Up 5 seconds      0.0.0.0:49153->80/tcp,
109   :::49153->80/tcp web2
110   05c359f8bcd6   nginx    "/docker-entrypoint...." About a minute ago Up About a minute  0.0.0.0:80->80/tcp,
111   :::80->80/tcp   web1
112
113   $ curl localhost:49153
114
115 3)host와 container 모두 자동으로 연결하기
116   $ sudo docker run -P(대문자) 80 -d --name web3 nginx
117   $ sudo docker ps -a
118   CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
119   NAMES
120   8ae0560aa57c   nginx    "/docker-entrypoint...." 3 seconds ago Up 2 seconds      0.0.0.0:49154->80/tcp,
121   :::49154->80/tcp web3
122   8e4372270de9   nginx    "/docker-entrypoint...." 3 minutes ago Up 3 minutes      0.0.0.0:49153->80/tcp,
123   :::49153->80/tcp web2
124   05c359f8bcd6   nginx    "/docker-entrypoint...." 4 minutes ago Up 4 minutes      0.0.0.0:80->80/tcp, :::80->80/tcp
125   web1
126
127
128
129
130
131
132
133
134
135
136
137
138
139 3. user-defined network 구성하기
140 1)기본 bridge외에 새로 생성하기
141   $ sudo docker network ls
142   NETWORK ID     NAME      DRIVER    SCOPE
143   32ce6dec4771   bridge   bridge    local
144   ef8f1c31a15d   host     host      local
145   ee449dfed7eb   none     null      local
146
147   $ sudo docker network create --driver bridge --subnet 192.168.100.0/24 \
148   > --gateway 192.168.100.254 mynet
149   df7b218797e7216e1b39549a94ab9b0b2b5d2946be63233ed8ac1b17a62742c6
150
151   $ sudo docker network ls
152   NETWORK ID     NAME      DRIVER    SCOPE
153   32ce6dec4771   bridge   bridge    local
154   ef8f1c31a15d   host     host      local
155   df7b218797e7   mynet    bridge    local
156   ee449dfed7eb   none     null      local
157
158 2)새로 생성한 bridge로 Container 생성하기
159   $ sudo docker run -it --name busybox1 --net mynet busybox
160   / # ifconfig
161   eth0      Link encap:Ethernet HWaddr 02:42:C0:A8:64:01
162             inet addr:192.168.100.1 Bcast:192.168.100.255 Mask:255.255.255.0
163             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
164             RX packets:14 errors:0 dropped:0 overruns:0 frame:0
165             TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
166             collisions:0 txqueuelen:0
167             RX bytes:1252 (1.2 KiB) TX bytes:0 (0.0 B)
168
169   lo        Link encap:Local Loopback
170             inet addr:127.0.0.1 Mask:255.0.0.0
171             UP LOOPBACK RUNNING MTU:65536 Metric:1
172             RX packets:0 errors:0 dropped:0 overruns:0 frame:0
173             TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
174             collisions:0 txqueuelen:1000
175             RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
176
177   /# exit
178
179   $ sudo docker inspect mynet

```

```

161 [
162   {
163     "Name": "mynet",
164     "Id": "df7b218797e7216e1b39549a94ab9b0b2b5d2946be63233ed8ac1b17a62742c6",
165     "Created": "2021-10-21T08:59:00.152346729Z",
166     "Scope": "local",
167     "Driver": "bridge",
168     "EnableIPv6": false,
169     "IPAM": {
170       "Driver": "default",
171       "Options": {},
172       "Config": [
173         {
174           "Subnet": "192.168.100.0/24",
175           "Gateway": "192.168.100.254"
176         }
177       ]
178     },
179     "Internal": false,
180     "Attachable": false,
181     "Ingress": false,
182     "ConfigFrom": {
183       "Network": ""
184     },
185     "ConfigOnly": false,
186     "Containers": {},
187     "Options": {},
188     "Labels": {}
189   }
190 ]

```

### 3) Container 생성시 ip 지정하기

```

194 $ sudo docker run -it --name busybox2 --net mynet --ip 192.168.100.100 busybox
195 / # ifconfig
196 eth0      Link encap:Ethernet  HWaddr 02:42:C0:A8:64:64
197           inet addr:192.168.100.100  Bcast:192.168.100.255  Mask:255.255.255.0
198           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
199           RX packets:8 errors:0 dropped:0 overruns:0 frame:0
200           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
201           collisions:0 txqueuelen:0
202           RX bytes:736 (736.0 B)  TX bytes:0 (0.0 B)

203
204 lo        Link encap:Local Loopback
205           inet addr:127.0.0.1  Mask:255.0.0.0
206           UP LOOPBACK RUNNING  MTU:65536  Metric:1
207           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
208           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
209           collisions:0 txqueuelen:1000
210           RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

211
212 / # ping -c 4 8.8.8.8

```

## 4. Container 간 통신하기

### 1) 첫번째 방법

#### -MySQL 실행하기

```

219 $ docker run -d -p 3306:3306 \
220 > -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
221 > --name mysql \
222 > mysql:5.7

223
224 $ docker exec -it mysql mysql
225 mysql> CREATE DATABASE wp CHARACTER SET utf8;
226 mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
227 mysql> FLUSH PRIVILEGES;
228 mysql> show databases;
229 +-----+
230 | Database |
231 +-----+
232 | information_schema |
233 | mysql |
234 | performance_schema |
235 | sys |
236 | wp |
237 +-----+
238 5 rows in set (0.00 sec)
239 mysql> quit

```

#### -WordPress 실행하기

```

242 $ docker run -d -p 8080:80 \
243 > -e WORDPRESS_DB_HOST=host.docker.internal \ <---Linux에서는 연결안됨. WSL만 가능
244 > -e WORDPRESS_DB_NAME=wp \

```

```
245 > -e WORDPRESS_DB_USER=wp \
246 > -e WORDPRESS_DB_PASSWORD=wp \
247 > --name wordpress
248 > wordpress
249
```

```
250 -브라우저에서 연결
251 http://localhost:8080
252
253
```

## 2)두번째 방법

-MySQL 실행하기

```
256 $ docker run -d -p 3306:3306 \
257 > -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
258 > --name mysql \
259 > mysql:5.7
260
```

-MySQL에 wp 데이터베이스 생성 및 wp 계정 생성

```
261 $ docker exec -it mysql mysql
262 mysql> CREATE DATABASE wp CHARACTER SET utf8;
263 mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
264 mysql> FLUSH PRIVILEGES;
265 mysql> show databases;
266 +-----+
267 | Database |
268 +-----+
269 | information_schema |
270 | mysql |
271 | performance_schema |
272 | sys |
273 | wp |
274 +-----+
275 5 rows in set (0.00 sec)
276 mysql> quit
277
278
```

-app-network 라는 이름으로 wordpress와 MySQL이 통신할 네트워크 만들기

```
279 $ docker network create app-network
280
281
```

-MySQL container에 네트워크를 추가

```
282 $ docker network connect app-network mysql
283
284
```

-network option 사용하기

-WordPress를 app-network에 속하게 하고 mysql을 이름으로 접근한다.

```
287 $ docker run -dp 8080:80 \
288 > --network=app-network \
289 > -e WORDPRESS_DB_HOST=mysql \
290 > -e WORDPRESS_DB_NAME=wp \
291 > -e WORDPRESS_DB_USER=wp \
292 > -e WORDPRESS_DB_PASSWORD=wp \
293 > wordpress
294
```

-웹 브라우저에서 확인

```
295 http://HOST-IP:8080
296
297
298
```

## 3)세번째 방법

- wordpress와 mysql 컨테이너 삭제

```
300 $ sudo docker rm -f `docker ps -a -q`
301 $ sudo docker ps -a
302 $ sudo docker rmi `docker images -q`
303
304
```

~/dbdata 디렉토리 삭제

```
305 $ sudo rm -rf /dbdata
306
307
```

-MySQL 실행하기

```
308 $ sudo docker run -d -p 3306:3306 \
309 > --name mysql -v /dbdata:/var/lib/mysql \
310 > -e MYSQL_ROOT_PASSWORD=wordpress \
311 > -e MYSQL_PASSWORD=wordpress mysql:5.7
312
313
```

-MySQL에 wp 데이터베이스 생성 및 wp 계정 생성

```
314 $ docker exec -it mysql bash
315 bash-4.2# mysql -h localhost -u root -p
316 Enter password:wordpress
317 mysql> CREATE DATABASE wp CHARACTER SET utf8;
318 mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
319 mysql> FLUSH PRIVILEGES;
320 mysql> show databases;
321 +-----+
322 | Database |
323 +-----+
324 | information_schema |
325 | mysql |
326 | performance_schema |
327 | sys |
328
```

```
329 | wp |
330 +-----+
331 5 rows in set (0.00 sec)
332 mysql> quit
333 bash-4.2# exit
334 exit
335
336 $ sudo docker ps -a
337
338 -wordpress container 실행
339 $ sudo docker run -dp 8080:80 \
340 > --name wordpress --link mysql:mymysql \ <--link의 이름의 앞부분은 mysql의 Container의 이름, 뒷부분은 자유
341 > -e WORDPRESS_DB_PASSWORD=wordpress \
342 > -e WORDPRESS_DB_HOST=mysql \
343 > -e WORDPRESS_DB_NAME=wp \
344 > -e WORDPRESS_DB_USER=wp \
345 > wordpress
346
347 $ sudo docker ps -a
348
349 -웹 브라우저에서 확인
350 http://HOST-IP:8080
```