

```

1 Lab. Container Network
2 1. Container Network 사용하기
3   1)docker0 사용 확인하기
4     $ ip addr
5     $ sudo brctl show
6
7     $ sudo docker run --name busybox1 -it busybox
8     /# ifconfig
9     eth0    Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
10    inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
11    UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
12    RX packets:9 errors:0 dropped:0 overruns:0 frame:0
13    TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
14    collisions:0 txqueuelen:0
15    RX bytes:806 (806.0 B)  TX bytes:0 (0.0 B)
16
17    /# ping -c 4 8.8.8.8  <----외부 통신 가능
18    /# iptables -t nat -L -v
19
20
21  2)자동으로 172.17.0.x의 아이피 부여 확인하기
22    -다른 세션을 열어서
23    $ sudo docker run --name busybox1 -it busybox
24    /# ifconfig
25    eth0    Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
26    inet addr:172.17.0.3  Bcast:172.17.255.255  Mask:255.255.0.0
27    UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
28    RX packets:9 errors:0 dropped:0 overruns:0 frame:0
29    TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
30    collisions:0 txqueuelen:0
31    RX bytes:806 (806.0 B)  TX bytes:0 (0.0 B)
32
33
34    -또 다른 세션을 열어서
35    $ sudo docker run -d -p 80:80 --name web nginx
36    $ sudo docker inspect web
37    $ curl 172.17.0.4
38    $ sudo iptables -t nat -L -v
39    Chain PREROUTING (policy ACCEPT 1 packets, 84 bytes)
40    pkts bytes target    prot opt in     out     source            destination
41    815 42348 DOCKER    all  --  any    any     anywhere          anywhere          ADDRTYPE
42    match dst-type LOCAL
43
44    Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
45    pkts bytes target    prot opt in     out     source            destination
46
47    Chain OUTPUT (policy ACCEPT 9 packets, 1174 bytes)
48    pkts bytes target    prot opt in     out     source            destination
49    0    0 DOCKER    all  --  any    any     anywhere          !localhost/8      ADDRTYPE
50    match dst-type LOCAL
51
52    Chain POSTROUTING (policy ACCEPT 9 packets, 1174 bytes)
53    pkts bytes target    prot opt in     out     source            destination
54    1513 93953 MASQUERADE all  --  any    !docker0 172.17.0.0/16    anywhere
55    0    0 MASQUERADE tcp  --  any    any     172.17.0.4      172.17.0.4      tcp dpt:http
56
57    Chain DOCKER (2 references)
58    pkts bytes target    prot opt in     out     source            destination
59    0    0 RETURN    all  --  docker0 any     anywhere          anywhere
60    0    0 DNAT      tcp  --  !docker0 any     anywhere          anywhere          tcp dpt:http
61    to:172.17.0.4:80
62
63  2. Port-Forwarding
64    1)host의 port와 container의 port 지정해서 연결하기
65    $ sudo docker run -p 80:80 -d --name web1 nginx

```

```

65
66 $ sudo docker ps
67 CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
68 05c359f8bcd6   nginx    "/docker-entrypoint...." About a minute ago Up About a minute
69 0.0.0.0:80->80/tcp, :::80->80/tcp    web1
70
71 $ curl localhost:80
72
73 2)host의 port를 랜덤으로 연결하기
74 $ sudo docker run -p 80 -d --name web2 nginx
75 $ sudo docker ps
76 CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
77 8e4372270de9   nginx    "/docker-entrypoint...." 6 seconds ago Up 5 seconds
78 0.0.0.0:49153->80/tcp, :::49153->80/tcp web2
79 05c359f8bcd6   nginx    "/docker-entrypoint...." About a minute ago Up About a minute
80 0.0.0.0:80->80/tcp, :::80->80/tcp    web1
81
82 $ curl localhost:49153
83
84 3)host와 container 모두 자동으로 연결하기
85 $ sudo docker run -P(대문자) 80 -d --name web3 nginx
86 $ sudo docker ps -a
87 CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
88 8ae0560aa57c   nginx    "/docker-entrypoint...." 3 seconds ago Up 2 seconds
89 0.0.0.0:49154->80/tcp, :::49154->80/tcp web3
90 8e4372270de9   nginx    "/docker-entrypoint...." 3 minutes ago Up 3 minutes
91 0.0.0.0:49153->80/tcp, :::49153->80/tcp web2
92 05c359f8bcd6   nginx    "/docker-entrypoint...." 4 minutes ago Up 4 minutes
93 0.0.0.0:80->80/tcp, :::80->80/tcp    web1
94
95 3. user-defined network 구성하기
96 1)기본 bridge외에 새로 생성하기
97 $ sudo docker network ls
98 NETWORK ID     NAME      DRIVER    SCOPE
99 32ce6dec4771   bridge   bridge    local
100 ef8f1c31a15d   host     host      local
101 ee449dfed7eb   none     null      local
102
103 $ sudo docker network create --driver bridge --subnet 192.168.100.0/24 \
104 > --gateway 192.168.100.254 mynet
105 df7b218797e7216e1b39549a94ab9b0b2b5d2946be63233ed8ac1b17a62742c6
106
107 $ sudo docker network ls
108 NETWORK ID     NAME      DRIVER    SCOPE
109 32ce6dec4771   bridge   bridge    local
110 ef8f1c31a15d   host     host      local
111 df7b218797e7   mynet     bridge    local
112 ee449dfed7eb   none     null      local
113
114 2)새로 생성한 bridge로 Container 생성하기
115 $ sudo docker run -it --name busybox1 --net mynet busybox
116 / # ifconfig
117 eth0    Link encap:Ethernet HWaddr 02:42:C0:A8:64:01
118         inet addr:192.168.100.1 Bcast:192.168.100.255 Mask:255.255.255.0
119         UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
120         RX packets:14 errors:0 dropped:0 overruns:0 frame:0
121         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
122         collisions:0 txqueuelen:0
123         RX bytes:1252 (1.2 KiB) TX bytes:0 (0.0 B)
124
125 lo      Link encap:Local Loopback
126         inet addr:127.0.0.1 Mask:255.0.0.0

```

```

123         UP LOOPBACK RUNNING MTU:65536 Metric:1
124         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
125         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
126         collisions:0 txqueuelen:1000
127         RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
128     /# exit
129
130
131 $ sudo docker inspect mynet
132 [
133     {
134         "Name": "mynet",
135         "Id": "df7b218797e7216e1b39549a94ab9b0b2b5d2946be63233ed8ac1b17a62742c6",
136         "Created": "2021-10-21T08:59:00.152346729Z",
137         "Scope": "local",
138         "Driver": "bridge",
139         "EnableIPv6": false,
140         "IPAM": {
141             "Driver": "default",
142             "Options": {},
143             "Config": [
144                 {
145                     "Subnet": "192.168.100.0/24",
146                     "Gateway": "192.168.100.254"
147                 }
148             ]
149         },
150         "Internal": false,
151         "Attachable": false,
152         "Ingress": false,
153         "ConfigFrom": {
154             "Network": ""
155         },
156         "ConfigOnly": false,
157         "Containers": {},
158         "Options": {},
159         "Labels": {}
160     }
161 ]

```

3) Container 생성시 ip 지정하기

```

165 $ sudo docker run -it --name busybox2 --net mynet --ip 192.168.100.100 busybox
166 / # ifconfig
167 eth0    Link encap:Ethernet HWaddr 02:42:C0:A8:64:64
168         inet addr:192.168.100.100 Bcast:192.168.100.255 Mask:255.255.255.0
169         UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
170         RX packets:8 errors:0 dropped:0 overruns:0 frame:0
171         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
172         collisions:0 txqueuelen:0
173         RX bytes:736 (736.0 B) TX bytes:0 (0.0 B)
174
175     lo    Link encap:Local Loopback
176         inet addr:127.0.0.1 Mask:255.0.0.0
177         UP LOOPBACK RUNNING MTU:65536 Metric:1
178         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
179         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
180         collisions:0 txqueuelen:1000
181         RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
182
183     / # ping -c 4 8.8.8.8

```

4. Container 간 통신하기

1) 첫번째 방법

-MySQL 실행하기

```

190 $ docker run -d -p 3306:3306 \
191 > -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
192 > --name mysql \
193 > mysql:5.7
194
195 $ docker exec -it mysql mysql
196 mysql> CREATE DATABASE wp CHARACTER SET utf8;
197 mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
198 mysql> FLUSH PRIVILEGES;
199 mysql> show databases;
200 +-----+
201 | Database      |
202 +-----+
203 | information_schema |
204 | mysql          |
205 | performance_schema |
206 | sys            |
207 | wp              |
208 +-----+
209 5 rows in set (0.00 sec)
210 mysql> quit
211

```

212 -WordPress 실행하기

```

213 $ docker run -d -p 8080:80 \
214 > -e WORDPRESS_DB_HOST=host.docker.internal \ <---Linux에서는 연결안됨. WSL만 가능
215 > -e WORDPRESS_DB_NAME=wp \
216 > -e WORDPRESS_DB_USER=wp \
217 > -e WORDPRESS_DB_PASSWORD=wp \
218 > --name wordpress
219 > wordpress
220

```

221 -브라우저에서 연결

```

222 http://localhost:8080
223
224

```

225 2)두번째 방법

226 -app-network 라는 이름으로 wordpress와 MySQL이 통신할 네트워크 만들기

```

227 $ docker network create app-network
228

```

229 -MySQL containier에 네트워크를 추가

```

230 $ docker network connect app-network mysql
231

```

232 -network option 사용하기

233 -WordPress를 app-network에 속하게 하고 mysql을 이름으로 접근한다.

```

234 $ docker stop wordpress
235 $ docker rm -f wordpress
236 $ docker run -dp 8080:80 \
237 > --network=app-network \
238 > -e WORDPRESS_DB_HOST=mysql \
239 > -e WORDPRESS_DB_NAME=wp \
240 > -e WORDPRESS_DB_USER=wp \
241 > -e WORDPRESS_DB_PASSWORD=wp \
242 > wordpress
243

```

244 -웹 브라우저에서 확인

```

245 http://192.168.56.101:8080
246
247

```

248 3)세번째 방법

```

249 $ sudo docker run -d --name mysql \
250 > -v /dbdata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=wordpress \
251 > -e MYSQL_PASSWORD=wordpress mysql:5.7
252

```

```

253 $ sudo docker ps -a
254

```

```

255 $ sudo docker run -d --name wordpress --link mysql:mymysql \ <--link의 이름의 앞부분은 mysql의
Container의 이름, 뒷부분은 자유

```

```
256 > -e WORDPRESS_DB_PASSWORD=wordpress -p 80:80 \
257 > wordpress:4
258
259 $ sudo docker ps -a
260
261 -브라우저에서 연결 후 홈페이지 설정과 글 수정
262   http://192.168.56.101:80
263
264 - wordpress와 mysql 컨테이너 삭제
265   $ sudo docker rm -f wordpress
266   $ sudo docker rm -f mysql
267   $ sudo docker ps -a
268   $ sudo docker rmi -f mysql:5.7
269   $ sudo docker rmi -f wordpress:4
270
271 -다시 wordpress와 mysql 컨테이너 다운로드 후 실행
272
273   $ sudo docker run -d --name mysql \
274   > -v /dbdata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=wordpress \
275   > -e MYSQL_PASSWORD=wordpress mysql:5.7
276
277   $ sudo docker ps -a
278
279   $ sudo docker run -d --name wordpress --link mysql:mymysql \ <--link의 이름의 앞부분은
mysql의 Container의 이름, 뒷부분은 자유
280   > -e WORDPRESS_DB_PASSWORD=wordpress -p 80:80 \
281   > wordpress:4
282
283   $ sudo docker ps -a
284
285 -브라우저에서 이전에 수정했던 글이 있는지 확인하기
286   http://192.168.56.101:80
```