Lab. Container Network

1. Container Network 사용하기

1)docker0 사용 확인하기

```
$ ip addr
   1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
      link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
      inet 127.0.0.1/8 scope host lo
         valid_lft forever preferred_lft forever
      inet6 ::1/128 scope host
         valid_lft forever preferred_lft forever
   2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
      link/ether 02:6b:81:64:e1:32 brd ff:ff:ff:ff:ff:ff
      inet 10.0.10.23/24 metric 100 brd 10.0.10.255 scope global dynamic eth0
         valid_lft 1923sec preferred_lft 1923sec
      inet6 fe80::6b:81ff:fe64:e132/64 scope link
         valid_lft forever preferred_lft forever
   3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
      link/ether 02:42:69:d8:e7:3d brd ff:ff:ff:ff:ff:ff
      inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
         valid_lft forever preferred_lft forever
      inet6 fe80::42:69ff:fed8:e73d/64 scope link
         valid_lft forever preferred_lft forever

$ sudo brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.024269d8e73d       no

$ sudo docker run --name busybox -it busybox
/# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:876 (876.0 B)  TX bytes:0 (0.0 B)

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

/# ping -c 4 8.8.8.8                  <----외부 통신 가능 확인
```

2)자동으로 172.17.0.x의 IP Address 부여 확인하기

-다른 세션을 열어서

```
$ sudo docker run --name busybox1 -it busybox
/# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.3  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:806 (806.0 B)  TX bytes:0 (0.0 B)
```

-또 다른 세션을 열어서

```
$ sudo docker run -d -p 80:80 --name web nginx
$ sudo docker inspect web
$ curl 172.17.0.4
$ sudo iptables -t nat -L -v
   Chain PREROUTING (policy ACCEPT 1 packets, 84 bytes)
    pkts bytes target     prot opt in     out     source               destination
     815 42348 DOCKER     all  --  any    any     anywhere             anywhere             ADDRTYPE match dst-type LOCAL

   Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
    pkts bytes target     prot opt in     out     source               destination

   Chain OUTPUT (policy ACCEPT 9 packets, 1174 bytes)
    pkts bytes target     prot opt in     out     source               destination
       0     0 DOCKER     all  --  any    any     anywhere             !localhost/8         ADDRTYPE match dst-type LOCAL

   Chain POSTROUTING (policy ACCEPT 9 packets, 1174 bytes)
    pkts bytes target     prot opt in     out     source               destination
    1513 93953 MASQUERADE  all  --  any    !docker0  172.17.0.0/16       anywhere
       0     0 MASQUERADE  tcp  --  any    any     172.17.0.4           172.17.0.4           tcp dpt:http

   Chain DOCKER (2 references)
```

```
    pkts bytes target     prot opt in     out     source               destination
      0     0 RETURN     all  --  docker0 any     anywhere             anywhere
      0     0 DNAT       tcp  --  !docker0 any    anywhere             anywhere            tcp dpt:http to:172.17.0.4:80
```

2. Port-Forwarding

1)host의 port와 container의 port 지정해서 연결하기

```
$ sudo docker run -p 80:80 -d --name web1 nginx

$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS             PORTS                                   NAMES
05c359f8bcd6   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:80->80/tcp, :::80->80/tcp       web1

$ curl localhost:80
```

2)host의 port를 랜덤으로 연결하기

```
$ sudo docker run -p 80 -d --name web2 nginx
$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS             PORTS                                   NAMES
8e4372270de9   nginx     "/docker-entrypoint.…"   6 seconds ago       Up 5 seconds       0.0.0.0:49153->80/tcp, :::49153->80/tcp   web2
05c359f8bcd6   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:80->80/tcp, :::80->80/tcp       web1

$ curl localhost:49153
```

3)host와 container 모두 자동으로 연결하기

```
$ sudo docker run -P(대문자) 80 -d --name web3 nginx
$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                   NAMES
8ae0560aa57c   nginx     "/docker-entrypoint.…"   3 seconds ago   Up 2 seconds   0.0.0.0:49154->80/tcp, :::49154->80/tcp   web3
8e4372270de9   nginx     "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   0.0.0.0:49153->80/tcp, :::49153->80/tcp   web2
05c359f8bcd6   nginx     "/docker-entrypoint.…"   4 minutes ago   Up 4 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp       web1
```

3. user-defined network 구성하기

1)기본 bridge외에 새로 생성하기

```
$ sudo docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
32ce6dec4771   bridge    bridge    local
ef8f1c31a15d   host      host      local
ee449dfed7eb   none      null      local

$ sudo docker network create --driver bridge --subnet 192.168.100.0/24 \
> --gateway 192.168.100.254 mynet
df7b218797e7216e1b39549a94ab9b0b2b5d2946be63233ed8ac1b17a62742c6

$ sudo docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
32ce6dec4771   bridge    bridge    local
ef8f1c31a15d   host      host      local
df7b218797e7   mynet     bridge    local
ee449dfed7eb   none      null      local
```

2)새로 생성한 bridge로 Container 생성하기

```
$ sudo docker run -it --name busybox1 --net mynet busybox
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:C0:A8:64:01
          inet addr:192.168.100.1  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1252 (1.2 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
/# exit


$ sudo docker inspect mynet
```

```
[
    {
        "Name": "mynet",
        "Id": "df7b218797e7216e1b39549a94ab9b0b2b5d2946be63233ed8ac1b17a62742c6",
        "Created": "2021-10-21T08:59:00.152346729Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "192.168.100.0/24",
                    "Gateway": "192.168.100.254"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

3)Container 생성시 ip 지정하기

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

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

/ # ping -c 4 8.8.8.8
```

4. Container 간 통신하기

1)첫번째 방법

-MySQL 실행하기

```
$ docker run -d -p 3306:3306 \
> -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
> --name mysql \
> mysql:5.7

$ docker exec -it mysql mysql
mysql> CREATE DATABASE wp CHARACTER SET utf8;
mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
mysql> FLUSH PRIVILEGES;
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wp                 |
+--------------------+
5 rows in set (0.00 sec)
mysql> quit
```

-WordPress 실행하기

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

```
> -e WORDPRESS_DB_USER=wp \
> -e WORDPRESS_DB_PASSWORD=wp \
> --name wordpress
> wordpress
```

-브라우저에서 연결

http://localhost:8080

2)두번째 방법

-MySQL 실행하기

```
$ docker run -d -p 3306:3306 \
> -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
> --name mysql \
> mysql:5.7
```

-MySQL에 wp 데이터베이스 생성 및 wp 계정 생성

```
$ docker exec -it mysql mysql
mysql> CREATE DATABASE wp CHARACTER SET utf8;
mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
mysql> FLUSH PRIVILEGES;
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wp                 |
+--------------------+
5 rows in set (0.00 sec)
mysql> quit
```

-app-network 라는 이름으로 wordpress와 MySQL이 통신할 네트워크 만들기

```
$ docker network create app-network
```

-MySQL containier에 네트워크를 추가

```
$ docker network connect app-network mysql
```

-network option 사용하기

-WordPress를 app-network에 속하게 하고 mysql을 이름으로 접근한다.

```
$ docker run -dp 8080:80 \
> --network=app-network \
> -e WORDPRESS_DB_HOST=mysql \
> -e WORDPRESS_DB_NAME=wp \
> -e WORDPRESS_DB_USER=wp \
> -e WORDPRESS_DB_PASSWORD=wp \
> wordpress
```

-웹 브라우저에서 확인

http://HOST-IP:8080

3)세번째 방법

- wordpress와 mysql 컨테이너 삭제

```
$ sudo docker rm -f `docker ps -a -q`
$ sudo docker ps -a
$ sudo docker rmi `docker images -q`
```

-/dbdata 디렉토리 삭제

```
$ sudo rm -rf /dbdata
```

-MySQL 실행하기

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

-MySQL에 wp 데이터베이스 생성 및 wp 계정 생성

```
$ docker exec -it mysql bash
bash-4.2# mysql -h localhost -u root -p
Enter password:wordpress
mysql> CREATE DATABASE wp CHARACTER SET utf8;
mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
mysql> FLUSH PRIVILEGES;
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
```

```
| wp                 |
+--------------------+
5 rows in set (0.00 sec)
mysql> quit
bash-4.2# exit
exit

$ sudo docker ps -a

-wordpress container 실행
$ sudo docker run -dp 8080:80 \
> --name wordpress --link mysql:mymysql \   <--link의 이름의 앞부분은 mysql의 Container의 이름, 뒷부분은 자유
> -e WORDPRESS_DB_PASSWORD=wordpress \
> -e WORDPRESS_DB_HOST=mysql \
> -e WORDPRESS_DB_NAME=wp \
> -e WORDPRESS_DB_USER=wp \
> wordpress

$ sudo docker ps -a

-웹 브라우저에서 확인
http://HOST-IP:8080
```