Team bbansrun

Docker Major Commands related to Networking

Starts with “docker network”

```
jiwon@jeonjiwon-ui-MacBookAir ~ docker network

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
```

Docker Network List

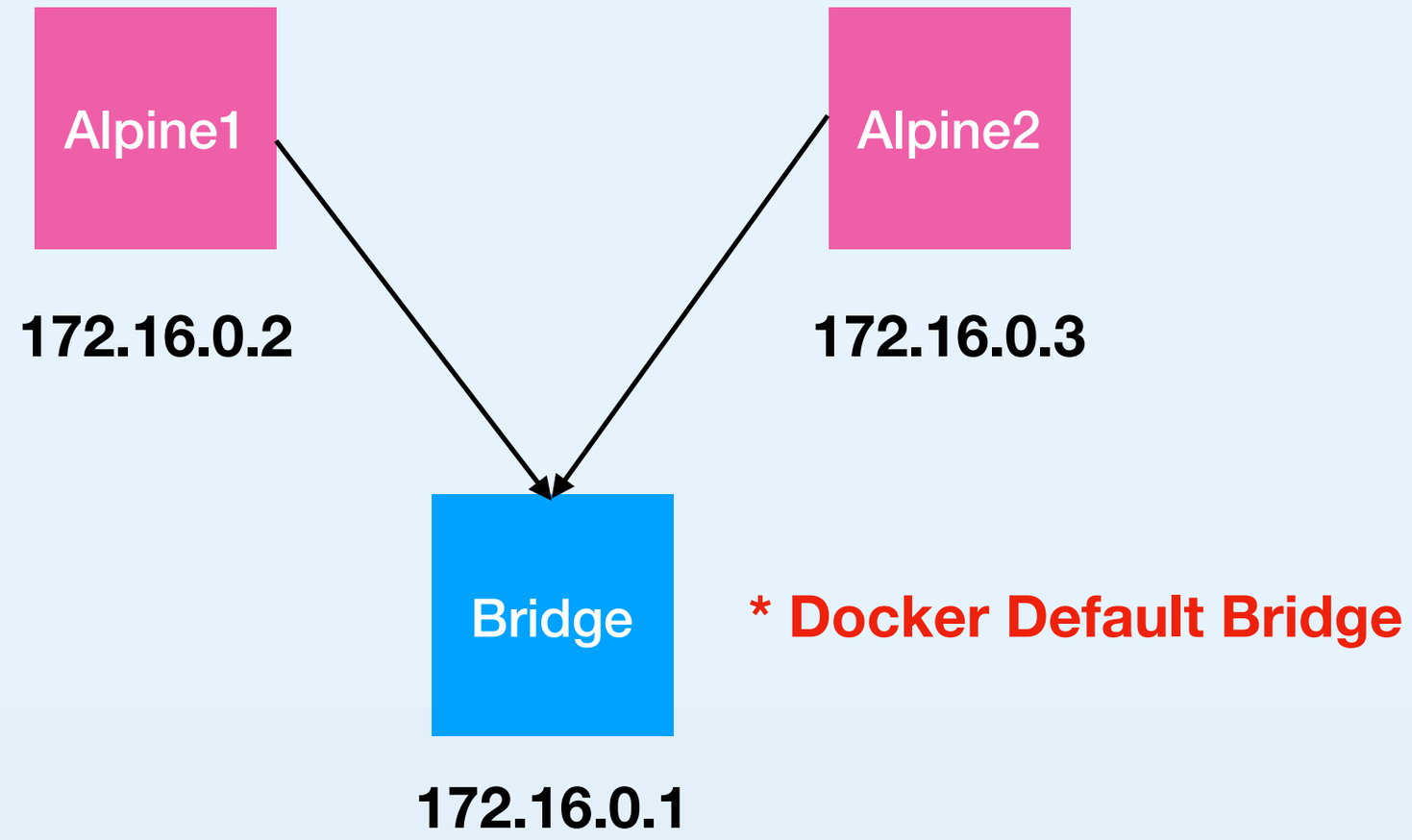
```
jiwon@jeonjiwon-ui-MacBookAir ~ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
a2251afe2ac1        bridge              bridge              local
e27b61ce3274        host                host                local
ac7094c27b7a        none                null                local
```

Inspect the bridge network
to see what containers are connected to it

> docker network inspect [NETWORK]


```
[
    {
        "Name": "bridge",
        "Id": "a2251afe2ac1324d55200f9e91328dd0089064614121e8a63c3587982c89a019",
        "Created": "2019-08-24T05:18:28.565642457Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "03ae8fa794a567c302f357bdf5b90de819c641f84c17f9971f5c9c1d7caec707": {
                "Name": "alpine2",
                "EndpointID": "bfc116acf8087e8baea0598a4be8b001974dc1362285ce4c6ce1ba50fed6fbc0",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            },
            "4f1cd5029db18ba9c75cd507bf1ae162ac4ef5b79513825e4268f1d9ae95e78e": {
                "Name": "alpine1",
                "EndpointID": "c0c680a3297776167ff8951afbceff3e90b7045d1b3c4be38836353b6ea4841f",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Network Configuration Figure



Create User-defined Bridge Network

> docker network create —driver bridge [YOUR_NETWORK_NAME]

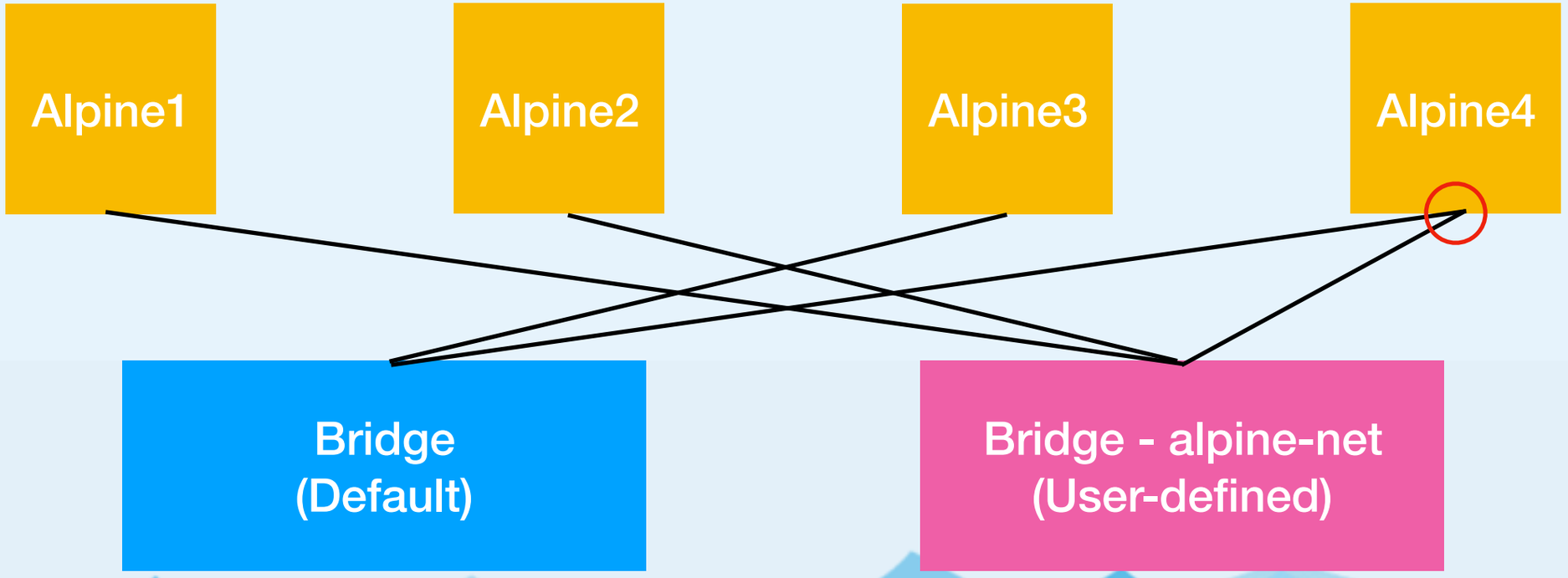

```
[
    {
        "Name": "alpine-net",
        "Id": "92124d2f9c555f03e44e2368789988618ce7cef49b8d59a668666df293a1ca2d",
        "Created": "2019-08-24T07:08:23.059039431Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

Connect a container to a network

> docker network connect [NETWORK] [CONTAINER_NAME]

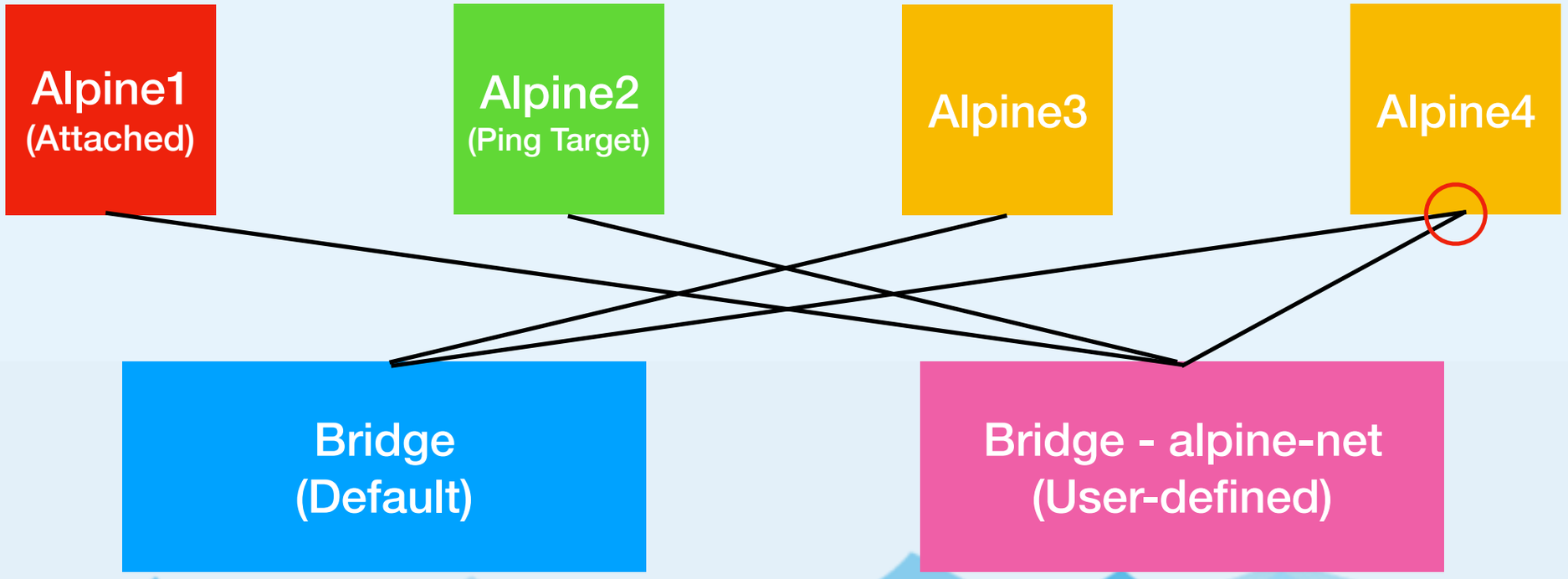
Example

```
$ docker run -dit --name alpine1 --network alpine-net alpine ash
$ docker run -dit --name alpine2 --network alpine-net alpine ash
$ docker run -dit --name alpine3 alpine ash
$ docker run -dit --name alpine4 --network alpine-net alpine ash
$ docker network connect bridge alpine4
```



Automatic Service Discovery (In User-defined network)

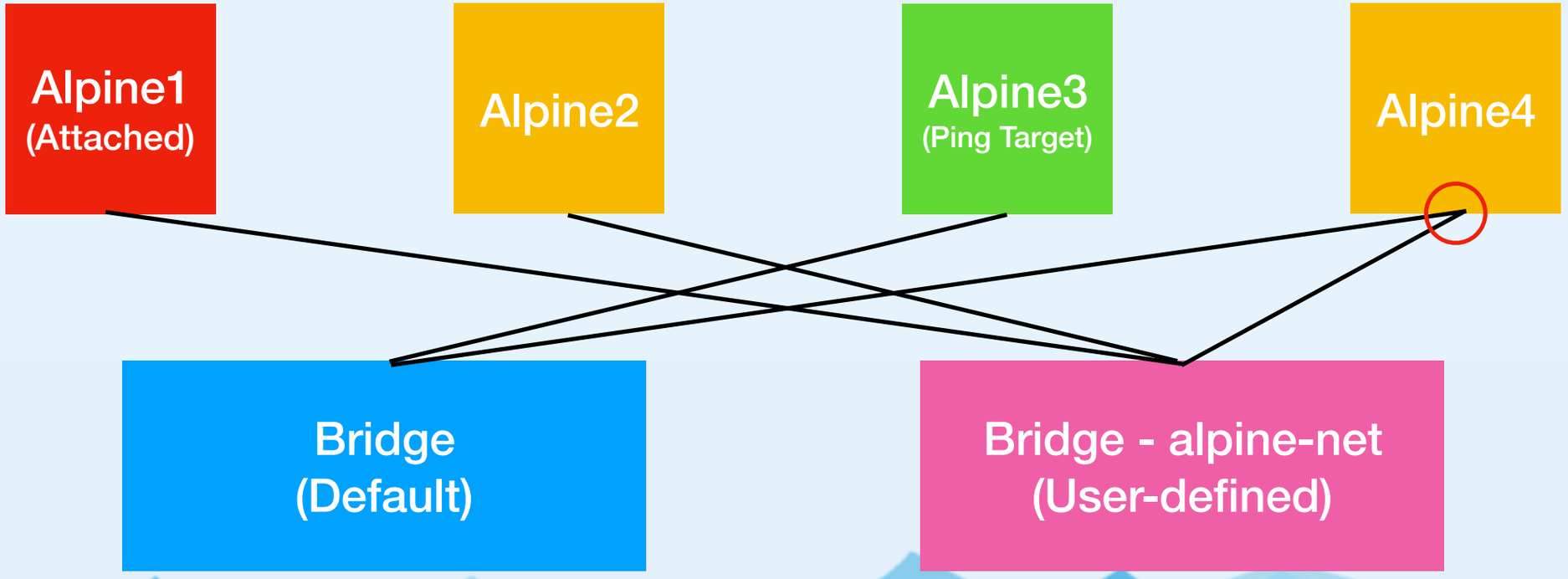
```
$ docker container attach alpine1

# ping -c 2 alpine2

PING alpine2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.085 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.090 ms

--- alpine2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.085/0.087/0.090 ms
```



Automatic Service Discovery (In User-defined network)

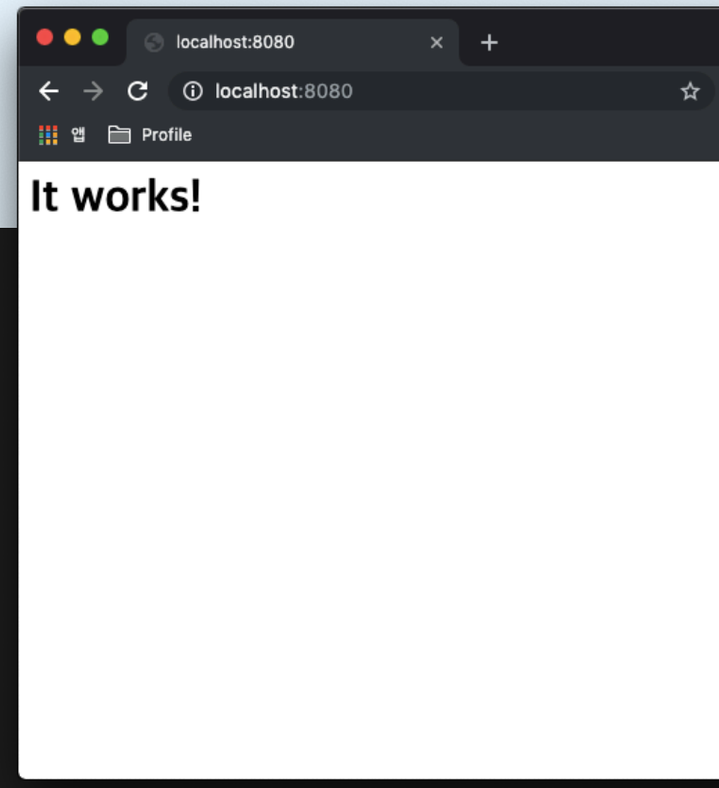
```
$ docker container attach alpine1

# ping -c 2 alpine3

ping: bad address 'alpine3'
```

Because the container ‘Alpine3’
is **not in user-defined network**
(The opposite case (alpine3 -> alpine1) cannot also be true.)



Expose Container Application outside

> docker run -d [HOST_PORT]:[EXPOSE_PORT] [CONTAINER_NAME] [IMAGE]

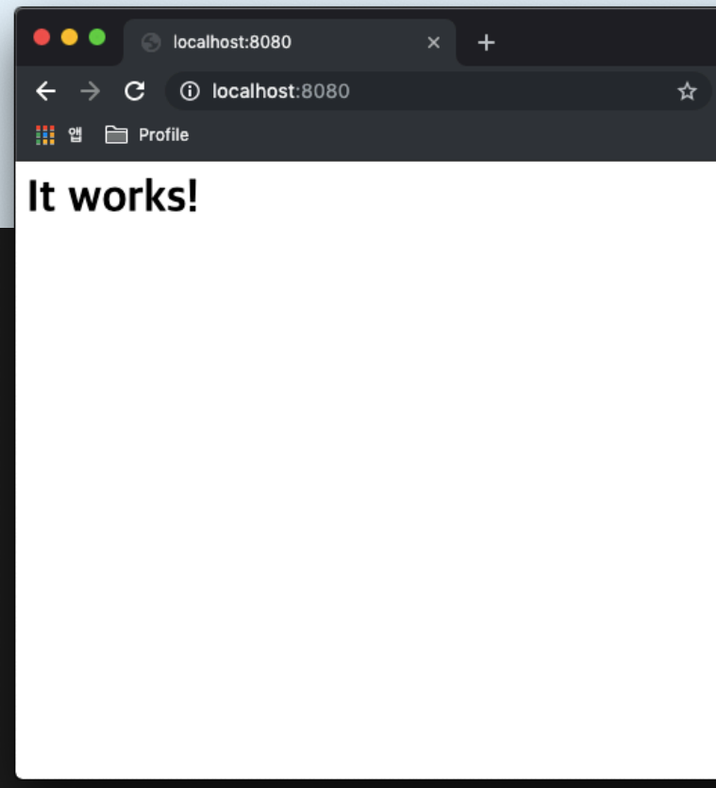



```
jiwon@jeonjiwon-ui-MacBookAir ~ docker run -d -p 8080:80 --name web_svr01 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
1ab2bdfe9778: Pull complete
174a8e3bca83: Pull complete
c8e4c9e94892: Pull complete
4568916ecf2d: Pull complete
533f5cf513cb: Pull complete
Digest: sha256:98caed3e3a90ed9db8d25dcbb98eebe0ce56358a9dbbc940d7eb66a8e2b88252
Status: Downloaded newer image for httpd:latest
d1fa7d9c22a688011f8eaabc91d749cdde80a4ea4197591b78eb241683a01aac
jiwon@jeonjiwon-ui-MacBookAir ~ docker ps -a
CONTAINER ID   IMAGE     COMMAND              CREATED          STATUS          PORTS                  NAMES
d1fa7d9c22a6   httpd     "httpd-foreground"   8 seconds ago    Up 6 seconds    0.0.0.0:8080->80/tcp   web_svr01
```

Expose Container Application outside

docker run -d -p 8080:80 —name web_svr01 httpd

Docker Host 8080 포트로 들어온 요청을
컨테이너의 80번 포트로 Forwarding




```
jiwon@jeonjiwon-ui-MacBookAir ~ docker run -d -p 8080:80 --name web_svr01 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
1ab2bdfe9778: Pull complete
174a8e3bca83: Pull complete
c8e4c9e94892: Pull complete
4568916ecf2d: Pull complete
533f5cf513cb: Pull complete
Digest: sha256:98caed3e3a90ed9db8d25dcbb98eebe0ce56358a9dbbc940d7eb66a8e2b88252
Status: Downloaded newer image for httpd:latest
d1fa7d9c22a688011f8eaabc91d749cdde80a4ea4197591b78eb241683a01aac
jiwon@jeonjiwon-ui-MacBookAir ~ docker ps -a
CONTAINER ID   IMAGE     COMMAND              CREATED          STATUS          PORTS                  NAMES
d1fa7d9c22a6   httpd     "httpd-foreground"   8 seconds ago    Up 6 seconds    0.0.0.0:8080->80/tcp   web_svr01
```

Docker-proxy 프로세스가 해당 req 처리

iptables in Linux (pf.conf in MacOS)

Packet (Outside)

```
root@~~# iptables -t nat -L -n
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  0.0.0.0/0            0.0.0.0/0           ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  0.0.0.0/0           !127.0.0.0/8         ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.17.0.0/16       0.0.0.0/0
MASQUERADE  tcp  --  172.17.0.2          172.17.0.2          tcp dpt:80

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:8080 to:172.17.0.2:80
```

Outside

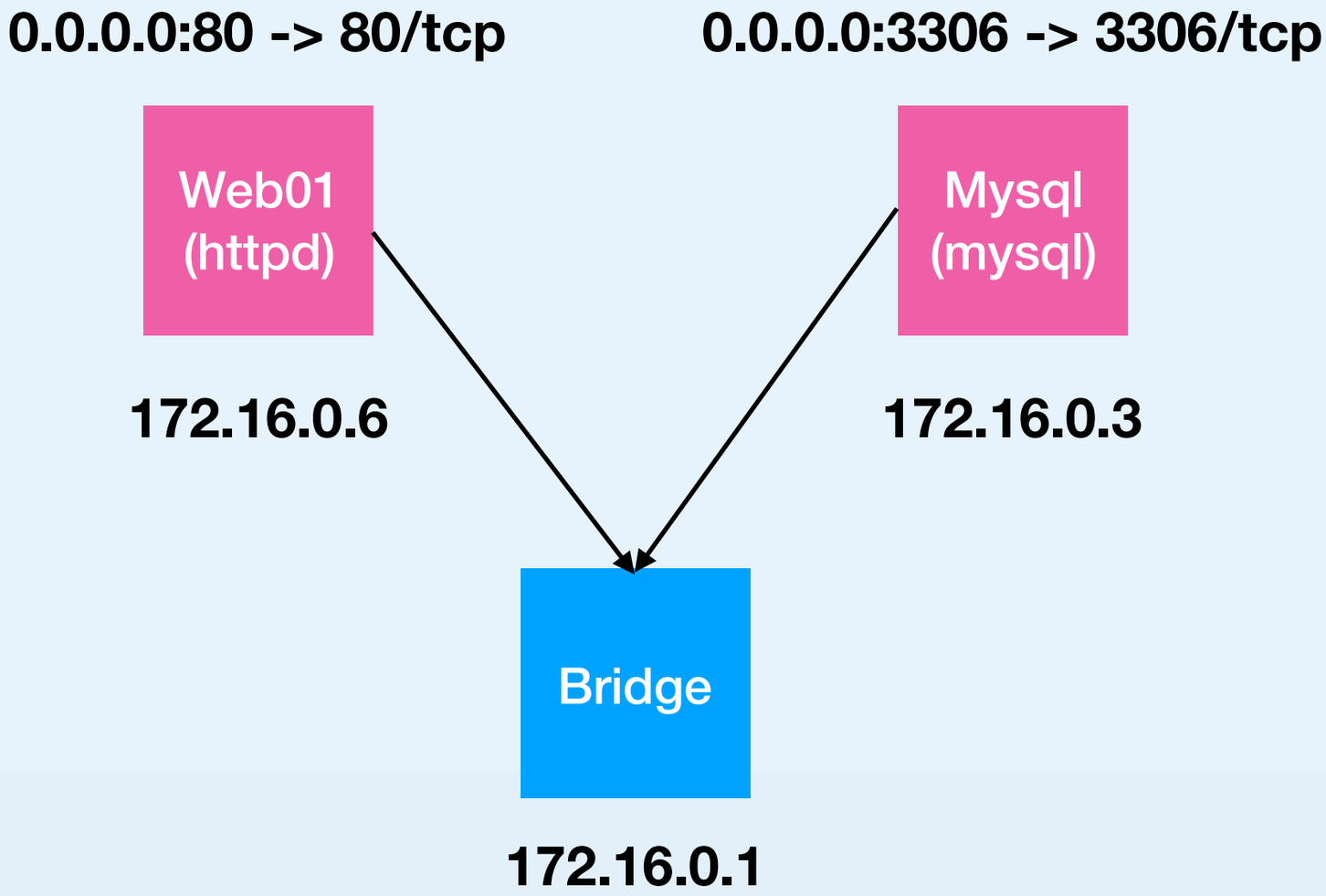
Response

iptables rule 및 port forwarding setup은 docker-daemon에서 관리

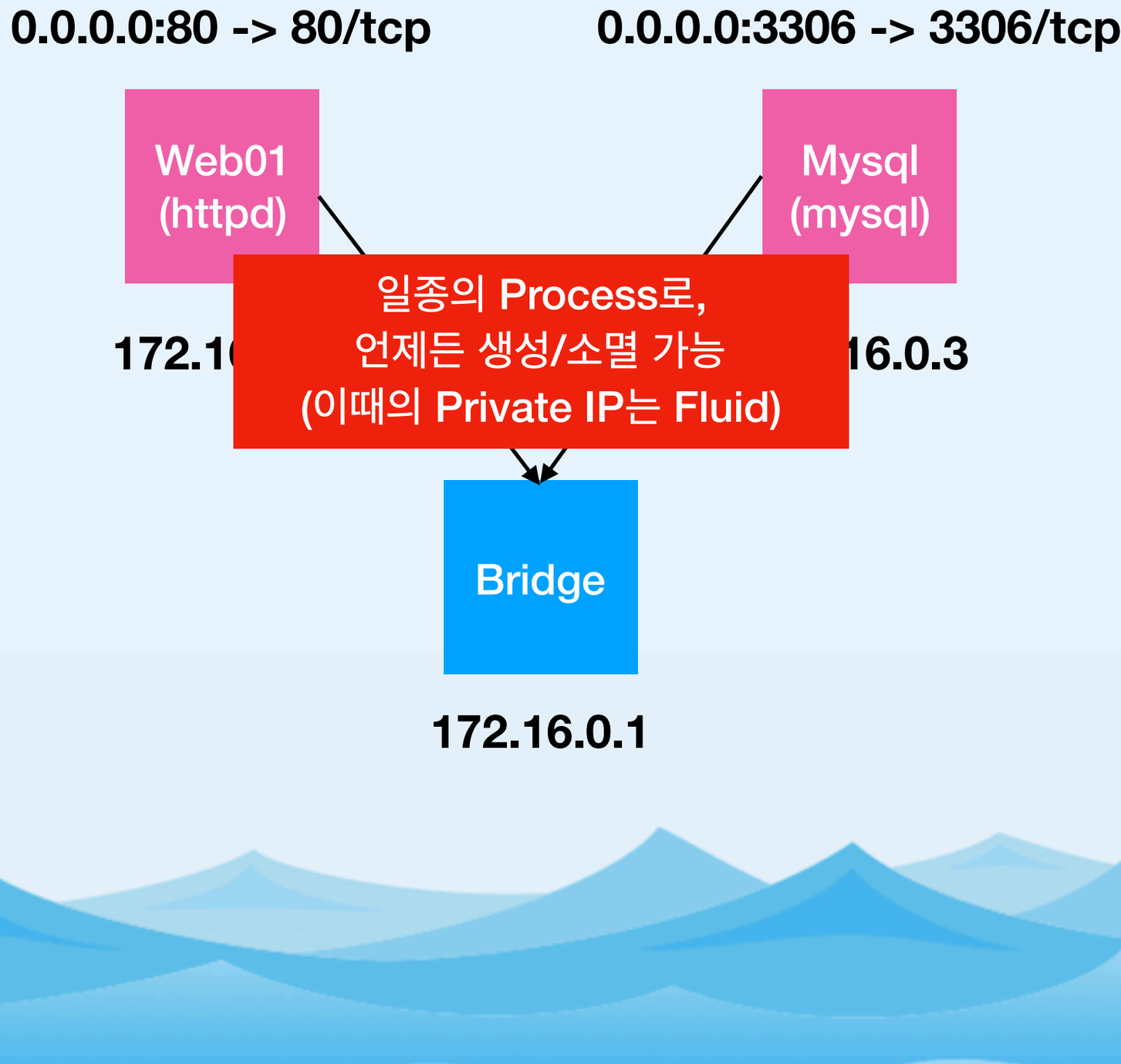
Team bbansrun

Linking between Containers

Fluid Container IPs

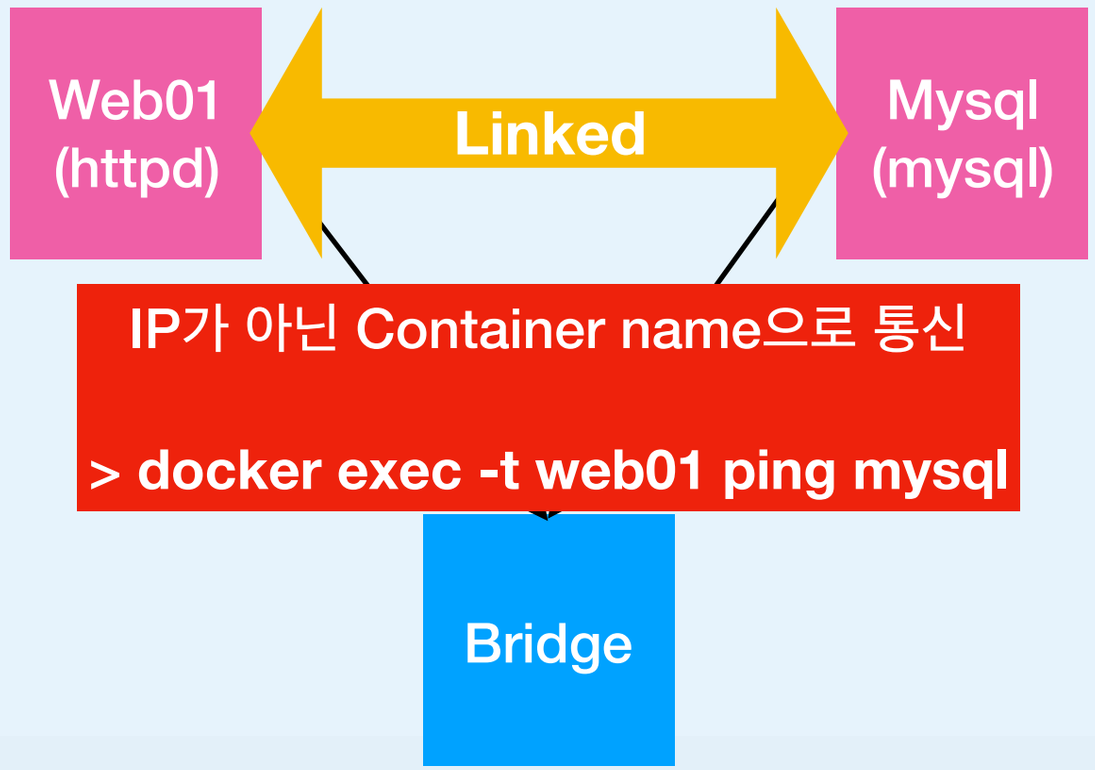


Fluid Container IPs



Interlocking Containers

> docker run -d --name web01 --link mysql httpd



172.16.0.1

Interlocking Principles

Web01
(httpd)

Docker daemon이 Container 일부 환경 변경

/etc/hosts

```
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.3      mysql 17b6c5f037a9
```

mysql 컨테이너가 재기동되어 IP가 갱신되는 경우
web01 컨테이너의 /etc/hosts 파일 자동 갱신

Pros and Cons

Pros

동적 IP 현상에 따른 이슈 해결

Cons

동일 host 내의 containers에만 유효
-> Orchestration 혹은 Dynamic DNS 구축 필요

References

- https://docs.docker.com/network/network-tutorial-standalone/
- https://bluese05.tistory.com/53?category=559611