



SPRAWOZDANIE

PROGRAMOWANIE W CHMURZE OBLICZENIOWEJ

IMIĘ I NAZWISKO: Piotr Czajka

NUMER LABORATORIUM 3

GRUPA: 7.1.2

Data wykonywania ćwiczenia: 25.10.2018

Spis treści

1	Cel laboratorium	3
2	Przebieg ćwiczenia	3
2.1	Zadanie pierwsze	3
2.1.1	Podpunkt pierwszy	3
2.1.2	Podpunkt drugi	4
2.1.3	Podpunkt trzeci	4
2.1.4	Konfiguracje sieci	4
2.1.5	Pytanie pierwsze:	11
2.2	Zadanie drugie	11
2.2.1	Pytanie pierwsze	12
2.2.2	Pytanie drugie	13
2.2.3	Pytanie trzecie:	14
2.3	Zadanie trzecie:	15
2.3.1	Pytanie pierwsze:	15
2.3.2	Pytanie drugie:	16

1 Cel laboratorium

Celem laboratorium było zapoznanie się z konfiguracją połączeń sieciowych i komunikacją w środowisko Docker

2 Przebieg ćwiczenia

2.1 Zadanie pierwsze

Najpierw należało wykonać odpowiednią sieć. Robi to skrypt zamieszczony poniżej. Ponadto zawiera on komentarze w miejscach wartych zainteresowania:

```
1 sysctl net.ipv4.conf.all.forwarding=1
2 sudo iptables -P FORWARD ACCEPT
3 # Dzięki tym regulom możliwa jest komunikacja kontener -> host
4
5 docker pull alpine
6 docker pull ubuntu
7 docker pull nginx
8 docker pull tomcat
9
10 docker run -itd --name T1 alpine sh
11
12 docker network create -d bridge --subnet 10.0.10.0/24 bridge1
13 docker run -itd --name T2 --net bridge --expose 80 -p 80:80 -p
    ↪ 10.0.10.1:8000:80 nginx sh
14 docker network connect bridge1 T2
15
16 docker exec -itd T2 /bin/bash -c 'service nginx start'
17 # od teraz możemy się połączyć z nginx na kontenerze T2 wchodząc na
18 # 'localhost' lub '10.0.10.1:8000' z komputera hosta
19
20
21 docker run -itd --name D1 --net bridge1 alpine sh
22
23 docker network create -d bridge bridge2
24 docker run -itd --net bridge2 --name S1 ubuntu sh
25
26 docker run -itd --name D2 --expose 8080 --net bridge2 -p 8081:8080 -p
    ↪ 10.0.10.0:8080:8080 tomcat sh
27 # Na moim dockerze domyślnie tomcat słucha na 8080
28 docker network connect bridge1 D2
29
30 docker exec D2 /bin/bash -c '/usr/local/tomcat/bin/startup.sh'
31
32 docker run -itd --name late --net bridge2 ubuntu bash
33 # przez podanie --net kontener połączy się z tą siecią, zamiast z domyślną
34 docker network connect bridge1 late
35 # łączymy go do drugiej sieci
```

2.1.1 Podpunkt pierwszy

W moim przypadku zmapowałem porty 8080 i 8081 do kontenera D2, ponieważ server Tomcat w tym kontenerze słuchał na porcie 80

2.1.2 Podpunkt drugi

Realizują go następujące linie:

```
1 | docker run -itd --name late --net bridge2 ubuntu bash
2 | # przez podanie --net kontener połączy się z tą siecią, zamiast z domyślną
3 | docker network connect bridge1 late
4 | # łączymy go do drugiej sieci
```

2.1.3 Podpunkt trzeci

Tu chyba jest błąd w instrukcji. Kontener D1 został połączony z siecią ze zdefiniowanym subnetem, myślę, że chodziło raczej o kontener D2, który razem z kontenerem S1 został połączony z siecią bez filtrowania. A zrealizowały to następujące linie.

Tu tworzę sieć:

```
1 | docker network create -d bridge bridge2
```

Tu tworzę D2 i przyłączam go do tej sieci:

```
1 | docker run -itd --name D2 --expose 8080 --net bridge2 -p 8081:8080 -p
   | ↪ 10.0.10.0:8080:8080 tomcat sh
2 | # Na moim dockerze domyślnie tomcat słucha na 8080
```

To samo dla S1:

```
1 | docker run -itd --net bridge2 --name S1 ubuntu sh
```

Komunikację kontener -> host osiągam za pomocą modyfikacji na hoscie za pomocą tych poleceń:

```
1 | sysctl net.ipv4.conf.all.forwarding=1
2 | sudo iptables -P FORWARD ACCEPT
```

Możliwość połączenia z kontenera do hosta potwierdza poniższy screen:



The screenshot shows two terminal windows. The left window is a root shell inside a container (root@4b7872d15060) where a netcat listener is running on 172.21.0.1:9000. It receives a connection from 10.0.10.1 and says 'Hello~'. The right window is a host terminal (root@kali) where a netcat listener is running on 0.0.0.0:9000. It receives a connection from 10.0.10.33 and says 'Hello~'.

2.1.4 Konfiguracje sieci

Sieć domyślna:

```
1 | [
2 |   {
3 |     "Name": "bridge",
4 |     "Id": "30397f853ba3046df1cb98e855c359ed046750e1b3bbd7a87e5d21feb6af5636
   |     ↪ ",
5 |     "Created": "2018-11-04T11:15:27.158287058+01:00",
6 |     "Scope": "local",
7 |     "Driver": "bridge",
8 |     "EnableIPv6": false,
9 |     "IPAM": {
10 |       "Driver": "default",
11 |       "Options": null,
12 |       "Config": [
13 |         {
14 |           "Subnet": "172.17.0.0/16",
15 |           "Gateway": "172.17.0.1"
16 |         }
   |     ]
   |   }
   | ]
```

```

17     ]
18 },
19 "Internal": false,
20 "Attachable": false,
21 "Ingress": false,
22 "ConfigFrom": {
23     "Network": ""
24 },
25 "ConfigOnly": false,
26 "Containers": {
27     "18e13c6761e8cd1fcd1cefd3399f3964c4f613d9851b5ce7e2d8349c84d4f9ce":
28         ↪ {
29             "Name": "T1",
30             "EndpointID": "392018
31                 ↪ ee522fd1edbaba3e48481e900f4982ce3805b8ca697a1803640d48cd61
32                 ↪ ",
33             "MacAddress": "02:42:ac:11:00:02",
34             "IPv4Address": "172.17.0.2/16",
35             "IPv6Address": ""
36         },
37     "78c6a45bdeb8df2e39d32f558dae2146f606be41d530c6f9c4c4505107dd27c0":
38         ↪ {
39             "Name": "T2",
40             "EndpointID": "
41                 ↪ ae1a96b1adda93d28e1764af57275fb066ca3397a80821b02843c7b13221cf59
42                 ↪ ",
43             "MacAddress": "02:42:ac:11:00:03",
44             "IPv4Address": "172.17.0.3/16",
45             "IPv6Address": ""
46         }
47 },
48 "Options": {
49     "com.docker.network.bridge.default_bridge": "true",
50     "com.docker.network.bridge.enable_icc": "true",
51     "com.docker.network.bridge.enable_ip_masquerade": "true",
52     "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
53     "com.docker.network.bridge.name": "docker0",
54     "com.docker.network.driver.mtu": "1500"
55 },
56 "Labels": {}
57 }
58 ]

```

Jak widać tu:

```

1     "18e13c6761e8cd1fcd1cefd3399f3964c4f613d9851b5ce7e2d8349c84d4f9ce":
2         ↪ {
3             "Name": "T1",
4             "EndpointID": "392018
5                 ↪ ee522fd1edbaba3e48481e900f4982ce3805b8ca697a1803640d48cd61
6                 ↪ ",
7             "MacAddress": "02:42:ac:11:00:02",
8             "IPv4Address": "172.17.0.2/16",
9             "IPv6Address": ""
10         },
11     "78c6a45bdeb8df2e39d32f558dae2146f606be41d530c6f9c4c4505107dd27c0":

```

```

9         ↪ {
10         "Name": "T2",
11         "EndpointID": "
12         ↪ ae1a96b1adda93d28e1764af57275fb066ca3397a80821b02843c7b13221cf59
13         ↪ ",
14         "MacAddress": "02:42:ac:11:00:03",
15         "IPv4Address": "172.17.0.3/16",
16         "IPv6Address": ""
17     },
18 ],

```

Do sieci tej zostały podłączone kontenery T1 i T2

Sieć bridge1:

```

1  [
2      {
3          "Name": "bridge1",
4          "Id": "879ac8eafe76f732024be35949cb30d1d2495eb984716aa9f4764964f8683dcb
5          ↪ ",
6          "Created": "2018-11-04T13:06:18.367167366+01:00",
7          "Scope": "local",
8          "Driver": "bridge",
9          "EnableIPv6": false,
10         "IPAM": {
11             "Driver": "default",
12             "Options": {},
13             "Config": [
14                 {
15                     "Subnet": "10.0.10.0/24"
16                 }
17             ]
18         },
19         "Internal": false,
20         "Attachable": false,
21         "Ingress": false,
22         "ConfigFrom": {
23             "Network": ""
24         },
25         "ConfigOnly": false,
26         "Containers": {
27             "1b92a8a5ef4c57e765b1be21b3ad9eb61a5ee958b7dccc344b0db1643ff680a2":
28             ↪ {
29                 "Name": "D1",
30                 "EndpointID": "05
31                 ↪ bac4148c4d5699049d7f76a44931fd12f9d20b0ab1cb96f97828da60460986
32                 ↪ ",
33                 "MacAddress": "02:42:0a:00:0a:03",
34                 "IPv4Address": "10.0.10.3/24",
35                 "IPv6Address": ""
36             },
37             "2722f61dcfa1dbf6acdb53b009e9d31afff617b78993a9b842a3b9b01775dd94":
38             ↪ {

```

```

34         "Name": "D2",
35         "EndpointID": "1
           ↪ cdf370cc1997a892c1b0c2b15cb206b3861e19b7e6930417684cfd4148cc5b8
           ↪ ",
36         "MacAddress": "02:42:0a:00:0a:04",
37         "IPv4Address": "10.0.10.4/24",
38         "IPv6Address": ""
39     },
40     "78c6a45bdeb8df2e39d32f558dae2146f606be41d530c6f9c4c4505107dd27c0":
           ↪ {
41         "Name": "T2",
42         "EndpointID": "0215
           ↪ bcf68c3538e88a1a19085d76d96cbd731baacfd3a408d3984dded28f1d7c
           ↪ ",
43         "MacAddress": "02:42:0a:00:0a:02",
44         "IPv4Address": "10.0.10.2/24",
45         "IPv6Address": ""
46     },
47     "8588ab9134cc76c6c08e866005da526b1f0fe7f8fb00828dc69fb0d2af0bfc3e":
           ↪ {
48         "Name": "late",
49         "EndpointID": "
           ↪ d412021cd143d5a46aac5cf5e1c8b6100912bb62eb143a32174dd8a29a6b4649
           ↪ ",
50         "MacAddress": "02:42:0a:00:0a:05",
51         "IPv4Address": "10.0.10.5/24",
52         "IPv6Address": ""
53     }
54 },
55 "Options": {},
56 "Labels": {}
57 }
58 ]

```

Hosty D1, D2, T2, late połączone z tą siecią, wszystkie mają IP z subnetu 10.0.10.0:

```

1         "1b92a8a5ef4c57e765b1be21b3ad9eb61a5ee958b7dccc344b0db1643ff680a2":
           ↪ {
2         "Name": "D1",
3         "EndpointID": "05
           ↪ bac4148c4d5699049d7f76a44931fd12f9d20b0ab1cb96f97828da60460986
           ↪ ",
4         "MacAddress": "02:42:0a:00:0a:03",
5         "IPv4Address": "10.0.10.3/24",
6         "IPv6Address": ""
7     },
8     "2722f61dcfa1dbbf6acdb53b009e9d31afff617b78993a9b842a3b9b01775dd94":
           ↪ {
9         "Name": "D2",
10        "EndpointID": "1
           ↪ cdf370cc1997a892c1b0c2b15cb206b3861e19b7e6930417684cfd4148cc5b8
           ↪ ",
11        "MacAddress": "02:42:0a:00:0a:04",
12        "IPv4Address": "10.0.10.4/24",
13        "IPv6Address": ""
14    },

```

```

15         "78c6a45bdeb8df2e39d32f558dae2146f606be41d530c6f9c4c4505107dd27c0":
16             ↪ {
17                 "Name": "T2",
18                 "EndpointID": "0215
19                 ↪ bcf68c3538e88a1a19085d76d96cbd731baacfd3a408d3984dded28f1d7c
20                 ↪ ",
21                 "MacAddress": "02:42:0a:00:0a:02",
22                 "IPv4Address": "10.0.10.2/24",
23                 "IPv6Address": ""
24             },
25         "8588ab9134cc76c6c08e866005da526b1f0fe7f8fb00828dc69fb0d2af0bfc3e":
26             ↪ {
27                 "Name": "late",
28                 "EndpointID": "
29                 ↪ d412021cd143d5a46aac5cf5e1c8b6100912bb62eb143a32174dd8a29a6b4649
30                 ↪ ",
31                 "MacAddress": "02:42:0a:00:0a:05",
32                 "IPv4Address": "10.0.10.5/24",
33                 "IPv6Address": ""
34             }
35     },
36 }

```

Część mówiąca o subnet jest tu:

```

1     "Config": [
2         {
3             "Subnet": "10.0.10.0/24"
4         }
5     ]

```

I sieć bridge2:

```

1 [
2     {
3         "Name": "bridge2",
4         "Id": "d529ca748528d5b027e7bdec8991dd707804e24453837e248b9c9de2779df3b5
5         ↪ ",
6         "Created": "2018-11-04T13:06:20.466015092+01:00",
7         "Scope": "local",
8         "Driver": "bridge",
9         "EnableIPv6": false,
10        "IPAM": {
11            "Driver": "default",
12            "Options": {},
13            "Config": [
14                {
15                    "Subnet": "172.21.0.0/16",
16                    "Gateway": "172.21.0.1"
17                }
18            ]
19        },
20        "Internal": false,
21        "Attachable": false,
22        "Ingress": false,

```



```

22     "ConfigFrom": {
23         "Network": ""
24     },
25     "ConfigOnly": false,
26     "Containers": {
27         "2722f61dcfa1dbf6acdb53b009e9d31afff617b78993a9b842a3b9b01775dd94":
28             ↪ {
29                 "Name": "D2",
30                 "EndpointID": "3
31                 ↪ a63c4b1156930b0a8192a50b222a1c2343018b3906d14b7f8ebc0d42d63130b
32                 ↪ ",
33                 "MacAddress": "02:42:ac:15:00:03",
34                 "IPv4Address": "172.21.0.3/16",
35                 "IPv6Address": ""
36             },
37         "4b7872d1506020683a065037eb0bdb0a8eec3171726dcb55df85ab4010f883cf":
38             ↪ {
39                 "Name": "S1",
40                 "EndpointID": "34
41                 ↪ ebe3f0cd9706fcd60ac45eae78acf5f624aa61f2b4e9887630992e55b6c865
42                 ↪ ",
43                 "MacAddress": "02:42:ac:15:00:02",
44                 "IPv4Address": "172.21.0.2/16",
45                 "IPv6Address": ""
46             },
47         "8588ab9134cc76c6c08e866005da526b1f0fe7f8fb00828dc69fb0d2af0bfc3e":
48             ↪ {
49                 "Name": "late",
50                 "EndpointID": "579
51                 ↪ b95924fe8eacd8e39ce3fb31a886caceda41a447078a8e80f90c7750e8fa4
52                 ↪ ",
53                 "MacAddress": "02:42:ac:15:00:04",
54                 "IPv4Address": "172.21.0.4/16",
55                 "IPv6Address": ""
56             }
57     },
58     "Options": {},
59     "Labels": {}
60 }
61 ]

```

Połączone hosty:

```

1     "Containers": {
2         "1b92a8a5ef4c57e765b1be21b3ad9eb61a5ee958b7dccc344b0db1643ff680a2":
3             ↪ {
4                 "Name": "D1",
5                 "EndpointID": "05
6                 ↪ bac4148c4d5699049d7f76a44931fd12f9d20b0ab1cb96f97828da60460986
7                 ↪ ",
8                 "MacAddress": "02:42:0a:00:0a:03",
9                 "IPv4Address": "10.0.10.3/24",
10                "IPv6Address": ""
11            },
12        "2722f61dcfa1dbf6acdb53b009e9d31afff617b78993a9b842a3b9b01775dd94":
13            ↪ {

```

```

10         "Name": "D2",
11         "EndpointID": "1
           ↳ cdf370cc1997a892c1b0c2b15cb206b3861e19b7e6930417684cfd4148cc5b8
           ↳ ",
12         "MacAddress": "02:42:0a:00:0a:04",
13         "IPv4Address": "10.0.10.4/24",
14         "IPv6Address": ""
15     },
16     "78c6a45bdeb8df2e39d32f558dae2146f606be41d530c6f9c4c4505107dd27c0":
           ↳ {
17         "Name": "T2",
18         "EndpointID": "0215
           ↳ bcf68c3538e88a1a19085d76d96cbd731baacfd3a408d3984dded28f1d7c
           ↳ ",
19         "MacAddress": "02:42:0a:00:0a:02",
20         "IPv4Address": "10.0.10.2/24",
21         "IPv6Address": ""
22     },
23     "8588ab9134cc76c6c08e866005da526b1f0fe7f8fb00828dc69fb0d2af0bfc3e":
           ↳ {
24         "Name": "late",
25         "EndpointID": "
           ↳ d412021cd143d5a46aac5cf5e1c8b6100912bb62eb143a32174dd8a29a6b4649
           ↳ ",
26         "MacAddress": "02:42:0a:00:0a:05",
27         "IPv4Address": "10.0.10.5/24",
28         "IPv6Address": ""
29     }
30 },

```

Został przydzielony do niej taki oto subnet i brama domyślna za pomocą której możemy komunikować się z komputerem hosta:

```

1         "Config": [
2             {
3                 "Subnet": "10.0.10.0/24"
4             }
5         ]
6     },

```

Tablica routingu T2:

```

root@78c6a45bdeb8:/# ip route show
default via 172.17.0.1 dev eth1
10.0.10.0/24 dev eth0 proto kernel scope link src 10.0.10.2
172.17.0.0/16 dev eth1 proto kernel scope link src 172.17.0.3
root@78c6a45bdeb8:/# 

```

Tablica routingu D2:

```
root@2722f61dcfa1:/usr/local/tomcat# ip route show
default via 10.0.10.1 dev eth1
10.0.10.0/24 dev eth1 proto kernel scope link src 10.0.10.4
172.21.0.0/16 dev eth0 proto kernel scope link src 172.21.0.3
```

2.1.5 Pytanie pierwsze:

Poniższy screen przedstawia, jak komunikat 'Hello' poprawnie został przesłany z kontenera D2 na host posługując się programem 'netcat'. Komunikacja przebiegała przez port 9000:

```
10.0.10.0/24 dev eth1 proto kernel scope link src 10.0.10.4
172.21.0.0/16 dev eth0 proto kernel scope link src 172.21.0.3
root@2722f61dcfa1:/usr/local/tomcat# nc 172.21.0.1 9000
Hello from D2

root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab3] on
git:master x 2365882 "Lab5 ISM done"
14:32:34 > nc -l -p 9000 0.0.0.0
Hello from D2
```

To samo dla S1 -> host:

```
root@4b7872d15060:/# nc 172.21.0.1 9000
(UNKNOWN) [172.21.0.1] 9000 (?): Connection refused
root@4b7872d15060:/# nc 172.21.0.1 9000
Hello from S1

root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab3] on
git:master x 2365882 "Lab5 ISM done"
14:39:05 > nc -l -p 9000 0.0.0.0
Hello from S1
```

Podpunkt a:

Możliwym jest przekazanie parametru '-ip-range' przy poleceniu 'docker network create'. Np '-ip-range=172.28.5.0/24'

2.2 Zadanie drugie

Po uruchomieniu zalinkowanych kontenerów zawartość zmiennych systemowych w T1 prezentuje się tak:

```
1 BASH=/bin/bash
2 BASHOPTS=cmdhist:complete_fullquote:extquote:force_ignores:hostcomplete:
   ↪ interactive_comments:progcomp:promptvars:sourcepath
3 BASH_ALIASES=()
4 BASH_ARGC=()
5 BASH_ARGV=()
6 BASH_CMDS=()
7 BASH_EXECUTION_STRING=set
8 BASH_LINENO=()
9 BASH_SOURCE=()
10 BASH_VERSINFO=( [0]="4" [1]="4" [2]="12" [3]="1" [4]="release" [5]="x86_64-pc-
   ↪ linux-gnu" )
11 BASH_VERSION='4.4.12(1)-release'
12 DIRSTACK=()
13 EUID=0
14 GROUPS=()
15 HOME=/root
16 HOSTNAME=1bc816f34352
17 HOSTTYPE=x86_64
18 IFS=$' \t\n'
19 MACHTYPE=x86_64-pc-linux-gnu
20 MYLINK_ENV_NGINX_VERSION=1.15.5-1~stretch
21 MYLINK_ENV_NJS_VERSION=1.15.5.0.2.4-1~stretch
22 MYLINK_NAME=/T1/mylink
23 MYLINK_PORT=tcp://172.17.0.2:80
```

```

24 MYLINK_PORT_8000_TCP=tcp://172.17.0.2:8000
25 MYLINK_PORT_8000_TCP_ADDR=172.17.0.2
26 MYLINK_PORT_8000_TCP_PORT=8000
27 MYLINK_PORT_8000_TCP_PROTO=tcp
28 MYLINK_PORT_80_TCP=tcp://172.17.0.2:80
29 MYLINK_PORT_80_TCP_ADDR=172.17.0.2
30 MYLINK_PORT_80_TCP_PORT=80
31 MYLINK_PORT_80_TCP_PROTO=tcp
32 NGINX_VERSION=1.15.5-1~stretch
33 NJS_VERSION=1.15.5.0.2.4-1~stretch
34 OPTERR=1
35 OPTIND=1
36 OSTYPE=linux-gnu
37 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
38 PPID=0
39 PS4='+ '
40 PWD=/
41 SHELL=/bin/bash
42 SHELLOPTS=braceexpand:hashall:interactive-comments
43 SHLVL=1
44 TERM=xterm
45 UID=0
46 _=bash

```

Te zmienne używane są przez kod linkujący:

```

1 MYLINK_ENV_NGINX_VERSION=1.15.5-1~stretch
2 MYLINK_ENV_NJS_VERSION=1.15.5.0.2.4-1~stretch
3 MYLINK_NAME=/T1/mylink
4 MYLINK_PORT=tcp://172.17.0.2:80
5 MYLINK_PORT_8000_TCP=tcp://172.17.0.2:8000
6 MYLINK_PORT_8000_TCP_ADDR=172.17.0.2
7 MYLINK_PORT_8000_TCP_PORT=8000
8 MYLINK_PORT_8000_TCP_PROTO=tcp
9 MYLINK_PORT_80_TCP=tcp://172.17.0.2:80
10 MYLINK_PORT_80_TCP_ADDR=172.17.0.2
11 MYLINK_PORT_80_TCP_PORT=80
12 MYLINK_PORT_80_TCP_PROTO=tcp

```

A tak prezentuje się plik `/etc/hosts`:

```

1 127.0.0.1      localhost
2 ::1           localhost ip6-localhost ip6-loopback
3 fe00::0       ip6-localnet
4 ff00::0       ip6-mcastprefix
5 ff02::1       ip6-allnodes
6 ff02::2       ip6-allrouters
7 172.17.0.2     mylink 877b2110d546 T2
8 172.17.0.3     1bc816f34352

```

Linker oczywiście używa wpisu z 7. linii.

2.2.1 Pytanie pierwsze

Ping z maszyny T1 na T2 jest możliwy, T1 ma też odpowiedni wpis w `/etc/hosts`. Natomiast ping z T2 na T1 kończy się niepowodzeniem. Tam też nie ma wpisu w `/etc/hosts` wskazującego na kontener T1. Potwierdza to poniższy zrzut ekranu:

```

root@1bc816f34352:/# ping T2
PING mylink (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: icmp_seq=0 ttl=64 time=0.249 ms
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.141 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.157 ms
^C--- mylink ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.141/0.182/0.249/0.048 ms
root@1bc816f34352:/#

```

2.2.2 Pytanie drugie

Tryb sieci host:

Docker odmawia utworzenia kontenera z linkiem do innego kontenera. W tym trybie kontener korzysta z natywnego stosu sieciowego hosta, a nie z oddzielnych sieci, nie ma więc czego linkować ze sobą.

```

$ docker run -itd --name bar --net host --link foo:mylink nginx bash
docker: Error response from daemon: conflicting options: host type networking can't be used with links. This would result in undefined behavior.

```

User defined bridge:

Polecenie tworzenia linków działa bez zarzutów. Jest ono jednak zbędne, bo w tym trybie urządzenia i tak widzą się całkowicie, potwierdza to ping z foo na bar:

```

root@db01253f32a0:/# ping foo
PING foo (10.0.10.2): 56 data bytes
64 bytes from 10.0.10.2: icmp_seq=0 ttl=64 time=0.235 ms
64 bytes from 10.0.10.2: icmp_seq=1 ttl=64 time=0.156 ms
^C--- foo ping statistics ---

```

Plik hosts w foo:

```

root@db01253f32a0:/# cat /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
10.0.10.3       db01253f32a0
root@db01253f32a0:/# ^C

```

I hosts w bar:

```
root@adaf84d10aa7:/# cat /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
10.0.10.2     adaf84d10aa7
root@adaf84d10aa7:/#
```

2.2.3 Pytanie trzecie:

Utworzyłem kontener 'bar' w domyślnej sieci i 'foo' w sieci user-defined-brige. Po stworzeniu foo z linkiem do bar plik '/etc/hosts' w 'foo' nie posiada wymaganych wpisów:

```
root@3c5844589a3d:/# cat /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
10.0.10.2     3c5844589a3d
```

Ping foo -> bar też nie działa.

```
root@3c5844589a3d:/# ping bar
ping: unknown host
root@3c5844589a3d:/#
```

Próba stworzenia linku w trybie sieci host kończy się takim samym błędem jak wcześniej. Wniosek - nie da się.

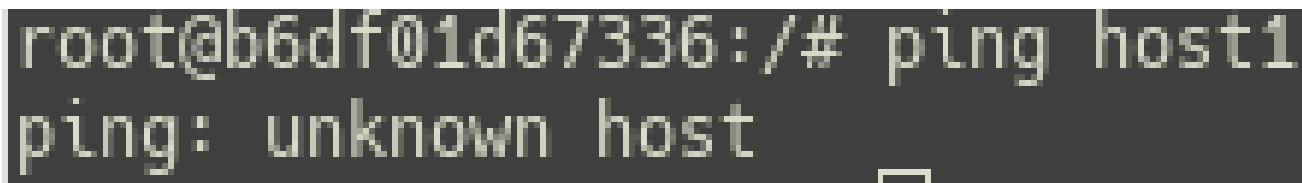
2.3 Zadanie trzecie:

Skrypt zmodyfikowany o użycie aliasów:

```
1 sysctl net.ipv4.conf.all.forwarding=1
2 sudo iptables -P FORWARD ACCEPT
3 # Dzięki tym regulom możliwa jest komunikacja kontener -> host
4
5 docker pull alpine
6 docker pull ubuntu
7 docker pull nginx
8 docker pull tomcat
9
10 docker run -itd --name T1 alpine sh
11
12 docker network create -d bridge --subnet 10.0.10.0/24 bridge1
13 docker run -itd --name T2 --net bridge --expose 80 -p 80:80 -p
    ↪ 10.0.10.1:8000:80 nginx sh
14 docker network connect bridge1 T2
15
16 docker exec -itd T2 /bin/bash -c 'service nginx start'
17
18 docker run -itd --name D1 --net bridge1 --network-alias host1 alpine sh # tu
    ↪ tez ustawienie aliasu, od teraz można się w obrębie sieci bridge1
    ↪ komunikować z D1 za pomocą aliasu host1
19
20 docker network create -d bridge bridge2
21 docker run -itd --net bridge2 --network-alias host2 --name S1 ubuntu sh #
    ↪ ustawiamy tu alias
22
23 docker run -itd --name D2 --expose 8080 --net bridge2 --network-alias apa2 -p
    ↪ 8081:8080 -p 10.0.10.0:8080:8080 tomcat sh # ustawiamy tu alias do
    ↪ komunikacji z D2 w obrębie sieci 'bridge2'
24 docker network connect --alias apa1 bridge1 D2 # a tu alias do komunikacji w
    ↪ obrębie sieci bridge1
25
26 docker exec D2 /bin/bash -c '/usr/local/tomcat/bin/startup.sh'
27
28 docker run -itd --name late --net bridge2 ubuntu bash
29 docker network connect bridge1 late
```

2.3.1 Pytanie pierwsze:

Nie, taka komunikacja nie jest możliwa, co widać na poniższym screenie, ping host2 -> host1 się nie powiodł.



```
root@b6df01d67336:/# ping host1
ping: unknown host
```

Takie aliasy działają tylko w obrębie tej samej sieci. Linux udostępnia konstrukcję zwaną "network namespace", który umożliwia oddzielenie od siebie logicznych fragmentów sieci i zdefiniowanie dla nich zupełnie niezależnych reguł, aliasów itd., z czego korzysta też Docker.

2.3.2 Pytanie drugie:

Aby korzystać z tych aliasów trzeba by zmodyfikować `'/etc/hosts'` na komputerze macierzystym, czego Docker nie robi podczas tworzenia tych kontenerów, możliwym też by była translacja nazw za pomocą pośredniego serwera DNS, co nie zostało skonfigurowane. Więc - domyślnie nie jest to możliwe od razu.