Container Network

Saturday, April 27, 2024 3:01 PM

Lab-1: how to create custom bridge Network

```
 3  docker network ls
 4  ip a s
 5  docker network create  --subnet  192.168.16.0/24 --driver bridge  front
 6  docker network create  --subnet  192.168.17.0/24 --driver bridge  backend
 7  docker rm -f $(docker ps -q -a)
 8  docker run -itd  --name=db  --network  backend  alpine
 9  docker exec -it  db  bash
10  docker exec -it  db  sh
11  ifconfig
12  apt install net-tools
13  ifconfig
14  docker run -itd  --name=app   --network  backend  alpine
15  docker network connect front app
16  docker exec -it app bash
17  docker exec -it app sh
18  docker run -itd  --name=web    --network  front   alpine
19  docker  inspect  db |  grep -i  ipaddress
20  docker exec -it web sh
21  docker  network connect  front db
22  docker exec -it web sh
23  docker  network connect  front db
24  docker  inspect  db |  grep -i  ipaddress
25  docker exec -it web sh
26  docker network disconnect  front  db
27  docker network disconnet front app
28  docker network disconnect front app
29  docker  network inspect  front |  grep -i name
30  docker  network inspect  backend  |  grep -i name
31  docker rm -f $(docker ps -q -a)
32  docker network rm front
33  docker network ls
34  docker network prune
```

4) How to access container from External World.

```
48  docker run -d --name=web1   -p  8888:80  nginx
 49  ifconfig
 50  netstat -tunlp
 51  docker run -d --name=web1   -p  8881:80  nginx
 52  docker run -d --name=web2   -p  8881:80  nginx
 53  docker ps
 54  docker exec -it web1 bash
 55  docker exec -it web2 bash
 56  docker exec -it web1 bash
 57  iptables-save  >  rule-save
```

Static Port forwarding

Dynamic :

Name Discovery by defaulted in the Bride network.

```
@DockerHost:~#
root@DockerHost:~# docker run -itd --name=con1  --network backend  alpine
285eafccb975cb673cc468b9e663632f3e7088a8c2f127b2dc1e8fe5d5fc89c3
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~# docker  inspect con1 | grep -i ipaddress
        "SecondaryIPAddresses": null,
        "IPAddress": "",
            "IPAddress": "192.168.17.2",
root@DockerHost:~#
root@DockerHost:~# docker stop con1
con1
root@DockerHost:~#
root@DockerHost:~# docker run -itd --name=con2   --network backend  alpine
9fdd87d1bbff9c160331c002b8a7dbfa9f2faee7659f075ab66a14174837c10b
root@DockerHost:~#
root@DockerHost:~# docker inspect con2 |  grep -i ipaddress
        "SecondaryIPAddresses": null,
        "IPAddress": "",
            "IPAddress": "192.168.17.2",
root@DockerHost:~# docker start con1
con1
```

```
root@DockerHost:~# docker inspect con1 |  grep -i ipaddress
        "SecondaryIPAddresses": null,
        "IPAddress": "",
              "IPAddress": "192.168.17.3",
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~# docker exec -it con1 sh
/ # ping con2
PING con2 (192.168.17.2): 56 data bytes
64 bytes from 192.168.17.2: seq=0 ttl=64 time=1.363 ms
64 bytes from 192.168.17.2: seq=1 ttl=64 time=0.155 ms
64 bytes from 192.168.17.2: seq=2 ttl=64 time=0.109 ms
64 bytes from 192.168.17.2: seq=3 ttl=64 time=0.100 ms
64 bytes from 192.168.17.2: seq=4 ttl=64 time=0.105 ms
64 bytes from 192.168.17.2: seq=5 ttl=64 time=0.112 ms
64 bytes from 192.168.17.2: seq=6 ttl=64 time=0.088 ms
64 bytes from 192.168.17.2: seq=7 ttl=64 time=0.083 ms
64 bytes from 192.168.17.2: seq=8 ttl=64 time=0.102 ms
64 bytes from 192.168.17.2: seq=9 ttl=64 time=0.110 ms
64 bytes from 192.168.17.2: seq=10 ttl=64 time=0.147 ms
64 bytes from 192.168.17.2: seq=11 ttl=64 time=0.129 ms
64 bytes from 192.168.17.2: seq=12 ttl=64 time=0.109 ms
64 bytes from 192.168.17.2: seq=13 ttl=64 time=0.134 ms
64 bytes from 192.168.17.2: seq=14 ttl=64 time=0.120 ms
64 bytes from 192.168.17.2: seq=15 ttl=64 time=0.131 ms
64 bytes from 192.168.17.2: seq=16 ttl=64 time=0.126 ms
64 bytes from 192.168.17.2: seq=17 ttl=64 time=0.110 ms
64 bytes from 192.168.17.2: seq=18 ttl=64 time=0.145 ms
^C
--- con2 ping statistics ---
19 packets transmitted, 19 packets received, 0% packet loss
round-trip min/avg/max = 0.083/0.183/1.363 ms
/ # exit
```

1) Host Network:

container directl get base machine network acces, it does get seprate ipaddress . Base machine all ip become contains ips.

Cons:

can not run more than if they use same port number

1) None

Mac vlan:

```
docker network create -d macvlan \
  --subnet=172.16.86.0/24 \
  --gateway=172.16.86.1 \
  -o parent=eth0 pub_net
```

From <https://docs.docker.com/network/drivers/macvlan/>

```
root@DockerHost:~# docker network prune
WARNING! This will remove all custom networks not used by at least one container.
Are you sure you want to continue? [y/N] y
Deleted Networks:
backend

root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.88.2    0.0.0.0         UG    100    0        0 ens33
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
192.168.88.0    0.0.0.0         255.255.255.0   U     100    0        0 ens33
192.168.88.2    0.0.0.0         255.255.255.255 UH    100    0        0 ens33
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~# docker network create -d macvlan \
  --subnet=172.16.86.0/24 \
  --gateway=172.16.86.1 \
  -o parent=eth0 pub_net
Error response from daemon: invalid subinterface vlan name eth0, example formatting is eth0.10
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~# docker network create --driver  macvlan   --subnet=192.168.88.0/24   --gateway=192.168.88.2    -o
parent=ens3     3   physnet
a30bb72a7bc86ff508af134e0972bad3fc27409d8c595e4aa925126ba3c6c43c
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~# docker run -itd --name=server1 --network physnet  alpine
f7143ceac32dffffedbaed913817c12a2f2cebf67a2acf7aba99c038d4942568
root@DockerHost:~#
root@DockerHost:~# docker exec -it  server1  sh
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:C0:A8:58:01
```

```
        inet addr:192.168.88.1  Bcast:192.168.88.255  Mask:255.255.255.0
        inet6 addr: fe80::42:c0ff:fea8:5801/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B)  TX bytes:656 (656.0 B)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        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)

/ # read escape sequence
root@DockerHost:~#
root@DockerHost:~# docker run -itd --name=server2 --network physnet  alpine
root@DockerHost:~# docker run -itd --name=server2 --network physnet  alpine
fc42dcb9906e27787d61e9d66889c503e7161c4eef233a339af8a535be4cfe40
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~#
root@DockerHost:~# docker exec -it  server2  sh
/ #
/ # ifconfig
eth0    Link encap:Ethernet  HWaddr 02:42:C0:A8:58:03
        inet addr:192.168.88.3  Bcast:192.168.88.255  Mask:255.255.255.0
        inet6 addr: fe80::42:c0ff:fea8:5803/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:4 errors:0 dropped:0 overruns:0 frame:0
        TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:240 (240.0 B)  TX bytes:516 (516.0 B)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        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
root@DockerHost:~# docker run -itd --name=server3 --network physnet  nginx
ac4d390daf0ee746a18f2493aa5138b5d4cbc89d93a329c8459db343ac26bf93
root@DockerHost:~#
root@DockerHost:~# docker inspect server3 |  grep -i ipaddress
          "SecondaryIPAddresses": null,
          "IPAddress": "",
                "IPAddress": "192.168.88.4",
root@DockerHost:~#
```

```
root@DockerHost:~# docker inspect server3 |  grep -i ipaddress
root@DockerHost:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
ac4d390daf0e   nginx     "/docker-entrypoint.…"   14 minutes ago   Up 14 minutes            server3
fc42dcb9906e   alpine    "/bin/sh"                15 minutes ago   Up 15 minutes            server2
f7143ceac32d   alpine    "/bin/sh"                16 minutes ago   Up 16 minutes            server1
root@DockerHost:~# docker inspect server1  |  grep -i mac
        "MacAddress": "",
            "MacAddress": "02:42:c0:a8:58:01",
root@DockerHost:~#
root@DockerHost:~# docker inspect server2  |  grep -i mac
        "MacAddress": "",
            "MacAddress": "02:42:c0:a8:58:03",
root@DockerHost:~#
root@DockerHost:~# docker inspect server3   |  grep -i mac
        "MacAddress": "",
            "MacAddress": "02:42:c0:a8:58:04",
root@DockerHost:~# history
   1  clear
   2  ifconfig
   3  ip a s
   4  docker run -d --name=web2   192.168.88.133:8883:80   nginx
   5  docker run -d --name=web2   -p  192.168.88.133:8883:80   nginx
   6  docker rm -f $(docker ps -q -a)
   7  clear
   8  docker network prune
   9  route -n
  10  docker network create -d macvlan   --subnet=172.16.86.0/24   --gateway=172.16.86.1   -o parent=eth0 pub_net
  11  docker network create --driver  macvlan   --subnet=192.168.88.0/24   --gateway=192.168.88.2    -o parent=ens33
physnet
  12  docker run -itd --name=server1 --network physnet  alpine
  13  docker exec -it  server1  sh
  14  docker run -itd --name=server2 --network physnet  alpine
  15  docker exec -it  server2  sh
  16  docker run -itd --name=server3 --network physnet  nginx
  17  docker inspect server3 |  grep -i ipaddress
  18  docker ps
  19  docker inspect server1  |  grep -i mac
  20  docker inspect server2  |  grep -i mac
  21  docker inspect server3   |  grep -i mac
  22  history
root@DockerHost:~# docker rm -f $(docker ps -q -a)
ac4d390daf0e
fc42dcb9906e
f7143ceac32d
root@DockerHost:~#
root@DockerHost:~# docker network prune
WARNING! This will remove all custom networks not used by at least one container.
Are you sure you want to continue? [y/N] y
Deleted Networks:
physnet

root@DockerHost:~# docker network create --driver ipvlan    --subnet=192.168.88.0/24   --gateway=192.168.88.2    -o
parent=ens33        physnet
939e7a6b1cabfe20e2c9cd3d1dfabcf88bc656c745645bd20d408ee5e976abbb
```

```
root@DockerHost:~#
root@DockerHost:~# history
    1  clear
    2  ifconfig
    3  ip a s
    4  docker run -d --name=web2   192.168.88.133:8883:80   nginx
    5  docker run -d --name=web2   -p  192.168.88.133:8883:80   nginx
    6  docker rm -f $(docker ps -q -a)
    7  clear
    8  docker network prune
    9  route -n
   10  docker network create -d macvlan   --subnet=172.16.86.0/24   --gateway=172.16.86.1   -o parent=eth0 pub_net
   11  docker network create --driver  macvlan   --subnet=192.168.88.0/24   --gateway=192.168.88.2    -o parent=ens33
physnet
   12  docker run -itd --name=server1 --network physnet  alpine
   13  docker exec -it  server1  sh
   14  docker run -itd --name=server2 --network physnet  alpine
   15  docker exec -it  server2  sh
   16  docker run -itd --name=server3 --network physnet  nginx
   17  docker inspect server3 |  grep -i ipaddress
   18  docker ps
   19  docker inspect server1  |  grep -i mac
   20  docker inspect server2  |  grep -i mac
   21  docker inspect server3   |  grep -i mac
   22  history
   23  docker rm -f $(docker ps -q -a)
   24  docker network prune
   25  docker network create --driver ipvlan    --subnet=192.168.88.0/24   --gateway=192.168.88.2    -o parent=ens33
physnet
   26  history
root@DockerHost:~#
```