Docker task 5 (network customization)

1. sudo docker network create my_net1

   output :

   2a63ef56bd174544d9c382c4f40917cab0c77570c4fd14621a0ab976d613b1fa
   (This command is used to create a custom network with our desired network name)
2. sudo docker network ls

   output :

   NETWORK ID NAME DRIVER SCOPE
   4c887c280368 bridge bridge local
   3bc05a0ac7ef host host local
   2a63ef56bd17 my_net1 bridge local
   e27977a07865 none null local
   (This command is used to list all the docker networks available in our local machine)
3. sudo docker run -dit --name container1 --network my_net1 ubuntu

   output :

   d16376927c4c71ed0be21129a6057fcf8ccbdb431c3d7cc55cf4b91ccaf4f4c0
   (This command is used to create a new container and attach the container with our custom network)
4. sudo docker container1 inspect

   output : It shows many info but we spefically focus here on network section to inspect the wheather it uses our attached custom network (my_net1), also we can check the gateway address, ip address of the container through this command

```
"Networks": {
       "my_net1": {
           "IPAMConfig": null,
           "Links": null,
           "Aliases": null,
           "MacAddress": "02:42:ac:12:00:02",
           "DriverOpts": null,
   "NetworkID":
   "2a63ef56bd174544d9c382c4f40917cab0c77570c4fd14621a0ab976d613b1fa",
   "EndpointID":
   "ba4bb9f970616d087b57215e5e79bf21384a8c1eb5be4a747c8a599a100b09b9",
           "Gateway": "172.18.0.1",
           "IPAddress": "172.18.0.2",
```

```
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "DNSNames": [
                "container1",
                "d16376927c4c"
```

Let's create another new container with the same network, then try to ping with previous container made of same network..

5. sudo docker run -it --name container2 --network my_net1 ubuntu

   root@0a2b48533394:/# apt update && apt install -y iputils-ping
   (This command is used to install packages of iputils-ping, so that the ping cmd will work inside the container)

   root@0a2b48533394:/# ping container1
   PING container1 (172.18.0.2) 56(84) bytes of data.
   64 bytes from container1.my_net1 (172.18.0.2): icmp_seq=1 ttl=64 time=0.060 ms
   64 bytes from container1.my_net1 (172.18.0.2): icmp_seq=2 ttl=64 time=0.044 ms
   64 bytes from container1.my_net1 (172.18.0.2): icmp_seq=3 ttl=64 time=0.049 ms...

   (Both containers pinged with eachother because both use same network)

Now let's see brief on the concept of subnet creation and gateway,

6. sudo docker network create --subnet 172.21.0.0/16 my_subnet_net1

```
output :

 0a34222fe0dbaf9ad405d3d633878b2dc2431f8f87021a5af408d19d6396676d

This command is used to create our own custom network with setting our own
subnet...

  In realtime the ISP provider provides a public IP with customized subnet range
  according to the need of the customer, through subnet the ip is seperated and
  given for each catogeries such as for network address,broadcast
  address,gateway address and for client systems.

   In docker, docker itself provide an ip with subnet range defaultly for our
   custom network, but by using this subnet command we can customize the our
```

```
  custom network's subnet range and docker network's private ip address
  according to our requirement
```

Let's create two containers with same volume but with different networks and check the volume's data presence whether it shares the data stored in it with both the containers...

sudo docker run -dit --name container3 --network my_subnet_net1 -v myvolume:/home/data ubuntu

```
 output :
 baadc97be7bdb28e497bd6dae04d3c73f04d6224df324fe6f6177b585a18fec7
```

sudo docker run -dit --name container4 --network my_net1 -v myvolume:/home/info ubuntu

```
output :
 6e0eb31cf8fe2883eb3dcc271309b524a0bb71108e28a15cec0d0e3bc7df56a0
```

sudo docker exec -it container3 bash

```
output :
 root@baadc97be7bd:/# cd /home/data
 root@baadc97be7bd:/home/data# touch file1.txt file2.txt
```

Press ctrl+p followed by ctrl+q, to exit container without pushing the container into its exit state,

sudo docker exec -it container4 bash

```
output :
root@6e0eb31cf8fe:/# cd /home/info
root@6e0eb31cf8fe:/home/info# ls
file1.txt  file2.txt  file3.txt



 Explanation :

 Docker volumes are shared across containers regardless of their network
 configurations, so the data stored in the shared volume should be visible in
 both containers even if they are on different networks
```