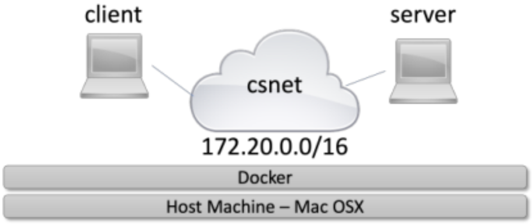
Contents

1 Docker Walkthrough

Date: 12-09-2022
Note taker: Silent

1.1 Introduction

- Docker use is optional for assignments
  - Docker Playground in web-browser possible
- Docker provides:
  - Implementation of container
  - virtualisation abstraction



1.2 Overview on Installation of Docker

1. Install Docker
   - Linux users must also remember to enable the service and start it
2. Create a network
3. Create two containers
4. Connect the containers to the network
5. Start containers and execute programmes

1.2.1 Installing Docker on Windows/Mac

- Docker Desktop

1.2.2 Installing Docker on Linux

- Docker
  - Debian based
  - Arch based
    - `sudo pacman -S yay base-devel`
    - `yay -S docker-git`
  - Gentoo based
    - `sudo emerge app-containers/docker app-containers/docker-cli`

1.2.2.1 Systemd based - Debian/Arch etc

- `sudo systemctl enable docker`
- `sudo systemctl start docker`

1.2.2.2 Init Systems

1.2.2.2.1 OpenRC - Gentoo

- `sudo rc-update add docker`
- `sudo rc-service docker start`
- If encountering a crash from docker, manually solve it by `sudo rc-service docker zap`

1.2.2.3 User permisions

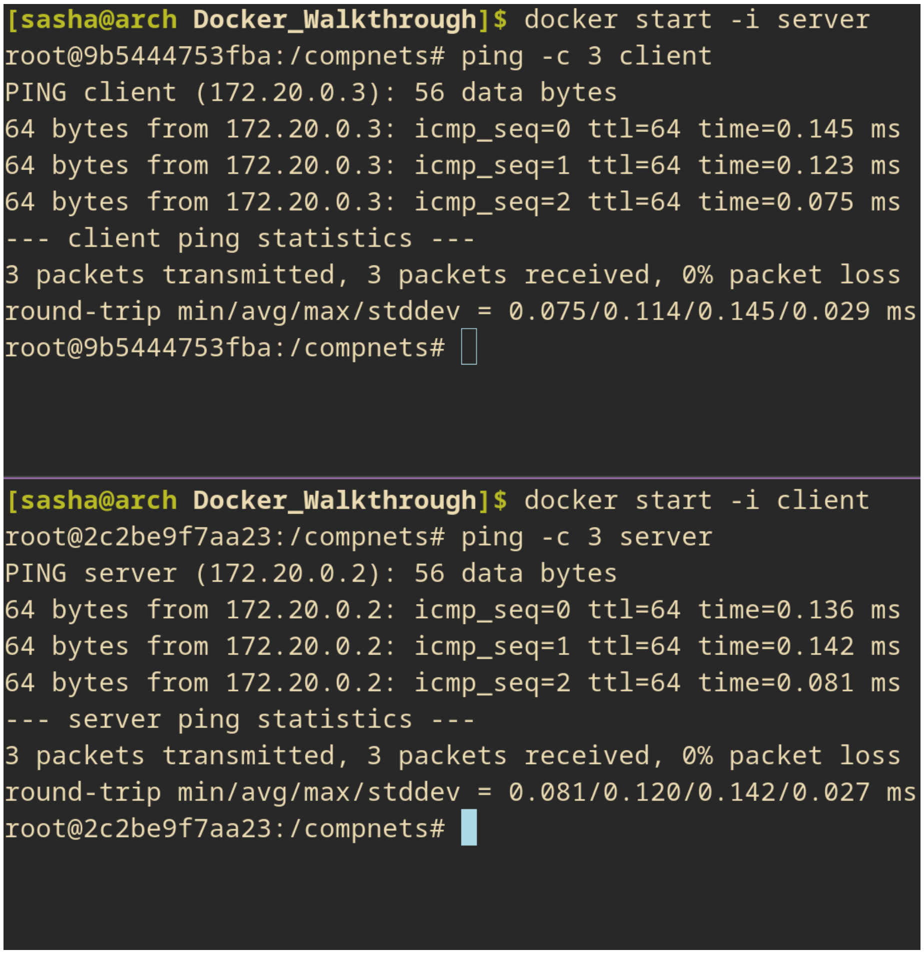
- `sudo usermod -aG docker <username>`

1.2.3 Creating a Network & Containers

1. Open a terminal
2. Create a small bridged network called *csnet*
   - IPv4 network with range 172.20.0.0 - 172.20.255.255
   - `docker network create -d bridge --subnet 172.20.0.0/16 csnet`
3. Create container image named csnetimage based on Dockerfile present in your current working directory
   - `docker build -t csnetimage .`
4. Create container called *client*
   - `docker create -ti --name client --cap-add=all -v ~/compnets:/compnets csnetimage /bin/bash`
5. Create container called *server*
   - `docker create -ti --name server --cap-add=all -v ~/compnets:/compnets csnetimage /bin/bash`
6. Connect container called *client* to csnet
   - `docker network connect csnet client`
7. Connect container called *server* to csnet
   - `docker network connect csnet server`

1.3 Starting & Pinging Containers

- You should be able to start the containers and test some basic communication between them.
- Let us ping the server and client
  1. To start your client run on a separate terminal
     `docker start -i server`
  2. To start your server run on a separate terminal
     `docker start -i client`
  3. Ping the client 3 times
     `ping -c 3 client`
  4. Ping the server 3 times
     `ping -c 3 server`
  5. If all is successful you should have something similar to what you see in the image below



```
[sasha@arch Docker_Walkthrough]$ docker start -i server
root@9b5444753fba:/compnets# ping -c 3 client
PING client (172.20.0.3): 56 data bytes
64 bytes from 172.20.0.3: icmp_seq=0 ttl=64 time=0.145 ms
64 bytes from 172.20.0.3: icmp_seq=1 ttl=64 time=0.123 ms
64 bytes from 172.20.0.3: icmp_seq=2 ttl=64 time=0.075 ms
--- client ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.075/0.114/0.145/0.029 ms
root@9b5444753fba:/compnets#
```

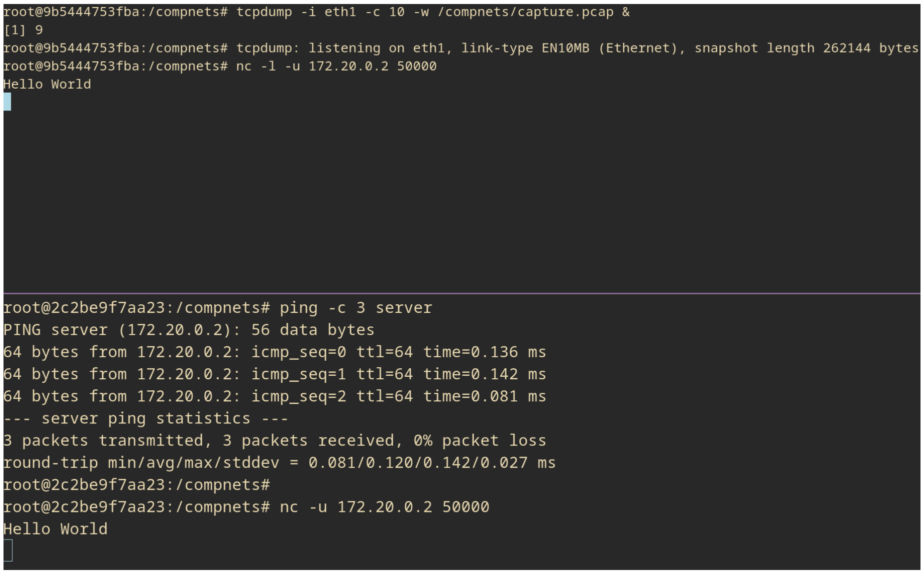
```
[sasha@arch Docker_Walkthrough]$ docker start -i client
root@2c2be9f7aa23:/compnets# ping -c 3 server
PING server (172.20.0.2): 56 data bytes
64 bytes from 172.20.0.2: icmp_seq=0 ttl=64 time=0.136 ms
64 bytes from 172.20.0.2: icmp_seq=1 ttl=64 time=0.142 ms
64 bytes from 172.20.0.2: icmp_seq=2 ttl=64 time=0.081 ms
--- server ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.081/0.120/0.142/0.027 ms
root@2c2be9f7aa23:/compnets#
```

1.4 Starting & Executing Programmes

- Programme used to demonstrate communication between containers is a versatile tool called `netcat` or `nc`
- Server will start netcat to listen for incoming UDP datagrams on a given IP address and port number.
  - Port number `50000` will be used since it is udp protocol
- The client will send what is typed in the terminal after the start of netcat to the IP address and port number as UDP datagram
- Run two sides of netcat, one acting as a server listening to a given IP address and port number, and the other acting as a client, sending what is typed in the client terminal to the server.
- By starting two interactive containers in individual terminals you will be able to see the programmes executing as if they were executed on two individual machines.
- These commands will allow for that
  1. `docker start -i server`
  2. On the same terminal write
     `nc -l -u 172.20.0.2 50000`
  3. On a new terminal
     `docker start -i client`
  4. On the same terminal write
     `nc -u 172.20.0.2 50000`
  5. Down below you will see capturing traffic using tcdump will allow us to see input from terimnal 2 to be outputed to both terminals

1.5 Capturing Traffic using tcpdump

- tcpdump is used to capture packets
- Running `tcpdump -D` will output a numbered list. We will be using eth1 in our case.
- Let's capture 10 packets from eth1, In order to be allowed this, the parameter `--cap-add=all` allows the continer to use capabilities in linux to capture traffic in containers
  1. On terminal 1 enter:
     `docker start -i server`
  2. Once entered container:
     `tcpdump -i eth1 -c 10 -w /compnets/capture.pcap &`
  3. You should now be listening on eth1, then execute:
     `nc -l -u 172.20.0.2 50000`
  4. On terminal 2 enter:
     `docker start -i client`
  5. Once entered container:
     `nc -u 172.20.0.2 50000`
  6. All packets from tcpdump will be saved to a file "/compnets/capture.pcap"
  7. If you type on terminal 2,
     `Hello World` and then click enter. You should be able to see that your sever has capture that packet as well. Reference image below



```
root@9b5444753fba:/compnets# tcpdump -i eth1 -c 10 -w /compnets/capture.pcap &
[1] 9
root@9b5444753fba:/compnets# tcpdump: listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
root@9b5444753fba:/compnets# nc -l -u 172.20.0.2 50000
Hello World
```

```
root@2c2be9f7aa23:/compnets# ping -c 3 server
PING server (172.20.0.2): 56 data bytes
64 bytes from 172.20.0.2: icmp_seq=0 ttl=64 time=0.136 ms
64 bytes from 172.20.0.2: icmp_seq=1 ttl=64 time=0.142 ms
64 bytes from 172.20.0.2: icmp_seq=2 ttl=64 time=0.081 ms
--- server ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.081/0.120/0.142/0.027 ms
root@2c2be9f7aa23:/compnets#
root@2c2be9f7aa23:/compnets# nc -u 172.20.0.2 50000
Hello World
```

1.6 Capturing Traffic using Wireshark

- Wireshark can be executed directly on containers
  - On Windows:
    - On the server docker type install wireshark:

      ```
      docker start -i server
      apt install -y wireshark
      ```

      Make sure to type yes, for when it asks at the end of install.
    - Make a dir and give yourself permissions:

      ```
      mkdir /tmp/foobar
      chmod 700 /tmp/foobar
      ```
    - Export your environment variables:

      ```
      export DISPLAY=host.docker.internal:0
      export LIBGL_ALWAYS_INDIRECT=1
      export XDG_RUNTIME_DIR=/tmp/foobar
      ```
    - Open wireshark:

      ```
      wireshark -i -u 172.20.0.2 50000 &
      ```
    - Start capturing packets:

      ```
      nc -l -u 172.20.0.2 50000
      ```
    - On a separate terminal, open the client docker with:

      ```
      docker start -i client
      nc -u 172.20.0.0.2 50000
      ```
    - If you type on the client and click enter, e.g.:

      ```
      test
      ```
    - Wireshark will now update to make the packet visible
- Wireshark alternative (terminal user interface tui)
  - Reference: tshark
  - Use termshark, it's like wireshark but on a terminal

1.6.1 Setting up termshark

- This is optional, if you have wireshark, you can skip this step

1. On terminal 1 enter:
   `docker start -i server`
2. You should now be listening on eth1, then execute so it runs on the background:
   `nc -l -u 172.20.0.2 50000 &`
3. Fix you TERM variable with:
   `export TERM=screen-256-color`
4. Run:
   `term shark -i eth1`
5. On terminal 2 enter:
   `docker start -i client`
6. Once entered container:
   `nc -u 172.20.0.2 50000`
7. If you type on terminal 2,
   `Hello World` and then click enter. You should be able to see that your sever has capture that packet as well. Reference image below
8. You should now see terminal 1 have termshark open. Use arrow keys to move around termshark