

#015 Docker network

Introduction

this is part 15 from the journey it's a long journey(360 day) so go please check previous parts , and if you need to walk in the journey with me please make sure to follow because I may post more than once in 1 Day but surely I will post daily at least one 😊.

And I will cover lot of tools as we move on.

Download app_015

```
(base) in ~  
> cd Documents/DevOpsJourney/app_013/  
(base) (master)in ~/Documents/DevOpsJourney/app_013  
> 1  
Dockerfile app.py requirements.txt  
(base) (master)in ~/Documents/DevOpsJourney/app_013  
>
```

if you follow [part 9](#)

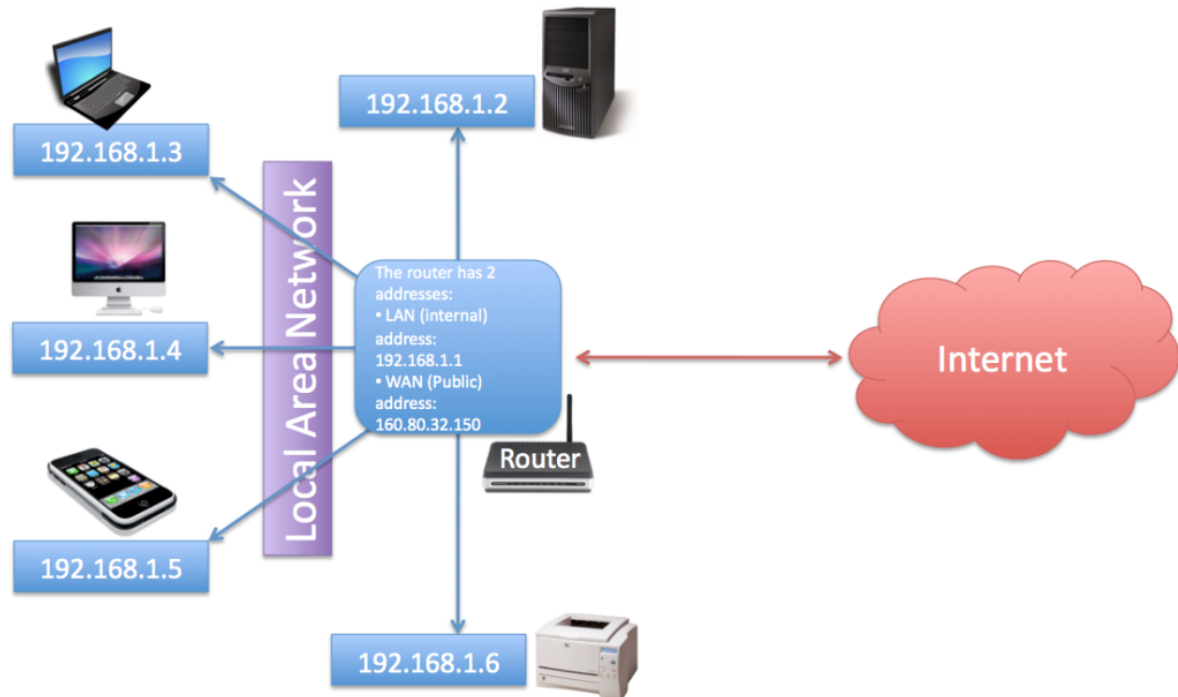
```
cd location/DevOpsJourney/app_015/  
git pull
```

replace location with where you put the DevOpsJourney

if your new go to [part 9](#) and do same steps will download old lecture files and new one.

Networks in general

Creating and connecting a local network to the Internet through a router



This image is part of the Bioinformatics Web Development tutorial at http://www.cellbiol.com/bioinformatics_web_development/

long story , short .

basically our local network work we have let's say 2 pcs they connected to same router , they shared under unique ip address and they can communicate threw this ip given by router.

that's not all for sure but it's good to start talking about docker.

docker network work in same way you can communicate over network you made or default one used by docker called bridge.

you can link 2 containers(or more) threw this network you don't need the ip of the 2 containers just names, we will see why.

create our network

```
/bin/bash
docker networks
(base) (master)in ~/Documents/DevOpsJourney
> docker network create --driver bridge network1
8c67b2f37498a5d5f61b93d95ab7200f1a578774c3d23fc1a92a21df68c42256
(base) (master)in ~/Documents/DevOpsJourney
>
```

```
docker network create --driver bridge network1
```

now inspect the network we create

```
(base) (master)in ~/Documents/DevOpsJourney
> docker inspect network1
[
  {
    "Name": "network1",
    "Id": "8c67b2f37498a5d5f61b93d95ab7200f1a578774c3d23fc1a92a21df68c42256",
    "Created": "2020-06-17T21:26:54.707665111+03:00",
    "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": {}
  }
]
(base) (master)in ~/Documents/DevOpsJourney
>
```

```
docker inspect network1
```

we can see that we get a special ip for our network 172.18.0.0

building images

first make sure you are on DevOpsJourney/app_015/ folder then run

```
docker image build -t c1-server container1/
docker image build -t c2-client container2/
```

```
(base) (master)in ~/Documents/DevOpsJourney/app_015
> docker image build -t c1-server container1/
Sending build context to Docker daemon 4.608kB
Step 1/8 : FROM python:3.9-rc-alpine
---> a206803e6cb1
Step 2/8 : RUN mkdir /app
---> Using cache
---> cccdc7a8e00
Step 3/8 : WORKDIR /app
---> Using cache
---> 1c43ee4da9d4
Step 4/8 : COPY requirements.txt requirements.txt
---> Using cache
---> 30fcc80c614c
Step 5/8 : RUN pip install -r requirements.txt
---> Using cache
---> 4cb299d53b91
Step 6/8 : COPY . .
---> 1cedc2f06c59
Step 7/8 : LABEL maintainer="Omar ElKhatib"
---> Running in ac21c201e187
Removing intermediate container ac21c201e187
---> 194401223646
Step 8/8 : CMD python app.py
---> Running in b7b4e08e007e
Removing intermediate container b7b4e08e007e
---> ca9c1819b905
Successfully built ca9c1819b905
Successfully tagged c1-server:latest
(base) (master)in ~/Documents/DevOpsJourney/app_015
>
```

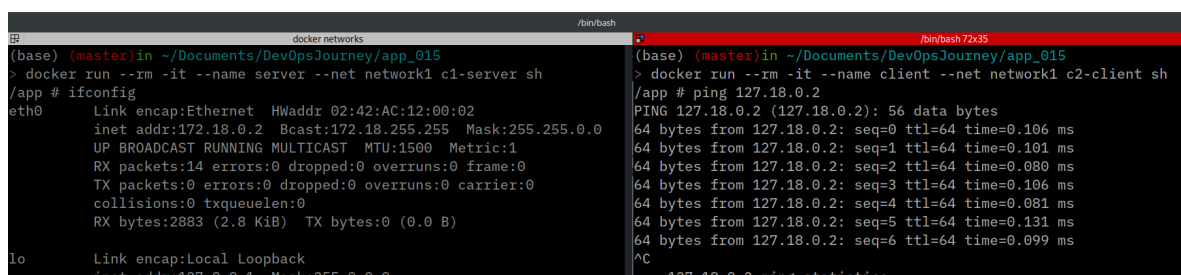
```

(base) (master)in ~/Documents/DevOpsJourney/app_015
> docker image build -t c2-client container2/
Sending build context to Docker daemon 4.096kB
Step 1/8 : FROM python:3.9-rc-alpine
---> a206803e6cb1
Step 2/8 : RUN mkdir /app
---> Using cache
---> cccdcd7a8e00
Step 3/8 : WORKDIR /app
---> Using cache
---> 1c43ee4da9d4
Step 4/8 : COPY requirements.txt requirements.txt
---> Using cache
---> 30fcc80c614c
Step 5/8 : RUN pip install -r requirements.txt
---> Using cache
---> 4cb299d53b91
Step 6/8 : COPY . .
---> 4801307b3723
Step 7/8 : LABEL maintainer="Omar ElKhatib"
---> Running in 37399d52e2eb
Removing intermediate container 37399d52e2eb
---> 0740be483676
Step 8/8 : CMD python app.py
---> Running in 7c07a54219e0
Removing intermediate container 7c07a54219e0
---> 935f5360c9dd
Successfully built 935f5360c9dd
Successfully tagged c2-client:latest
(base) (master)in ~/Documents/DevOpsJourney/app_015
>

```

connect containers

let's now run our containers under same network , again I am using terminator to split terminal you can just create a new tab and leave old one running , or just go get terminator is amazing!



```

docker networks
(base) (master)in ~/Documents/DevOpsJourney/app_015
> docker run --rm -it --name server --net network1 c1-server sh
/app # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:12:00:02
          inet addr:172.18.0.2  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2883 (2.8 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0

(base) (master)in ~/Documents/DevOpsJourney/app_015
> docker run --rm -it --name client --net network1 c2-client sh
/app # ping 127.18.0.2
PING 127.18.0.2 (127.18.0.2): 56 data bytes
64 bytes from 127.18.0.2: seq=0 ttl=64 time=0.106 ms
64 bytes from 127.18.0.2: seq=1 ttl=64 time=0.101 ms
64 bytes from 127.18.0.2: seq=2 ttl=64 time=0.080 ms
64 bytes from 127.18.0.2: seq=3 ttl=64 time=0.106 ms
64 bytes from 127.18.0.2: seq=4 ttl=64 time=0.081 ms
64 bytes from 127.18.0.2: seq=5 ttl=64 time=0.131 ms
64 bytes from 127.18.0.2: seq=6 ttl=64 time=0.099 ms
^C
--- 127.18.0.2 ping statistics ---

```

in tab 1

```
docker run --rm -it --name server --net network1 c1-server sh
```

the key in this line is `--net network1` all other stuff we talk about before so go check in previous lectures.

`--net network1` meaning i need this container to use `network1` which is the network we create in the start of this lecture

same in tab2

```
docker run --rm -it --name client --net network1 c2-client sh
```

in tab1 i am going to type inside the interactive shell

```
ifconfig
```

it's a linux tool to get information about network

we can see `addr:172.18.0.2` which is the special id for our container here.

in tab 2 , type

```
ping 172.18.0.2
```

ping is tool to make sure that we can connect to a server

and we see is sending request to server without any problems.

time to have fun

the app that i made is composed of 2 containers 1 for server(Container1 folder) and other for client (container2 folder)

first go to `container1/app.py` and change the host to the ip of the server container in our case it's `172.18.0.2`

also same to `container2/app.py` (client)

edit also to the host ip which is again `172.18.0.2`

```
1 app.py
import socket

def server_program():
    # get the hostname
    host = '172.18.0.2'
    port = 5000 # initiate port no above 1024
```

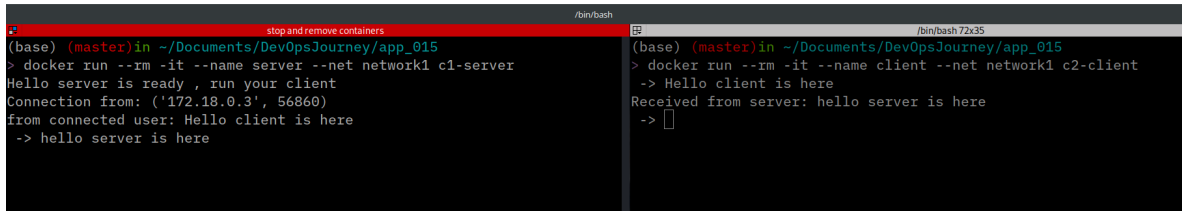
let's try to run now our server and client , I am splitting the terminal using terminator

in tab 1

```
docker run --rm -it --name server --net network1 c1-server
```

in tab 2

```
docker run --rm -it --name client --net network1 c2-client
```



The screenshot shows two terminal windows side-by-side. The left window, titled 'stop and remove containers', shows the execution of 'docker run --rm -it --name server --net network1 c1-server'. It then displays a connection from '172.18.0.3' on port 56860, with the message 'Hello client is here' received from the connected user. The right window, titled '/bin/bash 72x35', shows the execution of 'docker run --rm -it --name client --net network1 c2-client'. It displays 'Hello client is here' being sent to the server, and 'Received from server: hello server is here' being received from the server. Both windows show a prompt '->' at the bottom.

as we see , it's an chat app between server and client and it's work!!!