Nov 2017

47 / 49
Dec 2017

Dec 2017

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/29)

outside the container, no

nside the container, no… just the lo and eth0 interfaces

rom this docker run command

docker run -it --net ournet --cap-add NET_ADMIN --mac-address 02:6d:9d:0b:c0:fd --ip 192.168.2.26 ubuntu

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever

24: eth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:6d:9d:0b:c0:fd brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.2.26/24 scope global eth0
       valid_lft forever preferred_lft forever
```

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/30)

run, but changed the ip to fit my range

Nov 2017

docker run -it --net ournet --cap-add NET_ADMIN --mac-address 02:6d:9d:0b:c0:fd --ip 187.x.x.244 alpine:3.4 /bin/sh

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
   valid_lft forever preferred_lft forever
290: eth0@if287: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
link/ether 02:6d:9d:0b:c0:fd brd ff:ff:ff:ff:ff:ff
inet 187.x.x.244/22 scope global eth0
   valid_lft forever preferred_lft forever
```

same behavior - can't ping external or anything other than the other containers

47 / 49

Dec 2017

Dec 2017

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/31)

and u can ping the gateway normally, and the GW is specified for the network create right?

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/32)

nooo…sadly I cannot ping the gateway

```
PING 187.x.x.1 (187.x.x.1): 56 data bytes

--- 187.x.x.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
```

these ips, even tho they look like the external ones, are just placebo

´m still locked into some kinda of internal network, need to break free (bass riff plays)

yes, same gw as the one obtained with the ip_address_for_network script

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/33)

oh, I wanted to answer one question… if the busybox can dhcp why can't a normal container…

well, busybox executes the dhcp command and protocol, BUT, it cannot set its ip address either…

that function is not available… can only set thru the docker run or docker connect commands (which override the default mechanism)…

docker connect cannot set the mac address…

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/34)

so, let me see if I understand

> if you run ‘docker run’ and tell the container to run udhcpc, for instance, that would not work cause the container ip would have already been set
>
> that’s why we do the busybox before, so it prepares the lease and we know before hand the correct ip that needs to be assigned during ‘docker run’ - since it’s the only time when we can assign it, cause once the container is created its not possible to change its ip - is that corrrect?

unfortunately, I cannot confirm if that works or not, because the dhcp is not getting any answer
starting to suspect its got something to do with kali linux not being able to bridge properly
even without containers, if I try to create a bridge network, I cannot get ip working on it

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/35)

correct on the 1st part…

have the following

1 physical - works
1 vmware on physical (above) does not work

1 physical - windows
1 vmware ubuntu - works
1 vmware mac - does not work

on the systems that do NOT work, I can docker run -it --net ournet --mac-address mmmm -ip ppppp ubuntu
and I can ping the gateway and the naeserver at 8.8.8.8 (on the other side of my gateway), but cannot ping anything else on this local network

on the systems that work, I can ping anything without problem

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/36)

yes, I’m creating a new vm with debian on it
gonna test that
must be kalis fault
hold on and thank you

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/37)

s your actual goal here to simulate a network test environment, or just to run containers that can be seen from the network?

Reading through the thread it sounds a lot like you’re just trying to run containers and get access to them. The normal way to do this is to use `docker run -p 7777:8888` to map port 7777 on the host to port 8888 in the container, and then have external processes connect to your host’s (or in your case your VM’s) port 7777 (the two ports can be and frequently are the same). Docker provides a NAT-based environment where this just works; containers don’t need to run DHCP clients or ever really worry about what their own IP addresses are. A container’s IP address will be on a private network, not the network the host is plugged into, and that’s okay.

s there a reason this standard setup won’t work for you?

()

Nov 2017

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/38)

n my case, I have an application I want to run multiple instances of, and it doesn't play nice with ports, doesn't allow NAT, has a private protocol that embeds the source IP address in the DATA packet, and the receiver opens a NEW connection back to the source on a hard coded port…

what a pain in the butt…

but this app does api simuluation/virtualization, and as part of a test run, i would spin up N number of these to simulate the apis being used… and other teams could be doing the same… I don't want anyone to have to coordinate with each other on fixed target systems…(we have that mess now) I was able to build this on my local hardware & linux in a matter of a couple weeks (learn docker, learn app comandline, design docker container, build startup scripts, add pipework, … add to jenkins test jobs,

47 / 49

need a class A network, dhcp (so short lease time), a farm of docker hosts and some army of docker containers. (both elastic) spinning up a VM takes too long…and is very heavy weight for this 5 minute test cycle. this is not swarm, as each container, while running the same software, will be executing a different api simulation.

Dec 2017

Dec 2017

but when it came time to run this on work systems (all virtual) or AWS (moving to testing in the cloud). .it all fell apart…
SO close… we need a network driver that can listen for multiple mac addresses, which are set dynamically.

another fun project to add to the list!..

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/39)

no, I want this: have unique external ip on containers. just like host has one, which can access and be accessed from external network.

no port mapping, pls.

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/40)

quick update. tried on debian VM, same results, no game.

gonna install ubuntu via dual boot, direclty on the machine to see what's the outcome will be.

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/41)

made sure to turn off promiscuous mode on my on hardware linux… and the script above still works for me…

modified my old app scripts to use this instead of pipework, and it works fine on some hosts and not others…(no different than the old pipework approach)

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/42)

installed ubuntu directly on pc - still does not work
same problem, step 3 keeps sending discover and no response

've been researching for how to create a virtual eth interface on linux that would get is ip via dhcp to then connect the docker to it, maybe hat would be easier but no luck with that either.

Nov 2017

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/43)

on your physical install on the physical network interface ??? do

```
ip link set ???? promisc on
```

47 / 49
Dec 2017

hen try again, and

Dec 2017

```
ip link show
```

should show promiscuous mode on

```
2: eth0: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 1c:c1:de:50:8e:cb brd ff:ff:ff:ff:ff:coffee:
```

()

Nov '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/44)

ried again with promisc mode on, unfortunately, same issue on step 3.

()

Dec '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/45)

OMG! I did it! thank the Lord!

nly tested in ubuntu directly installed on pc
gotta test more

ut I found out a way to get the lease
hank you so much for all your help and attention
best regards

()

Dec '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/46)

So, tell us what u did!

()

Dec '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/47)

Nov 2017

ah ok, I thought no one would ask haha!

so, grab a drink, some snacks and enjoy the reading

01 I First, I create a macvlan interface on the host, call it “eth1”, and link it to “eth0”. The second line is optional, you can specity the mac for this new interface, if you don’t, a random one will be used. Bring eth1 up (3rd line)

```
ip link add dev eth1 link eth0 type macvlan
ip link set eth1 addr <mac address>
ip link set eth1 up
dhclient eth1 -v
```

and now the thing that changed the game - I run dhclient on the new created interface and boom, it got a lease! same range as public ip from host, external, looking good. no docker so far, but hold on, we’re getting there. so, now if I run “ifconfig” I can see both eth0 and eth1 have ip addresses assigned to them, tho, I cannot use both simultaneously from the host, as a matter of fact, I could not use the new received ip on eth1 from the host - but I was cool, that’s was not my goal anyway. the missing part was now found, I had a “LEASE!” {angelical choir plays} let’s move on.

47 / 49

Dec 2017

Dec 2017

02 I So, I took note of eth1 info (ip route, ifconfig):

```
-ip
-mac
-subnet
-gateway
```

03 I good, now, before continuing we gotta take eth1 down and change it’s mac, because our soon to be launched “Container of the Victory” will have these specs and we don’t neeeeed no more trouble, right?

```
ifconfig eth1 0.0.0.0
ip link set eth1 down
ip link set eth1 addr <change to a different mac>
```

04 I Time to create the ‘Docker Network’ using the gathered info:

```
docker network create -d macvlan \
   --subnet=<subnet from above> \
   --gateway=<gateway from above>  \
   --ip-range=<compatible with eth1 ip> \
   -o parent=eth0 <docker network name>
```

05 I Finally, launch the container using:

```
  docker run  --net --mac-address --ip
   --net = <the docker network we just created>
   --mac-address = <mac from above>
   --ip = <ip from above>

docker  run --rm  -it --name <container name> --net=<docker network name> --mac-address <mac from
above> --ip=<ip from above> alpine:3.4 /bin/sh
```

Voilá! you gotta a container in true bridge mode! 😎

the process was tested and worked on:

```
- ubuntu 16.04 directly installed on machine
- windows 10 > virtual box > kali linux
- windows 10 > vmware > debian
- windows 10 > vmware > kali
```

Nov 2017

'all vms in bridge mode, did not test without promisc mode

:hen I did a bash script to automate the process and added the option to create multiple containers, each with it's own ip, and another one to :lean up the mess and remove everything once you done. good fun!

f you wish to create more than one container, repeat steps 1, 2, 3 and 5, but use eth2, eth3 etc
iote that you may need to set the subnet mask to acommodate a larger range than the default one when creating mutiple containers. Also, /our isp needs to allow mutiples ips, if you can run a vm in bridge mode ,you're probably good to go, but test to see what your limit is.

()

47 / 49

Dec '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/48)

Dec 2017

nteresting but it sounds like you need a network / container right? Also you could use the little routine that I posted for creating the MAC address so you don't have to change it

Dec 2017

()

Dec '17 (/t/how-to-set-up-a-container-just-like-a-virtual-machine-in-bridge-mode-meaning-the-container-gets-its-own-external-ip/41831/49)

> Interesting but it sounds like you need a network / container right?

what do you mean?

The changing mac thing can be avoided if you choose to delete eth1 instead of just taking it down. But, once I´m finished, I release the ip, so need eth1 to be up again. So, it does not matter which mac I use, I need to change it before bringing the container up, got it?

Container with dedicated internal IP accessible f… (https://forums.docker.com/t/container-with-dedicated-internal-ip-accessible-from-entire-network/42720/2)