DNS not working within docker containers when host uses dnsmasq and Google's DNS server are firewalled?

Asked 7 years, 5 months ago Modified 3 years, 2 months ago Viewed 146k times

49

The symptom is: the host machine has proper network access, but programs running within containers can't resolve DNS names (which may appear to be "can't access the network" before investigating more).

```
$ sudo docker run -ti mmoy/ubuntu-netutils /bin/bash
root@082bd4ead733:/# ping www.example.com
... nothing happens (timeout) ... ^C
root@082bd4ead733:/# host www.example.com
... nothing happens (timeout) ... ^C
```

(The docker image mmoy/ubuntu-netutils is a simple image based on Ubuntu with `ping` and `host` included, convenient here since the network is broken and we can't `apt install` these tools)

The issue comes from the fact that docker automatically configured Google's public DNS as DNS server within the container:

```
root@082bd4ead733:/# cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN

nameserver 8.8.8.8
nameserver 8.8.4.4
```

This just works in many configurations, but obviously doesn't when the host runs on a network where Google's public DNS are filtered by some firewall rules.

The reason this happened is:

- Docker first tries configuring the same DNS server(s) on the host and within the container.
- The host runs dnsmasq, a DNS caching service. dnsmasq acts as a proxy for DNS requests, hence the apparent DNS server in the host's `/etc/resolve.conf` is `nameserver 127.0.1.1`, i.e. localhost.
- The host's dnsmasq listens only for requests comming from localhost and blocks requests coming from the docker container.
- Since using `127.0.1.1` within docker doesn't work, docker falls back to Google's public DNS, which do not work either.

There may be several reasons why DNS is broken within docker containers. This question (and answers) covers the case where:

- dnsmasq is used. To check whether this is the case:
  - Run `ps -e | grep dnsmasq` on the host. If the output is empty, you're not running dnsmasq.
  - Check the host's resolv.conf, it probably contains an entry like `nameserver 127.0.1.1`. If it contains `nameserver 127.0.0.53`, you're probably running `systemd-resolved` instead of dnsmasq. If so, you won't be able to use the solution forwading DNS requests to dnsmasq (the one using `listen-address=172.17.0.1`). systemd-resolved versions earlier than 247 hardcoded the fact that it listens only on the 'lo' interface hence there's no easy way to adapt this solution with these versions. Other answers below will work with systemd-resolved.
- Google's public DNS is filtered. Run `host www.example.com 8.8.8.8`. If it fails or times out, then you are in this situation.

What are the solutions to get a proper DNS configuration in this configuration?

docker dns localhost dnsmasq

Share Improve this question Follow edited Jul 21, 2022 at 20:49

asked Apr 24, 2018 at 9:30
Matthieu Moy
16.9k ● 6 ● 44 ● 71

7 Answers

Sorted by: Highest score (default)

40

A clean solution is to configure docker+dnsmasq so than DNS requests from the docker container are forwarded to the dnsmasq daemon running on the host.

For that, you need to configure dnsmasq to listen to the network interface used by docker, by adding a file `/etc/NetworkManager/dnsmasq.d/docker-bridge.conf`:

```
$ cat /etc/NetworkManager/dnsmasq.d/docker-bridge.conf
listen-address=172.17.0.1
```

Then restart network manager to have the configuration file taken into account:

```
sudo service network-manager restart
```

Once this is done, you can add `172.17.0.1`, i.e. the host's IP address from within docker, to the list of DNS servers. This can be done either using the command-line:

```
$ sudo docker run -ti --dns 172.17.0.1 mmoy/ubuntu-netutils bash
root@7805c7d153cc:/# ping www.example.com
PING www.example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=54 time=86.6 ms
```

... or through docker's configuration file `/etc/docker/daemon.json` (create it if it doesn't exist):

```
$ cat /etc/docker/daemon.json
{
  "dns": [
    "172.17.0.1",
        "8.8.8.8",
        "8.8.4.4"
  ]
}
```

(this will fall back to Google's public DNS if dnsmasq fails)

You need to restart docker to have the configuration file taken into account:

```
sudo service docker restart
```

Then you can use docker as usual:

```
$ sudo docker run -ti mmoy/ubuntu-netutils bash
root@344a983908cb:/# ping www.example.com
PING www.example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=54 time=86.3 ms
```

Share Improve this answer Follow

answered Apr 24, 2018 at 12:39

Matthieu Moy
16.9k ● 6 ● 44 ● 71

Sign up to request clarification or add additional context in comments. ✕

7 Comments

Add a comment

dashesy Over a year ago

what is "172.17.0.1"? default docker bridge gateway? docker install should automatically take care of this

3 Reply ...

Matthieu Moy Over a year ago

That's written in the answer: " `172.17.0.1` , i.e. the host's IP address from within docker". The problem is that by default, `172.17.0.1` does not reply to DNS requests (since dnsmasq only listens to the local interface), so just configuring `172.17.0.1` would not be sufficient, one needs some config on the dnsmasq side. I guess docker's developers and packagers decided that installing docker should not modify dnsmasq's config.

4 Reply

Szczepan Hołyszewski Over a year ago

`$ cat /etc/NetworkManager/dnsmasq.d/docker-bridge.conf` - you lost me right there. A "clean solution" is one where a service (dnsmasq) is completely agnostic as to what other services (docker) will be using it. If a solution requires dnsmasq to be specifically "aware" of docker (as evidenced by a file with "docker" in its name going under `dnsmasq.d/`), then it is not a clean solution.

1 Reply

Matthieu Moy Over a year ago

If you don't like the fact that the name includes "docker", you can give the file any other name ending with `.conf` . But in any case, you have to tell dnsmasq to listen properly on 172.17.0.1, because this is where requests coming from docker will be directed. Docker here is a bit more than just a daemon running on the machine, it appears like a separate machine with a separate IP address, and by default dnsmasq will reject its requests.

0 Reply

Szczepan Hołyszewski Over a year ago

What distribution does this solution pertain to? Here on Ubuntu, the directory /etc/NetworkManager does not exist.

1 Reply

5

A brutal and unsafe solution is to avoid containerization of the network, and use the same network on the host and on the container. This is unsafe because this gives access to all the network resources of the host to the container, but if you do not need this isolation this may be acceptable.

To do so, just add `--network host` to the command-line, e.g.

```
$ sudo docker run -ti --network host mmoy/ubuntu-netutils /bin/bash
root@ubuntu1604:/# ping www.example.com
PING www.example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=55 time=86.5 ms
64 bytes from 93.184.216.34: icmp_seq=2 ttl=55 time=86.5 ms
```

Share Improve this answer Follow

answered Apr 24, 2018 at 9:30

Matthieu Moy
16.9k ● 6 ● 44 ● 71

4 Comments

Add a comment

David R. Over a year ago

Is there a better way than this to fix the issue on systems using systemd-resolved? I'm on Ubuntu 18.04. I did try a custom bridge network, it didn't work. `docker network create -d bridge testnet` then `docker run -it --network=testnet debian:stretch bash` and then `apt update` fails. Oh, and iptables is flushed, and my local machine can `nslookup 8.8.8.8 google.com` just fine.

0 Reply

Matthieu Moy Over a year ago

Did you try this option: stackoverflow.com/questions/49998099/… ? You won't get the DNS caching within docker, but that should work with systemd-resolved (only the commands to find the real DNS server should change).

0 Reply

Matthieu Moy Over a year ago

`nslookup 8.8.8.8 google.com` => I guess you meant `nslookup google.com 8.8.8.8` ? That's the weird part: if this works, you're probably not in the situation described by this question. Try the exact same `nslookup` or `host` queries within and outside a container and see what the differences are.

0 Reply

X99 Over a year ago

I had this issue and no DNS config solved it at all. But your solution worked fine, thanks a lot!

1 Reply

5

One way is to use a user defined network for your container. In that case the container's `/etc/resolv.conf` will have the nameserver `127.0.0.11` (a.k.a. the Docker's embedded DNS server), which can forward DNS requests to the host's loopback address properly.

```
$ cat /etc/resolv.conf
nameserver 127.0.0.1
$ docker run --rm alpine cat /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
$ docker network create demo
557079c79ddf6be7d6def935fa0c1c3c8290a0db4649c4679b84f6363e3dd9a0
$ docker run --rm --net demo alpine cat /etc/resolv.conf
nameserver 127.0.0.11
options ndots:0
```

If you use `docker-compose`, it will set up a custom network for your services automatically (with a file format v2+). Note, however, that while `docker-compose` runs containers in a user-defined network, it still builds them in the **default network**. To use a custom network for builds you can specify the `network` parameter in the build configuration (requires file format v3.4+).

Share Improve this answer Follow answered Sep 9, 2018 at 7:42

Eugene Yarmash
152k ●44 ●344 ●391

Comments

Add a comment

2

I just had to deal with this last night and eventually remembered that docker run has a set of options for handling it. I used --dns to specify the DNS server I want the container to use. Works like a champ and no need to hack my docker host. There are other options for the domain name and search suffixes.

Share Improve this answer Follow

answered Feb 7, 2020 at 14:56

Mike Diehn
192 ●7

Comments

Add a comment

1

Since the automatic DNS discovery is guilty here, you may override the default setting in docker's configuration.

First, get the IP of the DNS server dnsmasq is using with e.g.:

```
$ sudo kill -USR1 `pidof dnsmasq`
$ sudo tail /var/log/syslog
[...]
Apr 24 13:20:19 host dnsmasq[2537]: server xx.yy.zz.tt1#53: queries sent 0, retried or
failed 0
Apr 24 13:20:19 host dnsmasq[2537]: server xx.yy.zz.tt2#53: queries sent 0, retried or
failed 0
```

The IP addresses correspond to the `xx.yy.zz.tt` placeholders above.

Alternatively, if your system is using systemd-resolve instead of dnsmasq, run:

```
$ resolvectl status | grep 'Current DNS'
Current DNS Server: xx.yy.zz.tt
```

You can set the DNS at `docker run` time with the `--dns` option:

```
$ sudo docker run --dns xx.yy.zz.tt1 --dns xx.yy.zz.tt2 -ti mmoy/ubuntu-netutils bash
root@6c5d08df5dfd:/# ping www.example.com
```

```
PING www.example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=54 time=86.6 ms
64 bytes from 93.184.216.34: icmp_seq=2 ttl=54 time=86.6 ms
```

One advantage of this solution is that there is no configuration file involved, hence no risk of forgetting about the configuration and running into troubles later because of a specific config: you're getting this DNS configuration if and only if you type the `--dns` option.

A drawback is that you won't get any DNS caching in the containers, hence DNS resolution will be slower.

Alternatively you may set it permanently in Docker's configuration file, `/etc/docker/daemon.json` (create it, on the host, if it doesn't exist):

```
$ cat /etc/docker/daemon.json
{
    "dns": ["xx.yy.zz.tt1", "xx.yy.zz.tt2"]
}
```

You need to restart the docker daemon to take the `daemon.json` file into account:

```
sudo service docker restart
```

Then you can check the configuration:

```
$ sudo docker run -ti mmoy/ubuntu-netutils bash
root@56c74d3bd94b:/# cat /etc/resolv.conf
nameserver xx.yy.zz.tt1
nameserver xx.yy.zz.tt2
root@56c74d3bd94b:/# ping www.example.com
PING www.example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=54 time=86.5 ms
```

Note that this hardcodes the DNS IP in your configuration files. This is strongly discouraged if your machine is a laptop that connects to different networks, and may be problematic if your internet service provider changes the IP of the DNS servers.

Share Improve this answer Follow

edited Aug 26, 2021 at 12:03

answered Apr 24, 2018 at 11:33

Matthieu Moy
16.9k ● 6 ● 44 ● 71

Comments

Add a comment

-1

Since `dnsmasq` is the issue, one option is to disable it on the host. This works, but will disable DNS caching for all applications running on the host, hence is a really bad idea if the host is used for applications other than docker.

If you're sure you want to go this way, uninstall `dnsmasq`, e.g. on Debian-based systems like Ubuntu, run `apt remove dnsmasq`.

You may then check that `/etc/resolv.conf` within the container points to the DNS server used by the host.

Share Improve this answer Follow

answered Apr 24, 2018 at 11:16
Matthieu Moy
16.9k ● 6 ● 44 ● 71

Comments

Add a comment

-1

I had problems with the DNS resolver in our docker containers. I tried a lot of different things, and in the end, I just figured that my VPS in **Hostgator didn't have installed** by default NetworkManager-tui (nmtui), **I just installed and reboot it**.

```
sudo yum install NetworkManager-tui
```

And reconfigured my `resolv.conf` with default DNS as `8.8.8.8`.

```
nano /etc/resolv.conf
```

Share Improve this answer Follow

answered Mar 20, 2021 at 0:43
Sebastian Cardona Osorio
495 ● 8 ● 8

Comments

Add a comment

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

docker dns localhost dnsmasq

See similar questions with these tags.