~jpetazzo/How to configure Docker to start containers on a specific IP address range

Tweet

A recurring question on the Docker mailing list and on the Docker IRC channel is "how can I change the network range used by Docker?". While Docker itself doesn't have a configuration option to change this network range (yet!), it is very easy to change it, and here is how.

Docker's default behavior

When you (or your distro's init scripts) start the Docker daemon, the daemon will check if it was given a `-b` option on the command-line. This option specifies the name of the bridge interface to be used by Docker. All the containers will be bound to this bridge. If the `-b` option is not specified, Docker will use the name `docker0` instead.

Then, Docker will check if that bridge interface actually exists. If it does, it will use it – and use whatever IP address and netmask are configured on this address. For instance, if you already have a bridge `br0` setup with IP address `10.3.3.100/24`, and start the Docker daemon with `-b br0`, then containers will be started on IP addresses from `10.3.3.1` to `10.3.3.99`, then (skipping the bridge address) from `10.3.3.101` to `10.3.3.254`.

If the interface doesn't exist, Docker will create it, and assign an IP address to it. But of course, it cannot just pick a random IP address: it would *always* conflict with someone's IP addressing plan out there. So Docker *tries* to be smart. It tries a number of different ranges, until it finds one that doesn't overlap with an existing route on your system, or with your DNS server. (You can see the whole list in network.go.)

Hell is paved with good intentions

But Docker only knows about your directly connected routes (using the `ip route` command) and your DNS server (checking `/etc/resolv.conf`). The first address that Docker tries to use is `172.17.42.1/16`. Suppose that your machine's IP address is `192.168.1.2/24`, your default gateway is `192.168.1.1/24`, and you happen to have an internal server on `172.17.6.6`, reachable through your default gateway. Docker won't "see" the route to that server (it will only see the default route), and it won't be able to "know" that it shouldn't use that network.

In other words, Docker network allocation scheme is not bullet-proof. It's still useful, because instead of working 99% of the time, it probably works 99.99% (I'm completely making up those numbers); but the remaining 0.01% still need a solution.

So what should I do?

If you are in that 0.01%, the solution is very simple: just create your own bridge, configure it with a fixed address, tell Docker to use it. Done.

If you do it manually, it will look like this (on Ubuntu):

```
stop docker
ip link add br0 type bridge
ip addr add 172.30.1.1/20 dev br0
ip link set br0 up
docker -d -b br0
```

If you want to persist your changes across server reboots, you can add the bridge to `/etc/network/interfaces`. On my laptop, I have the following definition in that file:

```
auto br0
iface br0 inet static
        address 10.1.1.1
        netmask 255.255.255.0
        bridge_ports dummy0
        bridge_stp off
        bridge_fd 0
```

My version of the ifupdown scripts require that a `bridge_ports` option is present, otherwise, it doesn't recognize the interface as a bridge. Therefore, I put a dummy interface in it. Also, for bonus points, I disabled the STP protocol and reduced the forwarding delay to zero.

Then, I updated my Docker init script to add `-b br0`.

Note: I used `br0` because I also have other VMs running on this machine (using QEMU, VirtualBox, and sometimes KVM) and I configured everything to use `br0`, so my containers and my VirtualBox VMs can communicate directly. But to make things simpler, you can just use the name `docker0` in your interfaces definition file, and Docker will pick it up automatically without extra configuration.

But I don't want to edit my system files; can't Docker do this?

Not yet. But it would be reasonable to extend the `-b` option to specify the address and netmask to use; for instance `-b br0` would still use the `br0` interface "as-is", but `-b br0=192.168.1.1/24` would create the interface and assign an IP address.

Docker is an Open Source project, and contributing is really easy. If you really need that feature, it could be the perfect opportunity to learn Go :-)

Seriously, though, if you want to implement this, don't hesitate to open a GitHub issue (after having read the contributing guidelines) to indicate that you will be working on it; and we'll look forward to reviewing your pull requests!