gesellix / **docker-haproxy-network**

Showcase using the Docker 1.10 private network in combination with the HAProxy 1.6 DNS resolution

**18** commits | **1** branch | **0** releases | **1** contributor

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

**gesellix** Update README.md — Latest commit `fadd43b` on Jun 15, 2016

| | | |
|---|---|---|
| haproxy | simplify example, add link to nginx demo | 3 years ago |
| http-server | actually use the Docker layer caching | 3 years ago |
| .gitignore | add the reverse proxy and a simple demo web app | 3 years ago |
| README.md | Update README.md | 3 years ago |

README.md

# Docker networking and DNS resolution with HAProxy

This little demo shows how to:

- create a private Docker network for a reverse proxy and a backend application
- leverage the new HAProxy name resolution to reflect a changed backend ip address

***Note**: with the current HAProxy release 1.6.5 the dns resolver seems to be broken (as of 2016-06-15)*

## Docker networking

With the shiny Docker 1.9 release, we can now define dedicated networks on top of the standard `docker0` bridge. Docker already allows to choose between another bridged network on your host and a virtual/overlay network across host boundaries.

The next step came with Docker 1.10, introducing a container embedded DNS server in user-defined networks. The DNS server replaces the previous mechanism modifying the `/etc/hosts` file inside containers. For user-defined networks we can now rely on using `/etc/resolv.conf` and ask the configured DNS server to resolve container names to ip addresses.

This demo focuses on the locally bridged network, switching to an overlay network with its need for a central key/value store (e.g. Consul) would make it too complex. The concept as such wouldn't change, though.

A deep introduction has been posted in the official announcement on Docker networking.

## HAProxy DNS resolution at runtime

HAProxy 1.6 brings a nice feature especially for using it in a Docker environment. Before, HAProxy would resolve backend ip addresses only at startup, so that ip address changes wouldn't be recognized at runtime. If you were using HAProxy as reverse proxy in front of Docker containers, you had to reload the proxy configuration on every container restart: normally, you wouldn't have the same ip address after stopping and starting a container. Obviously, this applies not only to Docker containers, but to every environment where servers are dynamically changed.

The new HAProxy release allows you to configure a dns nameserver, which effectively will be asked for an ip address when an old ip address can't be connected to anymore. The HAProxy 1.6 announcement shows a nice example, which we're going to use in this demo.

**Edit**: another example for the new HAProxy 1.6 dns resolution, but using the old Docker `--link` has been described at the official HAProxy blog: HAProxy and container ip changes in docker.

## Try this at home

You'll need Docker

Ah, well, you got it already. Great!

Now clone this repo and `cd` into the project root:

```
git clone https://github.com/gesellix/docker-haproxy-network.git
cd docker-haproxy-network
```

## Preparing the environment

Now create the example application image:

```
docker build -t app-image -f http-server/Dockerfile http-server
```

... then the HAProxy image:

```
docker build -t proxy-image -f haproxy/Dockerfile haproxy
```

Then we prepare a private Docker network, so that our containers can connect to it:

```
docker network create mynetwork
```

Now we only need to run the proxy with a dns server and the application, and connect them to `mynetwork`. We might first run them and connect them afterwards, or we can already connect them along with the `docker run` command. Let's go with the second option. Please note that the dns server at `127.0.0.11` is the current default for every Docker container. Let's hope that the ip address won't change across future Docker releases. If so, we can still fallback to reading the `/etc/resolv.conf`.

```
docker run -dit --name app --net mynetwork app-image
docker run -dit --name proxy --net mynetwork -p 80:80 proxy-image
```

The proxy-image image used above is already configured to use Docker's internal dns server. If you prefer to use your own dns server, you can override the default like follows. Please note that ip addresses are resolved from inside the Docker container so that e.g. `127.0.0.1` doesn't resolve to your host's ip address.

```
DNS_IP=127.0.0.11
DNS_PORT=53
docker run -dit --name app --net mynetwork app-image
docker run -dit --name proxy --net mynetwork -p 80:80 -e DNS_TCP_ADDR=$DNS_IP -e DNS_TCP_PORT=$DNS_PORT proxy-image
```

Note that we don't expose any port on the `app` container. This is because we don't need to when we only want the containers to communicate with each other. We want the proxy HTTP port to be exposed on our public interface, though. The proxy will use Docker's embedded DNS server for container name resolution in `mynetwork`.

## Hello World

Both containers should be running in the background now and you can check if the setup is working by navigating your browser to the Docker daemon host. When using native Linux or the Docker for Mac beta this is most probably at http://localhost:80, while for Windows and Mac OS users with Docker Machine it should be http://192.168.99.100:80.

If everything works fine, you should see the classic *hello world*.

You can check the current container ip addresses with

```
docker inspect --format '{{ .NetworkSettings.Networks.mynetwork.IPAddress }}{{ .Name }}' app proxy
```

So far so good - standard stuff. Let's try to break it!

## Handling changed ip addresses

We now want the HAProxy to get into trouble by restarting our app, and making it available at a changed ip address. That's why we're going to stop it and start another container on the `mynetwork`, so that the old ip address will be taken:

```
docker kill app; docker wait app; docker rm app
docker run -dit --name placeholder-app --net mynetwork busybox:latest sh -c "ping 127.0.0.1 > /dev/null"
```

Then we're going to restart the app again:

```
docker run -dit --name app --net mynetwork app-image
```

Our app container should now have a changed ip address, we can check that with:

```
docker inspect --format '{{ .NetworkSettings.Networks.mynetwork.IPAddress }}{{ .Name }}' app proxy placeholder-app
```

A HAProxy before version 1.6 would now need to be restarted, because it wouldn't be able to connect to the old ip address anymore. Since we already configured our `proxy` to dynamically resolve changed ip addresses, everything should still work.

You can verify that by refreshing your browser. Do it. And smile 😄

## Alternatives

- Similar demo for Nginx

## Questions, Suggestions, Anything?

Does it work for you? Do you need more details or did I tell something wrong? You can contact me here at the issue tracker or via Twitter @gesellix!