

# Instalação HAProxy Docekr HAproxytec

Link: <https://hub.docker.com/r/haproxytech/haproxy-debian>

## . Quick reference

- **Where to get help:**

[HAProxy mailing list](#), [HAProxy Community Slack](#) or [#haproxy on Libera.chat](#)

- **Where to file issues:**

<https://github.com/haproxytech/haproxy-docker-debian/issues>

- **Maintained by:**

[HAProxy Technologies](#)

- **Supported architectures:** ([more info](#))

`linux/amd64`, `linux/arm64`, `linux/arm/v7`

- **Image updates:**

[commits to haproxytech/haproxy-docker-debian](#), [top level haproxytech/haproxy-docker-debian](#) [image folder](#)

- **Source of this description:**

[README.md](#)

## What is HAProxy?

HAProxy is the fastest and most widely used open-source load balancer and application delivery controller. Written in C, it has a reputation for efficient use of both processor and memory. It can proxy at either layer 4 (TCP) or layer 7 (HTTP) and has additional features for inspecting, routing and modifying HTTP messages.

It comes bundled with a web UI, called the HAProxy Stats page, that you can use to monitor error rates, the volume of traffic and latency. Features can be toggled on by updating a single configuration file, which provides a syntax for defining routing rules, rate limiting, access controls, and more.

Other features include:

- SSL/TLS termination
- Gzip compression

- Health checking
- HTTP/2
- gRPC support
- Lua scripting
- DNS service discovery
- Automatic retries of failed connections
- Verbose logging

logo

## How to use this image

This image is being shipped with a trivial sample configuration and for any real life use it should be configured according to the [extensive documentation](#) and [examples](#). We will now show how to override shipped haproxy.cfg with one of your own.

### Create a Dockerfile

```
FROM haproxytech/haproxy-debian:3.0
COPY haproxy.cfg /usr/local/etc/haproxy/haproxy.cfg
```

### Build the container

```
$ docker build -t my-haproxy .
```

### Test the configuration file

```
$ docker run -it --rm my-haproxy haproxy -c -f /usr/local/etc/haproxy/haproxy.cfg
```

### Run the container

```
$ docker run -d --name my-running-haproxy my-haproxy
```

You will also need to publish the ports your HAProxy is listening on to the host by specifying the `-p` option, for example `-p 8080:80` to publish port 8080 from the container host to port 80 in the container.

### Use volume for configuration persistency

```
$ docker run -d --name my-running-haproxy -v /path/to/etc/haproxy:/usr/local/etc/haproxy:ro
haproxytech/haproxy-debian:3.0
```

Note that your host's `/path/to/etc/haproxy` folder should be populated with a file named `haproxy.cfg` as well as any other accompanying files local to `/etc/haproxy`.

## Reloading config

To be able to reload HAProxy configuration, you can send `SIGUSR2` to the container:

```
$ docker kill -s USR2 my-running-haproxy
```

## Enable Data Plane API

[Data Plane API](#) sidecar is being distributed by default in all 2.0+ images and to enable it there are a few steps required:

1. define one or more users through `userlist`
2. enable dataplane api process through `program api`
3. enable haproxy.cfg to be read/write mounted in Docker, either by defining volume being r/w or by rebuilding image with your own haproxy.cfg
4. expose dataplane TCP port in Docker with `--expose`

Relevant part of haproxy.cfg is below:

```
userlist haproxy-dataplaneapi
    user admin insecure-password mypassword

program api
    command /usr/bin/dataplaneapi --host 0.0.0.0 --port 5555 --haproxy-bin /usr/sbin/haproxy --
config-file /usr/local/etc/haproxy/haproxy.cfg --reload-cmd "kill -SIGUSR2 1" --restart-cmd
"kill -SIGUSR2 1" --reload-delay 5 --userlist haproxy-dataplaneapi
    no option start-on-reload
```

To run such image we would use the following command (note that volume containing haproxy.cfg is mounted r/w and port tcp/5555 is being exposed):

```
$ docker run -d --name my-running-haproxy --expose 5555 -v
/path/to/etc/haproxy:/usr/local/etc/haproxy:rw haproxytech/haproxy-debian
```

## License

View [license information](#) for the software contained in this image.

As with all Docker images, these likely also contain other software which may be under other licenses (such as Bash, etc from the base distribution, along with any direct or indirect dependencies of the primary software being contained).

