haugene / **docker-transmission-openvpn**

Docker container running Transmission torrent client with WebUI while connecting to OpenVPN
https://hub.docker.com/r/haugene/tran…

433 commits | **2** branches | **22** releases | **74** contributors | GPL-3.0

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

haugene Reverting to ppa source, not running patched version anymore #418 — Latest commit `c0b959e` 4 days ago

openvpn	Updated cryptostorm openvpn files, fixes #371	10 days ago
plugins/rss	Use separate container for the RSS plugin	2 months ago
proxy	adding ARM (Raspberry Pi) support	2 years ago
transmission	Merge pull request #395 from aerickson/True_fixes	6 days ago
DockerEnv	Merge pull request #395 from aerickson/True_fixes	6 days ago
Dockerfile	Reverting to ppa source, not running patched version anymore #418	4 days ago
Dockerfile.alpine	Install UFW with dockerfile	23 days ago
Dockerfile.armhf	Merge pull request #401 from infothrill/infothrill/arm-stretch	15 days ago
LICENSE	initial commit. working, but needs more features	3 years ago
README.md	Add Mullvad default config and update README. Also remove RSS-contain…	2 months ago
docker-compose-armhf.yml	Use separate container for the RSS plugin	2 months ago
docker-compose.yml	Use separate container for the RSS plugin	2 months ago

README.md

Transmission with WebUI and OpenVPN

Docker container running Transmission torrent client with WebUI while connecting to OpenVPN. It bundles certificates and configurations for a bunch of VPN providers and if you're using PIA as provider it will update Transmission hourly with assigned open port. Please read the instructions below, and read it again before posting an issue :)

about:maintenance (aka I NEED HELP)

The classic story! I created this image for my own use and figured that sharing is caring, right? A lot has happened since then. With the help of contributors (thank you!) and feature requests from the community the number of providers, features and users have increased quite drastically.

If you're interested in joining as a collaborator: Find an issue and fix it, then mention in the pull-request that you're interested in helping out on a more permanent basis. That would make my day ;)

Run container from Docker registry

The container is available from the Docker registry and this is the simplest way to get it. To run the container use this command:

```
$ docker run --cap-add=NET_ADMIN --device=/dev/net/tun -d \
             -v /your/storage/path/:/data \
             -v /etc/localtime:/etc/localtime:ro \
```

```
-e "OPENVPN_PROVIDER=PIA" \
-e "OPENVPN_CONFIG=Netherlands" \
-e "OPENVPN_USERNAME=user" \
-e "OPENVPN_PASSWORD=pass" \
--log-driver json-file \
--log-opt max-size=10m \
-p 9091:9091 \
haugene/transmission-openvpn
```

You must set the environment variables `OPENVPN_PROVIDER`, `OPENVPN_USERNAME` and `OPENVPN_PASSWORD` to provide basic connection details.

The `OPENVPN_CONFIG` is an optional variable. If no config is given, a default config will be selected for the provider you have chosen. Find available OpenVPN configurations by looking in the openvpn folder of the GitHub repository. The value that you should use here is the filename of your chosen openvpn configuration *without* the .ovpn file extension. For example:

```
-e "OPENVPN_CONFIG=ipvanish-AT-Vienna-vie-c02"
```

As you can see, the container also expects a data volume to be mounted. This is where Transmission will store your downloads, incomplete downloads and look for a watch directory for new .torrent files. By default a folder named transmission-home will also be created under /data, this is where Transmission stores its state.

Supported providers

This is a list of providers that are bundled within the image. Feel free to create an issue if your provider is not on the list, but keep in mind that some providers generate config files per user. This means that your login credentials are part of the config an can therefore not be bundled. In this case you can use the custom provider setup described later in this readme. The custom provider setting can be used with any provider.

Provider Name	Config Value (`OPENVPN_PROVIDER`)
Anonine	`ANONINE`
AnonVPN	`ANONVPN`
BlackVPN	`BLACKVPN`
BTGuard	`BTGUARD`
Cryptostorm	`CRYPTOSTORM`
Cypherpunk	`CYPHERPUNK`
FrootVPN	`FROOT`
FrostVPN	`FROSTVPN`
Giganews	`GIGANEWS`
HideMe	`HIDEME`
HideMyAss	`HIDEMYASS`
IntegrityVPN	`INTEGRITYVPN`
IPredator	`IPREDATOR`
IPVanish	`IPVANISH`
Ivacy	`IVACY`
IVPN	`IVPN`
Mullvad	`MULLVAD`

Provider Name	Config Value (`OPENVPN_PROVIDER`)
Newshosting	`NEWSHOSTING`
NordVPN	`NORDVPN`
OVPN	`OVPN`
Perfect Privacy	`PERFECTPRIVACY`
Private Internet Access	`PIA`
PrivateVPN	`PRIVATEVPN`
proXPN	`PROXPN`
PureVPN	`PUREVPN`
RA4W VPN	`RA4W`
SaferVPN	`SAFERVPN`
SlickVPN	`SLICKVPN`
Smart DNS Proxy	`SMARTDNSPROXY`
SmartVPN	`SMARTVPN`
TigerVPN	`TIGER`
TorGuard	`TORGUARD`
UsenetServerVPN	`USENETSERVER`
Windscribe	`WINDSCRIBE`
VPNArea.com	`VPNAREA`
VPN.AC	`VPNAC`
VPN.ht	`VPNHT`
VPNBook.com	`VPNBOOK`
VPNTunnel	`VPNTUNNEL`
VyprVpn	`VYPRVPN`

Required environment options

Variable	Function	Example
`OPENVPN_PROVIDER`	Sets the OpenVPN provider to use.	`OPENVPN_PROVIDER=provider` . Supported providers and their config values are listed in the table above.
`OPENVPN_USERNAME`	Your OpenVPN username	`OPENVPN_USERNAME=asdf`
`OPENVPN_PASSWORD`	Your OpenVPN password	`OPENVPN_PASSWORD=asdf`

Network configuration options

Variable	Function	Example
`OPENVPN_CONFIG`	Sets the OpenVPN endpoint to connect to.	`OPENVPN_CONFIG=UK Southampton`

Variable	Function	Example
`OPENVPN_OPTS`	Will be passed to OpenVPN on startup	See OpenVPN doc
`LOCAL_NETWORK`	Sets the local network that should have access.	`LOCAL_NETWORK=192.168.0.0/24`

Firewall configuration options

When enabled, the firewall blocks everything except traffic to the peer port and traffic to the rpc port from the LOCAL_NETWORK and the internal docker gateway.

If TRANSMISSION_PEER_PORT_RANDOM_ON_START is enabled then it allows traffic to the range of peer ports defined by TRANSMISSION_PEER_PORT_RANDOM_HIGH and TRANSMISSION_PEER_PORT_RANDOM_LOW.

Variable	Function	Example
`ENABLE_UFW`	Enables the firewall	`ENABLE_UFW=true`

Alternative web UIs

You can override the default web UI by setting the `TRANSMISSION_WEB_HOME` environment variable. If set, Transmission will look there for the Web Interface files, such as the javascript, html, and graphics files.

Combustion UI and Kettu come bundled with the container. You can enable either of them by setting `TRANSMISSION_WEB_UI=combustion` or `TRANSMISSION_WEB_UI=kettu`, respectively. Note that this will override the `TRANSMISSION_WEB_HOME` variable if set.

Variable	Function	Example
`TRANSMISSION_WEB_HOME`	Set Transmission web home	`TRANSMISSION_WEB_HOME=/path/to/web/ui`
`TRANSMISSION_WEB_UI`	Use the specified bundled web UI	`TRANSMISSION_WEB_UI=combustion` or `TRANSMISSION_WEB_UI=kettu`

Transmission configuration options

You may override transmission options by setting the appropriate environment variable.

The environment variables are the same name as used in the transmission settings.json file and follow the format given in these examples:

Transmission variable name	Environment variable name
`speed-limit-up`	`TRANSMISSION_SPEED_LIMIT_UP`
`speed-limit-up-enabled`	`TRANSMISSION_SPEED_LIMIT_UP_ENABLED`
`ratio-limit`	`TRANSMISSION_RATIO_LIMIT`
`ratio-limit-enabled`	`TRANSMISSION_RATIO_LIMIT_ENABLED`

As you can see the variables are prefixed with `TRANSMISSION_`, the variable is capitalized, and `-` is converted to `_`.

PS: `TRANSMISSION_BIND_ADDRESS_IPV4` will be overridden to the IP assigned to your OpenVPN tunnel interface. This is to prevent leaking the host IP.

User configuration options

By default everything will run as the root user. However, it is possible to change who runs the transmission process. You may set the following parameters to customize the user id that runs transmission.

Variable	Function	Example
`PUID`	Sets the user id who will run transmission	`PUID=1003`
`PGID`	Sets the group id for the transmission user	`PGID=1003`

RSS plugin

The Transmission RSS plugin can optionally be run as a separate container. It allow to download torrents based on an RSS URL, see Plugin page.

```
$ docker run -d -e "RSS_URL=http://.../xxxxx.rss" \
      --link <transmission-container>:transmission \--link
```

Use docker env file

Another way is to use a docker env file where you can easily store all your env variables and maintain multiple configurations for different providers. In the GitHub repository there is a provided DockerEnv file with all the current transmission and openvpn environment variables. You can use this to create local configurations by filling in the details and removing the # of the ones you want to use.

Please note that if you pass in env. variables on the command line these will override the ones in the env file.

See explanation of variables above. To use this env file, use the following to run the docker image:

```
$ docker run --cap-add=NET_ADMIN --device=/dev/net/tun -d \
              -v /your/storage/path/:/data \
              -v /etc/localtime:/etc/localtime:ro \
              --env-file /your/docker/env/file \
              -p 9091:9091 \
              haugene/transmission-openvpn
```

Access the WebUI

But what's going on? My http://my-host:9091 isn't responding? This is because the VPN is active, and since docker is running in a different ip range than your client the response to your request will be treated as "non-local" traffic and therefore be routed out through the VPN interface.

How to fix this

The container supports the `LOCAL_NETWORK` environment variable. For instance if your local network uses the IP range 192.168.0.0/24 you would pass `-e LOCAL_NETWORK=192.168.0.0/24`.

Alternatively you can reverse proxy the traffic through another container, as that container would be in the docker range. There is a reverse proxy being built with the container. You can run it using the command below or have a look in the repository proxy folder for inspiration for your own custom proxy.

```
$ docker run -d \
      --link <transmission-container>:transmission \
      -p 8080:8080 \
      haugene/transmission-openvpn-proxy
```

Known issues, tips and tricks

Use Google DNS servers

Some have encountered problems with DNS resolving inside the docker container. This causes trouble because OpenVPN will not be able to resolve the host to connect to. If you have this problem use dockers --dns flag to override the resolv.conf of the container. For example use googles dns servers by adding --dns 8.8.8.8 --dns 8.8.4.4 as parameters to the usual run command.

Restart container if connection is lost

If the VPN connection fails or the container for any other reason loses connectivity, you want it to recover from it. One way of doing this is to set environment variable `OPENVPN_OPTS=--inactive 3600 --ping 10 --ping-exit 60` and use the `--restart=always` flag when starting the container. This way OpenVPN will exit if ping fails over a period of time which will stop the container and then the Docker deamon will restart it.

Running it on a NAS

Several popular NAS platforms supports Docker containers. You should be able to set up and configure this container using their web interfaces. Remember that you need a TUN/TAP device to run the container. To set up the device it's probably simplest to install a OpenVPN package for the NAS. This should set up the device. If not, there are some more detailed instructions below.

Questions?

If you are having issues with this container please submit an issue on GitHub. Please provide logs, docker version and other information that can simplify reproducing the issue. Using the latest stable verison of Docker is always recommended. Support for older version is on a best-effort basis.

Adding new providers

If your VPN provider is not in the list of supported providers you could always create an issue on GitHub and see if someone could add it for you. But if you're feeling up for doing it yourself, here's a couple of pointers.

You clone this repository and create a new folder under "openvpn" where you put the .ovpn files your provider gives you. Depending on the structure of these files you need to make some adjustments. For example if they come with a ca.crt file that is referenced in the config you need to update this reference to the path it will have inside the container (which is /etc/openvpn/...). You also have to set where to look for your username/password.

There is a script called adjustConfigs.sh that could help you. After putting your .ovpn files in a folder, run that script with your folder name as parameter and it will try to do the changes descibed above. If you use it or not, reading it might give you some help in what you're looking to change in the .ovpn files.

Once you've finished modifying configs, you build the container and run it with OPENVPN_PROVIDER set to the name of the folder of configs you just created (it will be lowercased to match the folder names). And that should be it!

So, you've just added your own provider and you're feeling pretty good about it! Why don't you fork this repository, commit and push your changes and submit a pull request? Share your provider with the rest of us! :) Please submit your PR to the dev branch in that case.

Using a custom provider

If you want to run the image with your own provider without building a new image, that is also possible. For some providers, like AirVPN, the .ovpn files are generated per user and contains credentials. They should not be added to a public image. This is what you do:

Add a new volume mount to your `docker run` command that mounts your config file: `-v /path/to/your/config.ovpn:/etc/openvpn/custom/default.ovpn`

Then you can set `OPENVPN_PROVIDER=CUSTOM` and the container will use the config you provided. If you are using AirVPN or other provider with credentials in the config file, you still need to set `OPENVPN_USERNAME` and `OPENVPN_PASSWORD` as this is required by the startup script. They will not be read by the .ovpn file, so you can set them to whatever.

Note that you still need to modify your .ovpn file as described in the previous section. If you have an separate ca.crt file your volume mount should be a folder containing both the ca.crt and the .ovpn config.

Controlling Transmission remotely

The container exposes /config as a volume. This is the directory where the supplied transmission and OpenVPN credentials will be stored. If you have transmission authentication enabled and want scripts in another container to access and control the transmission-daemon, this can be a handy way to access the credentials. For example, another container may pause or restrict transmission speeds while the server is streaming video.

Running on ARM (Raspberry PI)

Since the Raspberry PI runs on an ARM architecture instead of x64, the existing x64 images will not work properly. To support users that wish to run this container on a Raspberry Pi, there are 2 additional Dockerfiles created. The Dockerfiles supported by the Raspberry PI are Dockerfile.armhf -- there is also an example docker-compose-armhf file that shows how you might use Transmission/OpenVPN and the corresponding nginx reverse proxy on an RPI machine.

Make it work on Synology NAS

Here are the steps to run it on a Synology NAS (Tested on DSM 6) :

- Connect as *admin* to your Synology SSH
- Switch to root with command `sudo su -`
- Enter your *admin* password when prompted
- Create a TUN.sh file anywhere in your synology file system by typing `vim /volume1/foldername/TUN.sh` replacing *foldername* with any folder you created on your Synology
- Paste @timkelty 's script :

```
#!/bin/sh

# Create the necessary file structure for /dev/net/tun
if ( [ ! -c /dev/net/tun ] ); then
        if ( [ ! -d /dev/net ] ); then
                mkdir -m 755 /dev/net
        fi
        mknod /dev/net/tun c 10 200
fi

# Load the tun module if not already loaded
if ( !(lsmod | grep -q "^tun\s") ); then
        insmod /lib/modules/tun.ko
fi
```

- Save the file with [escape] + `:wq!`
- Go in the folder containing your script : `cd /volume1/foldername/`
- Check permission with `chmod 0755 TUN.sh`
- Run it with `./TUN.sh`
- Return to initial directory typing `cd`
- Create the DNS config file by typing `vim /volume1/foldername/resolv.conf`
- Paste the following lines :

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

- Save the file with [escape] + `:wq!`

- Create your docker container with a the following command line:

```
# Tested on DSM 6.1.4-15217 Update 1, Docker Package 17.05.0-0349
docker run \
    --cap-add=NET_ADMIN \
    --device=/dev/net/tun \
    -d \
    -v /volume1/foldername/resolv.conf:/etc/resolv.conf \
    -v /volume1/yourpath/:/data \
    -e "OPENVPN_PROVIDER=PIA" \
    -e "OPENVPN_CONFIG=Netherlands" \
    -e "OPENVPN_USERNAME=XXXXX" \
    -e "OPENVPN_PASSWORD=XXXXX" \
    -e "LOCAL_NETWORK=192.168.0.0/24" \
    -e "OPENVPN_OPTS=--inactive 3600 --ping 10 --ping-exit 60" \
    -e "PGID=100" \
    -e "PUID=1234" \
    -p 9091:9091 \
    --sysctl net.ipv6.conf.all.disable_ipv6=0 \
    --name "transmission-openvpn-syno" \
    haugene/transmission-openvpn:latest
```

- To make it work after a nas restart, create an automated task in your synology web interface : go to **Settings Panel > Task Scheduler ** create a new task that run `/volume1/foldername/TUN.sh` as root (select '*root*' in 'user' selectbox). This task will start module that permit the container to run, you can make a task that run on startup. These kind of task doesn't work on my nas so I just made a task that run every minute.

- Enjoy

systemd Integration

On many modern linux systems, including Ubuntu, systemd can be used to start the transmission-openvpn at boot time, and restart it after any failure.

Save the following as `/etc/systemd/system/transmission-openvpn.service` , and replace the OpenVPN PROVIDER/USERNAME/PASSWORD directives with your settings, and add any other directives that you're using.

This service is assuming that there is a `bittorrent` user set up with a home directory at `/home/bittorrent/` . The data directory will be mounted at `/home/bittorrent/data/` . This can be changed to whichever user and location you're using.

OpenVPN is set to exit if there is a connection failure. OpenVPN exiting triggers the container to also exit, then the `Restart=always` definition in the `transmission-openvpn.service` file tells systems to restart things again.

```
[Unit]
Description=haugene/transmission-openvpn docker container
After=docker.service
Requires=docker.service

[Service]
User=bittorrent
TimeoutStartSec=0
ExecStartPre=-/usr/bin/docker kill transmission-openvpn
ExecStartPre=-/usr/bin/docker rm transmission-openvpn
ExecStartPre=/usr/bin/docker pull haugene/transmission-openvpn
ExecStart=/usr/bin/docker run \
        --name transmission-openvpn \
        --cap-add=NET_ADMIN \
        --device=/dev/net/tun \
        -v /home/bittorrent/data/:/data \
        -e "OPENVPN_PROVIDER=TORGUARD" \
        -e "OPENVPN_USERNAME=bittorrent@example.com" \
        -e "OPENVPN_PASSWORD=hunter2" \
        -e "OPENVPN_CONFIG=Netherlands" \
        -e "OPENVPN_OPTS=--inactive 3600 --ping 10 --ping-exit 60" \
```

```
        -e "TRANSMISSION_UMASK=0" \
        -p 9091:9091 \
        --dns 8.8.8.8 \
        --dns 8.8.4.4 \
        haugene/transmission-openvpn
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Then enable and start the new service with:

```
$ sudo systemctl enable /etc/systemd/system/transmission-openvpn.service
$ sudo systemctl restart transmission-openvpn.service
```

If it is stopped or killed in any fashion, systemd will restart the container. If you do want to shut it down, then run the following command and it will stay down until you restart it.

```
$ sudo systemctl stop transmission-openvpn.service
# Later ...
$ sudo systemctl start transmission-openvpn.service
```