Docker: VPN with Port Forwarding (WireGuard) & qBitTorrent

Sample Docker Compose configuration for running qBitTorrent as a container routed through another Mullvad container.

Note: In theory this *should* work with other VPN providers, but I have only tested it with Mullvad specifically. Port forwarding will only work if the VPN provider supports port forwarding, but torrenting might still work regardless. Keep in mind that without port forwarding, you'll usually end up with less peers and thus downloads make take longer. You will also restrict the amount of peers you can upload (seed) to.

Update - June 2023

Mullvad is removing support for port forwarding. I'm keeping this page up for historical reasons (and I'll try to generalize places where I've named Mullvad directly), but keep in mind that you'll *have to* rely on other VPN providers that support port forwarding.

The /r/VPNTorrents subreddit has a thread for VPN recommendations that might be useful to those that are looking into alternatives.

I am personally using AirVPN. Feel free to use my referral link if you plan on using it as well :)

Port forwarding

For port forwarding to work, you have to set the "Listening port" (Settings -> Connection -> Listening port) in qBitTorrent to the correct forwarded port from your VPN provider.

I also recommend forcing qBitTorrent to use the VPN network interface (Advanced -> Network interface), though I'm not sure if it matters much

`docker-compose.yml` file

```
version: '3.5'
services:
  vpn:
    container_name: wireguard
    image: jordanpotter/wireguard
    cap_add:
      - NET_ADMIN
      - SYS_MODULE
    sysctls:
      net.ipv4.conf.all.src_valid_mark: 1
      net.ipv6.conf.all.disable_ipv6: 0
    volumes:
      # Generate a file here: https://mullvad.net/en/account/#/wireguard-config/
      # and put it in the same folder as `docker-compose.yml`
      # Make sure to name it `mullvad.conf` exactly.
      #
      # Alternatively: change the line below to match your local filename. If you use `my-vpn-provider.conf` instead, the line should look similar to:
      # ./my-vpn-provider.conf:/etc/wireguard/mullvad.conf
      #
      # The last part after `:` isn't important, besides the fact it needs to be within `/etc/wireguard`.
      #
      # Keep in mind that the "Network interface" within qBitTorrent (second screenshot on wiki) is named
      # based on the filename of `mullvad.conf` in `/etc/wireguard/mullvad.conf`
      - ./mullvad.conf:/etc/wireguard/mullvad.conf
    ports:
      # I run NGINX on the host server for a reverse proxy.
      # The secondary `8888` matches the `WEBUI_PORT` on the qBitTorrent container.
      #
      # This means that the host's port "8080" will now forward to port "8888" on the qBitTorrent container.
      - "127.0.0.1:8080:8888"
    restart: unless-stopped

  qbittorrent:
    container_name: qbit-wg
    # `:14.3.9` locks the qBitTorrent version to v4.3.9.
    # I recommend using this for the time being, because 4.4.x seems to be having various issues, including performance and memory leak issues.
    image: ghcr.io/linuxserver/qbittorrent:14.3.9
```

```
    depends_on:
      - vpn
    network_mode: "service:vpn"
    restart: unless-stopped
    volumes:
      # For persisting qBitTorrent configuration files
      - ./config:/config
      # Hard drives with more storage space, for download directories.
      # - /data/media:/data/media
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Oslo
      - WEBUI_PORT=8888
```

NGINX configuration file

```
# This is loosely based on the following:
# qBitTorrent GitHub wiki: https://github.com/qbittorrent/qBittorrent/wiki/NGINX-Reverse-Proxy-for-Web-UI
# My personal NGINX bootstrapping script:
https://gist.github.com/Decicus/2f09db5d30f4f24e39de3792bba75b72
# Someone's reddit comment while I was researching, I wish I could find it again...

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name qbittorrent.example.com;
    root /var/www/html;

    ssl_certificate /srv/ssl/qbittorrent/fullchain.pem;
    ssl_certificate_key /srv/ssl/qbittorrent/key.pem;

    server_tokens off;

    # These files are included as part of my NGINX bootstrapping script:
https://gist.github.com/Decicus/2f09db5d30f4f24e39de3792bba75b72
    # So normally I'd just put `include ssl_params.conf` and call it a day.
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
```

```
GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;

    # dhparams will have to be created here
    # See this script file: https://gist.github.com/Decicus/2f09db5d30f4f24e39de3792bba75b72#file-generate-dhparams-sh
    ssl_dhparam /etc/nginx/dhparams.pem;
    ssl_session_cache builtin:1000 shared:SSL:10m;
    ssl_session_timeout 1d;
    ssl_session_tickets off;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";
    add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload";

    # CSP that whitelists a userstyle I use in my browser, for the qBitTorrent web UI: https://docs.theme-park.dev/themes/qbittorrent/
    # Might wanna tweak it to your own liking if you don't use it.
    # Commented by default, because it generally causes more headaches than it helps...
    # add_header Content-Security-Policy "default-src https://use.fontawesome.com https: moz-extension: 'self' 'unsafe-inline'; style-src https://gilbn.github.io https://theme-park.dev https://qbittorrent.example.com https://raw.githubusercontent.com https://use.fontawesome.com 'unsafe-inline'; img-src https:";

    index index.nginx-debian.html index.html index.htm;

    charset utf-8;

    location / {
        # IP-based whitelisting. I whitelist my home IP, but you can choose to use something like HTTP auth if you want.
        # Some might also be fine with exposing qBitTorrent's login page directly.
        # allow 127.0.0.1;
        # allow 192.168.0.0/24;
        # deny all;

        proxy_pass http://127.0.0.1:8080;
        proxy_http_version 1.1;
        http2_push_preload on; # Enable http2 push
```

```
        # The `8888` port here has to match the one in the `WEBUI_PORT` in docker-compose.yml
        # Or else you will have issues...
        proxy_set_header   Host              127.0.0.1:8888;
        proxy_set_header   X-Forwarded-Proto  $scheme;
        proxy_set_header   X-Forwarded-Host   $http_host;
        proxy_set_header   X-Forwarded-For    $remote_addr;
        proxy_set_header   X-Real-IP          $remote_addr;
        proxy_hide_header   Content-Security-Policy;

        # optionally, you can adjust the POST request size limit, to allow adding a lot of torrents at once
    #
    # 100M might be a bit overkill, but for large torrents with tons of files,
    # you'll likely run into problems if you choose not to adjust this at all.
        client_max_body_size 100M;

        # since v4.2.2, is possible to configure qBittorrent
        # to set the "Secure" flag for the session cookie automatically.
        # However, that option does nothing unless using qBittorrent's built-in HTTPS functionality.
        # For this use case, where qBittorrent itself is using plain HTTP
        # (and regardless of whether or not the external website uses HTTPS),
        # the flag must be set here, in the proxy configuration itself:
        proxy_cookie_path / "/; Secure";
    }

    location /.well-known {
        auth_basic "off";
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    access_log /var/log/nginx/qbittorrent-access.log combined;
    error_log  /var/log/nginx/qbittorrent-error.log error;

    location ~ /\.ht {
        deny all;
    }
}
```

Revision #12
Created 6 July 2021 12:48:36 by Alex Thomassen
Updated 8 August 2023 08:01:56 by Alex Thomassen