# 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

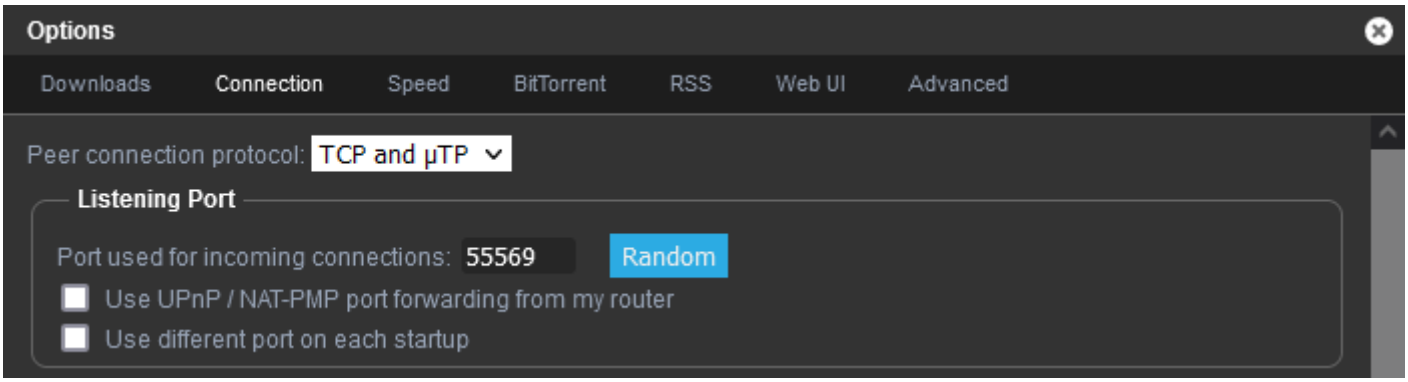
Mullvad is removing support for port forwarding. I'm keeping this page up for historical reasons (and I'll try to generalize places where I've named Mullvad directly), but keep in mind that you'll *have to* rely on other VPN providers that support port forwarding.

The /r/VPNTorrents subreddit had a thread for VPN recommendations that might be useful to those that are looking into alternatives. Subreddit has been banned, so an archive of the thread from March 2025 is linked.

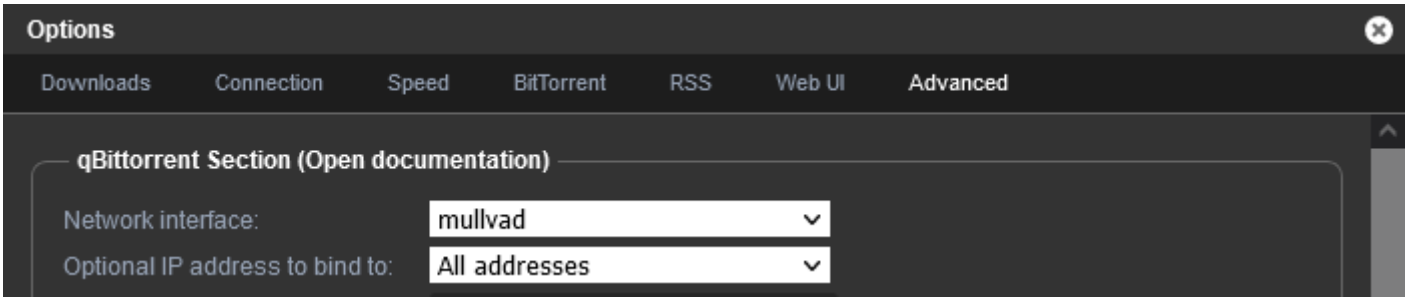
I am personally using AirVPN. Feel free to use my referral link if you plan on using it as well :)

# 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

```
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:
            ### I use AirVPN, so first I have a "device": https://airvpn.org/devices/
            ### Then I make sure to open a port (with P2P support) via:
https://airvpn.org/ports/
            ### For that generated port, I choose the device I created earlier.

            ### The WireGuard config is then generated via https://airvpn.org/generator/
            ### Select one of the servers closest to where you're hosting your qBitTorrent
setup
            ### Download the .conf file, rename it to vpn.conf and put it in the same folder
as the docker-compose.yml file.
            - ./vpn.conf:/etc/wireguard/vpn.conf
        ports:
            ### I run NGINX on the host server for a reverse proxy.
            ### The secondary `8080` matches the `WEBUI_PORT` on the qBitTorrent container
configuration below.
```

```
            - "127.0.0.1:8888:8080"
        restart: unless-stopped

    ### Uses the LinuxServer.io image of qBitTorrent: https://github.com/linuxserver/docker-qbittorrent/pkgs/container/qbittorrent
    ### You might have to manually update it to 5.1.x, 5.2.x or even 6.x.x eventually.
    ### Keep in mind though that depending on how recent those versions are, they may not be compatible with some private trackers.
    ### Some private trackers have a torrent client whitelist, so while 5.1.x might be whitelisted now, it might be smart to hold off for a few days when 5.2.x or 6.x.x gets released.
    qbittorrent:
        container_name: qbittorrent
        image: ghcr.io/linuxserver/qbittorrent:5.1.2
        restart: unless-stopped
        depends_on:
            - vpn
        ### The `network_mode` line is important. It will force all networking to go through the VPN container. Works as a killswitch should the VPN connection drop.
        network_mode: "service:vpn"
        volumes:
            ### This holds the qBitTorrent configuration. If you want, you could rely on Docker volumes instead. I prefer have it all in the same directory as my Docker Compose yml files.
            - ./config:/config
            ### /data is where I point my downloads towards, you might use /mnt - or something else. Up to you!
            - /data:/data
        environment:
            ### For permissions, you might have to adjust this depending on your user or group ID in Linux.
            ### You can use `id your-username-here` on the server/system you're running Docker to figure out what user ID (uid) and group ID (gid) you wanna use.
            # - PUID=112
            # - PGID=114
            ### Depending on where you live, you might want to change the line below. Europe/Oslo should work for anyone in Central European Time though :)
            - TZ=Europe/Oslo
            - WEBUI_PORT=8080
```

# 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...

### Comments that are prefixed with 3 hashtags (such as this one), should not be uncommented. Otherwise they will break the NGINX config.

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 / {
        ### Only 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. If so,
leave this commented.
        # allow 127.0.0.1;
        # allow 192.168.0.0/24;
        # deny all;

        ### If you want to use a mixture of IP address whitelisting, with fallback to HTTP
authentication where the IP whitelist fails, you can uncomment the section below.
        ### This would allow you to access qBitTorrent on other networks (e.g. when
traveling), but still add an extra layer of protection besides the qBitTorrent's web UI
authentication.
        # satisfy any;
        # allow 127.0.0.1;
        # allow 192.168.0.0/24;
        ### On Debian or Ubuntu, you can install the `apache2-utils` package via apt: `apt
```

```
install apache2-utils`
        ### Which will allow you to run: `htpasswd -c /etc/nginx/.htpasswd-qbittorrent the-
username-you-want-to-use-here`
        ### The command above will prompt for password, unless you specify it after the
username in the command itself.
        ### Note that with HTTP authentication, *both* the username and password are case
sensitive.
        # auth_basic "Restricted Area";
        # auth_basic_user_file /etc/nginx/.htpasswd-qbittorrent;
        ### Don't forget to uncomment this line if you use the HTTP auth + IP whitelist mix
above.
        # deny all;

        ### Make sure the `8888` port here matches the first port in the docker-compose.yml
file
        ### E.g. if your "ports" line has: "127.0.0.1:8123:8080", you would put: `proxy_pass
http://127.0.0.1:8123;`
        proxy_pass http://127.0.0.1:8888;
        proxy_http_version 1.1;

        ### The `8080` 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               127.0.0.1:8080;
        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;
    }
}
```

Created 2021-07-06 10:48:36 UTC by Alex Thomassen
Updated 2025-11-09 20:29:52 UTC by Alex Thomassen