

qBittorrentVPN (hotio) [most UTD notes]

[Link](#)

```
version: "3.7"

services:
  qbittorrent:
    container_name: qbittorrentvpn
    image: ghcr.io/hotio/qbittorrent
    ports:
      - 8992:8992
      - 8118:8118
    environment:
      - WEBUI_PORTS=8992/tcp,8992/udp
      - PUID=998
      - PGID=100
      - UMASK=002
      - TZ=Europe/Paris
      - VPN_ENABLED=true
      - VPN_LAN_NETWORK=192.168.0.0/24
      - VPN_CONF=wg0
      # - VPN_ADDITIONAL_PORTS
      - VPN_IP_CHECK_DELAY=5
      - PRIVOX_ENABLED=false
    volumes:
      - /srv/path/Files/QBittorrentVPN/config:/config
      - /srv/path/Files/QBittorrentVPN/downloads:/downloads
      - /srv/path/Files/QBittorrentVPN/skins:/skins
    cap_add:
      - NET_ADMIN
    sysctls:
      - net.ipv4.conf.all.src_valid_mark=1
      # - net.ipv6.conf.all.disable_ipv6=0
    restart: unless-stopped
```

Default credentials are admin and password is in the logs after container launch

make sure `/srv/path/Files/QBittorrentVPN/downloads` is owned by 998:100 (or whatever puid:pgid you chose)

```
chown -R 998:100 /srv/path/Files/QBittorrentVPN/downloads/
```

Optionnal : set `WebUI\HostHeaderValidation=false` in the `qBittorrent.conf`

If Web UI Stuck on "Unacceptable file type, only regular file is allowed", go to:
"/home/qbittorrent/.config/qBittorrent" and edit the config file:
"WebUI\AlternativeUIEnabled=true" to "WebUI\AlternativeUIEnabled=false"

Alternative WebUI :

<https://github.com/bill-ahmed/qbit-matUI/releases>

<https://github.com/WDaan/VueTorrent/releases>

<https://github.com/jagannatharjun/qbt-theme>

MIGHT NOT WORK WITH SOME BROWSERS ! If so, try a different one.

If VPN stops working, check `wg0.conf` file, change `0.0.0.0/1,128.0.0.0/1` >> `0.0.0.0/0`

Install plugins

E.g.: [Jackett](#)

cd to `./data/nova3/engines`

nano `jackett.json`

```
{
  "api_key": "YOUR_API_KEY_HERE",
  "tracker_first": true,
  "url": "http://127.0.0.1:9117"
}
```

nano `jackett.py`

```
#VERSION: 3.5
# AUTHORS: Diego de las Heras (ngosang@hotmail.es)
# CONTRIBUTORS: ukharley
#             hannsen (github.com/hannsen)

import json
import os
import xml.etree.ElementTree
from urllib.parse import urlencode, unquote
from urllib import request as urllib_request
from http.cookiejar import CookieJar

from novaprinter import prettyPrinter
from helpers import download_file

#####
# load configuration from file
CONFIG_FILE = 'jackett.json'
CONFIG_PATH = os.path.join(os.path.dirname(os.path.realpath(__file__)), CONFIG_FILE)
CONFIG_DATA = {
    'api_key': 'YOUR_API_KEY_HERE', # jackett api
    'tracker_first': False,        # (False/True) add tracker name to beginning of search
    result
    'url': 'http://127.0.0.1:9117', # jackett url
}

def load_configuration():
    global CONFIG_PATH, CONFIG_DATA
    try:
        # try to load user data from file
        with open(CONFIG_PATH) as f:
            CONFIG_DATA = json.load(f)
    except ValueError:
        # if file exists but it's malformed we load add a flag
        CONFIG_DATA['malformed'] = True
    except Exception:
```

```

        # if file doesn't exist, we create it
        with open(CONFIG_PATH, 'w') as f:
            f.write(json.dumps(CONFIG_DATA, indent=4, sort_keys=True))

# do some checks
if any(item not in CONFIG_DATA for item in ['api_key', 'tracker_first', 'url']):
    CONFIG_DATA['malformed'] = True

load_configuration()

#####

class jackett(object):
    name = 'Jackett'
    url = CONFIG_DATA['url'] if CONFIG_DATA['url'][-1] != '/' else CONFIG_DATA['url'][:-1]
    api_key = CONFIG_DATA['api_key']
    supported_categories = {
        'all': None,
        'anime': ['5070'],
        'books': ['8000'],
        'games': ['1000', '4000'],
        'movies': ['2000'],
        'music': ['3000'],
        'software': ['4000'],
        'tv': ['5000'],
    }

    def download_torrent(self, download_url):
        # fix for some indexers with magnet link inside .torrent file
        if download_url.startswith('magnet:?'):
            print(download_url + " " + download_url)
            response = self.get_response(download_url)
            if response is not None and response.startswith('magnet:?'):
                print(response + " " + download_url)
            else:
                print(download_file(download_url))

    def search(self, what, cat='all'):
        what = unquote(what)

```

```

category = self.supported_categories[cat.lower()]

# check for malformed configuration
if 'malformed' in CONFIG_DATA:
    self.handle_error("malformed configuration file", what)
    return

# check api_key
if self.api_key == "YOUR_API_KEY_HERE":
    self.handle_error("api key error", what)
    return

# prepare jaxxet url
params = [
    ('apikey', self.api_key),
    ('q', what)
]
if category is not None:
    params.append(('cat', ','.join(category)))
params = urlencode(params)
jacket_url = self.url + "/api/v2.0/indexers/all/results/torznab/api?%s" % params
response = self.get_response(jacket_url)
if response is None:
    self.handle_error("connection error", what)
    return

# process search results
response_xml = xml.etree.ElementTree.fromstring(response)
for result in response_xml.find('channel').findall('item'):
    res = {}

    title = result.find('title')
    if title is not None:
        title = title.text
    else:
        continue

    tracker = result.find('jackettindexer')
    tracker = '' if tracker is None else tracker.text
    if CONFIG_DATA['tracker_first']:

```

```

        res['name'] = '[%s] %s' % (tracker, title)
    else:
        res['name'] = '%s [%s]' % (title, tracker)

    res['link'] = result.find(self.generate_xpath('magneturl'))
    if res['link'] is not None:
        res['link'] = res['link'].attrib['value']
    else:
        res['link'] = result.find('link')
        if res['link'] is not None:
            res['link'] = res['link'].text
        else:
            continue

    res['size'] = result.find('size')
    res['size'] = -1 if res['size'] is None else (res['size'].text + ' B')

    res['seeds'] = result.find(self.generate_xpath('seeders'))
    res['seeds'] = -1 if res['seeds'] is None else int(res['seeds'].attrib['value'])

    res['leech'] = result.find(self.generate_xpath('peers'))
    res['leech'] = -1 if res['leech'] is None else int(res['leech'].attrib['value'])

    if res['seeds'] != -1 and res['leech'] != -1:
        res['leech'] -= res['seeds']

    res['desc_link'] = result.find('comments')
    if res['desc_link'] is not None:
        res['desc_link'] = res['desc_link'].text
    else:
        res['desc_link'] = result.find('guid')
        res['desc_link'] = '' if res['desc_link'] is None else res['desc_link'].text

    # note: engine_url can't be changed, torrent download stops working
    res['engine_url'] = self.url

    prettyPrinter(self.escape_pipe(res))

```

```

def generate_xpath(self, tag):
    return './{http://torznab.com/schemas/2015/feed}attr[@name="%s"]' % tag

```

```

# Safety measure until it's fixed in prettyPrinter
def escape_pipe(self, dictionary):
    for key in dictionary.keys():
        if isinstance(dictionary[key], str):
            dictionary[key] = dictionary[key].replace('|', '%7C')
    return dictionary

def get_response(self, query):
    response = None
    try:
        # we can't use helpers.retrieve_url because of redirects
        # we need the cookie processor to handle redirects
        opener =
urllib_request.build_opener(urllib_request.HTTPCookieProcessor(CookieJar()))
        response = opener.open(query).read().decode('utf-8')
    except urllib_request.HTTPError as e:
        # if the page returns a magnet redirect, used in download_torrent
        if e.code == 302:
            response = e.url
    except Exception:
        pass
    return response

def handle_error(self, error_msg, what):
    # we need to print the search text to be displayed in qBittorrent when
    # 'Torrent names only' is enabled
    prettyPrinter({
        'seeds': -1,
        'size': -1,
        'leech': -1,
        'engine_url': self.url,
        'link': self.url,
        'desc_link': 'https://github.com/qbittorrent/search-plugins/wiki/How-to-configure-
Jackett-plugin', # noqa
        'name': "Jackett: %s! Right-click this row and select 'Open description page' to
open help. Configuration file: '%s' Search: '%s'" % (error_msg, CONFIG_PATH, what) # noqa
    })

```

```
if __name__ == "__main__":  
    jackedt_se = jackedt()  
    jackedt_se.search("ubuntu server", 'software')
```

Go to qBittorrent, search tab, install plugin, enter path `/data/nova3/engines/jackedt.py`

Revision #13

Created 10 April 2022 23:46:05 by Admin

Updated 21 November 2024 22:16:21 by Admin