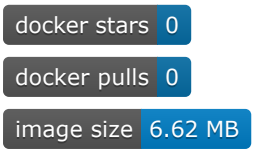
OpenVPN for Docker



OpenVPN server in a Docker container complete with an EasyRSA PKI CA.

FORK

- From kylemanna/openvpn | GitHub kylemanna/docker-openvpn
- Updated to the latest Alpine version
- Multi Arch Image with RaspberryPi support

Quick Start

- Pick a name for the `$OVPN_DATA` data volume container. It's recommended to use the `ovpn-data-` prefix to operate seamlessly with the reference systemd service. Users are encourage to replace `example` with a descriptive name of their choosing.

```
OVPN_DATA="ovpn-data-example"
```

- Initialize the `$OVPN_DATA` container that will hold the configuration files and certificates. The container will prompt for a passphrase to protect the private key used by the newly generated certificate authority.

```
docker volume create --name $OVPN_DATA
docker run -v $OVPN_DATA:/etc/openvpn --rm
martinbouillaud/openvpn:latest ovpn_genconfig -u
udp://VPN.SERVERNAME.COM
docker run -v $OVPN_DATA:/etc/openvpn --rm -it
martinbouillaud/openvpn:latest ovpn_initpki
```

- Start OpenVPN server process

```
docker run -v $OVPN_DATA:/etc/openvpn -d -p 1194:1194/udp --cap-
add=NET_ADMIN martinbouillaud/openvpn:latest
```

- Generate a client certificate without a passphrase

```
docker run -v $OVPN_DATA:/etc/openvpn --rm -it
martinbouillaud/openvpn:latest easyrsa build-client-full CLIENTNAME
nopass
```

- Retrieve the client configuration with embedded certificates

```
docker run -v $OVPN_DATA:/etc/openvpn --rm
martinbouillaud/openvpn:latest ovpn_getclient CLIENTNAME >
CLIENTNAME.ovpn
```

Next Steps

More Reading

Miscellaneous write-ups for advanced configurations are available in the docs folder.

Systemd Init Scripts

A `systemd` init script is available to manage the OpenVPN container. It will start the container on system boot, restart the container if it exits unexpectedly, and pull updates from Docker Hub to keep itself up to date.

Please refer to the systemd documentation to learn more.

Docker Compose

If you prefer to use `docker-compose` try this :

```
version: '3.8'
services:
  openvpn:
    container_name: openvpn
    image: martinbouillaud/openvpn:latest
    restart: always
    cap_add:
     - NET_ADMIN
    ports:
     - "1194:1194/udp"
    volumes:
     - /etc/openvpn:/etc/openvpn
```

Debugging Tips

- Create an environment variable with the name DEBUG and value of 1 to enable debug output (using "docker -e").

```
  docker run -v $OVPN_DATA:/etc/openvpn -p 1194:1194/udp --cap-
add=NET_ADMIN -e DEBUG=1 martinbouillaud/openvpn:latest
```

- Test using a client that has openvpn installed correctly

```
  $ openvpn --config CLIENTNAME.ovpn
```

- Run through a barrage of debugging checks on the client if things don't just work

```
  $ ping 8.8.8.8    # checks connectivity without touching name
resolution
  $ dig google.com  # won't use the search directives in
resolv.conf
  $ nslookup google.com # will use search
```

- Consider setting up a systemd service for automatic start-up at boot time and restart in the event the OpenVPN daemon or Docker crashes.

How Does It Work?

Initialize the volume container using the `martinbouillaud/openvpn:latest` image with the included scripts to automatically generate:

- Diffie-Hellman parameters
- a private key
- a self-certificate matching the private key for the OpenVPN server
- an EasyRSA CA key and certificate
- a TLS auth key from HMAC security

The OpenVPN server is started with the default run cmd of `ovpn_run`

The configuration is located in `/etc/openvpn`, and the Dockerfile declares that directory as a volume. It means that you can start another container with the `-v` argument, and access the configuration.
The volume also holds the PKI keys and certs so that it could be backed up.

To generate a client certificate, `martinbouillaud/openvpn:latest` uses EasyRSA via the `easyrsa` command in the container's path. The `EASYRSA_*` environmental variables place the PKI CA under `/etc/openvpn/pki`.

Conveniently, `martinbouillaud/openvpn:latest` comes with a script called `ovpn_getclient`, which dumps an inline OpenVPN client configuration file. This single file can then be given to a client for access to the VPN.

To enable Two Factor Authentication for clients (a.k.a. OTP) see this document.

OpenVPN Details

We use `tun` mode, because it works on the widest range of devices. `tap` mode, for instance, does not work on Android, except if the device is rooted.

The topology used is `net30`, because it works on the widest range of OS. `p2p`, for instance, does not work on Windows.

The UDP server uses`192.168.255.0/24` for dynamic clients by default.

The client profile specifies `redirect-gateway def1`, meaning that after establishing the VPN connection, all traffic will go through the VPN. This might cause problems if you use local DNS recursors which are not directly reachable, since you will try to reach them through the VPN and they might not answer to you. If that happens, use public DNS resolvers like those of Google (8.8.4.4 and 8.8.8.8) or OpenDNS (208.67.222.222 and 208.67.220.220).

Security Discussion

The Docker container runs its own EasyRSA PKI Certificate Authority. This was chosen as a good way to compromise on security and convenience. The container runs under the assumption that the OpenVPN container is running on a secure host, that is to say that an adversary does not have access to the PKI files under `/etc/openvpn/pki`. This is a fairly reasonable compromise because if an adversary had access to these files, the adversary could manipulate the function of the OpenVPN server itself (sniff packets, create a new PKI CA, MITM packets, etc).

- The certificate authority key is kept in the container by default for simplicity. It's highly recommended to secure the CA key with some passphrase to protect against a filesystem compromise. A more secure system would put the EasyRSA PKI CA on an offline system (can use the same Docker image and the script `ovpn_copy_server_files` to accomplish this).
- It would be impossible for an adversary to sign bad or forged certificates without first cracking the key's passphase should the adversary have root access to the filesystem.
- The EasyRSA `build-client-full` command will generate and leave keys on the server, again possible to compromise and steal the keys. The keys generated need to be signed by the CA which the user hopefully configured with a passphrase as described above.

- Assuming the rest of the Docker container's filesystem is secure, TLS + PKI security should prevent any malicious host from using the VPN.

Benefits of Running Inside a Docker Container

The Entire Daemon and Dependencies are in the Docker Image

This means that it will function correctly (after Docker itself is setup) on all distributions Linux distributions such as: Ubuntu, Arch, Debian, Fedora, etc. Furthermore, an old stable server can run a bleeding edge OpenVPN server without having to install/muck with library dependencies (i.e. run latest OpenVPN with latest OpenSSL on Ubuntu 12.04 LTS).

It Doesn't Stomp All Over the Server's Filesystem

Everything for the Docker container is contained in two images: the ephemeral run time image (martinbouillaud/openvpn:latest) and the `$OVPN_DATA` data volume. To remove it, remove the corresponding containers, `$OVPN_DATA` data volume and Docker image and it's completely removed. This also makes it easier to run multiple servers since each lives in the bubble of the container (of course multiple IPs or separate ports are needed to communicate with the world).

Some (arguable) Security Benefits

At the simplest level compromising the container may prevent additional compromise of the server. There are many arguments surrounding this, but the take away is that it certainly makes it more difficult to break out of the container. People are actively working on Linux containers to make this more of a guarantee in the future.

Differences from jpetazzo/dockvpn

- No longer uses serveconfig to distribute the configuration via https
- Proper PKI support integrated into image
- OpenVPN config files, PKI keys and certs are stored on a storage volume for re-use across containers
- Addition of tls-auth for HMAC security

Originally Tested On

- Docker hosts:
  - server a Digital Ocean Droplet with 512 MB RAM running Ubuntu 14.04
- Clients
  - Android App OpenVPN Connect 1.1.14 (built 56)
    - OpenVPN core 3.0 android armv7a thumb2 32-bit
  - OS X Mavericks with Tunnelblick 3.4beta26 (build 3828) using openvpn-2.3.4
  - ArchLinux OpenVPN pkg 2.3.4-1