OpenVPN 3 Linux

The OpenVPN 3 Linux project is a new client built on top of the OpenVPN 3 Core Library, which is also used in the various OpenVPN Connect clients and OpenVPN for Android (need to be enabled via the settings page in the app).

This client is built around a completely different architecture in regards to usage. It focuses more on allowing ordinary, unprivileged users on a system to start and manage their own VPN sessions. This is possible by building on features available in modern Linux distrubtions. OpenVPN 3 Linux builds on the D-Bus infrastructure, which enables a better privilege separation between various components. In the end this results in OpenVPN 3 Linux requiring very little privileges to run, and only an isolated service responsible for configuring network settings runs with just the few needed elevated privileges to achieve its job. The rest of the OpenVPN 3 Linux runs with no extra privileges.

Even though the project name carries "Linux", it doesn't mean it is restricted to Linux only. Any platform which has D-Bus available should be capable of running this client in theory. But since D-Bus is most commonly used in Linux environments, this will naturally be the primary focus for the project.

The release notes are stored in git tags in the project git repository. They can also be viewed here: https://codeberg.org/OpenVPN/openvpn3-linux/tags

Pre-built packages

Since the release of OpenVPN 3 Linux v21, we will provide packages via different software repositories. Users requiring production stable version should only use the software repositories for stable releases. All the distributions targeting the Enterprise Linux or Long-Term Stable releases will be available through this channel. Other distributions may need to use the repositories for development/beta releases. The stable versions will not have as frequent releases as the development/beta releases.

Supported distributions:

Distribution Vendor	Release	Release name (`DISTRIBUTION`)	Architecture	DCO support	Repositories
Debian	12, 13	bookworm, trixie	amd64, arm64 (*0)	yes	Stable, openSUSE Build Service
Fedora	41, 42, 43, Rawhide (*1)	-	aarch64 (*0), ppc64le, s390x, x86_64	yes	Fedora Copr
Red Hat Enterprise Linux	8	-	aarch64 (*0), ppc64le (*2), s390x (*2), x86_64	yes	Stable, Fedora Copr
Red Hat Enterprise Linux	9	-	aarch64 (*0), ppc64le (*2), s390x (*2), x86_64	yes	Stable, Fedora Copr
Red Hat Enterprise Linux	10	-	aarch64 (*0), ppc64le (*2), s390x (*2), x86_64	yes	Fedora Copr
Ubuntu (LTS)	22.04	jammy	amd64, arm64 (*0)	yes	Stable, openSUSE Build Service
Ubuntu (LTS)	24.04	noble	amd64, arm64 (*0)	yes	Stable, openSUSE Build Service
Ubuntu	25.04	plucky	amd64, arm64 (*0)	yes	openSUSE Build Service
Ubuntu	25.10	questing	amd64, arm64 (*0)	yes	openSUSE Build Service

In many cases, the Red Hat Enterprise Linux packages will also work on Alma Linux and Rocky Linux.

Footnotes

0. ARM64/aarch64 architectures are in tech-preview; please report back your experiences if you use OpenVPN 3 LInux on this platform - good or bad
1. Fedora Rawhide is a moving target and there will be periods where there will not be updates available until we sort out the required changes to the build environment.
2. The ppc64le and s390x are not fully supported and only available via the Fedora Copr repositories. Consider these platforms tech-preview.

Stable repository - Debian / Ubuntu

Ensure you have the needed support packages already installed:

```
# apt install apt-transport-https curl
```

Retrieve the OpenVPN Inc package signing key:

```
# mkdir -p /etc/apt/keyrings    ### This might not exist in all distributions
# curl -sSfL https://packages.openvpn.net/packages-repo.gpg >/etc/apt/keyrings/openvpn.asc
```

Replace the `DISTRIBUTION` part in the command below using the release name from the table above to set up the apt source listing:

```
# echo "deb [signed-by=/etc/apt/keyrings/openvpn.asc] https://packages.openvpn.net/openvpn3/debian DISTRIBUTION main" >>/etc/apt/sources.list.d/openvpn3.list
```

Example for Debian 12:

```
# echo "deb [signed-by=/etc/apt/keyrings/openvpn.asc] https://packages.openvpn.net/openvpn3/debian bookworm main" >>/etc/apt/sources.list.d/openvpn3.list
```

To install OpenVPN 3 Linux, run these commands:

```
# apt update
# apt install openvpn3-client
```

Stable repository - Red Hat Enterprise Linux

Red Hat Enterprise Linux 8 and 9 need to install this package:

```
# dnf install https://packages.openvpn.net/openvpn-openvpn3-epel-repo-1-1.noarch.rpm
```

Red Hat Enterprise Linux 10 need to install this package:

```
# dnf install https://packages.openvpn.net/openvpn-openvpn3-rhel+epel-repo-1-1.noarch.rpm
```

In addition, the Fedora EPEL package and the corresponding Code Ready Builder (or PowerTools on CentOS) must be installed.

To install OpenVPN 3 Linux, run this command:

```
# dnf install openvpn3-client
```

Development/beta repository - Debian / Ubuntu

This repository will have more frequent releases than the stable repository, but packages from this repository will not have been through the same level of QA testing before releases.

We're using openSUSE Build Service to build for the various non-LTS Ubuntu releases and to build Development/Beta versions for all Debian/Ubuntu distributions.

You can find a list of supported Debian/Ubuntu Distributions on the Stable Download page or Beta Download page. Use them as follows (example for Ubuntu 24.04):

```
$ sudo -s
# mkdir -p /etc/apt/keyrings # directory does not exist on older releases
# curl -fsSL https://download.opensuse.org/repositories/isv:/OpenVPN:/Stable/xUbuntu_24.04/Release.key | tee /etc/apt/keyrings/obs-isv-openvpn-snapshots.asc
# echo "deb [arch=<arch> signed-by=/etc/apt/keyrings/obs-isv-openvpn-snapshots.asc] https://download.opensuse.org/repositories/isv:/OpenVPN:/Stable/xUbuntu_24.04 ./" > /etc/apt/sources.list.d/obs-isv-openvpn-snapshots.list
```

Replace `Stable` with `Beta` to use Development/Beta packages.

Fedora Copr repository - Fedora / Red Hat Enterprise Linux

This repository will have more frequent releases than the stable repository, but packages from this repository will not have been through the same level of QA testing before releases.

Ensure the `dnf copr` functionality is installed and ready. Then enable the Fedora Copr repository for OpenVPN 3 Linux:

```
# dnf copr enable dsommers/openvpn3
```

Then OpenVPN 3 Linux can be installed:

```
# dnf install openvpn3-client
```

OpenVPN Data Channel Offload

The OpenVPN Data Channel Offload (OpenVPN DCO) is a kernel module which can accellerate the OpenVPN traffic throughput. OpenVPN 3 Linux uses the same kernel module as OpenVPN 2.6. For Debian and Ubuntu distributions, install the `openvpn-dco-dkms` package. Fedora and Red Hat Enterprise Linux distributions need to install the `kmod-ovpn-dco` package.

With this in installed, VPN sessions can be started with the Data Channel Offload enabled. To test it on an existing configuration:

```
$ openvpn3 session-start --dco true --config CONFIG_NAME
```

To make this persistent each time, use the OpenVPN 3 Configuration Manager:

```
$ openvpn3 config-import --persistent --name CONFIG_NAME --config /path/to/configuration/profile.ovpn
$ openvpn3 config-manager --show --name CONFIG_NAME --dco true
```

Then each time the VPN configuration is started, either via `openvpn3 session-start` or the systemd `openvpn3-sessions@.service` unit file, DCO will be enabled. Please do verify that the log output does indicate that DCO has truly been enabled, as it might be disabled on-the-fly if your configuration profile is not DCO compliant.

A DCO compliant configuration profile cannot use compression features and must use an AEAD based cipher (like AES-GCM or ChaCha20-Poly1305).

Quick start - how to use OpenVPN 3 Linux

With the openvpn3 packages installed, everything should be ready to be used. By default any user account on the system should be able to start and manage their own VPN sessions.

Using `openvpn2`

For users familiar with the classic OpenVPN 2.x command line, the `openvpn2` front-end aims to be fairly close to old behaviour.

```
$ openvpn2 --config ${MY_CONFIGURATION_FILE} --verb 6
```

Replace ${MY_CONFIGURATION_FILE} with the OpenVPN configuration file you want to use. If this configuration includes the `--daemon` option, the VPN session will be started in the background and the user is given the command line back again. To further manage this VPN session, the `openvpn3 session-manage` command line interface must be used. Without `--daemon` the console will be filled with log data from the VPN session and the session can be disconnected via a simple CTRL-C in the terminal.

For more information, see `openvpn2 --help`, `openvpn3 session-manage --help` as well as the openvpn2 and openvpn3-session-manage man pages.

Using `openvpn3`

For more advanced usage, the `openvpn3` command line offers a lot more features. Configuration profiles in OpenVPN 3 Linux are managed by a Configuration Manager before the VPN session is started via the Session Manager. The `openvpn3` utility gives access to the features these manager services provides.

Starting a one-shot configuration profile

A "one-shot configuration profile" means that the configuration file is parsed, loaded and deleted from the the configuration manage as soon as the VPN session has been attempted started. No configuration file is available for re-use after this approach. This is achieved by giving the configuration file to the `openvpn3 session-start` command directly.

```
$ openvpn3 session-start --config ${MY_CONFIGURATION_FILE}
```

Importing a configuration file for re-use and starting a VPN session

Using this approach, an imported configuration file can be used several times and access to the configuration file itself is not needed to start VPN tunnels. By default, configuration profiles imported are only available to the user who imported the configuration file. But OpenVPN 3 Linux also provides an Access Control List feature via openvpn3 config-acl to grant access to specific or all users on the system.

```
$ openvpn3 config-import --config ${MY_CONFIGURATION_FILE}
```

This loads the configuration profile and stores it in memory-only. That means, if the system is rebooted, the configuration profile is not preserved. If the `--persistent` argument is added to the command line above, the configuration profile will be saved to disk in a directory only accessible by the `openvpn` user. Whenever the Configuration Manager is started, configuration files imported with `--persistent` will be automatically loaded as well.

To list all available configuration profiles, run this command:

```
$ openvpn3 configs-list
```

A configuration file typically contains generic options to be able to connect to a specific server, regardless of the device itself. OpenVPN 3 Linux also supports setting more host-specific settings on a configuration profile as well. This is handled via the `openvpn3 config-manage` interface. Any settings here will also be preserved across boots if the configuration profile was imported with the `--persistent` argument.

Starting a new VPN session from an imported configuration profile

When a configuration profile is available via `openvpn3 configs-list`, it can easily be started via `openvpn3 session-start` using the configuration profile name (typically the filename used during the import)

```
$ openvpn3 session-start --config ${CONFIGURATION_PROFILE_NAME}
```

or it is possible to use the D-Bus path to the configuration profile:

```
$ openvpn3 session-start --config-path /net/openvpn/v3/configuration/.........
```

In either of these cases is it necessarily to have access to the configuration profile on disk. As long as configuration profiles are available via `openvpn3 configs-list`, all needed to start a VPN session should be present.

Managing a running VPN session

Once a VPN session has started, it should be seen in `openvpn3 sessions-list`:

```
$ openvpn3 sessions-list
```

Using the `openvpn3 session-manage` there are a few things which can be done, but most typically it is the `--disconnect` or `--restart` alternatives which is most commonly used.

```
$ openvpn3 session-manage --config ${CONFIGURATION_PROFILE_NAME} --restart
```

This disconnects and re-connects to the server again, re-establishing the connection. The `${CONFIGURATION_PROFILE_NAME}` is the configuration name as displayed in `openvpn3 sessions-list`. It is also possible to use the D-Bus path to the session as well:

```
$ openvpn3 session-manage --session-path /net/openvpn/v3/sessions/..... --disconnect
```

This command above will disconnect a running session. Once this operation has completed, it will be removed from the `openvpn3 sessions-list` overview.

It is also possible to retrieve real-time tunnel statistics from running sessions:

```
$ openvpn3 session-stats --config ${CONFIGURATION_PROFILE_NAME}
      $ openvpn3 session-stats --session-path /net/openvpn/v3/sessions/.....
```

And to retrieve real-time log events as they occur, run the `openvpn3 log` command line below:

```
$ openvpn3 log --config ${CONFIGURATION_PROFILE_NAME}
```

This might be quite silent, as it does not provide any log events from the past. Issue an `openvpn3 session-manage --restart` from a different terminal, and log events will occur. You may want to boost the log-level with `--log-level 6`. Valid log levels are from 0 to 6, where 6 is the most verbose.

Note that the maximum log level is configured centrally. If you don't get more output with higher log levels increase maximum log level first with `openvpn3-admin` (note that this command needs to be executed as root):

```
# openvpn3-admin log-service --log-level 6
```

VPN sessions are also owned by the user which started it. But the Session Manager also provides its own Access Control List feature via `openvpn3 session-acl`.

Further information

- man pages:
  - openvpn3-linux(7) - Main overview
  - openvpn3(1) - `openvpn3` command line interface
  - openvpn2(1) - `openvpn2` command line interface which is similar to the classic OpenVPN 2.x interface
  - openvpn3-systemd(8) - Managing OpenVPN 3 Linux via systemd `systemctl`
  - More man pages

- Developers / D-Bus API documentation
    - D-Bus Primer - Understanding D-Bus
    - OpenVPN 3 D-Bus overview - Overview of all D-Bus services which are provided and used
    - Debugging - How to debug OpenVPN 3 Linux
    - More D-Bus documentation