CONTENTS

RELATED

Initial Server Setup with Ubuntu 12.04
View

How To Install Ruby on Rails on Ubuntu 12.04 LTS (Precise Pangolin) with RVM
View

// Tutorial //

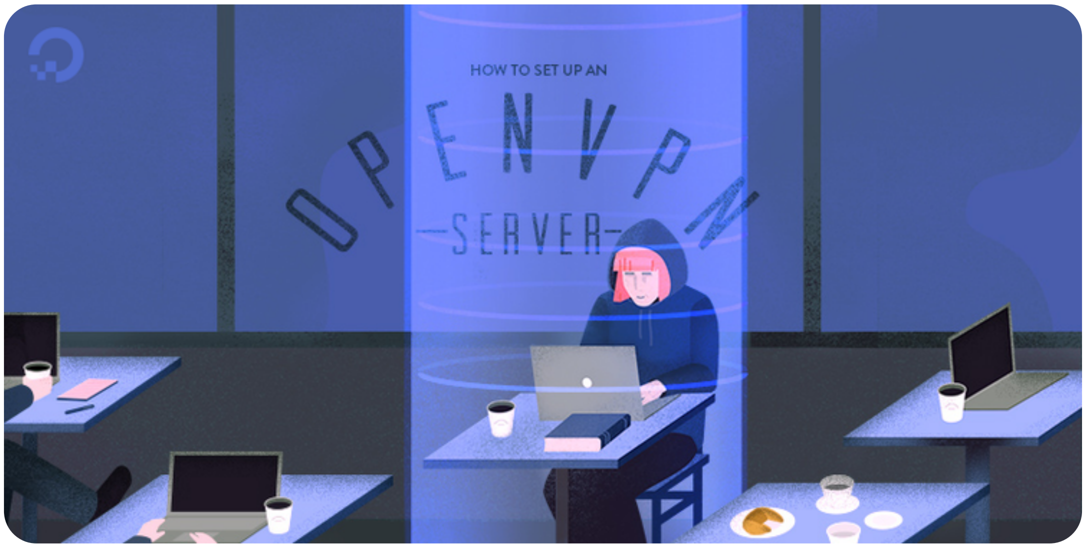
How To Set Up and Configure an OpenVPN Server on Ubuntu 20.04

Published on May 6, 2020

Ubuntu VPN Ubuntu 20.04

By Jamon Camisso

English



Not using Ubuntu 20.04 ?
Choose a different version or distribution.

Ubuntu 20.04

# Introduction

A Virtual Private Network (VPN) allows you to traverse untrusted networks as if you were on a private network. It gives you the freedom to access the internet safely and securely from your smartphone or laptop when connected to an untrusted network, like the WiFi at a hotel or coffee shop.

When combined with HTTPS connections, this setup allows you to secure your wireless logins and transactions. You can circumvent geographical restrictions and censorship, and shield your location and any unencrypted HTTP traffic from untrusted networks.

OpenVPN is a full featured, open-source Transport Layer Security (TLS) VPN solution that accommodates a wide range of configurations. In this tutorial, you will set up OpenVPN on an Ubuntu 20.04 server, and then configure it to be accessible from a client machine.

Note: If you plan to set up an OpenVPN Server on a DigitalOcean Droplet, be aware that we, like many hosting providers, charge for bandwidth overages. For this reason, please be mindful of how much traffic your server is handling.

See this page for more info.

# Prerequisites

To follow this tutorial, you will need:

- One Ubuntu 20.04 server with a sudo non-root user and a firewall enabled. To set this up, you can follow our Initial Server Setup with Ubuntu 20.04 tutorial. We will refer to this as the OpenVPN Server throughout this guide.
- A separate Ubuntu 20.04 server set up as a private Certificate Authority (CA), which we will refer to as the CA Server throughout this guide. After executing the steps from the Initial Server Setup Guide on this server, you can follow steps 1 to 3 of our guide on How To Set Up and Configure a Certificate Authority (CA) on Ubuntu 20.04 to accomplish that.

Note: While it is technically possible to use your OpenVPN Server or your local machine as your CA, this is not recommended as it opens up your VPN to some security vulnerabilities. Per the official OpenVPN documentation, you should place your CA on a standalone machine that's dedicated to importing and signing certificate requests. For this reason, this guide assumes that your CA is on a separate Ubuntu 20.04 server that also has a non-root user with sudo privileges and a basic firewall enabled.

In addition to that, you'll need a client machine which you will use to connect to your OpenVPN Server. In this guide, we'll call this the OpenVPN Client . For the purposes of this tutorial, it's recommended that you use your local machine as the OpenVPN client.

With these prerequisites in place, you are ready to begin setting up and configuring an OpenVPN Server on Ubuntu 20.04.

Note: Please note that if you disable password authentication while configuring these servers, you may run into difficulties when transferring files between them later on in this guide. To resolve this issue, you could re-enable password authentication on each server. Alternatively, you could generate an SSH keypair for each server, then add the OpenVPN Server's public SSH key to the CA machine's `authorized_keys` file and vice versa. See How to Set Up SSH Keys on Ubuntu 20.04 for instructions on how to perform either of these solutions.

# Step 1 – Installing OpenVPN and Easy-RSA

The first step in this tutorial is to install OpenVPN and Easy-RSA. Easy-RSA is a public key infrastructure (PKI) management tool that you will use on the OpenVPN Server to generate a certificate request that you will then verify and sign on the CA Server.

To start off, update your OpenVPN Server's package index and install OpenVPN and Easy-RSA. Both packages are available in Ubuntu's default repositories, so you can use `apt` for the installation:

```
$ sudo apt update
$ sudo apt install openvpn easy-rsa
```
Copy

Next you will need to create a new directory on the OpenVPN Server as your non-root user called `~/easy-rsa`:

```
$ mkdir ~/easy-rsa
```
Copy

Now you will need to create a symlink from the `easyrsa` script that the package installed into the `~/easy-rsa` directory that you just created:

```
$ ln -s /usr/share/easy-rsa/* ~/easy-rsa/
```
Copy

Note: While other guides might instruct you to copy the `easy-rsa` package files into your PKI directory, this tutorial adopts a symlink approach. As a result, any updates to the `easy-rsa` package will be automatically reflected in your PKI's scripts.

Finally, ensure the directory's owner is your non-root sudo user and restrict access to that user using `chmod`:

```
$ sudo chown sammy ~/easy-rsa
$ chmod 700 ~/easy-rsa
```
Copy

Once these programs are installed and have been moved to the right locations on your system, the next step is to create a Public Key Infrastructure (PKI) on the OpenVPN server so that you can request and manage TLS certificates for clients and other servers that will connect to your VPN.

# Step 2 – Creating a PKI for OpenVPN

Before you can create your OpenVPN server's private key and certificate, you need to create a local Public Key Infrastructure directory on your OpenVPN server. You will use this directory to manage the server and clients' certificate requests instead of making them directly on your CA server.

To build a PKI directory on your OpenVPN server, you'll need to populate a file called `vars` with some default values. First you will `cd` into the `easy-rsa` directory, then you will create and edit the `vars` file using nano or your preferred text editor.

```
$ cd ~/easy-rsa
$ nano vars
```

Copy

Once the file is opened, paste in the following two lines:

~/easy-rsa/vars

```
set_var EASYRSA_ALGO "ec"
set_var EASYRSA_DIGEST "sha512"
```

These are the only two lines that you need in this `vars` file on your OpenVPN server since it will not be used as a Certificate Authority. They will ensure that your private keys and certificate requests are configured to use modern Elliptic Curve Cryptography (ECC) to generate keys and secure signatures for your clients and OpenVPN server.

Configuring your OpenVPN & CA servers to use ECC means when a client and server attempt to establish a shared symmetric key, they can use Elliptic Curve algorithms to do their exchange. Using ECC for a key exchange is significantly faster than using plain Diffie-Hellman with the classic RSA algorithm since the numbers are much smaller and the computations are faster.

Background: When clients connect to OpenVPN, they use asymmetric encryption (also known as public/private key) to perform a TLS handshake. However, when transmitting encrypted VPN traffic, the server and clients use symmetric encryption, which is also known as shared key encryption.

There is much less computational overhead with symmetric encryption compared to asymmetric: the numbers that are used are much smaller, and modern CPUs integrate instructions to perform optimized symmetric encryption operations. To make the switch from asymmetric to symmetric encryption, the OpenVPN server and client will use the Elliptic Curve Diffie-Hellman (ECDH) algorithm to agree on a shared secret key as quickly as possible.

Once you have populated the `vars` file you can proceed with creating the PKI directory. To do so, run the `easyrsa` script with the `init-pki` option. Although you already ran this command on the CA server as part of the prerequisites, it's necessary to run it here because your OpenVPN server and CA server have separate PKI directories:

```
$ ./easyrsa init-pki
```

Copy

Note that on your OpenVPN server there is no need to create a Certificate Authority. Your CA server is solely responsible for validating and signing certificates. The PKI on your VPN server is only used as a convenient and centralized place to store certificate requests and public certificates.

After you've initialized your PKI on the OpenVPN server, you are ready to move on to the next step, which is creating an OpenVPN server certificate request and private key.

# Step 3 — Creating an OpenVPN Server Certificate Request and Private Key

Now that your OpenVPN server has all the prerequisites installed, the next step is to generate a private key and Certificate Signing Request (CSR) on your OpenVPN server. After that you'll transfer the request over to your CA to be signed, creating the required certificate. Once you have a signed certificate, you'll transfer it back to the OpenVPN server and install it for the server to use.

To start, navigate to the `~/easy-rsa` directory on your OpenVPN Server as your non-root user:

```
$ cd ~/easy-rsa
```
Copy

Now you'll call the `easyrsa` with the `gen-req` option followed by a Common Name (CN) for the machine. The CN can be anything you like but it can be helpful to make it something descriptive. Throughout this tutorial, the OpenVPN Server's CN will be `server`. Be sure to include the `nopass` option as well. Failing to do so will password-protect the request file which could lead to permissions issues later on.

Note: If you choose a name other than `server` here, you will have to adjust some of the instructions below. For instance, when copying the generated files to the `/etc/openvpn` directory, you will have to substitute the correct names. You will also have to modify the `/etc/openvpn/server.conf` file later to point to the correct `.crt` and `.key` files.

```
$ ./easyrsa gen-req server nopass
```
Copy

```
Output
Common Name (eg: your user, host, or server name) [server]:

Keypair and certificate request completed. Your files are:
req: /home/sammy/easy-rsa/pki/reqs/server.req
key: /home/sammy/easy-rsa/pki/private/server.key
```

This will create a private key for the server and a certificate request file called `server.req`. Copy the server key to the `/etc/openvpn/server` directory:

```
$ sudo cp /home/sammy/easy-rsa/pki/private/server.key /etc/openvpn/server/
```
Copy

After completing these steps, you have successfully created a private key for your OpenVPN server. You have also generated a Certificate Signing Request for the OpenVPN server. The CSR is now ready for signing by your CA. In the next section of this tutorial you will learn how to sign a CSR with your CA server's private key.

# Step 4 – Signing the OpenVPN Server's Certificate Request

In the previous step you created a Certificate Signing Request (CSR) and private key for the OpenVPN server. Now the CA server needs to know about the `server` certificate and validate it. Once the CA validates and relays the certificate back to the OpenVPN server, clients that trust your CA will be able to trust the OpenVPN server as well.

On the OpenVPN server, as your non-root user, use SCP or another transfer method to copy the `server.req` certificate request to the CA server for signing:

```
$ scp /home/sammy/easy-rsa/pki/reqs/server.req sammy@your_ca_server_ip:/tmp
```
Copy

If you followed the prerequisite How To Set Up and Configure a Certificate Authority (CA) on Ubuntu 20.04 tutorial, the next step is to log in to the CA server as the non-root user that you created to manage your CA. You'll `cd` to the `~/easy-rsa` directory where you created your PK and then import the certificate request using the `easyrsa` script:

```
$ cd ~/easy-rsa
$ ./easyrsa import-req /tmp/server.req server
```
Copy

```
Output
```

```
. . .
The request has been successfully imported with a short name of: server
You may now use this name to perform signing operations on this request.
```

Next, sign the request by running the `easyrsa` script with the `sign-req` option, followed by the request type and the Common Name. The request type can either be `client` or `server`. Since we're working with the OpenVPN server's certificate request, be sure to use the `server` request type:

```
$ ./easyrsa sign-req server server
```
Copy

In the output, you'll be prompted to verify that the request comes from a trusted source. Type `yes` then press `ENTER` to confirm:

```
Output
You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 3650 days:

subject=
commonName = server


Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
. . .
Certificate created at: /home/sammy/easy-rsa/pki/issued/server.crt
```

Note that if you encrypted your CA private key, you'll be prompted for your password at this point.

With those steps complete, you have signed the OpenVPN server's certificate request using the CA server's private key. The resulting `server.crt` file contains the OpenVPN server's public encryption key, as well as a signature from the CA server. The point of the signature is to tell anyone who trusts the CA server that they can also trust the OpenVPN server when they connect to it.

To finish configuring the certificates, copy the `server.crt` and `ca.crt` files from the CA server to the OpenVPN server:

```
$ scp pki/issued/server.crt sammy@your_vpn_server_ip:/tmp
$ scp pki/ca.crt sammy@your_vpn_server_ip:/tmp
```
Copy

Now back on your OpenVPN server, copy the files from `/tmp` to `/etc/openvpn/server`:

```
$ sudo cp /tmp/{server.crt,ca.crt} /etc/openvpn/server
```
Copy

Now your OpenVPN server is nearly ready to accept connections. In the next step you'll perform some additional steps to increase the security of the server.

Step 5 — Configuring OpenVPN Cryptographic Material

For an additional layer of security, we'll add an extra shared secret key that the server and all clients will use with OpenVPN's `tls-crypt` directive. This option is used to obfuscate the TLS certificate that is used when a server and client connect to each other initially. It is also used by the OpenVPN server to perform quick checks on incoming packets: if a packet is signed using the pre-shared key, then the server processes it; if it is not signed, then the server knows it is from an untrusted source and can discard it without having to perform additional decryption work.

This option will help ensure that your OpenVPN server is able to cope with unauthenticated traffic, port scans, and Denial of Service attacks, which can tie up server resources. It also makes it harder to identify

OpenVPN network traffic.

To generate the `tls-crypt` pre-shared key, run the following on the OpenVPN server in the `~/easy-rsa` directory:

```
$ cd ~/easy-rsa
$ openvpn --genkey --secret ta.key
```

Copy

The result will be a file called `ta.key`. Copy it to the `/etc/openvpn/server/` directory:

```
$ sudo cp ta.key /etc/openvpn/server
```

Copy

With these files in place on the OpenVPN server you are ready to create client certificates and key files for your users, which you will use to connect to the VPN.

# Step 6 — Generating a Client Certificate and Key Pair

Although you can generate a private key and certificate request on your client machine and then send it to the CA to be signed, this guide outlines a process for generating the certificate request on the OpenVPN server. The benefit of this approach is that we can create a script that will automatically generate client configuration files that contain all of the required keys and certificates. This lets you avoid having to transfer keys, certificates, and configuration files to clients and streamlines the process of joining the VPN.

We will generate a single client key and certificate pair for this guide. If you have more than one client, you can repeat this process for each one. Please note, though, that you will need to pass a unique name value to the script for every client. Throughout this tutorial, the first certificate/key pair is referred to as `client1`.

Get started by creating a directory structure within your home directory to store the client certificate and key files:

```
$ mkdir -p ~/client-configs/keys
```

Copy

Since you will store your clients' certificate/key pairs and configuration files in this directory, you should lock down its permissions now as a security measure:

```
$ chmod -R 700 ~/client-configs
```

Copy

Next, navigate back to the EasyRSA directory and run the `easyrsa` script with the `gen-req` and `nopass` options, along with the common name for the client:

```
$ cd ~/easy-rsa
$ ./easyrsa gen-req client1 nopass
```

Copy

Press `ENTER` to confirm the common name. Then, copy the `client1.key` file to the `~/client-configs/keys/` directory you created earlier:

```
$ cp pki/private/client1.key ~/client-configs/keys/
```

Copy

Next, transfer the `client1.req` file to your CA Server using a secure method:

```
$ scp pki/reqs/client1.req sammy@your_ca_server_ip:/tmp
```

Copy

Now log in to your CA Server. Then, navigate to the EasyRSA directory, and import the certificate request:

```
$ cd ~/easy-rsa
$ ./easyrsa import-req /tmp/client1.req client1
```
Copy

Next, sign the request the same way as you did for the server in the previous step. This time, though, be sure to specify the `client` request type:

```
$ ./easyrsa sign-req client client1
```
Copy

When prompted, enter `yes` to confirm that you intend to sign the certificate request and that it came from a trusted source:

```
Output
Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
```

Again, if you encrypted your CA key, you'll be prompted for your password here.

This will create a client certificate file named `client1.crt`. Transfer this file back to the server:

```
$ scp pki/issued/client1.crt sammy@your_server_ip:/tmp
```
Copy

Back on your OpenVPN server, copy the client certificate to the `~/client-configs/keys/` directory:

```
$ cp /tmp/client1.crt ~/client-configs/keys/
```
Copy

Next, copy the `ca.crt` and `ta.key` files to the `~/client-configs/keys/` directory as well, and set the appropriate permissions for your sudo user:

```
$ cp ~/easy-rsa/ta.key ~/client-configs/keys/
$ sudo cp /etc/openvpn/server/ca.crt ~/client-configs/keys/
$ sudo chown sammy.sammy ~/client-configs/keys/*
```
Copy

With that, your server and client's certificates and keys have all been generated and are stored in the appropriate directories on your OpenVPN server. There are still a few actions that need to be performed with these files, but those will come in a later step. For now, you can move on to configuring OpenVPN.

Step 7 — Configuring OpenVPN

Like many other widely used open-source tools, OpenVPN has numerous configuration options available to customize your server for your specific needs. In this section, we will provide instructions on how to set up an OpenVPN server configuration based on one of the sample configuration files that is included within this software's documentation.

First, copy the sample `server.conf` file as a starting point for your own configuration file:

```
$ sudo cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /etc/openvpn/ser
$ sudo gunzip /etc/openvpn/server/server.conf.gz
```
Copy

Open the new file for editing with the text editor of your choice. We'll use nano in our example:

```
$ sudo nano /etc/openvpn/server/server.conf
```
Copy

We'll need to change a few lines in this file. First, find the `HMAC` section of the configuration by searching for the `tls-auth` directive. This line will be enabled by default. Comment it out by adding a `;` to the beginning of the line. Then add a new line after it containing the value `tls-crypt ta.key` only:

/etc/openvpn/server/server.conf

```
;tls-auth ta.key 0 # This file is secret
tls-crypt ta.key
```

Next, find the section on cryptographic ciphers by looking for the `cipher` lines. The default value is set to `AES-256-CBC`, however, the `AES-256-GCM` cipher offers a better level of encryption, performance, and is well supported in up-to-date OpenVPN clients. We'll comment out the default value by adding a `;` sign to the beginning of this line, and then we'll add another line after it containing the updated value of `AES-256-GCM`:

/etc/openvpn/server/server.conf

```
;cipher AES-256-CBC
cipher AES-256-GCM
```

Right after this line, add an `auth` directive to select the HMAC message digest algorithm. For this, `SHA256` is a good choice:

/etc/openvpn/server/server.conf

```
auth SHA256
```

Next, find the line containing a `dh` directive, which defines Diffie-Hellman parameters. Since we've configured all the certificates to use Elliptic Curve Cryptography, there is no need for a Diffie-Hellman seed file. Comment out the existing line that looks like `dh dh2048.pem` or `dh dh.pem`. The filename for the Diffie-Hellman key may be different than what is listed in the example server configuration file. Then add a line after it with the contents `dh none`:

/etc/openvpn/server/server.conf

```
;dh dh2048.pem
dh none
```

Next, we want OpenVPN to run with no privileges once it has started, so we need to tell it to run with a user nobody and group nogroup. To enable this, find and uncomment the `user nobody` and `group nogroup` lines by removing the `;` sign from the beginning of each line:

/etc/openvpn/server/server.conf

```
user nobody
group nogroup
```

# (Optional) Push DNS Changes to Redirect All Traffic Through the VPN

The settings above will create the VPN connection between your client and server, but will not force any connections to use the tunnel. If you wish to use the VPN to route all of your client traffic over the VPN, you will likely want to push some extra settings to the client computers.

To get started, find and uncomment the line containing `push "redirect-gateway def1 bypass-dhcp"`. Doing this will tell your client to redirect all of its traffic through your OpenVPN Server. Be aware that enabling this functionality can cause connectivity issues with other network services, like SSH:

/etc/openvpn/server/server.conf

```
push "redirect-gateway def1 bypass-dhcp"
```

Just below this line, find the `dhcp-option` section. Again, remove the `;` from the beginning of both of the lines to uncomment them:

/etc/openvpn/server/server.conf

```
push "dhcp-option DNS 208.67.222.222"
push "dhcp-option DNS 208.67.220.220"
```

These lines will tell your client to use the free OpenDNS resolvers at the listed IP addresses. If you prefer other DNS resolvers you can substitute them in place of the highlighted IPs.

This will assist clients in reconfiguring their DNS settings to use the VPN tunnel as the default gateway.

# (Optional) Adjust the Port and Protocol

By default, the OpenVPN server uses port `1194` and the UDP protocol to accept client connections. If you need to use a different port because of restrictive network environments that your clients might be in, you can change the `port` option. If you are not hosting web content on your OpenVPN server, port `443` is a popular choice since it is usually allowed through firewall rules.

To change OpenVPN to listen on port 443, open the `server.conf` file and find the line that looks like this:

/etc/openvpn/server/server.conf

```
port 1194
```

Edit it so that the port is 443:

/etc/openvpn/server/server.conf

```
# Optional!
port 443
```

Oftentimes, the protocol is restricted to that port as well. If so, find the `proto` line below the `port` line and change the protocol from `udp` to `tcp`:

/etc/openvpn/server/server.conf

```
# Optional!
proto tcp
```

If you do switch the protocol to TCP, you will need to change the `explicit-exit-notify` directive's value from `1` to `0`, as this directive is only used by UDP. Failing to do so while using TCP will cause errors when you start the OpenVPN service.

Find the `explicit-exit-notify` line at the end of the file and change the value to `0`:

/etc/openvpn/server/server.conf

```
# Optional!
explicit-exit-notify 0
```

If you have no need to use a different port and protocol, it is best to leave these settings unchanged.

# (Optional) Point to Non-Default Credentials

If you selected a different name during the `./easyrsa gen-req server` command earlier, modify the `cert` and `key` lines in the `server.conf` configuration file so that they point to the appropriate `.crt` and `.key` files. If you used the default name, `server`, this is already set correctly:

/etc/openvpn/server/server.conf

```
cert server.crt
key server.key
```

When you are finished, save and close the file.

You have now finished configuring your OpenVPN general settings. In the next step, we'll customize the server's networking options.

# Step 8 — Adjusting the OpenVPN Server Networking Configuration

There are some aspects of the server's networking configuration that need to be tweaked so that OpenVPN can correctly route traffic through the VPN. The first of these is *IP forwarding*, a method for determining where IP traffic should be routed. This is essential to the VPN functionality that your server will provide.

To adjust your OpenVPN server's default IP forwarding setting, open the `/etc/sysctl.conf` file using `nano` or your preferred editor:

```
$ sudo nano /etc/sysctl.conf
```
Copy

Then add the following line at the bottom of the file:

/etc/sysctl.conf

```
net.ipv4.ip_forward = 1
```

Save and close the file when you are finished.

To read the file and load the new values for the current session, type:

```
$ sudo sysctl -p
```
Copy

```
Output
net.ipv4.ip_forward = 1
```

Now your OpenVPN server will be able to forward incoming traffic from one ethernet device to another. This setting makes sure the server can direct traffic from clients that connect on the virtual VPN interface out over its other physical ethernet devices. This configuration will route all web traffic from your client via your server's IP address, and your client's public IP address will effectively be hidden.

In the next step you will need to configure some firewall rules to ensure that traffic to and from your OpenVPN server flows properly.

# Step 9 — Firewall Configuration

So far, you've installed OpenVPN on your server, configured it, and generated the keys and certificates needed for your client to access the VPN. However, you have not yet provided OpenVPN with any instructions on where to send incoming web traffic from clients. You can stipulate how the server should handle client traffic by establishing some firewall rules and routing configurations.

Assuming you followed the prerequisites at the start of this tutorial, you should already have `ufw` installed and running on your server. To allow OpenVPN through the firewall, you'll need to enable masquerading, an iptables concept that provides on-the-fly dynamic network address translation (NAT) to correctly route client connections.

Before opening the firewall configuration file to add the masquerading rules, you must first find the public network interface of your machine. To do this, type:

```
$ ip route list default
```
Copy

Your public interface is the string found within this command's output that follows the word "dev". For example, this result shows the interface named `eth0`, which is highlighted below:

```
Output
default via 159.65.160.1 dev eth0 proto static
```

When you have the interface associated with your default route, open the `/etc/ufw/before.rules` file to add the relevant configuration:

```
$ sudo nano /etc/ufw/before.rules
```
Copy

UFW rules are typically added using the `ufw` command. Rules listed in the `before.rules` file, though, are read and put into place before the conventional UFW rules are loaded. Towards the top of the file, add the highlighted lines below. This will set the default policy for the `POSTROUTING` chain in the `nat` table and masquerade any traffic coming from the VPN. Remember to replace `eth0` in the `-A POSTROUTING` line below with the interface you found in the above command:

/etc/ufw/before.rules

```
#
# rules.before
#
# Rules that should be run before the ufw command line added rules. Custom
# rules should be added to one of these chains:
#   ufw-before-input
#   ufw-before-output
#   ufw-before-forward
#

# START OPENVPN RULES
# NAT table rules
*nat
:POSTROUTING ACCEPT [0:0]
# Allow traffic from OpenVPN client to eth0 (change to the interface you discovered!)
-A POSTROUTING -s 10.8.0.0/8 -o eth0 -j MASQUERADE
COMMIT
# END OPENVPN RULES

# Don't delete these required lines, otherwise there will be errors
*filter
. . .
```

Save and close the file when you are finished.

Next, you need to tell UFW to allow forwarded packets by default as well. To do this, open the `/etc/default/ufw` file:

```
$ sudo nano /etc/default/ufw
```
Copy

Inside, find the `DEFAULT_FORWARD_POLICY` directive and change the value from `DROP` to `ACCEPT`:

/etc/default/ufw

```
DEFAULT_FORWARD_POLICY="ACCEPT"
```

Save and close the file when you are finished.

Next, adjust the firewall itself to allow traffic to OpenVPN. If you did not change the port and protocol in the `/etc/openvpn/server.conf` file, you will need to open up UDP traffic to port `1194`. If you modified the port and/or protocol, substitute the values you selected here.

In case you forgot to add the SSH port when following the prerequisite tutorial, add it here as well:

```
$ sudo ufw allow 1194/udp
```
Copy

```
$ sudo ufw allow OpenSSH
```

Note: If you are using a different firewall or have customized your UFW configuration, you may need to add additional firewall rules. For example, if you decide to tunnel all of your network traffic over the VPN connection, you will need to ensure that port `53` traffic is allowed for DNS requests, and ports like `80` and `443` for HTTP and HTTPS traffic respectively. If there are other protocols that you are using over the VPN then you will need to add rules for them as well.

After adding those rules, disable and re-enable UFW to restart it and load the changes from all of the files you've modified:

```
$ sudo ufw disable
$ sudo ufw enable
```

Copy

Your server is now configured to correctly handle OpenVPN traffic. With the firewall rules in place, we can start the OpenVPN service on the server.

# Step 10 – Starting OpenVPN

OpenVPN runs as a `systemd` service, so we can use `systemctl` to manage it. We will configure OpenVPN to start up at boot so you can connect to your VPN at any time as long as your server is running. To do this, enable the OpenVPN service by adding it to `systemctl`:

```
$ sudo systemctl -f enable openvpn-server@server.service
```

Copy

Then start the OpenVPN service:

```
$ sudo systemctl start openvpn-server@server.service
```

Copy

Double check that the OpenVPN service is active with the following command. You should see `active (running)` in the output:

```
$ sudo systemctl status openvpn-server@server.service
```

Copy

```
Output
● openvpn-server@server.service - OpenVPN service for server
     Loaded: loaded (/lib/systemd/system/openvpn-server@.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2020-04-29 15:39:59 UTC; 6s ago
       Docs: man:openvpn(8)
             https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
             https://community.openvpn.net/openvpn/wiki/HOWTO
   Main PID: 16872 (openvpn)
     Status: "Initialization Sequence Completed"
      Tasks: 1 (limit: 1137)
     Memory: 1.0M
     CGroup: /system.slice/system-openvpn\x2dserver.slice/openvpn-server@server.service
             └─16872 /usr/sbin/openvpn --status /run/openvpn-server/status-server.log --status-version
. . .
. . .
Apr 29 15:39:59 ubuntu-20 openvpn[16872]: Initialization Sequence Completed
```

We've now completed the server-side configuration for OpenVPN. Next, you will configure your client machine and connect to the OpenVPN Server.

# Step 11 – Creating the Client Configuration Infrastructure

Creating configuration files for OpenVPN clients can be somewhat involved, as every client must have its own config and each must align with the settings outlined in the server's configuration file. Rather than writing a single configuration file that can only be used on one client, this step outlines a process for building a client configuration infrastructure which you can use to generate config files on-the-fly. You will first create a "base" configuration file then build a script which will allow you to generate unique client config files, certificates, and keys as needed.

Get started by creating a new directory where you will store client configuration files within the `client-configs` directory you created earlier:

```
$ mkdir -p ~/client-configs/files
```
Copy

Next, copy an example client configuration file into the `client-configs` directory to use as your base configuration:

```
$ cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf ~/client-configs/base.co
```
Copy

Open this new file using `nano` or your preferred text editor:

```
$ nano ~/client-configs/base.conf
```
Copy

Inside, locate the `remote` directive. This points the client to your OpenVPN server address — the public IP address of your OpenVPN server. If you decided to change the port that the OpenVPN server is listening on, you will also need to change `1194` to the port you selected:

~/client-configs/base.conf

```
. . .
# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote your_server_ip 1194
. . .
```

Be sure that the protocol matches the value you are using in the server configuration:

~/client-configs/base.conf

```
proto udp
```

Next, uncomment the `user` and `group` directives by removing the `;` sign at the beginning of each line:

~/client-configs/base.conf

```
# Downgrade privileges after initialization (non-Windows only)
user nobody
group nogroup
```

Find the directives that set the `ca`, `cert`, and `key`. Comment out these directives since you will add the certs and keys within the file itself shortly:

~/client-configs/base.conf

```
# SSL/TLS parms.
# See the server config file for more
# description. It's best to use
# a separate .crt/.key file pair
# for each client. A single ca
# file can be used for all clients.
;ca ca.crt
;cert client.crt
;key client.key
```

Similarly, comment out the `tls-auth` directive, as you will add `ta.key` directly into the client configuration file (and the server is set up to use `tls-crypt`):

~/client-configs/base.conf

```
# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth ta.key 1
```

Mirror the `cipher` and `auth` settings that you set in the `/etc/openvpn/server/server.conf` file:

~/client-configs/base.conf

```
cipher AES-256-GCM
auth SHA256
```

Next, add the `key-direction` directive somewhere in the file. You must set this to "1" for the VPN to function correctly on the client machine:

~/client-configs/base.conf

```
key-direction 1
```

Finally, add a few commented out lines to handle various methods that Linux based VPN clients will use for DNS resolution. You'll add two similar, but separate sets of commented out lines. The first set is for clients that *do not* use `systemd-resolved` to manage DNS. These clients rely on the `resolvconf` utility to update DNS information for Linux clients.

~/client-configs/base.conf

```
; script-security 2
; up /etc/openvpn/update-resolv-conf
; down /etc/openvpn/update-resolv-conf
```

Now add another set of lines for clients that use `systemd-resolved` for DNS resolution:

~/client-configs/base.conf

```
; script-security 2
; up /etc/openvpn/update-systemd-resolved
; down /etc/openvpn/update-systemd-resolved
; down-pre
; dhcp-option DOMAIN-ROUTE .
```

Save and close the file when you are finished.

Later in Step 13 - Installing the Client Configuration step of this tutorial you will learn how to determine how DNS resolution works on Linux clients and which section to uncomment.

Next, we'll create a script that will compile your base configuration with the relevant certificate, key, and encryption files and then place the generated configuration in the `~/client-configs/files` directory. Open a new file called `make_config.sh` within the `~/client-configs` directory:

```
$ nano ~/client-configs/make_config.sh
```
Copy

Inside, add the following content:

~/client-configs/make_config.sh

```
#!/bin/bash

# First argument: Client identifier

KEY_DIR=~/client-configs/keys
```
Copy

```
OUTPUT_DIR=~/client-configs/files
BASE_CONFIG=~/client-configs/base.conf

cat ${BASE_CONFIG} \
    <(echo -e '<ca>') \
    ${KEY_DIR}/ca.crt \
    <(echo -e '</ca>\n<cert>') \
    ${KEY_DIR}/${1}.crt \
    <(echo -e '</cert>\n<key>') \
    ${KEY_DIR}/${1}.key \
    <(echo -e '</key>\n<tls-crypt>') \
    ${KEY_DIR}/ta.key \
    <(echo -e '</tls-crypt>') \
    > ${OUTPUT_DIR}/${1}.ovpn
```

Save and close the file when you are finished.

Before moving on, be sure to mark this file as executable by typing:

```
$ chmod 700 ~/client-configs/make_config.sh
```

Copy

This script will make a copy of the `base.conf` file you made, collect all the certificate and key files you've created for your client, extract their contents, append them to the copy of the base configuration file, and export all of this content into a new client configuration file. This means that, rather than having to manage the client's configuration, certificate, and key files separately, all the required information is stored in one place. The benefit of using this method is that if you ever need to add a client in the future, you can run this script to quickly create a new config file and ensure that all the important information is stored in a single, easy-to-access location.

Please note that any time you add a new client, you will need to generate new keys and certificates for it before you can run this script and generate its configuration file. You will get some practice using this script in the next step.

# Step 12 – Generating Client Configurations

If you followed along with the guide, you created a client certificate and key named `client1.crt` and `client1.key`, respectively, in Step 6. You can generate a config file for these credentials by moving into your `~/client-configs` directory and running the script you made at the end of the previous step:

```
$ cd ~/client-configs
$ ./make_config.sh client1
```

Copy

This will create a file named `client1.ovpn` in your `~/client-configs/files` directory:

```
$ ls ~/client-configs/files
```

Copy

```
Output
client1.ovpn
```

You need to transfer this file to the device you plan to use as the client. For instance, this could be your local computer or a mobile device.

While the exact applications used to accomplish this transfer will depend on your device's operating system and your personal preferences, a dependable and secure method is to use SFTP (SSH file transfer protocol) or SCP (Secure Copy) on the backend. This will transport your client's VPN authentication files over an encrypted connection.

Here is an example SFTP command which you can run from your local computer (macOS or Linux). This will copy the `client1.ovpn` file we've created in the last step to your home directory:

```
local$ sftp sammy@openvpn_server_ip:client-configs/files/client1.ovpn ~/
```
Copy

Here are several tools and tutorials for securely transferring files from the OpenVPN server to a local computer:

- WinSCP
- How To Use SFTP to Securely Transfer Files with a Remote Server
- How To Use Filezilla to Transfer and Manage Files Securely on your VPS

# Step 13 – Installing the Client Configuration

This section covers how to install a client VPN profile on Windows, macOS, Linux, iOS, and Android. None of these client instructions are dependent on one another, so feel free to skip to whichever is applicable to your device.

The OpenVPN connection will have the same name as whatever you called the `.ovpn` file. In regards to this tutorial, this means that the connection is named `client1.ovpn`, aligning with the first client file you generated.

# Windows

Installing

Download the OpenVPN client application for Windows from OpenVPN's Downloads page. Choose the appropriate installer version for your version of Windows.

Note: OpenVPN needs administrative privileges to install.

After installing OpenVPN, copy the `.ovpn` file to:

```
C:\Program Files\OpenVPN\config
```

When you launch OpenVPN, it will automatically locate the profile and make it available.

You must run OpenVPN as an administrator each time it's used, even by administrative accounts. To do this without having to right-click and select Run as administrator every time you use the VPN, you must preset this from an administrative account. This also means that standard users will need to enter the administrator's password to use OpenVPN. On the other hand, standard users can't properly connect to the server unless the OpenVPN application on the client has admin rights, so the elevated privileges are necessary.

To set the OpenVPN application to always run as an administrator, right-click on its shortcut icon and go to Properties . At the bottom of the Compatibility tab, click the button to Change settings for all users . In the new window, check Run this program as an administrator.

Connecting

Each time you launch the OpenVPN GUI, Windows will ask if you want to allow the program to make changes to your computer. Click Yes. Launching the OpenVPN client application only puts the applet in the system tray so that you can connect and disconnect the VPN as needed; it does not actually make the VPN connection.

Once OpenVPN is started, initiate a connection by going into the system tray applet and right-clicking on the OpenVPN applet icon. This opens the context menu. Select client1 at the top of the menu (that's your `client1.ovpn` profile) and choose Connect.

A status window will open showing the log output while the connection is established, and a message will show once the client is connected.

Disconnect from the VPN the same way: Go into the system tray applet, right-click the OpenVPN applet icon, select the client profile and click Disconnect.

### macOS

Installing

Tunnelblick is a free, open source OpenVPN client for macOS. You can download the latest disk image from the Tunnelblick Downloads page. Double-click the downloaded `.dmg` file and follow the prompts to install.

Towards the end of the installation process, Tunnelblick will ask if you have any configuration files. Answer I have configuration files and let Tunnelblick finish. Open a Finder window and double-click `client1.ovpn`. Tunnelblick will install the client profile. Administrative privileges are required.

Connecting

Launch Tunnelblick by double-clicking the Tunnelblick icon in the Applications folder. Once Tunnelblick has been launched, there will be a Tunnelblick icon in the menu bar at the top right of the screen for controlling connections. Click on the icon, and then the Connect client1 menu item to initiate the VPN connection. If you are using custom DNS settings with Tunnelblick, you may need check “Allow changes to manually-set network settings” in the advanced configuration dialog.

### Linux

Installing

If you are using Linux, there are a variety of tools that you can use depending on your distribution. Your desktop environment or window manager might also include connection utilities.

The most universal way of connecting, however, is to just use the OpenVPN software.

On Ubuntu or Debian, you can install it just as you did on the server by typing:

```
client$ sudo apt update
client$ sudo apt install openvpn
```
Copy

On CentOS you can enable the EPEL repositories and then install it by typing:

```
client$ sudo dnf install epel-release
client$ sudo dnf install openvpn
```
Copy

Configuring Clients that use **`systemd-resolved`**

First determine if your system is using `systemd-resolved` to handle DNS resolution by checking the `/etc/resolv.conf` file:

```
client1$ cat /etc/resolv.conf
```
Copy

```
Output
# This file is managed by man:systemd-resolved(8). Do not edit.
. . .

nameserver 127.0.0.53
options edns0
```

If your system is configured to use `systemd-resolved` for DNS resolution, the IP address after the `nameserver` option will be `127.0.0.53`. There should also be comments in the file like the output that is shown that explain how `systemd-resolved` is managing the file. If you have a different IP address than `127.0.0.53` then chances are your system is not using `systemd-resolved` and you can go to the next section on configuring Linux clients that have an `update-resolv-conf` script instead.

To support these clients, first install the `openvpn-systemd-resolved` package. It provides scripts that will force `systemd-resolved` to use the VPN server for DNS resolution.

```
client$ sudo apt install openvpn-systemd-resolved
```
Copy

One that package is installed, configure the client to use it, and to send all DNS queries over the VPN interface. Open the client's VPN file:

```
client$ nano client1.ovpn
```
Copy

Now uncomment the following lines that you added earlier:

client1.ovpn

```
script-security 2
up /etc/openvpn/update-systemd-resolved
down /etc/openvpn/update-systemd-resolved
down-pre
dhcp-option DOMAIN-ROUTE .
```

Configuring Clients that use **`update-resolv-conf`**

If your system is not using `systemd-resolved` to manage DNS, check to see if your distribution includes an `/etc/openvpn/update-resolv-conf` script instead:

```
client1$ ls /etc/openvpn
```
Copy

```
Output
update-resolv-conf
```

If your client includes the `update-resolv-conf` file, then edit the OpenVPN client configuration file that you transferred earlier:

```
client$ nano client1.ovpn
```
Copy

Uncomment the three lines you added to adjust the DNS settings:

client1.ovpn

```
script-security 2
up /etc/openvpn/update-resolv-conf
down /etc/openvpn/update-resolv-conf
```

If you are using CentOS, change the `group` directive from `nogroup` to `nobody` to match the distribution's available groups:

client1.ovpn

```
group nobody
```

Save and close the file.

Connecting

Now, you can connect to the VPN by just pointing the `openvpn` command to the client configuration file:

```
client$ sudo openvpn --config client1.ovpn
```
Copy

This should connect you to your VPN.

Note: If your client uses `systemd-resolved` to manage DNS, check the settings are applied correctly by running the `systemd-resolve --status` command like this:

```
client$ systemd-resolve --status tun0
```

Copy

You should see output like the following:

```
Output
Link 22 (tun0)
. . .
         DNS Servers: 208.67.222.222
                      208.67.220.220
          DNS Domain: ~.
```

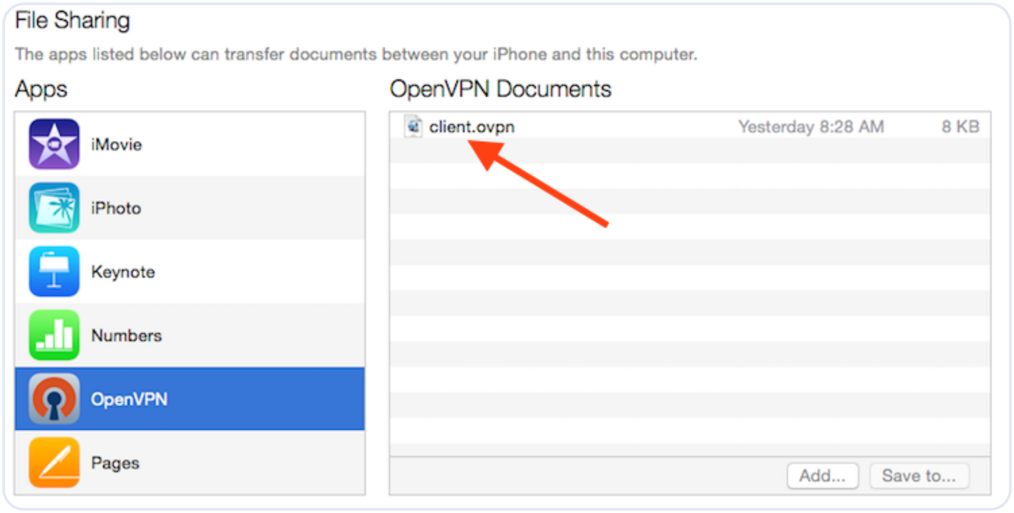
If you see the IP addresses of the DNS servers that you configured on the OpenVPN server, along with the `~.` setting for *DNS Domain* in the output, then you have correctly configured your client to use the VPN server's DNS resolver. You can also check that you are sending DNS queries over the VPN by using a site like DNS leak test.com.

# iOS

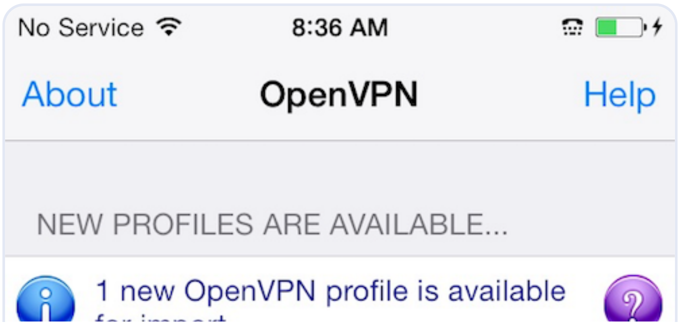
Installing

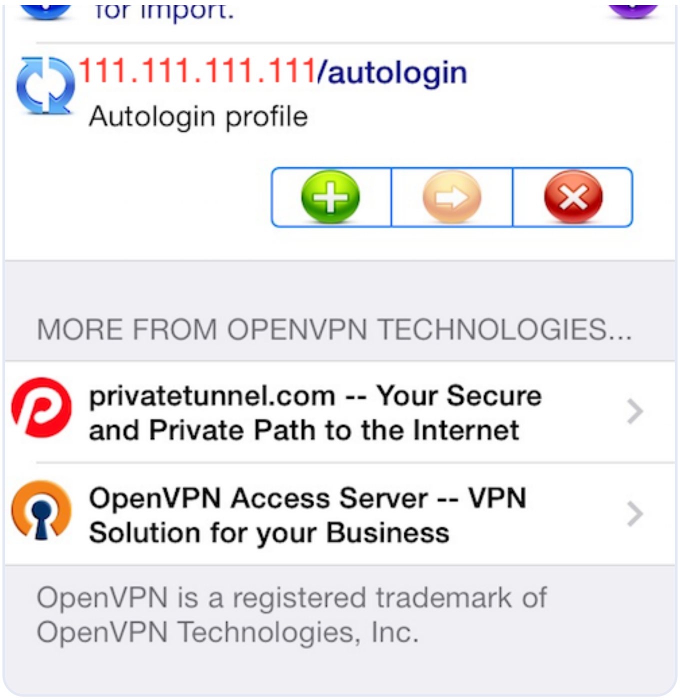
From the iTunes App Store, search for and install OpenVPN Connect, the official iOS OpenVPN client application. To transfer your iOS client configuration onto the device, connect it directly to a computer.

The process of completing the transfer with iTunes is outlined here. Open iTunes on the computer and click on iPhone > apps. Scroll down to the bottom to the File Sharing section and click the OpenVPN app. The blank window to the right, OpenVPN Documents, is for sharing files. Drag the `.ovpn` file to the OpenVPN Documents window.



Now launch the OpenVPN app on the iPhone. You will receive a notification that a new profile is ready to import. Tap the green plus sign to import it.

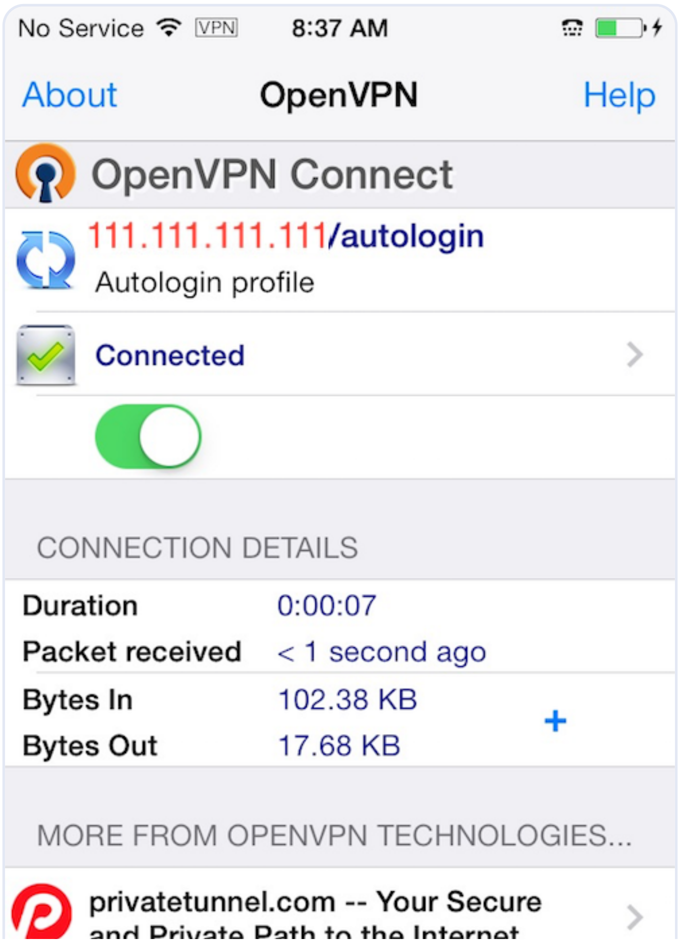

No Service
8:36 AM
About
OpenVPN
Help
NEW PROFILES ARE AVAILABLE...
1 new OpenVPN profile is available

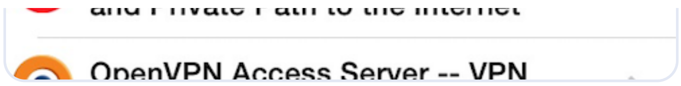

for import.
111.111.111.111/autologin
Autologin profile
MORE FROM OPENVPN TECHNOLOGIES...
privatetunnel.com -- Your Secure and Private Path to the Internet
OpenVPN Access Server -- VPN Solution for your Business
OpenVPN is a registered trademark of OpenVPN Technologies, Inc.

Connecting

OpenVPN is now ready to use with the new profile. Start the connection by sliding the Connect button to the On position. Disconnect by sliding the same button to Off.

Note: The VPN switch under Settings cannot be used to connect to the VPN. If you try, you will receive a notice to only connect using the OpenVPN app.


No Service VPN 8:37 AM
About
OpenVPN
Help
OpenVPN Connect
111.111.111.111/autologin
Autologin profile
Connected
CONNECTION DETAILS
Duration 0:00:07
Packet received < 1 second ago
Bytes In 102.38 KB
Bytes Out 17.68 KB
MORE FROM OPENVPN TECHNOLOGIES...
privatetunnel.com -- Your Secure and Private Path to the Internet



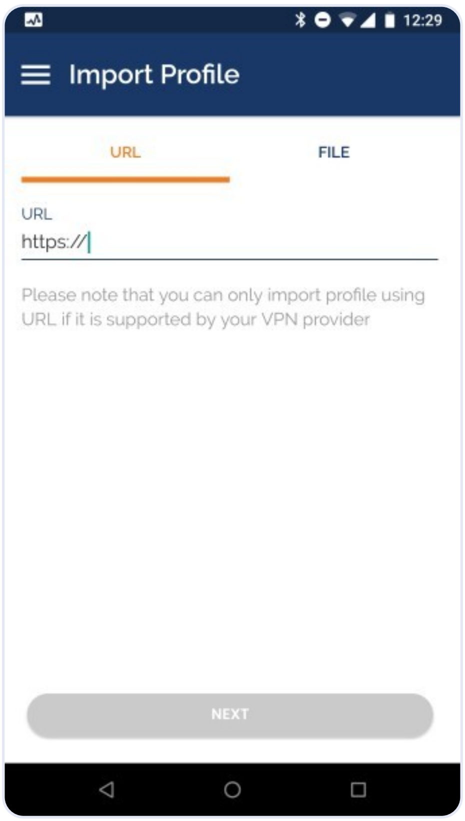
# Android

Installing

Open the Google Play Store. Search for and install Android OpenVPN Connect, the official Android OpenVPN client application.

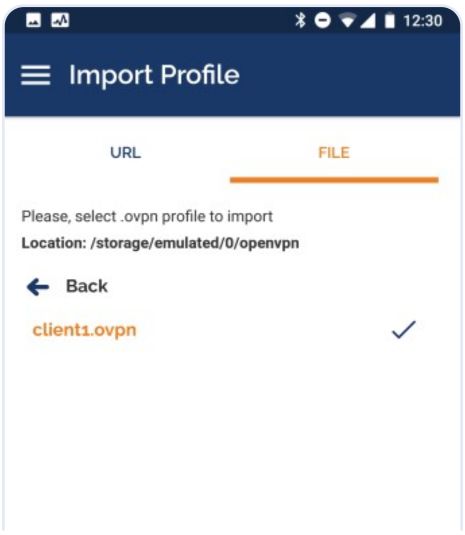
You can transfer the `.ovpn` profile by connecting the Android device to your computer by USB and copying the file over. Alternatively, if you have an SD card reader, you can remove the device's SD card, copy the profile onto it and then insert the card back into the Android device.

Start the OpenVPN app and tap the `FILE` menu to import the profile.

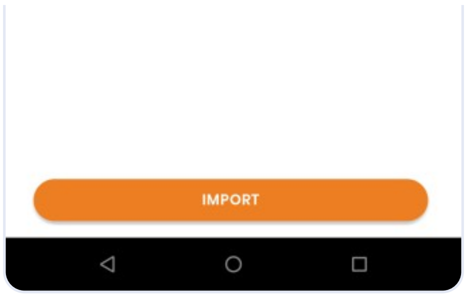


Please note that you can only import profile using URL if it is supported by your VPN provider

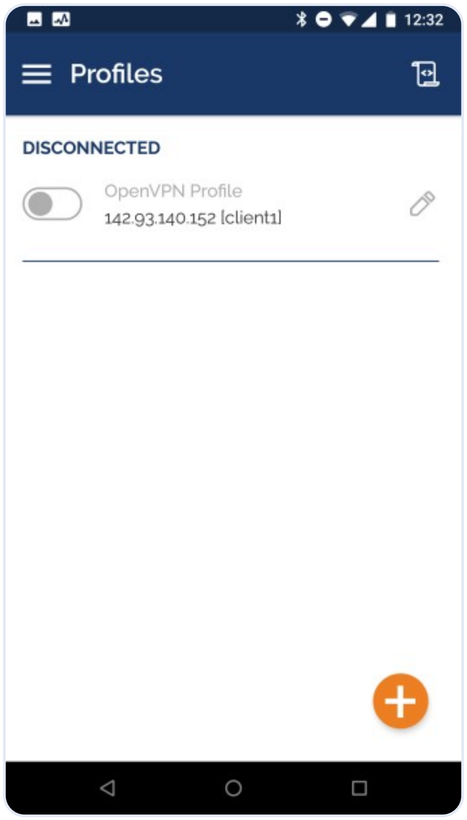
Then navigate to the location of the saved profile (the screenshot uses `/storage/emulated/0/openvpn`) and select your `.ovpn` file. Tap the `IMPORT` button to finish importing this profile.



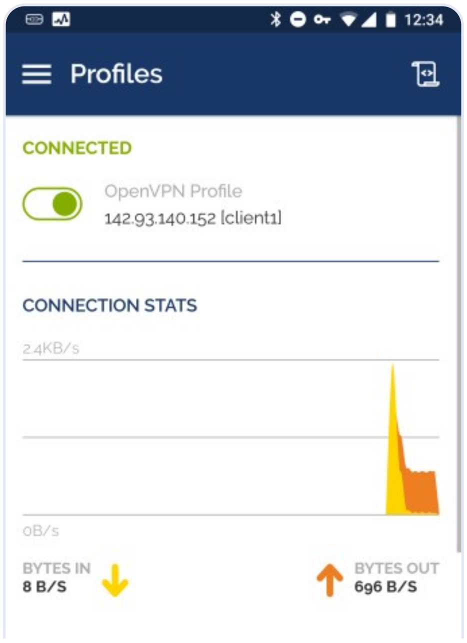
Please, select .ovpn profile to import
Location: /storage/emulated/0/openvpn

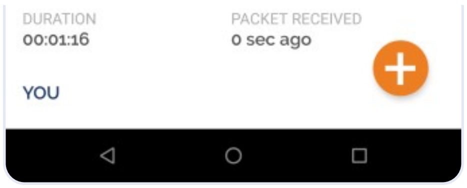


Connecting Once the profile is added, you will see a screen like this:



To connect, tap the toggle button close to the profile you want to use. You'll see real time stats of your connection and traffic being routed through your OpenVPN server:





To disconnect, just tap the toggle button on the top left once again. You will be prompted to confirm that you want to disconnect from your VPN.

# Step 14 – Testing Your VPN Connection (Optional)

Note: This method for testing your VPN connection will only work if you opted to route all your traffic through the VPN in Step 7 when you edited the `server.conf` file for OpenVPN.

Once everything is installed, a simple check confirms everything is working properly. Without having a VPN connection enabled, open a browser and go to DNSLeakTest.

The site will return the IP address assigned by your internet service provider and as you appear to the rest of the world. To check your DNS settings through the same website, click on Extended Test and it will tell you which DNS servers you are using.

Now connect the OpenVPN client to your Droplet's VPN and refresh the browser. A completely different IP address (that of your VPN server) should now appear, and this is how you appear to the world. Again, DNSLeakTest's Extended Test will check your DNS settings and confirm you are now using the DNS resolvers pushed by your VPN.

# Step 15 – Revoking Client Certificates

Occasionally, you may need to revoke a client certificate to prevent further access to the OpenVPN server.

To do so, follow the example in the prerequisite tutorial on How to Set Up and Configure a Certificate Authority on Ubuntu 20.04 under the *Revoking a Certificate* section.

Once you have revoked a certificate for a client using those instructions, you'll need to copy the generated `crl.pem` file to your OpenVPN server in the `/etc/openvpn/server` directory:

```
$ sudo cp /tmp/crl.pem /etc/openvpn/server/
```
Copy

Next, open the OpenVPN server configuration file:

```
$ sudo nano /etc/openvpn/server/server.conf
```
Copy

At the bottom of the file, add the `crl-verify` option, which will instruct the OpenVPN server to check the certificate revocation list that we've created each time a connection attempt is made:

/etc/openvpn/server/server.conf

```
crl-verify crl.pem
```

Save and close the file.

Finally, restart OpenVPN to implement the certificate revocation:

```
$ sudo systemctl restart openvpn-server@server.service
```
Copy

The client should no longer be able to successfully connect to the server using the old credential.

To revoke additional clients, follow this process:

1. Revoke the certificate with the `./easyrsa revoke client_name` command
2. Generate a new CRL
3. Transfer the new `crl.pem` file to your OpenVPN server and copy it to the `/etc/openvpn/server/` directory to overwrite the old list.
4. Restart the OpenVPN service.

You can use this process to revoke any certificates that you've previously issued for your server.

# Conclusion

You should now have a fully operational virtual private network running on your OpenVPN Server. You can browse the web and download content without worrying about malicious actors tracking your activity.

There are several steps you could take to customize your OpenVPN installation even further, such as configuring your client to connect to the VPN automatically or configuring client-specific rules and access policies. For these and other OpenVPN customizations, you should consult the official OpenVPN documentation.

To configure more clients, you only need to follow steps 6 and 11-13 for each additional device. To revoke access to clients, follow step 15.

About the authors

Jamon Camisso Author

Still looking for an answer? Ask a question Search for more help

Was this helpful? Yes No

Comments Follow-Up Questions

10 Comments

B I U S H1 H2 H3

Leave a comment...

This textbox defaults to using `Markdown` to format your answer.

You can type `!ref` in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

alihkousha • October 23, 2022

hi I get this output on step 10 trying activating OPENVPN:

```
● openvpn-server@server.service - OpenVPN service for server
     Loaded: loaded (/lib/systemd/system/openvpn-server@.service; enabled; vendor preset: enal
     Active: activating (auto-restart) (Result: exit-code) since Sun 2022-10-23 10:17:38 UTC;
       Docs: man:openvpn(8)
             https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
             https://community.openvpn.net/openvpn/wiki/HOWTO
    Process: 25450 ExecStart=/usr/sbin/openvpn --status /run/openvpn-server/status-server.log
   Main PID: 25450 (code=exited, status=1/FAILURE)
     Status: "Pre-connection initialization successful"
        CPU: 25ms
```

I changed protocol and port but i set them to defeault values and add them on firewall

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
1194/udp                   ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
1194/udp (v6)              ALLOW       Anywhere (v6)
```

Reply

peter53 • September 17, 2022

Created a cloud-agnostic Ansible playbook to automate OpenVPN server setup based on this tutorial: https://github.com/ptr-dorjin/ansible-vpn-server

Used the tutorial several times manually, but then decided to automate it. Should work in Digital Ocean, as well :)

@jamonation, thanks again for the tutorial. Any plans on adding 22.04 version?

Reply

Paul Gonzalez • September 8, 2022

Hey, just some friendly help for others: my server wouldn't start in step 10. Checked /var/log /syslog and found the error "openvpn[2255]: failed to find GID for group nobody". I used the command

```
compgen -g
```

to list my Ubuntu 22.04 groups, and found that I didn't have nobody, but I did have nogroup. I changed /etc/openvpn/server/server.conf from group nobody to group nogroup and the server started right up.

Show replies Reply

GiantAquaCrab • May 28, 2022

Thanks a lot for the detailed tutorial! BTW, it still works in Ubuntu 22.04.

Reply

GiantAquaCrab • May 28, 2022

Thanks a lot for the detailed tutorial! BTW, it still works in Ubuntu 22.04.

Reply

Hanpulat • March 11, 2022

Guys, pls help, tried 3 times and always the same error:

```
sammy@ubuntu-s-1vcpu-1gb-amd-fra1-01:~/easy-rsa$ ./easyrsa sign-req server server
./easyrsa: 1: ./vars: /home/sammy/easy-rsa/vars: Permission denied

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020


You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 1080 days:

subject=
    commonName                = \0D


Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes

Aborting without confirmation.
sammy@ubuntu-s-1vcpu-1gb-amd-fra1-01:~/easy-rsa$ ./easyrsa sign-req server server
./easyrsa: 1: ./vars: /home/sammy/easy-rsa/vars: Permission denied

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020


You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 1080 days:

subject=
    commonName                = \0D


Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes
Using configuration from /home/sammy/easy-rsa/pki/safessl-easyrsa.cnf
Enter pass phrase for /home/sammy/easy-rsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'^M'
ERROR: adding extensions in section default
140715882181952:error:2206D06D:X509 V3 routines:X509V3_parse_list:invalid null value:../crypt
140715882181952:error:22097069:X509 V3 routines:do_ext_nconf:invalid extension string:../cryp
140715882181952:error:22098080:X509 V3 routines:X509V3_EXT_nconf:error in extension:../crypto,

Easy-RSA error:

signing failed (openssl output above may have more detail)
```

Reply

kirkofthefleet • February 28, 2022

Hello,

I have tried this tutorial four times and keep getting stuck trying to start the OpenVPN-Server service. The logs show the following:

openvpn[9822]: ERROR: Cannot open TUN/TAP dev /dev/net/tun: No such file or directory (errno=2) openvpn[9822]: Exiting due to fatal error systemd[1]: openvpn-server@server.service: Main process exited, code=exited, status=1/FAILURE systemd[1]: openvpn-server@server.service: Failed with result 'exit-code'.

I can't figure out what isn't working. Does anyone have any advice?

Show replies   Reply

cliffporter84 • February 19, 2022

I'm running OpenVPN on Windows. I'm gettin this error:

Sat Feb 19 22:41:55 2022 TLS Error: TLS key negotiation failed to occur within 60 seconds (check your network connectivity) Sat Feb 19 22:41:55 2022 TLS Error: TLS handshake failed

My /var/log/syslog looks something like this

Feb 20 03:42:00 testVPN openvpn[726]: tls-crypt unwrap error: packet authentication failed Feb 20 03:42:00 testVPN openvpn[726]: TLS Error: tls-crypt unwrapping failed from [AF_INET]164.153.58.194:44479 Feb 20 03:42:02 testVPN openvpn[726]: tls-crypt unwrap error: packet authentication failed Feb 20 03:42:02 testVPN openvpn[726]: TLS Error: tls-crypt unwrapping failed from [AF_INET]164.153.58.194:44479 Feb 20 03:42:06 testVPN openvpn[726]: tls-crypt unwrap error: packet authentication failed Feb 20 03:42:06 testVPN openvpn[726]: TLS Error: tls-crypt unwrapping failed from [AF_INET]164.153.58.194:44479 Feb 20 03:42:09 testVPN kernel: [8531.640236] [UFW BLOCK] IN=eth0 OUT= MAC=b2:4e:67:db:ed:40:fe:00:00:00:01:01:08:00 SRC=167.94.146.19 DST=161.35.58.34 LEN=44 TOS=0x00 PREC=0x20 TTL=39 ID=49971 PROTO=TCP SPT=6151 DPT=39804 WINDOW=1024 RES=0x00 SYN URGP=0 Feb 20 03:42:15 testVPN openvpn[726]: tls-crypt unwrap error: packet authentication failed Feb 20 03:42:15 testVPN openvpn[726]: TLS Error: tls-crypt unwrapping failed from [AF_INET]164.153.58.194:44479 Feb 20 03:42:31 testVPN openvpn[726]: tls-crypt unwrap error: packet authentication failed Feb 20 03:42:31 testVPN openvpn[726]: TLS Error: tls-crypt unwrapping failed from [AF_INET]164.153.58.194:44479 Feb 20 03:42:34 testVPN kernel: [8556.711951] [UFW BLOCK] IN=eth0 OUT= MAC=b2:4e:67:db:ed:40:fe:00:00:00:01:01:08:00 SRC=79.124.62.130 DST=161.35.58.34 LEN=40 TOS=0×00 PREC=0×00 TTL=245 ID=62601 PROTO=TCP SPT=49848 DPT=53777 WINDOW=1024 RES=0×00 SYN URGP=0 Feb 20 03:42:40 testVPN kernel: [8562.832978] [UFW BLOCK] IN=eth0 OUT= MAC=b2:4e:67:db:ed:40:fe:00:00:00:01:01:08:00 SRC=198.251.80.182 DST=161.35.58.34 LEN=40 TOS=0×00 PREC=0×00 TTL=241 ID=9060 PROTO=TCP SPT=6697 DPT=80 WINDOW=8192 RES=0×00 SYN URGP=0 Feb 20 03:42:47 testVPN kernel: [8569.737093] [UFW BLOCK] IN=eth0 OUT= MAC=b2:4e:67:db:ed:40:fe:00:00:00:01:01:08:00 SRC=183.136.225.42 DST=161.35.58.34 LEN=44 TOS=0×00 PREC=0×00 TTL=106 ID=24601 PROTO=TCP SPT=13239 DPT=8125 WINDOW=29200 RES=0×00 SYN URGP=0 Feb 20 03:43:06 testVPN openvpn[726]: tls-crypt unwrap error: packet authentication failed Feb 20 03:43:06 testVPN openvpn[726]: TLS Error: tls-crypt unwrapping failed from [AF_INET]164.153.58.194:44869 Feb 20 03:43:07 testVPN openvpn[726]: tls-crypt unwrap error: packet authentication failed Feb 20 03:43:07 testVPN openvpn[726]: TLS Error: tls-crypt unwrapping failed from [AF_INET]164.153.58.194:44869 Feb 20 03:43:11 testVPN openvpn[726]: tls-crypt unwrap error: packet authentication failed Feb 20 03:43:11 testVPN openvpn[726]: TLS Error: tls-crypt unwrapping failed from [AF_INET]164.153.58.194:44869 Feb 20 03:43:11 testVPN kernel: [8594.507807] [UFW BLOCK] IN=eth0 OUT= MAC=b2:4e:67:db:ed:40:fe:00:00:00:01:01:08:00 SRC=198.251.80.182 DST=161.35.58.34 LEN=40 TOS=0×00 PREC=0×00 TTL=249 ID=9060 PROTO=TCP SPT=6697 DPT=80 WINDOW=8192 RES=0×00 SYN URGP=0

when I run ufw status, I see the following:

Status: active

To Action From

1194/udp ALLOW Anywhere OpenSSH ALLOW Anywhere 1194/tcp ALLOW Anywhere 1194/udp (v6) ALLOW Anywhere (v6) OpenSSH (v6) ALLOW Anywhere (v6) 1194/tcp (v6) ALLOW Anywhere (v6)

I followed the tutorial as closely as I could. I have no idea what is going on. I have tried connecting through Windows and Android and I get the same sort of timeout errors.

Show replies Reply

Flávio Tomazio • June 22, 2021

Is there any way to password protect the .ovpn file?

Ex.: In an environment that this VPN is used to access a service/server/ssh restricted to the VPN, but for some reason another user had to physically/remotely access your computer. With a password that must be entered at all times, this user can be prevented from connecting to the VPN and accessing those sensitive services that require a connection via VPN.

Reply

BigTealShark • April 26, 2021

I see this tutorial has been updated from the original 16.04 and 18.04 versions. Would you also mind perhaps updating the screenshots of the OpenVPN application running on iOS to something more recent about than 10 years ago? iTunes doesn't even exist anymore! 😆

Reply

Load More Comments

Web hosting without headaches

Try Cloudways, the #1 managed hosting provider for agencies & developers, with $100 in free credit Learn more →

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

MySQL

Docker

Kubernetes

All tutorials →

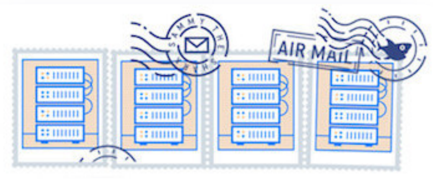
Free Managed Hosting →

Congratulations on unlocking the whale ambience easter egg! Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.

Thank you to the Glacier Bay National Park & Preserve and Merrick079 for the sounds behind this easter egg.

Interested in whales, protecting them, and their connection to helping prevent climate change? We recommend checking out the Whale and Dolphin Conservation.

Reset easter egg to be discovered again / Permanently dismiss and hide easter egg



Get our biweekly newsletter

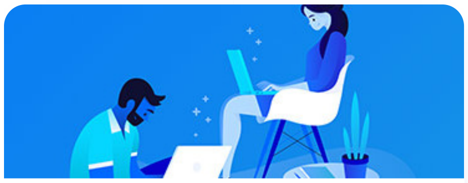
Sign up for Infrastructure as a Newsletter.

Sign up →

Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

Learn more →



Become a contributor

You get paid; we donate to tech nonprofits.

Learn more →

Featured on Community

Kubernetes Course Learn Python 3 Machine Learning in Python Getting started with Go Intro to Kubernetes

DigitalOcean Products

Cloudways Virtual Machines Managed Databases Managed Kubernetes Block Storage Object Storage Marketplace VPC Load Balancers

Search by Droplet name or IP (Cmd+B)
Create
USAGE $0.00
PROJECTS
jonsmith
New Project
MANAGE
Droplets
Kubernetes
Volumes
jonsmith
DEFAULT
Update your project information under Settings
Move Resources
Resources
Activity
Settings
SPACES (1)
jon-db
https://jon-db.nyc3.digitaloceanspaces.com