How To Set Up an OpenVPN Server on Debian 10

Posted July 19, 2019 34k SECURITY DEBIAN VPN DEBIAN 10

By Justin Ellingwood and Mark Drake
Become an author

Not using **Debian 10**? Choose a different version:

- CentOS 7 ❯
- Debian 9 ❯
- Debian 8 ❯

A previous version of this tutorial was written by Justin Ellingwood

Introduction

Want to access the Internet safely and securely from your smartphone or laptop when connected to an untrusted network such as the WiFi of a hotel or coffee shop? A Virtual Private Network (VPN) allows you to traverse untrusted networks privately and securely as if you were on a private network. The traffic emerges from the VPN server and continues its journey to the destination.

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 the untrusted network.

OpenVPN is a full-featured, open-source Secure Socket Layer (SSL) VPN solution that accommodates a wide range of configurations. In this tutorial, you will set up an OpenVPN server on a Debian 10 server and then configure access to it from Windows, macOS, iOS and/or Android. This tutorial will keep the installation and configuration steps as simple as possible for each of these setups.

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 complete this tutorial, you will need access to a Debian 10 server to host your OpenVPN service. You will need to configure a non-**root** user with `sudo` privileges before you start this guide. You can follow our Debian 10 initial server setup guide to set up a user with appropriate permissions. The linked tutorial will also set up a **firewall**, which is assumed to be in place throughout this guide.

Additionally, you will need a separate machine to serve as your certificate authority (CA). While it's 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 Debian 10 server that also has a non-**root** user with `sudo` privileges and a basic firewall.

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 Debian 10 for instructions on how to perform either of these solutions.

When you have these prerequisites in place, you can move on to Step 1 of this tutorial.

Step 1 — Installing OpenVPN and EasyRSA

To start off, update your **VPN server's** package index and install OpenVPN. OpenVPN is available in Debian's default repositories, so you can use `apt` for the installation:

```
$ sudo apt update
$ sudo apt install openvpn
```

OpenVPN is a TLS/SSL VPN. This means that it utilizes certificates in order to encrypt traffic between the server and clients. To issue trusted certificates, you will set up your own simple certificate authority (CA). To do this, we will download the latest version of EasyRSA, which we will use to build our CA public key infrastructure (PKI), from the project's official GitHub repository.

As mentioned in the prerequisites, we will build the CA on a standalone server. The reason for this approach is that, if an

attacker were able to infiltrate your server, they would be able to access your CA private key and use it to sign new certificates, giving them access to your VPN. Accordingly, managing the CA from a standalone machine helps to prevent unauthorized users from accessing your VPN. Note, as well, that it's recommended that you keep the CA server turned off when not being used to sign keys as a further precautionary measure.

To begin building the CA and PKI infrastructure, use `wget` to download the latest version of EasyRSA on **both your CA machine and your OpenVPN server**. To get the latest version, go to the **Releases** page on the official EasyRSA GitHub project, copy the download link for the file ending in `.tgz`, and then paste it into the following command:

```
$ wget -P ~/ https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.6/EasyRSA-unix-v3.0.6.tgz
```

Then extract the tarball:

```
$ cd ~
$ tar xvf EasyRSA-unix-v3.0.6.tgz
```

You have successfully installed all the required software on your server and CA machine. Continue on to configure the variables used by EasyRSA and to set up a CA directory, from which you will generate the keys and certificates needed for your server and clients to access the VPN.

Step 2 — Configuring the EasyRSA Variables and Building the CA

EasyRSA comes installed with a configuration file which you can edit to define a number of variables for your CA.

On your **CA machine**, navigate to the EasyRSA directory:

```
$ cd ~/EasyRSA-v3.0.6/
```

Inside this directory is a file named `vars.example`. Make a copy of this file, and name the copy `vars` without a file extension:

```
$ cp vars.example vars
```

Open this new file using your preferred text editor:

```
$ nano vars
```

Find the settings that set field defaults for new certificates. It will look something like this:

~/EasyRSA-v3.0.6/vars

```
. . .

#set_var EASYRSA_REQ_COUNTRY    "US"
#set_var EASYRSA_REQ_PROVINCE   "California"
#set_var EASYRSA_REQ_CITY       "San Francisco"
#set_var EASYRSA_REQ_ORG        "Copyleft Certificate Co"
#set_var EASYRSA_REQ_EMAIL      "me@example.net"
#set_var EASYRSA_REQ_OU         "My Organizational Unit"

. . .
```

Uncomment these lines and update the highlighted values to whatever you'd prefer, but do not leave them blank:

~/EasyRSA-v3.0.6/vars

```
. . .

set_var EASYRSA_REQ_COUNTRY    "US"
set_var EASYRSA_REQ_PROVINCE   "NewYork"
set_var EASYRSA_REQ_CITY       "New York City"
set_var EASYRSA_REQ_ORG        "DigitalOcean"
set_var EASYRSA_REQ_EMAIL      "admin@example.com"
set_var EASYRSA_REQ_OU         "Community"

. . .
```

When you are finished, save and close the file.

Within the EasyRSA directory is a script called `easyrsa` which is called to perform a variety of tasks involved with building and managing the CA. Run this script with the `init-pki` option to initiate the public key infrastructure on the CA server:

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

```
Output
. . .
init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /home/sammy/EasyRSA-v3.0.6/pki
```

After this, call the `easyrsa` script again, following it with the `build-ca` option. This will build the CA and create two important files — `ca.crt` and `ca.key` — which make up the public and private sides of an SSL certificate.

- `ca.crt` is the CA's public certificate file which, in the context of OpenVPN, the server and the client use to inform one another that they are part of the same web of trust and not someone performing a man-in-the-middle attack. For this reason, your server and all of your clients will need a copy of the `ca.crt` file.
- `ca.key` is the private key which the CA machine uses to sign keys and certificates for servers and clients. If an attacker gains access to your CA and, in turn, your `ca.key` file, they will be able to sign certificate requests and gain access to your VPN, impeding its security. This is why your `ca.key` file should **only** be on your CA machine and that, ideally, your CA machine should be kept offline when not signing certificate requests as an extra security measure.

If you don't want to be prompted for a password every time you interact with your CA, you can run the `build-ca` command with the `nopass` option, like this:

```
$ ./easyrsa build-ca nopass
```

In the output, you'll be asked to confirm the common name for your CA:

```
Output
. . .
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:
```

The common name is the name used to refer to this machine in the context of the certificate authority. You can enter any string of characters for the CA's common name but, for simplicity's sake, press `ENTER` to accept the default name.

With that, your CA is in place and it's ready to start signing certificate requests.

Step 3 — Creating the Server Certificate, Key, and Encryption Files

Now that you have a CA ready to go, you can generate a private key and certificate request from your server and then transfer the request over to your CA to be signed, creating the required certificate. You're also free to create some additional files used during the encryption process.

Start by navigating to the EasyRSA directory on your **OpenVPN server**:

```
$ cd EasyRSA-v3.0.6/
```

From there, run the `easyrsa` script with the `init-pki` option. Although you already ran this command on the CA machine, it's necessary to run it here because your server and CA will have separate PKI directories:

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

Then call the `easyrsa` script again, this time with the `gen-req` option followed by a common name for the machine. Again, this could be anything you like but it can be helpful to make it something descriptive. Throughout this tutorial, the OpenVPN server's common name will simply 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
```

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/` directory:

```
$ sudo cp ~/EasyRSA-v3.0.6/pki/private/server.key /etc/openvpn/
```

Using a secure method (like SCP, in our example below), transfer the `server.req` file to your CA machine:

```
$ scp ~/EasyRSA-v3.0.6/pki/reqs/server.req sammy@your_CA_ip:/tmp
```

Next, on **your CA machine**, navigate to the EasyRSA directory:

```
$ cd EasyRSA-v3.0.6/
```

Using the `easyrsa` script again, import the `server.req` file, following the file path with its common name:

```
$ ./easyrsa import-req /tmp/server.req server
```

Then 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`, so for the OpenVPN server's certificate request, be sure to use the `server` request type:

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

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

```
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                = server


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

If you encrypted your CA key, you'll be prompted for your password at this point.

Next, transfer the signed certificate back to your VPN server using a secure method:

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

Before logging out of your CA machine, transfer the `ca.crt` file to your server as well:

```
$ scp pki/ca.crt sammy@your_server_ip:/tmp
```

Next, log back into your OpenVPN server and copy the `server.crt` and `ca.crt` files into your `/etc/openvpn/` directory:

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

Then navigate to your EasyRSA directory:

```
$ cd EasyRSA-v3.0.6/
```

From there, create a strong Diffie-Hellman key to use during key exchange by typing:

```
$ ./easyrsa gen-dh
```

This may take a few minutes to complete. Once it does, generate an HMAC signature to strengthen the server's TLS integrity verification capabilities:

```
$ sudo openvpn --genkey --secret ta.key
```

When the command finishes, copy the two new files to your `/etc/openvpn/` directory:

```
$ sudo cp ~/EasyRSA-v3.0.6/ta.key /etc/openvpn/
$ sudo cp ~/EasyRSA-v3.0.6/pki/dh.pem /etc/openvpn/
```

With that, all the certificate and key files needed by your server have been generated. You're ready to create the corresponding certificates and keys which your client machine will use to access your OpenVPN server.

Step 4 — 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 server. The benefit of this is that we can create a script which 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
```

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
```

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 ~/EasyRSA-v3.0.6/
$ ./easyrsa gen-req client1 nopass
```

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/
```

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

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

Log in to your CA machine, navigate to the EasyRSA directory, and import the certificate request:

```
$ ssh sammy@your_CA_ip
$ cd EasyRSA-v3.0.6/
$ ./easyrsa import-req /tmp/client1.req client1
```

Then sign the request 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
```

At the prompt, 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
```

SSH back into your OpenVPN server and copy the client certificate to the `/client-configs/keys/` directory:

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

Next, copy the `ca.crt` and `ta.key` files to the `/client-configs/keys/` directory as well:

```
$ sudo cp ~/EasyRSA-v3.0.6/ta.key ~/client-configs/keys/
$ sudo cp /etc/openvpn/ca.crt ~/client-configs/keys/
```

With that, your server and client's certificates and keys have all been generated and are stored in the appropriate directories on your 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 on your server.

Step 5 — Configuring the OpenVPN Service

Now that both your client and server's certificates and keys have been generated, you can begin configuring the OpenVPN service to use these credentials.

Start by copying a sample OpenVPN configuration file into the configuration directory and then extract it in order to use it as a basis for your setup:

```
$ sudo cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /etc/openvpn/
$ sudo gzip -d /etc/openvpn/server.conf.gz
```

Open the server configuration file in your preferred text editor:

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

Find the HMAC section by looking for the `tls-auth` directive. This line should already be uncommented, but if isn't then remove the "**;**" to uncomment it.

/etc/openvpn/server.conf

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

Next, find the section on cryptographic ciphers by looking for the commented out `cipher` lines. The `AES-256-CBC` cipher offers a good level of encryption and is well supported. Again, this line should already be uncommented, but if it isn't then just remove the "**;**" preceding it:

/etc/openvpn/server.conf

```
cipher AES-256-CBC
```

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

/etc/openvpn/server.conf

```
auth SHA256
```

Next, find the line containing a `dh` directive which defines the Diffie-Hellman parameters. Because of some recent changes made to EasyRSA, the filename for the Diffie-Hellman key may be different than what is listed in the example server configuration file. If necessary, change the file name listed here by removing the `2048` so it aligns with the key you generated in the previous step:

/etc/openvpn/server.conf

```
dh dh.pem
```

Finally, find the `user` and `group` settings and remove the "**;**" at the beginning of each to uncomment these lines:

/etc/openvpn/server.conf

```
user nobody
group nogroup
```

The changes you've made to the sample `server.conf` file up to this point are necessary in order for OpenVPN to function. The changes outlined below are optional, though they too are needed for many common use cases.

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

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

There are a few directives in the `server.conf` file which you must change in order to enable this functionality. Find the `redirect-gateway` section and remove the semicolon "**;**" from the beginning of the `redirect-gateway` line to uncomment it:

/etc/openvpn/server.conf

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

Just below this, find the `dhcp-option` section. Again, remove the "**;**" from in front of both of the lines to uncomment them:

/etc/openvpn/server.conf

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

This will assist clients in reconfiguring their DNS settings to use the VPN tunnel for 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.

/etc/openvpn/server.conf

```
# Optional!
port 443
```

Oftentimes, the protocol is restricted to that port as well. If so, change `proto` from UDP to TCP:

/etc/openvpn/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:

/etc/openvpn/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 two settings as their defaults.

(Optional) Point to Non-Default Credentials

If you selected a different name during the `./build-key-server` command earlier, modify the `cert` and `key` lines that you see to point to the appropriate `.crt` and `.key` files. If you used the default name, "server", this is already set correctly:

/etc/openvpn/server.conf

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

When you are finished, save and close the file.

After going through and making whatever changes to your server's OpenVPN configuration are required for your specific use case, you can begin making some changes to your server's networking.

Step 6 — Adjusting the 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.

Adjust your server's default IP forwarding setting by modifying the `/etc/sysctl.conf` file:

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

Inside, look for the commented line that sets `net.ipv4.ip_forward`. Remove the "**#**" character from the beginning of the line to uncomment this setting:

/etc/sysctl.conf

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

Save and close the file when you are finished.

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

```
$ sudo sysctl -p
```

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

If you followed the Debian 10 initial server setup guide listed in the prerequisites, you should have a UFW firewall in place. Regardless of whether you use the firewall to block unwanted traffic (which you almost always should do), for this guide you need a firewall to manipulate some of the traffic coming into the server. Some of the firewall rules must be modified 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 | grep default
```

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 203.0.113.1 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
```

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
```

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.

Also, in case you didn't add the SSH port when completing the prerequisite tutorial, add it here as well:

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

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
```

Your server is now configured to correctly handle OpenVPN traffic.

Step 7 — Starting and Enabling the OpenVPN Service

You're finally ready to start the OpenVPN service on your server. This is done using the systemd utility `systemctl`.

Start the OpenVPN server by specifying your configuration file name as an instance variable after the systemd unit file name. The configuration file for your server is called `/etc/openvpn/server.conf`, so add `@server` to end of your unit file when calling it:

```
$ sudo systemctl start openvpn@server
```

Double-check that the service has started successfully by typing:

```
$ sudo systemctl status openvpn@server
```

If everything went well, your output will look something like this:

```
Output
● openvpn@server.service - OpenVPN connection to server
   Loaded: loaded (/lib/systemd/system/openvpn@.service; disabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-07-17 03:39:24 UTC; 29s ago
     Docs: man:openvpn(8)
           https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
           https://community.openvpn.net/openvpn/wiki/HOWTO
 Main PID: 3371 (openvpn)
   Status: "Initialization Sequence Completed"
    Tasks: 1 (limit: 3587)
   Memory: 1.2M
   CGroup: /system.slice/system-openvpn.slice/openvpn@server.service
           └─3371 /usr/sbin/openvpn --daemon ovpn-server --status /run/openvpn/server.status 10 --cd /etc/openvpn
--config /etc/openvpn/server.conf --writepid /run/openvpn/
```

You can also check that the OpenVPN `tun0` interface is available by typing:

```
$ ip addr show tun0
```

This will output a configured interface:

```
Output
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::dd60:3a78:b0ca:1659/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

After starting the service, enable it so that it starts automatically at boot:

```
$ sudo systemctl enable openvpn@server
```

Your OpenVPN service is now up and running. Before you can start using it, though, you must first create a configuration file for the client machine. This tutorial already went over how to create certificate/key pairs for clients, and in the next step we will demonstrate how to create an infrastructure that will generate client configuration files easily.

Step 8 — 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
```

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.conf
```

Open this new file in your text editor:

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

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 "`;`" 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:

~/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.conf` file:

~/client-configs/base.conf

```
cipher AES-256-CBC
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. Although you can include these directives in every client configuration file, you only need to enable them for Linux clients that ship with an `/etc/openvpn/update-resolv-conf` file. This script uses 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
```

If your client is running Linux and has an `/etc/openvpn/update-resolv-conf` file, uncomment these lines from the client's configuration file after it has been generated.

Save and close the file when you are finished.

Next, create a simple 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
```

Inside, add the following content, making sure to change `sammy` to that of your server's non-**root** user account:

~/client-configs/make_config.sh

```
#!/bin/bash

# First argument: Client identifier

KEY_DIR=/home/sammy/client-configs/keys
OUTPUT_DIR=/home/sammy/client-configs/files
BASE_CONFIG=/home/sammy/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-auth>') \
    ${KEY_DIR}/ta.key \
    <(echo -e '</tls-auth>') \
    > ${OUTPUT_DIR}/${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
```

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 this is that if you ever need to add a client in the future, you can just run this script to quickly create the 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 9 — 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 4. 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
$ sudo ./make_config.sh client1
```

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

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

```
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 using the `client1.ovpn` example which you can run from your local computer (macOS or Linux). It places the `.ovpn` file in your home directory:

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

Here are several tools and tutorials for securely transferring files from the 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 10 — 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 see 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.

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

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

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

```
client$ sudo yum install epel-release
client$ sudo yum install openvpn
```

Configuring

Check to see if your distribution includes an `/etc/openvpn/update-resolv-conf` script:

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

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

Next, edit the OpenVPN client configuration file you transfered:

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

If you were able to find an `update-resolv-conf` file, 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.

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

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

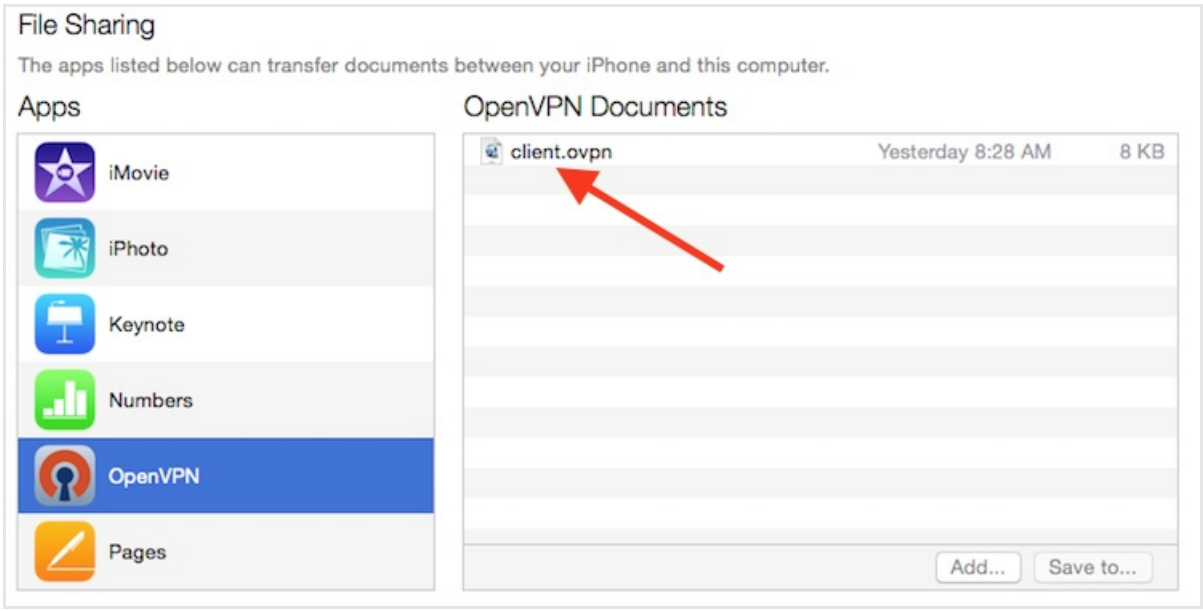
This should connect you to your VPN.

iOS

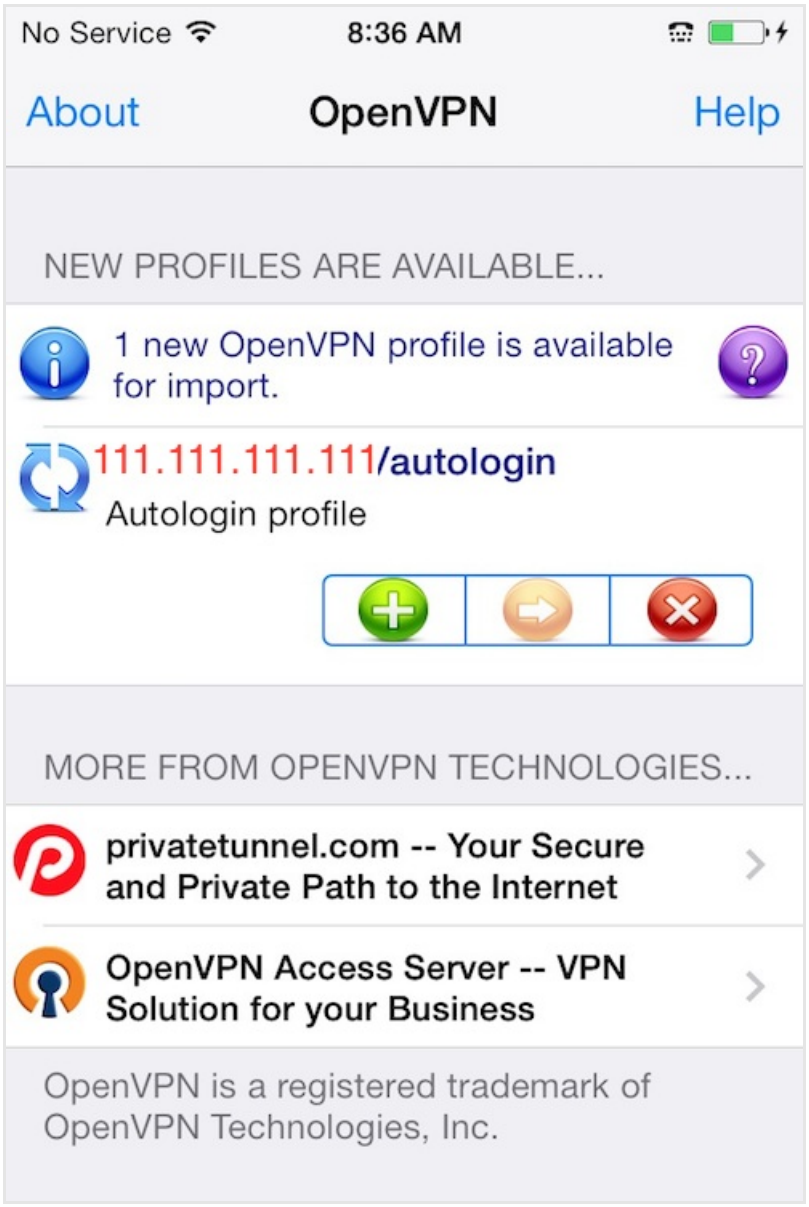
Installing

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.



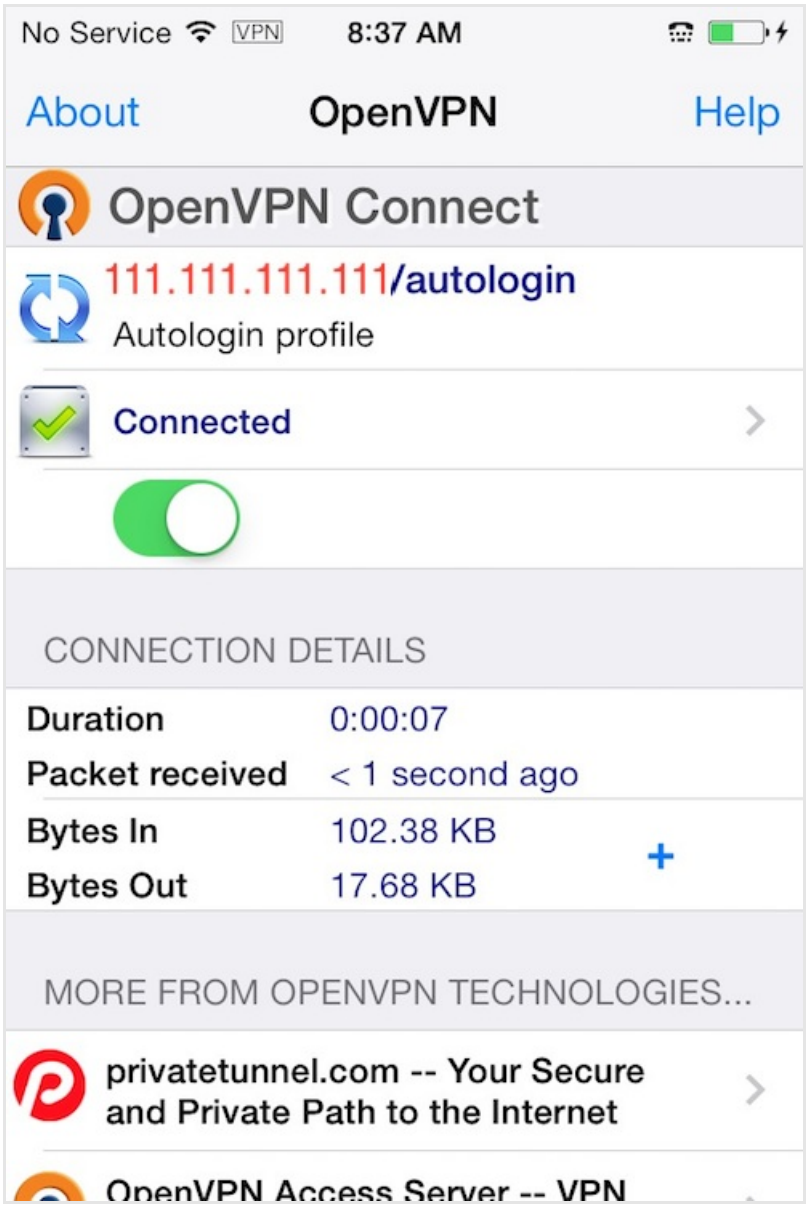
Now launch the OpenVPN app on the iPhone. You will receive a notification that a new profile is ready to import. Tap the green plus sign to import it.



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.



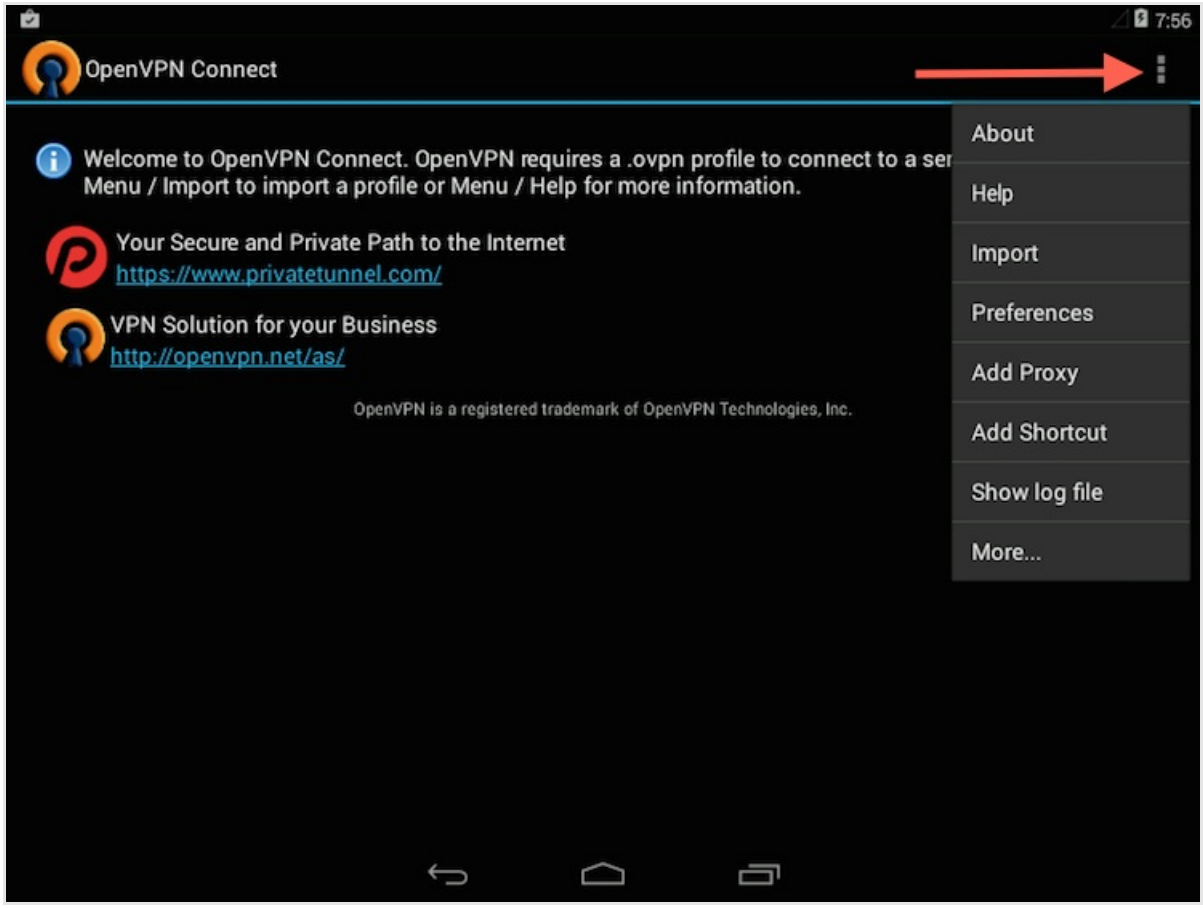
Android

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.

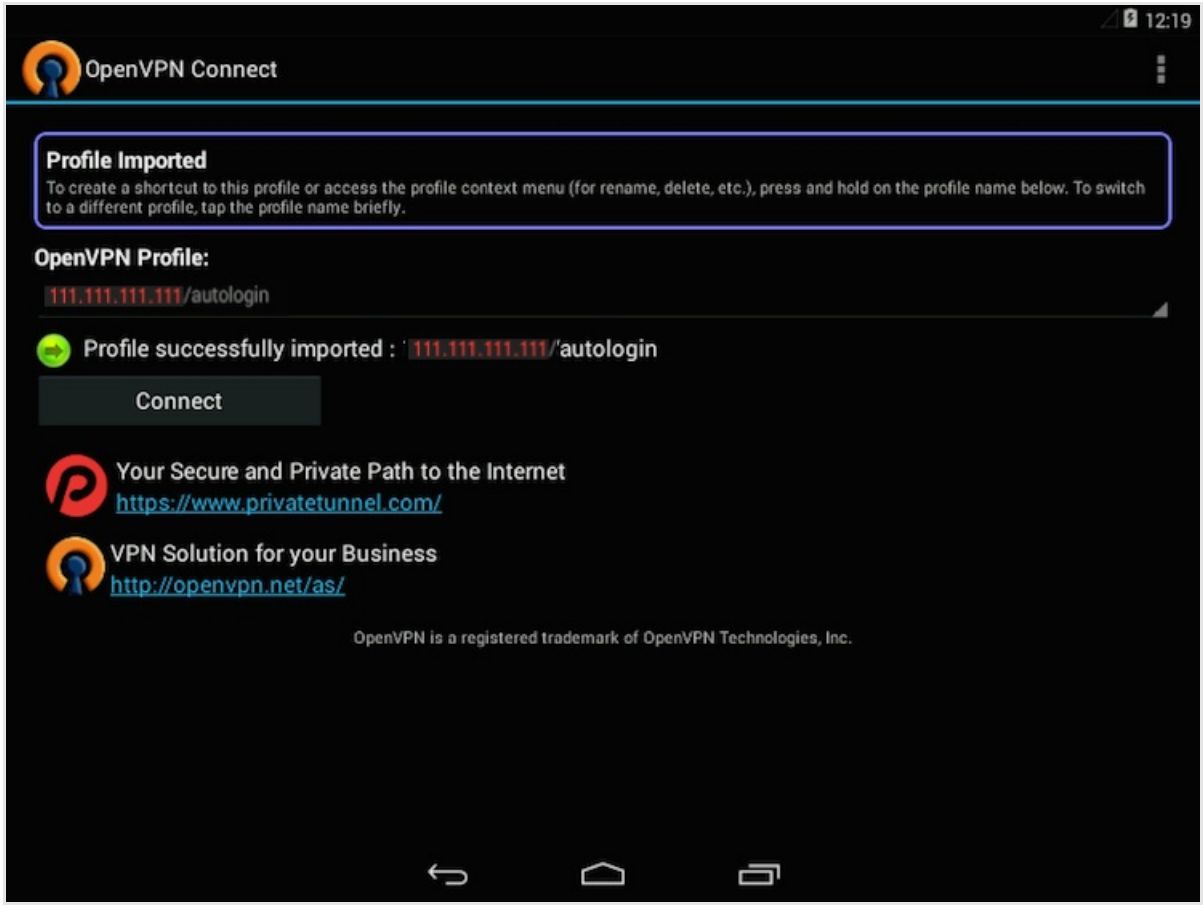
Start the OpenVPN app and tap the menu to import the profile.



Then navigate to the location of the saved profile (the screenshot uses `/sdcard/Download/`) and select the file. The app will make a note that the profile was imported.

Connecting

To connect, simply tap the **Connect** button. You'll be asked if you trust the OpenVPN application. Choose **OK** to initiate the connection. To disconnect from the VPN, go back to the OpenVPN app and choose **Disconnect**.



Step 11 — 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 5.

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 12 — Revoking Client Certificates

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

To do so, navigate to the EasyRSA directory on your CA machine:

```
$ cd EasyRSA-v3.0.6/
```

Next, run the `easyrsa` script with the `revoke` option, followed by the client name you wish to revoke:

```
$ ./easyrsa revoke client2
```

This will ask you to confirm the revocation by entering `yes`:

```
Output
Please confirm you wish to revoke the certificate with the following subject:

subject=
    commonName                = client2


Type the word 'yes' to continue, or any other input to abort.
  Continue with revocation: yes
```

After confirming the action, the CA will fully revoke the client's certificate. However, your OpenVPN server currently has no way to check whether any clients' certificates have been revoked and the client will still have access to the VPN. To correct this, create a certificate revocation list (CRL) on your CA machine:

```
$ ./easyrsa gen-crl
```

This will generate a file called `crl.pem`. Securely transfer this file to your OpenVPN server:

```
$ scp ~/EasyRSA-v3.0.6/pki/crl.pem sammy@your_server_ip:/tmp
```

On your OpenVPN server, copy this file into your `/etc/openvpn/` directory:

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

Next, open the OpenVPN server configuration file:

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

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.conf

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

Save and close the file.

Finally, restart OpenVPN to implement the certificate revocation:

```
$ sudo systemctl restart openvpn@server
```

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` 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 are now securely traversing the internet protecting your identity, location, and traffic from snoopers and censors.

To configure more clients, you only need to follow steps **4** and **9-11** for each additional device. To revoke access to clients, just follow step **12**.

By Justin Ellingwood and Mark Drake

Related

TUTORIAL

How To Deploy and Manage Your DNS Using DNSControl on Debian 10

DNSControl is an infrastructure-as-code tool that allows you to deploy and manage your DNS

TUTORIAL

How To Install Apache Kafka on Debian 10

Apache Kafka is a popular distributed message broker designed to efficiently handle large volumes of real-time data. In this

TUTORIAL

How To Set Up the code-server Cloud IDE Platform on Debian 10

In this tutorial, you will set up the code-server cloud IDE platform on your Debian 10 machine and

TUTORIAL

How To Install and Use ClickHouse on Debian 10

ClickHouse is an open-source, column-oriented analytics database created by Yandex for OLAP and big data use cases. In this

Still looking for an answer?

Ask a question

Search for more help

9 Comments

Leave a comment...

Sign In to Comment

danielvasome August 10, 2019

0

How to install this connection using a Mikrotik Router? Are there some best practices

Report

tonatihus August 16, 2019

1

Excellent tutorial! It was very useful.

- In “STEP 6”: in the example text, it is highlighted ‘eth0’ instead of ‘wlp11s0’
- In Steps 5 and 8: Isn’t redundant to add “key-direction 0”? The parameter is alraedy set at the end of the previous option (tls-auth).

Thank you very much!

Report

mdrake MOD August 27, 2019

0

Hello @tonatihus, and thank you very much for your comment.

Regarding your first bullet, that was an error on my part, thank you for catching that. I’ve updated those paragraphs to mention `eth0` instead of `wlp11s0`, as `eth0` is the default interface on a Debian 10 server from DigitalOcean. It should be noted, though, that servers provisioned from other sources or with a custom Debian image may have a different default public interface.

Regarding your second bullet, having both `key-direction 0` and `tls-auth ta.key 0` in the `server.conf` file is indeed redundant. I have updated this section to remove the instruction to add the `key-direction 0` line.

However, it is necessary to include `key-direction 1` in the client configuration file, because that file includes the `ta.key` file inline. See the Data Channel Encryption Options section of the OpenVPN Man page for more information.

Thanks again for your comment and for bringing these issues to our attention.

Report

sergey0084d6aea5031e91d4d7 September 8, 2019

0

Perfect manual as digitalocean always publish on it’s tutorials. But it is still unclear why it is not possible to perform this installation under pure root rights? Why should I avoid root and do it over sudo ?

Report

mdrake MOD September 12, 2019

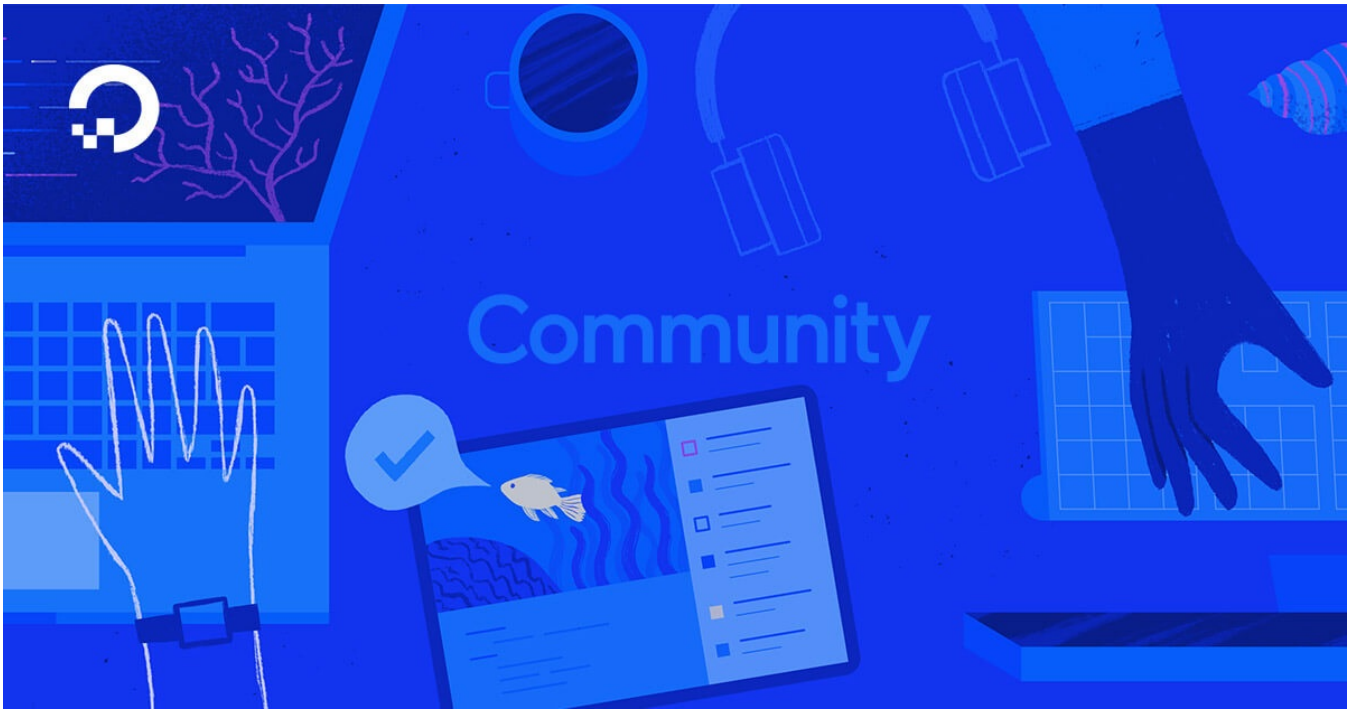
0

Hello @sergey0084d6aea5031e91d4d7, and thanks for your question.

To be clear, it’s entirely possible to install and operate OpenVPN as the **root** user. In fact, by default OpenVPN runs as the **root** user, though in this guide we configure it to run as the unprivileged **nobody** user.

Regardless, by installing and configuring it as a user with `sudo` privileges, you’re putting at least one extra layer of security between it and any potential attackers. Here’s a good general summary of why you should usually operate as a non-root user, from this tutorial:

> When you spin up a new server, a default account is created called `root`. This user has full system access and should be used only for administrative tasks. There are basically no restrictions on what you can do to your system as the root user, which is powerful, but extremely dangerous. Linux does not have an “undo” button.
>
> To alleviate this risk, we can create a new user, who has less privileges, but is more appropriately suited to everyday tasks. When you need the power of an administrative user, you can access that functionality through a command called `sudo`, which will temporarily elevate the privileges of a single command.



How To Add, Delete, and Grant Sudo Privileges to Users on a Debian VPS

by Justin Ellingwood

Adding and deleting users are basic tasks that you will have to preform on most servers. Granting sudo privileges to allow users to perform administrative functions is another common requirement. In this article, we will discuss how to create and delete users and assign administrative privileges on a Debian server.

Report

sergey0084d6aea5031e91d4d7 September 8, 2019

1

Also step 3 is killing generated CA.

~/EasyRSA-v3.0.6$ ./easyrsa init-pki

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

WARNING!!!

You are about to remove the EASYRSA_PKI at: /home/user/EasyRSA-v3.0.6/pki
and initialize a fresh PKI here.

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

Report

mdrake MOD September 12, 2019

0

Hello @sergey0084d6aea5031e91d4d7, and thanks for your comment.

It's necessary to run the `easyrsa` script with `init-pki` in Step 3 because you're running it on your OpenVPN server, not your CA machine. The OpenVPN server needs its own PKI directory, as that is where it can generate certificate requests. Likewise, the CA needs its own because that's where it can import and sign those requests.

Report

admin999391 September 15, 2019

0

Thanks for the tutorial, it is very helpful!

My server and a client both are using Debian 10. I have followed the steps here but when I run `sudo openvpn --config client1.ovpn` in my client, the connection is successfully established but I can not connect to internet or see other clients in the VPN. However, I have added client1.ovpn to Network Manager and when I activate the VPN through it, then the

connection works, I can connect to internet and see other clients in the VPN.

There is a weird thing:

After activating the VPN through the command line, the tun0 interface looks like:

```
tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen
100
    link/none
    inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::ef9f:2a38:e96e:3f7/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

But when I do it through Network Manager, the tun0 interface looks like:

```
tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen
100
    link/none
    inet 10.8.0.6 peer 10.8.0.5/32 brd 10.8.0.6 scope global noprefixroute tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::94cf:94fd:9b7b:573b/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Any ideas why it doesn't work when I run it through the command line?

Thanks!

Report

ijurisic December 4, 2019

0

How to split tunnel with OpenVPN(to access PC in my local LAN)?

I try with on client1.ovpn add this line:
route 192.168.0.0 255.255.0.0 net_gateway

But when try import to my network-manager I got this message:
Could not create new connection

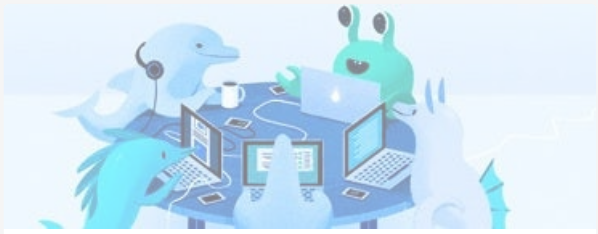
The VPN plugin failed to import the VPN connection correctly: configuration error:unsupport 3th agument net_gateway to "route".

Any idea how to fix?

Report

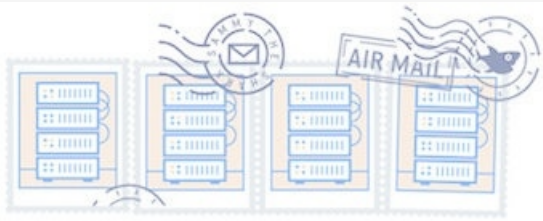
BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.



CONNECT WITH OTHER DEVELOPERS

Find a DigitalOcean Meetup near you.



GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.

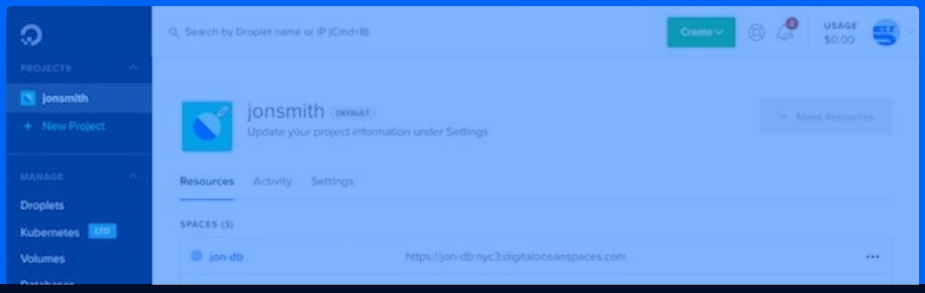
Featured on Community Kubernetes Course Learn Python 3 Machine Learning in Python Getting started with Go Intro to Kubernetes

DigitalOcean Products Droplets Managed Databases Managed Kubernetes Spaces Object Storage Marketplace

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

Learn More



Company

About

Products

Products Overview

Community

Tutorials