How To Set Up an OpenVPN Server on Ubuntu 16.04

(from DigitalOcean, May 2016)

If you are returning to this tutorial simply to configure more clients for an existing OpenVPN server - you only need to follow steps **6**, and **11-13** for each additional device. To revoke access to clients, follow step **14**.
If you've made alterations to the network interfaces (eg. adding more NICs or changing their names), then you'll have to go through step **8** as well, and make the necessary routing changes in the UFW *before rules*.

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, we'll set up an OpenVPN server on a Droplet and then configure access to it from Windows, OS X, iOS and Android. This tutorial will keep the installation and configuration steps as simple as possible for these setups.

Prerequisites

To complete this tutorial, you will need access to an Ubuntu 16.04 server.

You will need to configure a non-root user with sudo privileges before you start this guide. You can follow our Ubuntu 16.04 initial server setup guide to set up a user with appropriate permissions. The linked tutorial will also set up a firewall, which we will assume is in place during this guide.

When you are ready to begin, log into your Ubuntu server as your **sudo** user and continue below.

Step 1: Install OpenVPN

To start off, we will install **OpenVPN** onto our server. OpenVPN is available in Ubuntu's default repositories, so we can use **apt** for the installation. We will also be installing the **easy-rsa** package, which will help us set up an internal CA (certificate authority) for use with our VPN.

To update your server's package index and install the necessary packages type:

```
sudo apt-get update
sudo apt-get install openvpn easy-rsa
```

The needed software is now on the server, ready to be configured.

Step 2: Set Up the CA Directory

OpenVPN is an **TLS/SSL VPN**. This means that it utilizes certificates in order to encrypt traffic between the server and clients. In order to issue trusted certificates, we will need to set up our own simple **certificate authority** (CA).

To begin, we can copy the easy-rsa template directory into our home directory with the **make-cadir** command:

```
make-cadir ~/openvpn-ca
```

Move into the newly created directory to begin configuring the CA:

```
cd ~/openvpn-ca
```

Step 3: Configure the CA Variables

To configure the values our CA will use, we need to edit the **vars** file within the directory. Open that file now in your text editor:

```
nano vars
```

Inside, you will find some variables that can be adjusted to determine how your certificates will be created. We only need to worry about a few of these.

Towards the bottom of the file, find the settings that set field defaults for new certificates. It should look something like this:

`~/openvpn-ca/vars`

```
. . .
export KEY_COUNTRY="US"
export KEY_PROVINCE="CA"
export KEY_CITY="SanFrancisco"
export KEY_ORG="Fort-Funston"
export KEY_EMAIL="me@myhost.mydomain"
export KEY_OU="MyOrganizationalUnit"
. . .
```

Edit the values in red to whatever you'd prefer, but do not leave them blank, e.g:

`~/openvpn-ca/vars`

```
. . .
export KEY_COUNTRY="NZ"
export KEY_PROVINCE="AKL"
export KEY_CITY="AUCKLAND"
export KEY_ORG="Dycom"
export KEY_EMAIL="support@dycom.co.nz"
export KEY_OU="Private"
. . .
```

While we are here, we will also edit the **KEY_NAME** value just below this section, which populates the subject field. To keep this simple, we'll call it **server** in this guide, but you can give it really any another name (e.g *DycomServer*), as long as it's applicable. Still as we would be running only one OpenVPN server, 'server' is as applicable as any name could be:

`~/openvpn-ca/vars`

```
export KEY_NAME="server"
```

e.g. `export KEY_NAME="DycomServer"`

When you are finished, save and close the file.

Step 4: Build the Certificate Authority

Now, we can use the variables we set and the **easy-rsa** utilities to build our certificate authority.

Ensure you are in your **CA** directory, and then source the **vars** file you just edited, in order to create a number of variables in the shell session:

```
cd ~/openvpn-ca
source vars
```

sidenote:
`source` is a BASH shell built-in command that executes the content of the file passed as argument, in the current shell. It has a synonym in . (period).
Syntax

```
Source . filename [arguments]
```

You should see the following if it was sourced correctly:

Output
```
NOTE: If you run ./clean-all, I will be doing a rm -rf on
      /home/sammy/openvpn-ca/keys
```

Make sure we're operating in a clean environment by typing:

```
./clean-all
```

This bash script just wipes and creates a new KEY-DIR directory with a few default files.

Now, we can build our root CA by typing:

```
./build-ca
```

This will initiate the process of creating the **root certificate authority key** and **certificate**. Since we filled out the **vars** file, all of the values should be populated automatically. Just press ENTER through the prompts to confirm the selections:

Output
```
Generating a 2048 bit RSA private key
.................................................................
...................+++
..............................+++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [NZ]:
```

```
State or Province Name (full name) [AKL]:
Locality Name (eg, city) [Auckland]:
Organization Name (eg, company) [Dycom]:
Organizational Unit Name (eg, section) [Private]:
Common Name (eg, your name or your server's hostname) [Dycom CA]:
Name [Server]:
Email Address [support@dycom.co.nz]:
```

We now have a CA that can be used to create the rest of the files we need.

Step 5: Create the Server Certificate, Key, and Encryption Files

Next, we will generate our **server certificate** and **key pair**, as well as some additional files used during the **encryption** process.

Server Certificate and Key Pair (public/private)

Start by generating the OpenVPN server certificate and key pair. We can do this by typing:

Note: If you choose a name other than **server** here (see step 3), 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.

```
./build-key-server server
```

e.g. `./build-key-server DycomServer`

Once again, the prompts will have default values based on the argument we just passed in (server) and the contents of our **vars** file we sourced.

Feel free to accept the default values by pressing ENTER. Do *not* enter a **challenge password** for this setup. Towards the end, you will have to enter **y** to two questions to sign and commit the certificate:

```
Output
. . .

Certificate is to be certified until May  1 17:51:16 2026 GMT
(3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
```

```
Write out database with 1 new entries
Data Base Updated
```

SIDE NOTE:
A *challenge password* requested as part of key generation, is different from the *passphrase* used to encrypt the secret key (requested at key generation time, or when a plaintext key is later encrypted - and then requested again, each time the VPN or SSL-enabled service that uses it, starts up).

Key Exchange method

Next, we'll generate a few other items. We can generate strong Diffie-Hellman keys to use during key exchange by typing:

```
./build-dh
```

This might take a few minutes to complete.

Hash Function for Data Integrity and Authenticity

Afterwards, we generate an HMAC signature to strengthen the server's TLS integrity verification capabilities:

```
openvpn --genkey --secret keys/ta.key
```

Step 6: Generate a Client Certificate and Key Pair

Next, we can generate a client certificate and key pair. Although this can be done on the client machine and then signed by the server/CA for security purposes, for this guide we will generate the signed key on the server for the sake of simplicity.

We will generate a single client key/certificate for this guide, but if you have more than one client, you can repeat this process as many times as you'd like. Pass in a unique value to the script for each client. To configure more clients, you only need to follow steps **6**, and **11-13** (see later) for each additional device. If you've made alterations to the network interfaces (eg. adding more NICs or changing their names), then you'll have to go through step **8** as well, and make the necessary routing changes in the UFW *before rules*.

Because you may come back to this step at a later time, we'll re-source the **vars** file. We will use **client1** as the value for our first certificate/key pair for this guide, but you may want to use a more descriptive name (e.g. *dycom-client1*), seeing that you may prefer to create more than one key for different clients (you still can share the same client cert/keypair to different machines nonetheless).

To produce credentials without a password, to aid in automated connections, use the **build-key** command like this:

```
cd ~/openvpn-ca
source vars
./build-key client1                # e.g. ./build-key dycom-client1
```

If instead, you wish to create a password-protected set of credentials, use the **build-key-pass** command:

```
cd ~/openvpn-ca
source vars
./build-key-pass client1           # e.g. ./build-key dycom-client1
```

Enter a passphrase when prompted. Again, the defaults should be populated, so you can just hit **ENTER** to continue. Leave the challenge password blank (because you already entered a *passphrase* during key genertion) and make sure to enter **y** for the prompts that ask whether to sign and commit the certificate.

Step 7: Configure the OpenVPN Service

Next, we can begin configuring the OpenVPN service using the credentials and files we've generated.

Copy the Files to the OpenVPN Directory

To begin, we need to copy the files we need to the (server's) **/etc/openvpn** configuration directory.

We can start with all of the files that we just generated. These were placed within the **~/openvpn-ca/keys** directory as they were created. **Keep in mind** that the name of the server cert and key depends on the KEY_NAME value used for the server in steps 3 (and 5).

We need to move our CA cert, our server cert and key, the HMAC signature, and the Diffie-Hellman file:

```
cd ~/openvpn-ca/keys
sudo cp ca.crt server.crt server.key ta.key dh2048.pem /etc/openvpn
```

Next, we need to copy and unzip a sample OpenVPN configuration file into configuration directory so that we can use it as a basis for our setup. Note, depending on the name assigned to the server earlier, you could follow the pattern and use the same name for the configuration file:

```
gunzip -c /usr/share/doc/openvpn/examples/sample-config-
files/server.conf.gz | sudo tee /etc/openvpn/server.conf
```

*tee reads from standard input and writes to standard output (and files)

Adjust the OpenVPN Configuration

Now that our files are in place, we can modify the server configuration file (refer to step 5 for the name you've given to the server, and replace):

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

Basic Configuration

First, find the HMAC section by looking for the **tls-auth** directive. Remove the ";" to uncomment the **tls-auth** line. Below this, add the **key-direction** parameter set to "0":

`/etc/openvpn/server.conf`

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

Next, find the section on cryptographic ciphers by looking for the commented out cipher lines. The **AES-128-CBC** cipher offers a good level of encryption and is well supported. Remove the ";" to uncomment the **cipher AES-128-CBC** line:

`/etc/openvpn/server.conf`

```
cipher AES-128-CBC
```

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

`/etc/openvpn/server.conf`

```
auth SHA256
```

Finally, find the user and group settings and remove the ";" at the beginning of to uncomment those lines:

`/etc/openvpn/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 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.

You can do this, uncomment a few directives that will configure client machines to redirect all web traffic through the VPN. 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 202.37.101.1"
push "dhcp-option DNS 202.37.101.2"
```

Note: you could (or should?) alternatively use the default DNS entries stated in your router's configuration, eg. Cloudflare's ultra fast privacy-first public DNS

```
Eg.    1.1.1.1          #primary DNS
       1.0.0.1          #secondary DNS
```

This should 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 your OpenVPN server, port 443 is a popular choice since this is usually allowed through firewall rules.

`/etc/openvpn/server.conf`

```
# Optional!
port 443
```

Often if the protocol will be restricted to that port as well. If so, change proto from UDP to TCP:

`/etc/openvpn/server.conf`

```
# Optional!
proto tcp
```

If you have no need to use a different port, it is best to leave these two settings as their default.

(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 server, this should already be set correctly:

`/etc/openvpn/server.conf`

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

When you are finished, save and close the file.

Step 8: Adjust the Server Networking Configuration

Next, we need to adjust some aspects of the server's networking so that OpenVPN can correctly route traffic.

Allow IP Forwarding

First, we need to allow the server to forward traffic. This is fairly essential to the functionality we want our VPN server to provide.

We can adjust this setting by modifying the **/etc/sysctl.conf** file:

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

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

*`sysctl` is used to modify kernel parameters at runtime

Adjust the UFW (iptable) Rules to Masquerade (NAT) Client Connections

If you followed the Ubuntu 16.04 initial server setup guide in the prerequisites, you should have the UFW firewall in place. Regardless of whether you use the firewall to block unwanted traffic (which you almost always should do), we need the firewall in this guide to manipulate some of the traffic coming into the server. We need to modify the rules file to set up *masquerading*, an **iptables** concept that provides on-the-fly dynamic NAT to correctly route client connections.

Before we open the firewall configuration file to add masquerading, we need to find the public network interface of our machine. To do this, type:

```
ip route | grep default
```

Your public interface should follow the word "dev". For example, this result shows the interface named **wlp11s0**, which is highlighted below:

```
Output
default via 203.0.113.1 dev wlp11s0  proto static  metric 600
```

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
```

This file handles configuration that should be 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:

Note: Remember to replace *wlp11s0* 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 wlp11s0 (change to the interface you
discovered!)
-A POSTROUTING -s 10.8.0.0/8 -o wlp11s0 -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.

We need to tell UFW to allow forwarded packets by default as well. To do this, we will open the **/etc/default/ufw** file:

```
sudo nano /etc/default/ufw
```

Inside, find the **DEFAULT_FORWARD_POLICY** directive. We will change the value from DROP to ACCEPT:

/etc/default/ufw

```
DEFAULT_FORWARD_POLICY="ACCEPT"
```

Save and close the file when you are finished.

Open the OpenVPN Port and Enable the Changes

Next, we'll 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.

We'll also add the SSH port in case you forgot to add it when following the prerequisite tutorial:

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

Now, we can disable and re-enable UFW to load the changes from all of the files we've modified:

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

Our server is now configured to correctly handle OpenVPN traffic.

Step 9: Start and Enable the OpenVPN Service

We're finally ready to start the OpenVPN service on our server. We can do this using **systemd**.

We need to start the **OpenVPN** server by specifying our configuration file name as an instance variable after the *systemd* unit file name. Our configuration file for our server is called **/etc/openvpn/server.conf**, so we will add @server to end of our unit file when calling it (again, replace 'server' with the name you specified in step 3):

```
      sudo systemctl start openvpn@server
                                             e.g.   sudo systemctl start openvpn@DycomServer
```

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

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

If everything went well, your output should look something that looks 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 Tue 2016-05-03 15:30:05 EDT; 47s ago
     Docs: man:openvpn(8)
           https://community.openvpn.net/openvpn/wiki/Openvpn23ManPage
           https://community.openvpn.net/openvpn/wiki/HOWTO
  Process: 5852 ExecStart=/usr/sbin/openvpn --daemon ovpn-%i --status
/run/openvpn/%i.status 10 --cd /etc/openvpn --script-security 2 --config
/etc/openvpn/%i.conf --writepid /run/openvpn/%i.pid (code=exited, sta
 Main PID: 5856 (openvpn)
    Tasks: 1 (limit: 512)
   CGroup: /system.slice/system-openvpn.slice/openvpn@server.service
           └─5856 /usr/sbin/openvpn --daemon ovpn-server --status
/run/openvpn/server.status 10 --cd /etc/openvpn --script-security 2 --config
/etc/openvpn/server.conf --writepid /run/openvpn/server.pid

May 03 15:30:05 openvpn2 ovpn-server[5856]: /sbin/ip addr add dev tun0 local 10.8.0.1
peer 10.8.0.2
May 03 15:30:05 openvpn2 ovpn-server[5856]: /sbin/ip route add 10.8.0.0/24 via
10.8.0.2
May 03 15:30:05 openvpn2 ovpn-server[5856]: GID set to nogroup
May 03 15:30:05 openvpn2 ovpn-server[5856]: UID set to nobody
May 03 15:30:05 openvpn2 ovpn-server[5856]: UDPv4 link local (bound): [undef]
May 03 15:30:05 openvpn2 ovpn-server[5856]: UDPv4 link remote: [undef]
May 03 15:30:05 openvpn2 ovpn-server[5856]: MULTI: multi_init called, r=256 v=256
May 03 15:30:05 openvpn2 ovpn-server[5856]: IFCONFIG POOL: base=10.8.0.4 size=62,
ipv6=0
May 03 15:30:05 openvpn2 ovpn-server[5856]: IFCONFIG POOL LIST
May 03 15:30:05 openvpn2 ovpn-server[5856]: Initialization Sequence Completed
```

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

```
ip addr show tun0
```

You should see a configured interface:

```
Output
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
noqueue 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
```

If everything went well, enable the service so that it starts automatically at boot by using the following command:

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

The command above actually creates a *symlink* (similar to a *shortcut* in Windows) at */etc/systemd/system/multi-user.target.wants/openvpn@DycomServer.service* which is executed at startup.

Stop, Disable and Remove the OpenVPN Service

You **can** have more than one OpenVPN server configured on the same system, but cannot (and shouldn't attempt to) run more than one at the same. As mentioned earlier, our configuration file for our server is called **/etc/openvpn/server.conf**, so we add @server to end of our unit file when calling it (again, replace 'server' with the name you specified in step 3).

If you would like to *stop* that particular OpenVPN server, you'd run this:

```
sudo systemctl stop openvpn@server
                              e.g.   sudo systemctl stop openvpn@DycomServer
```

Should you wish to *disable* this particular OpenVPN server at startup, the following command would remove it from startup (ie. Delete the symlink previous created):

```
sudo systemctl disable openvpn@server
```

If you would like to go further and completely remove a particular OpenVPN configuration altogether – which can happen if you'd like to start over - simply remove the */etc/openvpn/server.conf* file, and the associated certificate *(*.crt)* and key *(*.key)* files:

```
sudo rm /etc/openvpn/server.*
```

Step 10: Create Client Configuration Infrastructure

Next, we need to set up a system that will allow us to create client configuration files easily.

Creating the Client Config Directory Structure

Create a directory structure within your home directory to store the files:

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

Since our client configuration files will have the client keys embedded, we should lock down permissions on our inner directory:

```
chmod 700 ~/client-configs/files
```

Creating a Base Configuration

Next, let's copy an example client configuration into our directory to use as our 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, we need to make a few adjustments.

First, locate the **remote** directive. This points the client to our OpenVPN server address. This should be the public IP address of your OpenVPN server. If you changed the port that the OpenVPN server is listening on, 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 server_IP_address_or_domain_name 1194
                                  #e.g. remote dycomserver.ddns.net 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 ";":

`~/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 we will be adding the certs and keys within the file itself:

`~/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
```

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

`~/client-configs/base.conf`

```
cipher AES-128-CBC
auth SHA256
```

Next, add the **key-direction** directive somewhere in the file. This must be set to "1" to work with the server:

`~/client-configs/base.conf`

```
key-direction 1
```

Finally, add a few **commented out** lines. We want to include these with every config, but should only enable them for Linux clients that ship with a **/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, you should uncomment these lines from the generated **OpenVPN** client configuration file.

Save the file when you are finished.

Creating a Configuration Generation Script

Next, we will create a simple script to compile our base configuration with the relevant certificate, key, and encryption files. This will place the generated configuration in the **~/client-configs/files** directory.

Create and open a file called make_config.sh within the ~/client-configs directory:

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

Inside, paste the following script:

```
                     ~/client-configs/make_config.sh

#!/bin/bash

# First argument: Client identifier

KEY_DIR=~/openvpn-ca/keys
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-auth>') \
    ${KEY_DIR}/ta.key \
    <(echo -e '</tls-auth>') \
    > ${OUTPUT_DIR}/${1}.ovpn
```

Save and close the file when you are finished.

Mark the file as executable by typing:

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

Step 11: Generate Client Configurations

Now, we can easily generate client configuration files.

If you followed along with the guide, you created a client certificate and key called **client1.crt** and **client1.key** (or dycom-client1.crt and dycom-client1.key, or similar) respectively, by running the `./build-key client1` command in step 6. We can generate a config for these credentials by moving them into our **~/client-configs** directory and using the script we made:

```
cp dycom-hacluster-client2.crt dycom-hacluster-client2.key ~/client-configs/
cd ~/client-configs
./make_config.sh client1
```

If everything went well, we should have a **client1.ovpn** file in our **~/client-configs/files** directory:

```
ls ~/client-configs/files

Output
client1.ovpn
```

Transferring Configuration to Client Devices

We need to transfer the client configuration file to the relevant device. 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 choice and device's operating system, you want the application 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 (remember, you have to get the config files at the client's end before he/she can connect!).

Here is an example SFTP command using our client1.ovpn example. This command can be run from your local computer (OS X or Linux). It places the **.ovpn** file in your home directory:

```
sftp sammy@openvpn_server_ip:client-configs/files/dycom-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 12: Install the Client Configuration

Now, we'll discuss how to install a client VPN profile on Windows, OS X, iOS, and Android. None of these client instructions are dependent on one another, so feel free to skip to whichever is applicable to you.

The OpenVPN connection will be called whatever you named the .ovpn file. In our example, this means that the connection will be called client1.ovpn for the first client file we generated.

Windows

Installing

The OpenVPN client application for Windows can be found on 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 makes it available.

OpenVPN must be run 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 can preset this, but this must be done 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 the VPN can be connected and disconnected 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** (e.g. dycom-client1) at the top of the menu (that's our **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**.

OS X

Installing

Tunnelblick is a free, open source OpenVPN client for Mac OS X. 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. It can be easier to answer **No** 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 Tunnelblick 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** menu item to initiate the VPN connection. Select the **client1** 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 Ubuntu or Debian, you can install it just as you did on the server by typing:

```
sudo apt-get update
sudo apt-get install openvpn
```

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

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

Configuring

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

```
ls /etc/openvpn

Output
update-resolve-conf
```

Next, edit the OpenVPN client configuration file you transfered:

```
nano client1.ovpn          #(dycom-client1.ovpn)
```

INSET: the following combines the previous two commands and opens `client1.ovpn` for editing if script `/etc/openvpn/update-resolv-conf` exists:

```
[[ -f /etc/openvpn/update-resolv-conf  ]] && nano client1.ovpn
```

Uncomment the three lines we placed in to adjust the DNS settings if you were able to find an update-resolv-conf file:

```
                        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 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:

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

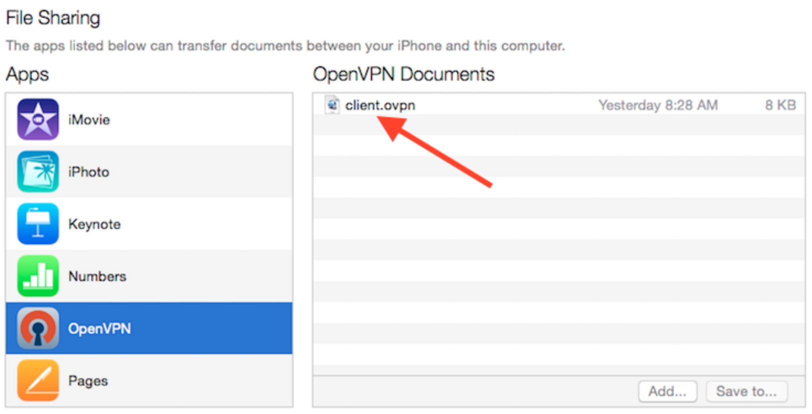
This should connect you to your server.

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.

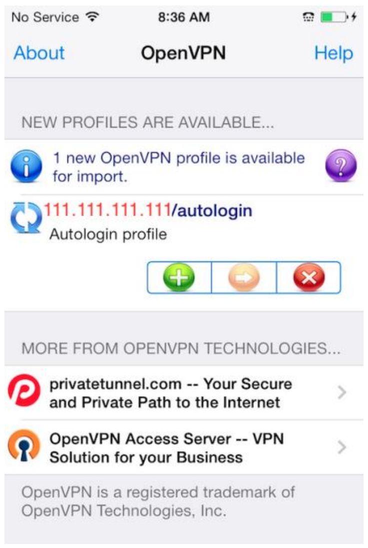
Completing the transfer with iTunes will be outlined here. Open iTunes on the computer and click on **iPhone > apps**. Scroll down to the bottom to the **File Sharing** section and click the OpenVPN app. The blank window to the right, **OpenVPN Documents**, is for sharing files. Drag the **.ovpn** file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


File Sharing

The apps listed below can transfer documents between your iPhone and this computer.

Apps	OpenVPN Documents		
iMovie	client.ovpn	Yesterday 8:28 AM	8 KB
iPhoto			
Keynote			
Numbers			
OpenVPN			
Pages			

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



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

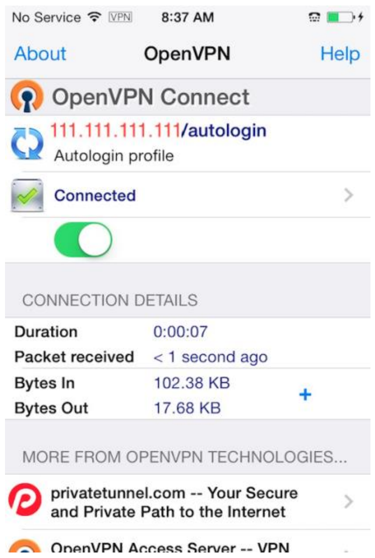
OpenVPN Access Server -- VPN Solution for your Business

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
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
OpenVPN Access Server -- VPN

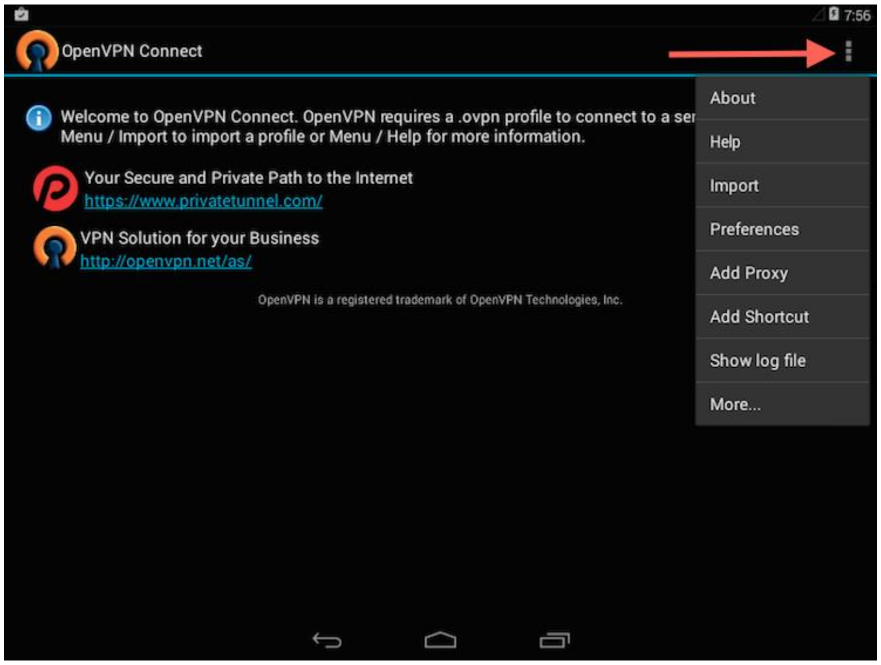
Android

Installing

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

The **.ovpn** profile can be transferred 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.



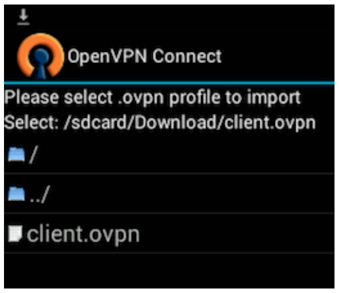
Welcome to OpenVPN Connect. OpenVPN requires a .ovpn profile to connect to a ser
Menu / Import to import a profile or Menu / Help for more information.

Your Secure and Private Path to the Internet
https://www.privatetunnel.com/

VPN Solution for your Business
http://openvpn.net/as/

OpenVPN is a registered trademark of OpenVPN Technologies, Inc.

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.

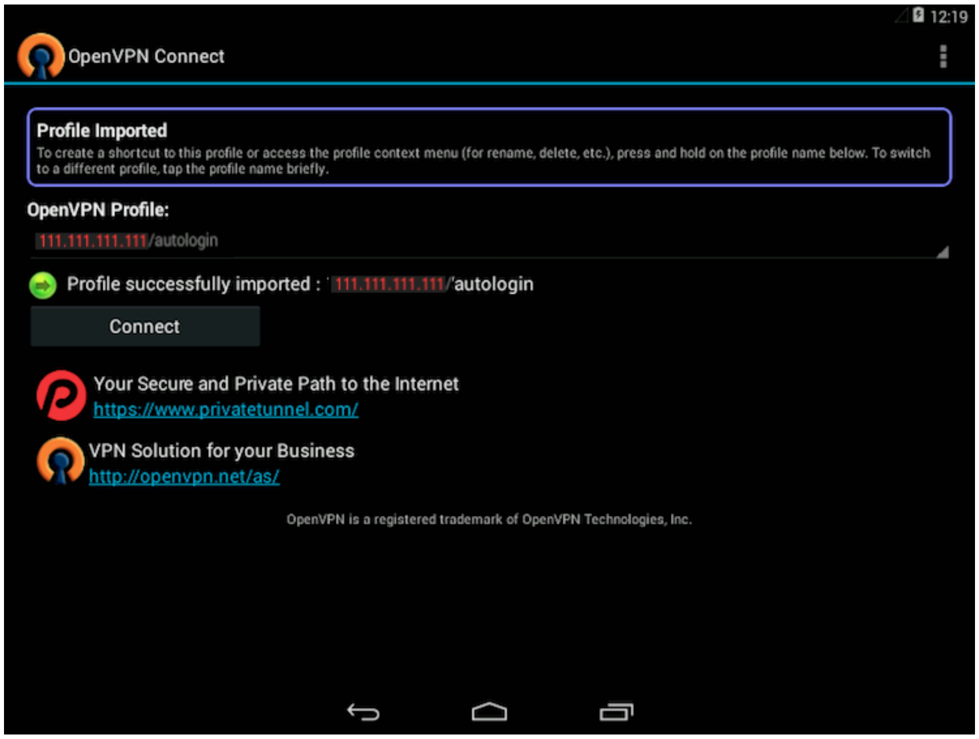


Please select .ovpn profile to import
Select: /sdcard/Download/client.ovpn

/
../
client.ovpn

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**.


12:19

OpenVPN Connect

Profile Imported
To create a shortcut to this profile or access the profile context menu (for rename, delete, etc.), press and hold on the profile name below. To switch to a different profile, tap the profile name briefly.

OpenVPN Profile:
111.111.111.111/autologin

Profile successfully imported : '111.111.111.111/'autologin

Connect

Your Secure and Private Path to the Internet
https://www.privatetunnel.com/

VPN Solution for your Business
http://openvpn.net/as/

OpenVPN is a registered trademark of OpenVPN Technologies, Inc.

The OpenVPN Android app ready to connect to the VPN

Step 13: Test Your VPN Connection

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. The completely different IP address of your VPN server should now appear. That is now 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 14: Monitoring Your VPN Connection

TBC

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, enter your **CA** directory and re-source the **vars** file:

```
cd ~/openvpn-ca
source vars
```

Next, call the **revoke-full** command using the client name that you wish to revoke:

```
./revoke-full client3
```

This will show some output, ending in **error 23**. This is normal and the process should have successfully generated the necessary revocation information, which is stored in a file called **crl.pem** within the **keys** subdirectory.

Transfer this file to the **/etc/openvpn** configuration directory:

```
sudo cp ~/openvpn-ca/keys/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, so that the OpenVPN server checks 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 now longer be able to successfully connect to the server using the old credential.

To revoke additional clients, follow this process:

1. Generate a new certificate revocation list by sourcing the **vars** file in the **~/openvpn-ca** directory and then calling the **revoke-full** script on the client name.
2. Copy the new certificate revocation list to the **/etc/openvpn** directory to overwrite the old list.
3. Restart the OpenVPN service.

This process can be used to revoke any certificates that you've previously issued for your server.

Conclusion

Congratulations! 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 **6**, and **11-13** for each additional device. To revoke access to clients, follow step **14**.