- Firstly, become root user by running command: sudo su to avoid using sudo during further configurations
- No other VPN service should be running during Open VPN configuration
- Applicable to Ubuntu 16.04 + or Cent OS 7 + servers
- Use command *apt-get install* for installing packages ubuntu server; *yum install* for installing packages on centos-based server
- SE Linux should be *permissive* or *disabled* to avoid any type of security blocking
- Check for firewall rules using *ufw, ip tables or firewall-cmd* as applicable before creating connections

Step 1: Install OpenVPN

```
# apt-get install openvpn easy-rsa
```

Step 2: Set Up the CA Directory

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

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

Step 3: Configure the CA Variables

Inside, you will find some variables that can be adjusted to determine how your certificates will be created.

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

```
export KEY_COUNTRY="US"
export KEY_PROVINCE="NY"
export KEY_CITY="New York City"
export KEY_ORG="DigitalOcean"
export KEY_EMAIL="admin@example.com"
export KEY_OU="Community"
```

While we are here, we will also edit the `KEY_NAME` value just below this section, which populates the subject field.

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

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:

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

```
# source vars
```

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
```

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) [US]:
State or Province Name (full name) [NY]:
Locality Name (eg, city) [New York City]:
Organization Name (eg, company) [DigitalOcean]:
Organizational Unit Name (eg, section) [Community]:
Common Name (eg, your name or your server's hostname) [DigitalOcean CA]:
Name [server]:
Email Address [admin@email.com]:
```

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

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` directroy, 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
```

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
```

We can generate a strong Diffie-Hellman keys to use during key exchange by typing:

```
# ./build-dh
```

This might take a few minutes to complete.

Afterwards, we can 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

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.

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.

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
```

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
```

Again, the defaults should be populated, so you can just hit **ENTER** to continue. Leave the challenge password blank and make sure to enter **y** for the prompts that ask whether to sign and commit the certificate.

Step 7: Configure the OpenVPN Service

To begin, we need to copy the files we need to the `/etc/openvpn` configuration directory.

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:

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

Now that our files are in place, we can modify the server configuration file:

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

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-256-CBC` cipher offers a good level of encryption and is well supported. Remove the "**;**" to uncomment the `cipher AES-256-CBC` line:

/etc/openvpn/server.conf

```
cipher AES-256-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.

/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 8.8.8.8"
push "dhcp-option DNS 208.67.220.220"
```

(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
```

Step 8: Adjust the Server Networking Configuration

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:

```
# 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
```

Adjust the UFW Rules to Masquerade 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

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:

```
# nano /etc/ufw/before.rules
```

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
```

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

```
# 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"
```

Open the OpenVPN Port and Enable the Changes

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.

```
# ufw allow 1194/udp
```

```
# ufw allow OpenSSH
```

```
# ufw reload
```

Step 9: Start and Enable the OpenVPN Service

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:

```
# systemctl start openvpn@server
```

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

```
# systemctl status openvpn@server
```

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

```
# systemctl enable openvpn@server
```

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

```
# ip addr show tun0
```

Or

```
# ifconfig
```

Step 10: Create Client Configuration Infrastructure

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 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-256-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.

Creating a Configuration Generation Script

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
```

Mark the file as executable by typing:

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

Step 11: Generate Client Configurations

you created a client certificate and key called `client1.crt` and `client1.key` respectively by running the `./build-key client1` command in step 6. We can generate a config for these credentials by moving into our `~/client-configs` directory and using the script we made:

```
# 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

Use winscp or filezilla to transfer the files from server to any client machine.

Step 12: Install the Client Configuration

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:

```
# apt-get install openvpn
```

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

```
# yum install epel-release

# 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
```

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:

```
# openvpn --config client1.ovpn
```

This should connect you to your server.

After successful connection use following command to verify the connection:

 # route -n

This will print all the routes for network and will show local IP of VPN as well as server's public IP as default route for internet traffic.

Step 13: 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:

```
# cp ~/openvpn-ca/keys/crl.pem /etc/openvpn
```

Next, open the OpenVPN server configuration file:

```
# 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
```

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.