This document covers the installation and management of OpenVPN (V. 2.4.9) onto CentOS8. OpenVPN allows the creation and management of private networks over the internet. CentOS8 is an opensource operating system that will be running the VPN server that the client systems will be connecting to in order to access the system. The requirements are to have Cent8 with root access.

Initial Setup and key generation:

We begin by installing the epel-repository that will enable us to access the latest packages. We will then install OpenVPN and Easy RSA. Type the following commands:

dnf install epel-release
dnf install openvpn

Once installed, we will begin the setup and generation of certificates. We can use dnf to get the easyrsa files, but for this document, we will pull from the github repository. The only difference is that we won't need to copy things over from the easy-rsa directory.

cd /etc/openvpn/
wget
https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.6/EasyRSA-unix-v3.0.6.tgz

Now extract the 'EasyRSA-unix-v3.0.6.tgz' file and rename the directory to 'easy-rsa'.

tar -xf EasyRSA-unix-v3.0.6.tgz mv EasyRSA-v3.0.6/ easy-rsa/; rm -f EasyRSA-unix-v3.0.6.tgz

Next we navigate to the new location of the Easy-RSA files and update the variables listing to suit our organization:
cd /etc/openvpn/easy-rsa/
vim vars.example

Note that there should be included with the files a vars.example that can be used as a template. Otherwise simply using vim to write out a new vars file can be used with the correct variables from a new document.

Ensure that you write out the file as vars, not vars.example. Additionally, make the vars file executable using chmod:
chmod +x vars

The next step in the process is to generate the keys and certificates. Located in the previous directory are the tools needed to get started. We begin by navigating to the appropriate directory and creating the Certificat Authority:

./easyrsa init-pki
./easyrsa build-ca

You will be prompted to create a password for the CA key. It is also worth noting that the server itself does not need to be the CA, another machine can act in that capacity. This would increase security, though all certs would need to be migrated to the other machine to be signed, so for the sake of this document our server will also be the CA. To avoid possible password usage issues that may come up later, create a .pass file within the openvpn directory with the passphrase written in clear text:
vim /etc/openvpn/exapmle.pass

Moving onto generating the server key and having our CA sign it, issue the following commands:

./easyrsa gen-req our-server passwordhere
./easyrsa sign-req server our-server

Optionally you can set the password to nopass which will disable the password for our server's key. We will be asked for the CA password in order to sign the ky for our new server key. We can verify our actions by using the OpenSSL command as follows:

Openssl verify -CAfile pki/ca.crt pki/issued/our-server.crt

Generating client keys is similar to that of the server key, though we must make sure to specify that it is a client:
./easyrsa gen-req exclient passwordhere
./easyrsa sign-req client exclient
*NOTE: The name 'exclient' is simply being used as a placeholder. Another name should be used, and that name should be substituted in other locations in this document that make use of it. The same goes for 'our-server' and it's usage throughout this document.

Again, optionally, we may disable the password option here, and we will want to verify the signing of the certificates with the OpenSSL command. Note the change in location:

openssl verify -CAfile pki/ca.crt pki/issued/exclient.crt

For additonal security, it is suggested that a Diffie-Hellman key be used. The generation of the key may take some time to process, and will be located in the pki directory:

./easyrsa gen-dh

Presuming we will be having multiple client keys used in our server, a Certificate Revoking List (CRL)will make the management simpler. The following commands can be used to revoke a client and update the CRL, and we will run it now to generate the CRL certificate:

./easyrsa gen-crl

Now that we have both a server and client key, we should move them for easier management. We will copy them from the pki directory into a server and client directory respectively, as well as the Diffie-Hellman and CRL key:

cp pki/ca.crt /etc/openvpn/server/
cp pki/issued/our-server.crt /etc/openvpn/server/
cp pki/private/our-server.key /etc/openvpn/server/

cp pki/ca.crt /etc/openvpn/client/keys
cp pki/issued/exclient.crt /etc/openvpn/client/keys
cp pki/private/exclient.key /etc/openvpn/client/keys

cp pki/dh.pem /etc/openvpn/server/
cp pki/crl.pem /etc/openvpn/server/

Additional Security and Cryptographic Key Generation:

Security can be increased by using an extra shared key that will be used by server and all clients; the tls-crypt. The following command will generate it, note the location as this will be needed for completion of the configuration files:
openvpn --genkey --secret /etc/openvpn/server/exkey.tlsauth

Copy the key to the ~/client/keys directory as well.

```
cp  /etc/openvpn/server/exkey.tlsauth /etc/openvpn/client/keys
```

OpenVPN Configuration:

Here we will edit the .conf file that is included in the example files. The location of the sample configuration files is the doc directory. We will copy them to the server directory:

cp /usr/share/doc/openvpn/sample/sample-config-files/server.conf /etc/openvpn/server

Optionally, we will provide a pre-edited .conf file as an example. Regardless, the following items need to be edited.

We will use the default port 1194 unless otherwise needed. See the comments within the .conf for further clarification:

cd /etc/openvpn/server
vim server.conf

Within the .conf, ensure that the certificates and keys accurately point to the correct locations, ie

ca /etc/openvpn/server/ca.crt

Additionally, add the CRL key and location.
crl-verify /etc/openvpn/server/crl.pem

The tls-crypt from earlier will also need to be addressed here; add the following:
tls-crypt /etc/openvpn/server/exkey.tlsauth

Uncomment “cipher AES-256-CBC” and add “auth SHA256”

Uncomment the following:
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 208.67.222.222"

The first will have clients redirect traffic through the VPN, while the second will point the client to the assigned DNS. Note, you may use any DNS. To alleviate a probable password issue, add the askpass option with the path to the .pass file created in a previous step:
askpass /etc/openvpn/example.pass

Note on managing the network: We will want to assign specific IP addresses to specific clients, we will end up wanting to use the subdirectory ccd for client-specific configuration files. We will need to uncomment the client-config-dir ccd line, route the specified address range, and create a file ccd/exclient

Enabling Port Forwarding and Configure Routing in Firewalld

Your organization may have their own firewall settings and configurations, so default to their rules and policies. We will need to enable port forwarding and cofigure the routing of firewalld.

We begin by enabling portforwarding (if not already enabled)

echo 'net.ipv4.ip_forward = 1' >> /etc/sysctl.conf

Add the OpenVPN service and the tun connection to the firewall, and ensure that they remain their permanently:

firewall-cmd --permanent --add-service=openvpn

firewall-cmd --permanent --zone=trusted --add-service=openvpn

firewall-cmd --permanent --zone=trusted --add-interface=tun0

firewall-cmd --permanent --add-masquerade

Create a variable to represent the primary network interface the server will be using, then add it to the routing policy, and implement these changes by reloading firewalld:

SERVERIP=$(ip route get 1.1.1.1 | awk 'NR==1 {print $(NF-2)}')

firewall-cmd --permanent --direct --passthrough ipv4 -t nat -A POSTROUTING -s 10.8.0.0/24 -o $SERVERIP -j MASQUERADE

firewall-cmd --reload

As OpenVPN is managed by sytemctl, we use it to start up the server:

systemctl -f enable openvpn-server@server

systemctl start openvpn-server@server

systemctl status openvpn-server@server

Once the server is running, we can check netstat to see all active connections:

netstat -plntu

Initializing automatic client key setup

Create a directory within the client directory to store the files that will be used to generate the configuration files needed by any client to join the network.

Mkdir /etc/openvpn/client/files

Similar to the server.conf file within the sample documents, we will use that as the base for our .conf.

cp /usr/share/doc/openvpn/sample/sample-config-files/client.conf /etc/openvpn/client/base.conf

Edit the new ‘base.conf’ file to have the server’s ip and the port (1194), the correct protocol, and comment out the ca, cert, and keys as well as the tls-auth. Add ‘auth SHA256’ under the cipher item. We will add ’key-direction 1’ which is needed for working in udp.

Additionally, we can add the following code, which will be needed for linux clients. We will simply uncomment them should they be needed(they wont):

; script-security 2

; up /etc/openvpn/update-resolv-conf

; down /etc/openvpn/update-resolv-conf

That finishes the base.conf. Now we will create the script that will generate configuration file and wrap it with the needed certs. Create a new file, and enter the following (Note the use of the exkey.tlsauth, as the name of your auth key may vary):

nano /etc/openvpn/client/make_config.sh

```
#!/bin/bash
# First argument: Client identifier
KEY_DIR=/etc/openvpn/client/keys
OUTPUT_DIR=/etc/openvpn/client/files
BASE_CONFIG=/etc/openvpn/client/base.conf

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

Also, mark the file as executable with the following:
chmod 700 /etc/openvpn/client/make_config.sh

This script will allow us to create and store a new config file with all of the necessary information. Navigate to it's location in the client directory and run it as follows, ensure that we use the name selected for our client from generating the certs earlier:

./make_config.sh exclient

This will generate an OpenVPN configuration file needed, an .ovpn, which will be stored in the files directory. Each .ovpn file will store all of the necessary certs, and so only that one file will need to be exported to the client machine.

Once the client receives the .ovpn file the can import it if on windows, or on a linux machine simply run:

Sudo openvpn --config exclient.ovpn

which should connect the client to the VPN.

Routing Clients to specific subnets

With regard to assigning specific IP addresses to specific clients, we can make use of the ccd. Initially, there seemed an easy way and a slightly more advanced way of sorting the traffic, but a third way, of merging the two ideas will be implemented. The easy way would be to simply use different certificates to manage multiple clients, with entire organizations using the same certs or simply using multiple instances of OpenVPN. The more advanced way would be using IP routing to sequester the clients to certain subnets. We will create rules that the certs will follow, and can issue the same cert out, if necessary. We will need to use the name of a client certificate that we have already created, and so we will make use of the exclient that was used.

We begin by uncommenting the following lines within the server.conf file:

```
;client-config-dir ccd
;route 192.168.4.0 255.255.255.0
```

The above range is an example. This will tell openVPN that the subnet listed should be routed to exclient. To assign these bounds we create a directory for the ccd that was mentioned earlier in the documentation,

mkdir /etc/openvpn/ccd

We create a file within the ccd titled exclient and add a line indicating it's behaviour:

vim /etc/openvpn/ccd/exclient

and add the line:

```
iroute 192.168.4.0 255.255.255.0
```

This will allow e

To assign a specific IP address We begin by uncommenting the following lines within the server.conf file:

```
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
```

The selected IP address is an example. To assign this address we create a directory for the ccd that was mentioned earlier in the documentation,

mkdir /etc/openvpn/ccd

We create or edit the ccd titled exclient and add a line indicating its behaviour:

vim /etc/openvpn/ccd/exclient

and add the line:

```
lfconfig-push 10.9.0.1 10.9.0.2
```

This will allow exclient to have a fixed IP address of 10.9.0.1