Set Up OpenConnect VPN Server (ocserv) on Ubuntu 20.04 with Let’s Encrypt

Last Updated: January 15th, 2023 Xiao Guoan (Admin)
145 Comments

Ubuntu

This tutorial is going to show you how to run your own VPN server by installing OpenConnect VPN server on Ubuntu 20.04. OpenConnect VPN server, aka **ocserv**, is an open-source implementation of Cisco AnyConnnect VPN protocol, which is widely used in businesses and universities. AnyConnect is an SSL-based VPN protocol that allows individual users to connect to a remote network.

Note: This tutorial also works on Ubuntu 20.10 and Ubuntu 21.04.

Why Set Up Your Own VPN Server?

- Maybe you are a VPN service provider or a system administrator, which behooves you to set up our own VPN server.
- You don’t trust the no-logging policy of VPN service providers, so you go the self-host route.
- You can use VPN to implement network security policy. For example, if you run your own email server, you can require users to log in only from the IP address of the VPN server by creating an IP address whitelist in the firewall. Thus, your email server is hardened to prevent hacking activities.
- Perhaps you are just curious to know how VPN server works.

Run Your Own VPN Server on Ubuntu 20.04
with OpenConnect (ocserv)

Features of OpenConnect VPN Server

- Lightweight and fast. In my test, I can watch YouTube 4K videos with OpenConnect VPN. YouTube is blocked in my country (China).
- Runs on Linux and most BSD servers.
- Compatible with Cisco AnyConnect client
- There are OpenConnect client software for Linux, MacOS, Windows and OpenWRT. For Android and iOS, you can use the Cisco AnyConnect Client.
- Supports password authentication and certificate authentication
- Supports RADIUS accounting.
- Supports virtual hosting (multiple domains).
- Easy to set up
- Resistant to deep packet inspection (DPI)

I particularly like the fact that compared to other VPN technologies, it is very easy and convenient for the end-user to use OpenConnect VPN. Whenever I install a Linux distro on my computer and want to quickly unblock websites or hide my IP address, I install OpenConnect client and connect to the server with just two lines of commands:

```
sudo apt install openconnect
sudo openconnect -b vpn.mydomain.com
```

There is also OpenConnect VPN client for Fedora, RHEL, CentOS, Arch Linux and OpenSUSE. You can easily install it with your package manager.

```
sudo dnf install openconnect
sudo yum install openconnect
sudo pacman -S openconnect
```

Requirements

To follow this tutorial, you will need a VPS (Virtual Private Server) that can access blocked websites freely (Outside of your country or Internet filtering system). I recommend Kamatera VPS, which features:

- 30 days free trial.
- Starts at $4/month (1GB RAM)
- High-performance KVM-based VPS
- 9 data centers around the world, including United States, Canada, UK, Germany, The Netherlands, Hong Kong, and Isreal.

Follow the tutorial linked below to create your Linux VPS server at Kamatera.

- How to Create a Linux VPS Server on Kamatera

Once you have a VPS running Ubuntu 20.04, follow the instructions below.

You also need a domain name to enable HTTPS for OpenConnect VPN. I registered my domain name from NameCheap because the price is low and they give whois privacy protection free for life.

Step 1: Install OpenConnect VPN Server on Ubuntu 20.04

Log into your Ubuntu 20.04 server. Then use `apt` to install the `ocserv` package from the default Ubuntu repository.

```
sudo apt update
sudo apt install ocserv
```

Once installed, the OpenConnect VPN server is automatically started. You can check its status with:

```
systemctl status ocserv
```

Sample output:

```
● ocserv.service - OpenConnect SSL VPN server
     Loaded: loaded (/lib/systemd/system/ocserv.service; enabled
; vendor preset: enabled)
     Active: active (running) since Sun 2020-04-12 19:57:08 HKT;
12s ago
       Docs: man:ocserv(8)
   Main PID: 216409 (ocserv-main)
      Tasks: 2 (limit: 9451)
     Memory: 1.6M
     CGroup: /system.slice/ocserv.service
             ├─216409 ocserv-main
             └─216429 ocserv-sm
```

Hint: If the above command doesn't quit immediately, you can press the Q key to gain back control of the terminal.

If it's not running, then you can start it with:

```
sudo systemctl start ocserv
```

By default OpenConnect VPN server listens on TCP and UDP port 443. If it's being used by web server, then the VPN server would probably fail to start. We will see how to change the port in OpenConnect VPN configuration file later.

If there's a firewall running on your server, then you will need to open port 80 and 443. For example, if you use UFW, then run the following command.

```
sudo ufw allow 80,443/tcp
```

Step 2: Install Let's Encrypt Client (Certbot) on Ubuntu 20.04 Server

The `gnutls-bin` package installed along with `ocserv` provides tools to create your own CA and server certificate, but we will obtain and install Let's Encrypt certificate. The advantage of using Let's Encrypt certificate is that it's free, easier to set up and trusted by VPN client software.

Run the following commands to install Let's Encrypt client (certbot) from the default Ubuntu repository.

```
sudo apt install certbot
```

To check the version number, run

```
certbot --version
```

Sample output:

```
certbot 0.40.0
```

Step 3: Obtain a Trusted TLS Certificate from Let's Encrypt

I recommend using the `standalone` or `webroot` plugin to obtain TLS certificate for ocserv.

Standalone Plugin

If there's no web server running on your Ubuntu 20.04 server and you want OpenConnect VPN server to use port 443, then you can use the standalone plugin to obtain TLS certificate from Let's Encrypt. Run the following command. Don't forget to set A record for your domain name.

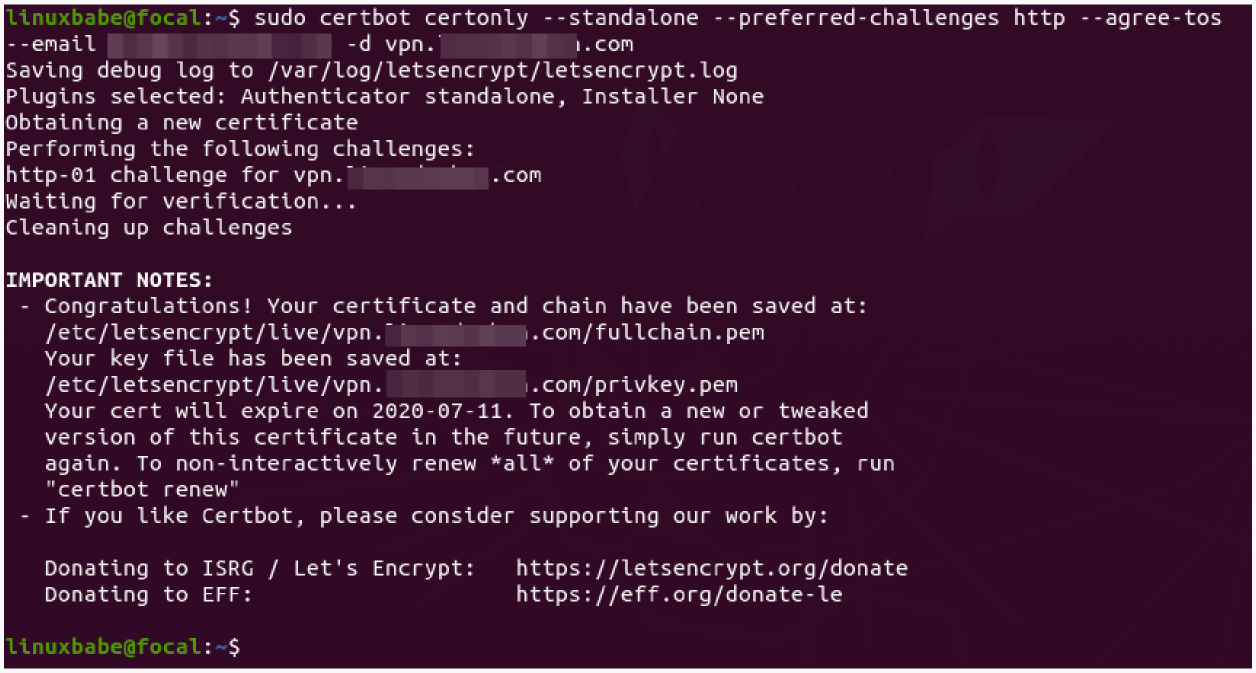
```
sudo certbot certonly --standalone --preferred-challenges http -
-agree-tos --email you@example.com -d vpn.example.com
```

Where:

- `certonly`: Obtain a certificate but don't install it.
- `--standalone`: Use the standalone plugin to obtain a certificate

- `--preferred-challenges http`: Perform http-01 challenge to validate our domain, which will use port 80.
- `--agree-tos`: Agree to Let's Encrypt terms of service.
- `--email`: Email address is used for account registration and recovery.
- `-d`: Specify your domain name.

As you can see the from the following screenshot, I successfully obtained the certificate.



```
linuxbabe@focal:~$ sudo certbot certonly --standalone --preferred-challenges http --agree-tos
--email [redacted] -d vpn.[redacted].com
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator standalone, Installer None
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for vpn.[redacted].com
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/vpn.[redacted].com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/vpn.[redacted].com/privkey.pem
   Your cert will expire on 2020-07-11. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot
   again. To non-interactively renew *all* of your certificates, run
   "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le

linuxbabe@focal:~$
```

Using webroot Plugin

If your Ubuntu 20.04 server has a web server listening on port 80 and 443, then it's a good idea to use the webroot plugin to obtain a certificate because the webroot plugin works with pretty much every web server and we don't need to install the certificate in the web server.

First, you need to create a virtual host for vpn.example.com.

Apache

If you are using Apache, then

```
sudo nano /etc/apache2/sites-available/vpn.example.com.conf
```

And paste the following lines into the file.

```
<VirtualHost *:80>
        ServerName vpn.example.com

        DocumentRoot /var/www/ocserv
</VirtualHost>
```

Save and close the file. Then create the web root directory.

```
sudo mkdir /var/www/ocserv
```

Set www-data (Apache user) as the owner of the web root.

```
sudo chown www-data:www-data /var/www/ocserv -R
```

Enable this virtual host.

```
sudo a2ensite vpn.example.com
```

Reload Apache for the changes to take effect.

```
sudo systemctl reload apache2
```

Once virtual host is created and enabled, run the following command to obtain Let’s Encrypt certificate using webroot plugin.

```
sudo certbot certonly --webroot --agree-tos --email you@exmaple.
com -d vpn.example.com -w /var/www/ocserv
```

Nginx

If you are using Nginx, then

```
sudo nano /etc/nginx/conf.d/vpn.example.com.conf
```

Paste the following lines into the file.

```
server {
      listen 80;
      server_name vpn.example.com;

      root /var/www/ocserv/;

      location ~ /.well-known/acme-challenge {
         allow all;
      }
}
```

Save and close the file. Then create the web root directory.

```
sudo mkdir -p /var/www/ocserv
```

Set www-data (Nginx user) as the owner of the web root.

```
sudo chown www-data:www-data /var/www/ocserv -R
```

Reload Nginx for the changes to take effect.

```
sudo systemctl reload nginx
```

Once virtual host is created and enabled, run the following command to obtain Let's Encrypt certificate using webroot plugin.

```
sudo certbot certonly --webroot --agree-tos --email you@exmaple.
com -d vpn.example.com -w /var/www/ocserv
```

Step 4: Edit OpenConnect VPN Server Configuration File

Edit ocserv main configuration file.

```
sudo nano /etc/ocserv/ocserv.conf
```

First, we need to configure password authentication. By default, password authentication through PAM (Pluggable Authentication Modules) is enabled, which allows you to use Ubuntu system accounts to login from VPN clients. This behavior can be disabled by commenting out the following line.

```
auth = "pam[gid-min=1000]"
```

If we want users to use separate VPN accounts instead of system accounts to login, we need to add the following line to enable password authentication with a password file.

```
auth = "plain[passwd=/etc/ocserv/ocpasswd]"
```

After finishing editing this config file, we will see how to use `ocpasswd` tool to generate the `/etc/ocserv/ocpasswd` file, which contains a list of usernames and encoded passwords.

Note: Ocserv supports client certificate authentication, but Let's Encrypt does not issue client certificate. You need to set up your own CA to issue client certificate.

Next, find the following two lines.

```
tcp-port = 443
udp-port = 443
```

Comment out the UDP port. (We will use TCP BBR algorithm to boost TCP speed.)

```
tcp-port = 443
#udp-port = 443
```

If you don't want ocserv to use TCP port 443 (there's a web server using port 443?), then change the TCP port number. Otherwise leave it alone.

Then find the following two lines. We need to change them.

```
server-cert = /etc/ssl/certs/ssl-cert-snakeoil.pem
server-key = /etc/ssl/private/ssl-cert-snakeoil.key
```

Replace the default setting with the path of Let's Encrypt server certificate and server key file.

```
server-cert = /etc/letsencrypt/live/vpn.example.com/fullchain.pe
m
server-key = /etc/letsencrypt/live/vpn.example.com/privkey.pem
```

Then, set the maximal number of clients. Default is 128. Set to zero for unlimited.

```
max-clients = 128
```

Set the number of devices a user is able to log in from at the same time. Default is 2. Set to zero for unlimited.

```
max-same-clients = 2
```

By default, keepalive packets are sent every 300 seconds (5 minutes). I prefer to use a short time (30 seconds) to reduce the chance of VPN connection dropout.

```
keepalive = 30
```

Next, find the following line. Change `false` to `true` to enable MTU discovery, which can optimize VPN performance.

```
try-mtu-discovery = false
```

You can set the time that a client is allowed to stay idle before being disconnected via the following two parameters. If you prefer the client to stay connected indefinitely, then comment out these two parameters.

```
idle-timeout=1200
mobile-idle-timeout=1800
```

After that, set the default domain to vpn.example.com.

```
default-domain = vpn.example.com
```

The IPv4 network configuration is as follows by default. This will cause problems because many home routers also set the IPv4 network range to `192.168.1.0/24`.

```
ipv4-network = 192.168.1.0
ipv4-netmask = 255.255.255.0
```

We can use another private IP address range (10.10.10.0/24) to avoid IP address collision, so change the value of `ipv4-network` to

```
ipv4-network = 10.10.10.0
```

Find the following two lines and uncomment them, so VPN clients will be given private IPv6 addresses.

```
ipv6-network = fda9:4efe:7e3b:03ea::/48
ipv6-subnet-prefix = 64
```

If you see the following line

```
ipv6-network = fda9:4efe:7e3b:03ea::/64
```

Please change it to:

```
ipv6-network = fda9:4efe:7e3b:03ea::/48
```

Now uncomment the following line to tunnel all DNS queries via the VPN.

```
tunnel-all-dns = true
```

The default DNS resolver addresses are as follows, which is fine.

```
dns = 8.8.8.8
dns = 1.1.1.1
```

Note: If you are a VPN service provider, then it's a good practice to run your own DNS resolver on the same server. If there's a DNS resolver running on the same server, then specify the DNS as

```
dns = 10.10.10.1
```

10.10.10.1 is the IP address of OpenConnect VPN server in the VPN LAN. This will speed up DNS lookups a little bit for clients because the network latency between the VPN server and the DNS resolver is eliminated.

Then comment out all the route parameters (add # symbol at the beginning of the following lines), which will set the server as the default gateway for the clients.

```
#route = 10.0.0.0/8
#route = 172.16.0.0/12
#route = 192.168.0.0/16
```

```
#route = fd00::/8
#route = default

#no-route = 192.168.5.0/255.255.255.0
```

Save and close the file Then restart the VPN server for the changes to take effect.

```
sudo systemctl restart ocserv
```

Step 5: Create VPN Accounts

Now use the ocpasswd tool to generate VPN accounts.

```
sudo ocpasswd -c /etc/ocserv/ocpasswd username
```

You will be asked to set a password for the user and the information will be saved to `/etc/ocserv/ocpasswd` file. To reset password, simply run the above command again.

Step 6: Enable IP Forwarding

In order for the VPN server to route packets between VPN clients and the Internet, we need to enable IP forwarding by running the following command.

```
echo "net.ipv4.ip_forward = 1" | sudo tee /etc/sysctl.d/60-custo
m.conf
```

Also, run the following two commands to enable TCP BBR algorithm to boost TCP speed.

```
echo "net.core.default_qdisc=fq" | sudo tee -a /etc/sysctl.d/60-
custom.conf

echo "net.ipv4.tcp_congestion_control=bbr" | sudo tee -a /etc/sy
sctl.d/60-custom.conf
```

Then apply the changes with the below command. The **-p** option will load sysctl settings from **/etc/sysctl.d/60-custom.conf** file. This command will preserve our changes across system reboots.

```
sudo sysctl -p /etc/sysctl.d/60-custom.conf
```

Step 7: Configure IP Masquerading in Firewall

We need to set up IP masquerading in the server firewall, so that the server becomes a virtual router for VPN clients. I will use UFW, which is a front end to the iptables firewall. Install UFW on Ubuntu with:

```
sudo apt install ufw
```

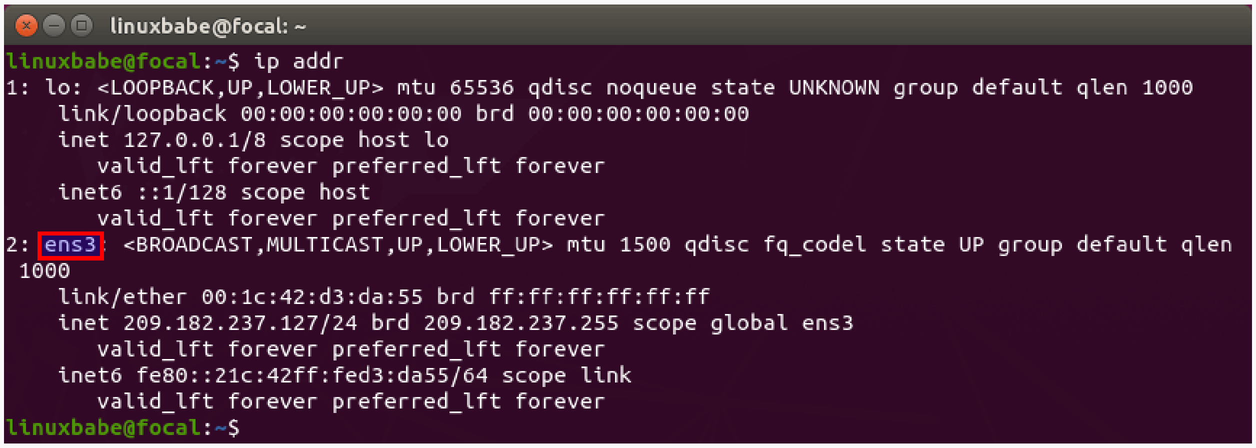
First, you need to allow SSH traffic.

```
sudo ufw allow 22/tcp
```

Then find the name of your server’s main network interface.

```
ip addr
```

As you can see, it’s named `ens3` on my Ubuntu server.



```
linuxbabe@focal: ~
linuxbabe@focal:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
 1000
    link/ether 00:1c:42:d3:da:55 brd ff:ff:ff:ff:ff:ff
    inet 209.182.237.127/24 brd 209.182.237.255 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::21c:42ff:fed3:da55/64 scope link
       valid_lft forever preferred_lft forever
linuxbabe@focal:~$
```

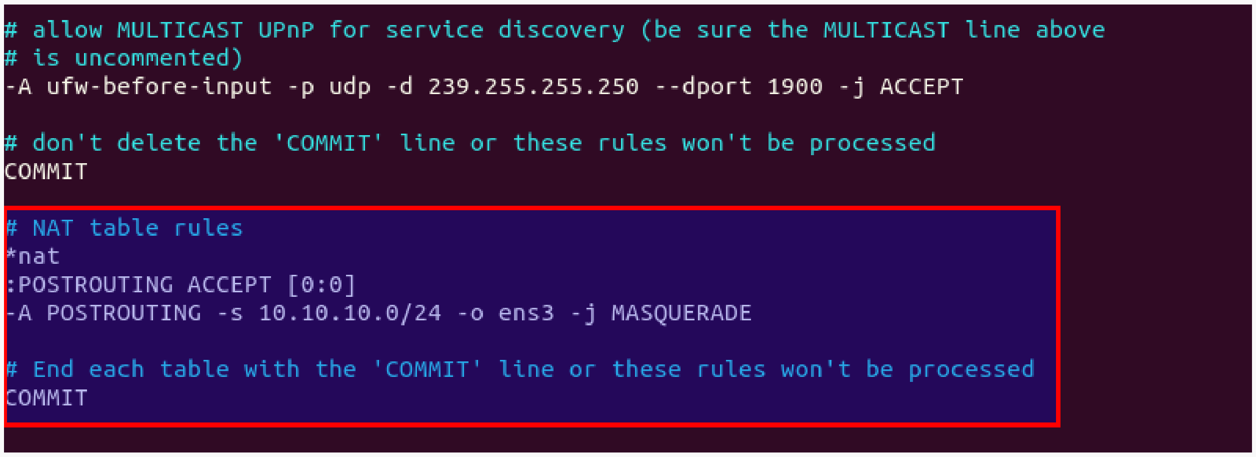
To configure IP masquerading, we have to add iptables command in a UFW configuration file.

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

By default, there are some rules for the `filter` table. Add the following lines at the end of this file. Replace `ens3` with your own network interface name.

```
# NAT table rules
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.10.10.0/24 -o ens3 -j MASQUERADE

# End each table with the 'COMMIT' line or these rules won't be
processed
COMMIT
```

In Nano text editor, you can go to the end of the file by pressing `Ctrl+W`, then pressing `Ctrl+V`.



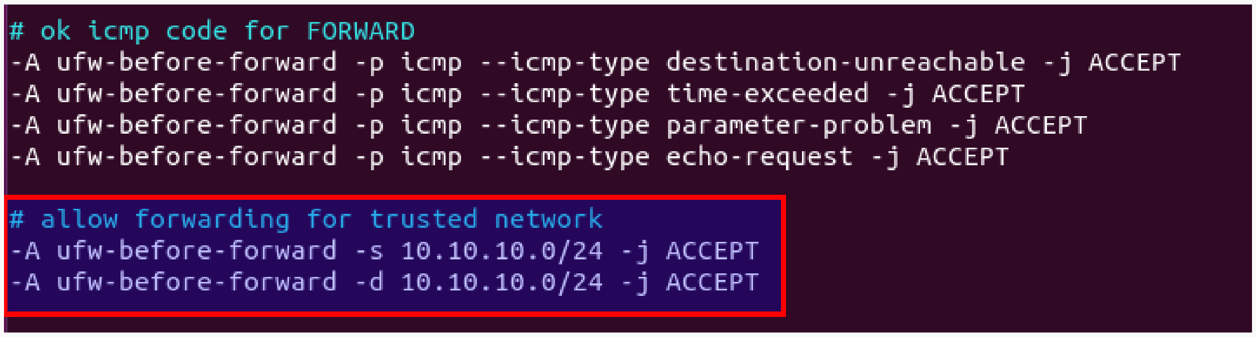
```
# allow MULTICAST UPnP for service discovery (be sure the MULTICAST line above
# is uncommented)
-A ufw-before-input -p udp -d 239.255.255.250 --dport 1900 -j ACCEPT

# don't delete the 'COMMIT' line or these rules won't be processed
COMMIT

# NAT table rules
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.10.10.0/24 -o ens3 -j MASQUERADE

# End each table with the 'COMMIT' line or these rules won't be processed
COMMIT
```

The above lines will append (**-A**) a rule to the end of of **POSTROUTING** chain of **nat** table. It will link your virtual private network with the Internet. And also hide your network from the outside world. So the Internet can only see your VPN server's IP, but can't see your VPN client's IP, just like your home router hides your private home network.

By default, UFW forbids packet forwarding. We can allow forwarding for our private network. Find the `ufw-before-forward` chain in this file and add the following 3 lines, which will accept packet forwarding if the source IP or destination IP is in the `10.10.10.0/24` range.

```
# allow forwarding for trusted network
```

```
-A ufw-before-forward -s 10.10.10.0/24 -j ACCEPT
-A ufw-before-forward -d 10.10.10.0/24 -j ACCEPT
```



```
# ok icmp code for FORWARD
-A ufw-before-forward -p icmp --icmp-type destination-unreachable -j ACCEPT
-A ufw-before-forward -p icmp --icmp-type time-exceeded -j ACCEPT
-A ufw-before-forward -p icmp --icmp-type parameter-problem -j ACCEPT
-A ufw-before-forward -p icmp --icmp-type echo-request -j ACCEPT

# allow forwarding for trusted network
-A ufw-before-forward -s 10.10.10.0/24 -j ACCEPT
-A ufw-before-forward -d 10.10.10.0/24 -j ACCEPT
```

Save and close the file. Then enable UFW.

```
sudo ufw enable
```

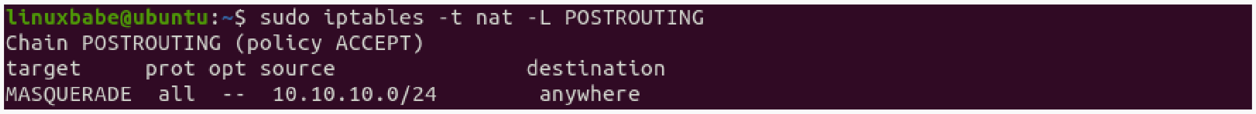
If you have enabled UFW before, then you can use systemctl to restart UFW.

```
sudo systemctl restart ufw
```

Now if you list the rules in the POSTROUTING chain of the NAT table by using the following command:

```
sudo iptables -t nat -L POSTROUTING
```

You can see the Masquerade rule.



```
linuxbabe@ubuntu:~$ sudo iptables -t nat -L POSTROUTING
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  10.10.10.0/24        anywhere
```

It can take some time for UFW to process the firewall rules. If the masquerade rule doesn't show up, then restart UFW again (`sudo systemctl restart ufw`).

Step 8: Open Port 443 in Firewall

Run the following command to open TCP and UDP port 443. If you configured a different port for ocserv, then change 443 to your configured port.

```
sudo ufw allow 443/tcp
sudo ufw allow 443/udp
```

Now OpenConnect VPN server is ready to accept client connections.

If you run a local DNS Resolver

For those of you who run a local DNS resolver, if you specified 10.10.10.1 as the DNS server for VPN clients, then you must allow VPN clients to connect to port 53 with the following UFW rule.

```
sudo ufw insert 1 allow in from 10.10.10.0/24
```

You also need to edit the BIND DNS server's configuration file (`/etc/bind/named.conf.options`) to allow VPN clients to send recursive DNS queries like below.

```
allow-recursion { 127.0.0.1; 10.10.10.0/24; };
```

Then restart BIND.

```
sudo systemctl restart named
```

How to Install and Use OpenConnect VPN client on Ubuntu 20.04 Desktop

Run the following command to install OpenConnect VPN command line client on Ubuntu desktop.

```
sudo apt install openconnect
```

You can Connect to VPN from the command line like below. `-b` flag will make it run in the background after connection is established.

```
sudo openconnect -b vpn.example.com:port-number
```

You will be asked to enter VPN username and password. If the connection is successfully established, you will see the following message.

```
Got CONNECT response: HTTP/1.1 200 CONNECTED
CSTP connected. DPD 90, Keepalive 32400
Connected tun0 as 192.168.1.139, using SSL
```

```
Established DTLS connection (using GnuTLS). Ciphersuite (DTLS1.2
)-(RSA)-(AES-256-GCM).
```

To stop the connection, run:

```
sudo pkill openconnect
```

To run the client non-interactively, use the following syntax.

```
echo -n password | sudo openconnect -b vpn.example.com -u userna
me --passwd-on-stdin
```

If you want to use Network Manager to manage VPN connection, then you also need to install these packages.

```
sudo apt install network-manager-openconnect network-manager-ope
nconnect-gnome
```

If you are successfully connected to the VPN server, but your public IP address doesn’t change, that’s because **IP forwarding** or **IP masquerading** is not working. I once had a typo in my iptables command (using a wrong IP address range), which caused my computer not being able to browse the Internet.

If you encounter the following error, then you should disable the UDP port in ocserv, which is explained later in the **speed optimization** section.

```
DTLS handshake failed: Resource temporarily unavailable, try aga
in
```

If you have the following error, it’s likely that your VPN username or password is wrong.

```
fgets (stdin): Inappropriate ioctl for device
```

Auto-Connect on System Startup

To let OpenConnect VPN client automatically connect to the server at boot time, we can create a systemd service unit.

```
sudo nano /etc/systemd/system/openconnect.service
```

Put the following lines to the file. Replace the red text.

```
[Unit]
  Description=OpenConnect VPN Client
  After=network-online.target systemd-resolved.service
  Wants=network-online.target

[Service]
  Type=simple
  ExecStart=/bin/bash -c '/bin/echo -n password | /usr/sbin/open
connect vpn.example.com -u username --passwd-on-stdin'
  KillSignal=SIGINT
  Restart=always
  RestartSec=2

[Install]
  WantedBy=multi-user.target
```

Save and close the file. Then enable this service so that it will start at boot time.

```
sudo systemctl enable openconnect.service
```

Explanation of the file content:

- `After=network-online.target systemd-resolved.service` and `Wants=network-online.target` make this service run after network is up. We want the `openconnect.service` start after the `systemd-resolved.service` because that will ensure the DNS server address set by OpenConnect won't be overridden by `systemd-resolved.service`.
- In reality, this service can still run before network is up. We add `Restart=always` and `RestartSec=2` to restart this service after 2 seconds if this service fails.
- Systemd doesn't recognise pipe redirection, so in the `ExecStart` directive, we wrap the comand in single quotes and run it with the Bash shell.

- Since OpenConnect VPN client will run as a systemd service, which runs in the background, there's no need to add `-b` flag to the `openconnect` command.
- The `KillSignal` directive tells Systemd to send the `SIGINT` signal when the `systemctl stop openconnect` command is issued. This will performs a clean shutdown by logging the session off, and restoring DNS server settings and the Linux kernel routing table.

To start this Systemd service immediately, run

```
sudo systemctl start openconnect
```

To stop this Systemd service, run

```
sudo systemctl stop openconnect
```

How to Automatically Restart OpenConnect Client When Resuming from Suspend

If your Ubuntu desktop goes into suspend state, the OpenConnect client would lose connection to the VPN server. To make it automatically restart when resuming from suspend, we need to create another systemd service unit.

```
sudo nano /etc/systemd/system/openconnect-restart.service
```

Add the following lines in the file.

```
[Unit]
Description=Restart OpenConnect client when resuming from suspen
d
After=suspend.target

[Service]
Type=simple
ExecStart=/bin/systemctl --no-block restart openconnect.service

[Install]
WantedBy=suspend.target
```

Save and close the file. Then enable this service.

```
sudo systemctl enable openconnect-restart.service
```

Automatic-Restart When VPN Connection Drops

Sometimes the VPN connection would drop due to other reasons. You can run the following command to check if the VPN client can ping the VPN server's private IP address (10.10.10.1). If the ping is unsuccessful, then the command on the right will be executed to restart the VPN client. `||` is the OR operator in Bash. It executes the command on the right only if the command on the left returned an error.

```
ping -c9 10.10.10.1 || systemctl restart openconnect
```

The ping will be done 9 times, i.e 9 seconds. You can use an **infinite loop** in the Bash shell to make the whole command run forever. Press `Ctrl+C` to stop it.

```
for ((; ; )) do (ping -c9 10.10.10.1 || systemctl restart openco
nnect) done
```

Now we can create a systemd service for this task.

```
sudo nano /etc/systemd/system/openconnect-check.service
```

Add the following lines to this file. We specify that this service should run after the `openconnect.service`.

```
[Unit]
Description=OpenConnect VPN Connectivity Checker
After=openconnect.service

[Service]
Type=simple
ExecStart=/bin/bash -c 'for ((; ; )) do (ping -c9 10.10.10.1 ||
systemctl restart openconnect) done'

[Install]
WantedBy=multi-user.target
```

Save and close the file. Then start this service.

```
sudo systemctl start openconnect-check
```

Enable auto-start at boot time.

```
sudo systemctl enable openconnect-check
```

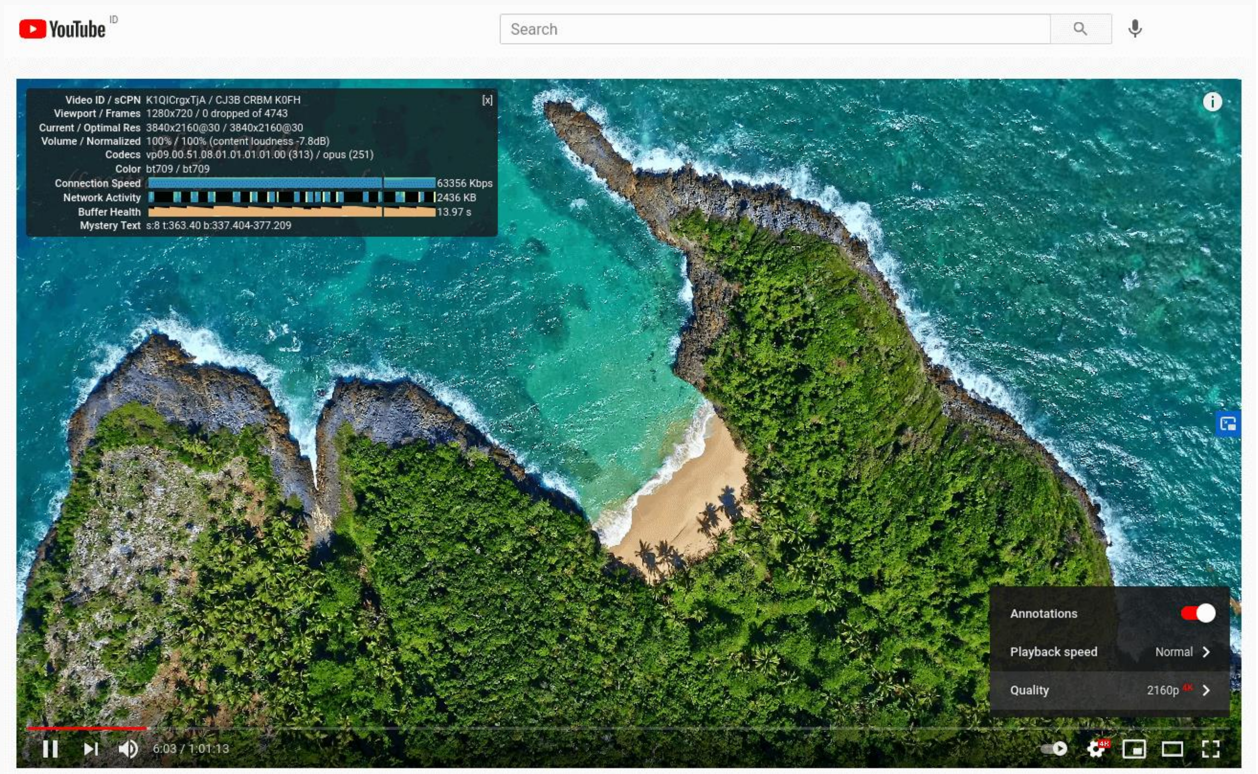
Once this service is started, the ping command will run forever. If the VPN connection drops, it will automatically restart `openconnect.service`.

OpenConnect GUI Client for Windows and macOS

They can be downloaded from OpenConnect GUI Github Page.

Speed

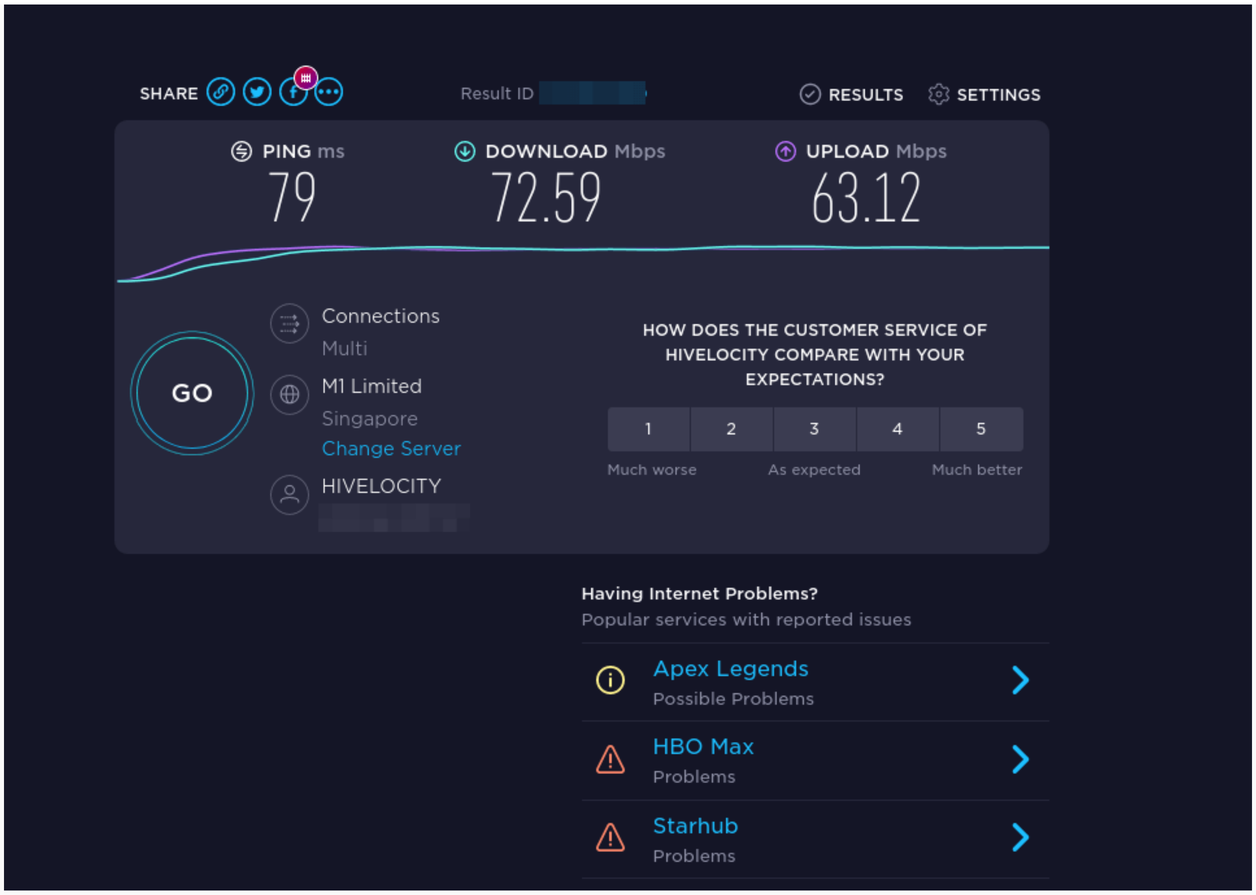
OpenConnect VPN is pretty fast. I can use it to watch 4k videos on YouTube. As you can see, my connection speed is **63356 Kbps**, which translates to **61 Mbit/s**.



```
    Video ID / sCPN K1QICrgxTjA / CJ3B CRBM K0FH
  Viewport / Frames 1280x720 / 0 dropped of 4743
Current / Optimal Res 3840x2160@30 / 3840x2160@30
Volume / Normalized 100% / 100% (content loudness -7.8dB)
             Codecs vp09.00.51.08.01.01.01.01.00 (313) / opus (251)
              Color bt709 / bt709
   Connection Speed 63356 Kbps
   Network Activity 2436 KB
      Buffer Health 13.97 s
       Mystery Text s:8 t:363.40 b:337.404-377.209
```

4K Video (Ultra HD) Unbelievable Beauty

And here's the test results on speedtest.net.



Speed Optimization

OpenConnect by default uses TLS over UDP protocol (DTLS) to achieve faster speed, but UDP can't provide reliable transmission. TCP is slower than UDP but can provide reliable transmission. One optimization tip I can give you is to disable DTLS, use standard TLS (over TCP), then enable TCP BBR to boost TCP speed.

To disable DTLS, comment out (add `#` symbol at the beginning) the following line in ocserv configuration file.

```
udp-port = 443
```

Save and close the file. Then restart ocserv service.

```
sudo systemctl restart ocserv.service
```

To enable TCP BBR, please check out the following tutorial. Note that you need to disable DTLS in ocserv, or TCP BBR won't work.

- How to Easily boost Ubuntu Network Performance by enabling TCP BBR

In my test, standard TLS with TCP BBR enabled is two times faster than DTLS.

Another very important factor affecting speed is how good the connection between your local computer and the VPN server is. If you live in the middle east and the VPN server is located in the U.S, the speed would be slow. Choose a data center that's close to where you live.

Also, check your CPU load average. (`htop` can be installed by `sudo apt install htop`).

```
htop
```

Make sure the CPU load average is under `1`. I once had a CPU load average of `3`, which caused a high latency between the VPN client and VPN server.

Auto-Renew Let's Encrypt Certificate

Edit root user's crontab file.

```
sudo crontab -e
```

Add the following line at the end of the file. It's necessary to reload ocserv service for the VPN server to pick up new certificate and key file.

```
@daily certbot renew --quiet && systemctl reload ocserv
```

Troubleshooting Tips

OpenVZ

Note that if you are using OpenVZ VPS, make sure you enable the TUN virtual networking device in VPS control panel. (If you use Kamtera VPS, then you have KVM-based VPS, so you don't have to worry about this.)

Log File

If you encounter any problems, then check OpenConnect VPN server log.

```
sudo journalctl -eu ocserv.service
```

I found that if I change port 443 to a different port, the great firewall of China will block this VPN connection.

Debugging Mode

If ocserv tells you that it can't load the `/etc/ocserv/ocserv.conf` file, you can stop ocserv.

```
sudo systemctl stop ocserv
```

Then run it in the foreground with debugging enabled.

```
sudo /usr/sbin/ocserv --foreground --pid-file /run/ocserv.pid --config /etc/ocserv/ocserv.conf --debug=10
```

Then output might give you some clues why ocserv isn't working.

Can't browse the Internet

If you are successfully connected to the VPN server, but you can't browse the Internet, that's because IP forwarding or IP masquerading is not working. I remember my VPS provider once did a platform upgrade, which changed the name of the main network interface from `ens3` to `enp3s0`, so I had to update the name in the UFW file (`/etc/ufw/before.rules`).

Syntax Error

If you see the following error when trying to establish VPN connection, it's probably because there's a syntax error in your ocserv config file. Check the journal (`sudo journalctl -eu ocserv`) to find out.

```
Got inappropriate HTTP CONNECT response: HTTP/1.1 401 Cookie is not acceptable
```

Restart Your Computer

If you see the following error when trying to establish VPN connection, it's likely a local computer problem. Try restarting your computer.

```
Server 'vpn.your-domain.com' requested Basic authentication whic
h is disabled by default
```

TLS connection was non-properly terminated

If you see the following error on the client when trying to establish a VPN connection,

```
SSL connection failure: The TLS connection was non-properly term
inated.
```

you probably should restart the `ocserv` service on the VPN server.

```
sudo systemctl restart ocserv
```

You can create a cron job to automatically restart `ocserv` once per day at 4 AM.

```
sudo crontab -e
```

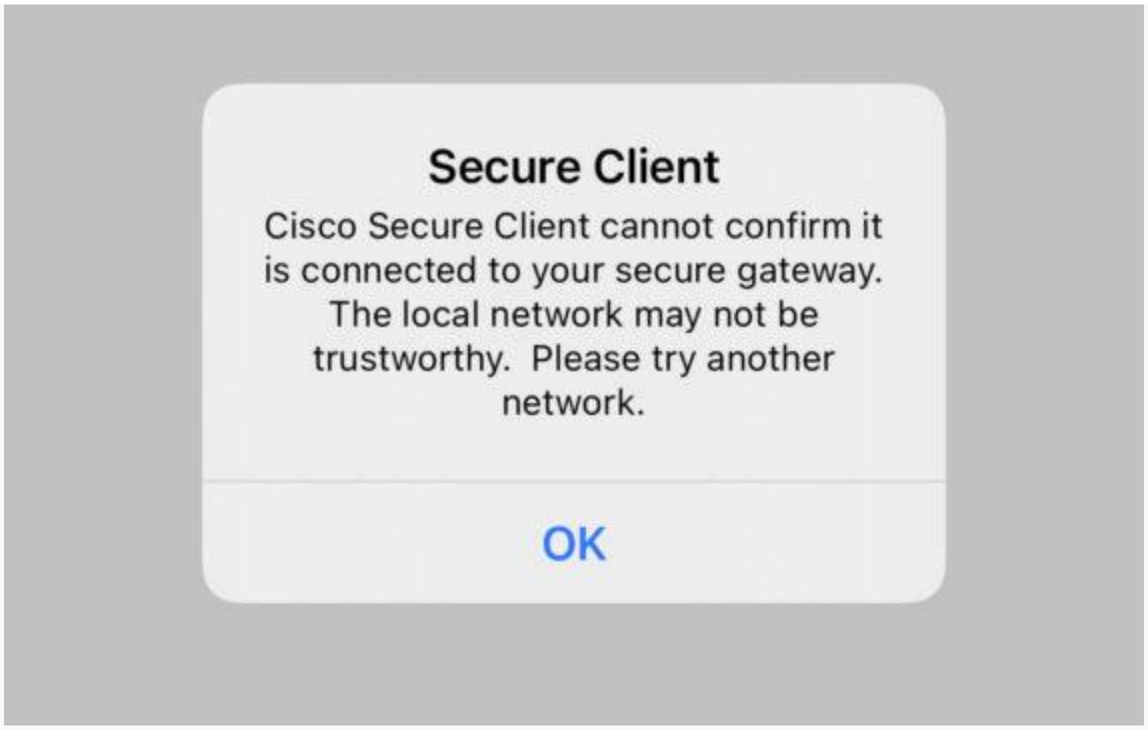
Add the following line.

```
0 4 * * * systemctl restart ocserv
```

Save and close the file.

The local network may not be trustworthy

If you use the Cisco AnyConnect VPN client on iOS, and you encounter the following error, it’s likely your TLS certificate has expired.


Secure Client

Cisco Secure Client cannot confirm it is connected to your secure gateway. The local network may not be trustworthy. Please try another network.

If the TLS certificate has expired, you will also see the following error when trying to establish a VPN connection on a Linux desktop.

```
Server certificate verify failed: certificate expired
Certificate from VPN server "vpn.your-domain.com" failed verific
ation.
Reason: certificate expired
To trust this server in future, perhaps add this to your command
line:
     --servercert pin-sha256:er1Kv/37ZxHpN6VESmYVS7vw4wXEB1oYELw
iBS2wcvc=
Enter 'yes' to accept, 'no' to abort; anything else to view: fge
ts (stdin): Operation now in progress
```

You will need to run the following command to renew TLS certificate.

```
sudo certbot renew --quiet
```

Don’t forget to restart ocserv.

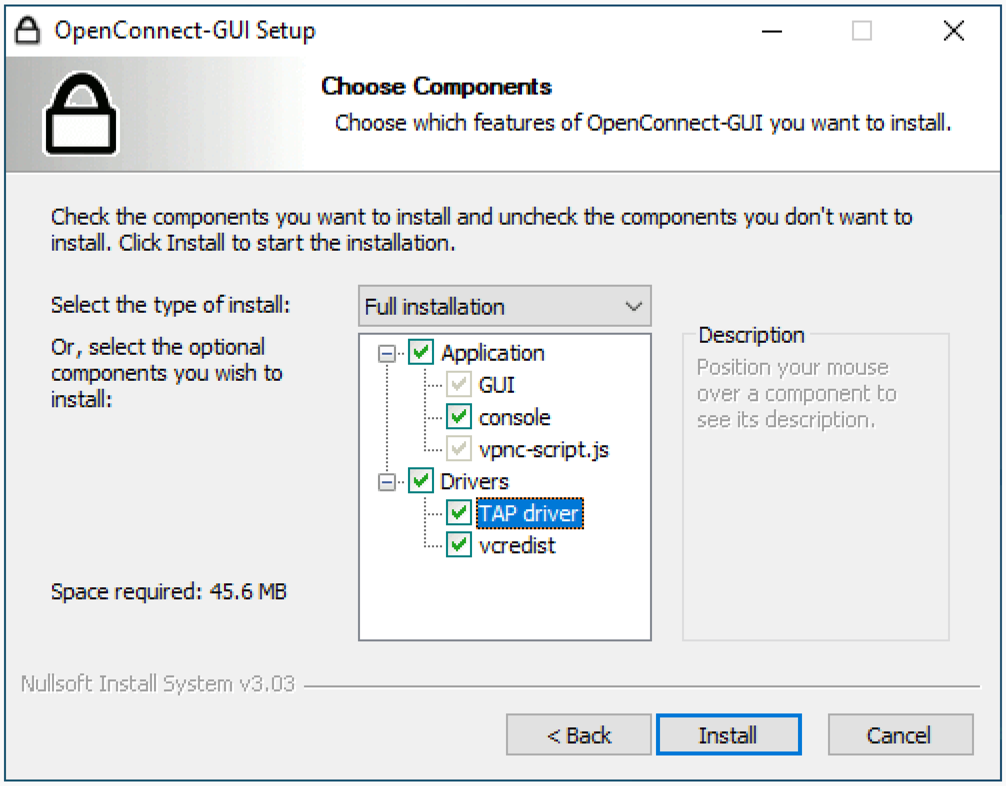
```
sudo systemctl restart ocserv
```

If this error still persists after renewing the certificate, be sure to check the `server-cert` and `server-key` parameter in the `/etc/ocserv/ocserv.conf` file. Maybe it's pointed to the wrong file.

No Tap-Windows Adapter installed on this system

If you use the OpenConnect GUI Windows client and found the "*No Tap-Windows Adapter installed on this system*" error in the logs (View – Log Window), this is likely because you have install other VPN clients afterward like OpenVPN.

To fix this error, you need to uninstall OpenConnect GUI client and reinstall it. In the setup wizard, you will have the option to install the TAP driver.


OpenConnect-GUI Setup
Choose Components
Choose which features of OpenConnect-GUI you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Install to start the installation.
Select the type of install:
Full installation
Or, select the optional components you wish to install:
Application
GUI
console
vpnc-script.js
Drivers
TAP driver
vcredist
Description
Position your mouse over a component to see its description.
Space required: 45.6 MB
Nullsoft Install System v3.03
< Back
Install
Cancel

How to Install the Latest Version of ocserv

Check your current `ocserv` version.

```
ocserv -v
```

Sometimes, the latest version of `ocserv` will fix an issue. You may also want to use a new feature that's only available in the latest release. Follow the instructions below to install the latest `ocserv` version.

Install build dependency packages.

```
sudo apt install -y git ruby-ronn libbsd-dev libsystemd-dev libp
cl-dev libwrap0-dev libgnutls28-dev libev-dev libpam0g-dev liblz
4-dev libseccomp-dev libreadline-dev libnl-route-3-dev libkrb5-d
ev libradcli-dev libcurl4-gnutls-dev libcjose-dev libjansson-dev
libprotobuf-c-dev libtalloc-dev libhttp-parser-dev protobuf-c-co
mpiler gperf nuttcp lcov libuid-wrapper libpam-wrapper libnss-wr
apper libsocket-wrapper gss-ntlmssp haproxy iputils-ping freerad
ius gawk gnutls-bin iproute2 yajl-tools tcpdump
```

Clone the ocserv Git repository.

```
git clone https://gitlab.com/openconnect/ocserv.git
```

Generate configuration scripts.

```
cd ocserv

autoreconf -fvi
```

Compile the source code. If you see deprecated warnings, you can ignore them.

```
./configure && make
```

Install the binaries.

```
sudo make install
```

The files will be install to `/usr/loca/bin/` and `/usr/local/sbin/`. Next, we need to copy the systemd service file.

```
sudo cp /lib/systemd/system/ocserv.service /etc/systemd/system/o
cserv.service
```

Edit this file.

```
sudo nano /etc/systemd/system/ocserv.service
```

Because the compiled version of ocserv binary is located at `/usr/local/sbin/ocserv`, we need to change

```
ExecStart=/usr/sbin/ocserv --foreground --pid-file /run/ocserv.p
id --config /etc/ocserv/ocserv.conf
```

to

```
ExecStart=/usr/local/sbin/ocserv --foreground --pid-file /run/oc
serv.pid --config /etc/ocserv/ocserv.conf
```

Save and close the file. Then reload systemd.

```
sudo systemctl daemon-reload
```

Restart ocserv service.

```
sudo systemctl restart ocserv
```

Make OpenConnect VPN server and web server use port 443 at the same time