Set Up Your Own WireGuard VPN Server on Ubuntu

This tutorial is going to show you how to set up your own WireGuard VPN server on Ubuntu. WireGuard is made specifically for the Linux kernel. It runs inside the Linux kernel and allows you to create fast, modern, and secure VPN tunnel.

Features of WireGuard VPN

- Lightweight and super fast speed, blowing OpenVPN out of the water.
- Cross-platform. WireGuard can run on Linux, BSD, macOS, Windows, Android, iOS, and OpenWRT.
- User authentication is done by exchanging public keys, similar to SSH keys.
- It assigns static tunnel IP addresses to VPN clients. Some folks may not like it, but it can be useful in some cases.
- Mobile devices can switch between Wi-Fi and mobile network seamlessly without dropping any connectivity.
- It aims to replace OpenVPN and IPSec in most use cases.

Prerequisites

This tutorial assumes that the VPN server and VPN client are both running **Ubuntu** operating system.

Step 1: Install WireGuard on Ubuntu Server and Desktop

Log into your Ubuntu server, then run the following commands to install WireGuard.

Ubuntu 20.04

Ubuntu 20.04 ships with Linux kernel 5.4, which has a built-in wireguard module.

```
sudo apt update
sudo apt install wireguard wireguard-tools
```

Ubuntu 18.04

Ubuntu 18.04 ships with Linux kernel 4.15, so users needs to install the hardware-enablement kernel first (HWE), which will install kernel 5.4 on your system.

```
sudo apt update
sudo apt install linux-generic-hwe-18.04-edge
```

Restart your Ubuntu 18.04 server and install WireGuard.

```
sudo shutdown -r now
sudo apt install wireguard wireguard-tools wireguard-dkms
```

Then use the same commands to install WireGuard on your local Ubuntu computer (the VPN client). Note that you also need to install the `openresolv` package on the client to configure DNS server.

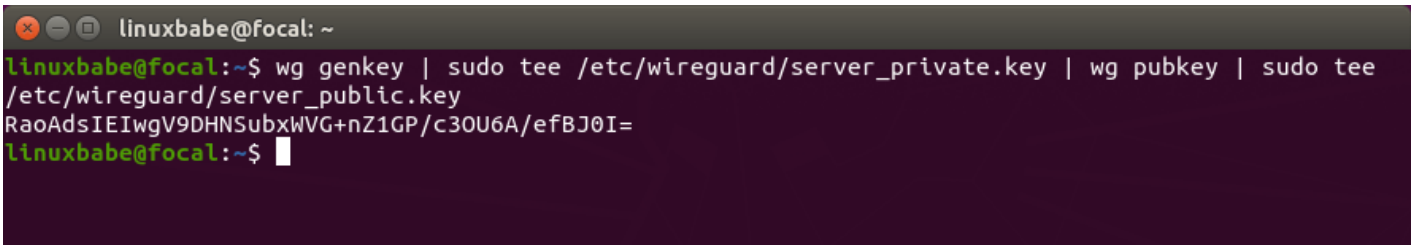
```
sudo apt install openresolv
```

Step 2: Generate Public/Private Keypair

Server

Run the following command on the Ubuntu server to create a public/private key pair, which will be saved under `/etc/wireguard/` directory.

```
wg genkey | sudo tee /etc/wireguard/server_private.key | wg pubkey | sudo tee
/etc/wireguard/server_public.key
```



Client

Run the following command to create a public/private key pair on the local Ubuntu computer (the VPN client).

```
wg genkey | sudo tee /etc/wireguard/client_private.key | wg pubkey | sudo tee
/etc/wireguard/client_public.key
```

Step 3: Create WireGuard Configuration File

Server

Use a command-line text editor like Nano to create a WireGuard configuration file on the Ubuntu server. `wg0` will be the network interface name.

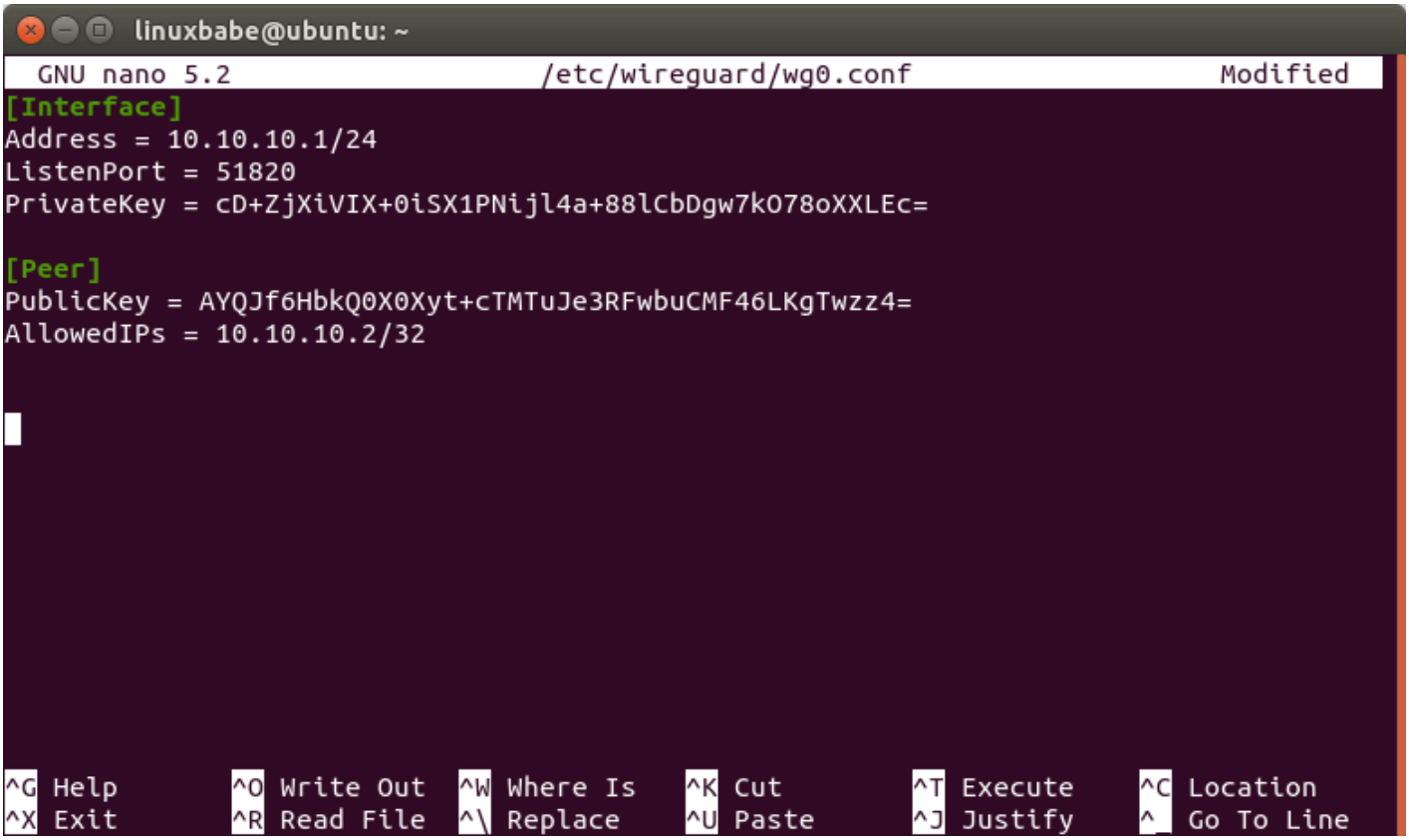
```
sudo nano /etc/wireguard/wg0.conf
```

Copy the following text and paste it to your configuration file. You need to use your own server private key and client public key.

```
[Interface]
Address = 10.10.10.1/24
ListenPort = 51820
PrivateKey = cD+ZjXiVIX+0iSX1PNijl4a+88lCbDgw7kO78oXXLEc=

[Peer]
```

```
PublicKey = AYQJf6HbkQ0X0Xyt+cTMTuJe3RFwbuCMF46LKgTwzz4=
AllowedIPs = 10.10.10.2/32
```



Where:

- **Address**: Specify the private IP address of the VPN server. Here I'm using the 10.10.10.0/24 network range, so it won't conflict with your home network range. (Most home routers use 192.168.0.0/24 or 192.168.1.0/24). 10.10.10.1 is the private IP address for the VPN server.
- **PrivateKey**: The private key of VPN server, which can be found in the `/etc/wireguard/server_private.key` file on the server.
- **ListenPort**: WireGuard VPN server will be listening on UDP port 51820, which is the default.
- **PublicKey**: The public key of VPN client, which can be found in the `/etc/wireguard/client_public.key` file on the client computer.
- **AllowedIPs**: IP addresses the VPN client is allowed to use. In this example, the client can only use the 10.10.10.2 IP address inside the VPN tunnel.

Save and close the file. (To save a file in Nano text editor, press `Ctrl+O`, then press Enter to confirm. Press `Ctrl+X` to exit.)

Change the file permission mode so that only root user can read the files.

```
sudo chmod 600 /etc/wireguard/ -R
```

Client

Use a command-line text editor like Nano to create a WireGuard configuration file on your local Ubuntu computer. `wg-client0` will be the network interface name.

```
sudo nano /etc/wireguard/wg-client0.conf
```

Copy the following text and paste it to your configuration file. You need to use your own client private key and server public key.

```
[Interface]
Address = 10.10.10.2/24
DNS = 10.10.10.1
PrivateKey = cOFA+x5UvHF+a3xJ6enLatG+DoE3I5PhMgKrMKkUyXI=

[Peer]
PublicKey = RaoAdsIEIwgV9DHNSubxWVG+nZ1GP/c3OU6A/efBJ0I=
AllowedIPs = 0.0.0.0/0
Endpoint = 12.34.56.78:51820
PersistentKeepalive = 25
```

Where:

- **Address**: Specify the private IP address of the VPN client.
- **DNS**: specify 10.10.10.1 (VPN server) as the DNS server. It will be configured via the `resolvconf` command. You can also specify multiple DNS servers for redundancy like this: `DNS = 10.10.10.1 8.8.8.8`
- **PrivateKey**: The client's private key, which can be found in the `/etc/wireguard/client_private.key` file on the client computer.
- **PublicKey**: The server's public key, which can be found in the `/etc/wireguard/server_public.key` file on the server.
- **AllowedIPs**: 0.0.0.0/0 represents the whole Internet, which means all traffic to the Internet should be routed via the VPN.
- **Endpoint**: The public IP address and port number of VPN server. Replace 12.34.56.78 with your server's real public IP address.
- **PersistentKeepalive**: Send an authenticated empty packet to the peer every 25 seconds to keep the connection alive. If PersistentKeepalive isn't enabled, the VPN server might not be able to ping the VPN client.

Save and close the file.

Change the file mode so that only root user can read the files.

```
sudo chmod 600 /etc/wireguard/ -R
```

Step 4: Enable IP Forwarding on the Server

In order for the VPN server to route packets between VPN clients and the Internet, we need to enable IP forwarding. Edit `sysctl.conf` file.

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

Add the following line at the end of this file.

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

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

```
sudo sysctl -p
```

Step 5: Configure IP Masquerading on the Server

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
```

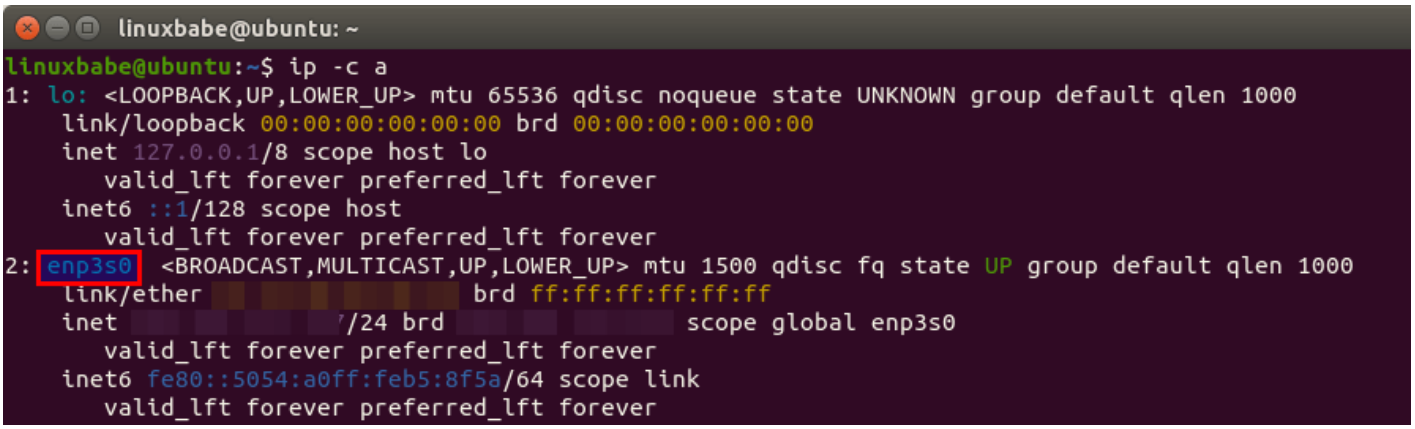
First, you need to allow SSH traffic.

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

Next, find the name of your server's main network interface.

```
ip -c a
```

As you can see, it's named `enp3s0` on my Ubuntu server.



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 `enp3s0` with your own network interface name.

```
# NAT table rules
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o enp3s0 -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 mDNS for service discovery (be sure the MULTICAST line above
# is uncommented)
-A ufw-before-input -p udp -d 224.0.0.251 --dport 5353 -j ACCEPT

# allow MULTICAST UPnP for service discovery (be sure the MULTICAST line above
# is uncommented)
-A ufw-before-input -p udp -d [redacted] --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 -o enp3s0 -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
```

Step 6: Install a DNS Resolver on the Server

Since we specify the VPN server as the DNS server for client, we need to run a DNS resolver on the VPN server. We can install the bind9 DNS server.

```
sudo apt install bind9
```

Once it's installed, BIND will automatically start. You can check its status with:

```
systemctl status bind9
```

Sample output:

```
● named.service - BIND Domain Name Server
     Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2020-05-17 08:11:26 UTC; 37s ago
       Docs: man:named(8)
   Main PID: 13820 (named)
      Tasks: 5 (limit: 1074)
     Memory: 14.3M
     CGroup: /system.slice/named.service
             └─13820 /usr/sbin/named -f -u bind
```

If it's not running, start it with:

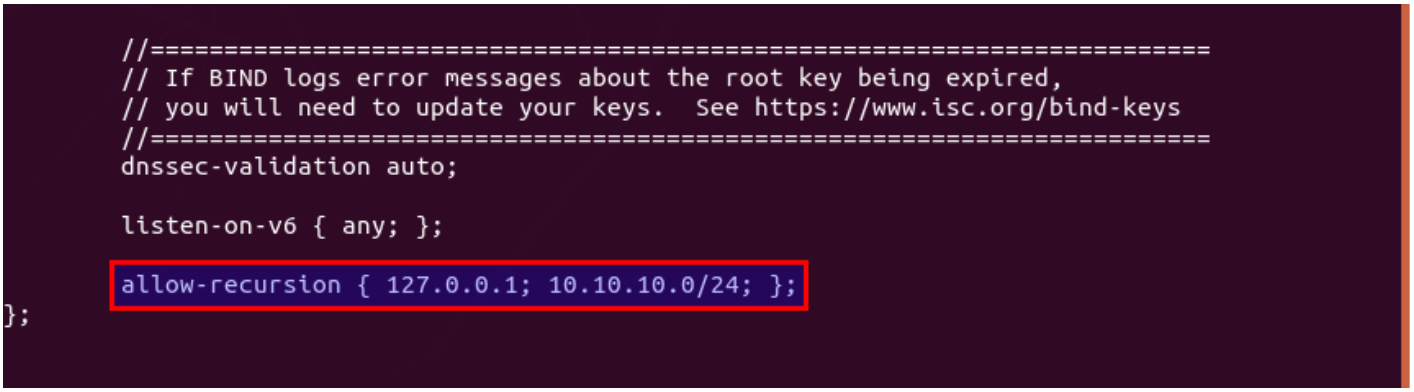
```
sudo systemctl start bind9
```

Edit the BIND DNS server's configuration file.

```
sudo nano /etc/bind/named.conf.options
```

Add the following line to allow VPN clients to send recursive DNS queries.

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



Save and close the file. Restart BIND9 for the changes to take effect.

```
sudo systemctl restart bind9
```

Then you need to run the following command to allow VPN clients to connect to port 53.

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

Step 7: Open WireGuard Port in Firewall

Run the following command to open UDP port 51820 on the server.

```
sudo ufw allow 51820/udp
```

Step 8: Start WireGuard

server

Run the following command on the server to start WireGuard.

```
sudo wg-quick up /etc/wireguard/wg0.conf
```

To stop it, run

```
sudo wg-quick down /etc/wireguard/wg0.conf
```

You can also use systemd service to start WireGuard.

```
sudo systemctl start wg-quick@wg0.service
```

Enable auto-start at system boot time.

```
sudo systemctl enable wg-quick@wg0.service
```

Check its status with the following command. Its status should be `active (exited)`.

```
systemctl status wg-quick@wg0.service
```

Now WireGuard server is ready to accept client connections.

Client

Start WireGuard.

```
sudo systemctl start wg-quick@wg-client0.service
```

Enable auto-start at system boot time.

```
sudo systemctl enable wg-quick@wg-client0.service
```

Check its status:

```
systemctl status wg-quick@wg-client0.service
```

Now go to this website: https://icanhazip.com/ to check your public IP address. If everything went well, it should display your VPN server's public IP address instead of your client computer's public

IP address.

You can also run the following command to get the current public IP address.

```
curl https://icanhazip.com
```

Troubleshooting Tips

Can’t ping

You can ping from the VPN server to VPN client (`ping 10.10.10.2`) to see if the tunnel works. If you see the following error message in the ping,

```
ping: sendmsg: Required key not available
```

it might be that the `AllowedIPs` parameter is wrong, like a typo. After fixing the typo, restart both the VPN server and VPN client.

Public IP Doesn’t Change

If the VPN tunnel is successfully established, but the client public IP address doesn’t change, that’s because the masquerading or forwarding rule in your UFW config file is not working. I once had a typo in the `/etc/ufw/before.rules` file, which caused my computer not to be able to browse the Internet.

Note that I don’t recommend using `SaveConfig=true` in the `[Interface]` section of the WireGuard configuration file. `SaveConfig` tells WireGuard to save the runtime configuration on shutdown. So if you add additional `[Peer]` in the configuration file and then restart WireGuard, your newly-added configs will be overwritten.

Enable Debug logging in Linux Kernel

If you use Linux kernel 5.6+, you can enable debug logging for WireGuard with the following command.

```
sudo su -
echo module wireguard +p > /sys/kernel/debug/dynamic_debug/control
```

Then you can view the debug logs with

```
sudo dmesg -wH
```

or

```
sudo journalctl -kf
```

Restart

If your VPN still doesn’t work, try restarting the VPN server.

```
sudo systemctl restart wg-quick@wg0.service
```

Then stop the VPN client.

```
sudo systemctl stop wg-quick@wg-client0.service
```

And upgrade software packages on the VPN client.

```
sudo apt update; sudo apt upgrade
```

Next, reboot the VPN client.

```
sudo shutdown -r now


sudo systemctl start wg-quick@wg-client0.service
```

If your WireGuard VPN can only work after a restart, consider adding a cron job to automatically restart the service.

```
sudo crontab -e
```

Add the following line in this file.

```
@daily systemctl restart wg-quick@wg0.service
```

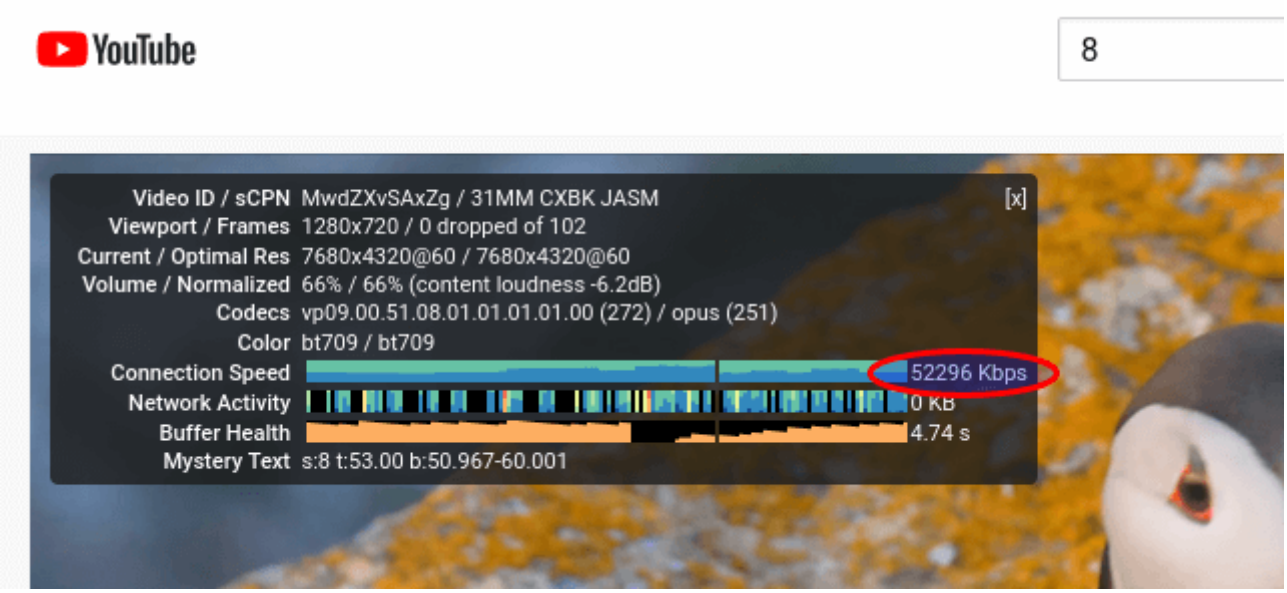
Speed Comparison between

WireGuard & OpenConnect

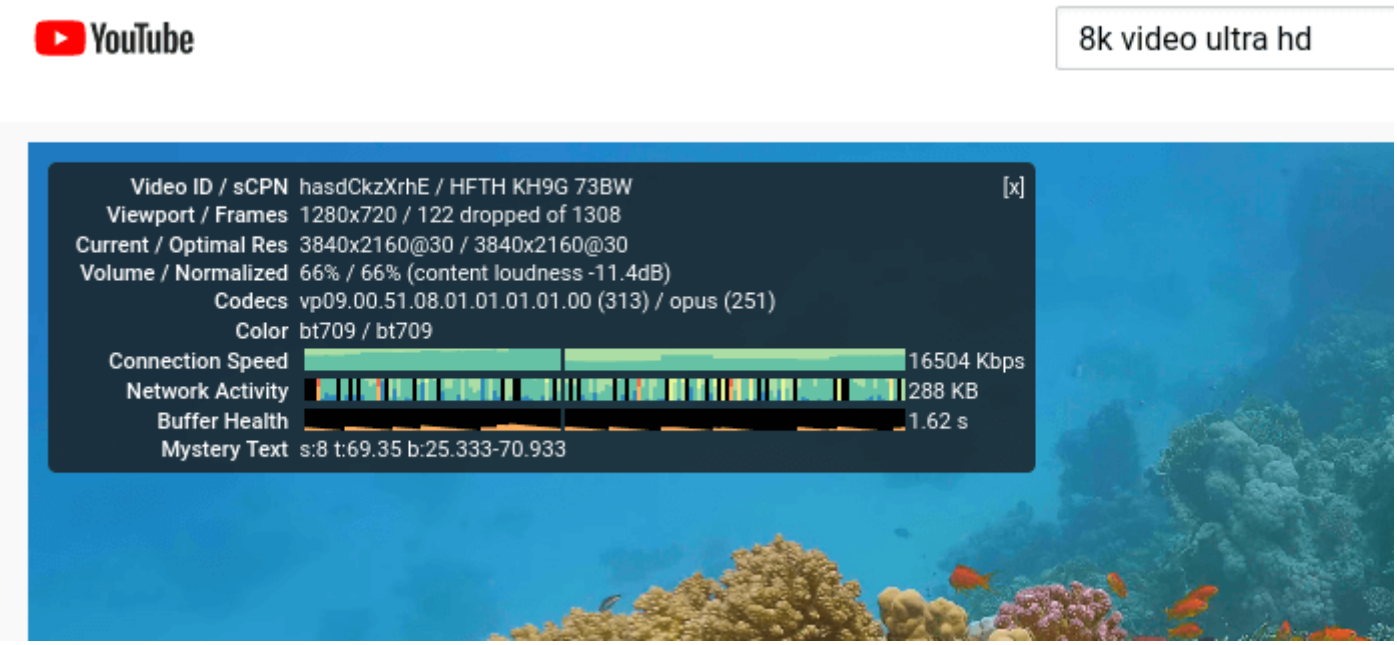
On one of my VPS servers, I installed both WireGuard and OpenConnect VPN server. The speed test is as follows. It might not look like fast to you, because the connection between my computer and the VPN server is very poor. How fast you can get depends on the latency and packet loss rate between the VPN client and VPN server.

- WireGuard is the winner. It's nearly 3 times faster than OpenConnect.
- OpenConnect over TCP is faster than OpenConnect over UDP. Surprise?

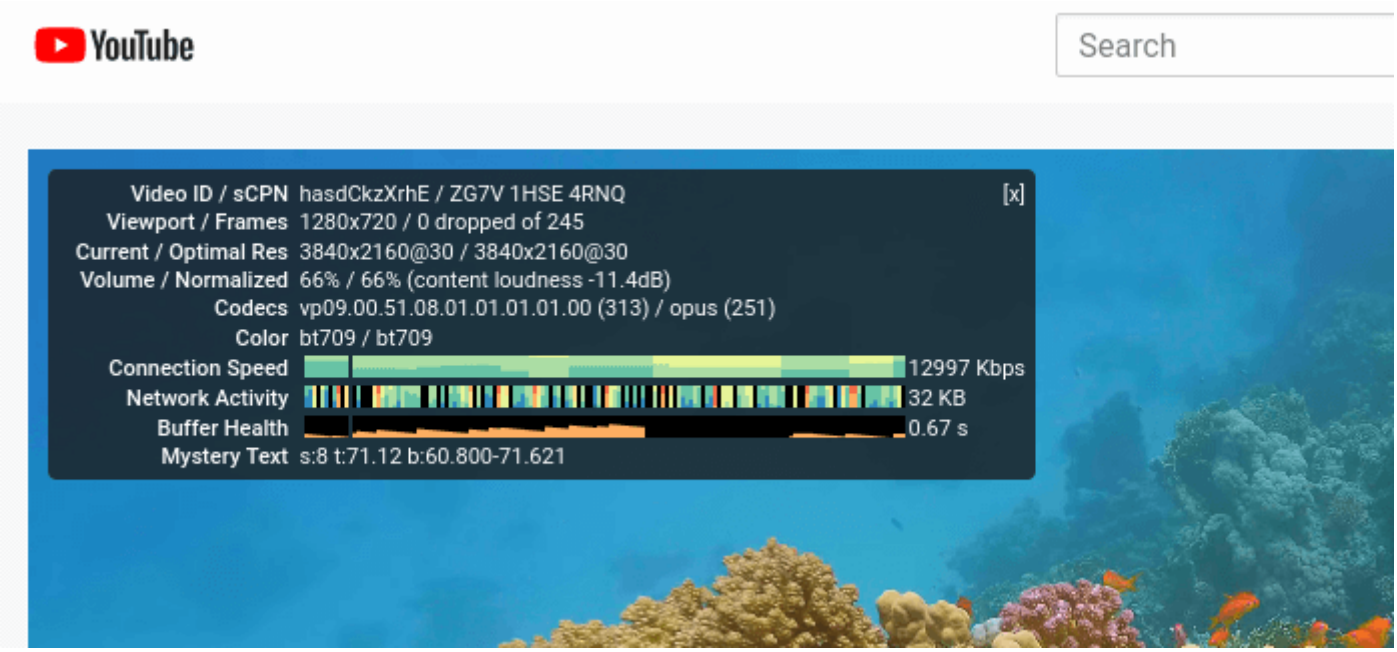
WireGuard is able to reach **52296 Kbps (about 51 Mbit/s)** when playing YouTube videos.



OpenConnect (TLS with TCP BBR algorithm) is able to reach **16504 Kbps (about 16 Mbit/s)** when playing YouTube videos.



OpenConnect (TLS on UDP) is able to reach **12997 Kbps (about 12.7 Mbit/s)** when playing YouTube videos.



Adding Additional VPN Clients

WireGuard is designed to associate one IP address with one VPN client. To add more VPN clients, you need to create a unique private/public key pair for each client, then add each VPN client's public key in the server's config file (`/etc/wireguard/wg0.conf`) like this:

```
[Interface]
Address = 10.10.10.1/24
```

```
PrivateKey = UIFH+XXjJ0g0uAZJ6vPqsbb/o68SYVQdmYJpy/FlGFA=
ListenPort = 51820

[Peer]
PublicKey = 75VNV7HqFh+3QIT5OHZkcjWfbjx8tc6Ck62gZJT/KRA=
AllowedIPs = 10.10.10.2/32

[Peer]
PublicKey = YYh4/1Z/3rtl0i7cJorcinB7T4UOIzScifPNEIESFD8=
AllowedIPs = 10.10.10.3/32

[Peer]
PublicKey = EVstHZc6QamzPgefDGPLFEjGyedJk6SZbCJttpzcvC8=
AllowedIPs = 10.10.10.4/32
```

Each VPN client will have a static private IP address (10.10.10.2, 10.10.10.3, 10.10.10.4, etc). Restart the WireGuard server for the changes to take effect.

```
sudo systemctl restart wg-quick@wg0.service
```

Then add WireGuard configuration on each VPN client as usual.

Automatic-Restart When VPN Connection Drops

Sometimes the VPN connection would drop due to various 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 > /dev/null || systemctl restart wg-quick@wg-client0.service
```

The ping will be done 9 times, i.e 9 seconds. You can use a **for loop** in the Bash shell to make the whole command run 6 times, i.e. 54 seconds.

```
for ((i=1; i<=6; i++)) do (ping -c9 10.10.10.1 > /dev/null || systemctl restart wg-quick@wg-client0.service) done
```

Now we can create a Cron job to automate this task. Edit the root user's crontab file on the VPN client.

```
sudo crontab -e
```

Bash isn't the default shell in Cron. You can add the following line at the beginning of the Crontab file to make it the default.

```
SHELL=/bin/bash
```

Then add the following line at the end of this file.

```
* * * * * for ((i=1; i<=6; i++)) do (ping -c9 10.10.10.1 > /dev/null || systemctl restart wg-quick@wg-client0.service) done
```

This Cron job will run every minute, and there will be 6 checks every minute. Save and close the file.

Advanced Usage

Now I will show you how to use **policy routing**, **split tunneling**, and **VPN kill switch** with WireGuard VPN. **Note** that it's not recommended to use them in conjunction with each other. If you use policy routing, then you should not enable split tunneling or VPN kill switch, and vice versa. This section is for advanced users. If you are a WireGuard beginner and don't know what they are used for, then don't apply the instructions in this section.

Policy Routing

By default, all traffic on the VPN client will be routed through the VPN server. Sometimes you may want to route only a specific type of traffic, based on the transport layer protocol and the destination port. This is known as policy routing.

Policy routing is configured on the client computer, and we need to stop the VPN connection first.

```
sudo systemctl stop wg-quick@wg-client0.service
```

Then edit the client configuration file.

```
sudo nano /etc/wireguard/wg-client0.conf
```

For example, if you add the following 3 lines in the `[interface]` section, then WireGuard will

create a routing table named “1234” and add the ip rule into the routing table. In this example, traffic will be routed through VPN server only when TCP is used as the transport layer protocol and the destination port is 25, i.e, when the client computer sends emails.

```
Table = 1234
PostUp = ip rule add ipproto tcp dport 25 table 1234
PreDown = ip rule delete ipproto tcp dport 25 table 1234
```

Note: The client should be running Ubuntu 20.04 or up in order to configure policy routing. The `ip` utility on Ubuntu 18.04 doesn’t support the `ipproto` and `dport` argument.

Save and close the file. Then start the WireGuard client.

```
sudo systemctl start wg-quick@wg-client0.service
```

Split Tunneling

By default, all traffic on the VPN client will be routed through the VPN server. Here’s how to enable split tunneling, so only traffic to the `10.10.10.0/24` IP range will be tunneled through WireGuard VPN. This is useful when you want to build a private network for several cloud servers, because VPN clients will run on cloud servers and if you use a full VPN tunnel, then you will probably lose connection to the cloud servers.

Edit the client configuration file.

```
sudo nano /etc/wireguard/wg-client0.conf
```

Change

```
AllowedIPs = 0.0.0.0/0
```

To

```
AllowedIPs = 10.10.10.0/24
```

So traffic will be routed through VPN only when the destination address is in the 10.10.10.0/24 IP range. Save and close the file. Then restart WireGuard client.

```
sudo systemctl restart wg-quick@wg-client0.service
```

You can also allow multiple IP ranges. Let's say the VPN server also manages the `10.10.20.0/24` network, then you can configure `AllowedIPs` on the VPN client like this:

```
AllowedIPs = 10.10.10.0/24, 10.10.20.0/24
```

So the VPN client can reach the `10.10.20.0/24` network via the VPN server, and vice versa.

VPN Kill Switch

By default, your computer can access the Internet via the normal gateway when the VPN connection is disrupted. You may want to enable the kill switch feature, which prevents the flow of unencrypted packets through non-WireGuard interfaces.

Stop the WireGuard client process.

```
sudo systemctl stop wg-quick@wg-client0.service
```

Edit the client configuration file.

```
sudo nano /etc/wireguard/wg-client0.conf
```

Add the following two lines in the `[interface]` section.

```
PostUp = iptables -I OUTPUT ! -o %i -m mark ! --mark $(wg show %i fwmark) -m addrtype ! --dst-type LOCAL -j REJECT
PreDown = iptables -D OUTPUT ! -o %i -m mark ! --mark $(wg show %i fwmark) -m addrtype ! --dst-type LOCAL -j REJECT
```

Like this:

```
[Interface]
```

```
Address = 10.10.10.2/24
DNS = 10.10.10.1
PrivateKey = cOFA+x5UvHF+a3xJ6enLatG+DoE3I5PhMgKrMKkUyXI=
PostUp = iptables -I OUTPUT ! -o %i -m mark ! --mark $(wg show %i fwmark) -m addrtype ! --dst-type LOCAL -j REJECT
PreDown = iptables -D OUTPUT ! -o %i -m mark ! --mark $(wg show %i fwmark) -m addrtype ! --dst-type LOCAL -j REJECT

[Peer]
PublicKey = RaoAdsIEIwgV9DHNSubxWVG+nZ1GP/c3OU6A/efBJ0I=
AllowedIPs = 0.0.0.0/0
Endpoint = 12.34.56.78:51820
PersistentKeepalive = 25
```

Save and close the file. Then start the WireGuard client.

```
sudo systemctl start wg-quick@wg-client0.service
```

Multiple Addresses in WireGuard Interface

A WireGuard interface can have multiple IP addresses. For example, you can have two IP addresses on the VPN client.

```
[Interface]
Address = 10.10.10.2/24
Address = 10.10.10.3/24
....
```

In this case, you need to allow multiple IP addresses on the VPN server for this particular client.

```
[Peer]
...
AllowedIPs = 10.10.10.2/32, 10.10.10.3/32
```

Revision #1
Created 11 July 2021 19:13:24 by Admin
Updated 11 July 2021 19:13:51 by Admin