Set Up Your Own WireGuard VPN Server on CentOS/RHEL

This tutorial is going to show you how to set up your own WireGuard VPN server on CentOS/RHEL. 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 **CentOS/RHEL** operating system.

Step 1: Install WireGuard on CentOS/RHEL Server and Desktop

Log into your CentOS/RHEL server, then run the following commands to install WireGuard.

CentOS 8

```
sudo dnf install elrepo-release epel-release -y
sudo dnf install kmod-wireguard wireguard-tools -y
```

RHEL 8

```
sudo dnf install https://www.elrepo.org/elrepo-release-8.el8.elrepo.noarch.rpm
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
sudo dnf install kmod-wireguard wireguard-tools -y
```

CentOS/RHEL 7

```
sudo yum install epel-release https://www.elrepo.org/elrepo-release-7.el7.elrepo.noarch.rpm
sudo yum install yum-plugin-elrepo
sudo yum install kmod-wireguard wireguard-tools -y
```

Then use the same commands to install WireGuard on your local CentOS/RHEL computer (the VPN client).

Step 2: Generate Public/Private Keypair

Server

Create a directory for WireGuard.

```
sudo mkdir -p /etc/wireguard/
```

Run the following command on the CentOS/RHEL 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
```

```
linuxbabe@centos:~$ wg genkey | sudo tee /etc/wireguard/server_private.key | wg pubkey | sudo tee
/etc/wireguard/server_public.key
vxyo4l4I3jWK+KZquNIDJF/hzQq29DOIxSUOrfNZZCs=
linuxbabe@centos:~$
```

Client

Create a directory for WireGuard.

```
sudo mkdir -p /etc/wireguard/
```

Run the following command to create a public/private key pair on the local CentOS/RHEL 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 CentOS/RHEL server. `wg0` will be the network interface name.

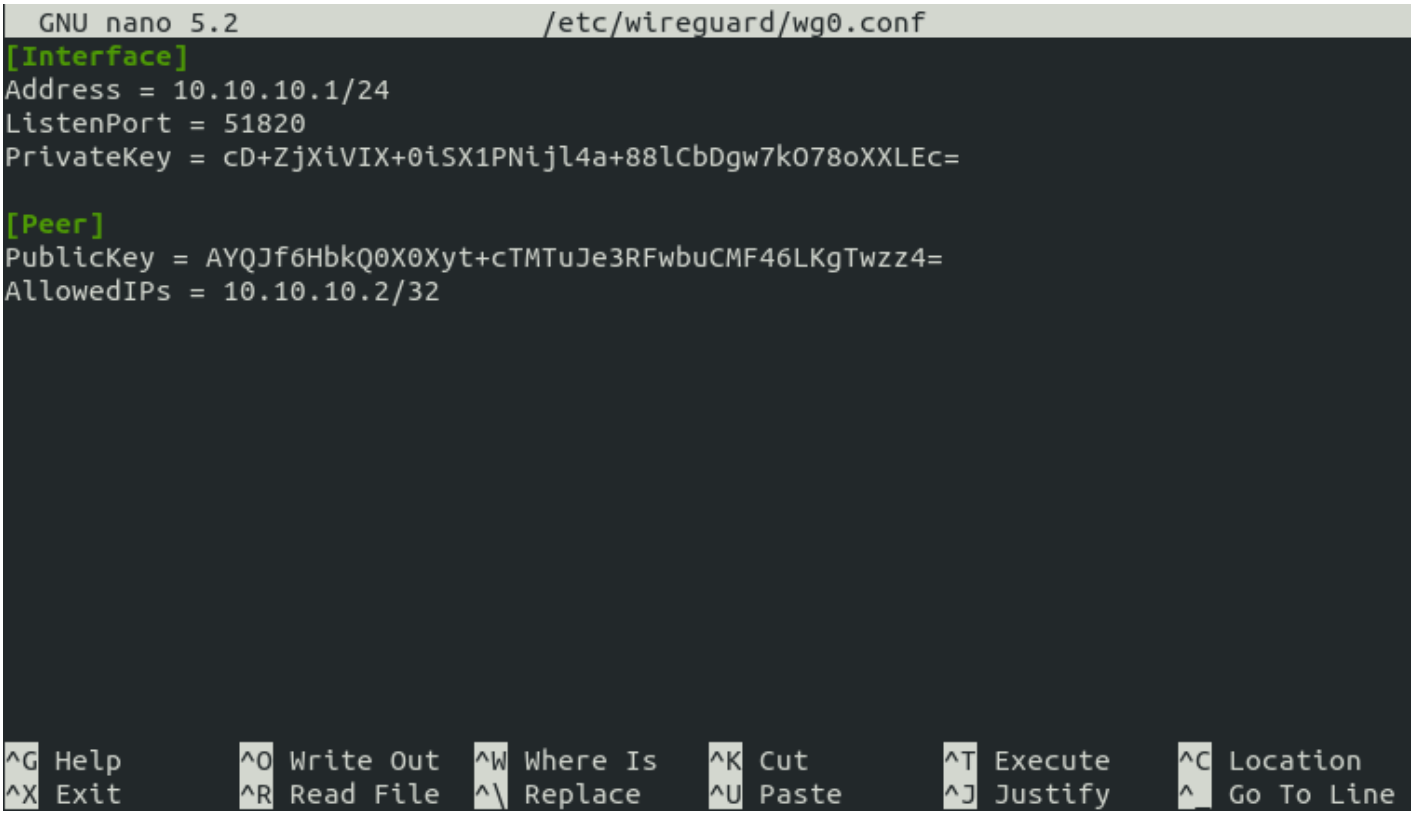
```
sudo dnf install nano
sudo nano /etc/wireguard/wg0.conf
```

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 CentOS/RHEL 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 = vxyo4l4I3jWK+KZquNIDJF/hzQq29DOIxSUOrfNZZCs=
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 (the 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

Run the following command to enable IP masquerading for the `10.10.10.0/24` subnet in the server firewall.

```
sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source
address="10.10.10.0/24" masquerade'
sudo systemctl reload firewalld
```

This will hide your VPN 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.

If your CentOS/RHEL can't find the `firewall-cmd` command, you need to install firewalld and start the service.

```
sudo dnf install firewalld


sudo systemctl start firewalld
```

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 dnf install bind
```

Start BIND 9 with:

```
sudo systemctl start named
```

And enable auto start at boot time:

```
sudo systemctl enable named
```

You can check its status with:

```
systemctl status named
```

Sample output:

```
● named.service - Berkeley Internet Name Domain (DNS)
   Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; vendor preset: disable>
   Active: active (running) since Sun 2020-05-17 11:07:34 EDT; 9s ago
  Process: 7203 ExecStop=/bin/sh -c /usr/sbin/rndc stop > /dev/null 2>&1 || /bin/kill -TE>
  Process: 7218 ExecStart=/usr/sbin/named -u named -c ${NAMEDCONF} $OPTIONS (code=exited,>
  Process: 7215 ExecStartPre=/bin/bash -c if [ ! "$DISABLE_ZONE_CHECKING" == "yes" ]; the>
```

```
 Main PID: 7220 (named)
    Tasks: 4 (limit: 5045)
   Memory: 55.5M
   CGroup: /system.slice/named.service
           └─7220 /usr/sbin/named -u named -c /etc/named.conf -4
```

Hint: If the above command doesn't quit immediately, press Q.

Edit the BIND main configuration file `/etc/named.conf`.

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

In the `options` clause, you can find the following two lines.

```
listen-on port 53 { 127.0.0.1; };
listen-on-v6 port 53 { ::1; };
```

This makes `named` listen on localhost only. If you want to allow clients in the same network to query domain names, then comment out these two lines. (add double slashes at the beginning of each line)

```
// listen-on port 53 { 127.0.0.1; };
// listen-on-v6 port 53 { ::1; };
```

Find the following line.

```
allow-query { localhost; };
```

Add the 10.10.10.0/24 network range so that VPN clients can send DNS queries. Note that you need to end each network range with a semicolon.

```
allow-query { localhost; 10.10.10.0/24; };
```

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

```
sudo systemctl restart named
```

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

```
sudo firewall-cmd --zone=public --permanent --add-rich-rule='rule family="ipv4" source
address="10.10.10.0/24" accept'
```

Step 7: Open WireGuard Port in Firewall

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

```
sudo firewall-cmd --permanent --add-port=51820/udp
sudo systemctl reload firewalld
```

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
```

If the start failed, you should check the log to find out what’s wrong.

```
sudo journalctl -eu wg-quick@wg0.service
```

Enable auto-start at system boot time with the following command.

```
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
```

If the start failed, you should check the log to find out what's wrong.

```
sudo journalctl -eu wg-quick@wg-client0.service
```

If you see the following error in the log, you can try rebooting the OS.

```
RTNETLINK answers: Operation not supported
```

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: `http://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

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.

If the VPN tunnel is successfully established, but the client public IP address doesn't change, that's because the masquerading in the firewall is not working. I once had a typo in the firewall rules, which caused my computer not being 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.

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 dnf update
```

Next, reboot the VPN client.

```
sudo shutdown -r now


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

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.

Policy Routing, Split Tunneling & VPN Kill Switch

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.

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
```

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

```
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
```

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
```

Revision #1
Created 11 July 2021 19:14:36 by Admin
Updated 11 July 2021 19:15:07 by Admin