WireGuard Tunnel

Service for IBM i

CONTENT

Introduction

WireGuard (https://www.wireguard.com/) is a modern open-source VPN already implemented in Linux Kernel. Kernel level support is coming for other OS also. It is a praised as fastest VPN engine compared to other engines and quite easy to install and setup.

This tutorial will show how to install and setup WireGuard in Cloud and to allow remote clients to connect to the Green Screens Server located in the cloud, but unavailable to the public network.

NOTE: WireGuard VPN use UDP protocol which is unreliable and might cause WebSocket connections to drop. There is a solutions to push UDP through TCP explained at the end of the document.

Disclaimer

By using this program, you agree with the following:

- WireGuard is free for use, and not chargeable.
- Use on your own risk. Green Screens Ltd. will not take any responsibility for damage.
- Green Screens Ltd. does not provide any technical support for WireGuard.

Requirements

WireGuard must be installed on a server access point and client workstations. Detailed instructions can be found on WireGuard web site shown at link below. Here are basic information to get started.

1. Install WireGuard on Linux access point server

    sudo apt get update
    sudo apt get install wireguard

2. Install WireGuard on workstations

    Download client for workstation operating system from https://www.wireguard.com/install/

First start

Before creating configuration, we need to collect a few parameters:

- Server public / private key
- Client public / private key
- Endpoint IP address and port
- Allowed IP addresses
- VPN network IP address (private range)
- Optional DNS IP address
- Optional server firewall rules
- Optional server routing table

Keys - simple encoded string where server public key is setup on the client, while client public key is setup on the server configuration.

Endpoint - is a WireGuard server IP clients use to connect to and opens secured VPN tunnel

Allowed IP - when WireGuard client is started, virtual network card (VNC) is added. IP addresses defined here, are routed by the operating system through created VNC. Other IP's if not blocked will use default network.

VPN network IP - is a IP address in private which depending on requirements might be in the same address range as server private network or if isolation is required, use different private network address range.

DNS IP - configured on virtual WireGuard interface on client side. When client use URL with names instead of Ip address, those are DNS servers available from WireGuard server side.

Firewall Rules - rules to be added on WireGuard server side upon start and removed upon stop

Routing Tables - rules to be added on WireGuard server side to allow packet redirections if required.

Generating Server keys

On server side use WireGurd provided tool to generate public and private key hashes. This command will output two files: wg0.priv and wg0.pub. They are simple one line text files.

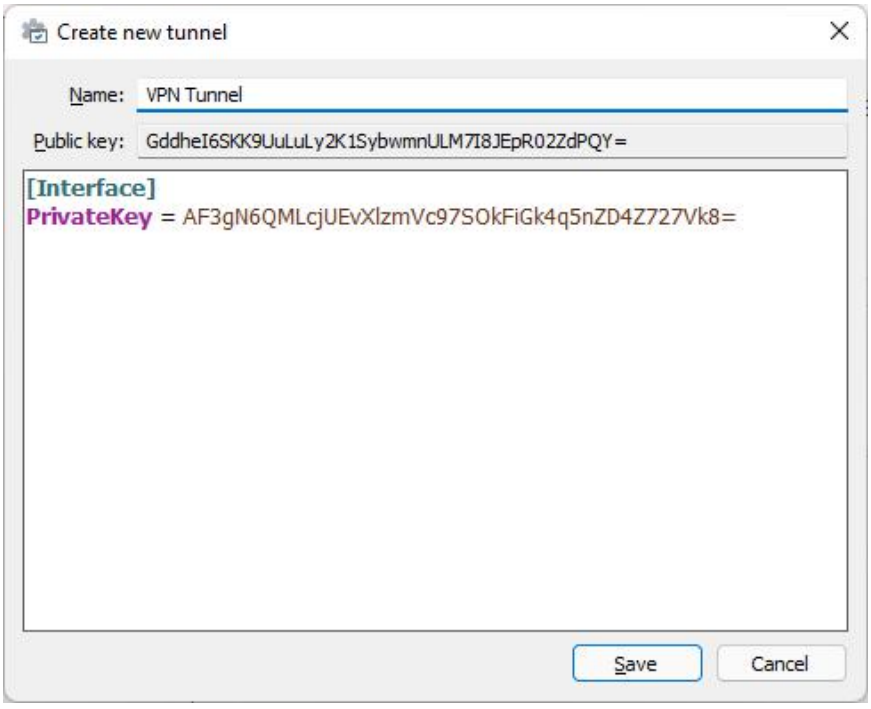
```
# change to the root user
sudo -s

# make sure files created are accessible to root only
umask 077

# generate public and private keys
wg genkey | tee wg0.priv | wg pubkey > wg0.pub
```

Generating Client keys

On client side, start WireGuard client, and choose option **Add Empty Tunnel**. New window will popup and auto generate public and private keys. Public key generated here will be added on the server side later. WireGuard server use client public key to verify client connection and encryption.



NOTE: Public keys must be exchanged between client and server configurations. Server public key must be added to the client configuration, while client public key must be added to the server configuration.

Configuration examples

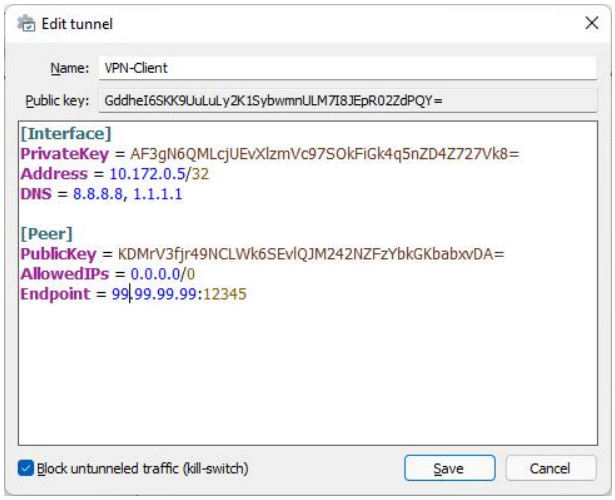
Server keys generated on WireGuard server:

Public: KDMrV3fjr49NCLWk6SEvlQJM242NZFzYbkGKbabxvDA=
Private: GJ0tEuPVN4YNXkUc/lEBdvPvoKhSHErfebRVVuNu6lM=

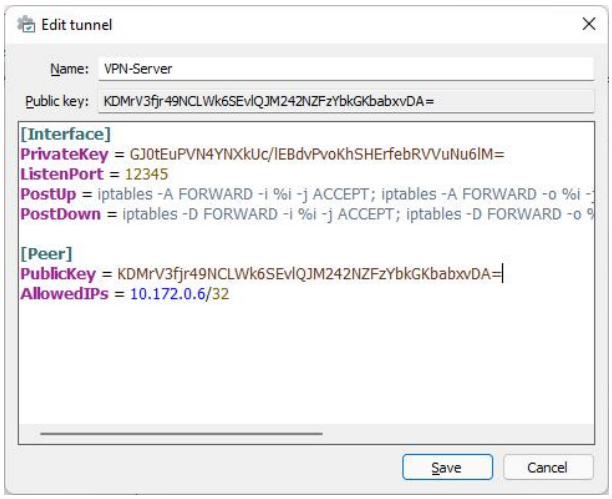
Client keys generated on WireGuard client:

Public: GddheI6SKK9UuLuLy2K1SybwmnULM7I8JEpR02ZdPQY=
Private: AF3gN6QMLcjUEvXlzmVc97SOkFiGk4q5nZD4Z727Vk8=

Client configuration, address is IP in the same range as on server side private network. Endpoint is WireGuard server access point client will connect to. AllowedIPs are IP addresses routed through VPN channel to the WireGuard server, while other IP's will be blocked or will use standard local network. This depends on "Block" flag visible at the bottom of the image.



On server side, AllowedIPs is client private network IP address to be accepted and routed to server internal network. PostUp and PostDown are optional but useful if firewall is used.



Auto start WireGuard on boot

To enable WireGuard start on system boot, use systemd service integration. Follow steps below.

1. Add the WireGuard service to systemd:

sudo systemctl enable wg-quick@wg0.service
sudo systemctl daemon-reload

2. Start the new service immediately:

sudo systemctl start wg-quick@wg0

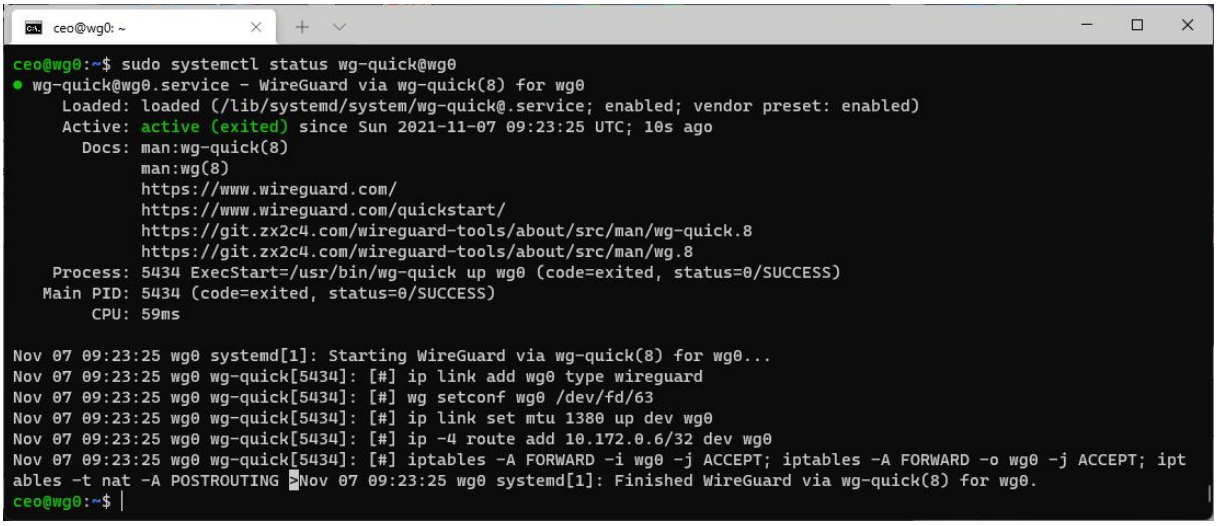
3. Check the service status:

sudo systemctl status wg-quick@wg0

4. To remove the service and clean up the system:

sudo systemctl stop wg-quick@wg0
sudo systemctl disable wg-quick@wg0.service
*sudo rm -i /etc/systemd/system/wg-quick@wg0**
sudo systemctl daemon-reload
sudo systemctl reset-failed

Status of WireGuard wg0 interface used for VPN connection.



Appendix A.

WireGuard can be used for anonymous Internet browsing as Internet exit point. To make this work, all what is required is proper routing table which will allow VPN received packets rerouting to a real network interface.

Server Side

Enable IP Forwarding

In Linux, IP forwarding must be enabled first by editing /etc/sysctl.conf and enabling or adding the following line if it does not exists **net.ipv4.ip_forward = 1**

1. Open sysctl.conf and uncomment or add net.ipv4.ip_forward = 1

```
sudo vi /etc/sysctl.conf
```

2. Reload configuration

```
sudo sysctl -p
```

Enable IP Masquerading

Append NAT IP masquerade to a routing table on main interface (not WireGuard interface). Usually starts with eth* or ens*.

```
sudo iptables -t nat -A POSTROUTING -s 10.172.0.0/24 -o eth0 -j MASQUERADE
```

Enable UDP routing

Append the rule to allow UDP traffic to the WireGuard server at listen port

```
iptables -A INPUT -p udp -m udp --dport 12345 -j ACCEPT
```

Enable data traffic

Append the rule to allow traffic forwarded to or from WireGuard interface named wg0

```
iptables -A FORWARD -i wg0 -j ACCEPT
iptables -A FORWARD -o wg0 -j ACCEPT
```

Client side

In WireGuard client side configuration add DNS servers IP address and make sure that IP address is in the same network range as WireGuard server private network. Otherwise, advanced routing table settings will be required.

```
[Interface]
PublicKey = wfDsD3uUhgt84ADnFgHD4rkZtedo2BVHHlEo80twv2E=
PrivateKey = fcaWE42f2fasdFG27j6cAEer32fa+2cuAPu4Rz/ShQ3w=
Address = 10.172.0.5/32
DNS = 8.8.8.8, 1.1.1.1

[Peer]
PublicKey = yfaver+fqwcas53rd2/TPi/FafAW/t+vAEBer3rrg3=
AllowedIPs = 0.0.0.0/0
Endpoint = 34.65.99.19:51820
```

Appendix B.

WireGuard use UDP protocol which is crucial for fast VPN, but not reliable as TCP. That might create connection reliability issues depending on ISP's, networks etc.

To improve network reliability, especially for terminal sessions, and other long running permanently opened socket connections, there are open-source tools which might help.

NOTE: Green Screens Ltd. Is not responsible for usage of this solutions neither gives any technical support. It is solely customer responsibility.

UDP Tunnel - https://manpages.ubuntu.com/manpages/focal/man1/udptunnel.1.html

On server side, start udptunnel service

```
udptunnel -s 443 127.0.0.1/51820
```

On client side, start udptunnel connection

```
udptunnel -c [SERVER PUBLIC IP]/443 127.0.0.1 50001
```

On client side, change WireGuard config file to point to a 127.0.0.1:50001, which is local udptunnel address.

UDP2RAW- https://github.com/wangyu-/udp2raw

A Tunnel which turns UDP Traffic into Encrypted FakeTCP/UDP/ICMP Traffic by using Raw Socket, helps you Bypass UDP FireWalls(or Unstable UDP Environment). Generally considered as better choice than udptunnel.

On server side, start udp2raw service

udp2raw -s -l "0.0.0.0:<desired port>" -r "127.0.0.1:<Wireguard server's port>"
-k "<desired password>" --raw-mode faketcp -a

On server client side, start udp2raw client

*udp2raw -c -l "127.0.0.1:*50001*" -r "<Wireguard server's IP address>:<server_port>"*
-k "<server password>" --raw-mode faketcp -a

On client side, change WireGuard config file to point to a 127.0.0.1:50001, which is local udptunnel address.