

WireGuard VPN Implementation Report

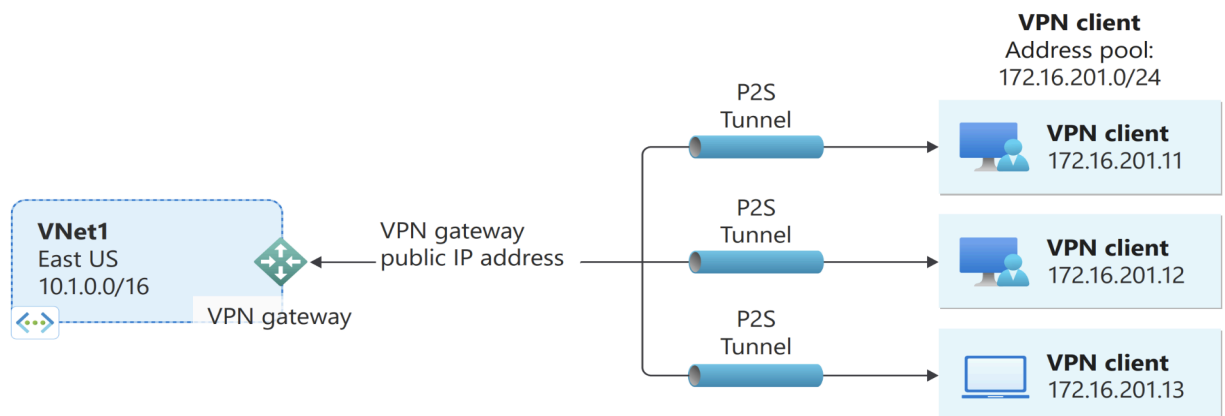


Author: Fouenang Miguel Bruce

Address: miguelfouenanf@gmail.com

Date: October 20, 2025

1. Introduction



A Virtual Private Network (VPN) provides a secure communication channel between two or more systems through **encryption** and **tunneling**. WireGuard is a modern, lightweight, and high-performance VPN protocol that uses advanced cryptography and is easy to configure compared to traditional VPNs such as OpenVPN and IPSec.

WireGuard's simplicity and speed make it ideal for secure point-to-point connections, as demonstrated in this project.

2. Objectives

The objective of this project is to implement a secure ****point-to-point VPN tunnel**** between two virtual machines: an **Alpine Linux server** and a **Linux Mint client**. The tunnel allows encrypted data communication within a private subnet (e.g., 10.0.0.0/24).

This setup ensures privacy and security for data transmission in a controlled virtual environment.

3. Methodology

The setup was implemented using two virtual machines running on a hypervisor (such as VirtualBox or Virt-Manager). The Alpine VM was configured as the VPN server, and the Linux Mint VM as the client. The WireGuard interface (**wg0**) was configured on both sides to create an encrypted link across the virtual network.

Key steps included package installation, key pair generation, configuration file setup, and interface activation.



shutterstock.com · 2602832919

4. Implementation

Server Configuration (Alpine Linux)

1. Install WireGuard:

```
`sudo apk add wireguard-tools`
```

2. Generate key pairs:

```
`wg genkey | tee privatekey | wg pubkey > publickey`
```

3. Configure /etc/wireguard/wg0.conf:

```
...  
[Interface]  
Address = 10.0.0.1/24  
PrivateKey = <SERVER_PRIVATE_KEY>  
ListenPort = 51820  
  
[Peer]  
PublicKey = <CLIENT_PUBLIC_KEY>  
AllowedIPs = 10.0.0.2/32  
...
```

4. Bring up the interface:

```
`sudo wg-quick up wg0`
```

Client Configuration (Linux Mint)

1. Install WireGuard:

```
`sudo apt install wireguard`
```

2. Generate key pairs:

```
`wg genkey | tee privatekey | wg pubkey > publickey`
```

3. Configure /etc/wireguard/wg0.conf:

```
...  
[Interface]  
Address = 10.0.0.2/24  
PrivateKey = <CLIENT_PRIVATE_KEY>  
[Peer]  
PublicKey = <SERVER_PUBLIC_KEY>  
Endpoint = 192.168.56.101:51820  
AllowedIPs = 10.0.0.0/24
```

PersistentKeepalive = 20

...

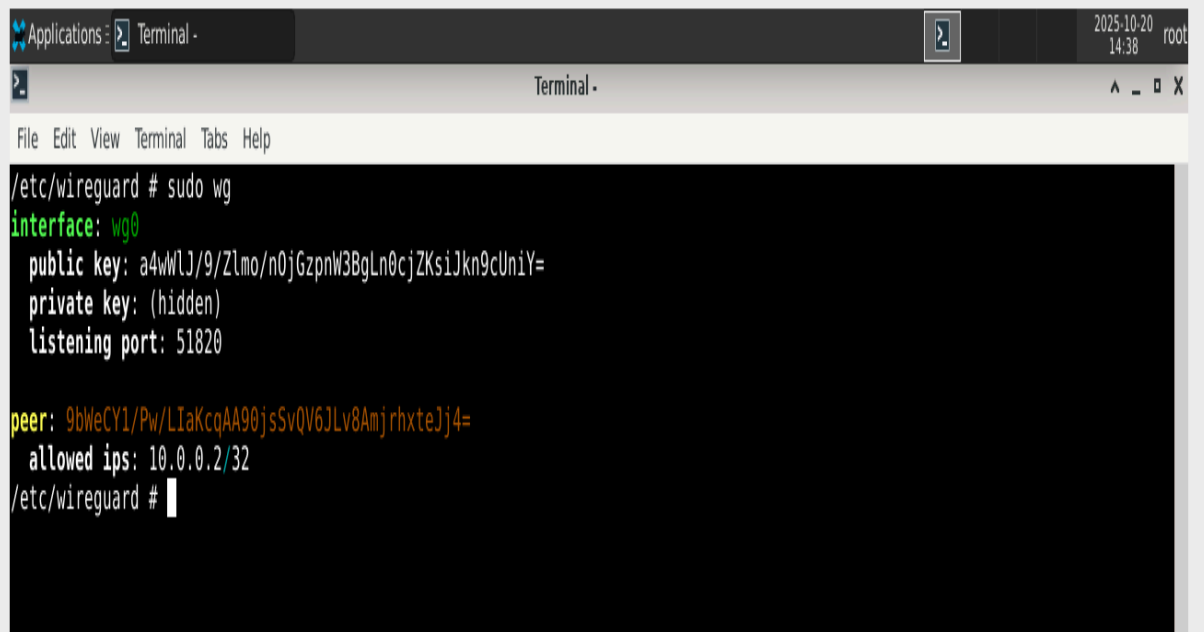
4. Bring up the interface:

``sudo wg-quick up wg0``

Vérification

-Check tunnel status:

``sudo wg``

A terminal window titled 'Terminal -' with a menu bar (File, Edit, View, Terminal, Tabs, Help) and a status bar (2025-10-20 14:38 root). The terminal shows the command `sudo wg` being executed in the `/etc/wireguard` directory. The output displays the configuration for the `wg0` interface, including its public key, private key (hidden), and listening port (51820). It also shows a peer configuration with its public key, allowed IP address (10.0.0.2/32), and the current status of the tunnel.

```
/etc/wireguard # sudo wg
interface: wg0
  public key: a4wWlJ/9/Zlmo/n0jGzpnW3BgLn0cjZKsiJkn9cUniY=
  private key: (hidden)
  listening port: 51820

peer: 9bWeCY1/Pw/LIaKcqAA90jsSvQV6JLv8AmjrhxteJj4=
  allowed ips: 10.0.0.2/32
/etc/wireguard #
```

```
bruce@bruce-VirtualBox: /etc/wireguard
bruce@bruce-VirtualBox: /etc/wireguard$ nano wg0.conf
bruce@bruce-VirtualBox: /etc/wireguard$ sudo wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.0.0.2/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
bruce@bruce-VirtualBox: /etc/wireguard$ sudo wg
interface: wg0
  public key: 9bWeCY1/Pw/LIaKcqAA90jsSvQV6JLv8AmjrhxteJj4=
  private key: (hidden)
  listening port: 39614

peer: a4wWlJ/9/Zlmo/n0jGzpnW3BgLn0cjZKsiJkn9cUniY=
  endpoint: 192.168.56.101:51820
  allowed ips: 10.0.0.0/24
  transfer: 0 B received, 296 B sent
  persistent keepalive: every 20 seconds
bruce@bruce-VirtualBox: /etc/wireguard$
```

- Test connectivity:

From client: **`ping 10.0.0.1`**

From server: **`ping 10.0.0.2`**

The tunnel was successfully established, with encrypted communication verified through basic ping tests.

```
bruce@bruce-VirtualBox:~$ sudo wg-quick up wg0
wg-quick: 'wg0' already exists
bruce@bruce-VirtualBox:~$ sudo wg
interface: wg0
  public key: 9bWeCY1/Pw/LIaKcQAA90jsSvQV6JLv8AmjrhxteJj4=
  private key: (hidden)
  listening port: 39614

peer: a4wWlJ/9/Zlmo/n0jGzpnW3BqLn0cjZKsiJkn9cUnlY=
  endpoint: 192.168.56.101:51820
  allowed ips: 10.0.0.0/24
  transfer: 0 B received, 6.07 KiB sent
  persistent keepalive: every 20 seconds
bruce@bruce-VirtualBox:~$ ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
^C
--- 10.0.0.1 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 13330ms

bruce@bruce-VirtualBox:~$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.130 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.076 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.147 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.025 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.062 ms
```

5. Conclusion

This project demonstrated a successful implementation of a secure VPN tunnel using WireGuard between two Linux-based virtual machines. The setup provided a reliable, encrypted communication channel and served as a practical example of network security in action.

Note:

To enhance this project further i will add :

- Add support for multiple peers.
- Implement firewall rules for added security.
- Conduct performance tests (e.g., throughput and latency).