# Tunnel L3 - Wireguard

# 1. Introduction : Pourquoi utiliser Wireguard ?

## 1.1. À quoi sert un tunnel Wireguard ?

Un tunnel **Wireguard** permet de créer un lien direct et sécurisé entre deux réseaux distants. Wireguard est un protocole VPN moderne, rapide et simple à configurer. **Exemple concret :**

- Vous avez un réseau local derrière une box opérateur standard.
- Vous voulez une IP publique sur une ou des machines sur votre réseau local

## 1.2. Avantages de Wireguard

- Extrêmement simple à configurer.
- Chiffrement intégré (pas besoin d'IPsec en complément).
- Connexion possible en IPv4 & IPv6.
- Règles de routage ajoutées automatiquement.
- Aucun port à ouvrir côté client (traverse le NAT nativement).
- Protocole léger et performant.

## 1.3. Inconvénients

- Moins de support natif sur certains routeurs (Cisco, Juniper...).
- Protocole plus récent que GRE ou IPIP.

---

# 2. Prérequis

## 2.1. Ce dont vous avez besoin

- Un **routeur compatible Wireguard** (Linux, MikroTik, etc.).
- Un **service Tunnel-IP**.

## 2.2. Ports et NAT

- Wireguard utilise **UDP** (port par défaut : 51820).
- Contrairement à GRE et IPIP, **aucune redirection de port n'est nécessaire côté client** . Wireguard traverse le NAT nativement grâce au mécanisme de keepalive.

---

# 3. Étape 1 : Créer le tunnel sur Tunnel-IP.com

## 3.1. Accéder au tableau de bord

1. Connectez-vous à panel.tunnel-ip.com
2. Allez dans **"Tunnels"** > **"Wireguard"**.
3. Remplissez les informations :
   - **Nom du tunnel** (ex : `Tunnel_X`).
   - **Endpoint** (IP publique du routeur / box, ex : `203.0.113.1`).
4. Validez. La plateforme s'occupera de choisir les IPs et de configurer le tunnel côté Tunnel-IP.com.
5. Attendez que le tunnel soit créé
6. Cliquez sur "Accéder" pour récupérer les détails du tunnel

Il vous suffit de configurer votre wireguard avec les paramètres indiqués en **1**, pour linux vous pouvez suivre ce tutoriel

## 3.2. Créer le subnet (Route côté plateforme)

1. Allez dans "Subnets"

2. Copiez le bloc d'IP publique et collez le dans le formulaire

3. Cliquez sur Créer

4. Une fois son statut à "Actif", la route est correctement installée sur les routeurs de la plateforme, il ne vous reste plus qu'à configurer le tunnel

---

# 4. Étape 2 : Configurer le tunnel

## 4.1. Exemple pour Linux (Ubuntu/Debian) avec wg-quick

### Étapes :

1. **Installer Wireguard** :

   ```
   apt update && apt install wireguard -y
   ```

2. **Créer le fichier de configuration** :

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

   Contenu du fichier (à adapter avec les paramètres fournis par la plateforme) :

   ```
   [Interface]
    PrivateKey = <votre_clé_privée>
    Address = 100.64.0.6/30
    Address = fd00:42:cafe:1::2/64

    [Peer]
    PublicKey = <clé_publique_plateforme>
    Endpoint = 172.16.126.33:51820
    AllowedIPs = 0.0.0.0/0, ::/0
    PersistentKeepalive = 25
   ```

- `PrivateKey` : Votre clé privée (générée automatiquement ou manuellement avec `wg genkey`).
- `Address` : IPs internes du tunnel (fournies par la plateforme).
- `PublicKey` : Clé publique du serveur Tunnel-IP.com.
- `Endpoint` : Adresse et port du serveur distant.
- `AllowedIPs` : Plages d'IPs autorisées à passer par le tunnel.
- `PersistentKeepalive` : Envoie un paquet toutes les 25 secondes pour maintenir la connexion à travers le NAT.

3. **Activer le tunnel** :

```
wg-quick up wg0
```

4. **Activer au démarrage** :

```
systemctl enable wg-quick@wg0
```

5. **Activer le forwarding IP** (pour permettre le routage) :

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

**(Pour rendre permanent, ajoutez *`net.ipv4.ip_forward=1`* dans *`/etc/sysctl.conf`*.)**

---

# 4.2. Exemple pour MikroTik

## Étapes :

1. **Créer l'interface Wireguard** :

```
/interface/wireguard/add name=wg0 listen-port=51820
```

2. **Récupérer la clé publique** (à fournir à la plateforme) :

```
/interface/wireguard/print
```

3. **Ajouter le peer (serveur Tunnel-IP.com)** :

```
/interface/wireguard/peers/add interface=wg0 \
   public-key="<clé_publique_plateforme>" \
   endpoint-address=172.16.126.33 \
   endpoint-port=51820 \
   allowed-address=0.0.0.0/0,::/0 \
   persistent-keepalive=25
```

4. **Assigner une IP au tunnel** :

```
/ip/address/add address=100.64.0.6/30 interface=wg0
 /ipv6/address/add address=fd00:42:cafe:1::2/64 interface=wg0
```

---

# 4.3. Note pour Cisco, Arista et Juniper

> Wireguard n'est pas nativement supporté sur la plupart des routeurs Cisco (IOS/XE), Arista (EOS) et Juniper (JunOS). Si vous devez utiliser l'un de ces équipements, vous pouvez :

- Utiliser un serveur Linux comme point de terminaison Wireguard, puis router le trafic vers votre routeur.
- Utiliser un autre type de tunnel (GRE, IPIP, IPsec) nativement supporté par ces équipements.

---

# 5. Étape 3 : Router les IP publiques vers le tunnel

# 5.1. Pourquoi ?

La plateforme vous route un bloc d'IP publiques (ex : `198.51.100.0/30`) sur votre IP interne du tunnel, afin que ces IPs fonctionnent correctement, vous devez router ce bloc d'IP sur votre réseau et le trafic sortant par le tunnel.

Pour cela, 3 principales méthodes s'offrent à vous :

- **NAT 1:1** (1 IP publique → 1 IP privée)
- **LAN public** (utilisation directe des IPs publiques)
- **Routage en /32** (1 IP publique par machine)

---

# 5.2. Rappel : Architecture du Tunnel

```
Internet → [Infrastructure Tunnel-IP.com] → (Tunnel Wireguard) → [Votre Routeur] → [Votre Réseau]
```

- La plateforme vous route un bloc public (ex: `198.51.100.0/30`).
- Votre routeur doit gérer ce bloc pour exposer vos services.

---

# 5.3. Cas d'Usage du Bloc /30

Un `/30` contient **4 adresses IP** :

- `198.51.100.0` : Adresse réseau (inutilisable en LAN).
- `198.51.100.1` : IP Utilisable
- `198.51.100.2` : IP Utilisable
- `198.51.100.3` : Adresse broadcast (inutilisable en LAN).

---

# 5.4. Méthode 1 : NAT 1:1 (Translation 1 IP publique → 1 IP privée)

**Avantages principaux :** Permet de ne pas avoir à modifier l'adressage privé de votre réseau, et permet d'utiliser l'IP de réseau et broadcast.

**Inconvénients :** Adressage moins clair (Conversion IP publique -> privée), charge plus lourde sur le routeur (Le routeur est en charge de la translation NAT)

# Schéma :

```
Internet → Infrastructure Tunnel-IP.com → [Votre Routeur] 198.51.100.0 → 192.168.1.100 (Privée)
```

# Configurations :

## Linux :

```
# Activer l'IP Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
# Ajouter l'IP sur une interface (on utilise la loopback par exemple)
ip a add 198.51.100.0/32 dev lo

# 1:1 Static NAT
iptables -t nat -A PREROUTING -d 198.51.100.0 -j DNAT --to-destination 192.168.1.100
iptables -t nat -A POSTROUTING -s 192.168.1.100 -j SNAT --to-source 198.51.100.0
```

### MikroTik :

```
/ip address add address=198.51.100.0/32 interface=eth0
/ip firewall nat add chain=dstnat dst-address=198.51.100.0 action=dst-nat to-
addresses=192.168.1.100
/ip firewall nat add chain=srcnat src-address=192.168.1.100 action=src-nat to-
addresses=198.51.100.0
```

---

## 5.5 Méthode 2 : LAN Public (Utilisation directe des IPs publiques)

**Avantages principaux :** Adressage propre, explicite

**Inconvénients :** Perte de 2 IPs utilisables (IP de réseau et IP de broadcast) + 1 IP est forcément assignée au routeur.

> **(Attention : Avec un /30 en LAN, vous n'avez qu'une seule IP utilisable. Pour plus d'IPs, prenez un bloc plus grand, ex: `/29`. ou faites du NAT 1:1 ou attribution en /32)**

## Configurations :

### Linux :

```
# Assigner l'IP de la passerelle (votre routeur)
ip addr add 198.51.100.1/30 dev eth1
```

### MikroTik :

```
/ip address add address=198.51.100.1/30 interface=ether1
```

# 5.6. Méthode 3 : Routage en /32 (1 IP publique par machine)

**Avantages principaux :** Adressage propre, explicite, aucune perte d'IP

**Inconvénients :** Non supporté par Windows, mal supporté par certains routeurs / OS, 100% du trafic passe par le routeur

# Configurations :

## Linux :

```
# Sur le routeur :
ip route add 198.51.100.0/30 dev eth0  # eth0 = interface vers la Machine A

# Sur la Machine A :
ip addr add 203.0.113.1/32 dev eth0
ip route add default via 192.168.1.100 dev eth0 # IP de votre routeur local
# Si error lors de l'ajout de la route, il faut ajouter une règle :
# ip route add 192.168.1.100 dev eth0
```

## MikroTik :

```
/ip route add dst-address=198.51.100.0/30 interface=LAN

# Sur la Machine A :
# Pareil que pour Linux, IP : 198.51.100.1 Gateway 192.168.1.100
```

# 5.7. Router le trafic sortant par le tunnel

> Si vous ne faites pas cette étape, les IPs ne fonctionneront pas

>

> Si vous avez configuré `AllowedIPs = 0.0.0.0/0` dans votre configuration Wireguard, tout le trafic passe déjà par le tunnel. Cependant, si vous souhaitez un routage plus fin (uniquement le trafic des IPs publiques), utilisez les méthodes ci-dessous :

# Linux (ip route + policy routing)

```
# Création d'une table de routage
echo "144 tunnel" >> /etc/iproute2/rt_tables

# Routage vers l'extérieur via l'IP locale de la plateforme
ip route add default via 100.64.0.5 table tunnel
# Tout le trafic en provenance de nos IPs publiques, sortent via le tunnel
ip rule add from 198.51.100.0/30 table tunnel
```

**Effet :** Tout hôte du réseau `198.51.100.0/30` sort par le tunnel, les autres via la gateway par défaut.

> Si vous êtes à l'aise avec les VRF, il est aussi possible de faire avec :

```
ip link add vrf-WG type vrf table 144 # On créer la VRF
ip link set dev wg0 master vrf-WG # On ajoute le tunnel dans la VRF
ip route add default via 100.64.0.5 table 144 # On ajoute la route par défaut
ip link set dev eth0 master vrf-WG # On ajoute notre interface LAN dans la VRF (Attention si elle est partagée avec votre réseau local, le réseau local ne marchera plus, pour cela vous pouvez créer une interface dummy ou utiliser une interface séparée)
```

---

# MikroTik

```
/routing/table/add name=WG-OUT fib
/routing/rule/add action=lookup-only-in-table table=WG-OUT src-address=198.51.100.0/30
/ip route add dst-address=0.0.0.0/0 gateway=100.64.0.5 routing-table=WG-OUT
```

> Si vous êtes à l'aise avec les VRF, il est aussi possible de faire avec :

```
/ip/vrf add name=WG-VRF interfaces=wg0,ether1
/ip route add routing-table=WG-VRF dst-address=0.0.0.0/0 gateway=100.64.0.5
```

---

# 6. Commandes spécifiques par constructeur

| Constructeur | Commande pour vérifier le tunnel | Commande pour vérifier le routage |
|---|---|---|
| **Linux** | `wg show` | `ip route` |
| **MikroTik** | `/interface wireguard print` | `/ip route print` |

# 7. Vérification et Dépannage

## 7.1. Tester la connectivité

- **Depuis votre routeur** :

```
ping 100.64.0.5 # Ping de l'ip distante du tunnel
  wg show # Vérifier l'état du handshake
```

- **Depuis une machine locale** :

```
ping 8.8.8.8 # Vérification internet
  curl -4 ifconfig.me # Vérification de l'IP sortante
```

## 7.2. Problèmes courants

| Symptôme | Cause Possible | Solution |
|---|---|---|
| Le tunnel ne s'établit pas (pas de handshake) | Clé publique/privée incorrecte | Vérifiez les clés avec `wg show` et comparez avec la plateforme |
| Le tunnel ne s'établit pas | Endpoint incorrect ou port UDP bloqué | Vérifiez l'endpoint et que le port UDP est accessible |
| Les IPs publiques ne pingent pas | Route manquante ou NAT mal configuré | Vérifiez `ip route` ou `/ip route print` |
| Latence élevée / Pertes de paquets | MTU trop grande | Réduisez la MTU : `ip link set mtu 1420 dev wg0` |

| | | |
|---|---|---|
| Connexion perdue derrière NAT | PersistentKeepalive non configuré | Ajoutez `PersistentKeepalive = 25` dans la configuration du peer |

---

# 8. Bonnes Pratiques

1. **Sécurité** :
   - Wireguard intègre le chiffrement nativement (ChaCha20, Poly1305, Curve25519).
   - Filtrez le trafic entrant avec un pare-feu (ex: `iptables`).
   - Protégez vos clés privées et ne les partagez jamais.
2. **Performance** :
   - La MTU par défaut de Wireguard est `1420`. Ajustez si nécessaire en fonction de votre connexion.
   - Activez le **PersistentKeepalive** (ex: `25` secondes) si vous êtes derrière un NAT.
3. **Redondance** :
   - Configurez un deuxième tunnel Wireguard pour le failover.

---

Revision #3
Created 2026-03-30 14:09:58 UTC by Landry JUGE
Updated 2026-03-30 14:28:52 UTC by Landry JUGE