WireGuard Fast VPN tunnels

START HERE

`wg genkey | tee privatekey | wg pubkey > publickey`
Make a key pair

`sudo wg-quick up <iface>`
Bring tunnel up

`sudo wg-quick down <iface>`
Bring tunnel down

`sudo wg show`
Peers and handshakes

`sudo systemctl enable --now wg-quick@<iface>`
Start at boot

`wg genpsk`
Preshared key

`sudo wg set <iface> peer <pubkey> remove`
Drop a peer

`sudo wg syncconf <iface> <(wg-quick strip <iface>)`
Reload live

Packet to 10.0.0.2 → `AllowedIPs` → **Peer** matching key → `encrypt, UDP` → **Endpoint** ip:port

CONCEPTS

Each peer is a public key plus the IPs it may use (AllowedIPs). The same list routes outgoing packets and filters incoming ones.

There is no server or client in the protocol; any peer with a reachable Endpoint can accept connections.

A tunnel is silent until traffic flows: no handshake is shown before the first packet.

KEYS

`wg genkey > privatekey`
Generate a private key
Run umask 077 first so the file is not world-readable.

`wg pubkey < privatekey > publickey`
Derive the public key

`wg genpsk > psk`
Generate a preshared key
Optional extra symmetric layer; both peers need the same value.

`wg genkey | tee privatekey | wg pubkey`
Both keys in one go

BRING UP AND DOWN

`sudo wg-quick up <iface>`
Start tunnel from /etc/wireguard/<iface>.conf

`sudo wg-quick down <iface>`
Stop tunnel and undo routes

`sudo wg-quick up <path>.conf`
Use a config file by path
The interface name is taken from the file name.

`sudo wg-quick save <iface>`
Write live state back to the file

`wg-quick strip <iface>`
Print config without wg-quick keys

`sudo systemctl enable --now wg-quick@<iface>`
Start now and at boot

`sudo systemctl restart wg-quick@<iface>`
Apply config changes (drops sessions)

`sudo ip link set <iface> up`
Raise a manually built interface

INSPECT

`sudo wg`
Show all interfaces and peers

`sudo wg show <iface>`
Show one interface

`sudo wg show <iface> latest-handshakes`
Last handshake per peer (epoch)

`sudo wg show <iface> transfer`
Bytes received and sent per peer

`sudo wg show <iface> allowed-ips`
AllowedIPs per peer

`sudo wg show <iface> endpoints`
Current endpoint per peer

`sudo wg show interfaces`
List WireGuard interfaces

`sudo wg show <iface> dump`
Tab-separated output for scripts

`sudo wg showconf <iface>`
Print live config in file format

CHANGE LIVE

`sudo wg set <iface> peer <pubkey> allowed-ips <cidr> endpoint <ip:port>`
Add or update a peer

`sudo wg set <iface> peer <pubkey> remove`
Remove a peer

`sudo wg set <iface> peer <pubkey> allowed-ips +<cidr>`
Add one range, keep others
Needs a wireguard-tools version that supports the +/- prefix.

`sudo wg set <iface> peer <pubkey> persistent-keepalive 25`
Keep NAT mapping alive

`sudo wg set <iface> peer <pubkey> preshared-key <file>`
Set a preshared key

`sudo wg setconf <iface> <file>`
Replace config (plain wg format)

`sudo wg syncconf <iface> <file>`
Apply only the differences
Does not accept wg-quick keys such as Address; feed it wg-quick strip output.

CONFIG FILE

[Interface]

Field	Meaning
`PrivateKey`	Own secret key
`Address`	Tunnel IP, e.g. 10.0.0.1/24
`ListenPort`	UDP port (51820 usual)
`DNS`	Resolver for the tunnel
`MTU`	Override detected MTU
`Table`	off stops route creation
`PostUp / PostDown`	Shell hooks; %i is iface
`SaveConfig`	Rewrite file on down

[Peer]

Field	Meaning
`PublicKey`	Peer's public key
`AllowedIPs`	IPs sent to peer
`Endpoint`	host:port to dial
`PersistentKeepalive`	Secs; 25 if NAT
`PresharedKey`	Optional extra secret

Minimal /etc/wireguard/wg0.conf

```
[Interface]
PrivateKey = <private key>
Address = 10.0.0.2/24

[Peer]
PublicKey = <peer public key>
Endpoint =
vpn.example.com:51820
AllowedIPs = 10.0.0.0/24
PersistentKeepalive = 25
```

chmod 600 the config: it holds the private key, and wg-quick warns if it is readable by others.

DNS in the file needs resolvconf (or a compatible tool) on the host.

ROUTING

AllowedIPs choices

Value	Effect
`10.0.0.2/32`	Only that one host
`10.0.0.0/24`	A whole subnet
`0.0.0.0/0, ::/0`	All traffic via peer

0.0.0.0/0 makes wg-quick set up policy routing so the default route goes through the tunnel; the endpoint itself stays reachable.

Overlapping AllowedIPs are allowed: the most specific match wins, and an identical range is moved to the later peer.

ROUTING (cont.)

`ip route get <ip>`
Check which path a destination uses

`sudo sysctl -w net.ipv4.ip_forward=1`
Let this host forward packets

`PostUp = iptables -t nat -A POSTROUTING -o <eth> -j MASQUERADE`
NAT clients in config

Full tunnel from a split-tunnel config: set AllowedIPs = 0.0.0.0/0, ::/0 on the client; or Table = off to manage routes yourself.

TROUBLESHOOT

`sudo wg show`
No handshake line means no contact yet

`sudo ss -ulpn | grep <port>`
Is the server listening?

`ping <peer tunnel ip>`
Test the tunnel

`sudo journalctl -u wg-quick@<iface>`
Service log

Handshake but no traffic: check AllowedIPs on both sides, IP forwarding and the firewall.

No handshake: check the UDP port is open, the Endpoint, and that each side has the other's public key, not its own.

RECIPES

Server and one client

```
# server: wg0.conf
[Interface]
Address = 10.0.0.1/24
ListenPort = 51820
PrivateKey = <server private>
[Peer]
PublicKey = <client public>
AllowedIPs = 10.0.0.2/32
```

Add a peer without downtime

```
sudo wg set wg0 peer <pubkey>
allowed-ips 10.0.0.3/32
# persist it
sudo wg-quick save wg0
```

Reload config, keep sessions

```
sudo wg syncconf wg0 <(wg-quick
strip wg0)
```

Route all client traffic

```
[Peer]
PublicKey = <server public>
Endpoint =
vpn.example.com:51820
AllowedIPs = 0.0.0.0/0, ::/0
```