WireGuard Server (Debian)

Notes from diagnosing a Debian WireGuard server with roaming clients (one peer per device) NAT'd out to the internet.

Symptom

The tunnel comes up and `wg show` reports a handshake, but nothing works: no ping to the server, no traffic to the internet, and the handshake counter never moves again.

The handshake is misleading. It is local to the server (`INPUT`), so it completes even when no packet can ever be forwarded. A frozen handshake counter is the consequence, not the fault.

Root Cause

The `FORWARD` chain only had the outbound half:

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

That permits client → internet. Replies from the internet → client match on `-o wg0` instead and were dropped by the default `FORWARD` policy. The missing rule:

```
iptables -A FORWARD -o wg0 -m conntrack --ctstate RELATED,ESTABLISHED -j
ACCEPT
```

Working Config

```
[Interface]
Address = 10.0.0.1/24
ListenPort = 51820

PostUp   = iptables -w -A FORWARD -i %i -j ACCEPT; iptables -w -A FORWARD -o
%i -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT; iptables -t nat -A
POSTROUTING -o eno2 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -m
conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT; iptables -t nat -D
POSTROUTING -o eno2 -j MASQUERADE

[Peer]
# phone
PublicKey = <key>
AllowedIPs = 10.0.0.2/32
```

Points worth keeping:

- Use `%i`, never a literal `wg0`. wg-quick substitutes the interface name.
- `-w` takes the xtables lock. Without it wg-quick races Docker or iptables-persistent at boot and the rules silently fail to apply.
- Do not add `PreUp`/`PreDown` that duplicate `PostUp`. MASQUERADE ends up added twice per start and `iptables -D` removes only one match, so the ruleset drifts on every restart.
- `-o eno2` must match the real WAN interface.
- `PreUp` runs before the interface exists. Anything referencing `%i` belongs in `PostUp`.

ip_forward

wg-quick only sets `net.ipv4.ip_forward` when a peer's `AllowedIPs` contains a default route. All-`/32` peers never trigger it, so set it separately:

```
echo 'net.ipv4.ip_forward=1' > /etc/sysctl.d/99-wireguard.conf
sysctl --system
sysctl net.ipv4.ip_forward
```

Verify

```
ping -c3 10.0.0.1     # client -> server over the tunnel
wg show wg0           # transfer rx and tx must both grow
ping -c3 1.1.1.1      # client -> internet through MASQUERADE
```

Both directions must move. rx without tx is the signature of the `FORWARD` bug above.

Regression: "it worked before"

Something else used to supply the stateful accept. Docker is the usual suspect: it inserts `RELATED,ESTABLISHED` into `FORWARD` and can set the policy to `DROP`, so removing or upgrading it takes the forwarding with it.

```
iptables -S FORWARD | grep -i docker
```

On Debian also confirm the backend has not split in two:

```
iptables -V
update-alternatives --display iptables
```

Debian 12 defaults `iptables` to `nf_tables`. If `iptables-persistent` restored `/etc/iptables/rules.v4` into the legacy backend, rules added by `PostUp` land in a different table than the restored ones and nothing matches.

Keepalive

`PersistentKeepalive` belongs on the `'client'`, not the server. A WireGuard server never initiates a handshake, it only answers. A client behind NAT with no keepalive goes quiet after one handshake, which looks like a dead tunnel but is normal.

```
[Peer]
PersistentKeepalive = 25
```

From:
https://memex.kyaruc.moe/ - **kyaruc memex**

Permanent link:
https://memex.kyaruc.moe/quickref:wireguard

Last update: **2026-11-19 Thu 03:28**