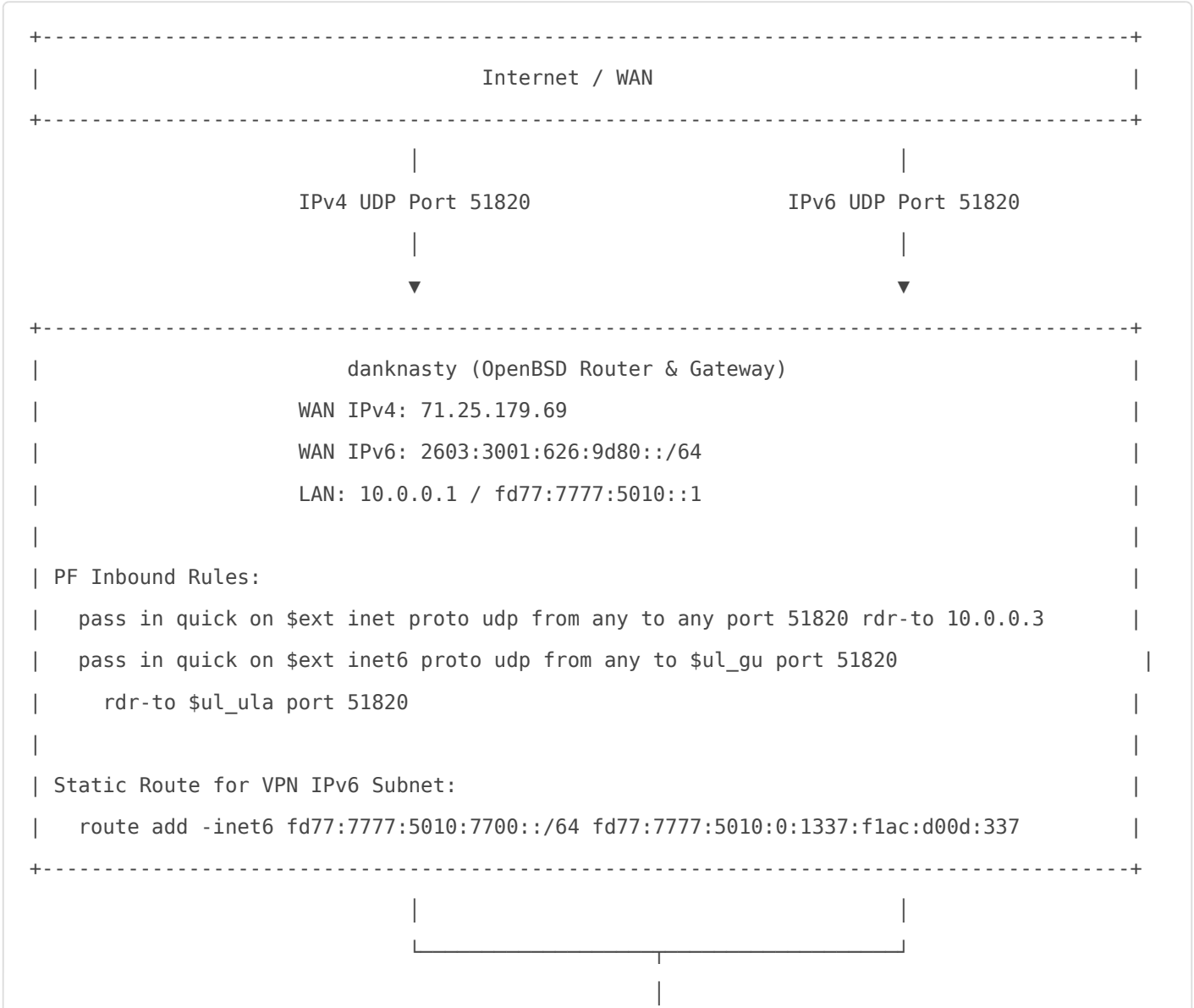


VPN

Overview & Architecture

This document describes the modern, high-performance **WireGuard VPN** service running on the internal infrastructure server `ul` (`10.0.0.3`) with full **IPv4 and IPv6 dual-stack** pass-through from the edge router `danknasty` (OpenBSD `pf`).

Network Topology



Router (danknasty) PF & Route Configuration

Tracked in `/home/luke/git/router_config/pf.conf`:

```
# Ports
wireguard = "51820"

# IPv4 Inbound
pass in quick on $ext inet proto udp from any to any port $wireguard rdr-to $ul_ipv4

# IPv6 Inbound
pass in quick on $ext inet6 proto udp from any to $ul_gu port $wireguard rdr-to $ul_ula port
$wireguard
```

IPv6 routing in `/home/luke/git/router_config/hostname.em1`:

```
!route add -inet6 fd77:7777:5010:7700::/64 fd77:7777:5010:0:1337:f1ac:d00d:337
```

DNS Access Control in `/home/luke/git/router_config/unbound.conf`:

```
access-control: 10.77.0.0/24 allow
access-control: fd77:7777:5010:7700::/64 allow
```

Server (ul) Interface Configuration: `/etc/wireguard/wg0.conf`

```
[Interface]
Address = 10.77.0.1/24, fd77:7777:5010:7700::1/64
ListenPort = 51820
PrivateKey = <SERVER_PRIVATE_KEY>

# Dual-stack packet forwarding and NAT masquerade
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables
-t nat -A POSTROUTING -s 10.77.0.0/24 -o eth77 -j MASQUERADE; ip6tables -A FORWARD -i wg0 -j
ACCEPT; ip6tables -A FORWARD -o wg0 -j ACCEPT; ip6tables -t nat -A POSTROUTING -s
fd77:7777:5010:7700::/64 -o eth77 -j MASQUERADE
```

```
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D FORWARD -o wg0 -j ACCEPT;
iptables -t nat -D POSTROUTING -s 10.77.0.0/24 -o eth77 -j MASQUERADE; ip6tables -D FORWARD -i
wg0 -j ACCEPT; ip6tables -D FORWARD -o wg0 -j ACCEPT; ip6tables -t nat -D POSTROUTING -s
fd77:7777:5010:7700::/64 -o eth77 -j MASQUERADE
```

```
# Peer entry template

[Peer]

PublicKey = <CLIENT_PUBLIC_KEY>

PresharedKey = <PRESHARED_KEY>

AllowedIPs = 10.77.0.X/32, fd77:7777:5010:7700::X/128
```

Client Management Utilities

Management scripts are installed in `/home/luke/bin/` on `ul`:

1. Adding Clients (`wg-addclient`)

```
# Syntax: wg-addclient <device-name> [optional_ip_octet]

# IP is automatically assigned to the next available address if omitted:

wg-addclient macbook

wg-addclient iphone


# Or manually specify an IP octet if preferred:

wg-addclient custom-device 10
```

This utility:

- Automatically allocates the next free IPv4 & IPv6 addresses.
- Generates private/public keypairs + Pre-shared Key (PSK).
- Creates the client configuration in `/etc/wireguard/clients/<name>.conf`.
- Injects the peer into `/etc/wireguard/wg0.conf` and live kernel interface dynamically.
- Renders an ASCII QR code in the terminal for instant mobile setup.

2. Listing Active Clients (`wg-listclients`)

```
wg-listclients
```

Prints all configured clients, their assigned IP addresses, and their WireGuard public keys.

3. Revoking / Removing Clients (`wg-delclient`)

```
# Syntax: wg-delclient <device-name>
wg-delclient macbook
```

This utility:

- Instantly drops the peer from the live in-kernel WireGuard routing table (`sudo wg set wg0 peer <pubkey> remove`), severing all active sessions.
- Removes the `[Peer]` block from `/etc/wireguard/wg0.conf`.
- Deletes the stored configuration file `/etc/wireguard/clients/<name>.conf`.

To display a QR code for mobile devices:

```
sudo qrencode -t ansiutf8 < /etc/wireguard/clients/<name>.conf
```

Client Setup Instructions (All Platforms)

1. Apple macOS & iOS

Setup via Mobile / QR Code (iOS)

1. Install **WireGuard** from the App Store.
2. Open the app, tap + -> **Create from QR code**.
3. On `ul`, run `sudo qrencode -t ansiutf8 < /etc/wireguard/clients/<client-name>.conf` and scan the terminal screen.
4. Name the tunnel `drheck` and toggle to connect.

Setup via Desktop App (macOS)

1. Install **WireGuard** from the Mac App Store.
2. Copy the `.conf` file from `/etc/wireguard/clients/<name>.conf` to your Mac.

3. Open WireGuard, click **Import tunnel(s) from file**, and select the `.conf` file.
 4. Click **Activate**.
-

2. Linux (systemd-networkd / NetworkManager / wg-quick)

Method A: wg-quick

1. Install `wireguard-tools`:
 - Arch Linux: `sudo pacman -S wireguard-tools`
 - Debian/Ubuntu: `sudo apt install wireguard-tools`
 - Fedora: `sudo dnf install wireguard-tools`
2. Save your configuration as `/etc/wireguard/drheck.conf`:

```
[Interface]
PrivateKey = <CLIENT_PRIVATE_KEY>
Address = 10.77.0.X/24, fd77:7777:5010:7700::X/64
DNS = 10.0.0.1, fd77:7777:5010::1

[Peer]
PublicKey = ubh+z69hcWcUKgHl3LVfm54sYkDZAaocIKNN5zlsZWA=
PresharedKey = <PRESHARED_KEY>
Endpoint = 71.25.179.69:51820
AllowedIPs = 0.0.0.0/0, ::/0
PersistentKeepalive = 25
```

3. Bring the tunnel up:

```
sudo wg-quick up drheck
# To enable on boot:
sudo systemctl enable --now wg-quick@drheck
```

Method B: NetworkManager

```
sudo nmcli connection import type wireguard file /path/to/<name>.conf
sudo nmcli connection up <name>
```

3. Android

1. Install **WireGuard** from Google Play Store or F-Droid.
2. Tap the floating **+** button in the bottom-right corner.
3. Select **Scan from QR code** and scan the output of:

```
sudo qrencode -t ansiutf8 < /etc/wireguard/clients/<name>.conf
```

4. Enter tunnel name `drheck` and tap to activate.
-

4. Microsoft Windows

1. Download and run the official installer from wireguard.com/install.
 2. Click **Add Tunnel** (or press `Ctrl+0`) and open `<name>.conf`.
 3. Click **Activate**.
-

5. OpenBSD / FreeBSD

OpenBSD (`hostname.wg0`)

Create `/etc/hostname.wg0`:

```
inet 10.77.0.X 255.255.255.0 NONE
inet6 fd77:7777:5010:7700::X 64
wgkey <CLIENT_PRIVATE_KEY>
wgpeer ubh+z69hcWcUKgHl3LVfm54sYkDZAocIKNN5zlsZWA= \
  wgresharedkey <PRESHARED_KEY> \
  wgendpoint 71.25.179.69 51820 \
  wgaip 0.0.0.0/0 wgaip ::/0
up
```

Run `sh /etc/netstart wg0`.

Split Tunneling vs Full Tunneling

By default, the client configuration routes all traffic (`AllowedIPs = 0.0.0.0/0, ::/0`) through the home network connection.

If you only want access to the internal `drheck.dev` LAN resources and DNS without tunneling general internet browsing, adjust `AllowedIPs` in the client's `[Peer]` section:

```
# Split Tunnel (Internal LAN & VPN Subnet Only)
AllowedIPs = 10.0.0.0/24, 10.77.0.0/24, fd77:7777:5010::/48
```

Backup & Disaster Recovery

WireGuard configuration, keys, and client management scripts are fully integrated into the host backup architecture under `~/git/backups/ul/scripts/`:

Automated Backup (`scripts/backup/wireguard`)

- **Archive Path:**

`/data/backups/services/current/wireguard/<TIMESTAMP>/wireguard_config.tar.xz`

- **Included Paths:**

- `/etc/wireguard/` (Server config `wg0.conf` and all client profiles in `clients/`)
- `/etc/modules-load.d/wireguard.conf`
- `/home/luke/bin/wg-addclient` , `/home/luke/bin/wg-delclient` , `/home/luke/bin/wg-listclients`

- **Parity Redundancy:** 10% PAR2 recovery volume generation (`par2create -r10`).

- **Cloud Replication:** Automatically bundled by `stage_cloud_archive` into GPG-encrypted cloud staging for off-site sync.

Disaster Recovery (`scripts/restore/wireguard`)

To restore the complete WireGuard service from the latest backup:

```
sudo /home/luke/git/backups/ul/scripts/restore/wireguard
```

This verifies PAR2 parity, unpacks files with intact permissions (`0700` / `0600`), ensures the kernel module is loaded, and enables/starts `wg-quick@wg0`.

Revision #4

Created 2026-09-26 11:51:46 UTC by luKe

Updated 2026-09-26 12:23:14 UTC by luKe