AI-04503 OnPrem vs GCP Diagram

How would I connect my NUC dockerlab Ubuntu server to my home network? Connect it my home router via rj45? How would I access my dockerlab from my notebook anywhere in the world if I'm traveling.

Setting up a home Docker lab on an Intel NUC and ensuring secure access while traveling is a classic, highly reliable home lab setup.

Here is how to structure both your local network connectivity and your remote access architecture.

1. Local Network Connection

Direct Ethernet (RJ45) to your router is the gold standard.

While the NUC likely has built-in Wi-Fi, Ethernet is vastly superior for a server environment for three main reasons:

- **Lower Latency & Higher Throughput:** Essential for transferring large container images, database backups, or file media.
- **Reliability:** Wi-Fi connection drops will disrupt running containers, active SSH sessions, or proxy connections.
- **Boot-Time Networking:** Ethernet interfaces initialize faster and more reliably during server reboots without relying on wpa_supplicant or Wi-Fi credentials.

Network Configuration Best Practice

1. **Physical Connection:** Plug an Cat5e or Cat6 Ethernet cable from the NUC's RJ45 port directly into an open LAN port on your primary home router.
2. **Static IP Assignment (DHCP Reservation):**
   Instead of hardcoding a static IP inside Ubuntu's `netplan` (which can cause IP conflicts if you swap routers), log into your home router's admin interface and create a **DHCP MAC Reservation** for the NUC's Ethernet MAC address.
   - This ensures your router always assigns the exact same local IP (e.g., `192.168.1.100`) to the NUC every time it boots.

2. Remote Access While Traveling

To access your NUC from your notebook from anywhere in the world, you have two primary architectural options: **Overlay Mesh VPNs** (recommended) or a **Reverse Proxy with Cloudflare**.

Option A: Overlay Mesh VPN (Tailscale or NetBird) — *Most Secure & Easiest*

For private, administrative access (SSH, Portainer, phpMyAdmin, internal web UI dashboards), an overlay mesh network built on **WireGuard** is the safest approach. You do **not** need to open any ports on your home router.

```
[Notebook on Travel Wi-Fi] ──(Encrypted WireGuard Tunnel)──> [NUC in Home Lab]
```

How it works:

1. **Install Tailscale (or NetBird)** on your Ubuntu NUC and on your travel notebook.
2. Log both devices into the same control account.
3. Tailscale creates an encrypted point-to-point tunnel between your notebook and NUC using WireGuard.
4. Your NUC receives a stable 100.x.x.x private IP. From your notebook anywhere in the world, you can connect to `100.x.x.x:8080` or SSH directly using `ssh user@100.x.x.x`.

> **Why this is best for traveling:** It exposes zero ports to the public internet, meaning zero exposure to automated port scanners or brute-force attacks while you are away.

Option B: Cloudflare Tunnels (Zero Trust) — *Best for Domain Names*

If you want to access your web applications via custom subdomains (e.g., `portainer.yourdomain.com` or `app.yourdomain.com`) without turning on a VPN client every time, **Cloudflare Tunnels** (`cloudflared`) is the modern standard.

```
[Notebook Browser] ──> [Cloudflare Edge] ──(Outbound Tunnel)──> [cloudflared Container on NUC] ──> [Docker Services]
```

How it works:

1. Run the `cloudflared` agent as a light Docker container on your NUC.
2. The container establishes an **outbound-only** connection to Cloudflare's edge network.
3. You point your domain's DNS routes inside Cloudflare to route traffic through the tunnel down to your local Docker containers or reverse proxy (e.g., Nginx Proxy Manager).
4. **Security Layer:** You can wrap administrative routes behind **Cloudflare Zero Trust / Access policies**, forcing a One-Time Password (OTP) sent to your email before letting anyone reach the login screen of your applications.

Recommended Next Steps

Depending on your goal, you can combine both:

- Use **Tailscale** for direct, raw SSH access and container management.
- Use **Cloudflare Tunnels** with a reverse proxy for smooth HTTP/HTTPS web app access via custom domain names.