VPN Setup

Project title: Secure Home VPN with WireGuard and Quad9 DNS provider

Name: Matheo

Date : June

---

2. Table of Contents

Table of Contents

---

1. Project Overview

Goal: What am I building and why?

The aim of this project is to create a secure remote access VPN using WireGuard on a Raspberry Pi.

Use case: Access home network from anywhere and privately

---

2. Architecture Diagram

- **Client connects to VPN using WireGuard:**
  - Your phone/laptop connects to your home Raspberry Pi using WireGuard.

    - This tunnel is encrypted end-to-end.
    - Port forwarding was configured on the router so even if you're outside the house I would be able to connect to the raspberry pi
- **Client has DNS set to 9.9.9.9 (Quad9):**
    - You manually specified this in the WireGuard config.
    - So **all DNS requests** (e.g. `google.com`) go to Quad9.
- **DNS and web traffic travel through the encrypted VPN tunnel:**
    - Everything from your client gets **tunneled into your Pi** at home.
    - This includes:
    - DNS requests → 9.9.9.9 (Quad9)
    - HTTPS/HTTP requests → websites
- **Raspberry Pi receives the traffic:**
    - It acts as the VPN gateway.
    - It **forwards DNS traffic to Quad9**, and other traffic to its final destination (e.g. websites).
- **Pi sends all outgoing traffic to your ISP (e.g. Telstra):**
    - ISP sees traffic **coming from the Pi**, not from the original device.
    - DNS requests go to Quad9.
    - Other internet traffic goes to respective sites (encrypted if HTTPS).

---

3. Hardware & Software

- Raspberry Pi model (Raspberry Pi 5 8GB RAM)
- Micro HDMI
- Mouse
- Keyboard
- SD Card (Preferably 128GB)
- Monitor
- Power Supply

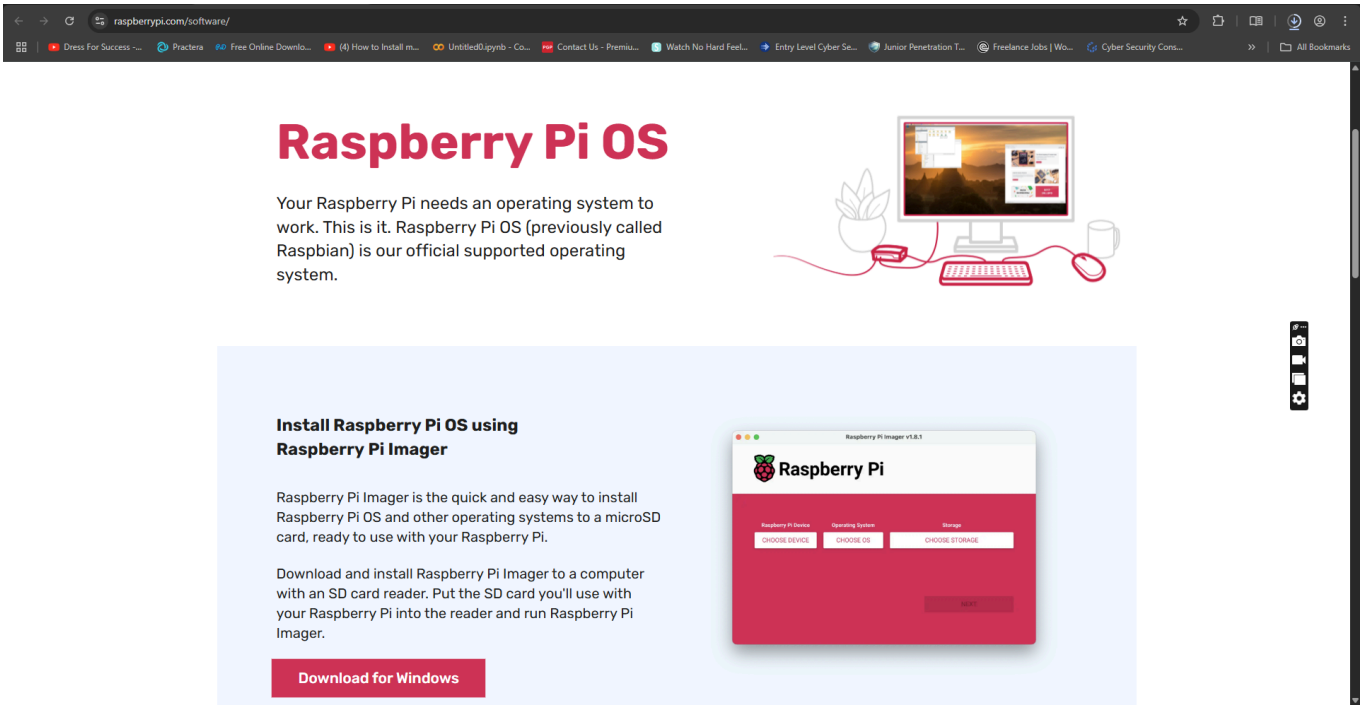
- OS : Raspberry Pi OS 64-bit

Applications Installed: WireGuard

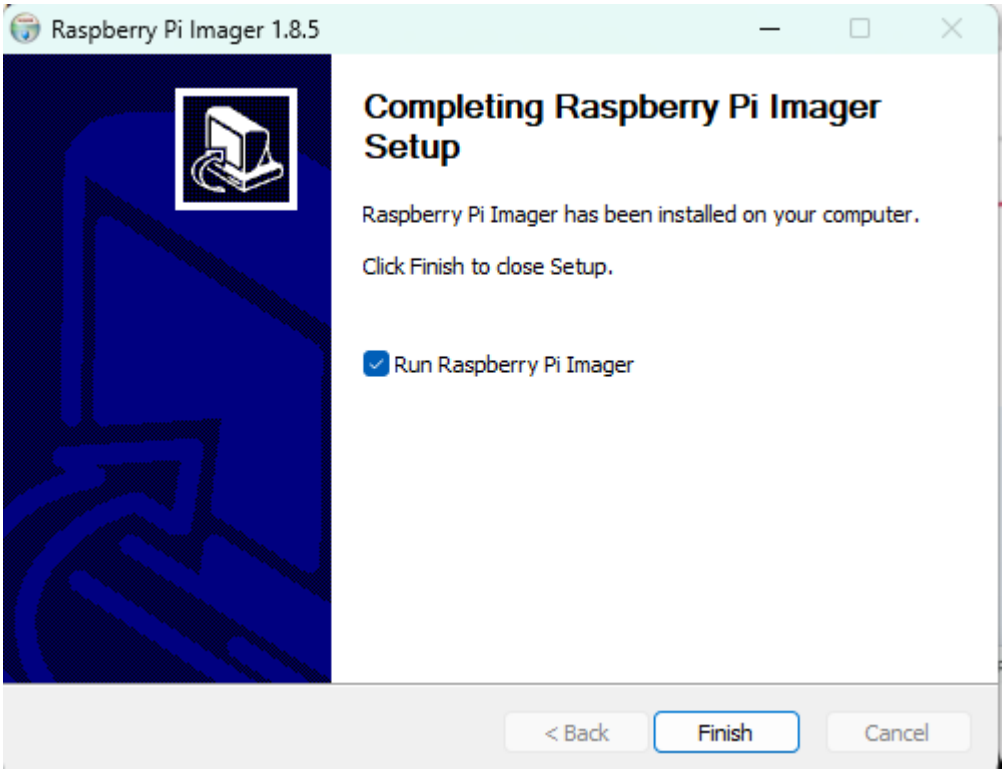
4. Configuration Steps

A. Raspberry Pi Setup

Downloading the Raspberry Pi OS from the main website:
https://www.raspberrypi.com/software/

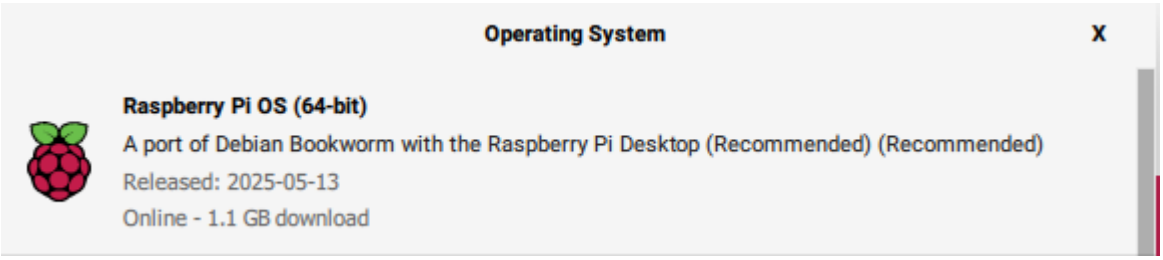


Successfully installing Raspberry Pi Imager, the utility I'll use to flash the OS onto our SD card.

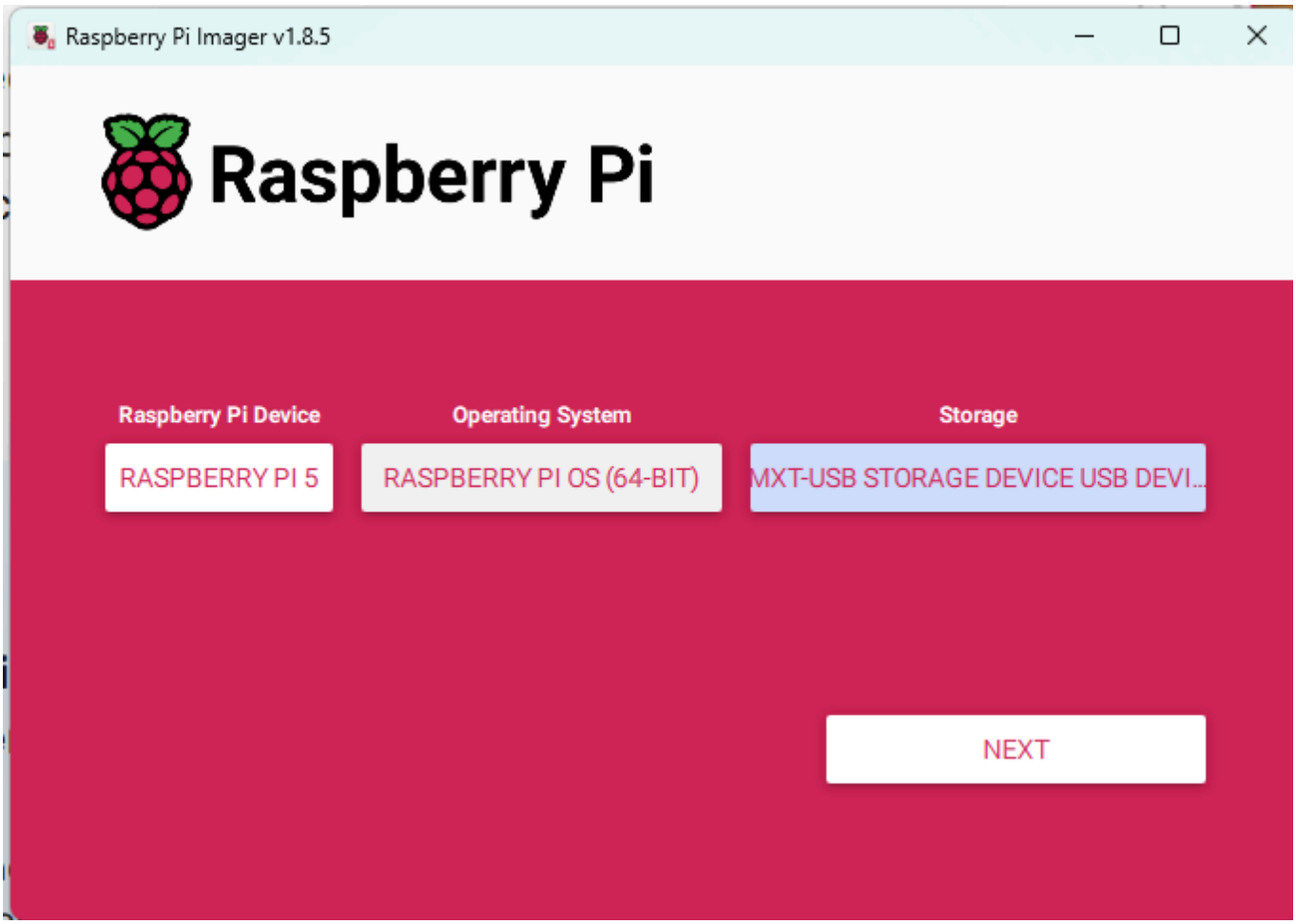


Choosing Device, OS, and Storage in Imager. Selecting the correct hardware (Raspberry Pi 5), preferred operating system (64-bit OS), and target SD card to prepare the bootable system image.

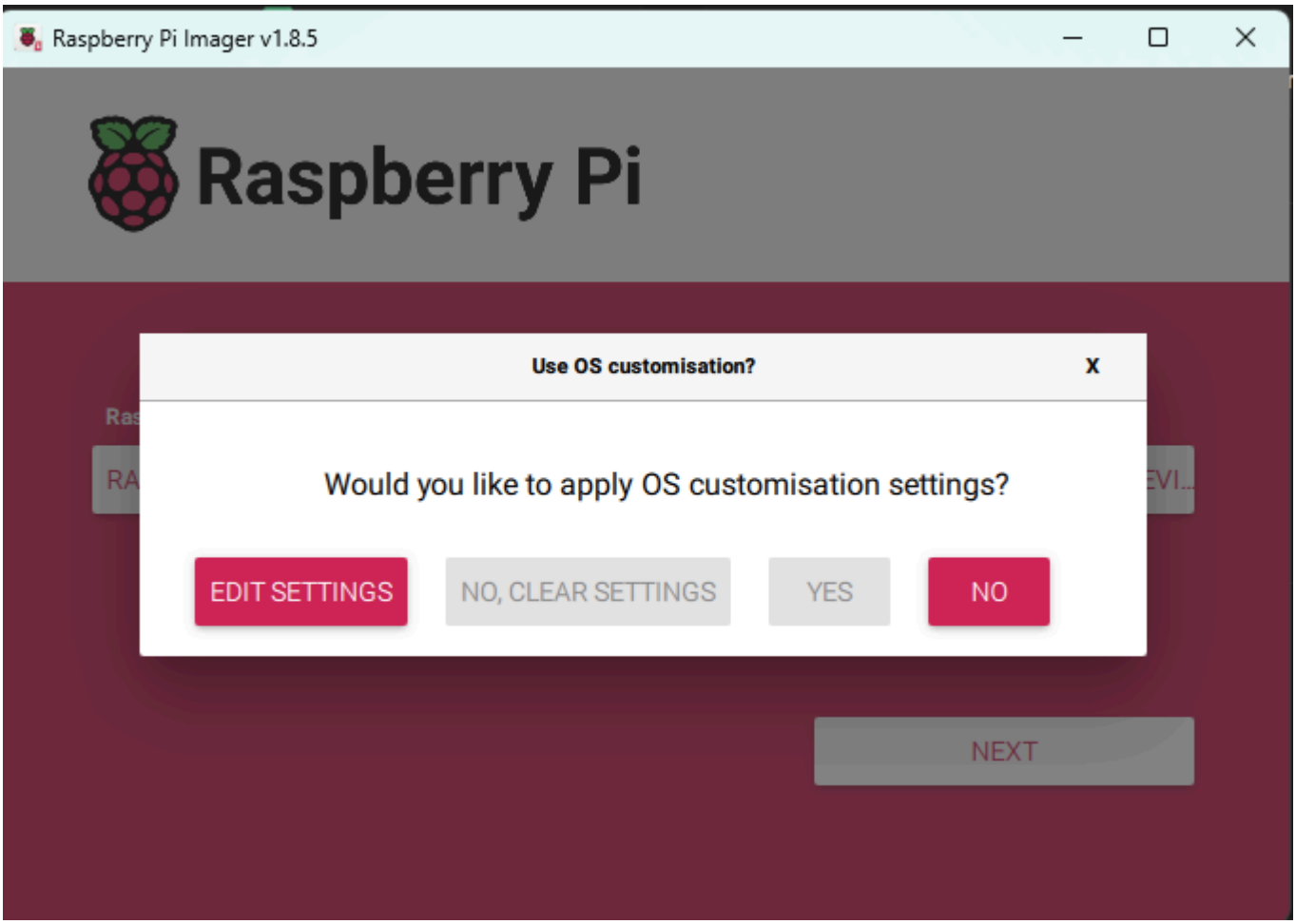
The Raspberry os 64 bit is the os I will be choosing



All chosen:

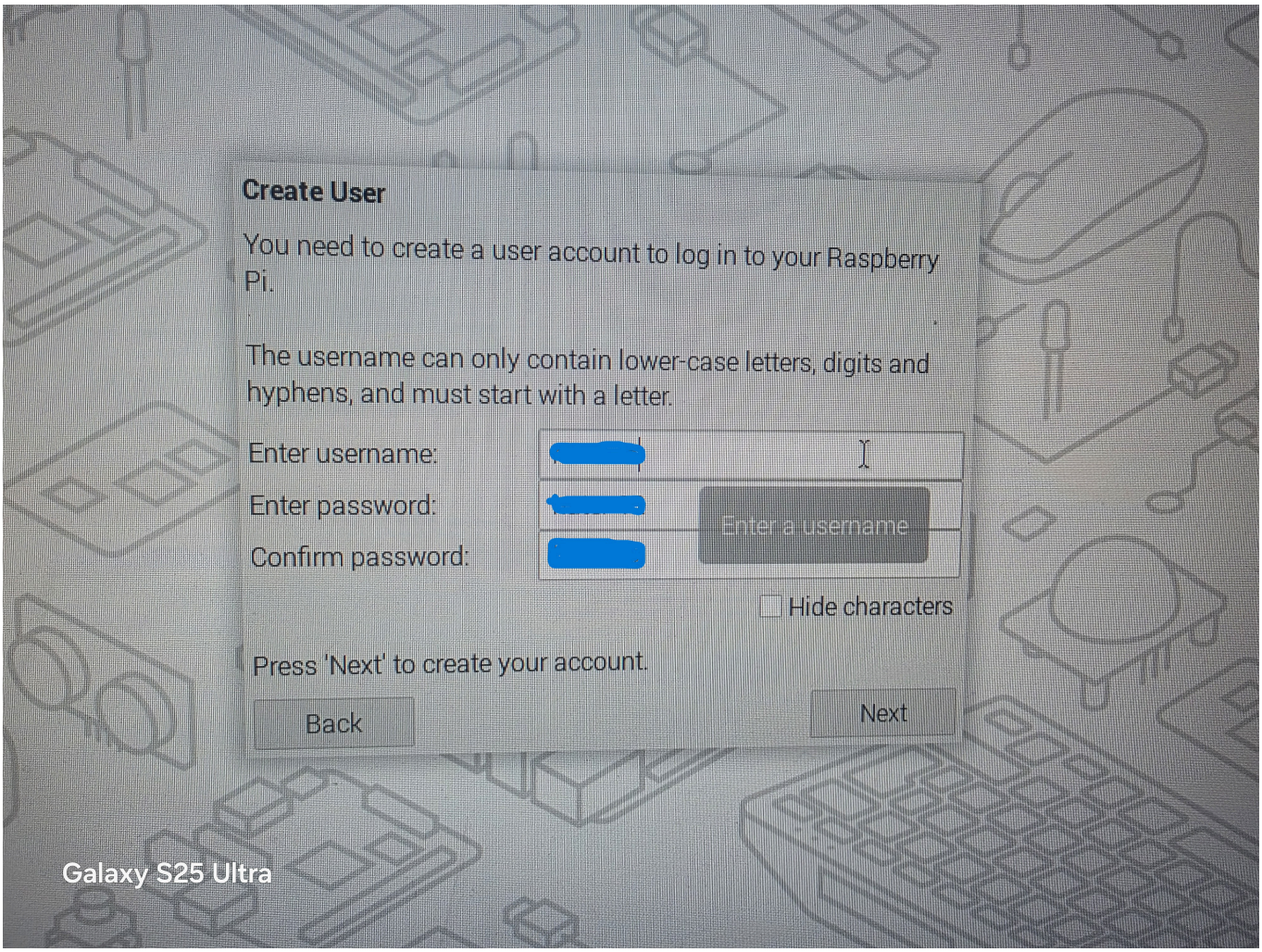


Prompt for OS Customization (Skipped) . Choosing to skip automated OS customizations so I can configure key settings like Wi-Fi, hostname, and user credentials manually later.



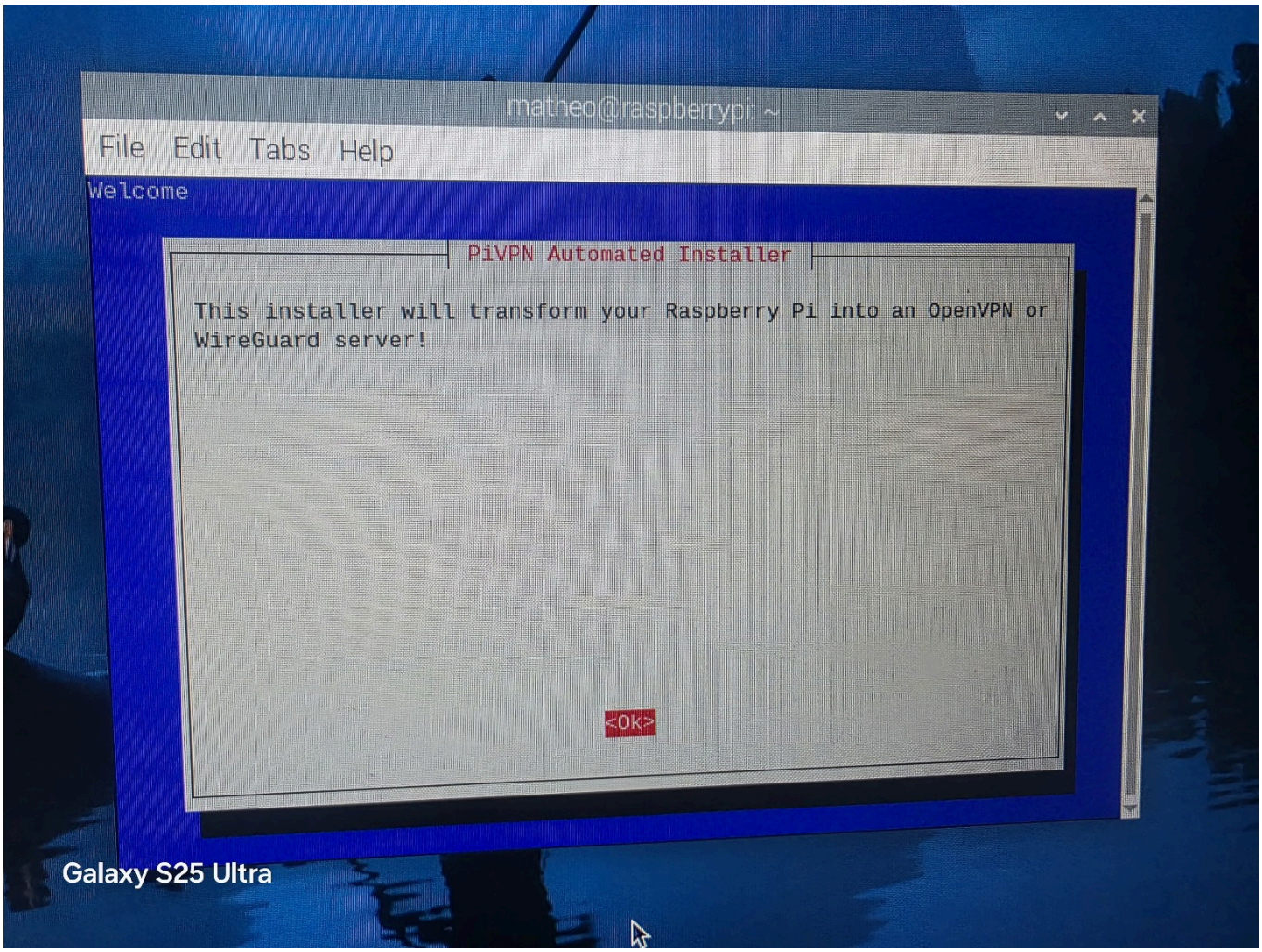
B. Raspberry Pi User Setup

Creating a New User Account . Manually setting up a secure user account that will manage the Raspberry Pi and VPN—ensuring tight control over admin privileges.

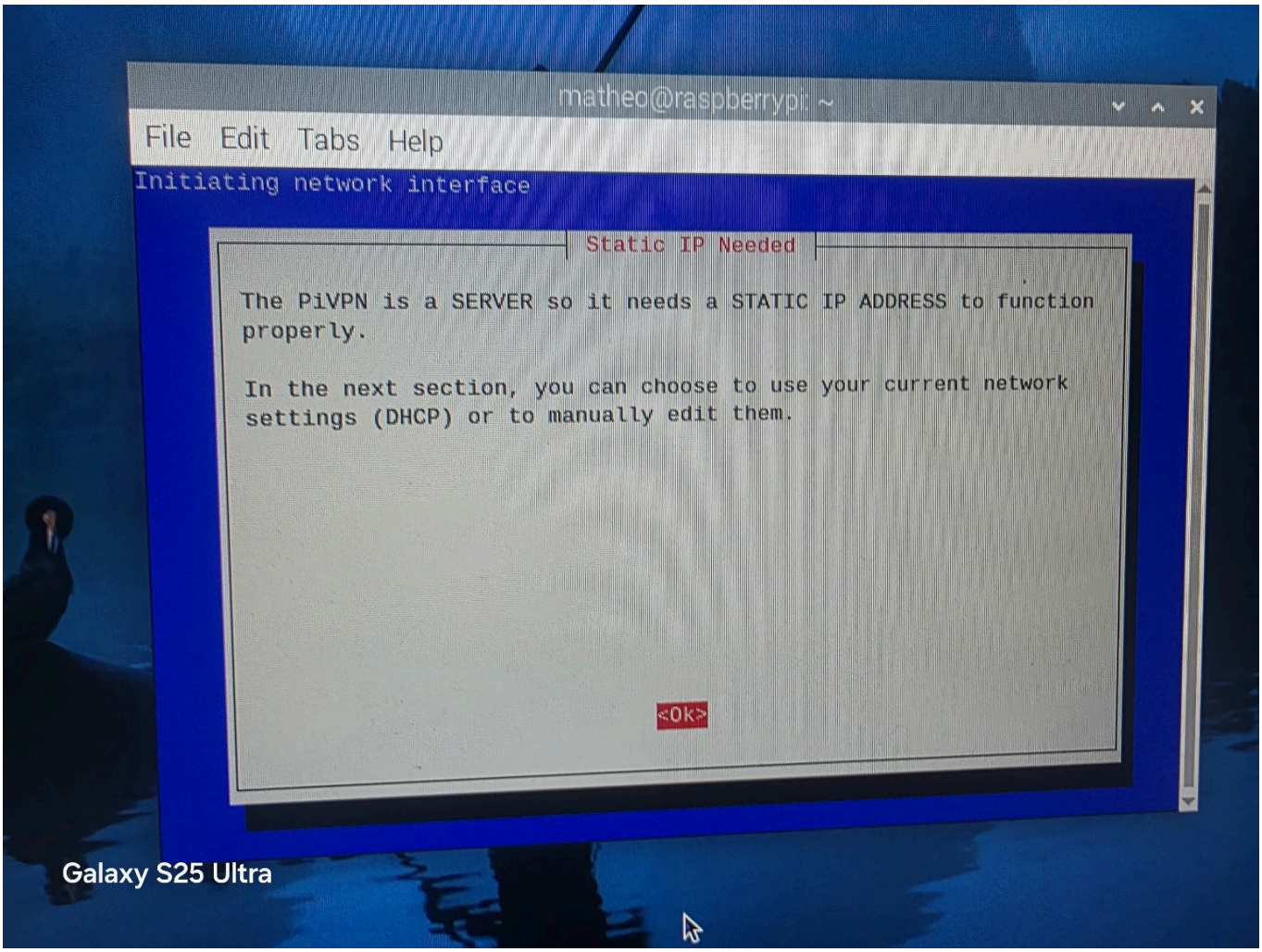


C. VPN Setup

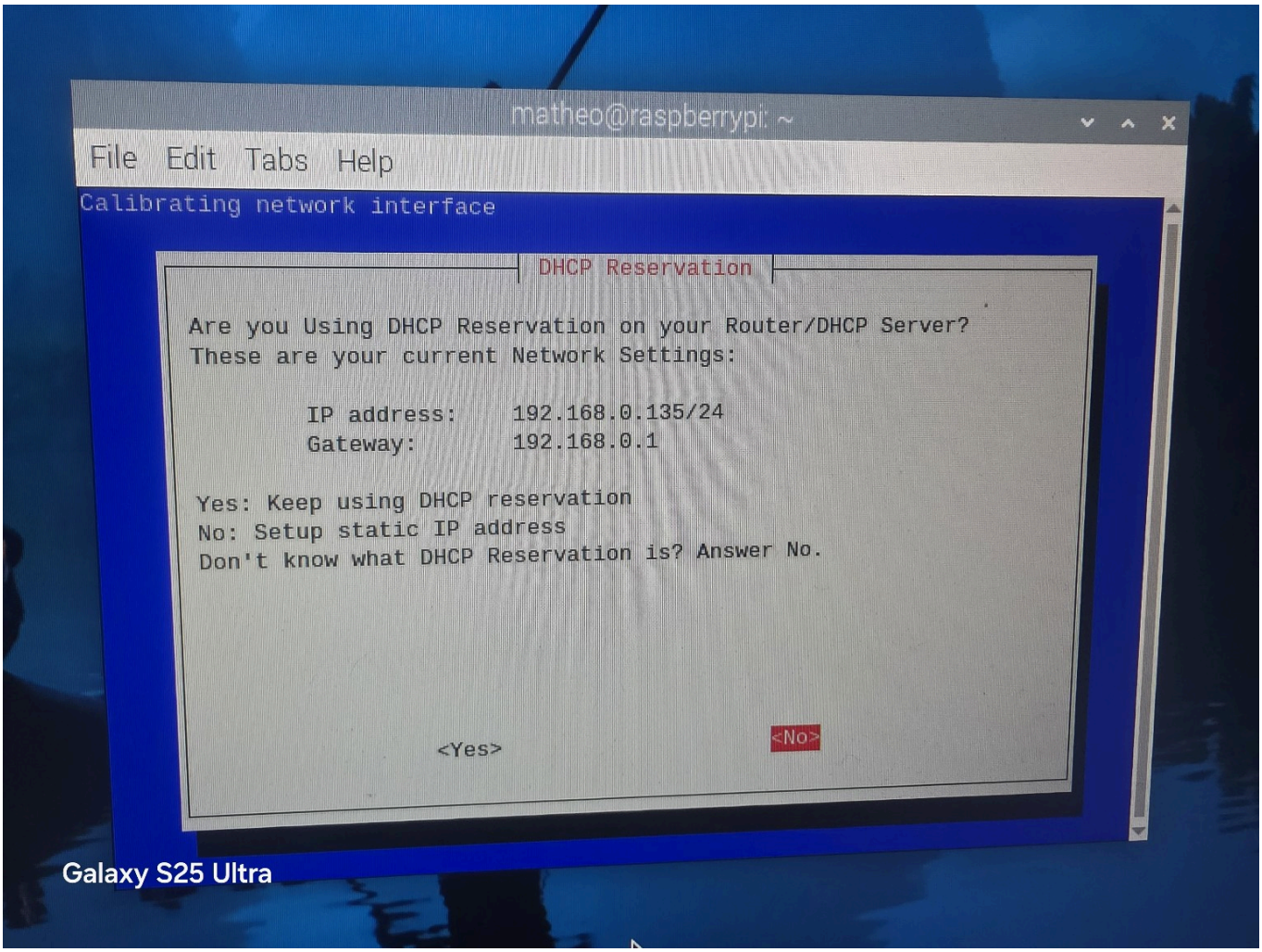
Launching PiVPN Installer from Terminal . Executing the PiVPN installation script, which automates setup of WireGuard on the Raspberry Pi for secure remote access.
curl -L https://install.pivpn.io | bash



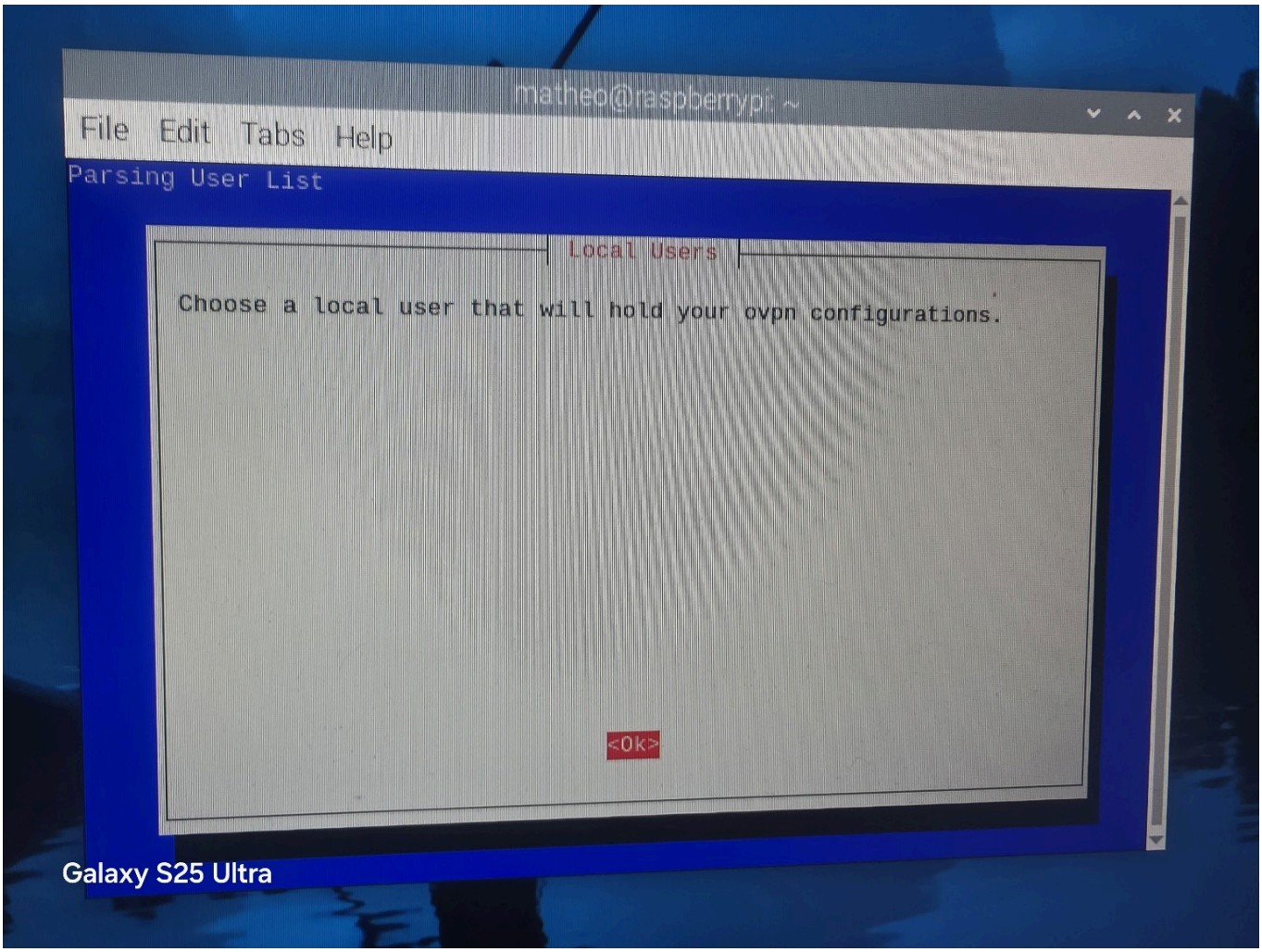
Network Interface Prompt for IP Addressing . Installer checks for static IP address requirements—critical so VPN clients can always reach the server reliably.



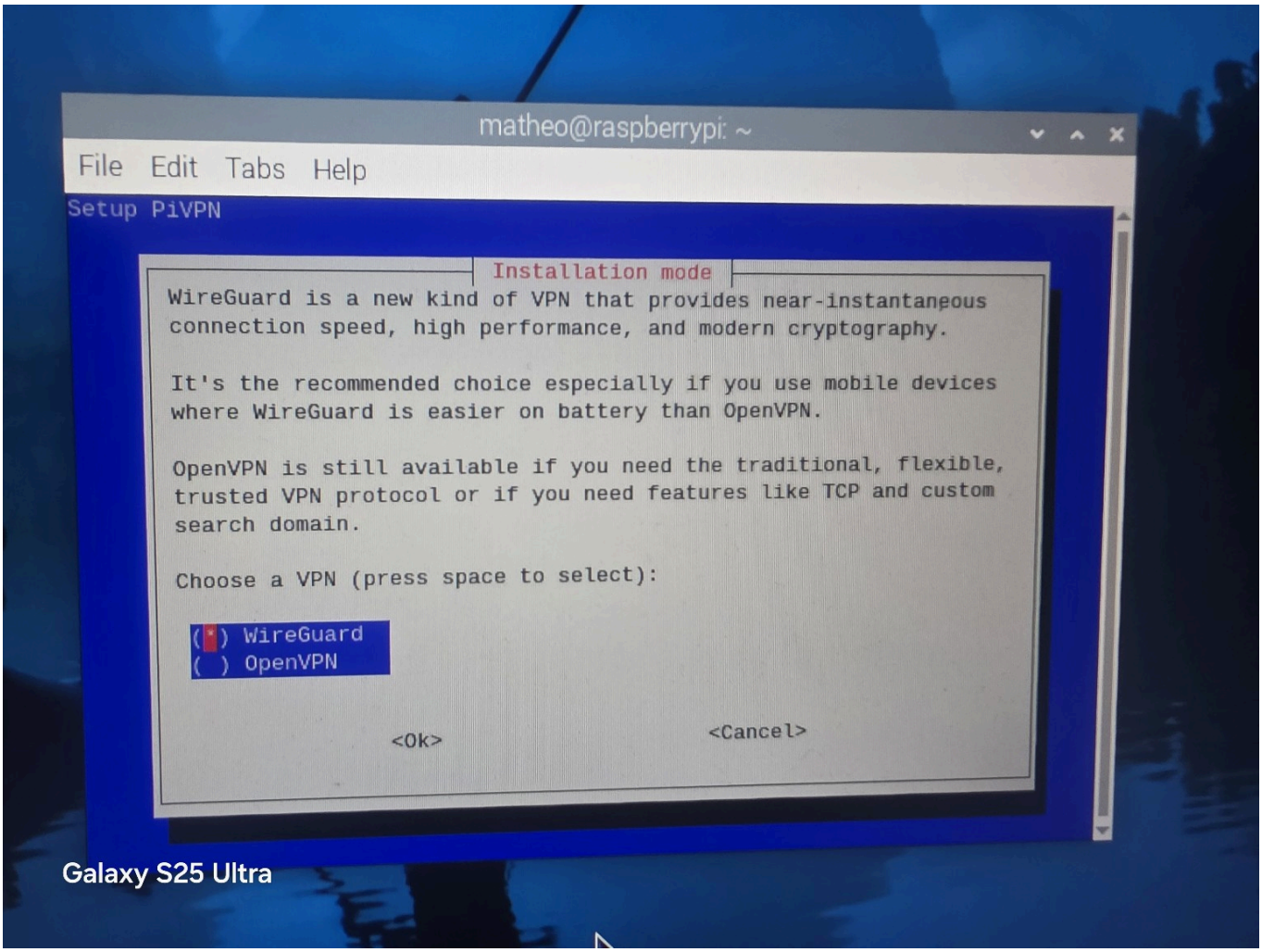
Accepting DHCP Reservation Setup . Retaining DHCP reservation on router to preserve consistent IP settings without switching to static manually.



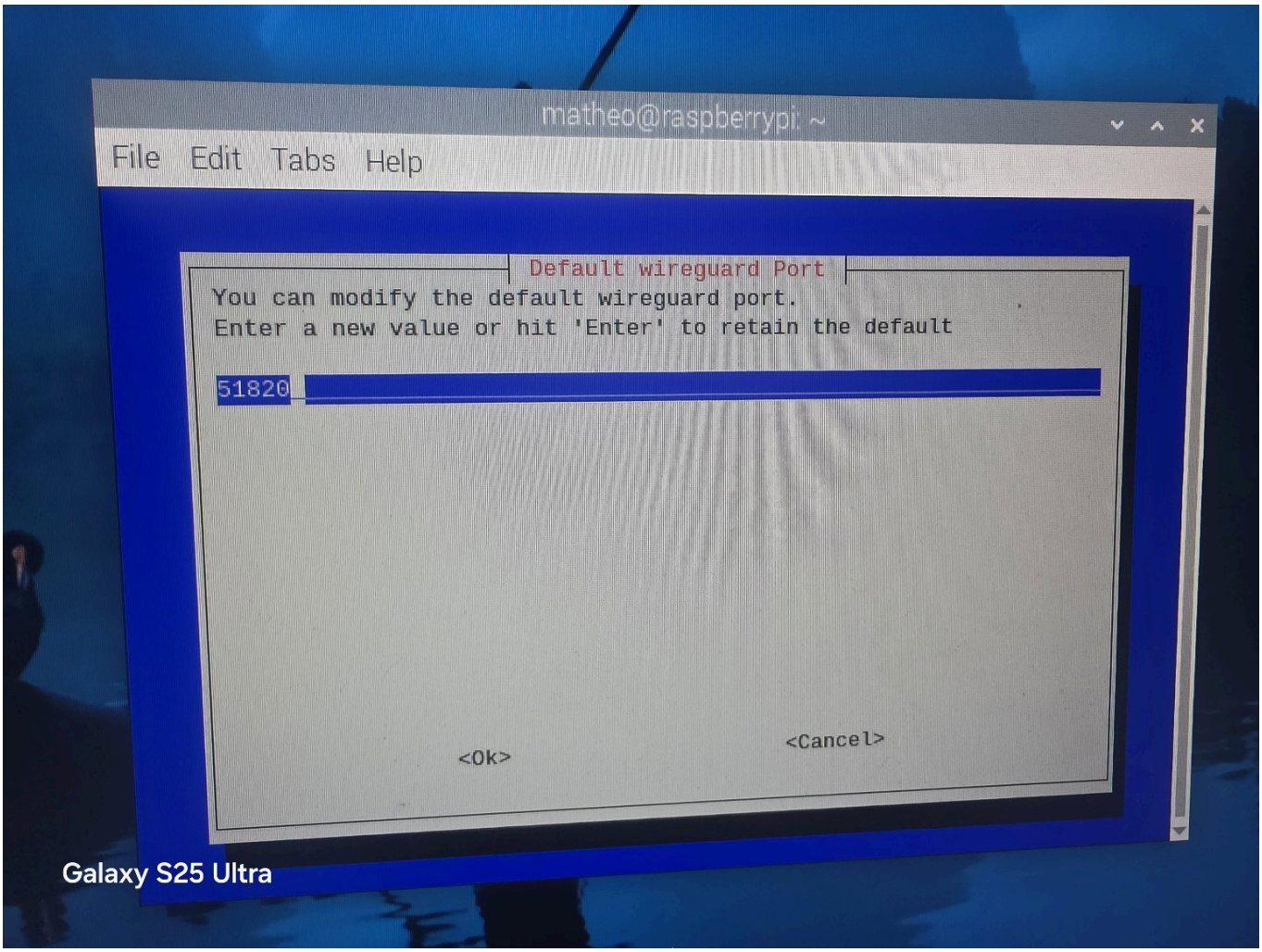
Selecting Raspberry Pi User for VPN Configs . Assigning a system user to host WireGuard VPN configurations—this centralizes security credentials and management.



Choosing VPN Protocol: WireGuard . Opting for WireGuard due to its high-performance encryption, fast handshakes, and mobile-friendliness over legacy protocols like OpenVPN.

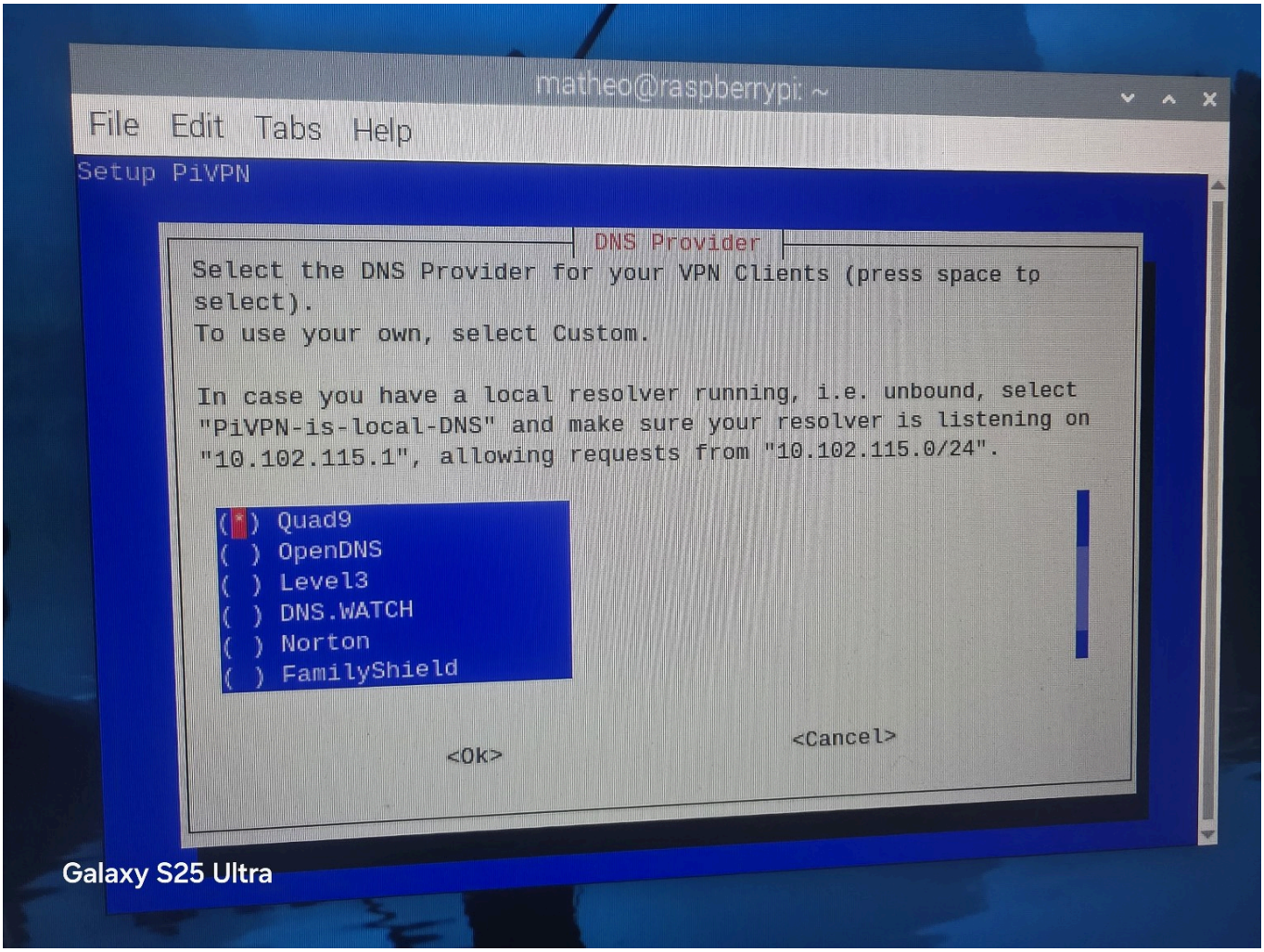


Confirming Default Port (51820) . Retaining WireGuard’s default UDP port—standardized and recognized by most router configurations and VPN apps.

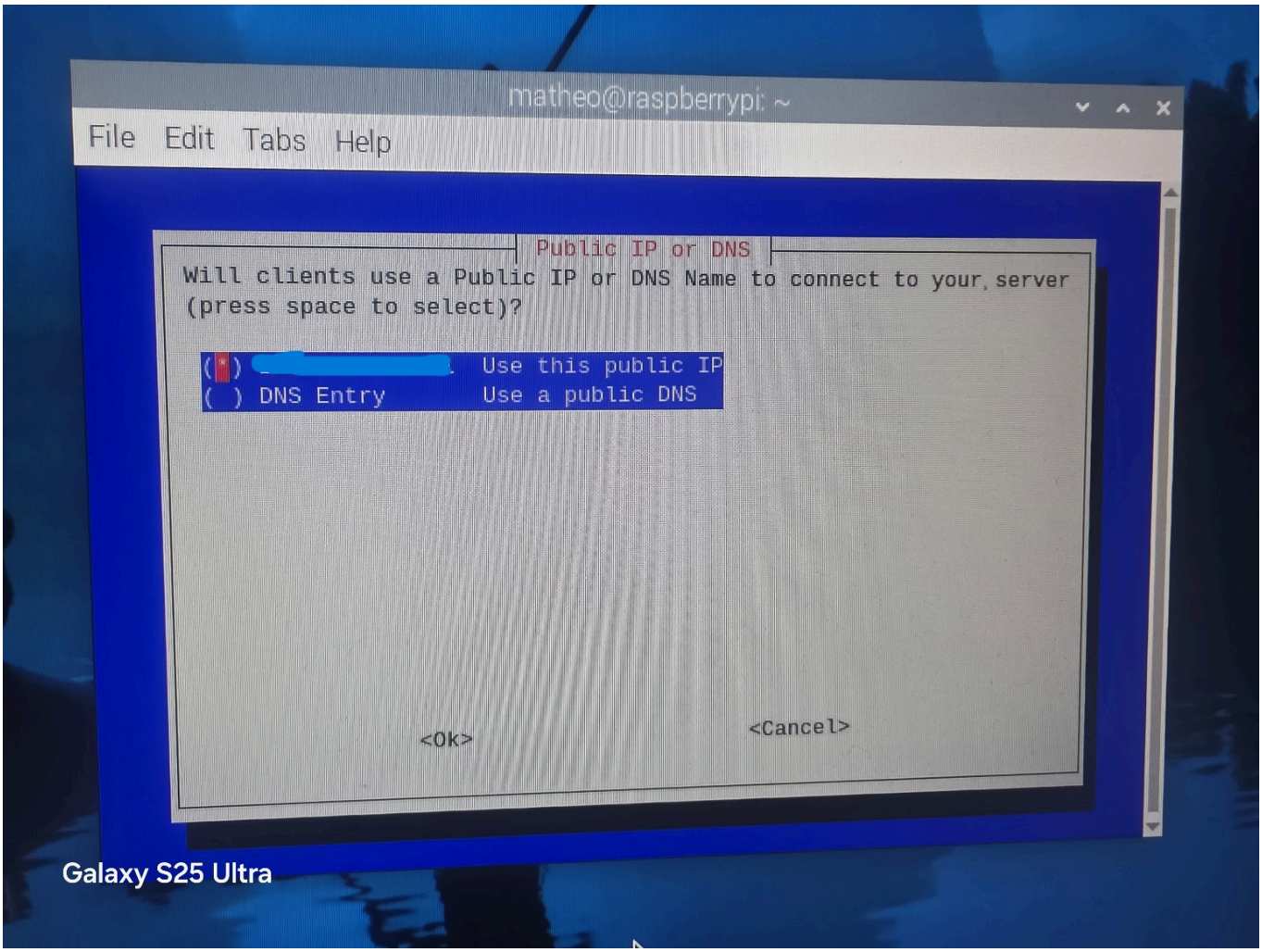


Choosing DNS Provider (Quad9) . Configuring DNS to use Quad9, a privacy-focused resolver that blocks access to malicious domains and preserves query confidentiality. This will work for VPN Clients only and not the Raspberry Pi itself so in order to have Raspberry pi use quad9 I

would have to configure it manually.

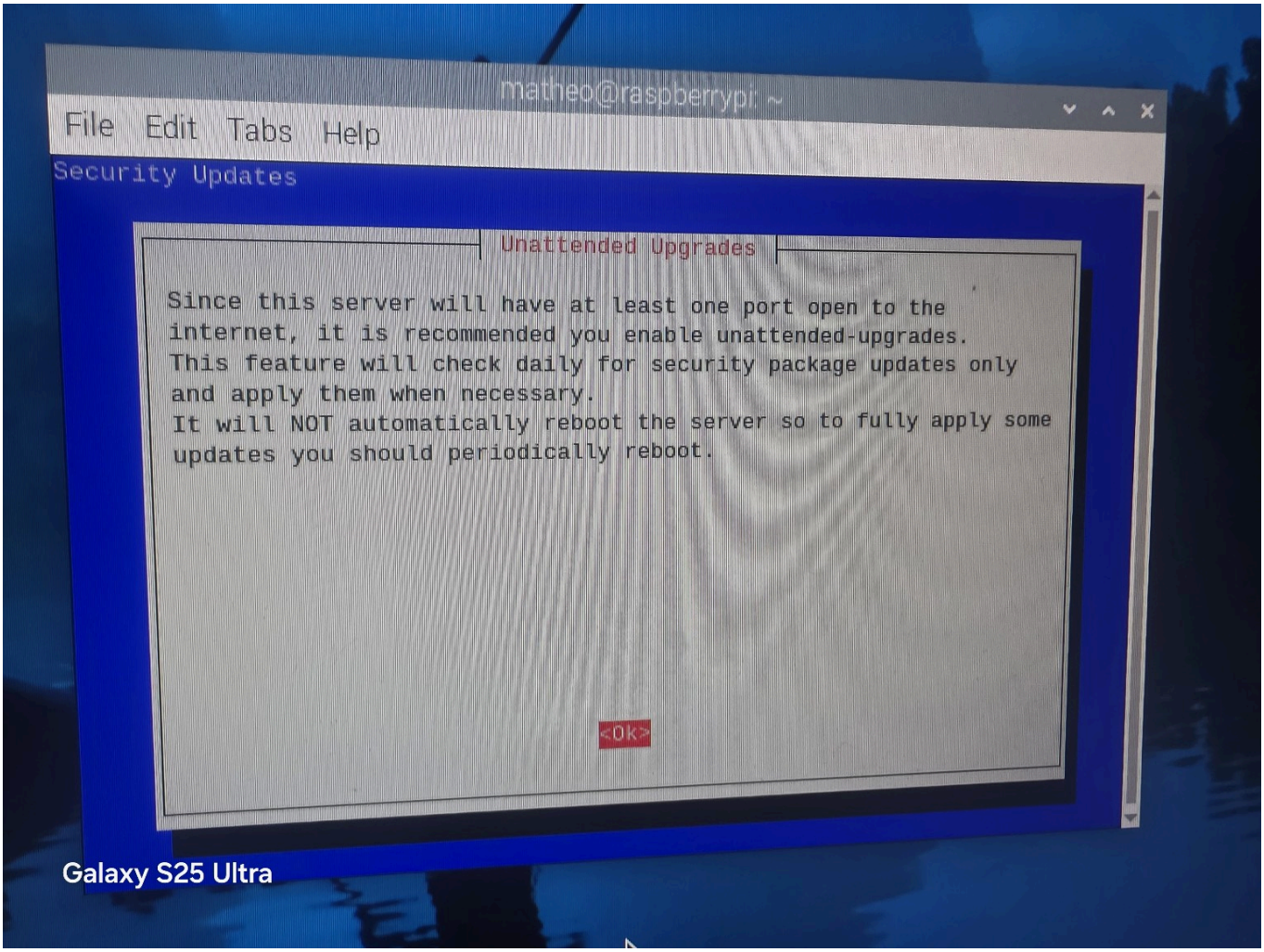


Specifying Public IP for Remote Access . Setting the home’s current public IP address as the VPN endpoint so clients can connect securely from anywhere.

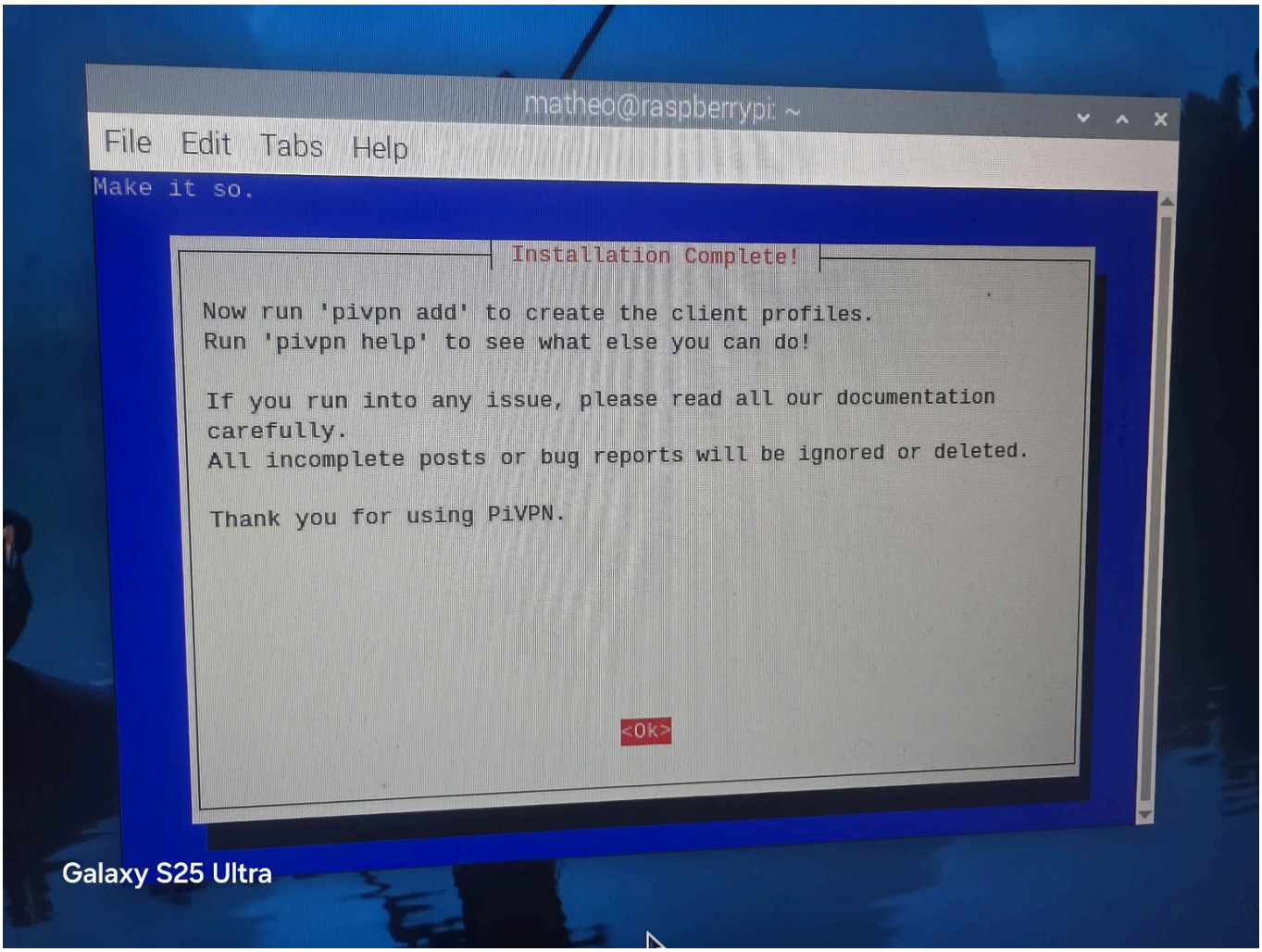


Enabling Unattended Security Updates . Hardening the VPN host by enabling daily security updates—vital when exposing services to the internet. It won't reboot your Pi automatically, so I

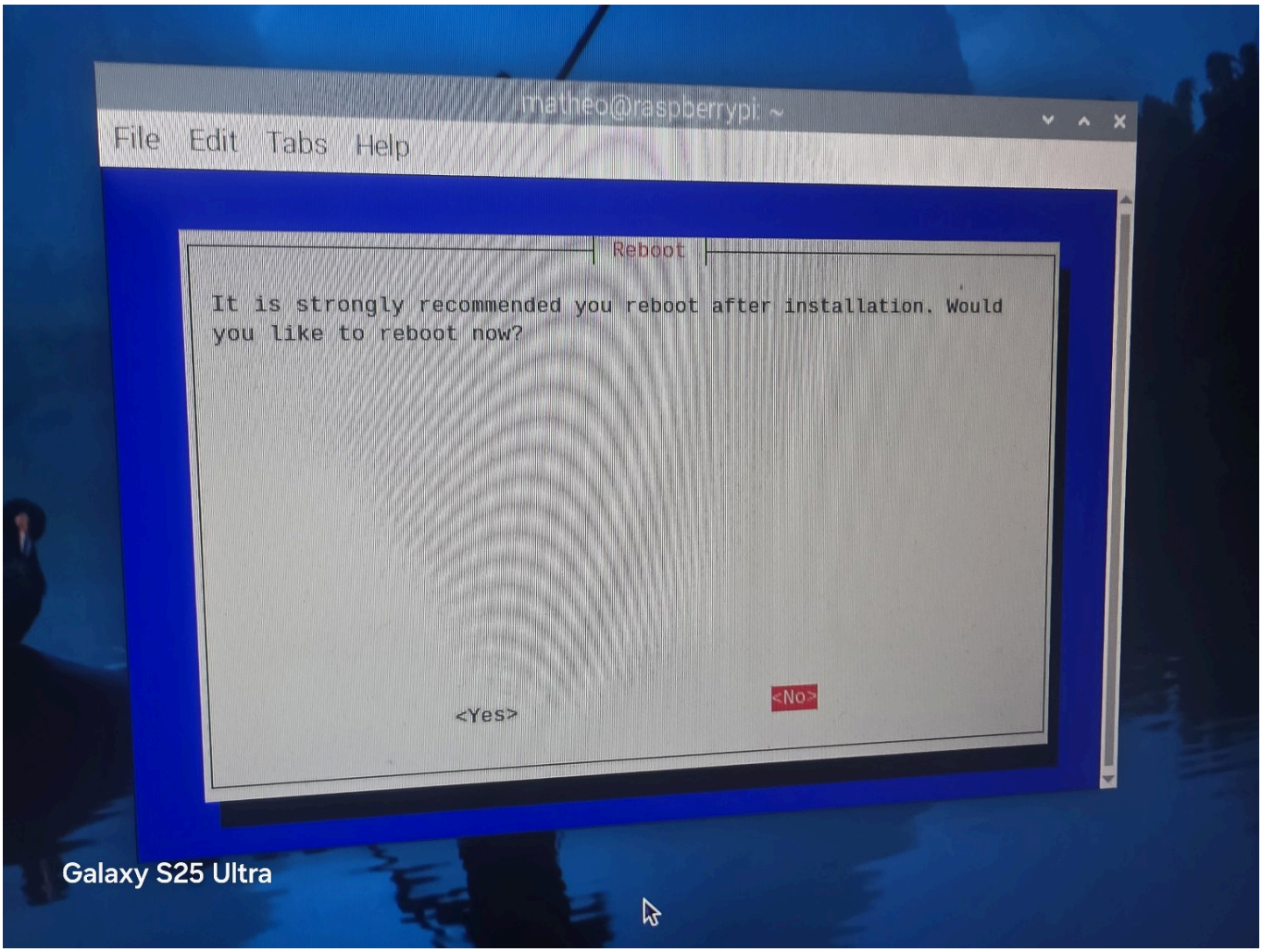
just have to restart it manually once in a while.



Installation Summary Screen . Installation complete! PiVPN and WireGuard are now configured. Next step: a quick reboot to finalize setup.

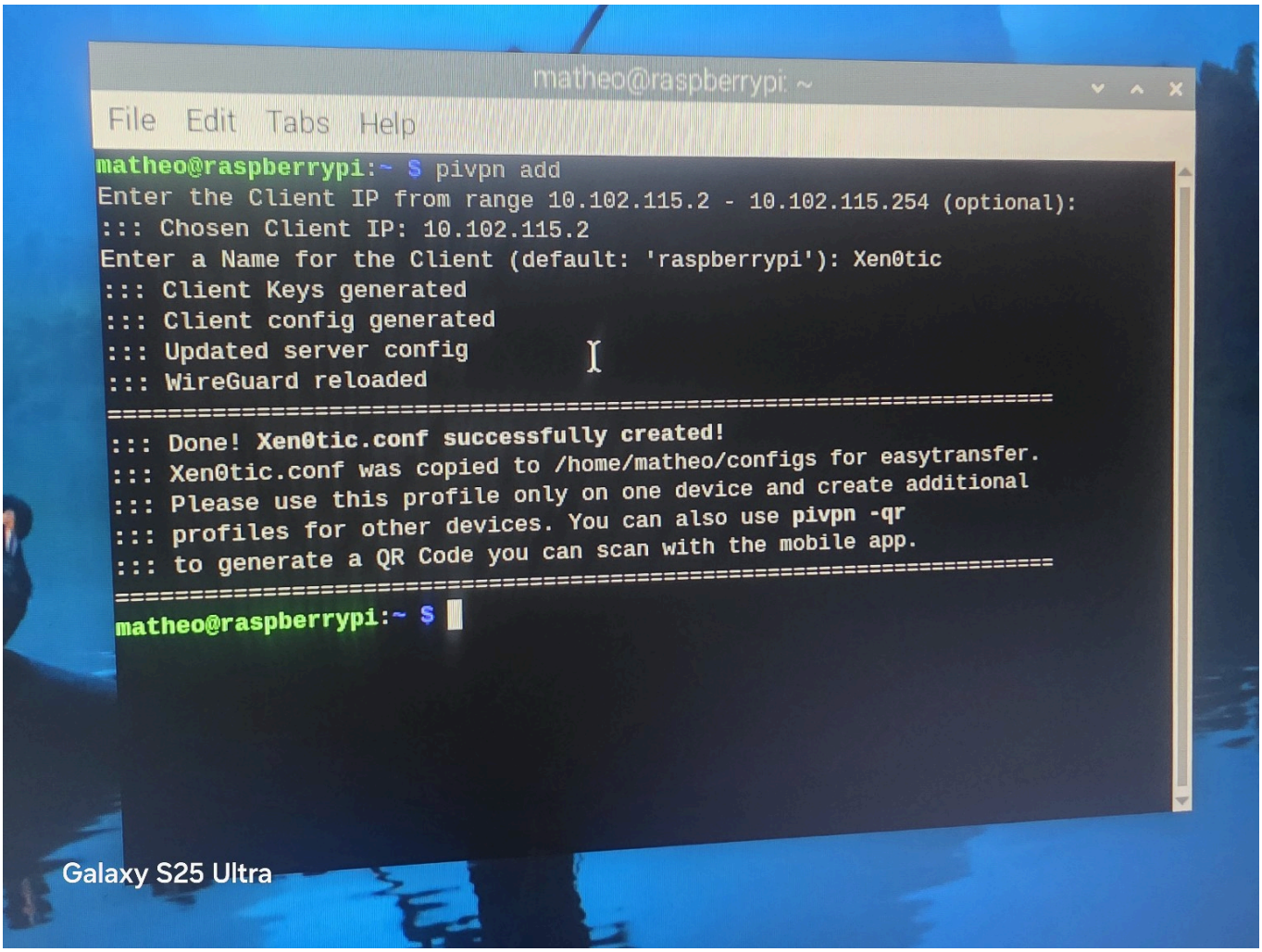


Reboot Prompt . Accepting system reboot—ensuring all PiVPN changes are properly applied before creating client profiles.



D. VPN Client Profile Setup

Creating VPN Client Profile via Terminal (`pivpn add` **)** . Creating a unique WireGuard client profile named 'Xen0tic'—generating private keys and configuration file for one device. I have to select a client IP but I don't really mind what it is so I just press Enter. Then I had to choose a name for the profile and I chose Xen0tic and clicked enter which successfully created the configuration file which I will have to share with everyone that needs to use this VPN.

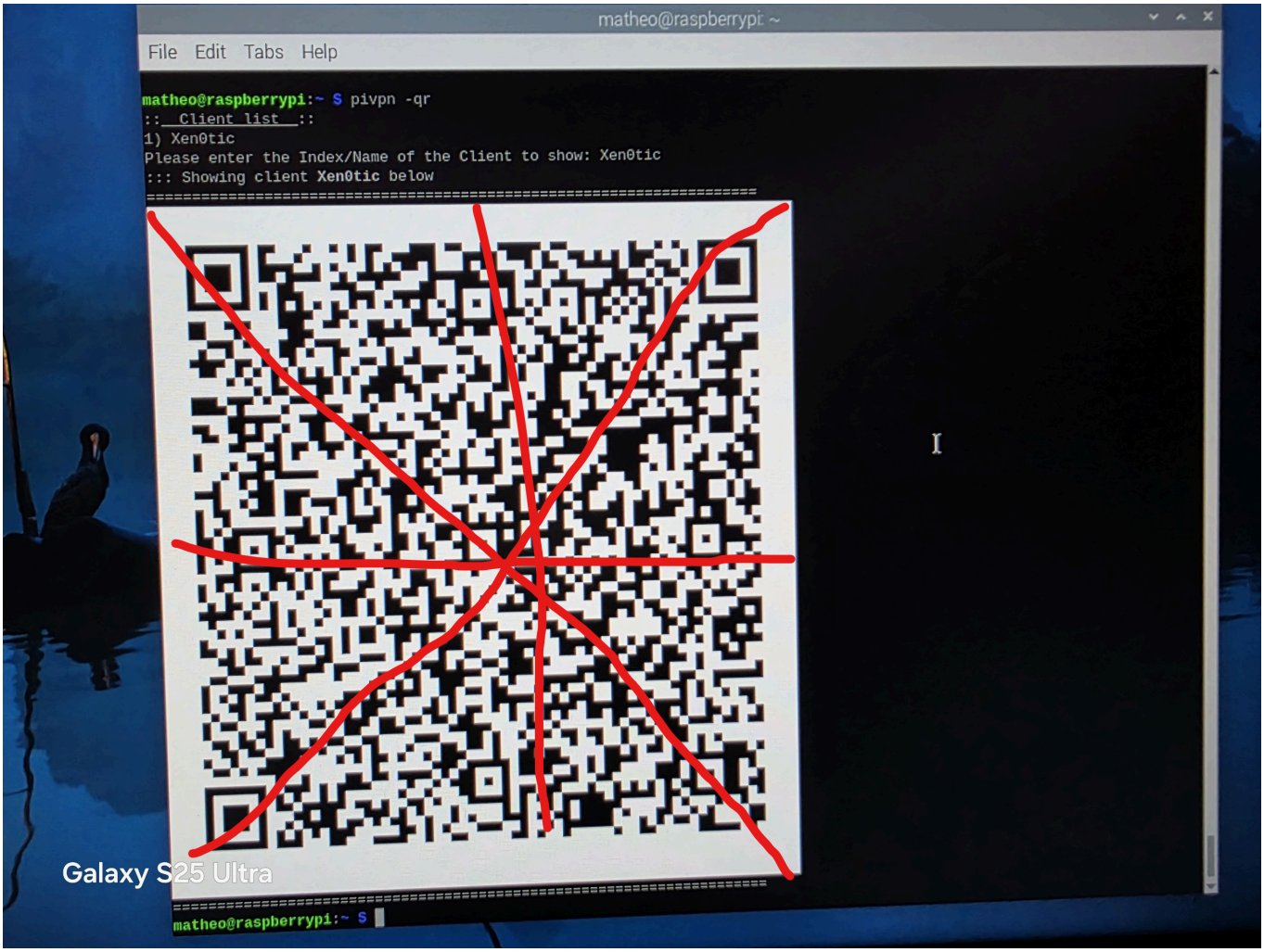


E. Wireguard Setup

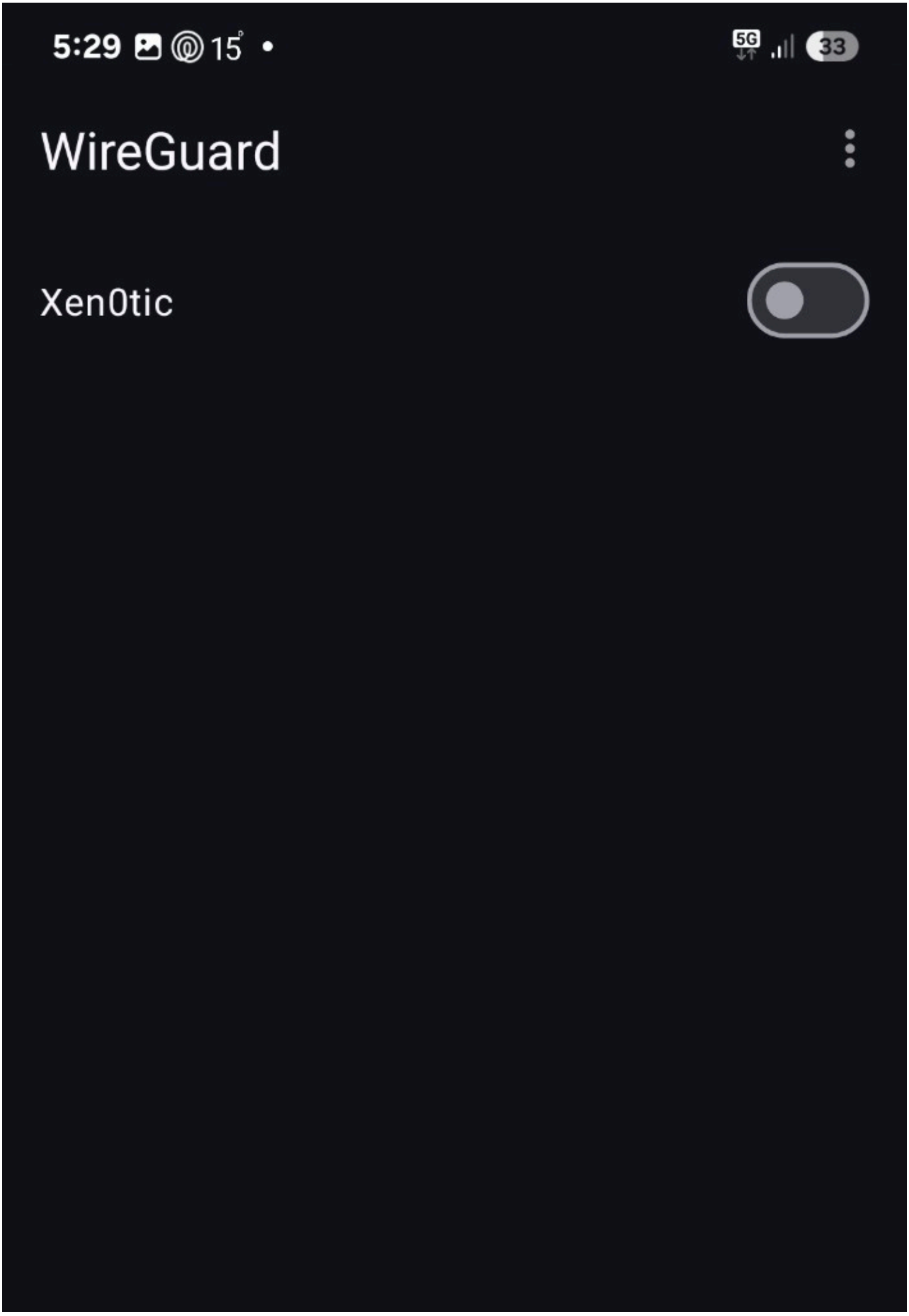
Next I downloaded WireGaurd on the phone to test it:

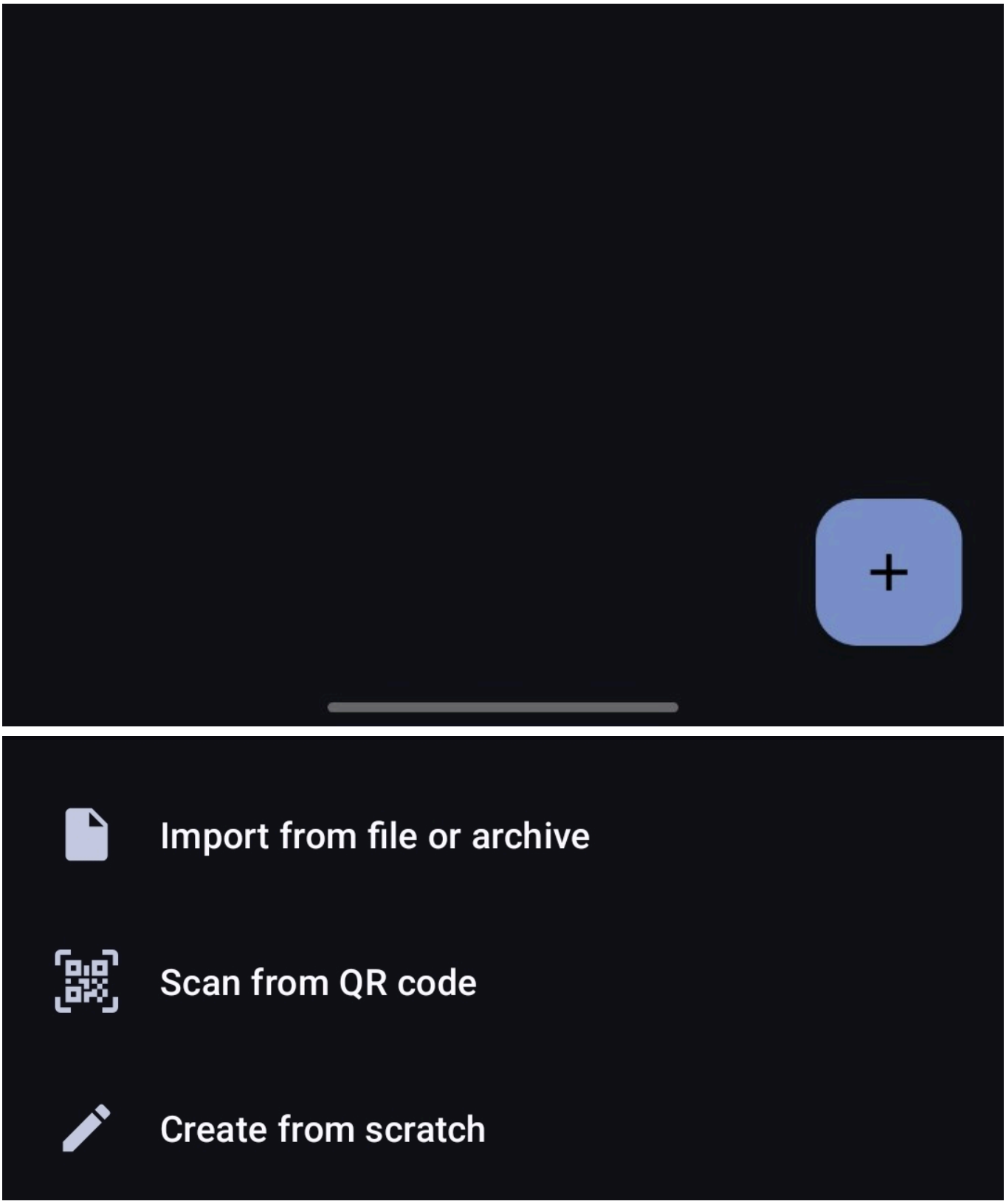
Generating and Scanning QR Code on Mobile . Running `pivpn -qr` to generate a scannable QR code—allowing mobile devices to instantly import VPN settings without manual

entry.

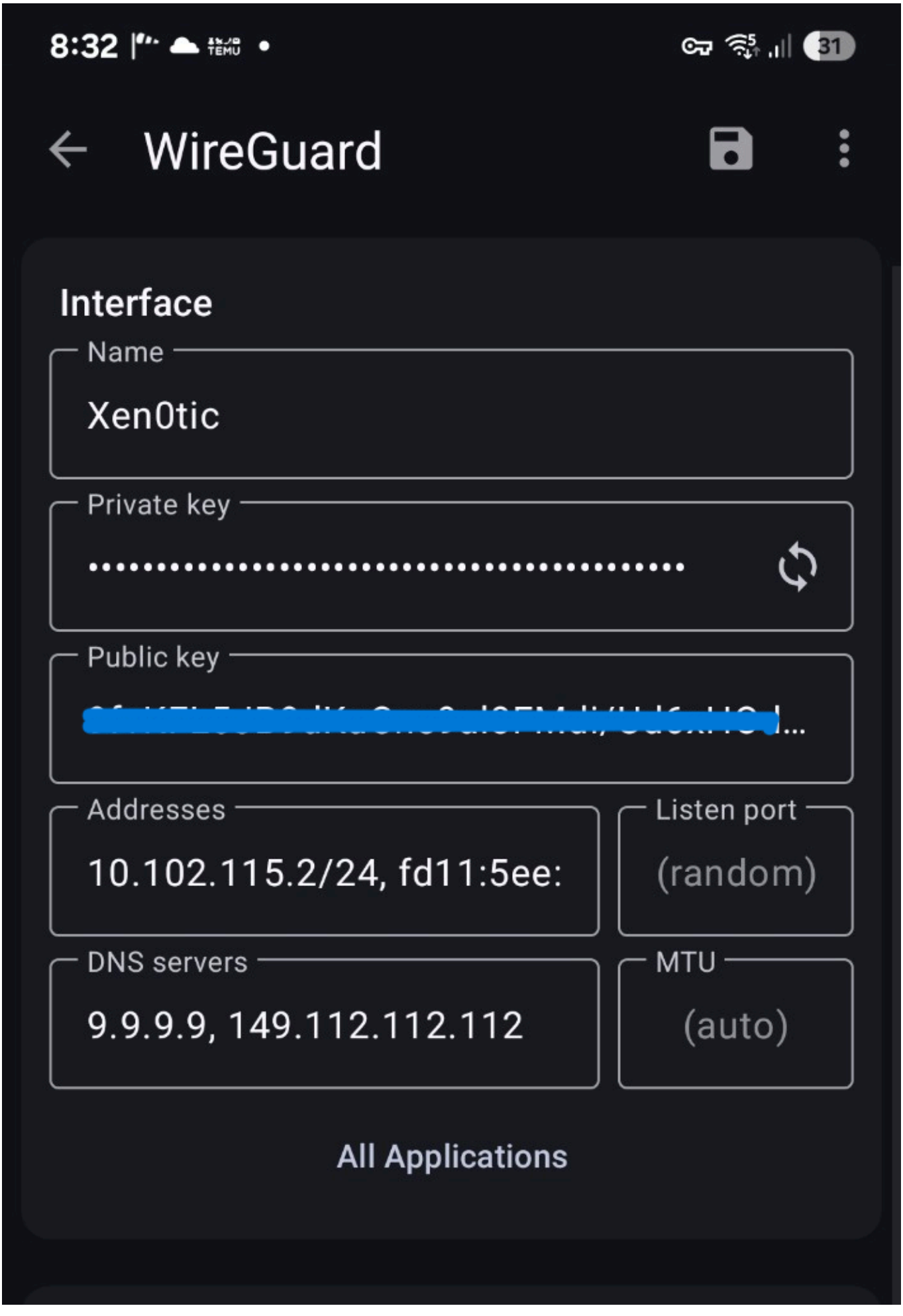

matheo@raspberrypi: ~
File Edit Tabs Help
matheo@raspberrypi:~ $ pivpn -qr
:: Client list ::
1) Xen0tic
Please enter the Index/Name of the Client to show: Xen0tic
::: Showing client Xen0tic below
matheo@raspberrypi:~ $
Galaxy S25 Ultra

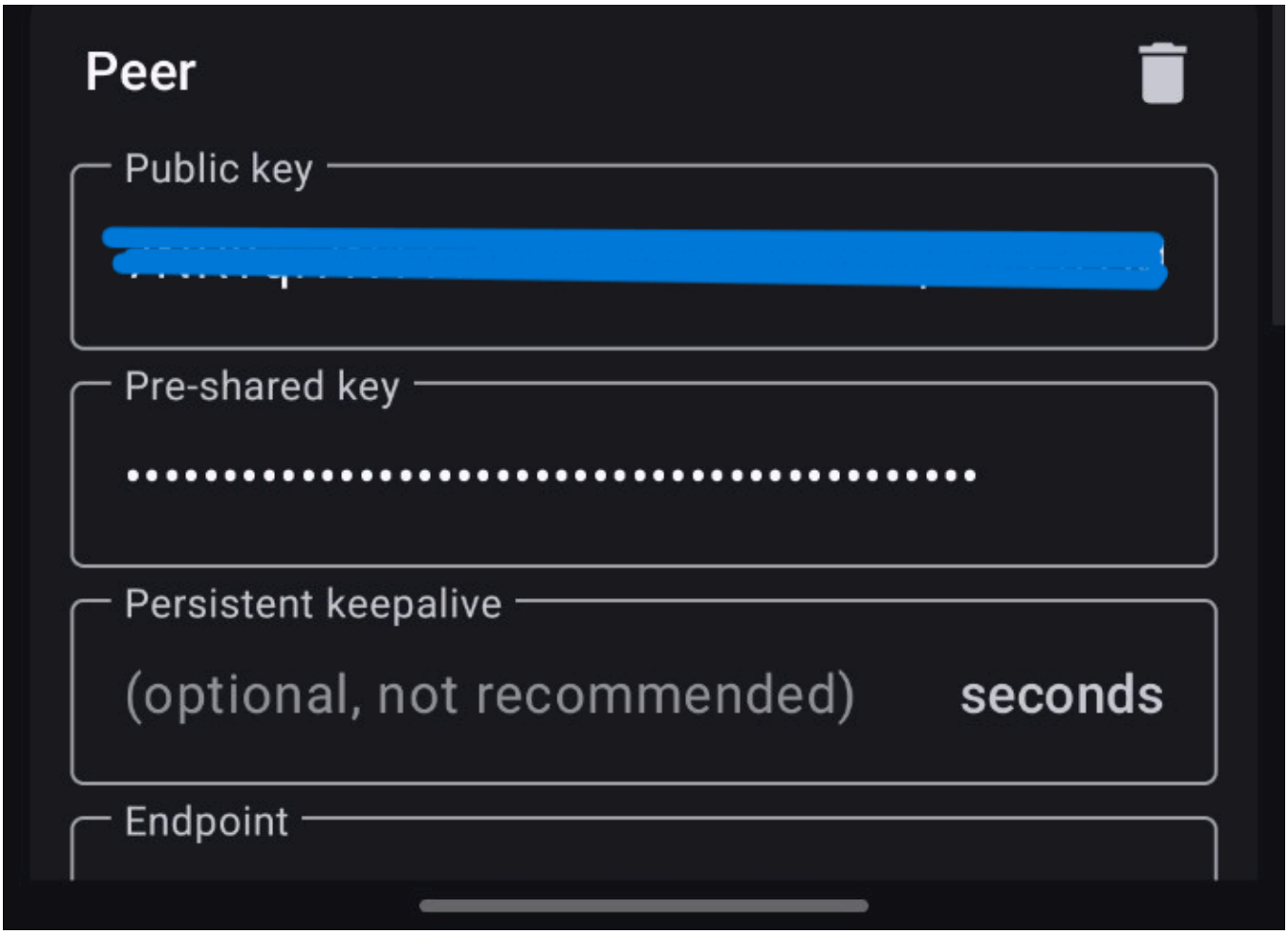
Next I pressed the Plus button on the bottom right and clicked scan QR code


5:29
15
5G
33
WireGuard
Xen0tic



WireGuard Mobile App Showing Imported Config . WireGuard app successfully displays the newly imported profile, ready to establish a secure tunnel back to the Raspberry Pi.

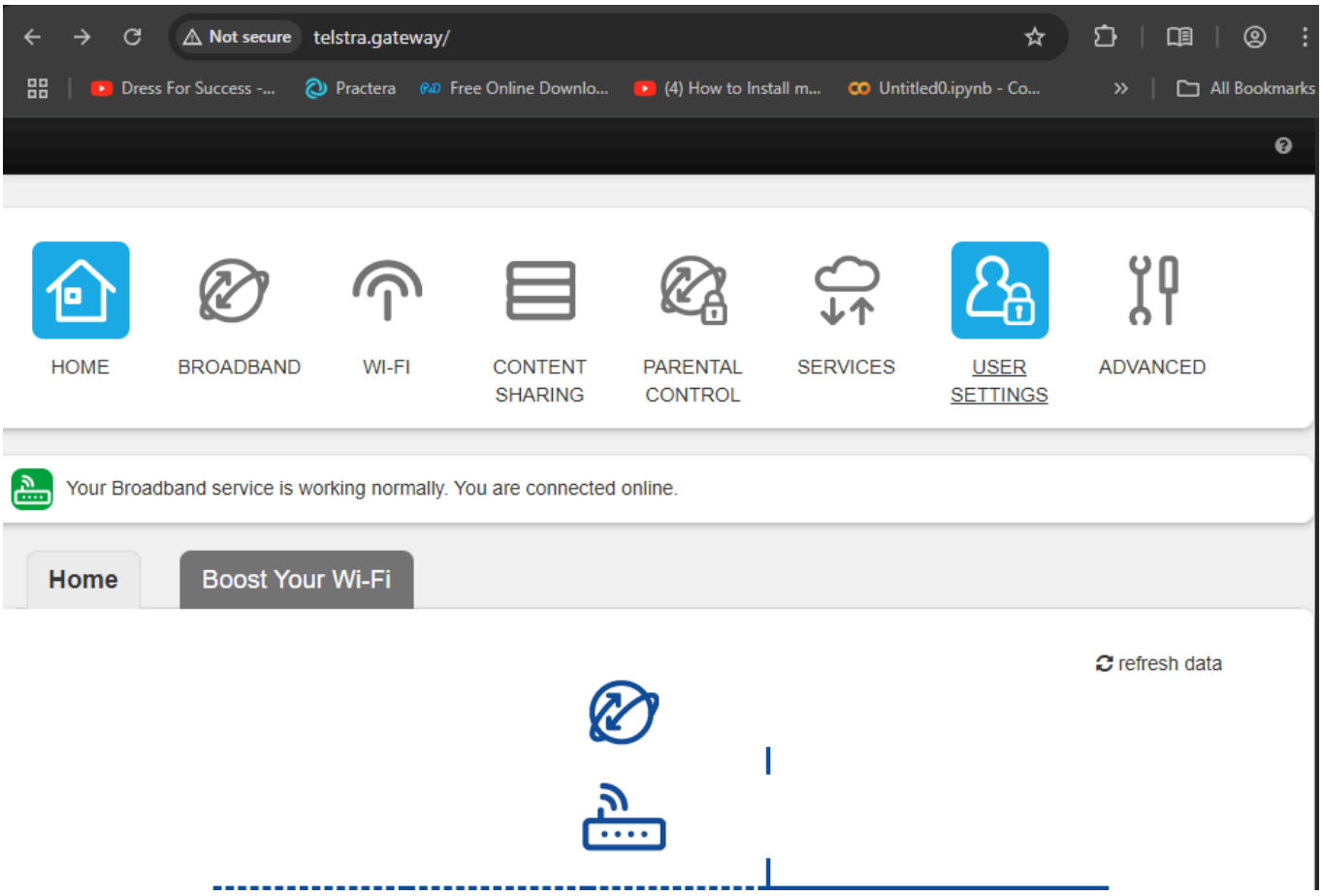

8:32
TEMU
31
WireGuard
Interface
Name
Xen0tic
Private key
Public key
Addresses
10.102.115.2/24, fd11:5ee:
Listen port
(random)
DNS servers
9.9.9.9, 149.112.112.112
MTU
(auto)
All Applications



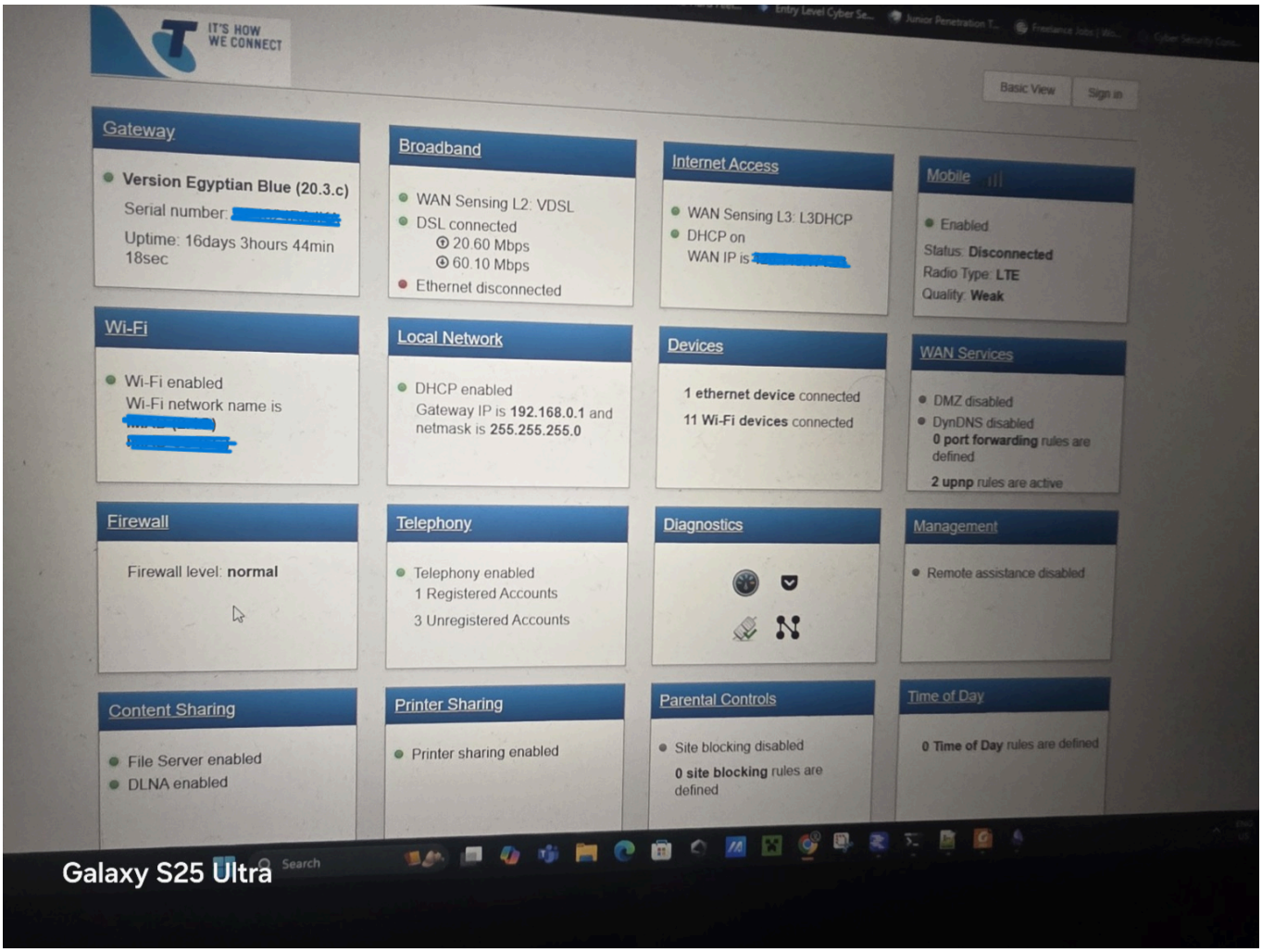
F. Router Configuration

Logging into Telstra Gateway Router Admin . Accessing the Telstra router’s admin interface to expose the Pi’s WireGuard service to the internet.

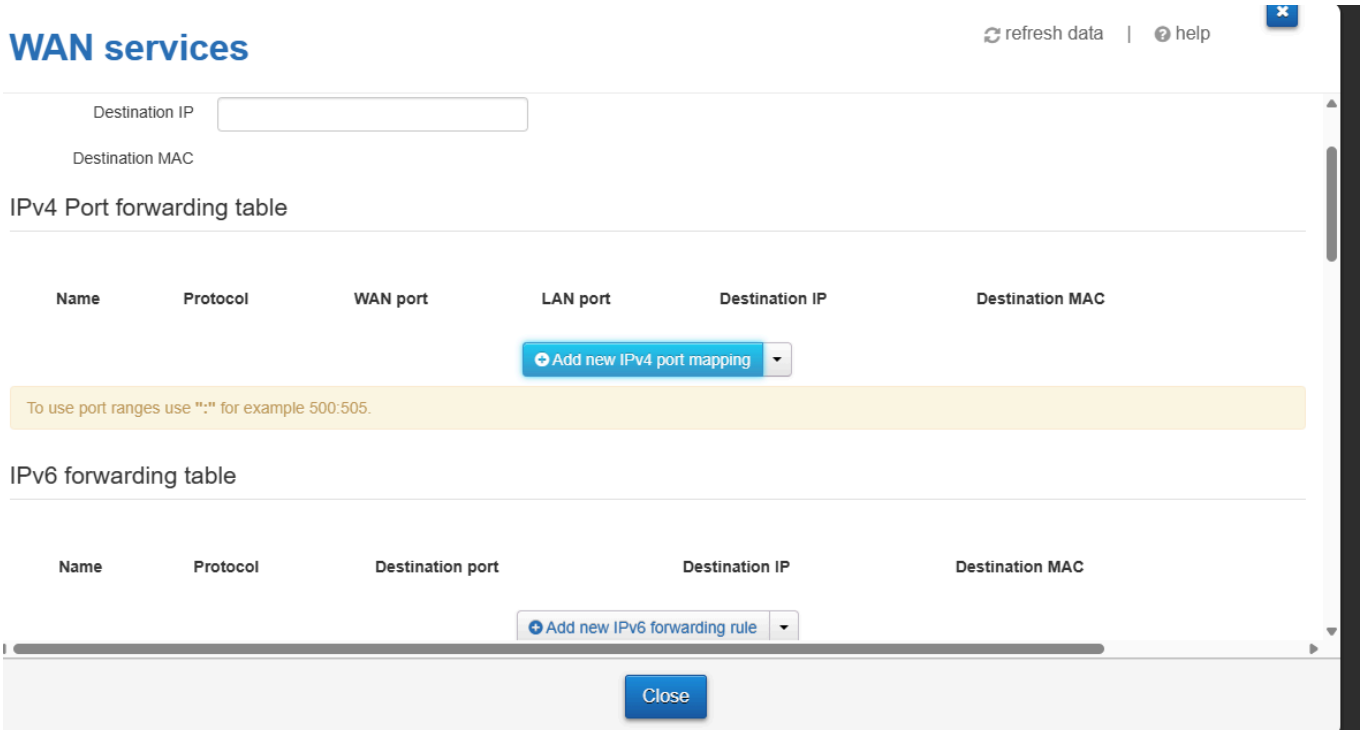
http://telstra.gateway/



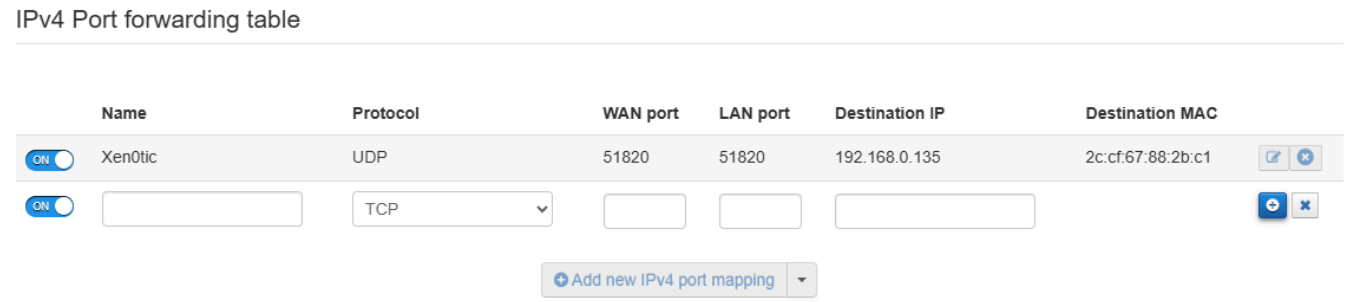
Navigating to Advanced Settings . Switching to the 'Advanced' tab to access WAN Services and Port Forwarding settings—where external traffic rules are defined.



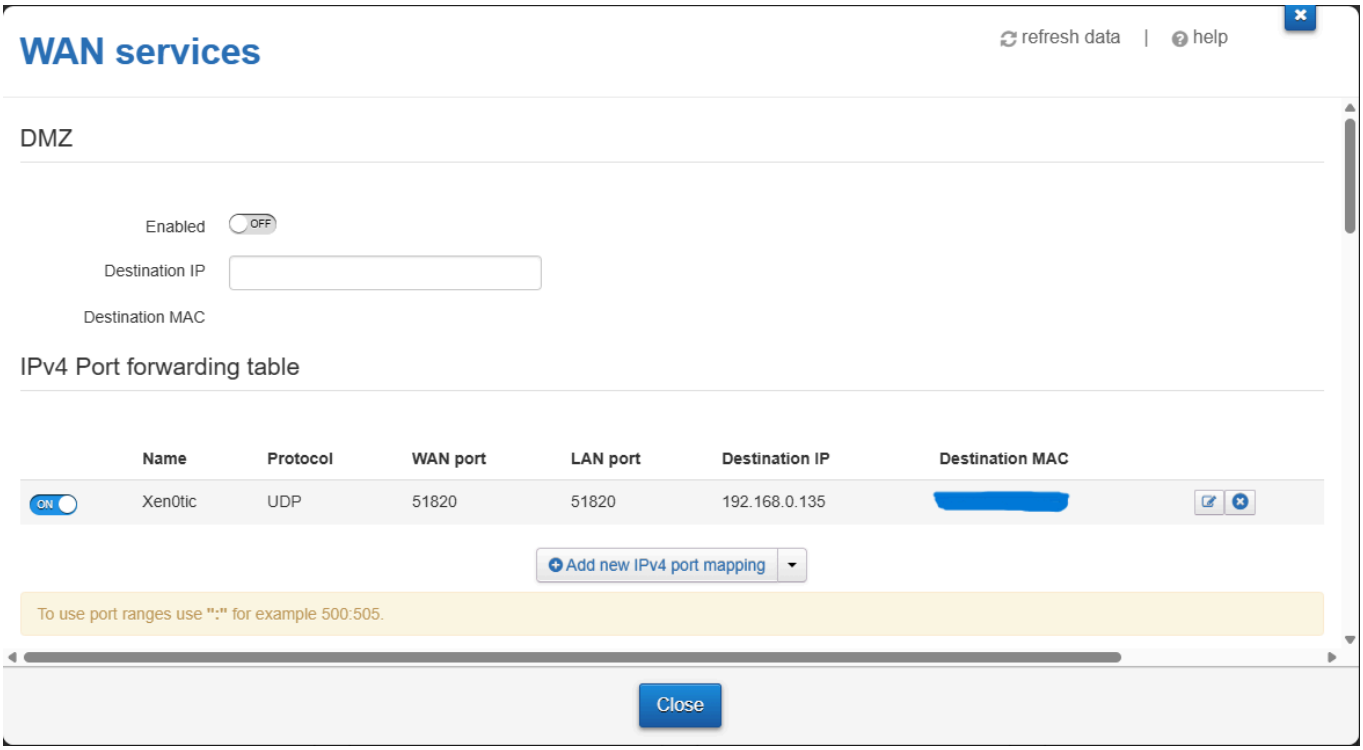
No Existing Port Forwards Yet . The router currently has no port forwarding rules set—indicating our VPN isn't externally accessible just yet.



Next I have to Click Add new IPv4 port mapping but in the below screenshot ive already added the vpn but in the future this is the section that need to be filled and to successfully add it you must click the little tiny blue plus button on the right side .



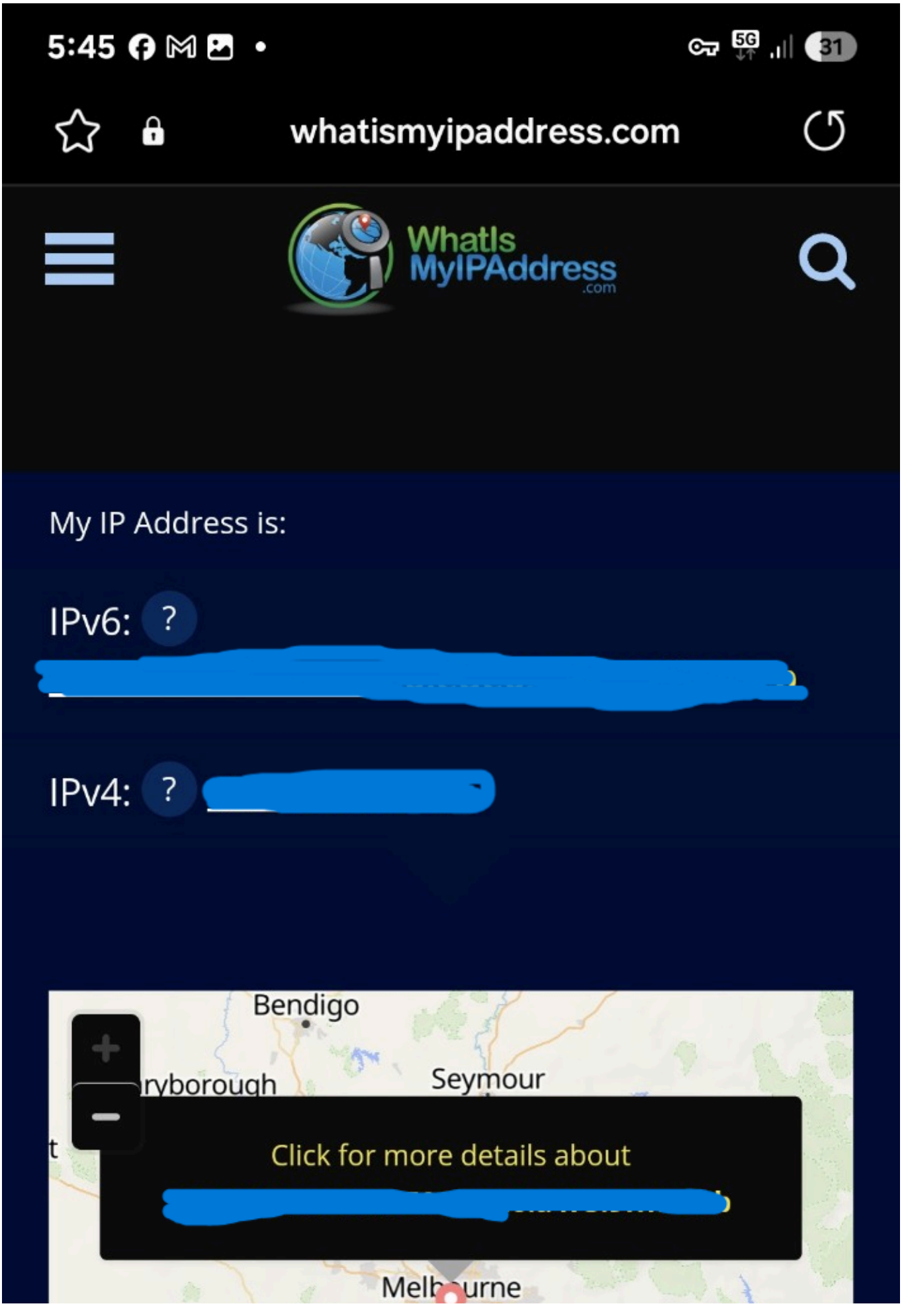
Configured VPN Port Forward Rule for UDP 51820 . Successfully mapping UDP port 51820 from WAN to the Raspberry Pi's internal IP—this lets VPN clients securely reach our home server.

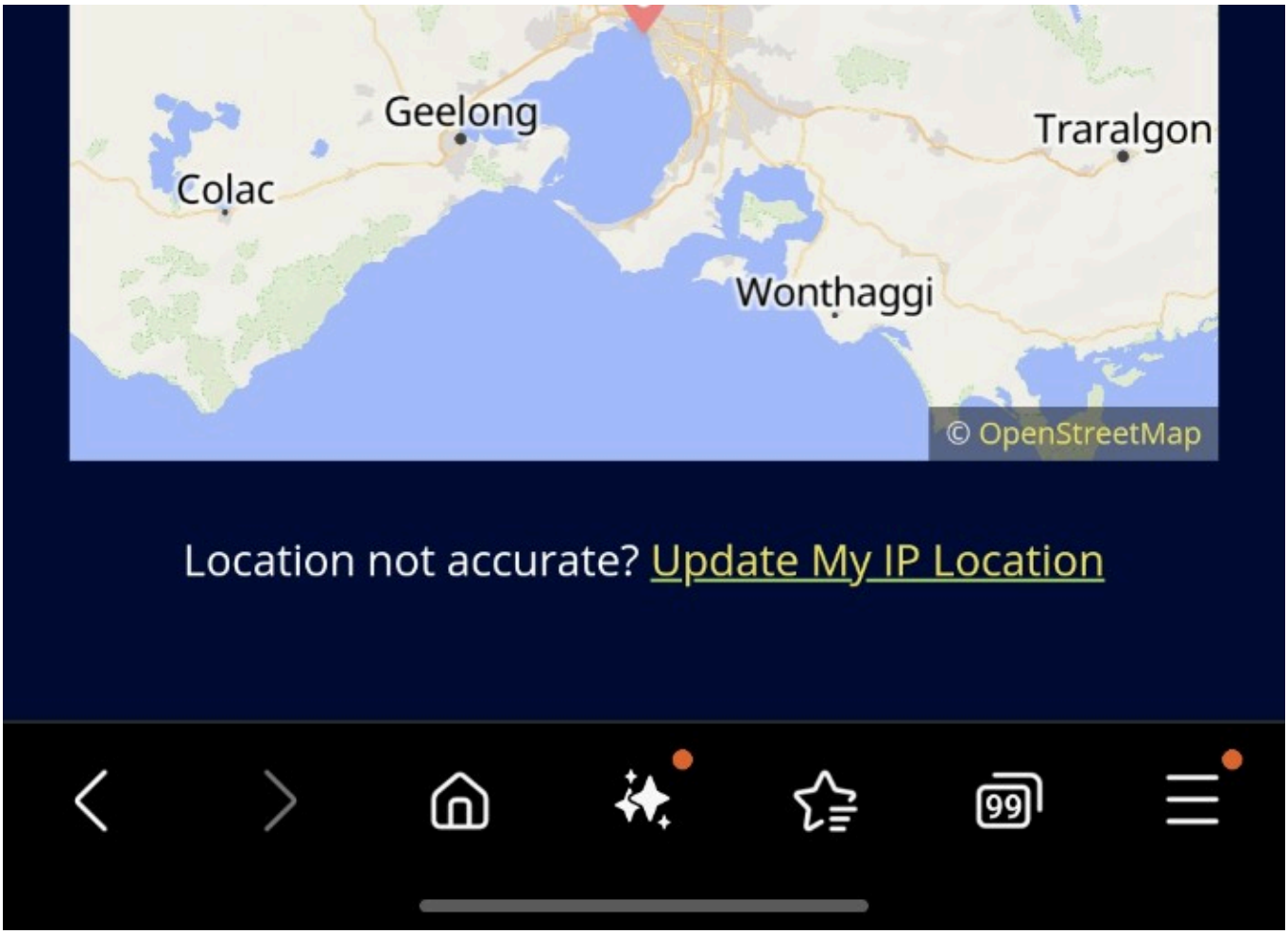


5. Testing & Validation

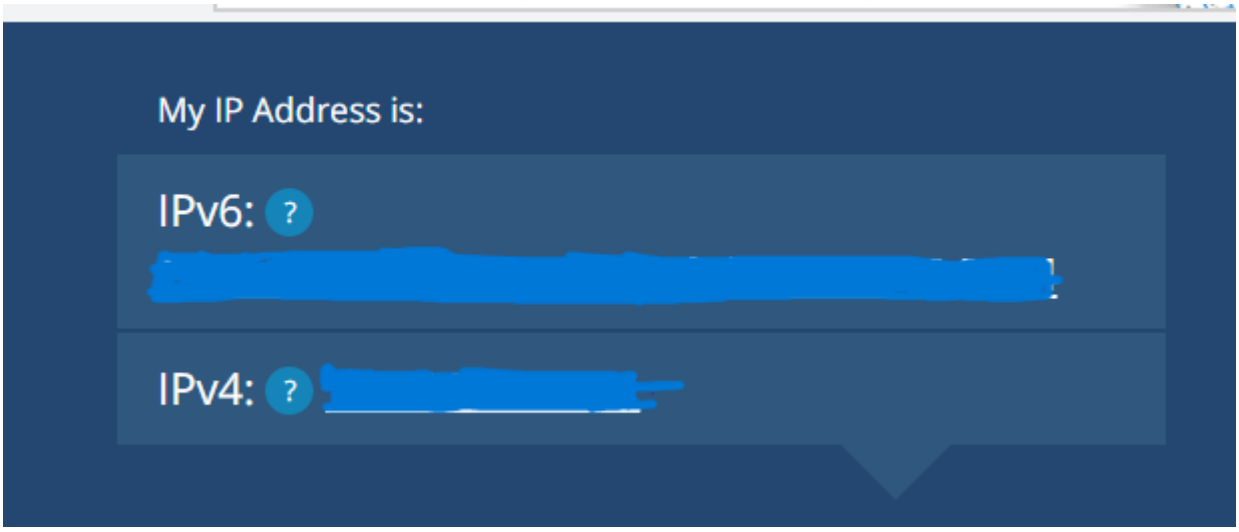
Checking IP with VPN On (Mobile Device) . Browsing to WhatIsMyIPAddress.com while VPN is active—shows external IP as the home's IP, confirming successful tunneling.

https://whatismyipaddress.com/


5:45
whatismyipaddress.com
WhatIs MyIPAddress .com
My IP Address is:
IPv6: ?
[redacted]
IPv4: ? [redacted]
Bendigo
Seymour
Click for more details about
[redacted]



Checking IP with VPN Off (Laptop on Home Network) . Running the same IP test from the home network to confirm both devices appear as coming from the same source—proof the VPN works!



6. Security Considerations

In this VPN configuration, **Quad9 (9.9.9.9)** was selected as the DNS resolver for its strong security posture. Quad9 blocks access to malicious domains based on threat intelligence feeds, making it a solid choice for privacy-conscious setups and baseline protection.

However, this setup did not include granular control over ad or tracker blocking. To improve on both performance and user customization, the next phase of this project involves integrating:

- **Pi-hole** as a local DNS filtering layer
- **Cloudflare (1.1.1.1)** as the upstream resolver

Cloudflare was selected for its fast resolution speeds, strong privacy policy, and support for DNS over HTTPS (DoH). When paired with Pi-hole, this combination allows for:

- Custom ad/tracker blocking
- Faster DNS resolution
- Improved local network visibility
- Optional filtering without relying entirely on third-party blocklists

This modular design allows switching upstream providers and blocklists as needed. It also keeps DNS queries internalized via the Pi-hole server while preserving speed and security through Cloudflare.

---

7. Known Issues / Future Improvements

- **No initial ad-blocking or DNS logging**: DNS queries are not filtered or logged so integrating pi-hole will solve this issue.
- **Fallback DNS exposure**: Initial DNS setup used public resolvers (Quad9) directly, which may expose metadata to third-party providers.
- **Basic firewall only**: IP whitelisting is limited; future improvements could include full `iptables` lockdown with port knock or more refined access control.
- **No user management/auditing**: VPN client usage is not currently logged or monitored, which limits accountability and auditing capability.

---

8. Conclusion

- **What I learned technically**
  I learned how to deploy and configure a secure VPN using WireGuard on a Raspberry Pi. This included setting up port forwarding on the router, generating server/client key pairs,

configuring the WireGuard interface, and assigning a privacy-focused DNS provider (Quad9) for all connected clients. I also practiced configuring system services and troubleshooting connectivity and DNS resolution issues.

- **What this shows about my security mindset**

This project highlights my focus on privacy and control over my network traffic. By routing all client DNS queries through a known secure provider (Quad9), I reduced reliance on my ISP's DNS and ensured encrypted resolution paths. It shows I'm security-conscious, capable of deploying secure infrastructure, and continuously think about data privacy, network segmentation, and potential future hardening strategies.

Benefits of the VPN I Just Set Up on Raspberry Pi

1. **Secure Remote Access to Home Network**
   - I can access my home network (and Raspberry Pi) from anywhere in the world.
   - All data is encrypted through a secure WireGuard tunnel.
2. **Mobile & Laptop Traffic Is Protected**
   - When connected, my phone or laptop routes all internet traffic through my home IP.
   - Useful for using public Wi-Fi safely without risk of spying.
3. **Real IP Masking**
   - My device’s IP becomes my home internet IP (even when using mobile data).
   - This helps bypass restrictions that may apply to my original network.
4. **Works Across Any External Network**
   - I don’t need to configure routers outside my home — it just works wherever I am.
   - Useful in places where VPNs or apps are blocked.
5. **Can Share With Family Securely**
   - I’ve already generated a `.conf` file that allows others to connect securely to my VPN.
   - No need for third-party VPN providers or accounts.
6. **Port Forwarding Configured**
   - My router is now set to forward VPN traffic to my Pi, so it accepts remote connections at all times.
7. **Low Cost, High Privacy**
   - No subscription fees.
   - Data is not shared with commercial VPN services.
   - I own and control the entire tunnel.