# Home server documentation

|  |
| --- |
|  |
| **Matej Papaj** |
| September 20, 2025 — Information about the steps taken — Student of dual education |

[Home server documentation 1](#_Toc913763163)

[Matej Papaj 1](#_Toc2027686280)

[September 20, 2025 1](#_Toc71637893)

[Information about the steps taken 1](#_Toc1481177088)

[Student of dual education 1](#_Toc1867377351)

## Introduction / Overview

This document provides details about the setup and configuration of my home server.

It includes hardware specifications, operating system installation, services, configurations, and lessons learned during the process.

## Hardware used

• Device: old acer notebook  
• CPU: inter i3-3110M  
• RAM: 6GB  
• Storage: 1TB HDD

## Operating system

• OS/Distro: Ubuntu server 24.04.03 LTS  
• Installation method: USB - Flash  
• Partitioning details: one partition - FULL

## Network setup

• Static IP configuration  
• Router/firewall rules ON

## System setup

### Initial set up

1. Downloading the ISO file from  
   [Get Ubuntu Server | Download | Ubuntu](https://ubuntu.com/download/server)
2. Setting ut the flash drive using rufus  
   [Rufus - Create bootable USB drives the easy way](https://rufus.ie/en/)
3. Changing boot order on the notebook and downloading the OS  
    only thing changed was downloading the ssh in the install stage

### Firewall setup

1. sudo ufw allow ssh  
    allows port 22 TCP
2. sudo ufw enable  
    enables the firewall

### Notebook lid setup

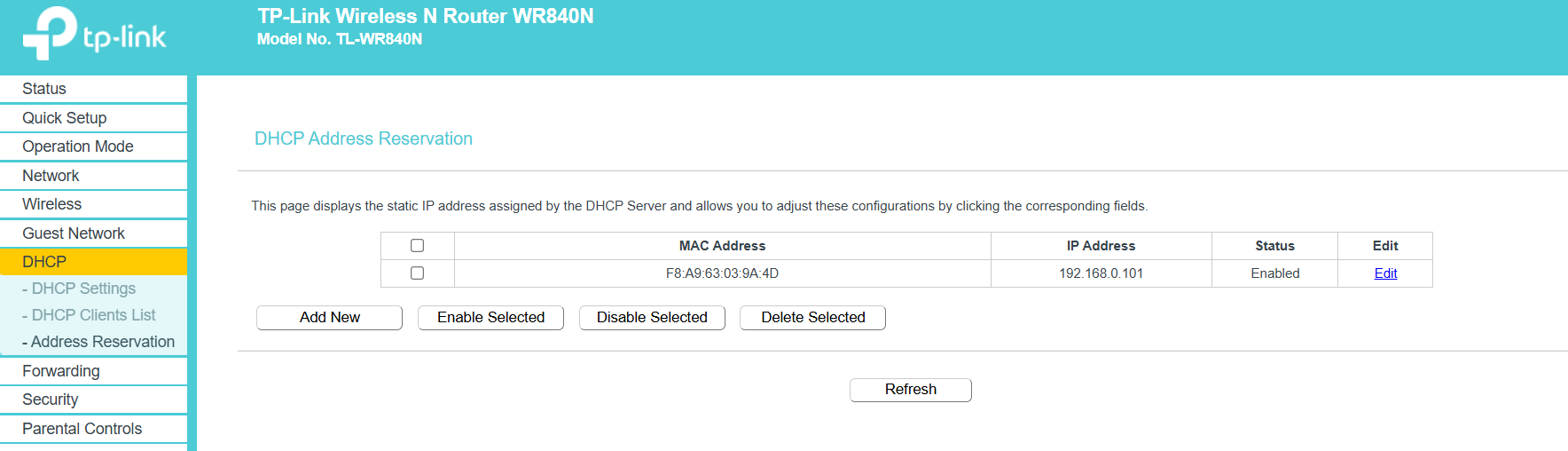
We want to keep this notebook closed since we will only be doing stuff from other pc for that however we need to disable the function that puts the notebook to sleep when we close its “lid”

1. Edit logind config  
    sudo vim /etc/systemd/logind.conf
2. Look for these lines:  
    #HandleLidSwitch=suspend  
    #IdleAction=suspend
3. Remove the # and set them to ignore:  
    HandleLidSwitch=ignore  
    IdleAction=ignore

## Router setup

Since we dont have a public IP adress for the server everything will be routed through the public IP of the router from the ISP. For that we had to give the server a static IP address on the internal network.

1. For that we go to router setting in my case it is 192.168.0.1
2. Then we go to DHCP and address reservation where we give a static IP to server based on its MAC address



1. For now this is everything we did in this initial steps

## Services / applications

### Ssh

As i have mentioned before the ssh was established upon the download of the OS, however now we need to enable the ssh connection so we dont need to specifically use the server but a pc on the network.

1. First we go to the client (the one connecting to the server)  
    for this Im using a w11 computer with git bash as my command line
2. First of all we will try to connect to the server if it works with  
    ssh username@ipOfTheServer
3. If everything works now we will set up the ssh key to this server
4. Once that is done and were back on the pc there should be a document .ssh
5. Now we create a ssh key  
    ssh-keygen -t ed25519 -C "text"
6. Then we go to server  
    ssh-copy-id -i ~/.ssh/id\_ed25519.pub username@server\_ip
7. And make a config file in .ssh to look like this  
    Host mrm  
    HostName 192.168.0.101  
    User mrm   
    Port 22  
    IdentityFile ~/.ssh/id\_ed25519
8. And we are done

### Wireguard

WireGuard is a **VPN protocol and software** designed to be simpler, faster, and more secure than older VPN solutions like OpenVPN or IPsec. Here we tried to use it as a tunnel to access the home server remotely.

I have tried to set it up but i have some sort of problem from the router side so i had to change the technology to a different one.

All that i found is that even if i have a public ip for router and try to port forward towards the server it wont workprobably some sort of blockage from the side of ISP.

### Tailscale

Tailscale is basically **WireGuard made easy**.

It uses WireGuard under the hood, but instead of you having to set up keys, configs, firewalls, and routing by hand, Tailscale handles all of that automatically.

**Purpose of Tailscale:**

* Creates a **zero-config VPN mesh network** between your devices.
* Lets your devices securely talk to each other, no matter where they are (home, school, office, cloud, mobile).
* Uses **WireGuard encryption** but adds a control plane for identity, key distribution, NAT traversal, and access control.
* Works with your existing logins (Google, Microsoft, GitHub, etc.) for authentication.

**What it’s used for:**

* Accessing your home PC from anywhere without port forwarding.
* Creating a secure private LAN across laptops, phones, servers, and VMs.
* Teams use it as a lightweight alternative to corporate VPNs.
* Developers use it to connect cloud servers (AWS, GCP, Azure) without exposing them to the public internet.

#### Installation steps

1. Create a tailscape account
2. Press add device
3. Choose the correct OS and the tailscape will provide a step-by-step guide

#### Configuration

Only configuration made was connecting the server to the tailscape account

### Prometheus and grafana

### Prometheus

Prometheus is an open-source monitoring and alerting system that’s widely used for collecting metrics and analyzing system performance.

**Purpose of Prometheus**

Prometheus is built to monitor systems, applications, and infrastructure by collecting, storing, and analyzing time-series metrics (values that c hange over time).

**What it’s used for**

* **System monitoring** – Track CPU, memory, disk, and network usage on servers.
* **Application monitoring** – Collect performance data (like request rates, response times, error counts) from apps.
* **Kubernetes / container monitoring** – Observe container health, pod resource usage, and scaling behavior.
* **Alerting** – Define conditions (e.g., “CPU > 90% for 5 minutes”) and trigger alerts to notify you before outages.
* **Troubleshooting & debugging** – Use historical metrics to figure out why something failed.
* **Capacity planning** – Spot trends in resource usage to predict when you’ll need to scale.
* **Service Level Objectives (SLOs) & SLAs** – Check if your system meets uptime/performance promises.

#### Instalation steps

1. Sudo apt install prometheus  
 2. sudo systemctl enable prometheus  
 3. sudo systemctl start prometheus  
 4. sudo ufw allow 9090/tcp  
 5. Sudo vtim /etc/prometheus/prometheus.yml  
 global:  
 scrape\_interval: 15s  
 scrape\_configs:  
 - job\_name: node  
 static\_configs:  
 - targets: ['localhost:9100']  
 6. sudo ufw reload  
 sudo ufw status - check firewall status

#### Node\_exporter

Its the part that monitors computer components and needs to be installed.

However when downloadidng prometheus the way we did it will be installed too. Under the name of **prometheus-node-exoporter**

### Grafana

* A **data visualization and monitoring tool**.
* It connects to different databases, time-series data stores (like Prometheus, InfluxDB), and even cloud services.
* Runs in a web browser, providing interactive dashboards.

**Purpose of grafana**

To collect, visualize, and analyze data in real time.

To give teams and businesses a single place to monitor system   
 performance, logs, metrics, and KPIs.

To alert users when something goes wrong (like high CPU usage, server   
 downtime, or unusual traffic).

**What it’s used for**

* **Infrastructure & Server Monitoring** – CPU, RAM, disk usage, network traffic.
* **Application Monitoring** – response times, error rates, user activity.
* **DevOps & Site Reliability** – integrates with Prometheus, Loki, Elasticsearch for logs and metrics.
* **Business Dashboards** – track sales, customer engagement, financial performance.
* **IoT & Smart Systems** – visualize sensor data in real-time.

#### Installation steps

1. Install necessary tools:  
 sudo apt-get install -y apt-transport-https software-properties-common wget  
 2. Import the GPG key:  
 sudo mkdir -p /etc/apt/keyrings/  
 wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee /etc/apt/keyrings/grafana.gpg > /dev/null  
 3. To add a repository for stable releases:  
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list  
 4. update the list of available packages  
 sudo apt-get update  
 5. install Grafana OSS  
 sudo apt-get install grafana  
 6. Enable grafana service on boot  
 sudo systemctl enable grafana-server  
 7. Start grafana  
 sudo systemctl start grafana-server

**grafana service can be found under the name of grafana-server**

8. Now go to IP:3000 u will be in GUI of grafana and login  
 default login is admin, admin  
 9. Change passwd  
 10. Go to connection -> data source -> add new data source and choose   
 prometheus  
 11. Choose prometheus url that is <http://localhost:9090> 12. Go to [Grafana dashboards | Grafana Labs](https://grafana.com/grafana/dashboards/?collector=nodeexporter) and choose dashboard that is working for node exporter, download the json.  
 13. Go back to grafana, home, create your first dashboard, import   
 dashboard, discard and import the json, load

Finito

Changing some settings

1. sudo vim /etc/default/prometheus  
 2. Add:  
 ARGS="--config.file=/etc/prometheus/prometheus.yml \  
 --storage.tsdb.path=/var/lib/prometheus \  
 --storage.tsdb.retention.time=30d"  
 3. sudo systemctl daemon- reload  
 4. sudo systemctl restart prometheus

### Setting up automatic updates

https://documentation.ubuntu.com/server/how-to/software/automatic-updates/

1. sudo vim /etc/apt/apt.conf.d/50unattended-upgrades
2. Add this: (or whatever u want)  
    remove // from "${distro\_id}:${distro\_codename}-updates";  
    uncheck and add mail to: Unattended-Upgrade::Mail "mail”;  
    Unattended-Upgrade::MailReport "always";  
    Unattended-Upgrade::Remove-Unused-Dependencies "true";  
    Unattended-Upgrade::Automatic-Reboot "true";  
    Unattended-Upgrade::Automatic-Reboot-Time "02:00";
3. sudo unattended-upgrade --dry-run –debug
4. sudo systemctl restart unattended-upgrades
5. sudo systemctl restart apt-daily.timer apt-daily-upgrade.timer

Centralized website for services

Dns server