Webserver mit Raspberry Pi 5

ZIEL: Aufbau eines eigenen Webservers auf einem Raspberry Pi zur Bereitstellung einer HTTPS-geschützten Zeitzonen-API für eine Netlify-Website.

Projektübersicht

Ziel dieses Projekts war es, einen Raspberry Pi 5 mit SSD und Pironman 5 Gehäuse als eigenen Webserver zu betreiben. Der Server sollte eine eigene REST-API mit Live-Zeiten für verschiedene Weltstädte zur Verfügung stellen. Die Daten werden über Flask bereitgestellt und über eine Netlify-Website(kostenlos) konsumiert.

Verwendete Hardware & Software

HARDWARE:

- Raspberry Pi 5 (8 GB RAM)
- 500 GB SSD (extern angeschlossen)
- Pironman 5 Gehäuse (mit aktiver Kühlung und RGB :p)

SOFTWARE:

- Raspberry Pi OS (64-bit)
- Flask (Python Web-Framework)
- Nginx (als Reverse Proxy)
- Certbot (Let's Encrypt für HTTPS-Zertifikate)
- DuckDNS (für dynamische DNS-Adresse)

Zielsetzung

- Eigenen Webserver betreiben
- API-Zeitdaten in Echtzeit für jegliche Städte mit deren Zeitzonen
- Anbindung der API an eine statische Netlify-Website (Website selfmade by Tim)
- HTTPS-Verschlüsselung
- Erreichbarkeit von überall

Umsetzungsschritte & Bash-Kommandos

1. ***System vorbereiten***:

```
sudo apt update && sudo apt upgrade -y sudo apt install python3
python3-pip python3-venv nginx -y
```

2. ***Flask und Flask-CORS installieren:***

```
pip3 install flask flask-cors --break-system-packages
```

- ***Flask neustarten nach Änderung (z.B. wenn man eine neue Zeitzone hinzufügt.)***

```
sudo systemctl restart zeitserver.service
```

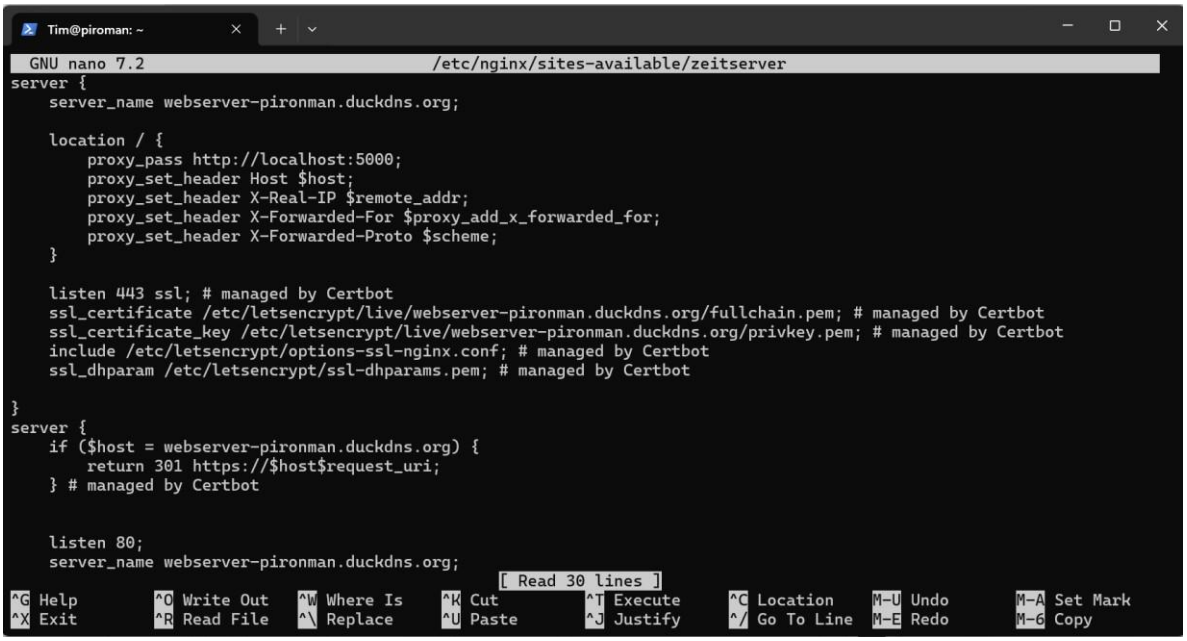
3. ***Projektordner anlegen und API erstellen:***

```
mkdir ~/zeitserver
cd ~/zeitserver
nano rtc.py
```

4. ***Flask-Server starten:***

```
nohup python3 ~/zeitserver/rtc.py &
```

5. *Nginx einrichten (Reverse Proxy):*



6. *DuckDNS einrichten:*

https://www.duckdns.org/update?domains=webserverpironman&token=persönlicher-Token&ip=

7. *Ports weiterleiten (Internet-Box):*

Port 80 -> 80 (HTTP)
Port 443 -> 443 (HTTPS)

8. *Zertifikat einrichten (Let's Encrypt):*

sudo apt install certbot python3-certbot-nginx -y sudo certbot --nginx -d webserver-pironman.duckdns.org

9. *Automatisierung vom Programm beim Anschluss an den Strom:*

```
sudo nano /etc/systemd/system/zeitserver.service
```

Danach:

```
[Unit]
Description=Flask Zeitserver
After=network.target

[Service]
User=Tim
WorkingDirectory=/home/Tim/zeitserver
ExecStart=/usr/bin/python3 /home/Tim/zeitserver/rtc.py
Restart=always

[Install]
WantedBy=multi-user.target
```

Somit werden folgende Befehle automatisch bei jedem Boot ausgeführt:

sudo systemctl daemon-reexec (Neustart der systemd-Hauptprozesse)
sudo systemctl daemon-reload (Ladet die «.service»-Dateien neu)
sudo systemctl enable zeitserver.service (Einmalige Aktivierung vom «.service»-Pfad)
sudo systemctl start zeitserver.service (Jetzt sofort starten)

Ergebnis

- Raspberry Pi stellt eine sichere Zeitzonen-API über HTTPS zur Verfügung
- Netlify-Website zeigt Uhrzeit dynamisch für jede Stadt
- Alle Anfragen sind abgesichert und frei von CORS-Problemen

Zukünftige Erweiterungen

- Autostart des Flask-Servers über systemd
- API-Zugriff mit Token absichern
- Uhrzeiten als animierte Digitaluhren darstellen
- Uptime-Monitoring

Fazit

Mit günstiger Hardware, freier Software und sauberem Setup wurde ein voll funktionsfähiger HTTPS-Webserver aufgebaut. Die Zeitdaten werden in Echtzeit bereitgestellt und professionell in eine moderne Webanwendung eingebunden.