Erstellung einer Task API und Inbetriebnahme auf einem Linux basiertem System

Erstellungsdatum: 22.06.2023

Autor:

Jarno Rohmert und Konstantin Prinz

Inhaltsverzeichnis

1 Einleitung **1**

2 Server Daten **2**

3 Server Einrichtung **3**

3.1 Zugriff auf den Raspberry Pi 3 Modell B+ 3

3.2 Netzwerkkonfiguration 4

3.3 Benutzer 4

3.4 Firewall 5

4 Apache **6**

4.1 Anforderungen 6

4.2 Aufsetzen des Apache Webservers 6

5 Task API **7**

6 Quellen **8**

Glossar

Apache Webserver Der Apache Webserver ist eine beliebte Software, die auf einem Computer läuft und Websites und andere webbasierte Anwendungen hostet, um sie über das Internet verfügbar zu machen.. 1, 6, 7

API Eine API (Application Programming Interfac) ist eine Schnittstelle womit man Informationen austauscht.. 7

Firewall Eine Firewall überwacht den Datenverkehr auf dem Computer und kann konfiguriert werden, um den eigenen Wünschen zu entsprechen. Eine Firewall schützt den Computer vor möglichen Angriffen.. 5

Raspberry Pi 3 Modell B+ Ein Einplatinencomputer, welcher oft in der Industrie genutzt wird aber auch für Hobby Tüftler geeignet ist.. 1–4, 6

Raspbian Raspbian ist ein Betriebssystem, welches auf Linux Debian aufbaut und speziell für den Raspberry Pi entwickelt wurde.. 2, 3

SSH SSH (Secure Shell) ist ein Netzwerkprotokoll, das eine sichere, verschlüsselte Verbindung zwischen einem Client und einem Server ermöglicht, um eine sichere Remote-Verwaltung und den sicheren Datenaustausch zu gewährleisten.. 3–5

WSGI WSGI (Web Server Gateway Interface) ist eine Spezifikation in der Python-Webentwicklung, die die Kommunikation zwischen Webservern und Webanwendungen erleichtert, indem sie eine einheitliche Schnittstelle definiert.. 6

1 Einleitung

Im Laufe des Unterrichts haben die Schüler eine eigene API (Application Programming Interface) mit der Programmiersprache Python entwickelt. Damit diese dann auch auf einem eigenen Server installiert und gestartet wird, erhielten die Schüler den Arbeitsauftrag einen Raspberry Pi 3 Modell B+ einzurichten und zu starten.

Die Ziele waren [Wic]:

- Netzwerkkonfiguration
- Informationsseite über das System erstellen
- 2 neue Benutzer erstellen und konfigurieren
- Firewall Regeln deklarieren
- Task API auf das System installieren
- Bonus : Task API als Container laufen lassen

Dazu sollen die Einstellungen nach einem Neustart noch funktionieren [Wic]

2 Server Daten

Wir haben auf einem Raspberry Pi 3 Modell B+ auf dem Raspbian als Betriebssystem läuft gearbeitet. Um Raspbian auf den Raspberry Pi 3 Modell B+ installiert zu kriegen, benötigt man eine micro SD-Karte. Diese Karte wurde zuvor mithilfe eines micro SD-Karten Leser mit dem Betriebssystem geflashed. Danach wird sie in den im Raspberry Pi 3 Modell B+ vorgesehenen micro SD-Karten Slot rein geschoben.

Eigene kleine Beschreibung der Kapazitäten des Raspberry Pi 3 Modell B+s
1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT) [Fou]

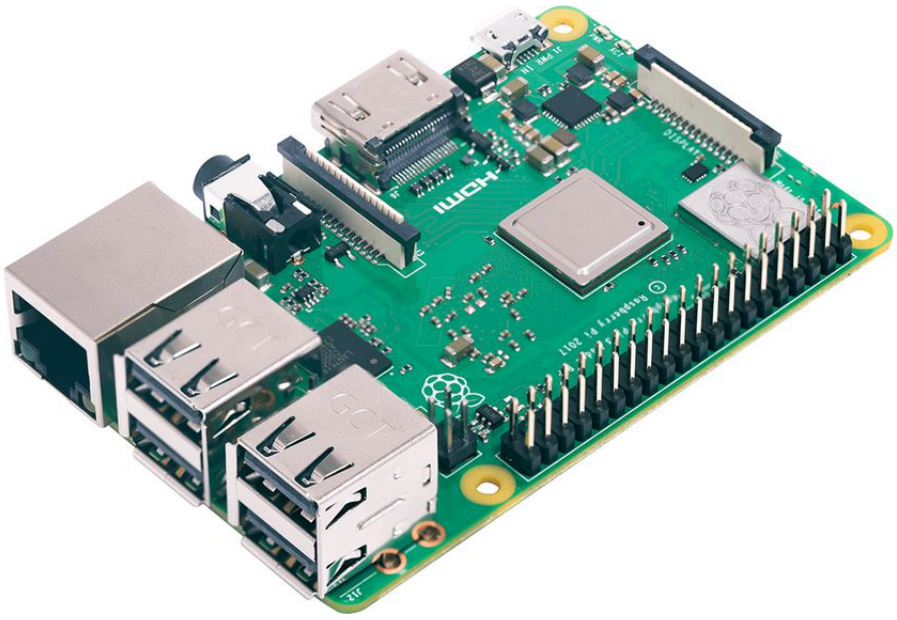

Abbildung 1: Raspberry Pi 3 Modell B+

3 Server Einrichtung

Allgemein wurde zuerst Raspbian geupdatet und alle anderen Programme auch

```
sudo apt-get update && upgrade
```

Der **Sudo** Befehl am Anfang signalisiert, dass das Kommando mit Administratoren Rechten ausgeführt werden soll (insofern der Benutzer dafür die Rechte hat)

Das Betriebssystem besteht aus Ordner Strukturen in denen man sich mit dem **cd** Kommando bewegen kann. Mit **cd ..** geht man einen Ordner zurück während man mit **cd home** man in den home Ordner geht. Der Schritt in den Home Ordner funktioniert nur wenn, der Home Ordner in dem selben Ordner befindet in dem man sich selbst befindet. Um herauszufinden welche andere Dateien und Ordner im aktuellen Ordner sind nutzt man **ls**. Der Befehl listet dann alles auf.

3.1 Zugriff auf den Raspberry Pi 3 Modell B+

Der Raspberry Pi 3 Modell B+ kann über verschiedene Wege angesprochen werden:

- SSH Verbindung
- HDMI Verbindung wenn nicht Headless (ohne Desktop)
- USB Verbindung

In unserem Fall haben wir SSH genutzt, da wird keine Grafische Oberfläche benötigen.

```
ssh pi@192.168.24.112
```

SSH am Anfang gibt an das wir per SSH auf das Gerät zurückgreifen wollen. Das **pi** gibt an mit welchem Benutzer man sich verbinden möchte. Gefolgt von dem @ und der IP des Raspberry Pi 3 Modell B+s

3.2 Netzwerkkonfiguration

Zunächst wurde beim ersten Zugriff auf dem Raspberry Pi 3 Modell B+ eine statische IPv4 Adresse vergeben. Dazu muss man die **/etc/dhcpcd.conf** Datei bearbeiten. Damit man diese bearbeiten kann wird der Befehl **nano** genutzt, mit dem man einen Editor mit der gewünschten Datei aufruft. Der bearbeitete Teil der Datei sieht wie folgt aus:

```
sudo nano /etc/dhcpcd.conf


interface eth0
static ip_address=192.168.24.112/24
static routers=192.168.24.254
static domain_name_servers=192.168.24.254
```

Dabei muss man die Zeile wo DHCP deklariert wird kommentieren, damit diese nicht mehr greift. Die Befehle sind auf unser System zugeschnitten und würden bei dem eigenen Gebrauch variieren.

Bei einem Schreibfehler kann man sich möglicherweise nicht mehr mit dem Raspberry Pi 3 Modell B+ verbinden, da er sich selbst nicht mehr verbinden kann

3.3 Benutzer

Die Benutzer Erstellung ist recht simpel gestaltet. In unserem Fall mussten wir den Nutzer **benutzer72** und **fernzugriff** erstellen

```
sudo useradd benutzer72 pw: benutzer72
Sudo useradd fernzugriff pw: fernzugriff
```

[ang]

Dadurch bekommen beide Benutzer auch ein Recht das Sie sich per SSH verbinden dürfen, was aber nicht der Aufgabenstellung entspricht. Um dies zu ändern muss man die **`ssh_config bearbeiten`**, welche in `/etc/ssh/ssh_config` liegt. In der Datei wird dann die Zeile **DenyUsers benutzer72** hinzugefügt, damit der Benutzer Benutzer 72 sich nicht mehr per SSH verbinden kann. Damit die Änderungen übernommen werden wird der Befehl **sudo systemctl restart sshd**[Git] genutzt, damit wird der SSH service neu gestartet.

Während der Konfiguration hatte der Benutzer fernzugriff kein eigenes Verzeichnis bekommen, was einige Fehler aufwarf. Um diese zu beheben nutzten wir den Befehl **`mkhomedir_helper fernzugriff`**[Sha] der bei einem vorhanden Benutzer ein eigen Verzeichnis erstellt.

3.4 Firewall

Es sollten Firewall Einstellungen vorgenommen werden, um den Server sicherer zu haben. Die gewünschten Einstellungen sind:

- Zugriff auf Webserver und die Web-Applikation aus allen Netzwerken erlauben
- Zugriff auf SSH-Dienst nur aus dem eigenen Netzwerk erlauben
- Alle anderen ankommenden Pakete verwerfen

[Wic]

Wir nutzen **Iptables**, um die Einstellungen umzusetzen.

Installation und Update der Iptables

```
sudo apt-get install iptables
```

[Tuc]

Um sich die bisherigen Regeln anzusehen die es gibt nutzt man **sudo iptables -L**[KB0], wo dann die Regeln gezeigt werden die es bisher gibt.

Die gewünschten Einstellungen dann:

```
Regeln fuer Flask:
sudo iptables -A INPUT -p tcp --dport 5000 -j ACCEPT

Regeln fuer ssh:
sudo iptables -A INPUT -p tcp --dport 22 -s 192.168.0.0/16 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 22 -s 127.0.0.0/8 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 22 -j DROP
```

Mit denen werden dann Anfragen am Port 5000 akzeptiert, und bei SSH werden Anfragen auf dem 22 Port akzeptiert wenn sie aus dem selben Netzwerk sind, sonst werden die geblockt.

Die Einstellungen werden mit **sudo -s iptables-save -c** [KB0] gespeichert und automatisch neu geladen.

Probleme gab es bei uns aber beim speichern, denn die Regeln die wir erstellt haben wurden nicht gespeichert auch wenn wir den Befehl dafür genutzt haben.

4 Apache

4.1 Anforderungen

Der Apache Webserver dient dem bereitstellen der Informationsseite und auch des hostens der Flask[Roh] Applikation. Durch das Hosten der Flask Applikation über den Apache Webserver wird mehr Stabilität gewährleistet, da der Apache Webserver WSGI unterstützt und ein Produktionsserver ist.

4.2 Aufsetzen des Apache Webservers

Zunächst wird Apache Webserver installiert mit **sudo apt -get install apache2**[Gla]. Theoretisch läuft der dann auch direkt schon unter der Ip vom Raspberry Pi 3 Modell B+, jedoch mit einer Platzhalter Html Seite.

Diese bearbeitet man wenn man in das Verzeichnis **var/www/html/** wechselt und die **Index.html** bearbeitet.

Damit dann der Flask Server[Roh] über den Apache Webserver läuft muss man **sudo apt-get install libapache2-mod-wsgi** installieren für die WSGI Erweiterung von Apache Webserver.

Wir sind den Schritten von der Offiziellen Flask Seite[Fla] gefolgt, kamen aber leider zu keinem Ergebnis. Daher wird der Flask Server[Roh] dann per Service gestartet und gestoppt, während der Apache Webserver für die Informationsseite zuständig ist.

5 Task API

Es wurde eine API mit Python und der Flask Bibliothek geschrieben, welche Todolisten erstellen und managen soll. Für mehr Info dafür siehe GitHub Link [Roh].

Aufgrund der Konfiguration vom Apache Webserver wird auf eine Service orientierte Herangehensweise umgeändert. Damit das neue Ziel umgesetzt werden kann nutzen wir **systemd** [Kha].

```
sudo apt-get install -y systemd
```

Dann wird eine eigene Service Datei verfasst in der diese Daten drin stehen:

```
[Unit]
Description=My test service
After=multi-user.target

[Service]
Type=simple
Restart=always
ExecStart=/usr/bin/python3 /home/pi/TaskiAPI/TaskAPI.py

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

[Kha]

Danach werden die Services neu gestartet und der neu erstellte aktiviert (es wird die Service Datei nach enable angegeben).

```
sudo apt-get install -y systemd
sudo systemctl enable taskapi.service
```

Start und Stopp Befehle für den Service und weitere für Status und Restart:

```
sudo systemctl start testapi.service
sudo systemctl status taskapi.service
sudo systemctl restart taskapi.service
sudo systemctl stop taskapi.service
```

6 Quellen

Literatur

[ang] Nicht angegeben. *How to Create Users in Linux (useradd Command)*. URL: https://linuxize.com/post/how-to-create-users-in-linux-using-the-useradd-command/ (besucht am 17. 05. 2023).

[Fla] Flask. *mod_wsgi (Apache)*. URL: https://flask.palletsprojects.com/en/2.0.x/deploying/mod_wsgi/ (besucht am 31. 05. 2023).

[Fou] Raspberry Foundation. *Raspberry Pi 3 Model B+*. URL: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/ (besucht am 23. 06. 2023).

[Git] Vivek Gite. *Delete SSH Keys Command for Linux and Unix*. URL: https://www.cyberciti.biz/faq/linux-unix-bsd-deleting-an-ssh-key/ (besucht am 24. 05. 2023).

[Gla] Erin Glass. *So installieren Sie den Apache-Webserver unter Ubuntu 20.04*. URL: https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04-de (besucht am 31. 05. 2023).

[KB0] KB0043436. *Configuring the firewall on Linux with iptables*. URL: https://help.ovhcloud.com/csm/en-dedicated-servers-firewall-iptables?id=kb_article_view&sysparm_article=KB0043436 (besucht am 07. 06. 2023).

[Kha] WasiUllah Khan. *Setup a python script as a service through systemctl/systemd*. URL: https://medium.com/codex/setup-a-python-script-as-a-service-through-systemctl-systemd-f0cc55a42267 (besucht am 21. 06. 2023).

[Roh] Jarno Rohmert. *TaskiAPI*. URL: https://github.com/jarn0r/TaskiAPI (besucht am 21. 06. 2023).

[Sha] Sagar Sharma. *Create Home Directory for Existing Users in Linux*. URL: https://linuxhandbook.com/create-home-directory-existing-user/ (besucht am 24. 05. 2023).

[Tuc] Dejan Tucakov. *Iptables Tutorial: Ultimate Guide to Linux Firewall*. URL: https://phoenixnap.com/kb/iptables-tutorial-linux-firewall (besucht am 07. 06. 2023).

[Wic] Christian Wichmann. *Deployment einer Web-Applikation*. URL: https://moodle.nibis.de/bbs_osb/pluginfile.php/202739/mod_resource/content/0/Einrichten%20eines%20Linux-Servers%20v04.pdf (besucht am 22. 06. 2023).