Dokumentation Deployment einer Rest API

Autor: Gerrit Koppe

Ausbildungsberuf: Fachinformatiker für Anwendungsentwicklung

Klasse: IFA12

Lernfeld 9: Netzwerke und Dienste bereitstellen

23. Juni 2023

Anmerkungen

Aus Gründen der Leserlichkeit wird in dieser Dokumentation das Wort „Server“ verwendet, wann immer vom Raspberry Pi die Rede ist.
Manche der hier in der Dokumentation beschriebenen Befehle mussten bei der Ausführung bestätigt werden. Die Bestätigung wird nicht explizit erwähnt.
Wann immer in einer Terminal Abbildung „#...“ steht, zeigt dies an, dass die Datei, welche jeweils bearbeitet wurde, weitere Zeilen hat, welche aber nicht relevant für den jeweils aktuellen Arbeitsschritt sind und dementsprechend ausgelassen wurden. Es kann außerdem anzeigen, dass die Kommandozeile nach Ausführung eines Befehls noch weitere Informationen anzeigt, diese allerdings ebenfalls ausgelassen wurden.
Aus technischen Gründen, wurde ab dem Kapitel „Einrichtung Webserver“ nicht mehr auf einem Raspberry Pi, sondern auf einem Ubuntu Server 22.04 gearbeitet. Es ist möglich, dass manche Kommandos sich zu denen, die auf einem Raspberry Pi ausgeführt werden würden, unterscheiden.

Inhaltsverzeichnis

1 Einleitung

In dieser Dokumentation wird die Konfiguration eines Raspberry Pi[1] als Host einer Rest API, sowie das Deployment besagter Rest API beschrieben.
Es wird zunächst auf allgemeine Vorbereitungen eingegangen. Anschließend wird beschrieben, wie die User und das Netzwerk, sowie die Firewall des Servers konfiguriert werden müssen. Abschließend wird die Installation des Webservers, sowie das Deployment der Rest API und der Nextcloud Lösung als Docker Container auf dem Server beschrieben.
Sämtliche Arbeitsschritte sind durch die Terminal Befehle dokumentiert, die während der Umsetzung ausgeführt wurden.

2 Vorbereitungen

Bevor die Aufgabe selbst begonnen werden konnte, musste zunächst eine SD Karte mit dem Raspbian OS beschrieben werden. Diese Aufgabe wurde von Herrn Wichmann durchgeführt.
Nachdem das Raspbian OS erfolgreich auf der SD Karte installiert wurde und die SD-Karte im Server verbaut wurde, musste der Server per Ethernet Kabel an ein Netzwerk angeschlossen werden, zu dem der Umsetzende Zugriff hatte.
Schlussendlich wurde eine Verbindung zum Server per SSH aufgebaut. Da noch keine Änderungen am Betriebssystem vorgenommen wurden, wurde die Verbindung mit dem Benutzer „pi“ und dem Passwort “raspberry“ aufgebaut:

```
C:\Users\gkoppe>ssh pi@192.168.24.113
pi@192.168.24.113's password:
```

3 Konfiguration der User

Nachdem das Betriebssystem des Servers installiert, der Server in das Netzwerk eingebunden und überprüft wurde, ob eine SSH Verbindung zum Server möglich ist, wurden zwei neue User angelegt, um den Server abzusichern, da der User „Pi“ der Standarduser des Betriebssystems ist und somit allgemein bekannt.
Es wurden insgesamt zwei neue User angelegt. Ein User „benutzer72“ mit grundlegenden Nutzungsrechten und ein Benutzer „fernzugriff“ mit administrativen Rechten. Des Weiteren kann der Benutzer „fernzugriff“ verwendet werden, um eine SSH-Verbindung zum Server aufzubauen.

3.1 Anlage neuer Benutzer

Zunächst wurde der user „benutzer72“ mit folgenden Befehlen angelegt:

[1] Fortan „Server“

```
pi@raspberry:~ $ sudo useradd -m benutzer72
pi@raspberry:~ $ sudo passwd benutzer72
New password:
Retype new password:
passwd: password updated successfully
```

Der Befehl `useradd` dient dazu, den neuen Benutzer anzulegen. Mit dem flag `-m` wird außerdem automatisch ein Home-Verzeichnis für den neuen Benutzer erzeugt. Des Weiteren wurde dem neuen Benutzer mittels des `passwd` Befehls ein neues Passwort zugewiesen.
Nachdem der Benutzer benutzer72 konfiguriert wurde, wurde ein neuer administrativer Nutzer „fernzugriff“ angelegt. Die Vorgehensweise war hier zunächst identisch zu der der Neuanlage von benutzer72:

```
pi@raspberry:~ $ sudo useradd -m fernzugriff
pi@raspberry:~ $ sudo passwd fernzugriff
New password:
Retype new password:
passwd: password updated successfully
```

3.2 Konfiguration des administrativen Benutzers

Da der Benutzer „fernzugriff“ administrative Rechte auf dem Server erhalten sollte, wurde er anschließend in die Gruppe „sudo“ aufgenommen:

```
pi@raspberry:~ $ sudo usermod -aG sudo fernzugriff
```

Hier dient der Befehl `usermod` allgemein dazu, einen User zu modifizieren. Der Flag `-aG` gibt an, dass der User einer Gruppe hinzugefügt werden soll, welche wiederum direkt hinter dem Flag definiert ist (in diesem Fall „sudo“).
Abschließend wurde dem Benutzer „fernzugriff“ noch das Recht gewährt, sich per SSH mit dem Server zu verbinden. Dafür wurde die Einstellung `AllowUsers` in der Datei `/etc/ssh/sshd_config`:

```
pi@raspberry:~ $ sudo nano /etc/ssh/sshd_config
```

```
        $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $

#Port 22
#AddressFamily any
AllowUsers     fernzugriff
#...
```

Nun besitzt der Benutzer „fernzugriff“ alle notwendigen Rechte, um ihn für administrative Tätigkeiten zu verwenden. Fortan wird der Benutzer „pi“ nicht mehr verwendet und alle Umsetzungen werden mit dem Benutzer „fernzugriff“ durchgeführt.
Um das System abzusichern, wurde außerdem, nachdem eine neue Verbindung mit dem Benutzer „fernzugriff“ aufgebaut wurde, das Passwort des Benutzers „pi“ geändert:

```
fernzugriff@raspberry:~ $ sudo passwd pi
New password:
Retype new password:
passwd: password updated successfully
```

4 Konfiguration des Netzwerks

Bislang nutzte der Server die IP-Adresse, welche ihm vom DHCP-Server des Netzwerks zugewiesen wurde. Fortan sollen aber statische Einstellungen für IP-Adresse, DNS-Server und Router verwendet werden.
Um dies zu konfigurieren, musste die Datei `/etc/dhcpcd.conf` angepasst werden:

```
fernzugriff@raspberry:~ $ sudo nano /etc/dhcpcd.conf
```

```
# Example static IP configuration
#interface eth0
static ip_address=192.168.24.113/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.24.254
static domain_name_servers=192.168.24.254
#...
```

Um diese Änderungen in Kraft zu setzen, gab es zwei Möglichkeiten. Entweder musste der dhcpcd Dienst über folgenden Befehl neugestartet werden:

```
fernzugriff@raspberry:~ $ sudo systemctl restart dhcpcd.service
```

Da durch den Neustart des Dienstes aber gleichzeitig auch die IP-Adresse geändert würde und somit die SSH Verbindung abgebrochen wäre, wurde sich dafür entschieden, den Server komplett neuzustarten:

```
fernzugriff@raspberry:~ $ sudo reboot
```

Somit ist der Server vollständig in das Netzwerk eingebunden.

5 Einrichten der Firewall

5.1 Installation

Im Standard ist auf Linux-basierten Systemen die Firewall IPTables installiert. Diese wurde zunächst um das Tool „UFW“[2] erweitert, welches es erlaubt, IPTables über simple Befehle zu konfigurieren. Um UFW zu installieren, wurde zunächst der apt Package Index aktualisiert, um die neueste Version von UFW im Zugriff zu haben:

```
fernzugriff@raspberry:~ $ sudo apt update
```

Anschließend wurde UFW installiert:

```
fernzugriff@raspberry:~ $ sudo apt install ufw
```

5.2 Konfiguration

Nachdem UFW erfolgreich installiert wurde, wurden die notwendigen Firewall Regeln für den Server eingerichtet.
Zunächst musste sichergestellt werden, dass SSH Verbindungen zum Server auch nach Aktivierung der Firewall weiterhin möglich sein würden, allerdings nur aus dem Netzwerk, in dem der Server sich befindet. Da dem Server zuvor, wie in Kapitel 4 beschrieben, eine statische IP-Adresse und somit statisch ein Netzwerk zugewiesen wurden, muss hierbei nicht darauf geachtet werden, dass das Netzwerk sich in der Zukunft ändern könnte.
Um SSH weiterhin zuzulassen, musste der Port 22 für alle eingehenden Pakete aus dem Netzwerk 192.168.24.0/24 freigeschaltet werden:

[2]Vgl. Quelle 1 in Kapitel 9.1.1

```
fernzugriff@raspberry:~ $ sudo ufw allow from 192.168.24.0/24
                          proto tcp to any port 22
Rules updated
```

Da im Standard bereits Regeln für den Port 22 definiert sind, mussten diese gelöscht werden. Der Grund dafür, dass zuerst eine neue Regel eingerichtet wurde, ist, dass ansonsten die Gefahr bestanden hätte, das System nicht mehr per SSH zu erreichen.
Um die alten SSH Regeln zu löschen, wurden ihre IDs benötigt. Um diese sehen zu können, musste UFW zunächst aktiviert werden, um anschließend alle Regeln anzeigen lassen zu können:

```
fernzugriff@raspberry:~ $ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
fernzugriff@raspberry:~ $ sudo ufw status numbered
Status: active

     To         Action      From
     --         ------      ----
[ 1] 22/tcp     ALLOW IN    192.168.24.0/24
[ 2] 22         ALLOW IN    Anywhere
[ 3] 22 (v6)    ALLOW IN    Anywhere (v6)
```

Die Regeln mit den IDs 2 und 3 mussten nun gelöscht werden, damit die selbst erstellte Regel für SSH Verbindungen korrekt arbeitet:

```
fernzugriff@raspberry:~ $ sudo ufw delete 3
Deleting:
    allow 22 (v6)
Proceed with operation (y|n)? y
Rule deleted
fernzugriff@raspberry:~ $ sudo ufw delete 2
Deleting:
    allow 22
Proceed with operation (y|n)? y
Rule deleted
```

Des Weiteren sollten ausgehende DNS Anfragen erlaubt werden. Das DNS Protokoll arbeitet über Port 53, entsprechend musste dieser für ausgehende Pakete freigegeben werden:

```
fernzugriff@raspberry:~ $ sudo ufw allow out to any port 53
Rule added
Rule added (v6)
```

Der Port, über den die Rest API, nachdem sie deployed sein würde, erreichbar sein soll, wurde ebenfalls freigegeben:

```
fernzugriff@raspberry:~ $ sudo ufw allow 13376
Rule added
Rule added (v6)
```

Außerdem soll der Webserver aus allen Netzwerken erreichbar sein. Da wir die HTTP Kommunikation nicht mit SSL verschlüsseln werden, reicht es hier, den Port 80 für alle Netzwerke freizugeben:

```
fernzugriff@raspberry:~ $ sudo ufw allow 80
Rule added
Rule added (v6)
```

Alle Pakete, die keiner dieser Regeln entsprechen, sollten abgelehnt werden. Um dies zu erreichen, wurde die Standard Regel für eingehende Pakete angepasst:

```
fernzugriff@raspberry:~ $ sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
```

Nachdem alle Regeln eingerichtet wurden, wurde noch einmal über den UFW Status überprüft, ob alle Regeln existierten:

```
fernzugriff@raspberry:~ $ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 13376                      ALLOW IN    Anywhere
[ 2] 22/tcp                     ALLOW IN    192.168.24.0/24
[ 3] 53                         ALLOW OUT   Anywhere                   (out)
[ 4] 80                         ALLOW IN    Anywhere
[ 5] 13376 (v6)                 ALLOW IN    Anywhere (v6)
[ 6] 53 (v6)                    ALLOW OUT   Anywhere (v6)              (out)
[ 7] 80 (v6)                    ALLOW IN    Anywhere (v6)
```

6 Einrichtung Webserver

Nachdem die Firewall Regeln eingerichtet wurden, konnte der Webserver installiert und konfiguriert werden. Die Wahl fiel hier auf den Webserver Apache2.

6.1 Installation

Da der Package Index bereits aktualisiert wurde, konnte der Webserver direkt installiert werden:

```
fernzugriff@raspberry:~ $ sudo apt install apache2
```

6.2 Konfiguration Webserver

Da der Apache2 Server im Standard bereits die Ports für HTTP abhört, mussten diese nicht mehr konfiguriert werden.
Nach der Installation wurde zunächst eine einfache Startseite für den Webserver eingerichtet. Dafür musste die Datei `/var/www/html/index.html`, welche bei der Installation des Apache2 angelegt wurde, angepasst werden:

```
fernzugriff@raspberry:~ $ cd /var/www/html
fernzugriff@raspberry:/var/www/html $ sudo nano index.html
```

```
<html>
    <body>
        <h1>Todo-Listen Verwaltung</h1>
        <h3>Gerrit Koppe</h3>
    </body>
</html>
```

Diese konnte nun erreicht werden, sobald die IP-Adresse des Servers im Browser aufgerufen wurde.

7 Einrichtung Rest API auf Betriebsystem

Um die Rest API auf dem Betriebsystem einzurichten, mussten zunächst die Skripte auf dem Server abgelegt werden. Zunächst wurde ein neuer Ordner `todo-api` im Home Verzeichnis des Users fernzugriff angelegt. Es wurde sich hier für das Homeverzeichnis entschieden, da die API Dateien und Ordner anlegt und das Skript ursprünglich nicht dafür gedacht war, auf einem Linux System zu arbeiten.

```
fernzugriff@raspberry:~ $ sudo mkdir todo-api
```

Da die Rest API sich in einem Git Repository befindet, konnte dieses in den neu erstellten Ordner geklont werden:

```
fernzugriff@raspberry:~ $ cd todo-api
fernzugriff@raspberry:~/todo-api $ sudo git clone
       "https://github.com/LeichtMatrosee/TodoRestServer.git"
```

Nachdem die Rest API auf den Server geklont wurde, mussten noch alle Python Libraries nachinstalliert werden, welche noch nicht auf dem Server existierten (in diesem Fall nur `flask_cors`, dies kann sich bei anderen Installationen unterscheiden).
Anschließend wurde überprüft, ob die API lauffähig war:

```
fernzugriff@raspberry:~/todo-api $ cd TodoRestServer/src
fernzugriff@raspberry:~/todo-api/TodoRestServer/src $ python3 App.py
```

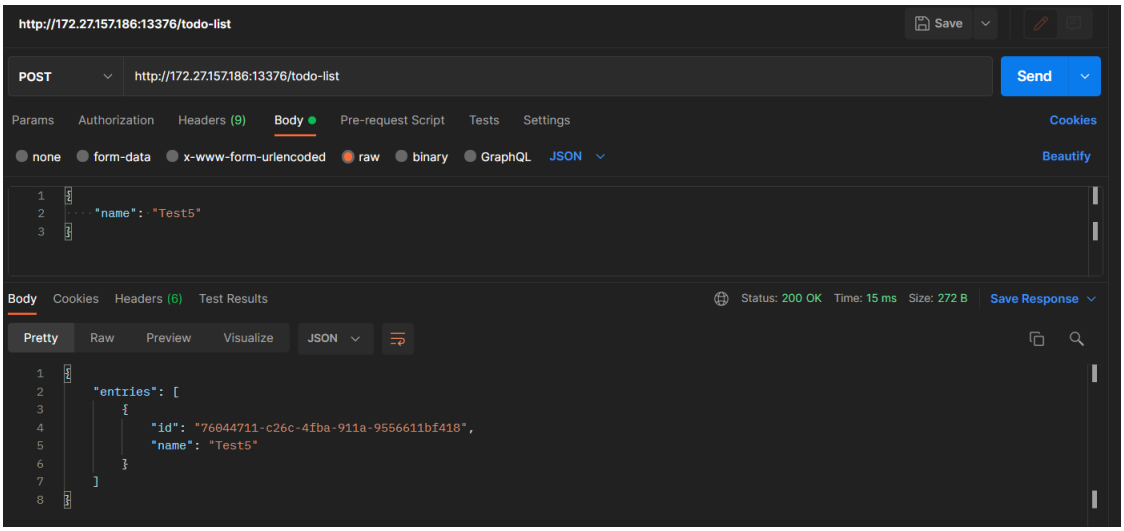

Abbildung 1: Teste POST

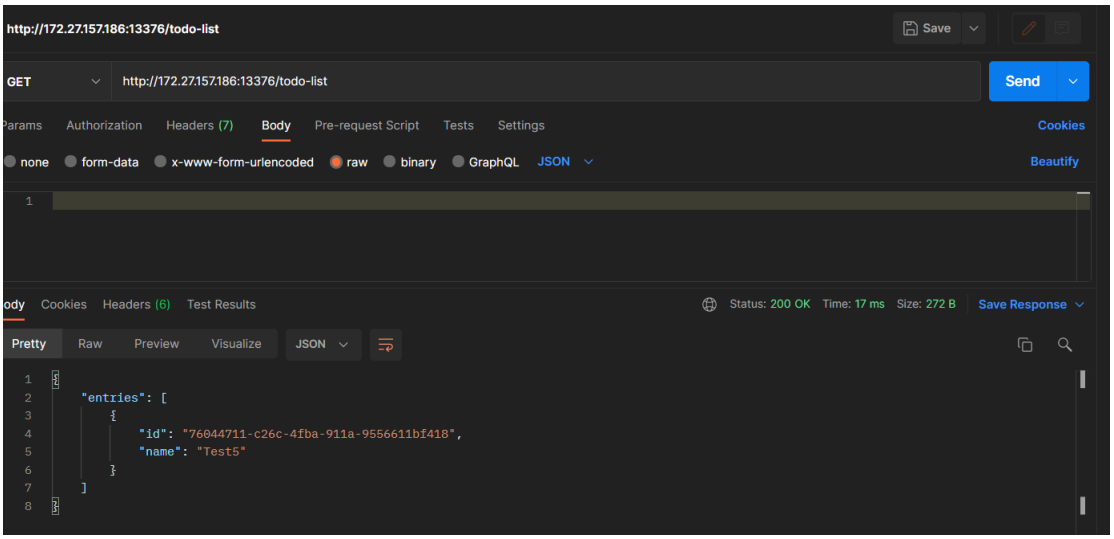

Abbildung 2: Teste GET

Es lassen sich neue Todo-Listen anlegen, sowie die existenten Todo-Listen abrufen, die Rest API wurde somit erfolgreich auf dem System installiert.
Nun muss noch sichergestellt werden, dass die API auch nach Neustart des Systems noch erreichbar ist. Dafür muss ein Dienst angelegt werden, der das Skript bei Start des Systems ausführt. Um den Dienst anzulegen, muss eine `.service` Datei im Systemverzeichnis des Servers angelegt werden:

```
fernzugriff@raspberry:~/todo-api $ cd /lib/systemd/system
fernzugriff@raspberry:/lib/systemd/system $ sudo nano todoapi.service
```

In die Datei muss folgender Text eingefügt werden:

```
[Unit]
Description=Todo Rest API
After=multi-user.target
Conflicts=getty@tty1.service

[Service]
Type=simple
ExecStart=/usr/bin/python3
    /home/fernzugriff/todo-api/TodoRestServer/src/App.py
WorkingDirectory=/home/fernzugriff/todo-api/working
StandardInput=tty-force

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

ExecStart ist hier der Befehl, der bei Start des Dienstes ausgeführt wird. Da ein Python Skript ausgeführt werden soll, Dienste aber nicht mit Umgebungsvariablen arbeiten und somit den Befehl python3 nicht direkt ausführen können, muss hier der gesamte Pfad zur python3 Datei angegeben werden.
WorkingDirectory gibt den Ordner an, in dem der Befehl ausgeführt werden soll. Schreibt der Befehl zum Beispiel in das Dateisystem, kann dieser Parameter genutzt werden, um Ablageort des ausgeführten Programms, sowie Speicherort der vom Programm geschriebenen Dateien voneinander zu trennen.
Da Dienste immer durch den User „root“ ausgeführt werden, die fehlenden Python-Module aber mit einem anderen Benutzer heruntergeladen wurden, müssen diese nun auch für den root user heruntergeladen werden. Dafür wird keine Anmeldung als root benötigt, es reicht, die Installation mit sudo Berechtigungen auszuführen:

```
fernzugriff@raspberry:~ $ sudo pip install flask-cors
```

Falls Unklarheit besteht, welche Module dem root User fehlen, kann das Skript der API mit sudo Berechtigungen ausgeführt werden, um zu überprüfen, ob es möglich ist, dieses mit erhöhten Berechtigungen auszuführen.
Nachdem der Dienst angelegt wurde, muss er nun noch aktiviert und gestartet werden. Zuvor muss aber der Daemon einmal neu geladen werden, damit dieser den neuen Dienst registriert:

```
fernzugriff@raspberry:~ $ sudo systemctl daemon-reload
fernzugriff@raspberry:~ $ sudo systemctl enable todoapi.service
Created symlink /etc/systemd/system/multi-user.target.wants/todoapi.service
    → /lib/systemd/system/todoapi.service.
fernzugriff@raspberry:~ $ sudo systemctl start todoapi.service
```

Durch `systemctl enable` wird der Dienst zunächst dem korrekten Userverzeichnis hinzugefügt, wodurch er in dem entsprechenden Kontext gestartet werden kann. Durch `systemctl start` wird der Dienst gestartet.
Nun wird das Skript beim Start des Servers direkt hochgefahren.

8 Nextcloud Container deployen

Im Folgenden wird beschrieben, wie die Filehosting Lösung Nextcloud als Docker Container deployed wurde.

8.1 Installation Docker

Um Nextcloud als Container zu installieren, muss zunächst Docker installiert werden. Da Docker von manchen Linux-Distributionen ausgeliefert wird, wurden zunächst alle Pakete deinstalliert, die in Konflikt mit den zu installierenden Paketen stehen könnten:

```
fernzugriff@raspberry:~ $ for pkg in docker.io docker-doc docker-compose
    podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

Der Befehl iteriert über alle Pakete in den angegebenen Repositories und deinstalliert die lokalen Kopien dieser Pakete, sollten sie installiert sein.
Nachdem die Pakete erfolgreich deinstalliert wurden, musten zunächst Hilfspakete installiert werden, die es ermöglichen, die notwendigen Voraussetzungen für Docker zu installieren:

```
fernzugriff@raspberry:~ $ sudo apt-get install ca-certificates curl gnupg
```

Anschließend musste der offizielle Docker GPG Key installiert werden, der dazu dient, die Kommunikation zwischen Docker und dem Docker Repository zu verschlüsseln[3]:

```
fernzugriff@raspberry:~ $ sudo install -m 0755 -d /etc/apt/keyrings
fernzugriff@raspberry:~
    $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
    --dearmor -o /etc/apt/keyrings/docker.gpg
fernzugriff@raspberry:~ $ sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Durch `chmod a+r` wurde des Weiteren jedem User die Berechtigung zur Nutzung des GPG Keys gegeben.
Nachdem der GPG Key installiert und für jeden Benutzer freigegeben war, konnte das Repository für Docker aufgesetzt werden:

[3]Vgl. Quelle 2 in Kapitel9.1

```
fernzugriff@raspberry:~ $ echo \
"deb [arch="$(dpkg --print-architecture)"
    signed-by=/etc/apt/keyrings/docker.gpg]
    https://download.docker.com/linux/ubuntu \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Final konnte nun Docker installiert werden:

```
fernzugriff@raspberry:~ $ sudo apt-get install docker-ce docker-ce-cli
    containerd.io docker-buildx-plugin docker-compose-plugin
```

Um zu überprüfen, ob die Installation erfolgreich war, konnte nun ein einfaches Docker Image gestartet werden:

```
fernzugriff@raspberry:~ $ sudo docker run hello-world
```

8.2 Einrichtung Nextcloud als Docker Container

Da Docker automatisch alle Images, die nicht bereits installiert sind, bei Ausführung aus der Image Registry herunterlädt und installiert, muss nur der Befehl ausgeführt werden, um das Nextcloud Image zu starten. Da Docker Container nicht persistent sind, muss allerdings noch ein Dateiverzeichnis angegeben werden, in dem die Dateien gespeichert werden:

```
fernzugriff@raspberry:~ $ sudo mkdir "nc-files"
fernzugriff@raspberry:~
    $ sudo docker run -v "/home/fernzugriff/nc-files" nextcloud
```

Nachdem der Befehl ausgeführt wurde, wird zunächst das Image, falls es noch nicht lokal installiert ist, heruntergeladen. Anschließend wird das Image als Container ausgeführt, wobei der Pfad, der hinter dem `-v` Parameter angegeben ist, als Volume für den Container dient.
Damit der Nextcloud Container auch beim Start des Systems verfügbar ist, musste auch in diesem Fall wieder ein Dienst angelegt werden. Da die Konfiguration identisch zu der aus Kapitel 7 ist, wird das Vorgehen hier nicht weiter erläutert:

```
fernzugriff@raspberry:~ $ cd /lib/systemd/system
fernzugriff@raspberry:/lib/systemd/system $ sudo nano nextcloud.service
```

```
[Unit]
Description=Nextcloud Service
After=multi-user.target
Conflicts=getty@tty7.service

[Service]
Type=simple
ExecStart=/usr/bin/docker run -v "home/fernzugriff/nc-files" nextcloud
StandardInput=tty-force

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

```
fernzugriff@raspberry:~ $ sudo systemctl daemon-reload
fernzugriff@raspberry:~ $ sudo systemctl enable nextcloud.service
Created symlink /etc/systemd/system/multi-user.target.wants/todoapi.service
    → /lib/systemd/system/nextcloud.service.
fernzugriff@raspberry:~ $ sudo systemctl start nextcloud.service
```

Um zu überprüfen, ob der neue Dienst aktiv ist, wurden zuletzt noch alle aktiven Docker Images angezeigt:

```
fernzugriff@raspberry:~ $ sudo docker ps
CONTAINER IDs     Image      COMMAND                  CREATED        #...
5f3bc0925087      nextcloud  "/entrypoint.sh apac..." 50 seconds ago #...
```

Nextcloud war somit erfolgreich installiert, die Daten persistiert und der Container wurde bei Start des Systems hochgefahren.

9 Anlagen

9.1 Quellen

9.1.1 Internetquellen

1. Ubuntu Wiki: UncomplicatedFirewall. 2022 - Online unter: `https://wiki.ubuntu.com/UncomplicatedFirewall` [11.06.2023]
2. GnuPG: The GNU Privacy Guard. Online unter: `https://gnupg.org/` [11.06.2023]