

# Webserver Raspberry Pi Zero W (Komplettlösung)

Installation mit Raspbian und RaspberryPi Zero W eines Webservers mit PHP7.3, phpMyAdmin und MySQL (MariaDB). Ausserdem einrichten eines FTP-Servers mit entsprechendem User. Zugriff von aussen mit DynDNS (No-IP) und sicheres Zertifikat (SSL).

## Hardware / Software

Raspberry Pi Zero W

Raspbian Buster 20.06.2019

## Quellen:

<https://tutorials-raspberrypi.de/webserver-installation-apache2/>

<https://tutorials-raspberrypi.de/webserver-installation-homeverzeichnis-aendern/>

<https://tutorials-raspberrypi.de/webserver-installation-php-5/>

<https://tutorials-raspberrypi.de/webserver-installation-teil-3-mysql/>

<https://pimylifeup.com/raspberry-pi-mysql/>

<https://tutorials-raspberrypi.de/webserver-installation-teil-4-phpmyadmin/>

<https://tutorials-raspberrypi.de/webserver-installation-teil-5-ftp-server/>

[https://www.thomas-krenn.com/de/wiki/FTP-Server unter Debian einrichten](https://www.thomas-krenn.com/de/wiki/FTP-Server_unter_Debian_einrichten)

<https://tutorials-raspberrypi.de/webserver-installation-teil-6-dns-server-via-no-ip/>

<https://tutorials-raspberrypi.de/raspberry-pi-ssl-zertifikat-kostenlos-mit-lets-encrypt-erstellen/>

[https://www.thomas-krenn.com/de/wiki/Webserver Verzeichnisse mit Passwort schützen](https://www.thomas-krenn.com/de/wiki/Webserver_Verzeichnisse_mit_Passwort_sch%C3%BCtzen)

<https://www.webhosterwissen.de/know-how/eigener-webserver/tutorial-e-mails-mit-eigenem-server-versenden-per-smtp-ubuntu-18-04/>

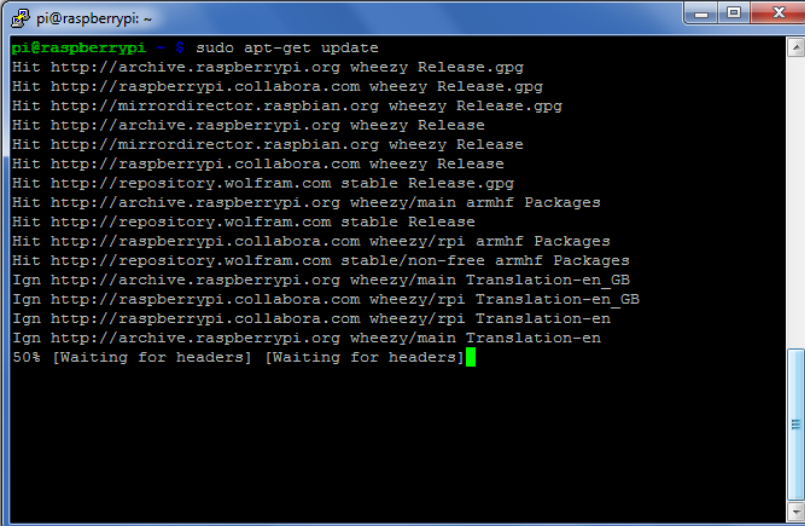
## Inhalt

1. Apache2 installieren .....	2
2. Wie ändere ich das Homeverzeichnis? .....	3
3. PHP7.3 installieren .....	4
4. MySQL (MariaDB) installieren.....	5
5. phpMyAdmin installieren .....	6
6. FTP-Server (proftpd) mit User einrichten .....	10
7. Zugriff Aussen mit Portforwarding (DynDNS mit No-IP).....	13
8. Autostart: Programm automatisch starten lassen.....	16
9. SSL-Zertifikat installieren (Zertifikat mit Letsencrypt) .....	18
10. Webserver-Verzeichnis mit Passwort schützen.....	21
11. Mails mit ext. Host (GMAIL) über Webserver versenden .....	23

# 1. Apache2 installieren

Eines der meistgenutzten Gebiete des Pi's ist wohl die Benutzung als Webserver. In den folgenden Tutorials wird Schritt-für-Schritt ein Webserver inkl. aller Services und was dazu benötigt wird eingerichtet. Den Anfang dazu bildet die Installation des Apache 2 HTTP Servers. Als aller erstes müssen die Pakete natürlich auf dem neusten Stand sein.

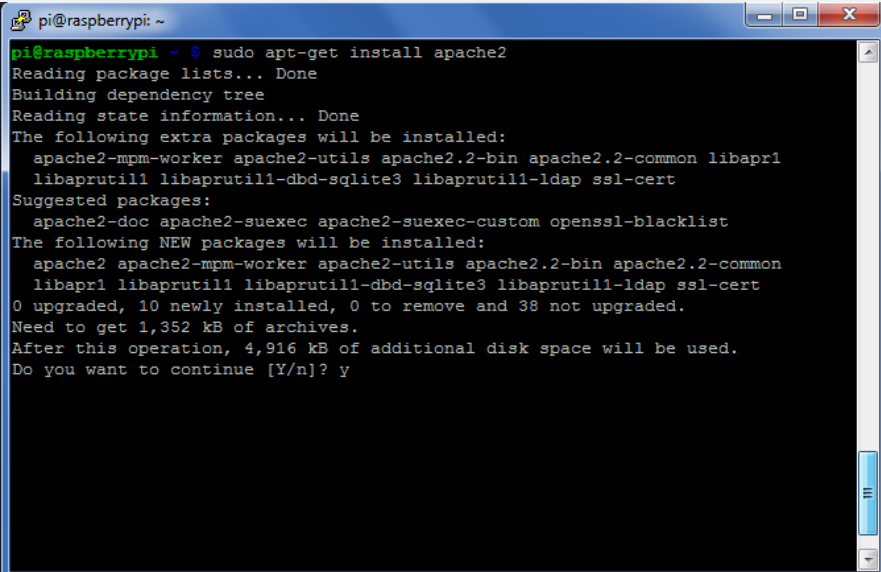
```
sudo apt-get update
```



```
pi@raspberrypi: ~  
pi@raspberrypi ~$ sudo apt-get update  
Hit http://archive.raspberrypi.org wheezy Release.gpg  
Hit http://raspberrypi.collabora.com wheezy Release.gpg  
Hit http://mirrordirector.raspbian.org wheezy Release.gpg  
Hit http://archive.raspberrypi.org wheezy Release  
Hit http://mirrordirector.raspbian.org wheezy Release  
Hit http://raspberrypi.collabora.com wheezy Release  
Hit http://repository.wolfram.com stable Release.gpg  
Hit http://archive.raspberrypi.org wheezy/main armhf Packages  
Hit http://repository.wolfram.com stable Release  
Hit http://raspberrypi.collabora.com wheezy/zpi armhf Packages  
Hit http://repository.wolfram.com stable/non-free armhf Packages  
Ign http://archive.raspberrypi.org wheezy/main Translation-en_GB  
Ign http://raspberrypi.collabora.com wheezy/zpi Translation-en_GB  
Ign http://raspberrypi.collabora.com wheezy/zpi Translation-en  
Ign http://archive.raspberrypi.org wheezy/main Translation-en  
50% [Waiting for headers] [Waiting for headers]
```

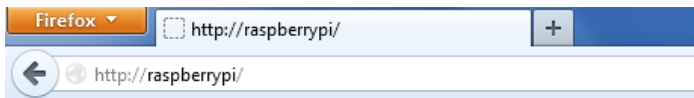
Danach wird der Apache2 noch herunter geladen und installiert.

```
sudo apt-get install apache2
```



```
pi@raspberrypi: ~  
pi@raspberrypi ~$ sudo apt-get install apache2  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following extra packages will be installed:  
  apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common libapr1  
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap ssl-cert  
Suggested packages:  
  apache2-doc apache2-suexec apache2-suexec-custom openssl-blacklist  
The following NEW packages will be installed:  
  apache2 apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common  
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap ssl-cert  
0 upgraded, 10 newly installed, 0 to remove and 38 not upgraded.  
Need to get 1,352 kB of archives.  
After this operation, 4,916 kB of additional disk space will be used.  
Do you want to continue [Y/n]? y
```

Und schon ist die Installation fertig. Um zu prüfen, ob alles geklappt hat, geben wir im Browser entweder <http://raspberrypi/> (Achtung: keine Domain-Endung) oder die interne IP-Adresse (192.168.0.xxx) ein.



## It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Ab sofort können Dateien ins Verzeichnis **/var/www** hochgeladen werden und sind dann entsprechend unter *http://raspberrypi/dateiname* zu erreichen, allerdings vorerst nur lokal aus dem Netzwerk und nicht von ausserhalb.

Wer will, kann auch das [Homeverzeichnis ändern](#).

Da bisher nur einfache HTML Dateien angezeigt werden können, erweitern wir im [nächsten Tutorial](#) den Pi um PHP, um auch nicht-statische Inhalte anzeigen zu können.

## 2. Wie ändere ich das Homeverzeichnis?

In diesem Tutorial wird das Ändern des Homeverzeichnis des zuvor eingerichteten Webserver gezeigt, womit z.B. direkt per FTP ins Hauptverzeichnis zugegriffen werden kann.

Wer das Standardverzeichnis von **/var/www** in ein anderes ändern will, geht wie folgt vor:

Der Pfad muss natürlich bestehen, ich erstelle testweise in meinem Homeverzeichnis (**/home/pi**) den Order *tutorials-raspberrypi.de*

```
sudo mkdir /home/pi/tutorials-raspberrypi.de
```

Nun muss der Pfad entsprechend in der Konfigurationsdatei von Apache2 geändert werden

```
sudo nano /etc/apache2/sites-available/default
```

Wir ändern den Pfad **/var/www** in **/home/pi/tutorials-raspberrypi.de** um.

```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/apache2/sites-available/default

<VirtualHost *:80>
  ServerAdmin webmaster@localhost

  DocumentRoot /home/pi/tutorials-raspberrypi.de/
  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>
  <Directory /home/pi/tutorials-raspberrypi.de/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>

  ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
  <Directory "/usr/lib/cgi-bin">
    AllowOverride None
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    Order allow,deny

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Als nächstes kopiere ich die index.html Datei in das neue Verzeichnis und benenne sie um (sind bereits Dateien vorhanden, ist das natürlich nicht nötig).

```
sudo cp /var/www/index.html /home/pi/tutorials-raspberrypi.de/index_neu.html
```

Jetzt noch den Server neustarten

```
sudo service apache2 restart
```

Rufen wir nun [http://raspberrypi/index\\_neu.html](http://raspberrypi/index_neu.html) auf (bzw. entsprechend mit IP 192.168.0.xxx), wird die Datei im neuen Verzeichnis angezeigt.

### 3. PHP7.3 installieren

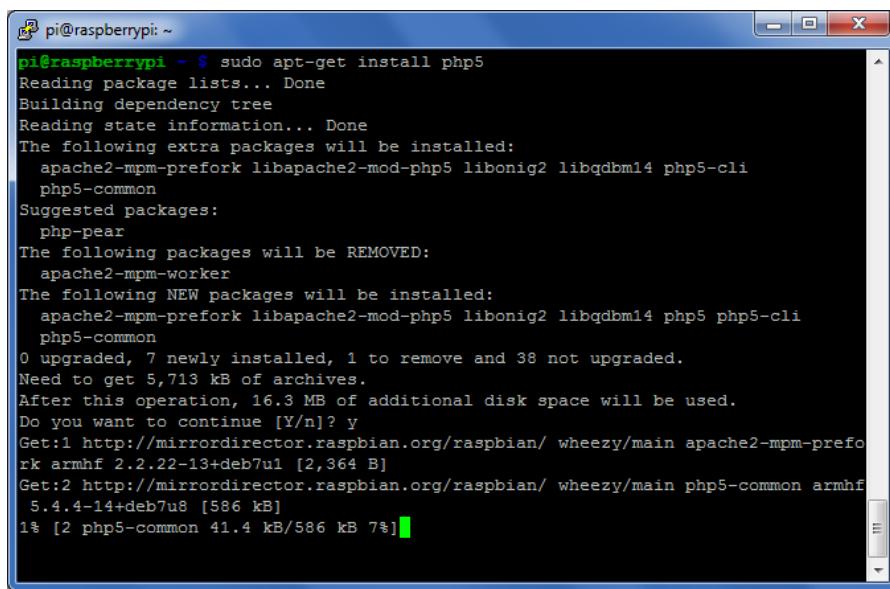
Teil 2: Um nicht nur reines HTML anzeigen und den Content dynamisch gestalten zu können, benötigen wir PHP. Dazu richten wir die benötigten Pakete ein und testen die Installation auf Funktionstüchtigkeit.

Die Installation von PHP7.3 ist ähnlich einfach wie die von Apache:

```
sudo apt-get install php7.3
```

PHP-Pakete installieren:

```
sudo apt install php7.3-cli php7.3-common php7.3-curl php7.3-mbstring  
php7.3-mysql php7.3-xml
```



```
pi@raspberrypi: ~  
pi@raspberrypi ~$ sudo apt-get install php5  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following extra packages will be installed:  
  apache2-mpm-prefork libapache2-mod-php5 libonig2 libqdbm14 php5-cli  
  php5-common  
Suggested packages:  
  php-pear  
The following packages will be REMOVED:  
  apache2-mpm-worker  
The following NEW packages will be installed:  
  apache2-mpm-prefork libapache2-mod-php5 libonig2 libqdbm14 php5 php5-cli  
  php5-common  
0 upgraded, 7 newly installed, 1 to remove and 38 not upgraded.  
Need to get 5,713 kB of archives.  
After this operation, 16.3 MB of additional disk space will be used.  
Do you want to continue [Y/n]? y  
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main apache2-mpm-prefo  
rk armhf 2.2.22-13+deb7u1 [2,364 B]  
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main php5-common armhf  
 5.4.4-14+deb7u8 [586 kB]  
1$ [2 php5-common 41.4 kB/586 kB 7$]
```

Nachdem die Installation fertig ist, wollen wir auch diesmal testen, ob alles geklappt hat.

Wir wechseln also in das Verzeichnis:

```
cd /var/www/html
```

Hier erstellen wir eine neue Datei *phpinfo.php*

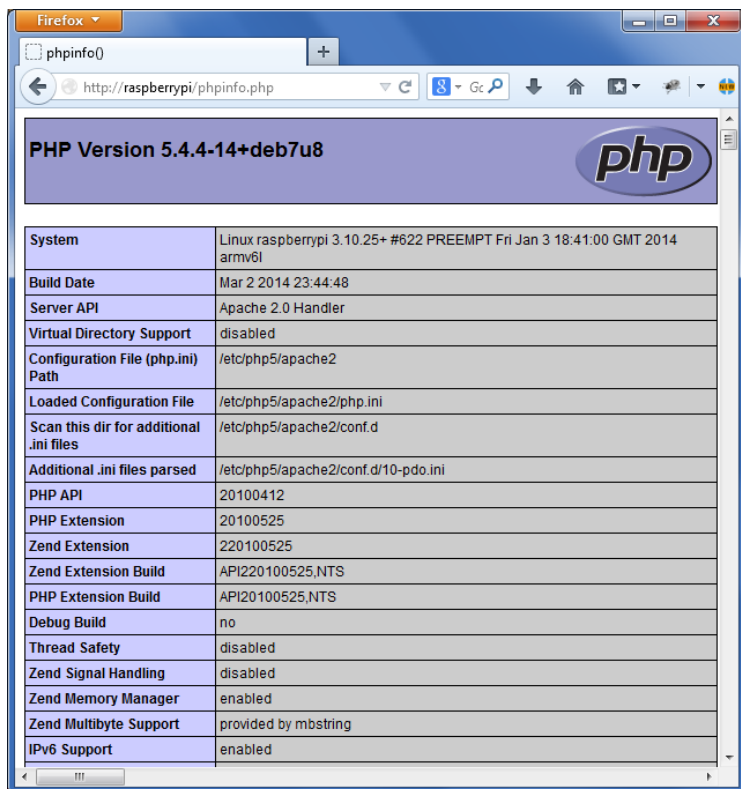
```
sudo nano "phpinfo.php"
```

und schreiben hinein

```
<?php  
  phpinfo();  
?>
```

Gespeichert wird wie immer mit STRG + O und mit STRG + X der Editor geschlossen.

Öffnen wir nun die Adresse <http://raspberrypi/phpinfo.php> bekommen wir folgendes zu sehen:



Das war es auch schon. Jetzt können bereits erste Skripte und Funktionen in PHP geschrieben werden.

Weiter geht es mit der [Installation von MySQL](#).

## 4. MySQL (MariaDB) installieren

MySQL-Server und mysql-client lassen sich nicht mehr installieren. Aus diesem Grund muss MariaDB (auch OpenSource und Freeware und effizienter) installiert werden:

```
sudo apt-get install mariadb-server
```

If you already have MySQL installed, it is likely that the package manager notifies you of a conflict and asks if it needs to uninstall MySQL. In this case, answer yes.

During the installation, MariaDB will configure itself. It's up to you to provide the administrator account for the database.

This done, it should ask you if you are sure to want to go under MariaDB. Again, answer yes.

Once the installation is complete, you will be able to access MariaDB as you did with MySQL, simply with the following command:

```
mysql -u user -p
```

Installieren des Admin-Passworts, entfernen des Blank-Users und der Test-Database:

```
sudo mysql_secure_installation
```

Zugriff auf die DB:

```
sudo mysql
```

```
systemctl start mariadb.service  
systemctl status mysqld service
```

## 5. phpMyAdmin installieren

Teil 4: Nach der Installation von Apache, PHP und MySQL ist phpMyAdmin an der Reihe. Dieser Schritt ist nicht unbedingt nötig, aber für das einfachere Auslesen der Daten im Browser sehr nützlich. Es bietet eine einfache Benutzeroberfläche, wobei in diesem Teil die Installation, Einrichtung und Benutzung gezeigt wird.

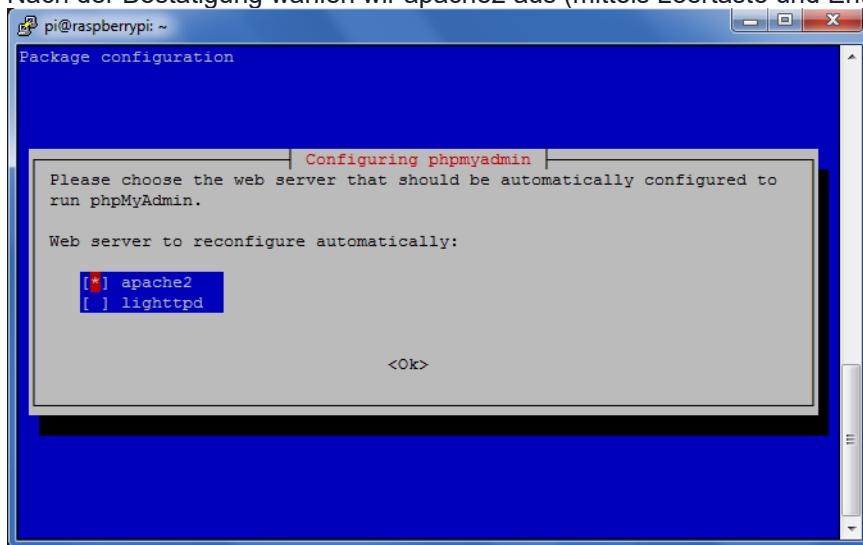
Zwar würde der Webserver auch ohne phpMyAdmin auskommen, aber zum Anzeigen der Datensätze ist dieser ganz praktisch.

Los geht es wieder mit dem Herunterladen und Installieren der Pakete

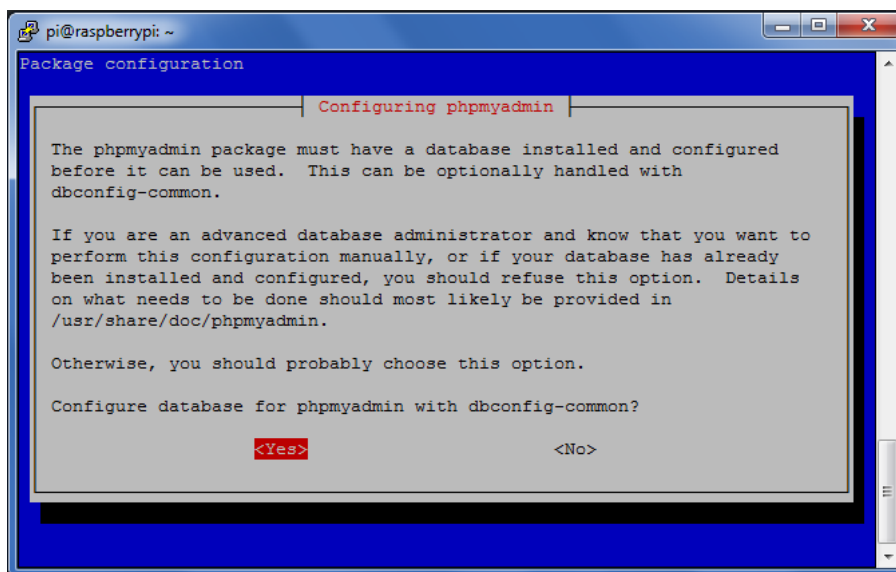
```
sudo apt-get install php5-mysql libapache2-mod-auth-mysql phpMyAdmin
```

```
sudo apt-get install php5-mysql phpmyadmin
```

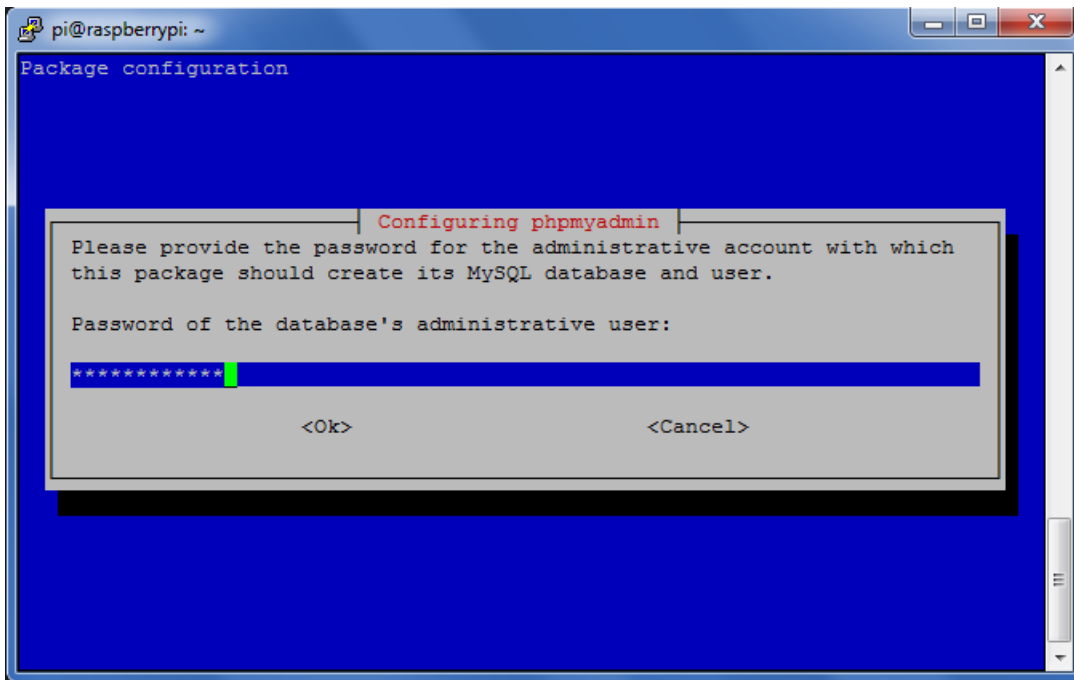
Nach der Bestätigung wählen wir apache2 aus (mittels Leertaste und Enter) und setzen es fort.



Daraufhin wird gefragt, ob einige Datenbanken, die phpMyAdmin benötigt erstellt werden sollen. Mit bestätigen und weiter. Danach immer root Admin-User angeben:

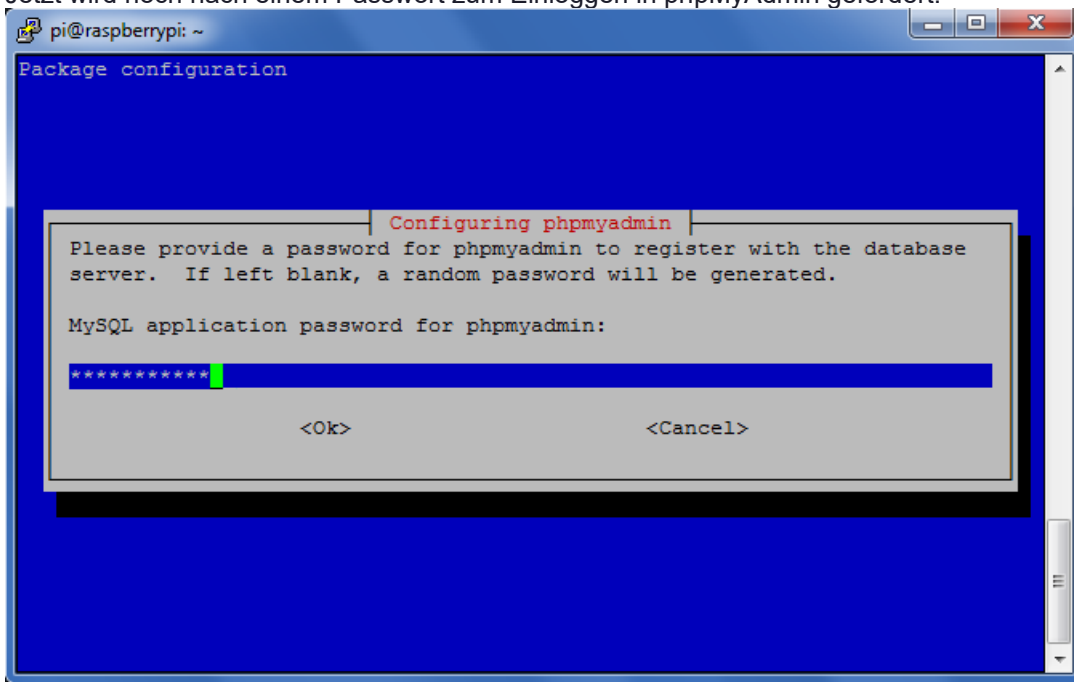


Als nächstes wird nach dem Passwort, welches für den root-User angelegt wurde gefragt:



phpMyAdmin Passwort festlegen

Jetzt wird noch nach einem Passwort zum Einloggen in phpMyAdmin gefordert.



Nach der Eingabe und Bestätigung muss Apache mit phpMyAdmin noch verknüpft werden. Dazu bearbeiten wir die Datei `/etc/php5/apache2/php.ini`  
`sudo nano /etc/php/7.3/apache2/php.ini`

Ganz am Ende der Datei fügen wir **extension=mysql.so** ein.

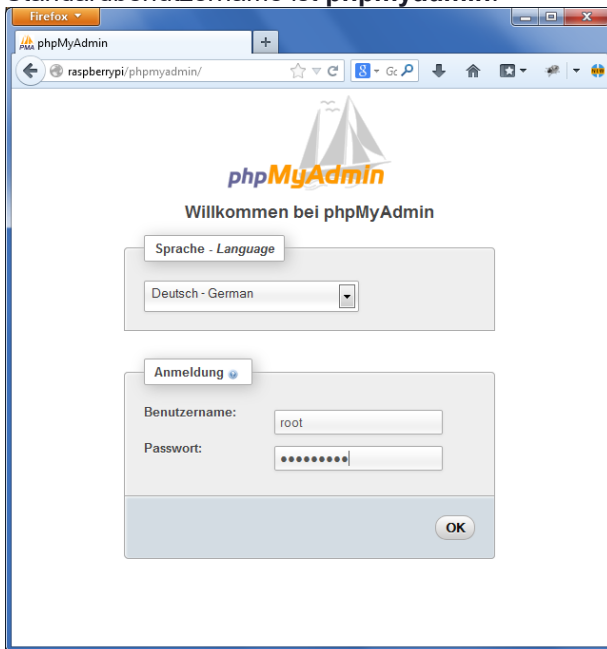
```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/php5/apache2/php.ini Modified
; Directory where to load mcrypt modes
; Default: Compiled in into libmcrypt (usually /usr/local/lib/libmcrypt)
;mcrypt.modes_dir=

[dba]
;dba.default_handler=

; Local Variables:
; tab-width: 4
; End:
extension=mysql.so
```

Dann nur noch mit STRG + O und STRG + X speichern und beenden.

Das war es bereits. Ab jetzt kann man sich unter <http://raspberrypi/phpmyadmin/> (bzw. 192.168.0.xxx) mit den zuvor eingegebenen Passwort für den phpMyAdmin einloggen. Der Standardbenutzername ist **phpmyadmin**.



Fertig. Jetzt können auch über das Interface Datenbanken erstellt und verwaltet werden.

#### Wenn PW oder Benutzername vergessen:

```
sudo cat /etc/phpmyadmin/config-db.php | grep -iA6 dbuser
```

Wenn sonst ein Fehler, kann mit folgendem Befehl Konfiguration neu gestartet werden ([Neuinstallation auswählen und Fehler 1045 «ignore»](#)):

```
sudo dpkg-reconfigure phpmyadmin
```

**Am Linux MySQL Server einen neuen Benutzer mit Root Rechten erstellen, damit man sich am Webinterface von phpmyadmin anmelden kann!**



Letztens ist folgendes Problem mit einem Linux **MySQL Server** sowie phpmyadmin aufgetreten. Dabei war nach der Installation von MySQL und phpmyadmin auf einem Linux Ubuntu Server 18.04 das Anmelden mit dem Root Account am Webinterface von phpmyadmin nicht möglich. Allerdings das Anmelden an der MySQL Konsole direkt schon. Erst das Anlegen eines neuen Benutzers lieferte die Lösung, den dieser hatte anschließend Zugriff auf das Webinterface von phpmyadmin. Da er mit administrativen Rechten erstellt wurde, konnte man damit auch neue Datenbanken erstellen. Dieser Artikel beschreibt, wie man sich am Linux MySQL Server per **Terminal** anmeldet und anschließend einen neuen Benutzer mit administrativen Rechten anlegt.

```
mysql -u root -p
```

Nachdem Anmelden an der MySQL Konsole, kann jetzt der neue Benutzer erstellt werden. Die folgenden vier Befehle liefern dazu genau die Vorgehensweise. Der neue Benutzer Chef erhält dabei volle administrative Berechtigungen. Am Ende sollte man den MySQL Server bzw. den Dienst neu starten.

```
create user 'chef'@'localhost' identified by 'Passw0rt!';
```

```
grant all privileges on *.* to 'chef'@'localhost' with grant option;
```

```
flush privileges;
```

**exit**

```
service mysql restart
```

Mit diesem Benutzer kann man sich jetzt am Webinterface von phpmyadmin anmelden. Da diesem, wie oben ersichtlich, sämtliche Berechtigungen erteilt wurden, können mit ihm natürlich auch neue Datenbanken erzeugt sowie gelöscht werden.

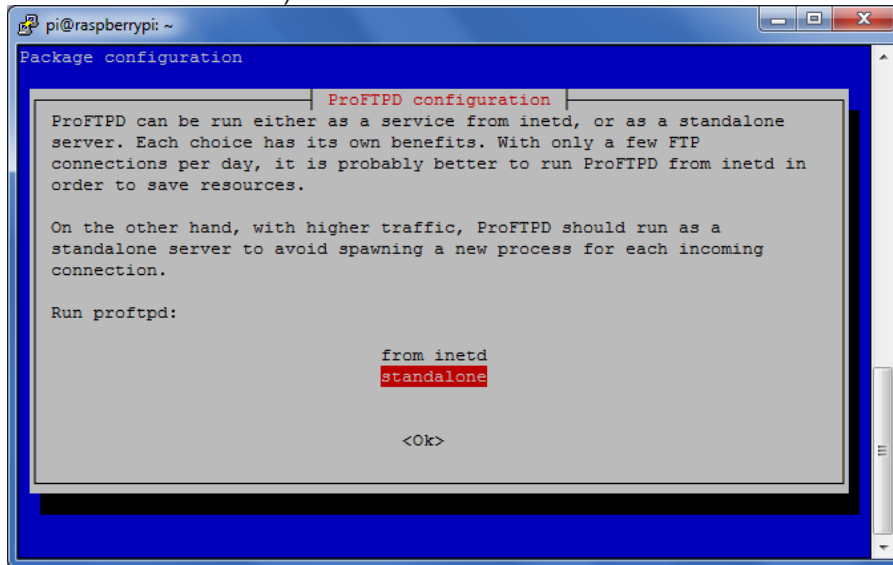
## 6. FTP-Server (proftpd) mit User einrichten

Zum einfachen Upload sämtlicher Dateien sollte auf einem Webserver kein FTP Zugang fehlen. Dazu richten wir in diesem Teil einen solchen FTP Server mit Zugang auf dem Pi ein. Dieses Tutorial kann auch zum einfachen Austauschen von Dateien zwischen PC und Raspberry benutzt werden und ist nicht zwingendermassen für einen Webserver.

Zunächst die Installation

```
sudo apt-get install proftpd
```

Kurz danach wird man nach der Startvariante gefragt. Hier wählen wir **standalone** (auf Deutsch **Servermodus**)



Im Grunde wären wir hier bereits fertig, allerdings hätte jetzt jeder Nutzer nur Zugriff auf sein eigenes Home Verzeichnis (z.B. /home/pi ). Deshalb erstellen wir einen neuen User. Dazu wechseln wir erst einmal das Verzeichnis.

```
cd /etc/proftpd/
```

Hier soll der virtuelle Benutzer erstellt werden. Im folgenden Beispiel erstelle ich den Benutzer **webserv\_user** mit dem Homeverzeichnis **/var/www/** (falls das Rootverzeichnis von Apache **geändert wurde**, sollte dies natürlich angepasst werden. Gleiches gilt, wenn ein FTP-User auf andere Verzeichnisse Zugriff haben soll).

```
sudo ftpasswd --passwd --name webserv_user --gid 33 --uid 33 --home /var/www/ --shell /bin/false
```

Nun nur noch das Passwort eingeben und bestätigen. Falls das Passwort des Benutzers zu einem späteren Zeitpunkt geändert werden soll, einfach wieder in das Verzeichnis wechseln und den Befehl erneut ausführen.

Um den virtuellen User noch freizuschalten, bearbeiten wir die Konfigurationsdatei

```
sudo nano /etc/proftpd/proftpd.conf
```

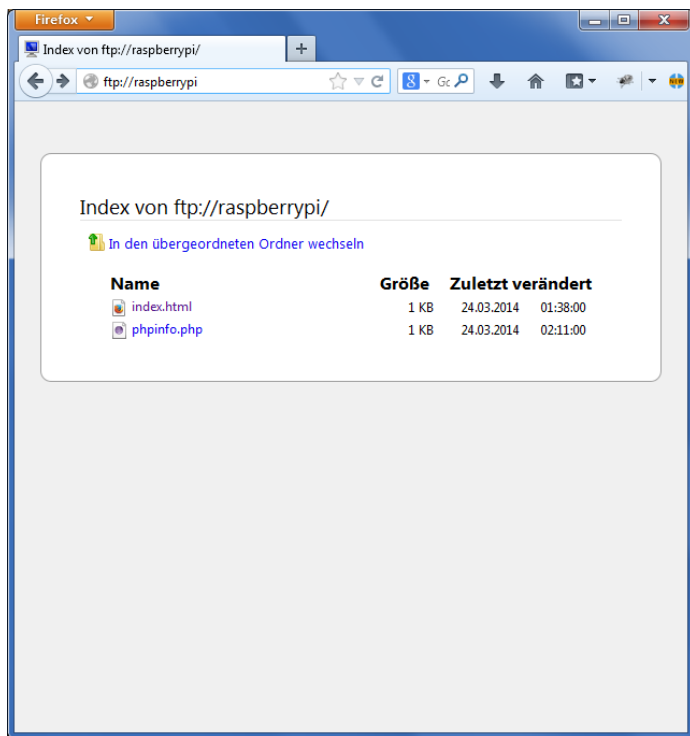
und fügen folgenden Code am Ende ein:

```
DefaultRoot ~  
AuthOrder mod_auth_file.c mod_auth_unix.c  
AuthUserFile /etc/proftpd/ftpd.passwd  
AuthPAM off  
RequireValidShell off
```

Mittels STRG + O und STRG + X speichern und beenden wir den Editor.

Zu guter Letzt starten wir den Server neu und probieren es aus.

```
sudo /etc/init.d/proftpd restart
```



Möglicherweise kann man sich nun trotzdem noch nicht einloggen, da entsprechende Rechte fehlen, diese setzen wir mit

```
chmod g+s /var/www/html
chmod 775 /var/www/html
```

(falls ein anderes Verzeichnis dem User zugewiesen wurde, muss es natürlich entsprechend angepasst werden)

Bei den Rechten kommt zuerst der Besitzer, dann die Gruppe, dann der User mit r=read, w=write, x=ausführen.

Mögliche Werte für:				
	chmod (octal)	umask (octal)	Symbolisch	Binäre I
Lesen, schreiben und ausführen	7	0	rwx	111
Lesen und Schreiben	6	1	rw-	110
Lesen und Ausführen	5	2	r-x	101
Nur lesen	4	3	r--	100
Schreiben und Ausführen	3	4	-wx	011
Nur Schreiben	2	5	-w-	010
Nur Ausführen	1	6	--x	001
Keine Rechte	0	7	---	000

Eventuell muss auch Firewall freigeschaltet werden:

```
sudo ufw allow 20
sudo ufw allow 21
```

Ab hier würde der FTP funktionieren. Wir wollen aber noch eine TLS-Verschlüsselung, damit die Dateien nicht im Klartext übertragen werden.

### SSL/TLS-verschlüsselte FTP-Verbindung mit mod\_tls

Das TLS Modul ermöglicht eine verschlüsselte Verbindung über SSL/TLS zum ProFTPD Server.

Achtung: Ohne Verschlüsselung überträgt das FTP-Protokoll sowohl Login- als auch normale Daten im Klartext! Der Einsatz von SSL/TLS wird für Produktivumgebungen daher dringend empfohlen.

Standardmässig unterstützt ProFTPD das TLS-Modul:

```
$ sudo proftpd -vv | grep tls
```

Es ist in `/etc/proftpd/modules.conf` bereits enthalten und automatisch aktiv.

### Zertifikat erstellen

Das folgende Beispiel verwendet als Zertifikat das selbst-signierte Snakeoil Zertifikat des `ssl-cert` Packages (siehe [Ubuntu default snakeoil SSL-Zertifikat erneuern](#)):

```
$ sudo apt install ssl-cert
$ sudo make-ssl-cert generate-default-snakeoil --force-overwrite
$ sudo ls -la /etc/ssl/certs/ssl-cert-snakeoil.pem
$ sudo ls -la /etc/ssl/private/ssl-cert-snakeoil.key
```

### TLS konfigurieren

Das in den Paketquellen von Debian Stretch vorhandene ProFTPD Paket in der Version 1.3.5b-4 unterstützt auch TLSv1.2.<sup>[1]</sup>

Im `conf.d` Verzeichnis wird wiederum eine eigene Konfigurationsdatei für SSL/TLS erstellt:

```
$ sudo vi /etc/proftpd/conf.d/tls.conf
<IfModule mod_tls.c>
    TLSEngine on
    TLSLog /var/log/proftpd/tls.log
    TLSProtocol TLSv1.2
    TLSRSACertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
    TLSRSACertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
    TLSVerifyClient off
    TLSRequired on
</IfModule>
```

Damit die Verzeichnisse mit TLS angezeigt werden muss in `/etc/proftpd/proftpd.conf` noch folgendes eingefügt werden (am Ende z.B.):

```
TLSOptions NoSessionReuseRequired
```

Anschliessend wird ProFTPD neu gestartet.

## Analyse bei Verbindungsproblemen

Bei Problemen beim Aufbau der FTP Verbindungen können folgende Dinge überprüft werden:

1. ProFTPD Dienst läuft: `$ sudo service proftpd status`
2. ProFTPD lauscht auf Port 21: `$ sudo netstat -tlnp|grep proftpd`
3. Fehlermeldungen im ProFTPD Log: `$ sudo tail -20 /var/log/proftpd/proftpd.log`
4. Fehlermeldungen im ProFTPD TLS Log: `$ sudo tail -20 /var/log/proftpd/tls.log`
5. Verbindungstest auf Port 21 mit telnet: `$ telnet 192.0.2.10 21`
6. Verbindungstest auf Port 21 mit TLS: `$ openssl s_client -connect 192.0.2.10:21 -starttls ftp`

## Probleme mit der Verbindung von extern (PassiveMode)

With the *passive* mode, most of the configuration burden is on the server side. The server administrator should setup the server as described below.

The firewall and NAT on the FTP server side have to be configured not only to allow/route the incoming connections on FTP port 21,<sup>2</sup> but also a range of ports for the incoming data connections. Typically, the FTP server software has a configuration option to setup a range of the ports, the server will use. And the same range has to be opened/routed on the firewall/NAT.

When the FTP server is behind a NAT, it needs to know it's external IP address, so it can provide it to the client in a response to PASV command.

Lösung:

```
sudo nano /etc/proftpd/proftpd.conf
```

Es müssen zwei Anpassungen vorgenommen werden. Zum einen muss `MasqueradeAddress` mit der aktuellen, öffentlichen IP eingefügt werden. Damit kann der FTP-Server im Passive-Modus dem Client die korrekte IP (nicht die vom internen Netz) weitergeben:

```
MasqueradeAddress 83.78.64.23  
MasqueradeAddress blubb.ddns.net
```

Ausserdem braucht der PassiveMode die ganze Port-Range >1024. Dies kann man jedoch nicht alles freigeben, da sonst gefährliche Ports offen waren. Deswegen kann man mit den `PassivePorts` die zu benutzenden Ports freigeben. Diese müssen dann aber auch im Router freigegeben und weitergeleitet werden:

```
PassivePorts 60000 60050
```

## 7. Zugriff Aussen mit Portforwarding (DynDNS mit No-IP)

In Teil 6 der Webserver Installation geht es um darum den Server voch ausserhalb des lokales Netzwerkes mittels eines DNS Servers (wie No-IP, DynDNS) erreichbar zu machen.

Mit den bisherigen Programmen konnten wir einen vollständigen Webserver aufsetzen, doch sollen die Dateien in den allermeisten Fällen auch über das Internet erreichbar sein. Dies wird in diesem Tutorial behandelt.

Als erstes benötigen wir einen DNS Anbieter, wobei ich No-IP bevorzuge (kostenlos). Also legen wir einen kostenlosen Account unter <https://www.noip.com/sign-up> an. Der einzige „Nachteil“ der Free Version ist, dass jeden Monat eine Mail kommt, in der man aufgefordert wird den Account zu bestätigen (mittels Capcha Eingabe).

Nach dem Login klicken wir auf *Add a Host* und wählen einen Hostnamen und als Domain eine der weiter unten gelisteten *No-IP Free Domains*. Als Host Type nehmen wir *DNS Host (A)*.

### Hostname Information

Hostname:	<input type="text" value="tutorials-raspberypi"/>	<input type="text" value="no-ip.org"/>	
Host Type:	<input checked="" type="radio"/> DNS Host (A) <input type="radio"/> DNS Host (Round Robin) <input type="radio"/> DNS Alias (CNAME)		
	<input type="radio"/> Port 80 Redirect <input type="radio"/> Web Redirect <input type="radio"/> AAAA (IPv6)		
IP Address:	<input type="text"/>		
Assign to Group:	<input type="text" value="- No Group -"/>	<a href="#">Configure Groups</a>	
Enable Wildcard:	Wildcards are a Plus / Enhanced feature. <a href="#">Upgrade Now!</a>		

**Accept Mail for your Domain**  
Let No-IP do the dirty work. Setup [POP](#) or [forwarding](#) for your name.

### Mail Options

MX Record	MX Priority	
Enter the name of your external mail exchangers (mx records) as hostnames <b>not</b> IP addresses.		
<input type="text"/>	<input type="text" value="5"/>	
If you would like a more MX records, please upgrade to <a href="#">No-IP Plus</a> or <a href="#">Enhanced</a> .		

Jetzt noch auf *Add Host* klicken und wir können zur Konfiguration auf dem Pi kommen. Das Paket muss von der Seite runter geladen werden

```
sudo wget http://www.noip.com/client/linux/noip-duc-linux.tar.gz
```

und entpackt werden:

```
sudo tar xf noip-duc-linux.tar.gz
```

Zum installieren des Pakets müssen wir erst in den Ordner wechseln

```
cd noip-2.*
```

und compilieren

```
sudo make install
```

Daraufhin wird eine Abfrage nach der Email Adresse kommen, sowie dem verwendeten Passwort. Als Intervall habe ich 30 Sekunden gelassen und auf „Do you wish to run something at successful update?“ mit nein geantwortet.

```
Auto configuration for Linux client of no-ip.com.
Please enter the login/email string for no-ip.com [REDACTED]
Please enter the password for user [REDACTED] *****

Only one host [REDACTED] is registered to this account.
It will be used.
Please enter an update interval:[30] 30
Do you wish to run something at successful update?[N] (y/N) n

New configuration file '/tmp/no-ip2.conf' created.
mv /tmp/no-ip2.conf /usr/local/etc/no-ip2.conf
pi@raspberrypi ~/noip-2.1.9-1 $
```

Den Service startest du mittels  
`sudo noip2`

aber Achtung: Startest du den Pi neu, wird No-IP nicht automatisch gestartet. Wie das geht, ist in [diesem Tutorial](#) beschrieben.

Damit der Service nun auch von aussen auf den Pi weiterleiten kann (beim Aufruf von *deine-domain.no-ip.org* o.ä.) musst du in deinem Router noch die Ports **80**, **81** und **8080** (TCP) freigeben. In meinem Router ist das unter dem Reiter **Port Forwarding** möglich. Als IP Adresse gibst du die interne IP des Raspberry's an. Bei mir ist das 192.168.1.83 (welche es genau bei dir ist, lässt sich auch in deinem Router feststellen).

#### Heimnetz Port Forwarding

**Port Forwarding bearbeiten**

Dienst: Benutzerdef. ▾

Computer: 192 168 1 83

Port(bereich/e): 2 Portbereiche ▾

1. Portbereich: 80 bis 81  
Protokoll: TCP ▾

2. Portbereich: 8080 bis 8080  
Protokoll: TCP ▾

anderen Ziel-Port für Portbereich(e) verwenden  
Ziel-Port:

Port Forwarding aktiv

Solltest du zu einem späteren Zeitpunkt die Konfiguration ändern wollen, findest du sie unter **/usr/local/etc/no-ip2.conf**

Beim Port-Forwarding kann man auch verschiedene Zugriffe definieren. Wenn ich zum Beispiel auf verschiedene Webserver von aussen zugreifen will:

<http://blubb.ddns.net:3333/>

Dazu braucht es einfach ein Forwarding von 3333 zur internen IP mit dem Port 80.

## 8. Autostart: Programm automatisch starten lassen

Immer wieder passiert es, dass man Programme installiert, aber diese nicht automatisch beim hochfahren starten. Um den Raspberry Pi Autostart nutzen zu können, braucht man lediglich die Informationen in der `/etc/rc.local` Datei im Linux System zu hinterlegen. Das ist z.B. bei [DNS Service noip2](#) der Fall. Im folgenden zeige ich wie man ein Programm ohne viel Mühe automatisch starten lässt, anhand vom Beispiel noip2.

### Raspberry Pi Autostart Skript

Als erstes muss im Verzeichnis `/etc/init.d/` ein Skript erstellt werden, mittels welchem das Programm gestartet wird, daher erstellen wir ein Skript (es muss nicht unbedingt eine Datei-Endung haben)

```
sudo nano /etc/init.d/NameDesSkripts
```

mit folgendem Inhalt:

Shell

```
1
2 #!/bin/sh
3 ### BEGIN INIT INFO
4 # Provides: noip
5 # Required-Start: $syslog
6 # Required-Stop: $syslog
7 # Default-Start: 2 3 4 5
8 # Default-Stop: 0 1 6
9 # Short-Description: noip server
10 # Description:
11 ### END INIT INFO
12 case "$1" in
13   start)
14     echo "noip wird gestartet"
15     # Starte Programm
16     /usr/local/bin/noip2
17     ;;
18   stop)
19     echo "noip wird beendet"
20     # Beende Programm
21     killall noip2
22     ;;
23   *)
24     echo "Benutzt: /etc/init.d/noip {start|stop}"
25     exit 1
26   ;;
27 esac
28 exit 0
29
```

Anstelle von noip2 kann hier natürlich auch jeden andere installierte Programm stehen, aber sei vorsichtig, dass auf keine Benutzerinteraktion gewartet wird (wie das Bestätigen bei apt-get), da es im schlimmsten Fall dazu kommt, dass beim booten auf die Eingabe gewartet wird und der Pi nicht startet.

Als nächstes weisen wir die benötigten Rechte zu (Lesen & Schreiben)

```
sudo chmod 755 /etc/init.d/NameDesSkripts
```

und testen das Skript indem wir es starten



```
sudo /etc/init.d/NameDesSkripts start
```

und gleich wieder stoppen:

```
sudo /etc/init.d/NameDesSkripts stop
```

Damit das Skript beim booten auch aufgerufen wird, führen wir folgendes aus:

```
sudo update-rc.d NameDesSkripts defaults
```

Nun sollte das Programm bei starten auch ausgeführt werden. Solltest du eines Tages dich umentscheiden und das Programm aus dem Autostart nehmen wollen, kannst du dies mit:

```
sudo update-rc.d -f NameDesSkripts remove
```

Wer mehr über das Thema erfahren will, kann dies [hier](#) tun.

### **Raspberry Pi Autostart – weitere Möglichkeiten**

Eine andere Option zum Starten eines Skripts oder Programms ist „Cron“. Dadurch ist es möglich einen Befehl (der ein Aufruf eines Programms o.ä. sein kann) zu einem bestimmten Zeitpunkt zu starten. Der Zeitpunkt kann dabei entweder z.B. um die gleiche Uhrzeit am Tag sein oder aber nach dem Hochfahren des Systems. Cron bietet sehr viele Anpassungsmöglichkeiten, welche [hier](#)eingesehen werden können.

## 9. SSL-Zertifikat installieren (Zertifikat mit Letsencrypt)

Um ein SSL / TLS Zertifikat auch sinnvoll nutzen zu können, sollte auf dem [Raspberry Pi 3](#) ein Webserver bzw. eine Anwendung, zu welcher z.B. eine HTTP(S) Verbindung aufgebaut werden soll, laufen. Folgende Anwendungen habe ich bereits in vorherigen Tutorials gezeigt:

- [Raspberry Pi Webserver Installation Teil 1 – Apache2](#)
- [Raspberry Pi Webserver Installation Teil 5 – FTP Server](#)
- [Raspberry Pi Node.js Webserver installieren und GPIOs schalten](#)

Darüber hinaus empfehle ich unbedingt einen dynamischen [DNS Server zu installieren](#), sofern dein Raspberry Pi keine statische IP Adresse hat. Normalerweise vergeben Internet Provider an Haushalte nur gegen Aufzahlung eine statische IP. Eine dynamische IP Adresse ändert sich spätestens einmal am Tag, d.h. es müsste eine andere IP Adresse abgerufen werden.

Mit einem dDNS Service kann dieses Problem umgangen werden, da eine Verbindung zu einem (kostenlosen) Domain aufgebaut wird, sobald sich die dynamische IP Adresse ändert. So kann man diese Domain anstelle der sich ändernden IP Adresse aufgerufen werden und der dDNS Service kümmert sich um alles weitere. Daneben kann deine richtige Domain (sofern du eine hast) auch per DNS Einträge auf diese „Zwischen-Domain“ zeigen.

Falls du einen Apache Server laufen lässt, kannst du diesen wie folgt beenden. Um einen Node.JS Server zu beenden, reicht es die Anwendung zu [killen](#) (ggf. Backgroundprozess beenden).

```
sudo service apache2 stop
```

Stelle ausserdem sicher, dass neben Port 80 auch der Port 443 in deinem Router (Empfehlung: [FRITZ!Box](#)) für die interne IP Adresse des Raspberry Pi's freigegeben ist.

### Let's Encrypt SSL Zertifikat erstellen

Um das Tool zum Erstellen des Zertifikats herunterzuladen, nutzen wir Git. Auf den neueren Raspbian Versionen ist dieses schon mit dabei. Falls es auf deinem Betriebssystem (z.B. einer Minimal Raspbian Version) nicht dabei sein sollte, kannst du es einfach nachinstallieren:

```
sudo apt-get install git
```

Wir laden die Dateien nun in unser Homeverzeichnis und gehen in diesen Ordner:

```
cd ~
```

```
git clone https://github.com/letsencrypt/letsencrypt
```

```
cd letsencrypt
```

Nun brauchst du alle Domains zur Hand, für welche dieses Zertifikat gelten soll. Wenn du bspw. Subdomains hast oder aber deine Domain mit und ohne „www“ aufgerufen wird, kannst du diese gleich alle angeben. In meinem Beispiel möchte ich das SSL Zertifikat lediglich für meine Domain, welche ich von NoIP bekommen habe, erstellen. Bei mehreren Domains solltest du die Hauptdomain als erstes angeben.

Der Befehl zum Erstellen des Let's Encrypt Zertifikats ist folgender (Domains und Mail ersetzen):

```
./letsencrypt-auto -d ERSTE_DOMAIN -d ZWEITE_DOMAIN --redirect -m  
DEINE_MAIL
```

Sollte es hier zu einem Fehler kommen «Certbot has problem setting up the virtual environment», muss folgendes beachtet werden (<https://github.com/certbot/certbot/issues/6933#issuecomment-481668226>):

«editiere die /etc/pip.conf (mit root-rechten) und kommentiere die Zeile „extra-index-url=https://www.piwheels.org/simple“ mit einem # am Zeilenanfang aus. Fertig.»

Danach relativ lange warte, AGBs akzeptieren usw.

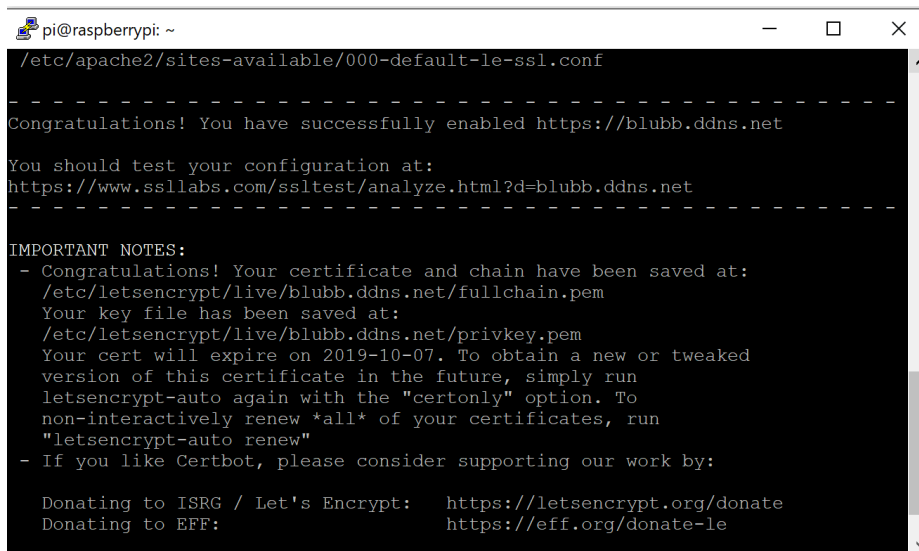
Der `--redirect` Parameter gibt an, dass normale HTTP Verbindungen automatisch zu HTTPS weitergeleitet werden. Die Mail Adresse ist für eine eventuelle Kontaktaufnahme nötig bzw. falls das Zertifikat eines Tages widerhergestellt werden müsste.

Tipp: Solltest du einen Apache Server laufen lassen, kannst du das Tool auch alle weiteren Einstellungen vornehmen lassen, indem du den weiteren Parameter `--apache` hinzufügst.

Du wirst nun aufgefordert die Nutzungsbedingungen zu lesen und zu akzeptieren.

Mit der Option `certonly` kannst du ausserdem angeben, dass lediglich die Zertifikate angelegt werden sollen. In dem Ordner `„/etc/letsencrypt/live/“` befindet sich dann ein neuer Ordner mit dem Namen unserer angegebenen Hauptdomain. Darin befinden sich vier Schlüsseldateien, welche gebraucht werden. Je nach Anwendung, ist der Einbau der SSL Zertifikate etwas anders. In Node.JS muss das Zertifikat z.B. [per Code geladen werden](#). Wie die manuelle Einrichtung für Apache2 funktioniert, [erfährst du hier](#).

Übrigens: Falls du Hilfe brauchst oder alle weiteren Parameters des Tools einsehen willst, kannst du dies einfach mittels `./letsencrypt-auto --help all`



```
pi@raspberrypi: ~  
/etc/apache2/sites-available/000-default-le-ssl.conf  
-----  
Congratulations! You have successfully enabled https://blubb.ddns.net  
  
You should test your configuration at:  
https://www.ssllabs.com/ssltest/analyze.html?d=blubb.ddns.net  
-----  
IMPORTANT NOTES:  
- Congratulations! Your certificate and chain have been saved at:  
  /etc/letsencrypt/live/blubb.ddns.net/fullchain.pem  
  Your key file has been saved at:  
  /etc/letsencrypt/live/blubb.ddns.net/privkey.pem  
  Your cert will expire on 2019-10-07. To obtain a new or tweaked  
  version of this certificate in the future, simply run  
  letsencrypt-auto again with the "certonly" option. To  
  non-interactively renew *all* of your certificates, run  
  "letsencrypt-auto renew"  
- If you like Certbot, please consider supporting our work by:  
  
  Donating to ISRG / Let's Encrypt:  https://letsencrypt.org/donate  
  Donating to EFF:                  https://eff.org/donate-le
```

## SSL Zertifikat erneuern

Alle Zertifikate von Let's Encrypt haben eine Laufzeit von 3 Monaten. Nach dieser Periode sind sie abgelaufen und müssen erneuert werden. Das Erneuern des Raspberry Pi SSL Zertifikats ist allerdings sehr einfach (Anpassen nicht vergessen):

```
./letsencrypt-auto -d ERSTE_DOMAIN -d ZWEITE_DOMAIN --redirect -m  
DEINE_MAIL --agree-tos --renew-by-default
```

Hierbei ist der Parameter `--renew-by-default` das Entscheidende. Die anderen (weiteren) Parameter sind identisch mit denen, die wir beim Erstellen angegeben haben.

Nun ist es allerdings so, dass man im Dauerbetrieb ja nicht unbedingt alle 3 Monate an die Aktualisierung des Zertifikats denken möchte. Daher bietet sich ein kleiner Trick an: Sollte der Raspberry Pi sowieso (fast) immer in Betrieb sein, können wir das Zertifikat auch früher aktualisieren, z.B. jeden Monat. Um dies automatisiert laufen zu lassen, nutzen wir [Cron](#):

```
sudo crontab -e
```

Ans Ende dieser Datei fügen wir folgende Zeile hinzu (angepasst wie oben):

Shell

```
0 1 2 * * /home/pi/letsencrypt/letsencrypt-auto -d ERSTE_DOMAIN -d ZWEITE_DOMAIN --redirect -m  
1 DEINE_MAIL --agree-tos --renew-by-default  
2  
3
```

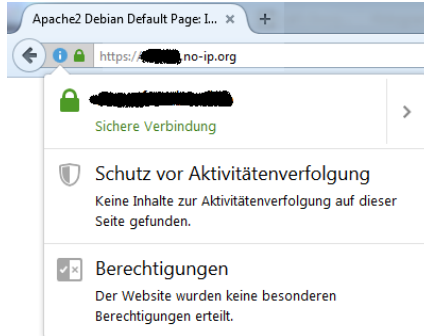
Damit wird am 1. jeden Monats um 02:00 nachts das Tool gestartet und unser Raspberry Pi SSL Zertifikat erneuert.

## Raspberry Pi SSL Zertifikat testen

Sofern der dynamische DNS Service läuft und dein Webserver wieder gestartet ist (`sudo /etc/init.d/apache2 start`) kannst du nun im Browser testen, ob dein Zertifikat erkannt wurde. Dazu rufst du einfach die Domain mit `https://` am Anfang auf. Speziell bei `apache2` ist es wichtig, dass die Datei `/etc/apache2/ports.conf` lediglich einen Eintrag hat, der Port 443 verwendet (alle anderen auskommentieren):

```
Listen 80
<IfModule mod_ssl.c>
    Listen 443
</IfModule>
```

Im Browser sollte nun das Zertifikat angezeigt werden:



Viele weitere Details finden sich ausserdem in der offiziellen [Let's Encrypt Dokumentation](#).

## 10. Webserver-Verzeichnis mit Passwort schützen

### Erstellen einer .passwd Datei

Damit man mit dem Apache Webserver Verzeichnisse und Dateien mit einem Passwort schützen kann, braucht man zunächst eine Datei, in welcher die Passwortdaten enthalten sind. Dazu erstellt man sich die Datei am besten nicht im Document-Root des Webservers (z.B. /var/www; hier könnte die Datei über das Internet ausgelesen werden), sondern im /root Verzeichnis.

```
vps140:~# htpasswd -cs .passwd testuser
New password:
Re-type new password:
Adding password for user testuser
vps140:~#
```

Der Schalter -c bewirkt dabei, dass eine neue Datei angelegt wird, der Schalter -s erzwingt eine SHA Verschlüsselung des Passwortes.

Man kann sich die Datei z. B. mit dem Befehl cat ansehen:

```
vps140:~# cat .passwd
testuser:{SHA}RcVxoVbdzvQTUacTvN3uW6fpVGA=
```

### Erstellen der .htaccess Datei

Um Verzeichnisse und Dateien im Apache Webserver mit einem Passwortschutz zu versehen kann man sich im entsprechenden Verzeichnis eine .htaccess Datei erstellen (z. B. mit dem Editor nano), welche dann einen Passwortschutz ermöglicht. Im Folgenden wird davon ausgegangen, dass das im Document-Root des Webservers befindliche Verzeichnis websvn mit einem Passwort geschützt werden soll.

```
vps140:/var/www/websvn# nano .htaccess
```

Die Datei sieht wie folgt aus:

```
AuthType Basic
AuthUserFile /root/.passwd
AuthName "websvn"
order deny,allow
allow from all
require valid-user
```

Die Zeile AuthUserFile gibt dabei an, wo sich die .passwd Datei befindet, anhand der sich Benutzer bei der Anmeldung authentifizieren können. Die Zeile require valid-user ermöglicht eine Angabe, wer alles Zugriff auf die Verzeichnisse und Dateien erhalten soll. Dabei gibt man mit valid-user an, dass alle in der .passwd Datei angelegten Benutzer Zugriff auf die Verzeichnisse und Dateien erhalten. (Sollen nur bestimmte Benutzer Zugriff erhalten kann man dies hier angeben, z. B. mit require testuser)

### Anpassen der Datei mit dem VirtualHost

Damit die Einstellungen auch wirksam werden, muss noch die Option AllowOverride von None auf All in der Datei /etc/apache2/apache2.conf eingefügt werden. Wenn nun also das Verzeichnis /var/www/html/geheim/ geschützt werden soll, muss im apache2.conf folgendes eingefügt werden:

```
vps140:/etc/apache2# nano apache2.conf

#Oli 2019-07-12 fuer geschuetztes Verzeichnis
```

```
<Directory /var/www/html/geheim/>
    AllowOverride All
</Directory>
```

Am besten dort einfügen, wo schon die anderen Directory-Direktiven sind.

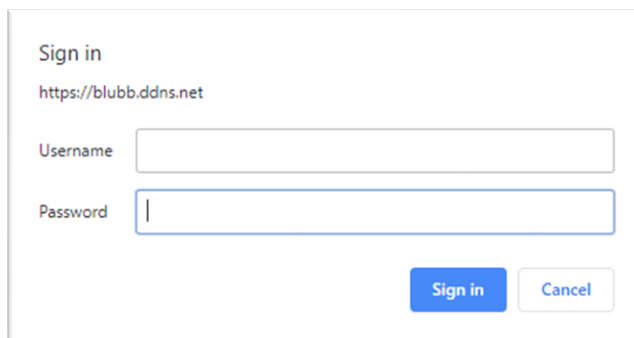
## Übernehmen der Einstellungen

Damit die Einstellung vom Webserver übernommen werden, muss dieser abschliessend nur noch neu gestartet werden.

```
vps140:~# /etc/init.d/apache2 restart
Forcing reload of web server (apache2)... waiting .
vps140:~#
```

## Ergebnis

Will man nun Zugriff auf die Website erscheint ein Fenster zur Authentifizierung.



Sign in  
https://blubb.ddns.net

Username

Password

## Fehlerhafte Logins in Log anzeigen

Man kann in den Logs nachschauen, wer sich versucht hat einzuloggen:

```
vps140:/var/log/apache2# nano access.log
83.78.64.23 - - [12/Jul/2019:09:21:17 +0100] "GET /geheim HTTP/1.1" 401 756 "-" "Mozilla/5.0 (Windows NT 10.0; Win6
83.78.64.23 - hoi [12/Jul/2019:09:21:22 +0100] "GET /geheim HTTP/1.1" 401 1316 "-" "Mozilla/5.0 (Windows NT 10.0; W
83.78.64.23 - webserv_user [12/Jul/2019:09:21:25 +0100] "GET /geheim HTTP/1.1" 401 756 "-" "Mozilla/5.0 (Windows NT
83.78.64.23 - webserv_user [12/Jul/2019:09:21:26 +0100] "GET /geheim HTTP/1.1" 301 611 "-" "Mozilla/5.0 (Windows NT
83.78.64.23 - webserv_user [12/Jul/2019:09:21:26 +0100] "GET /geheim/ HTTP/1.1" 200 2787 "-" "Mozilla/5.0 (Windows N
83.78.64.23 - [12/Jul/2019:09:21:26 +0100] "GET / HTTP/1.1" 401 1316 "-" "Mozilla/5.0 (Windows NT 10.0; Win6
```

```
vps140:/var/log/apache2# nano error.log
[Fri Jul 12 09:21:22.522020 2019] [auth_basic:error] [pid 5400] [client 83.78.64.23:64257] AH01618: user hoi not found: /geheim
[Fri Jul 12 09:21:25.727442 2019] [auth_basic:error] [pid 5400] [client 83.78.64.23:64257] AH01617: user webserv_user: authentication failure for "/geheim": Password Mismatch
```

Die erste Zeile war ein nicht vorhandener User «hoi». Die zweite Zeile war zwar korrekter User, aber falsches Passwort «Password Mismatch».

## 11. Mails mit ext. Host (GMAIL) über Webserver versenden

Leider ist das Versenden von E-Mails vom eigenen Server aus in Zeiten von Spam nicht einfach. Möchte man einen eigenen SMTP-Server betreiben, müsste man recht viele Anti-Spam-Massnahmen implementieren um zu gewährleisten, dass eure Emails auch beim Empfänger ankommen. Dieser Aufwand ist aber oftmals viel zu hoch.

Daher empfehle ich externe SMTP-Server für den E-Mail-Versand zu nutzen. Die Einrichtung ist einfach und, sofern ihr einen vertrauenswürdigen Anbieter wie z.B. Gmail nutzt, habt ihr keine Probleme mit Spam.

In diesem Tutorial erkläre ich, wie ihr Ubuntu 18.04 (Bionic Beaver) so konfiguriert, dass vom Server aus E-Mails über einen externen SMTP-Server (z.B. von Gmail) versendet werden kann. Bei anderen Ubuntu- und Linux-Versionen verläuft die Konfiguration analog.

Anschliessend zeige noch, wie ihr PHP konfigurieren könnt, damit E-Mails die per mail()-Befehl gesendet werden über den SMTP-Server versendet werden kann.

### Schritt 1 – Externer SMTP-Server (z.B. Gmail)

Es empfiehlt sich E-Mails über externe (vertrauenswürdige) SMTP-Server zu senden, da ansonsten die E-Mails von eurem Server oft im Spamfilter landen. Bei den meisten Webhosting-Tarifen werden Postfächer mit angeboten und ihr könnt den SMTP-Server von eurem Postfach nutzen.

Damit euer Server die E-Mails über diesen SMTP-Server versendet, braucht ihr folgende Infos:

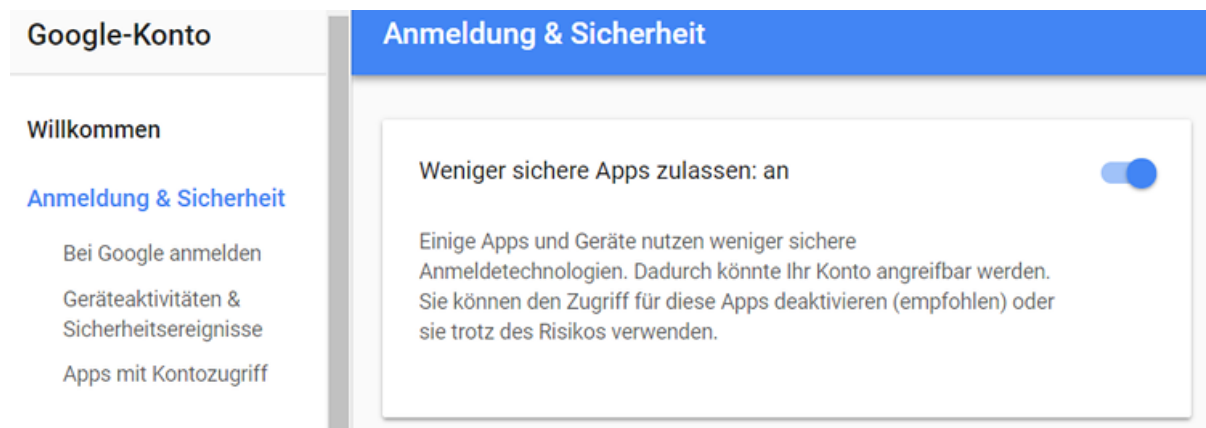
- Serveradresse (des SMTP-Servers)
- Benutzername
- Passwort

Diese Infos bekommt ihr i.d.R. vom Anbieter des E-Mail-Postfaches.

Alternativ kann man auch kostenlos ein Konto bei [Gmail](#) registrieren.

### GMail – Weniger sichere Apps zulassen

Der Versand per GMAIL klappt nur, wenn ihr in eurem Konto die Einstellung *Weniger sichere Apps zulassen* aktiviert. Aktiviert man diese Einstellung, dann können externe Anwendungen (sprich, euer Server) per Benutzername und Passwort über euer GMail-Konto E-Mail versenden.



Wenn ihr diese Einstellung nicht aktiviert, dann erhaltet die Fehlermeldung:

```
1 msmtplib: authentication failed (method LOGIN)...
2 msmtplib: server message: 534-5.7.14 your web browser and then try again.
3 msmtplib: server message: 534-5.7.14 Learn more at
```

```
4 msmtplib: server message: 534 5.7.14 https://support.google.com/mail/answer/78754 z4-v6sm17243464wrt.13 -
5 gsmtplib
6 msmtplib: could not send mail (account default from /etc/msmtplib)
```

Wie ihr die Einstellung weniger sichere Apps zulassen bearbeiten könnt ist hier beschrieben: [Weniger sicheren Apps den Zugriff auf Ihr Konto gestatten](#)

## Schritt 2 – MSMTPLIB installieren

Wir nutzen das msmtplib Package, welches es erlaubt das unter Ubuntu z.B. per Kommandozeile oder per PHP-Anwendung E-Mails über einen externen SMTP-Server versendet werden kann. Die Installation erfolgt wie folgt:

```
1 sudo apt update
2 sudo apt install -y msmtplib msmtplib-mta
```

## Schritt 3 – MSMTPLIB konfigurieren

Mit `msmtplib --version` kann herausgefunden werden, wo die Config-Files gespeichert sind. Beide Dateien sind nach einer neuen Installation nicht vorhanden (so dass man diese einfach nur auf die eigenen Bedürfnisse abändern könnte), allerdings gibt es auf der Website von msmtplib eine Beispiel-Konfiguration. Um MSMTPLIB zu konfigurieren, muss die Datei `/etc/msmtplib` bearbeitet werden:

```
1 sudo nano /etc/msmtplib
```

Diese befüllen wir mit folgenden Informationen (GMail als Beispiel):

```
# Set default values for all following accounts.
defaults

# Use the mail submission port 587 instead of the SMTP port 25.
port 587

# Always use TLS.
tls on

# Set a list of trusted CAs for TLS. The default is to use system settings, but
# you can select your own file.
tls_trust_file /etc/ssl/certs/ca-certificates.crt

# If you select your own file, you should also use the tls_crl_file command to
# check for revoked certificates, but unfortunately getting revocation lists and
# keeping them up to date is not straightforward.
#tls_crl_file ~/.tls-crls

# Mail account
# TODO: Use your own mail address
account bob@meindedomain.de

# Host name of the SMTP server
# TODO: Use the host of your own mail account
host smtp.meindedomain.de

# As an alternative to tls_trust_file/tls_crl_file, you can use tls_fingerprint
# to pin a single certificate. You have to update the fingerprint when the
# server certificate changes, but an attacker cannot trick you into accepting
# a fraudulent certificate. Get the fingerprint with
# $ msmtplib --serverinfo --tls --tls-certcheck=off --host=smtp.freemail.example
#tls_fingerprint 00:11:22:33:44:55:66:77:88:99:AA:BB:CC:DD:EE:FF:00:11:22:33
```



```

# Envelope-from address
# TODO: Use your own mail address
from bob@meindedomain.de

# Authentication. The password is given using one of five methods, see below.
auth on

# TODO: Use your own user name fpr the mail account
user bob@meindedomain.de

# Password method 1: Add the password to the system keyring, and let msmtplib get
# it automatically. To set the keyring password using Gnome's libsecret:
# $ secret-tool store --label=msmtplib \
# host smtp.freemail.example \
# service smtp \
# user joe.smith

# Password method 2: Store the password in an encrypted file, and tell msmtplib
# which command to use to decrypt it. This is usually used with GnuPG, as in
# this example. Usually gpg-agent will ask once for the decryption password.
#passwordeval gpg2 --no-tty -q -d ~/.msmtplib-password.gpg

# Password method 3: Store the password directly in this file. Usually it is not
# a good idea to store passwords in plain text files. If you do it anyway, at
# least make sure that this file can only be read by yourself.
# TODO: Use the password of your own mail account
password pAssW0Rd123

# Password method 4: Store the password in ~/.netrc. This method is probably not
# relevant anymore.

# Password method 5: Do not specify a password. Msmtplib will then prompt you for
# it. This means you need to be able to type into a terminal when msmtplib runs.

# Set a default account
# TODO: Use your own mail address
account default: bob@meindedomain.de

# Map local users to mail addresses (for crontab)
aliases /etc/aliases

```

Dabei muss *user* euer SMTP-Benutzerkonto sein, *from* die Absendeadresse und *password* euer SMTP-Passwort. Noch ein Hinweis auf die Authentifizierungsmethode: In diesem Beispiel nutzen wir die einfachste Form der Authentifizierung, indem das Passwort des Mail-Accounts direkt in der Konfiguration gespeichert wird. `msmtplib` unterstützt hier allerdings auch weitere Authentifizierungsmethoden. Mehr Informationen dazu findet man in der Dokumentation zu `msmtplib`.

Als nächstes sorgen wir dafür, dass nicht jeder Zugriff auf die Datei hat (besonders wichtig, wenn das Passwort direkt in der Konfiguration gespeichert ist):

```
chmod 600 /etc/msmtplibrc
```

Zum Schluss wird noch ein Alias angelegt, so dass die **Empfänger-Adresse** des Root-Accounts (oder des Accounts, mit dem später E-Mail versendet werden sollen) bekannt ist. Dies wird in der Datei angegeben, die auch schon ganz am Ende der Konfiguration von `msmtplib` aufgeführt wurde:

```
nano /etc/aliases
```

Hier wird nun die Empfänger-Adresse des Root-Accounts angegeben. An diese Adresse werden nun E-Mails verschickt, wenn z.B. ein Cronjob fehlschlagen sollte. Daneben wird noch eine

allgemeine „Fallback-Empfänger-Adresse“ angegeben, falls System-Meldungen nicht im Kontext des Root-Accounts auftreten:

```
root: admin@meinedomain.de
default: admin@meinedomain.de
```

#### Schritt 4 – E-Mail Versand testen

Mit folgendem Befehl könnt ihr die richtige Konfiguration von `msmtp` testen. Sofern alles funktioniert hat, sollte `test@email.de` eine E-Mail erhalten:

```
1 echo "Test Nachricht von meinem Server" | msmtp test@email.de
```

#### Schritt 5 – PHP für den E-Mail-Versand konfigurieren

MSMTP lässt sich verwenden, damit PHP-Anwendungen wie z.B. WordPress mittels dem PHP-Mail-Befehl E-Mails versenden kann. Hierzu müsst ihr die `php.ini`-Datei anpassen.

Den Pfad der `php.ini`-Datei lässt sich aus der [phpinfo.php](#) Datei entnehmen.

```
1 sudo nano /etc/php/7.3/apache2/php.ini
```

Sucht in dieser Datei nach `sendmail_path` und passt diesen Abschnitt wie folgt an:

```
1 [mail function]
2 ; For Win32 only.
3 ;SMTP = localhost #Diese Zeile auskommentieren
4 ;smtp_port = 25 #Diese Zeile auskommentieren
5
6 ; For Win32 only.
7 ;sendmail_from = me@example.com
8
9 ; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
10 sendmail_path = "/usr/bin/msmtp -t" #Diese Zeile wie folgt setzen
```

Kommentiert die Einstellungen `SMTP` und `smtp_port` aus, indem ihr ein `;` an den Satzanfang stellt.

Danach müsst ihr `sendmail_path` auf `/usr/bin/msmtp -t` setzen.

```
1 sudo systemctl restart apache2
```

Kontrolliert anschließend in der [phpinfo.php-Datei](#) ob die Einstellung `sendmail_path` korrekt auf `/usr/bin/msmtp -t` verweist:

<code>sendmail_from</code>	<code>no value</code>	<code>no value</code>
<code>sendmail_path</code>	<code>/usr/bin/msmtp -t</code>	<code>/usr/bin/msmtp -t</code>